# Universidad Nacional Autónoma de México

*FACULTAD DE INGENIERIA*

## CUANTIZACION VECTORIAL OPTIMA ESTRUCTURADA EN ARBOL PARA RECONSTRUCCION PROGRESIVA DE SEÑALES CON DECODIFICACION DE MEMORIA REDUCIDA.

*TESIS PROFESIONAL*

Para obtener el Titulo de INGENIERO EN COMPUTACION

por

Gabriel Enrique Robles De La Torre

México, D. F. 1992


**UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

INDICE

# I. INTRODUCCION.

La codificación de señales tiene,y tendrá en el futuro inmediato, un rango de aplicaciones cada vez más importante y amplio. Si atendemos solamente a la codificación vista como un proceso de compresión de datos, una multitud de situaciones nos viene a la mente : el problema de la televisión de alta definición realizada por medios digitales; la gestión de bases de datos que incorporan imágenes; el uso eficiente de recursos de transmisión de datos y, en general, el empleo a fondo de las capacidades de canales de comunicación y de medios de almacenamiento masivo, además de ofrecer la posibilidad de proporcionar el sustrato para simplificar los algoritmos de procesamiento que se apliquen a tales datos.

Es en este contexto que la técnica conocida como cuantización vectorial (VQ) se presenta como una alternativa poderosa a métodos convencionales de compresión de datos. Sin embargo, junto con su inherente ventaja sobre otras técnicas, la cuantización vectorial conlleva a diversos problemas en su implementación, relacionados principalmente con la complejidad tanto del proceso de puesta a punto del cuantizador como de la etapa de compresión. Un procedimiento de compresión de datos por cuantización vectorial exige un gran poder de cómputo en ambas etapas. Además, si se usan variaciones de la técnica básica de VQ para reducir la complejidad en la etapa de compresión, se paga un precio en la calidad de los resultados. Si ahora en adición a lo anterior se desean explotar algunas características interesantes de estas variaciones, se deberá disponer del doble de memoria de trabajo que la

requerida en la técnica básica para lograr la compresión/decompresión.

El propósito de esta tesis es proponer una solución al problema de complejidad de compresión, sin perjudicar la calidad del proceso y sin doblar la memoria requerida para hacer uso de una muy interesante característica posible en un esquema de VQ : la reconstrucción progresiva de señales. Para ello, procedamos a revisar la teoría de construcción de cuantizadores vectoriales junto con un resultado del campo de reconocimiento de patrones que nos fué útil en la construcción del cuantizador vectorial propuesto : el teorema de convergencia del perceptrón, usado aquí como un medio de minimización de funciones. Los algoritmos propuestos se ejemplifican con la compresión y reconstrucción progresiva de imágenes.

## II. Compresión de datos y Cuantización Vectorial.

Podemos representar a cualquier conjunto finito de entidades por medio de un conjunto también finito de bits, ó unidades básicas de información. Cada bit representa dos eventos. Si decimos que un conjunto de entidades I requiere originalmente N bits para representarse, y disponemos de algún medio para usar M < N bits para obtener un conjunto I' equivalente en algún sentido a I, entonces hemos realizado una compresión de la infomación original. El resultado es que si disponemos de un número de bits limitado -y ese es el caso por lo general- para usarse en la representación de entidades, podemos usarlos de una mejor manera. Pensemos en sistemas de memoria. Si para almacenar cierta información uso 100 kilobytes y tengo un algoritmo de compresión de información aplicable que obtiene un ahorro del 50 %, entonces sólo necesitaré 50 kilobytes. O bién, supongamos que tenemos un medio de comunicación que nos permite envier datos a tasas de 10 megabits por segundo. Con un algoritmo equivalente al anterior podríamos enviar información mediante el mismo canal a tasas de 20 megabits por segundo. La compresión de datos es obviamente una necesidad en ambientes de procesamiento de información que empleen volúmenes masivos de ella, por ejemplo, en aplicaciones que usen imágenes ó voz.

Usualmente, las técnicas de compresión se componen de dos etapas : la de compresión propiamente dicha,y la de decompresión, ó de codificación y decodificación, respectivamente. La información comprimida se usa para fines de almacenamiento ó transmisión, y la decomprimida para las aplicaciones que vengan al caso, según el uso. Por ejemplo, podemos

comprimir una imágen de satélite, almacenarla así y decomprimirla para desplegarla en una pantalla.

Podemos dividir los métodos de compresión de datos en dos categorías: algoritmos de compresión sin pérdidas y algoritmos de compresión con pérdidas. En el primer caso, el método de compresión y decompresión conserva exactamente la misma información original. Nada se pierde en el proceso. En el segundo caso, la técnica de compresión/decompresión origina un error en la información decomprimida. Esta última es semejante a la original no comprimida, pero no igual. La técnica que discutiremos a continuación, la cuantización vectorial, pertenece a esta categoría. Aunque se pierde algo de la información al comprimirla, la aproximación obtenida al decodificar es estadísticamente semejante, lo que es aceptable en muchas aplicaciones. Aclaremos ahora el concepto de cuantización y extendámoslo a cuantización vectorial.

En el proceso de digitalización de una señal encontramos una fase en la que esta es comparada contra los elementos de un conjunto discreto de entidades y aproximada a alguno de ellos; esto se conoce como cuantización. La compresión en esta etapa de cuantización es obtenida por el hecho de que si bién la señal de entrada -por lo general analógica- requiere un número potencialmente infinito de bits para ser representada, los valores discretos usados para cuantizar requerirán un número finito de ellos. En el caso que nos interesa aquí -el de compresión de imágenes- por lo general se trabaja con señales que han sido ya cuantizadas durante su etapa de captación. Así cada pixel (elemento de imágen) se representa con un número de bits determinado: 8, 24 o en el

caso de imágenes obtenidas por los satélites Landsat, por 56 bits. Se logrará entonces una compresión en estos casos si la imágen codificada requiere menos bits por pixel para ser representada que los empleados en la imágen original.

Volviendo al caso general, si el conjunto de entidades contra los cuales se compara una muestra de la señal de entrada para su cuantización se compone solamente de escalares (v.gr. números enteros) entonces el proceso se conoce como cuantización escalar. En el caso de que estas entidades sean vectores y la comparación se realice vector de cuantización vs. vector de muestras de la señal de entrada se hablará entonces de cuantización vectorial ( en inglés Vector Quantization, o de modo abreviado VQ).

De acuerdo a [1] examinemos ahora algo de notación y teoría de VQ conservando algunas palabras en inglés que consideramos más adecuadas para el caso.

