92

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

ANALISIS E IMPLEMENTACION DE UN SISTEMA DE COMPRESION PARA IMAGENES BINARIAS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

IRAM TERESITA OVIEDO CHAVEZ

JUAN MANUEL RODRIGUEZ BERNAL

DIRECTOR: DR. ROGELIO ALCANTARA SILVA

MEXICO, D. F. 1996

AGRADECIMIENTOS

*A la **Universidad Nacional Autónoma de México** y en especial a la **Facultad de Ingeniería** por la educación recibida en sus aulas.*

*A la **División de Estudios de Posgrado de la Facultad de Ingeniería** por las facilidades que nos brindó.*

*Al **Dr. Rogelio Alcántara Silva** por brindarnos su confianza y apoyo incondicional, además de la paciencia y motivación que tuvo en la realización de esta tesis.*

*A la **fundación UNAM** por el apoyo económico que nos brindó para llevar a cabo este proyecto.*

*A todos los **maestros de la Facultad de Ingeniería** por la formación que nos dieron.*

Dedicatoria.

A mis padres Francisco Oviedo y Guillermina Chávez, por darme su amor, confianza y apoyo para seguir adelante.

A mamita y Miriam, por su especial cariño.

A mis hermanos Francisco Javier, Ibeth, Aldo y Vianney, por su motivación día con día.

A mis amigos de la Facultad, por compartir a su lado momentos agradables.

A la vida por demostrarme que no hay barreras imposibles, cuando se tiene la ilusión de llegar al final.

Y a todos aquellos que colaboraron siempre conmigo.

Les dedico esta tesis, por brindarme lo mejor de cada uno, y por su disposición para ayudarme a lograrlo.

Iram Teresita Oviedo Chávez.

Dedicado con especial cariño a:

Mis padres,
que me enseñaron que en la vida nada
se logra sin sacrificios, y a quienes debo
todo lo que hasta ahora soy.

Salomón,
como un homenaje post mortem.

Mis hermanos,
que nunca dejaron de darme ánimos
para continuar siempre adelante.

Mis amigos de la Facultad,
con quienes compartí grandes momentos
dentro y fuera de este recinto del saber.

La vida,
de la que aprendí tanto en estos años.

A todos aquellos que me ayudaron,
de alguna u otra manera,
a llegar hasta este momento.

Juan Manuel Rodríguez Bernal.

ÍNDICE

pág.

CAPÍTULO I

INTRODUCCIÓN

CAPÍTULO I.

INTRODUCCIÓN

En nuestros días, la computación se ha convertido en un importante apoyo, para el desarrollo de la tecnología en los diferentes campos de investigación. Esto va desde el mejoramiento de aparatos de uso común (como cámaras fotográficas, videocaseteras, fax, etc.) hasta la creación de máquinas con un poder de procesamiento de millones de instrucciones por seg. (mips).

Una de las áreas que más interés ha despertado, es el *Procesamiento de Imágenes Digitales (PDI)*, la cual se dedica al manejo de imágenes por medio de equipo electrónico digital, ya sea generando, transportando, codificando o seleccionado la información contenida en éstas. Debido a esto, se busca manejar las imágenes en forma digital, transformando aquéllas que se adquieren por medio de equipo analógico. De ésta forma se ha llegado a un mejoramiento de las herramientas que realizan esta tarea.

El PDI fue un paso importante que permitió avances en otros campos de la ciencia y tecnología, tales como la astronomía, la medicina, la biología, la robótica, etc., donde se necesita obtener información visual. En algunos casos, esta tiene que ser modificada con gran facilidad y rapidez, pero manteniendo las características más importantes de la misma. De aquí la gran importancia al Procesamiento de Imágenes Digitales, como una rama de investigación.

Uno de los problemas al que nos enfrentamos actualmente, es el manejo de grandes volúmenes de información por medio de computadoras, ya sea locales o remotas. Esto nos lleva a buscar formas de almacenar en bases de datos, toda la información que se genere de los procesos cotidianos, realizados dentro de un centro de cómputo. Sin embargo, en muchas ocasiones estas bases de datos crecen en una forma acelerada, requiriendo a su vez mayor espacio en medios de almacenamiento tales como discos duros y cintas.