## II.1 Cuantización Vectorial.

Un cuantizador vectorial Q de dimensión k y tamaño N es un mapeo del espacio euclidiano k dimensional $\mathbb{R}^k$ en un conjunto finito C conteniendo N vectores de salida llamados codevectors o codewords. Así :

$$Q : \mathbb{R}^k \rightarrow C$$

Donde $C = ( y_1, y_2, \ldots, y_n )$ y $y_i \in \mathbb{R}^k$ para cada $i \in J = \{1,2,\ldots,N\}$.

El conjunto C es llamado code book y tiene tamaño N, significando esto que tiene N elementos distintos, siendo cada uno un vector en $\mathbb{R}^k$. La resolución de un cuantizador vectorial es $r = (\log_2 N)/k$ [1], lo cual mide el número de bits por componente usados para representar el vector de entrada. Si se necesitan M bits para representar la información original, entonces la tasa de compresión T obtenida con un cuantizador vectorial de resolución r se calcula como

$$T = M/r$$

Asociada con cada cuantizador de N puntos tenemos una partición de $\mathbb{R}^k$ en N regiones o células, $R_i$ para $i \in J$. La i-ésima región es definida por

$$R_i = \{ x \in \mathbb{R}^k : Q(x) = y_i \} \qquad (1)$$

algunas veces llamada la imágen inversa o pre-imágen de $y_i$ bajo el mapeo Q y puede ser denotado más concisamente por $R_i = Q^{-1}(y_i)$ .

Un cuantizador vectorial puede ser diferenciado en dos componentes, a saber : el codificador y el decodificador vectoriales. El codificador (encoder) es el mapeo de $\mathbb{R}^k$ al conjunto de índices J, y el decodificador (decoder) mapea J al conjunto de reproducciones C. De este modo :

$$E : \mathbb{R}^k \to J \quad y \quad D : J \to C \qquad (2)$$

La operación total de VQ puede describirse como la composición de dos operaciones :

$$Q(x) = D \circ E(x) = D(E(x)) \qquad (3)$$

Para problemas de almacenamiento o transmisión de información, podemos ver que es necesario trabajar solamente con los índices del conjunto J. El decodificador toma estos índices y produce el vector decodificado.

## II.2 MEDICION DEL COMPORTAMIENTO DE UN CUANTIZADOR VECTORIAL.

Una medida de distorsión d es una asignación de un costo no negativo $d(x,\hat{x})$ resultante de reproducir a un vector x mediante su aproximación $\hat{x}$. Podemos medir el desempeño de un sistema mediante una distorsión promedio $D = E(d(X,\hat{X}))$ entre la entrada y la reproducción final. Generalmente, el comportamiento de un sistema de compresión será bueno si la distorsión promedio es pequeña.

En la práctica, la medida general de comportamiento es la media muestral a largo plazo

$$\bar{d} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} d(X_i,\hat{X}_i) \qquad (4)$$

Donde $\{X_v\}$ es una secuencia de vectores a ser codificados. Si el proceso vectorial es estacionario y ergódico, entonces con probabilidad 1 este límite existe y es igual al valor esperado, esto es, $\bar{d} = D$.

La medida de distorsión puede ser escogida de muchas maneras. Puede

o no depender del vector de entrada, o puede o no tener en cuenta a ciertos componentes de este vector. Debido a que la medida de distorsión define la geometría de la partición k dimensional, hemos usado en este estudio a la distancia euclidiana al cuadrado

$$d(X,\hat{X}) = \sum_{i=1}^{k} (x_i - \hat{x}_i)^2 \qquad (5)$$

como nuestra función de distorsión. Esta medida define una partición cuyas fronteras consisten de hiperplanos en el espacio k dimensional. Es importante esta elección, porque como se verá después, se usarán discriminantes lineales (perceptrones) como elementos de diseño del codificador vectorial. Un perceptrón representa precisamente un hiperplano k dimensional.

## II.3 CUANTIZADORES DE VECINO MAS CERCANO

Definimos un cuantizador vectorial de vecino más cercano (nearest neighbor, NN) o Voronoi como aquel cuyas células están dadas por

$$R_i = \{ x : d(x,y_i) \leq d(x,y_j) \quad \forall j \in J \} \qquad (6)$$

dada la medida de distorsión $d(\cdot,\cdot)$ . En este cuantizador la codificación se realiza mediante una búsqueda exhaustiva en los vectores de C escogiendo aquel que origine la menor distorsión dado el vector de entrada. Esta clase de cuantizadores es muy importante, y veremos a

continuación sus características en cuanto a minimizar distorsión de acuerdo a ciertas condiciones.

# III. DISEÑO DE CUANTIZADORES VECTORIALES

## III.1 OPTIMALIDAD Y DISEÑO DE CUANTIZADORES VECTORIALES.

El objetivo principal en el diseño de cuantizadores vectoriales es encontrar un code book, especificando el decodificador, y una partición ó regla de codificación para definir el codificador, que maximice una medida total de comportamiento considerando la secuencia entera de vectores a ser codificados durante el tiempo de vida del cuantizador. Esta medida puede ser realizada mediante un promedio estadístico de una medida de distorsión o por medio del peor caso de esta distorsión.

Considerando aquí el primer criterio , el promedio estadístico de la distorsión para un cuantizador vectorial Q es

$$D = Ed(X,Q(X)) = \int d(x,Q(x))\, f_X(x)\, dx \qquad (7)$$

donde $f_X(x)$ es la distribución conjunta del vector X y la integración se entiende como una integral múltiple sobre el espacio k dimensional. Asumimos que el tamaño N del code book está dado, que el vector aleatorio k dimensional X está estadísticamente especificado y que una medición de distorsión d ha sido escogida. Indiquemos ahora las consideraciones de optimalidad para un cuantizador con estas premisas.

## III.2 OPTIMIZACIÓN DE UN CODIFICADOR DADO UN DECODIFICADOR.

CONDICION DE VECINO MAS CERCANO. Para un conjunto dado de niveles de salida Y las células de partición que satisfacen

$$R_i \subset \{ x : d(x,y_i) \leq d(x,y_j) \; ; \; \forall j \neq i \} \qquad (8)$$

Esto es

$$Q(x) = y_i \quad \text{solo si} \quad d(x,y_i) \leq d(y_j) \text{ para todo } j .$$

y entonces

$$d(x,Q(x)) = \min_{y_i \in Y} d(x,y_i) \qquad (9)$$

Proposición : Para un code book dado, una partición óptima es una que satisfaga la condición de vecino más cercano.

Prueba. Dado un codebook Y, la distorsión promedio de acuerdo a (7) tiene una cota inferior definida

$$D = \int d(x,Q(x)) f_X(x)\, dx \geq \int \min_{i \in J} d(x,y_i)\; f_X(x)\, dx \qquad (10)$$

y esta cota se obtiene si Q asigna el vector de reproducción con menor distorsión dado el vector de entrada.

## III.3 OPTIMIZACION DE UN DECODIFICADOR DADO UN CODIFICADOR

Ahora consideremos la optimalidad del code book para una partición dada. Esto conduce a la condición de centroide para especificar el code vector asociado con cada región de la partición. Definimos el centroide cent(R) de cualquier conjunto $R \in \mathbb{R}^k$ con probabilidad distinta de cero como el vector $y^*$ ( si existe ) que minimiza la distorsión entre un punto x en R promediada sobre la distribución de probabilidad de X dado que X cae en R. Así

$$y^* = \text{cent}(R) \quad \text{si} \quad E\{ d(X,y^*) \mid X \in R \} \leq E\{d(X,y) \mid X \in R \}$$

Para todo $y \in \mathbb{R}^k$ . De modo equivalente, podemos escribir esto como el mínimo inverso

$$\text{cent}(R) = \min_y{}^{-1} E\{ d(X,y) \mid X \in R \} \qquad (11)$$

Así, el centroide es un vector representativo en sentido estadístico de la región R. Para el caso de la distorsión cuadrática usada se tiene que $E\{ \| X - y \|^2 \}$ se minimiza si y es la media de X. De esta manera

$$\text{cent}(R) = E\{ X \mid X \in R \} \qquad (12)$$

Si asumimos que cada punto en R tiene igual probabilidad, el centroide se calcula como

$$\text{cent}(R) = \frac{1}{L} \sum_{i=1}^{L} x_i \qquad \text{para } x_i \in R$$

CONDICIÓN DE CENTROIDE. Para una partición $\{ R_i ; i=1,\ldots,N \}$ los code vector óptimos satisfacen

$$y_i = \text{cent}(R_i)$$

Prueba. La distorsión promedio está dada por

$$D = \sum_{i=1}^{N} \int d(X, y_i) \, f_X(x) \, dx$$

dado que la partición es fija, cada término puede minimizarse por separado hallando $y_i$ que minimize la distorsión esperada. Por definición, $y_i = \text{cent}(R)$ minimizará D.

Diremos que un cuantizador vectorial es localmente óptimo si cada pequeña perturbación de sus code words no incrementa la distorsión D. Notemos que el hecho de que las dos condiciones de optimalidad más arriba indicadas se cumplan no implica que el cuantizador sea globalmente óptimo.

## III.4 DISEÑO DE CUANTIZADORES VECTORIALES.

Las condiciones de optimalidad descritas permiten plantear un método iterativo de diseño de cuantizadores vectoriales óptimos. Todos los métodos de diseño parten de un code book inicial que iterativamente es mejorado. El trabajo de Lloyd [3] permite diseñar un cuantizador escalar óptimo para señales con estadística conocida. Partiendo del algoritmo I en este estudio, es posible generalizarle para el caso multidimensional que aquí nos interesa : cómo mejorar un code book. La versión generalizada según [1] es :

### III.4.1 ITERACIÓN DE LLOYD PARA MEJORAMIENTO DE CODE BOOKS DADA UNA ESTADISTICA CONOCIDA.

a) Dado un code book $Y_m = \{ y_i \}$ , encontrar la partición óptima en células de cuantización, esto es, usar la condición de vecino más cercano para formar las células de vecino más cercano :

$$R_i = \{ x : d(x,y_i) \leq d(x,y_j) \ ; \ \forall \ j \neq i \}$$

b) Usando la condición de centroide, hallar $Y_{m+1}$ , el codebook óptimo para las células encontradas.

Para aplicar esto necesitamos conocer la función de probabilidad de los vectores k dimensionales, y además efectuar integraciones en k dimensiones sobre regiones por lo general complejas. Estos factores son desventajas. Por ello es necesario obtener una versión de esta iteración

para datos empíricos. Si contamos con un número muy grande de muestras es posible demostrar que bajo ciertas condiciones se puede garantizar que si el cuantizador vectorial es óptimo para este conjunto de muestras lo será también para la función de probabilidad real, así como el tamaño del conjunto tienda a infinito [4]. De acuerdo a esto, se tiene la

III.4.2 ITERACIÓN DE LLOYD PARA DATOS EMPÍRICOS.

a) Dado un code book $Y_m = \{ y_i \}$,y un conjunto empírico de datos o secuencia de entrenamiento $T = \{ v_1, v_2, \ldots, v_n \}$ particiónese el conjunto de entrenamiento en conjuntos $S_i$, usando la Condición de Vecino más Cercano:

$$S_i = \{ x \in T : d(x,y_i) \leq d(x,y_j) \ \forall \ j \neq i \}$$

b) Usando la Condición de Centroides, calcular los centroides de los conjuntos $S_i$ encontrados para obtener el nuevo code book $Y_{m+1}$.

Esta iteración sólo puede disminuir ó dejar igual a la distorsión promedio, dado que el paso a) solo puede mejorar o dejar sin cambio al codificador dado el decodificador y b) también, para el decodificador dado el codificador, de acuerdo a las condiciones de optimalidad mencionadas.

El algoritmo de diseño aplicando esta definición es :

### III.4.3 ALGORITMO DE LLOYD GENERALIZADO .

1. Principiar con un code book inicial $Y_1$. Inicializar m=1 .
2. Dado el code book $Y_m$ realizar la iteración de Lloyd para generar el code book mejorado $Y_{m+1}$ .
3. Calcular la distorsión promedio para $Y_{m+1}$. Si ha cambiado ligeramente desde la última iteración, parar. Si no, entonces actualizar m+1 → m y regresar al paso 2.

Este algoritmo converge en un número finito de pasos produciendo code books mejores o al menos igualmente buenos siempre, dada una secuencia de entrenamiento finita [1].

### III.4.4 CODE BOOKS INICIALES.

Hay un número de técnicas para obtener code books iniciales [1]. La elección de un codebook inicial es muy importante, pues esto puede llevar a la obtención final de un code book localmente óptimo mejor ó peor. Describiremos solamente la técnica de división (splitting [2]). Esta consiste en obtener code books de tamaño cada vez más grande con una dimensión fija. Se inicia con el centroide de la secuencia de entrenamiento completa, $y_0$, que puede ser dividido en dos palabras (code words), $y_0$ y $y_0+\varepsilon$ , donde $\varepsilon$ es un vector de pequeña norma euclidiana. Este nuevo code book tiene dos palabras y no puede ser peor que el anterior, porque lo contiene. El algoritmo de Lloyd se puede aplicar a este code book de tamaño 2 para producir un código de resolución 1. Al

término de esto, cada una de las nuevas code words puede ser dividida a su vez, y el algoritmo de Lloyd se aplica de nuevo. Se continúa con este procedimiento hasta que se alcance el codebook de resolución deseada. Esta técnica es la base de la transmisión progresiva mediante VQ y de la construcción de cuantizadores vectoriales estructurados en árbol (tree structured vector quantizers, TSVQ [1]).