Además de este incremento en los datos manejados, existe un crecimiento de las fuentes de información que generan tal cantidad de datos, tales como usuarios, nodos en una red, conexiones remotas con otras redes, etc. De ésta forma, se busca una mejoría en los medios que nos permiten transportar la información, para facilitar su envío y recepción. Sin embargo, aún estos medios cuentan con una limitación en la cantidad de información a transmitir (ancho de banda), lo que tarde o temprano causará problemas ya sea en la velocidad o eficiencia.

En el caso de las imágenes digitales, la cantidad de datos necesarios (bits/bytes) para su representación es bastante elevada, sobre todo para imágenes a color o de vídeo. Una solución podría ser, aumentar o mejorar el número de dispositivos para transmitir o almacenar las imágenes que se manejen. Sin embargo, esto implicaría un aumento considerable en el costo, para la adquisición de equipo extra.

Considerando que, las imágenes representadas en su forma original contienen una cantidad significativa de redundancia (información repetitiva), se hace posible que mediante un proceso de compresión, esa información pueda ser descartada. Y así, podamos codificar dicha imagen, reduciendo los bits/bytes necesarios para representarla.

Si observamos el número de bits que se necesitan para representar una imagen binaria, de 1728 x 2376 pixeles 1 bit/pixel, esto nos daría aproximadamente 4×10^6 bits. Lo que significaría, una gran demanda en medios de almacenamiento y transmisión, para manejar una cantidad enorme de imágenes de este tipo. De aquí podemos decir, que es una necesidad el adecuar este tipo de imágenes para su manejo, de tal forma que se requieran menos recursos.

Las imágenes binarias son aquéllas que presentan únicamente 2 tonalidades (blanco y negro), y generalmente se utiliza un bit para representar cada tonalidad ("1" para el blanco y "0" para el negro, en la mayoría de los casos). A este tipo de imágenes pertenecen las imágenes de fax, facturas, diagramas, páginas de libros y documentos en general. El uso de esta clase de imágenes se hace indispensable en oficinas, en donde se necesita manipular los documentos originales, o bien en la transmisión de gráficas de resultados a distancia, lo que también conlleva los problemas que anteriormente describimos. Por lo que en éstos casos la compresión es una solución muy factible.

Como hemos visto, la compresión de información, es la mejor opción, para diferentes áreas de procesamiento, por lo cual, tratamos de diseñar una solución que hiciera la tarea de comprimir información, pero aplicado en particular a documentos digitalizados. Creemos que hay muy poca información sobre compresión de esta clase de imágenes, ya que la mayoría de esta, se centra sobre el procesamiento de documentos por medio de reconocimiento de caracteres (OCR).

En esta tesis planteamos un sistema de compresión original para imágenes binarias, el cual está diseñado a partir de algoritmos estudiados, planteando algunas modificaciones de programación y diseño que aumentan considerablemente la velocidad de procesamiento.

Este sistema de compresión, tiene como base conocimientos de algoritmos de codificación aplicados a imágenes en tonos de gris, pero los modificamos para adaptarlos al tipo de imágenes estudiado. Incluimos dentro del sistema un algoritmo de compresión con pérdida, el cual da mayor razón de compresión que los algoritmos utilizados en las etapas previas. Y en efecto, se obtuvieron resultados favorables, algunos de los cuales pudimos compararlos con tablas de algoritmos aplicados a este tipo de imágenes y otros evaluando la calidad de la imagen. En muchos casos los resultados superaron a los obtenidos por otros algoritmos.

Se programaron los algoritmos de codificación (Run Length, Huffman y VQ) por separado. Después se unieron en uno solo para formar el sistema de compresión propuesto. Por lo que el sistema consta de tres módulos de codificación, y tres de decodificación, en donde la imagen es recuperada por completo. Las variaciones mejoraron la razón de compresión, y el tiempo de procesamiento obteniendo una imagen de buena calidad.