# IV. VENTAJAS Y DESVENTAJAS DE LA CUANTIZACION VECTORIAL COMO TECNICA DE COMPRESION.

## IV.1 VENTAJAS.

Los métodos de compresión sin pérdidas permiten alcanzar tasas no mayores por lo general de 4 a 1. Además, esta tasa depende de la redundancia de la señal a comprimir. Se pueden obtener tasas muy altas para señales muy redundantes, pero si la señal no lo es mucho entonces la ganancia no es muy considerable. La cuantización vectorial permite lograr tasas mayores que las obtenidas por los métodos sin pérdidas, con un bajo error, y con el detalle importante de que el método de diseño de VQ descrito permite sacar provecho de las dependencias estadísticas presentes en la señal de entrada, cosa que no se puede lograr de modo general usando solamente cuantizadores escalares, y que implica que en sentido estadístico se comprime de la mejor manera posible, y que los bits usados para guardar las codificaciones vectoriales se usan de manera óptima, dado el codificador vectorial.

## IV.2 PROBLEMAS RELACIONADOS CON LA REALIZACION FISICA DE CUANTIZADORES VECTORIALES, Y SOLUCION A ELLOS.

Como hemos visto, dado un code book, el modo optimo de usarlo para minimizar error cuando codificamos consiste en buscar la codeword que arroje la mínima distorsión, dado un vector de entrada. Esto lleva al

siguiente problema : se debe encontrar esta codeword por medio de una búsqueda exhaustiva en el codebook. Pero si el tamaño de este tiende a crecer, entonces el tiempo de búsqueda (o de codificación) aumenta. Si trabajamos en espacios de dimensiones grandes el problema se complica más aún, porque cada evaluación de la función de distorsión se vuelve más compleja. Esto es un grave problema de VQ. Si decimos que el code book es de tamaño N, entonces hay que buscar N veces para encontrar a la codeword óptima. Se han propuesto soluciones para este problema de Full Search VQ, VQ de búsqueda completa, pero se paga un precio en optimalidad de codificación a cambio de rapidez, ó se opta por usar una modalidad de codificación que use códigos de longitud variable, lo que complica la gestión del sistema de compresión de datos.

Un método que resuelve drásticamente el problema de tiempo de codificación es la TSVQ (tree structured vector quantization, VQ estructurada en árbol) [1]. Esta técnica consiste en usar el algoritmo de splitting mencionado anteriormente, conservando cada code book intermedio. La estructura de árbol se obtiene partiendo del hecho de que de cada codeword en un codebook de resolución k obtenido por splitting se obtienen m codewords del codebook de resolución k+1 usando el mismo método. En otras palabras, cada codeword en un codebook de resolución k es hija de una codeword en el codebook de resolución k-1. Nosotros por simplicidad discutiremos la TSVQ en el caso binario (dos codewords hijas).

En TSVQ se usa este árbol de codewords para codificar, del siguiente modo: partiendo de un codebook de resolución 1, para cada vector de entrada X, se halla la codeword de mínima distorsión y luego se compara solamente con las codewords hijas de esta codeword. El proceso continúa

hasta llegar a las hojas del árbol de TSVQ, cuya última selección corresponde a la codebook usada como aproximación para X. Este método de codificación encontrará un código en un número logarítmico de búsquedas. Si el tamaño máximo de codebook usado en el TSVQ es N, entonces para el caso binario se codifica en $\log_2(N)$ pasos. Los problemas son dos; por un lado la codificación es subóptima, porque no se hace una búsqueda completa, y se puede dar el caso de que la codeword hallada no sea la de mínima distorsión. Por otro lado, aún en el supuesto de que se tomara el codebook de tamaño N y se usara para realizar una VQ de full search, en general el comportamiento de este cuantizador puede ser aún mejorado mediante el uso del algoritmo de Lloyd generalizado indicado más arriba, pues si se recuerda, el método de splitting servía como un generador de codebooks iniciales para el algoritmo de Lloyd.

## IV.3 TRANSMISION PROGRESIVA.

No obstante, la TSVQ tiene una capacidad interesante : cada etapa de búsqueda en el árbol corresponde a una aproximación de la reproducción final que va mejorando conforme se desciende en el árbol. Además, en el caso binario, cuando codificamos tomamos una de dos rutas para proseguir con la siguiente etapa de codificación. Entonces cada elección de ruta puede indicarse con un bit, 1 ó 0 según el caso. Es claro que cada codeword en las hojas estará relacionada con un patrón único de unos y ceros producto de las rutas que llevan a ellas desde la raíz. Este patrón de bits puede usarse como un código binario que identifica a cada codeword. La idea es la siguiente : usando un TSVQ, codificar. Si ahora

usamos un decodificador con la misma estructura de datos se puede reconstruir la ruta de aproximación progresiva a la codeword final. Entonces es posible obtener una reproducción progresiva de un vector codificado X. Cada bit decodificado entonces servirá para mejorar una reproducción anterior. Podemos pensar en la utilidad práctica de esto si lo asociamos con el procesamiento de imágenes. Supongamos que estamos haciendo uso de un sistema de base de datos que incluye consulta a imágenes. La transmisión de una imágen requiere un ancho de banda muy grande si se quiere realizar de manera rápida. Pero si este ancho de banda es pequeño en comparación con el adecuado para un manejo ágil de esta información, el tiempo de acceso a ésta es muy grande. Usando la reconstrucción progresiva es posible tener una reproducción aceptable de manera rápida aún con anchos de banda pequeños. Si se estuviera buscando en la base de datos un imágen y esto requiriera discriminar secuencialmente cada una de ellas, es probable que una aproximación intermedia nos permitiera hacer esa discriminación. El proceso se agiliza. No es posible hacer trasmisión ó reproducción progresiva de señales codificadas por full search. Sin embargo, para realizar la transmisión progresiva es necesario tener presente la estructura de árbol del TSVQ tanto en el codificador como en el decodificador. Esto implica que la cantidad de memoria requerida para hacer factible la transmisión progresiva se ve doblada respecto a la usada en un decodificador vectorial simple, debido precisamente a la estructura de árbol. Resumiendo, la TSVQ proporciona un esquema rápido de codificación, pero a cambio sufre una degradación en la calidad de la misma. No obstante, permite realizar variaciones de la técnica básica de VQ que pueden usarse en un rango más amplio de aplicaciones ingenieriles, esto con un costo

extra también : más memoria. Como mostraremos más adelante, es posible diseñar codificadores y decodificadores vectoriales de tal modo que se obtenga un comportamiento óptimo en codificación manteniéndose al mismo tiempo una complejidad de ésta en el orden de $\log_2$ del tamaño del decodificador de mayor resolución en el TSVQ, y que además sea capaz de soportar reconstrucción progresiva sin requerir el doble de memoria. Pero antes de discutir cómo, pasemos revista a un resultado importante que se usará como parte de la técnica de diseño de codificadores VQ óptimos.

## V. EL TEOREMA DE CONVERGENCIA DEL PERCEPTRON.

Este teorema se refiere a la demostración de que un conjunto de vectores k dimensionales S puede ser dividido en los conjuntos $S_1$ y $S_2$ tal que $S_1 \cap S_2 = \emptyset$ por medio de un hiperplano en el espacio k dimensional, y da las indicaciones básicas de un algoritmo que siempre converge en un número finito de pasos para conjuntos linealmente separables, esto es, conjunto que pueden ser clasificados siempre en dos categorías mutuamente excluyentes por medio de un hiperplano k dimensional. Se trata de un resultado en el campo de reconocimiento de patrones. La discusión que aquí presentamos está tomada de [5].

Con cada predicado $\phi$, mediante el cual se asegura que un patrón x posee una propiedad si es verdadero o no si es falso, asociamos la función binaria

$$\lceil \phi(x) \rceil = \begin{cases} 1 & \text{si } \phi(x) \text{ es verdadero,} \\ 0 & \text{si } \phi(x) \text{ es falso.} \end{cases}$$

Para dos vectores $x=(x_1, x_2, \ldots, x_N)$ y $W = (w_1, w_2, \ldots, w_N)$ usamos $x \cdot W$ para denotar el producto escalar usual con resultado $x_1w_1+x_2w_2+ \ldots +x_Nw_N$.
Definamos ahora los vectores $y = (x_1, x_2, \ldots, x_n, 1)$ y $w=(w_1, w_2, \ldots, w_n, -\theta)$. Ahora estamos en posibilidad de definir la función de decisión

$$r = \lceil w \cdot y \geq 0 \rceil$$

Lo que hemos hecho hasta ahora es indicar como calcular si el punto n dimensional y está "por debajo" o "por arriba" del hiperplano con coeficientes w. Ahora, consideremos el conjunto finito $Y_1$ de vectores correspondientes a la categoría 1, y un conjunto finito de vectores $Y_2$ perteneciendo a la categoría 2. Especificaremos una regla de corrección de error y mostraremos que si hay un vector $\hat{w}$ que permita discriminar entre estas categorías, entonces después de un número adecuado de iteraciones será posible obtenerlo con esta regla.

Iniciando con un vector w arbitrario, empleamos la siguiente regla de corrección : si w clasifica al vector actual y correctamente, dejamos a w sin cambio. Si la clasificación es incorrecta, cambiamos w por w' donde

$$w' = \begin{cases} w+y & \text{si y pertenecía a la categoría 1,} \\ w-y & \text{si y pertenecía a la categoría 2.} \end{cases}$$

La idea es como sigue : si y es de la categoría 1 pero fué mal clasificado entonces obtuvimos que $w \cdot y < 0$ cuando deberíamos haber tenido $w \cdot y \geq 0$ . Debido a que $y \cdot y > 0$ para cualquier vector no nulo, tenemos que

$$( w + y ) \cdot y = w \cdot y + y \cdot y > w \cdot y$$

Así, podemos decir que w' clasifica a y de mejor manera que w. Desafortunadamente, si clasificamos a y mejor corremos el riesgo de clasificar incorrectamente a otros vectores. Sin embargo, mostraremos que si existe un vector w que realice todas las clasificaciones de manera correcta, es alcanzable en un número finito de pasos.

Para simplificar la prueba, reemplazamos $Y_2$ con $Y_2' = \{ -Y \mid Y \in Y_2 \}$. Entonces decir que $Y_1$ y $Y_2$ son linealmente separables es decir que hay un vector $\hat{w}$ tal que

$$\hat{w} \cdot y > 0 \text{ para todo } y \in Y = Y_1 \cup Y_2'$$

El proceso de entrenamiento se basará ahora en Y : sea $S_y$ nuestra secuencia de entrenamiento, una secuencia infinita de patrones que contiene sólo elementos de la clase Y, cada uno de los cuales se presenta infinitamente a menudo. Entonces podemos generar la secuencia de vectores $\{ w^1, w^2, \ldots, w^k, \ldots \}$ como sigue, donde $w^1$ es arbitrario, y $y^k$ es el k-ésimo patrón en la secuencia de entrenamiento $S_y$. :

$$w^{k+1} = \begin{cases} w^k & \text{si } w^k \cdot y^k > 0, \\ w^k + y^k & \text{si no.} \end{cases}$$

Deseamos probar que eventualmente alcanzaremos un vector $w^k$ tal que $w^k \cdot y > 0$ para todo $y \in Y$ de modo que $w^k = w_0^k$ para todo $k > k_0$ .

Sea $k_1, k_2, k_3, \ldots$ la secuencia de intentos en los cuales el vector $w^{k1}$ es cambiado, y denotemos $w^{kj}$ por $\hat{w}^j$ y $y^{kj}$ por $\hat{y}^j$ . Entonces tenemos

$$\hat{w}^j \cdot \hat{y}^j \leq 0 \quad \text{y} \quad \hat{w}^{j+1} = \hat{w}^j + \hat{y}^j$$

para toda j, a menos de que $\hat{w}^j$ sea el vector final deseado. Tomando $\hat{w}^1 = 0$ tenemos

$$\hat{w}^{j+1} = \hat{y}^1 + \hat{y}^2 + \ldots + \hat{y}^j \qquad (1)$$

Para probar que el procedimiento termina debemos demostrar que j no

puede ser arbitrariamente grande.

Sea entonces w cualquier vector solución, esto es, $y \cdot w > 0$ para todo y en Y . Podemos definir un número positivo $\alpha$ por la ecuación

$$\alpha = \min \{ y' \cdot w \mid y' \in Y \} \qquad (2)$$

Combinando (1) y (2) deducimos que

$$\hat{w}^{j+1} \cdot w = (\hat{y}^1 + \hat{y}^2 + \ldots + \hat{y}^j) \cdot w \geq j\alpha$$

Ahora, la desigualdad de Cauchy-Schwarz dice que para cualquier par de vectores a,b tenemos $(a \cdot b)^2 \leq |a|^2 |b|^2$, donde $|a|$ es el módulo de a. En el caso presente

$$(\hat{w}^{j+1} \cdot \hat{w})^2 \leq |\hat{w}^{j+1}|^2 |w|^2$$

de aquí :

$$|\hat{w}^{j+1}|^2 \geq \frac{j^2\alpha^2}{|w|^2} \qquad (3)$$

De modo que el módulo al cuadrado del vector $\hat{w}^{j+1}$ debe crecer al menos cuadráticamente con el número de pasos.