En el capítulo dos, se dará un repaso general de los principales conceptos sobre el tema de compresión de imágenes. En el capítulo tres se mencionará la clasificación fundamental de las principales técnicas de compresión de imágenes que existen, según la forma de codificar cada imagen. En el capítulo cuatro se estudiarán algunos de los algoritmos aplicados exclusivamente a imágenes binarias. En el capítulo cinco, se da una explicación más detallada de las consideraciones de cada uno de los módulos que componen nuestro sistema , así como el desempeño de cada uno de ellos. Finalmente en el capítulo seis, se dan las conclusiones a las que se llegó con las simulaciones realizadas, así como las perspectivas de mejora del sistema propuesto.

CAPÍTULO II

FUNDAMENTOS BÁSICOS DE LA COMPRESIÓN

CAPÍTULO II.

FUNDAMENTOS BÁSICOS DE LA COMPRESIÓN

A continuación se describen los fundamentos y conceptos básicos, para comprender mejor la teoría de los algoritmos involucrados en la compresión de la información.

2.1 COMPRESIÓN DE INFORMACIÓN

La compresión de información, es el proceso de codificar un conjunto de datos para reducir los requerimientos de almacenamiento o transmisión. Esto se logra al eliminar lo que se denomina redundancia.

La redundancia es una parte de los datos, la cual nos proporciona información repetitiva, de la cual podemos prescindir. Se pueden reconocer tres tipos de redundancia [RAB91]:

- *Redundancia espacial*, es la correlación (o dependencia) que existe entre valores de pixeles vecinos.
- *Redundancia espectral*, se refiere a la correlación entre diferentes planos de color o de diferentes bandas espectrales.
- *Redundancia temporal*, se debe a la correlación entre diferentes estructuras dentro de una secuencia de imágenes.

De acuerdo con esto, podemos decir que existen dos diferentes clases de algoritmos de compresión: algoritmos de *compresión sin pérdida de información*, en donde los datos comprimidos, pueden descomprimirse y ser idénticos a los datos originales, mientras que con los algoritmos de *compresión con pérdida de información*, los datos descomprimidos son una aproximación aceptable a los originales, de acuerdo a algún criterio de fidelidad. Por ejemplo, con una imagen digitalizada puede ser solamente necesario, que la imagen descomprimida se vea igual de bien que la imagen original para el ojo humano.

Los tipos de datos que a los cuales se aplica actualmente la compresión incluye texto, programas fuentes, código objeto, mapas de bits, datos numéricos, gráficas, mapas, voz, música, datos científicos e instrumentales, facsímiles, imágenes a color o en tonos de grises, imágenes médicas, vídeo, animación y datos espaciales, entre otros.

2.2 COMPRESIÓN DE IMÁGENES.

La compresión de imágenes, es un tema derivado de la compresión de información, cuya finalidad es codificar una imagen con la menor información necesaria, pero sin perder las características suficientes para ser reconocida posteriormente, de acuerdo al uso que tendrá dicha imagen. Se pueden establecer diferentes criterios de evaluación, para definir que tanta y que tipo de información se requiere al recuperar la imagen. Así se pueden utilizar criterios cuantitativos como el error cuadrático promedio o bien cualitativos como las diferencias percibidas por el ojo humano.

El proceso consiste en obtener la información pura, de una imagen raster[1], referente a cada uno de los pixeles, y codificarla de tal forma que el número de bits necesarios para representarla sea menor al empleado en la imagen original.

2.3 USOS Y BENEFICIOS DE LA COMPRESIÓN

Los sistemas de compresión brindan varias ventajas en el procesamiento de imágenes digitales, principalmente son:

- La compresión permite minimizar, los requerimientos de espacio en almacenamiento de imágenes, datos, voz, etc. , disminuyendo gastos en equipo que realiza esta tarea.
- Al comprimir, aumenta la razón de transmisión, de tal forma que se podrá transmitir mayor información en menor tiempo.
- Los sistemas de compresión pueden permitir encriptamiento y codificación contra errores, aumentando así la seguridad de los datos en el canal de comunicación.

[1] Una imagen *raster* está formada por líneas, cada una de ellas sigue físicamente a la otra, es decir, el renglón i+1 sigue al renglón i.

En la figura 2.1 se ilustra un diagrama de bloques del proceso de compresión-descompresión, el cual se representa como una caja negra.

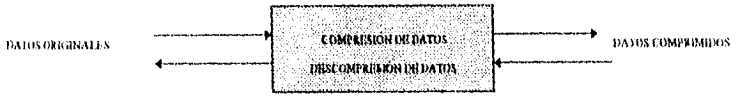


FIG. 2.1. Diagrama Básico de compresión de datos

2.4 MODELO VISUAL

Debido a que el sistema visual humano (SVH), se puede considerar como la etapa final del reconocimiento de imágenes, es importante identificar un modelo que lo represente. Tomando en cuenta que la distorsión en una imagen puede ser controlada, el objetivo de un sistema de codificación es explotar las limitaciones del SVH.

Las características a considerar del sistema SVH son: la *sensibilidad al contraste, la crominancia, la respuesta espacial y el efecto de enmascaramiento.* Se ha establecido ampliamente que la *sensibilidad* de un humano a los cambios de contraste en imágenes monocromáticas es de aproximadamente un 2 %. Por otro lado, en cuanto a la sensibilidad a los cambios de crominancia, se ha encontrado que el ser humano es muy sensible a los cambios de verde, sensible en forma moderada a los cambios de rojo y menos sensible a los cambios de azul. Se sabe que los receptores en el ojo: la retina, bastones y conos no responden linealmente a una excitación. En este sentido, se han sugerido varios modelos de representación, sin embargo, los que han tenido una mayor aceptación, son los que presentan una respuesta logarítmica y los que presentan una respuesta de raíz cúbica [RAB91].

En la actualidad se han elaborado bastantes estudios sobre la respuesta en frecuencia espacial del SVH, concluyéndose en forma general, que el ojo humano actúa como un filtro pasabanda no-lineal, como se puede observar cualitativamente en la **fig. 2.2**[RAB91]. La atenuación de frecuencias espaciales altas, se atribuye a las limitaciones de respuesta espacial que tiene la óptica del ojo. Por otro lado, la interacción de los bastones y conos, conocida con el nombre de inhibición lateral, provoca un efecto de diferenciación espacial dando como resultado una atenuación de las frecuencias bajas. Por último, el efecto de enmascaramiento del SVH se refiere a la reducción de sensibilidad de contraste, en la vecindad límite de los gradientes de luminancia (brillo) y color. En estas regiones existe una percepción reducida, de la distorsión causada por la codificación [RAB91].

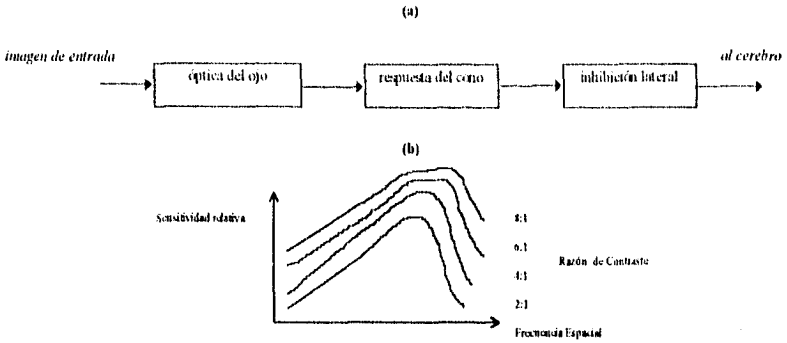


Figura 2.2 Modelado de la visión humana: (a) Modelo; (b) función de transferencia.

2.5 FORMACIÓN DE UNA IMAGEN DIGITAL

Una imagen digital puede ser generada apartir de diferentes tipos de fuentes por ejemplo: una cámara, un espectrógrafo, o una fotografía. Sin embargo, todas ellas son de tipo analógico. Por lo tanto para poder generar una imagen digital, es necesario muestrear dichas fuentes en forma discreta por medio de sensores. A estas muestras se les conoce con el nombre de pixeles o pels (*contraction of picture element*). Los valores de los pixeles producidos por el sensor son continuos sobre un rango finito y tienen una relación lineal con la intensidad de energía radiante de cada localidad muestreada.