Debemos mostrar ahora que este crecimiento no puede continuar indefinidamente. Debido a que $\hat{w}^{j+1} = \hat{w}^j + \hat{y}^j$ y $\hat{w}^j \cdot \hat{y}^j \leq 0$ tenemos que

para cada j

$$|\hat{w}^{j+1}|^2 = |\hat{w}^j|^2 + 2\,\hat{w}^j \cdot \hat{y}^j + |\hat{y}^j|$$
$$\leq |\hat{w}^j|^2 + |\hat{y}^j|^2,$$

lo que lleva, por aplicación repetida a que

$$|\hat{w}^{j+1}| \leq jM \quad \text{donde } M = \max\{\,|y|^2 \mid y \in Y\,\}$$

Esto nos dice que el módulo al cuadrado del vector $\hat{w}^{j+1}$ crece cuando más linealmente con el número de pasos j. Entonces, para cada j tenemos

$$\frac{j^2\alpha^2}{|w|^2} < |\hat{w}^{j+1}|^2 < jM \qquad \text{de manera que}$$

$$j < \frac{M\,|w|^2}{\alpha^2}$$

Así, el procedimiento de corrección de error debe terminar después de -cuando más- $\beta$ pasos, donde $\beta$ es el entero mayor que no exceda a $M|w|^2/\alpha^2$, para cualquier vector solución w. Debido a que cada patrón ocurre un número infinito de veces, la terminación puede ocurrir solamente si un vector solución es hallado, probando el teorema.

Comentarios. Como se verá en el capítulo siguiente, el uso de este teorema no es precisamente para clasificar vectores, sino para minimizar la diferencia $|n_1 - n_2|$, donde $n_1$ es el número de vectores en un

subconjunto $S_1$ de una secuencia de entrenamiento S de n vectores para los cuales, dado un vector solución w, $w \cdot y > 0$ ó $w \cdot y < 0$ $\forall$ y $\in S_1$ y $n_2 = n - n_1$. Esta minimización se puede ajustar a las premisas del teorema eligiendo el predicado $\phi$ de modo que la función de decisión r indicara este hecho precisamente. Este teorema es así válido para probar la convergencia del algoritmo que presentaremos.

# VI. DISEÑO DE CUANTIZADORES VECTORIALES ESTRUCTURADOS EN ARBOL CON CODIFICACION OPTIMA Y DECODIFICACION PARA RECONSTRUCCION PROGRESIVA DE MEMORIA REDUCIDA.

Este capitulo describe la técnica de diseño que proponemos para resolver el problema de optimalidad y de memoria para reconstrucción progresiva y complejidad de codificación reducidas. Lo que presentamos a continuación es la transcripción completa de un artículo remitido para su posible publicación.

## VI.1 PROGRESSIVE TRANSMISSION OF IMAGES USING VECTOR QUANTIZATION WITH FAST OPTIMUM FIXED LENGTH CODING AND REDUCED MEMORY DECODING.

Indexing terms : vector quantization,progressive transmission,encoding algorithms for vector quantization.

Abstract : fixed length coding for progressive transmission using vector quantization is usually suboptimal. In addition, the memory needed to decode is doubled when compared with the amount required for lookup decoding. This paper presents a solution to both problems, describing a general design technique for vector quantizers and a fast encoding algorithm.

Introduction: Progressive transmission of images is a technique in which

an image is transmitted and reconstructed with increased detail as more bits arrive[1]. Progressive transmission using vector quantization requires a particular tree structure on both encoder and decoder stages[1].This structure doubles the memory requirements of the decoder :it is necessary to keep the intermediate approximations. Also, the codes obtained are suboptimal because the encoder do not perform a full search for the nearest neighbor. Here we fixed both problems for the mean squared distortion and image coding.

Let $C = \{C_1, C_2, .. C_n\}$ be a set of k-dimensional vectors, with $n=2^x$ elements , $x \in Z^+ - \{0\}$ and $C_i^j$ is the centroid of the i'th partition $P_i^j$ of C at the j'th stage of design. We denote the number of vectors contained in the partition $P_i^j$ with $card(P_i^j)$. Let $C^0$ be the centroid of the entire set C calculated as follows :

$$C^0 = \frac{1}{card(C)} \sum_{\forall C_i \in C} C_i$$

Now we take one partition of C in two sets, $P_1^1$, $P_2^1$ with $card(P_1^1)$, $card(P_2^1)$ respectively and $P_1^1 \cup P_2^1 = C$ , $P_1^1 \cap P_2^1 = \emptyset$ . The centroids of these sets are

$$C_1^1 = \frac{1}{card(P_1^1)} \sum_{\forall C_i \in P_1^1} C_i$$

$$C_2^1 = \frac{1}{card(P_2^1)} \sum_{\forall C_i \in P_2^1} C_i$$

If we take now the centroid of these two centroids we obtain

$$C^1_{12} = \frac{1}{2}\left\{ \frac{1}{card(P^1_1)} \sum_{\forall C_i \in P^1_1} C_i + \frac{1}{card(P^1_2)} \sum_{\forall C_i \in P^1_2} C_i \right\}$$

In general, $C^1_{12} \neq C^0$. But if $card(P^1_1) = card(P^1_2) = card(C)/2$ it follows that $C^1_{12} = C^0$. Also, assuming this assertion is true it is easy to see that

$$C^1_1 = \frac{1}{2}\{ C^1_1 - C^1_2 \} + C^0$$

$$C^1_2 = \frac{1}{2}\{ C^1_2 - C^1_1 \} + C^0$$

So, we can obtain two centroids using the difference of two vectors multiplied by a constant and summing this to the original centroid. The same is true for the centroids of any successive partitions of C that satisfy the stated conditions. Iterating in this way $\log_2(n) - 1$ times we can obtain all the original k-dimensional vectors only performing successive sums of vector differences to the centroid of C. It is clear that we need only n-1 difference vectors and $C^0$ to achieve this. So, we need n k-dimensional vectors to do a kind of progressive approximation of the points on C. The relation to VQ is as follows : we can take an optimum code book designed with the LBG algorithm[2] with $n=2^x$ k-dimensional code words for some x and by means of the described procedure to construct a tree-structured code book using the centroids obtained. Then, we can use this new code book to encode, and we store the vector differences between its code words on each stage to decode and reconstruct.The gain is that, usually, TSVQ uses approximately 2n code words to decode. Using this result is possible to construct a decoder with n vectors. We achieved the required partitions using a result known

like Perceptron Convergence Theorem[3]. This theorem ensures that if one partition, with certain characteristics, of a given set exists, it is obtainable in a finite amount of time by means of a perceptron and a learning algorithm. Here the characteristics of the partition are only related to the equal number of elements into each of the two disjoint sets resulting of that partition. The algorithm is as follows :

Algorithm I.Encoder design.

1. Starting with an optimum code book $C = \{ C_1, C_2, .. C_n \}$ with $n = 2^x$ for some $x>0$ and $x$ integral, calculate the centroid of C and label it $C^0$. Store this centroid.

2. Using the Perceptron Convergence Theorem proceed to partition C in $P_1^1$ and $P_2^1$ so that $P_1^1 \cup P_2^1 = C$ , $P_1^1 \cap P_2^1 = \emptyset$ and $card(P_1^1) = card(P_2^1) = card(C)/2$ .

3. Calculate the centroids of $P_1^1$and $P_2^1$, $C_1^1$, $C_2^1$. Store these centroids to use on the decoder design.These are the code words of the resolution 1 code book.

4. Continue with steps 2 and 3 with each of these $P_1^1$ sets used like initial set C to obtain the next $P_1^2$ sets and so with each of their successive partition sets $P_1^j$ $j=3,4,\ldots,\log_2(n)-2$ $i=1,2,\ldots 2^j$. The set of centroids obtained for each j are the resolution j code books of the tree-structured encoder.

5. At stage $j=\log_2(n)-1$ we have sets with two elements. The centroids of these elements are calculated directly and stored.

6. The resolution $\log_2(n)$ code book consists of the elements of C .

To construct the decoder proceed as follows :

Algorithm II. Decoder design.

1. Start with the tree-structured code book obtained with algorithm I. We assume it is organized in a data structure with nodes containing two fields for two code words -code_word1 and code_word2- and a field for the difference of these -vector_difference- each, and pointers to child nodes, labeled left and right, and an special pointer named root, that points to the root of the tree-structured encoder.

2. Execute construct_decoder(root) using the recursive definition :

```
construct_decoder(node)
begin
   if node <> NULL then
      node->vector_difference=0.5*(code_word1-code_word2)
      construct_decoder(node->left_child)
      construct_decoder(node->right_child)
   endif
end
```

By definition, if a given input vector V to be coded is closest in stage i to code_word1 then the next node to be visited will be the right child, and the bit produced is 1. To decode simply begin with the root node. Initialize the reproduction R with $C^0$. In stage i with bit $b_i$ as input, we do R=R+vector_difference$_i$ and go to right child if $b_i$= 1, and R=R-vector_difference$_i$ and walk to left child if $b_i$ is 0. The goal of optimality was achieved as follows : by hypothesis, the lower stage code book on the tree has optimal code words if we perform a full search on it, because it was designed using the LBG algorithm. With this encoder we run the following algorithm :

Algorithm III. Fast Nearest Neighbor Encoder Design.

1. Start with an optimum code book C designed with the LBG algorithm and build a tree-structured (TS) encoder and decoder using algorithms I and II.

2. Generate a test vector U and encode it with the TS encoder, obtaining a code word $C_i$ . Store the distortion $d_{tsvq}$ obtained in this encoding..

3. Perform a full search on C for the nearest neighbor $C_j$ of U. If the distortion obtained $d_{fu}$ is below $d_{tsvq}$ then put $C_j$ in a list of neighbor code words $L_i$ associated with the region of the $C_i$ found before.

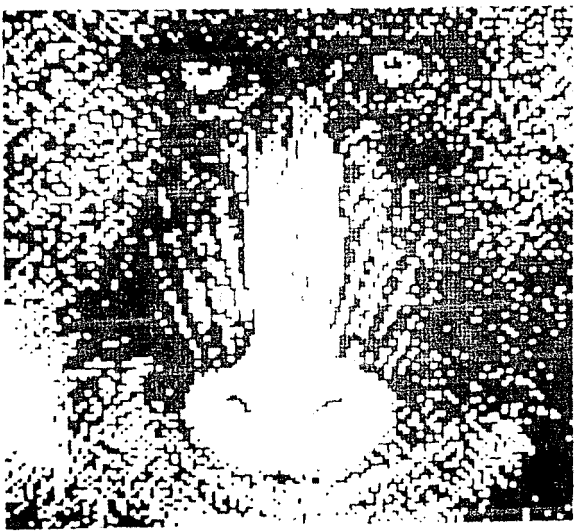
4. Repeat steps 2 and 3 a sufficient number of times.

5. Keep the tree-structured encoder and the lists of near code words. This is the new encoder.

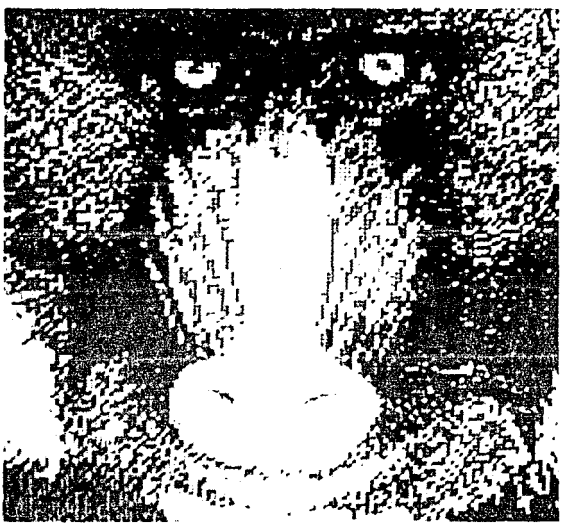
To encode use the TS code book to find a first approximation. Then perform a full search within the adequate list to find the global optimum code word.We can see that this algorithm will always find the nearest neighbor as follows : with the TS encoder we are mapped to certain k-dimensional regions. If we now perform the full search for all the points that lie in this region we will always find the best code words for all these points. As we take more points in this region we tend more and more to find the complete list of near code words that will contain the optimum code word that we desire.

SIMULATIONS . We used the image "baboon" with 512 x 512 pixels to construct our test vector quantizer with block of 4 x 4 pixels . Each pixel used one byte. To design the nearest neighbor encoder we generate 16-dimensional vectors with each component uniformly distributed in the

Imágen "baboon" original. 8 bits por pixel



Imágen "baboon" reconstruida usando 0.0625 bits por pixel .Primera etapa del proceso de reconstrucción progresiva del original. Tasa de compresión : 128 a 1.