La manera más común de realizar el muestreo sobre la imagen analógica, es utilizar una rejilla rectangular equidistante. De esta forma, obtenemos una pequeña porción de la imagen, representada por un valor definido dentro de un intervalo de intensidades de energía. Al número de porciones muestreadas por unidad de área, se le conoce como *tasa de muestreo* del sistema, y a su vez debe cumplir con el *Teorema de Muestreo de Nyquist*

Si la tasa de muestreo no cumple con este teorema, se presenta un *efecto de aliasing*, generado al tener frecuencias espaciales mayores a la mitad de la tasa de muestreo. Para evitar esto, se puede realizar un prefiltrado, para limitar el ancho de banda en la imagen.

En un sistema de digitalización de imágenes la tasa de muestreo se da frecuentemente en términos de la resolución del "*scanner*" utilizado, siendo ésta el inverso de la tasa de muestreo.

En general la resolución del "scanner" requerida depende directamente de la aplicación. Por ejemplo, para una radiografía de 14x17 *pulgadas* observada por un radiólogo a una distancia típica de 14 *pulgadas*, una resolución de scanner de 70 *µm/pixel* que alcanza a capturar frecuencias espaciales tan grandes como de 7 *ciclos/mm* es adecuada (esto se hace tomando en cuenta el Sistema Visual Humano SVH). Para un negativo de 35 *mm* de una fotografía, sujeto a una ampliación y a determinado procesamiento, una resolución de scanner de 12 *µm/pixel* es más común.[RAB91].

En seguida de la etapa de digitalización, viene una etapa de cuantización, donde cada valor de intensidad de energía es representado por un número entero de 0 a N que lo definirá. Tomando en cuenta que las imágenes digitalizadas serán observadas por una o varias personas en la mayoría de las aplicaciones, es importante realizar la cuantización en un dominio que vaya de acuerdo con el SVH de percepción. Por lo que, en muchos casos los valores de los pixeles son sujetos a no linealidades (logarítmicas o funciones de raíz cúbica) antes de ser cuantizados, consiguiéndose un ajuste o aproximación a las no linealidades del SVH.

El número de niveles de cuantización requerido para una adecuada representación de la imagen, depende también de la aplicación. Para documentos de texto binario, se requieren únicamente dos niveles (1 bit/pixel), es decir, cada punto muestra puede tomar uno de dos valores; por ejemplo, negro o blanco. Para escenas naturales o fotografías de tono continuo, es común usar 8 bits/pixel(256 niveles). Sin embargo, dependiendo del rango dinámico de la fuente y del tipo de salida del sensor (logarítmica o lineal), puede ser necesario emplear 10 o 12 bits/pixel.

Consecuentemente y con el propósito de corregir los errores que se pueden cometer al implementar analógicamente las no linealidades, los digitalizadores de imágenes más sofisticados, primero utilizan de 12 a 14 bits, para cuantizar los valores de los pixeles en un espacio lineal, y aprovechando la no linealidad de los datos digitales, pueden entonces recuantizar en 8 bits [RAB91].

2.6 PROCESO DE MUESTREO

El proceso de muestreo consiste en obtener valores representativos de una determinada señal o imagen, esto es representar una señal $x_c(t)$ en valores de $x_c(nt)$, uniformemente espaciados en múltiplos enteros del intervalo de tiempo T, conocido como *periodo de muestreo*.

Una forma conveniente de interpretar el muestreo es como un proceso de modulación o multiplicación como el que se muestra en la **figura 2.3** [SCH93]. La señal continua $x(t)$ es multiplicada (modulada) por un tren de impulsos periódico (función de muestreo) $s(t)$, obteniéndose una señal modulada $x_c(t)\, s(t)$.

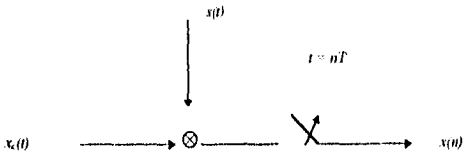


Figura 2.3 Interpretación del Proceso de Muestreo

Esta señal es discretizada en el tiempo, resultando

$$x(n) = \lim_{\varepsilon \to 0} \int_{t=nT-\varepsilon}^{nT+\varepsilon} x_c(t)s(t)dt$$

donde:

$$s(t) = \sum_{l=-\infty}^{\infty} u_0(t - lT)$$

y en donde $u_0(t)$ es una función impulso unitario ideal. En este contexto de interpretación, $x(n)$ representa el área bajo el impulso en un tiempo nT. Deduciéndose que:

$$x(n) = x_c\,(nT)$$

2. 6. 1 TEOREMA DEL MUESTREO DE NYQUIST

El teorema nos da las características mediante las cuales, una imagen muestreada en un conjunto finito con periodicidad T, puede ser recuperada completamente apartir de éste. Una señal analógica $x_c(t)$ puede generar una señal digital $x(n)$, sin embargo para que la señal $x(n)$ sea representativa de $x_c(t)$, se debe de cumplir que $x(n)$ no contenga ningún componente de frecuencia mayor o igual a $1/2T$ [SCH93].

Si este no es el caso, las muestras se encimarán y se producirá un efecto de *"aliasing"* que provoca que la señal reconstruida sea diferente de la original [SCH93]. En estas circunstancias, el Teorema de Muestreo establece la siguiente consideración:

Una señal que no contiene ninguna componente de frecuencia igual o mayor a un valor f_m , puede ser completamente determinada por un conjunto de valores espaciados regularmente, en un intervalo de tiempo $T = 1 / (2f_m)$.

Se puede decir entonces, que la frecuencia de muestreo de una señal se determina por el límite superior de su banda de frecuencia.

2.7 CUANTIZACIÓN

Cuantización es el proceso mediante el cual un valor continuo definido en un intervalo F, es representado por un valor discreto definido dentro de un intervalo L.

Si consideramos $\hat{f}$ como el resultado de cuantizar f, podemos escribir [RAB91]:

$$\hat{f} = Q(f) = r_i \quad , \quad d_{i-1} < f \leq d_i$$

donde Q representa la operación de cuantización, r_i para $1 \leq i \leq L$ representa los niveles de reconstrucción (cuantización) y d_i para $0 \leq i \leq L$ representa los $L + 1$ niveles de decisión. Si los niveles r_i y d_i se logran determinar de alguna forma, la cuantización de f es entonces un proceso determinístico, lo que permite escribir:

$$\hat{f} = Q(f) = f + e_Q$$

donde e_Q es el *error de cuantización* dado por :

$$e_Q = \hat{f} - f$$

Los criterios que generalmente se usan para determinar los niveles r_i y d_i , tienen como objetivo minimizar el error utilizado una medida de la distorsión.

Aquellos cuantizadores, en los que dichos niveles se encuentran igualmente espaciados, reciben el nombre de *cuantizadores uniformes*. Específicamente para estos cuantizadores tenemos que:

$$d_i - d_{i-1} = \Delta, \qquad 1 \leq i \leq L$$

$$r_i = \frac{d_i + d_{i+1}}{2} \quad , \qquad 1 \leq i \leq L$$

donde Δ es el *tamaño del paso de cuantización*. La siguiente figura es un ejemplo de un cuantizador uniforme.

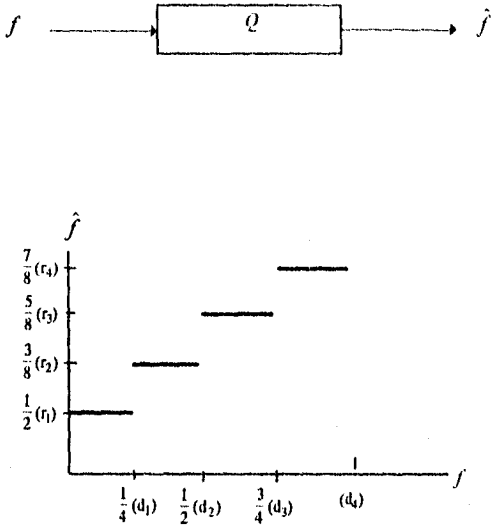


Figura 2.4 Cuantizador uniforme, con 4 niveles de reconstrucción y con valores de f que varían de 0 y 1