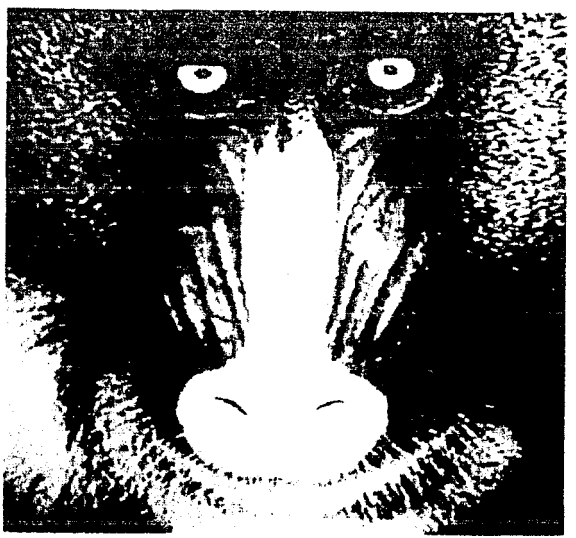


Reconstrucción con 0.125 bits por pixel. Segunda etapa .Tasa de compresión : 64 a 1.

Tercera etapa : imágen obtenida usando 0.1875 bits por pixel. Tasa de compresión : 42.66 a 1.

Cuarta etapa. 0.25 bits por pixel. Tasa de compresión : 32 a 1.

Quinta etapa. 0.3125 bits por pixel. Tasa de compresión : 25.6 a 1.



Sexta y última etapa de reconstrucción. Se usan 0.375 bits por pixel, lo que implica una tasa de compresión final de 21.333 a 1.

range 0 to 255. The original image and the progressive reconstructions using 0.0625, 0.125, 0.1875, 0.25, 0.3125 and 0.375 bpp are shown.

CONCLUSION. We presented a new approach to progressive transmission with reduced memory decoding, and a fast optimum encoding algorithm for VQ using mean squared distortion. This algorithm promises a substantial reduction in complexity encoding and, because it uses a partial full search, all previous improvements to the brute force full search[4] can be incorporated to it providing with progressive transmission capability useful also with improvements to the basic TSVQ technique[5]. With an adequate implementation only a small amount of extra memory is required, specially on the encoder.

Acknowledgements.We are very grateful with U.N.A.M. and IBM Scientific Center of Mexico for the support given to this work, and with the graphic designer Lorena Robles for the excellent photographs.

Robles De La Torre, Gabriel
Fernández Anaya, Guillermo.
Department of Computer Engineering.
Facultad de Ingeniería.
Universidad Nacional Autónoma de México.
México City 04510
México.

REFERENCES.

1. Tzou,Kou-Hu : 'Progressive image transmission: a review and comparison of techniques', Optical Engineering, 1987, 26,pp 581-589.

2. Linde, Y., Buzo, A., and Gray, R.M. : 'An algorithm for vector quantizer design',IEEE Trans. Commun., 1980,COM-28,(1), pp. 84-95.

3. Arbib, Michael A. : 'Brains, Machines, and Mathematics'. Springer Verlag, 1986, pp 61-69.

4. Soleymani M.R. and Morguera S. : 'A fast MMSE encoding technique for vector quantization',IEEE trans. Commun.,1989, 37, pp. 656-659.

5. Riskin E.A. : 'Variable rate vector quantization for medical image compression',IEEE trans. on Medical Imaging,1990,9,pp. 290-298.

## VII. CONCLUSIONES.

La técnica de diseño de cuantizadores vectoriales de baja complejidad de codificación óptima descrita efectivamente ofrece una solución tanto al problema de tiempo de codificación como a la cuestión de cómo realizar la reconstrucción progresiva de una señal mediante TSVQ usando solamente la mitad de la memoria requerida en el caso usual. La tasa de compresión lograda en el ejemplo mostrado es relevante, y más aún con el hecho de que el cuantizador vectorial empleado es extremadamente simple y su codebook pequeño. Esto promete mucho, dado que el valor de la señal a ruido entre la imágen original y la reconstrucción final es alta, de alrededor de 18 decibeles.

En las simulaciones que se realizaron para diseñar el codificador óptimo de memoria reducida en esta tesis se usó el método de Montecarlo para generar vectores con componentes uniformemente distribuidas, dentro del dominio de todas las imágenes representables con 8 bits; esto es, vectores con componentes en el rango de 0 a 255. Como se muestrea por Montecarlo de manera uniforme, se desprecia la información de la distribución empírica que se conoce ya en este momento del diseño, y que reside en los centroides obtenidos. Es posible acelerar dramáticamente el tiempo de obtención del codificador óptimo si ahora se generan vectores por Montecarlo en las cercanías de estos centroides y se usan en el diseño. Así, cada vector aleatorio será más útil en general para el proceso de diseño.

El problema mayor a nuestro juicio es la formación de codebooks lo suficientemente representativos estadísticamente como para ofrecer un comportamiento de calidad ante una muestra amplia de señales a codificar fuera del conjunto de entrenamiento. Las condiciones especificadas para los procesos estocásticos usados para modelar las señales empleadas en la justificación de la validez del algoritmo de LLoyd deben tomarse como una guía solamente. En un caso de aplicación real, más que disponerse a probar si una señal dada cumple con las características estadísticas adecuadas teóricamente deberá adecuarse el diseño del sistema de VQ de tal modo que se logre un comportamiento aceptable según el uso.

## VIII. Agradecimientos.

Quiero agradecer el apoyo brindado para la realización de este trabajo al Jefe del Departamento de Ingeniería en Computación de esta Facultad, Ing. Cristóbal Peña Olivo; a la Directora de Cómputo para la Investigación de la Dirección General de Servicios de Cómputo Académico, M. en C. Christine Allen; al personal del Departamento de Percepción Remota del Centro Científico de IBM de México, especialmente al Dr. Guillermo González y al Matemático Francisco Zamora González, con quienes es un gusto trabajar; al Centro Científico de IBM en general; a mi Director de Tesis, M. en I. Guillermo Fernández Anaya, y al soberbio equipo humano y material del Centro de Cómputo de la Dirección General de Administración Académica, sita en el edificio IIMAS, en cuyo sistema CDC Cyber 855 se corrieron las simulaciones necesarias para la elaboración de este trabajo de tesis.

## IX. Bibliografia General.

1. Gersho, A., Gray, R.M. : ' Signal Coding: Quantization and Compression', versión perliminar, 1988.

2. Linde, Y., Buzo, A. y Gray, R.M. : 'An algorithm for vector quantizer design', IEEE Transactions on Communications, vol. COM-28, pp 84-95, enero de 1980.

3. Lloyd, S.P. : 'Least squares quantization in PCM', reimpreso de un reporte técnico de los Laboratorios Bell en el ejemplar especial de marzo de 1982 sobre cuantización de IEEE Transactions on Information Theory.

4. Sabin, M.J. y Gray, R.M. : 'Global convergence and empirical consistency of the generalized Lloyd algorithm', IEEE transactions on Information Theory, Vol. IT-32, pp 148-155, marzo de 1986.

5. Arbib, Michael A. : 'Brains, Machines, and Mathematics'. Springer Verlag, 1986, pp 61-69.