A pesar de que la cuantización uniforme se obtiene de forma directa y como una aproximación natural, no siempre es óptima. Existen aplicaciones en las que por las características de la señal que se va ha cuantizar, es necesario que el número de niveles de cuantización sea mayor en algunas regiones que en otras. En otras palabras, se necesita realizar una *cuantización no uniforme*.

El criterio que normalmente se utiliza para determinar los niveles r_i y d_i de un cuantizador no uniforme, es el de la *minimización del error cuadrático promedio* (MSE). El ejemplo de un cuantizador que utiliza una medida de la distorsión basado en el criterio anterior, es el que se conoce con el nombre de *cuantizador Loyd - Max*. Para obtener los niveles de cuantización de este cuantizador, se realiza un proceso iterativo apartir de una función de densidad de probabilidad de entrada y se calculan los niveles para dicha señal con una densidad de probabilidad de amplitud normalmente distribuida [RAB91].

2.8 ENTROPÍA

Este concepto es definido en la compresión de imágenes, como la cantidad promedio de información (bits) a transmitir por cada símbolo. Sabemos que la imagen o la información que se transmite por un canal, se forma por una secuencia de símbolos elegidos de un conjunto finito (alfabeto de la fuente); cuya probabilidad de ocurrencia para cada uno de ellos es diferente dentro de un mensaje.

Para un sistema capaz de transmitir n niveles discretos en intervalos de tiempo λ, el número de combinaciones de diferentes señales que se presentan en T segundos es $n^{T/\lambda}$. Como la cantidad de información es proporcional al tiempo de transmisión, se puede obtener el logaritmo de $n^{T/\lambda}$ para determinar la información transmitida en T segundos como [HEL91]:

$$H = T/\lambda \ \log_2 n \quad \text{[bits]}$$

Por otro lado, a la máxima cantidad de información por segundo que es capaz de transmitir un sistema, se le define como *capacidad* y puede expresarse en bits por segundo:

$$C = H/T \quad [bps]$$

A la frecuencia relativa de ocurrencia de algún evento o combinación, se define como probabilidad de ocurrencia y se puede representar simbólicamente como P, donde:

$$P = \frac{\textit{número de veces que ocurre un evento}}{\textit{número total de posibilidades}}$$

Tomando el caso en el que la probabilidad de ocurrencia de los niveles de una señal o evento no sean iguales, se puede decir que la frecuencia de ocurrencia de cada nivel de la señal será representada por P_i, donde: $P_1 + P_2 + \ldots + P_n = 1$. Entonces, para cada intervalo la información contenida será de $-\log_2 P_i$ bits. Ahora, si sumamos la información en bits promedio por cada símbolo que aparece $t * P_i$ veces sobre un intervalo t, se puede escribir:

$$H = -t * \sum_{i=1}^{n} P_i \log_2 P_i \qquad \textit{bits en t periodos}$$

Para un intervalo T, se puede escribir:

$$H = -\frac{T}{\lambda} * \sum_{i=1}^{n} P_i \log_2 P_i \qquad \textit{bits en T segundos}$$

Para un mensaje con n símbolos posibles y con una probabilidad de ocurrencia de P_i a P_n, la información promedio por símbolo en intervalo de λ segundos es :

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \quad \textit{bits /intervalo del símbolo}$$

Esta última ecuación representa la definición matemática de *Entropía*, un término usado para definir el número promedio de bits requeridos para representar a cada símbolo [HEL91].

El grado de reducción obtenida como resultado del proceso de compresión se conoce como *razón de compresión*; esta razón mide la capacidad de datos comprimida en comparación con los datos originales, de tal manera que:

$$\text{Razón de compresión} = \frac{\text{Longitud de la cadena original de datos}}{\text{Longitud de la cadena comprimida de datos}}$$

De esta ecuación se deduce, que entre más grande sea la razón de compresión más eficiente es la técnica de compresión empleada.

$$\text{Entropía} = \frac{\text{Longitud de la cadena comprimida de datos}}{\text{Longitud de la cadena original de datos}}$$

La entropía es el recíproco de la razón de compresión y siempre debe ser menor que la unidad para que el proceso de compresión sea efectivo [HEL91]. La fracción de reducción de datos es la unidad menos la entropía (1-entropía).

2.9 EVALUACIÓN DE LA CALIDAD EN LAS IMÁGENES

De alguna forma, debemos calificar las características de la imagen en cuanto a la facilidad para poder reconocerla. Existen diferentes formas de evaluar la calidad de la imagen basadas en términos tanto subjetivos como numéricos sin embargo, todos tienen en común el establecer un patrón con el cual calificaremos diferentes aspectos como brillantez, distorsión, etc. No hay que perder de vista que dependiendo del tipo de aplicación que se desea realizar se tomarán en cuenta los patrones antes mencionados. En seguida presentamos algunos métodos para medir la calidad de la imagen.

2.9.1 ESCALAS SUBJETIVAS

Una medición de la calidad en una imagen para este caso, implica definir escalas que nos den una forma más o menos precisa de calificar la imagen.

En esta medición existen dos tipos de escalas: la calidad absoluta y distorsiones percibidas. En las primeras se cuenta con un número determinado de niveles que van desde inaceptable hasta excelente tomando como patrón de comparación otra imagen . En el caso de la segunda, se tiene una escala de niveles, en donde cada uno nos indica que tan perceptibles son las distorsiones en la imagen. En la siguiente tabla 2.1 se muestra un ejemplo de estas escalas subjetivas utilizadas en el área de transmisión de imágenes [PRA79].

Escalas de calidad en imágenes

Escala absoluta
5. Excelente
4. Buena
3. Regular
2. Mala
1. Indeseable
Escala de Distorsiones
1. No perceptibles
2. Muy poco perceptibles
3. Perceptibles, pero con distorsiones muy pequeñas en la imagen
4. Las imperfecciones en la imagen son aceptables
5. Poco aceptables
6. Inaceptables
7. Extremadamente inaceptables.

Tabla 2.1

2.9. 2 MEDICIONES NUMÉRICAS

Este tipo de mediciones se dividen en dos clases: *univariantes y bivariantes*. En las primeras la escala de medición es calibrada, y se evalúa la imagen que nos interesa en contra de una imagen predefinida como estándar. En la segunda se califican las distorsiones existentes de la imagen después de ser procesada con respecto a ésta, antes de realizar el proceso.

Las mediciones univariantes, se basan comúnmente en el espectro de frecuencia espacial de una imagen. En el dominio continuo, la frecuencia espacial de una imagen con una distribución de luminancia espacial $F(x,y)$, se define en términos de la transformada de Fourier de dos dimensiones como:

$$\Im(\omega_x,\omega_y) = \int_{-L}^{L}\int_{-L}^{L} F(x,y)\exp\left[-j\left(\omega_x x+\omega_y y\right)\right]dxdy$$

donde L representa los límites espaciales de la imagen , ω_x y ω_y denotan las frecuencias espaciales.

La transformada discreta de Fourier

$$\Im(u,v) = \frac{1}{(JK)^{1/2}} \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} F(j,k) \exp\left[-2\pi i \left(\frac{uj}{J} + \frac{vk}{K} \right) \right]$$

puede ser utilizada para un arreglo de imagen discreta $F(x,y)$, donde se asume que la imagen esta uniformemente muestreada sobre una rejilla de dimensiones J x K y donde (u,v) representa las componentes de frecuencia espacial.

Las mediciones típicas, incluyen en todas las frecuencias la energía integrada o el valor medio de energía, dichas mediciones han sido de mucha utilidad, cuando se evalúa el funcionamiento de sistemas de transmisión de imágenes con una respuesta lineal. Para el caso de sistemas de codificación con una respuesta no lineal, las mediciones realizadas son frecuentemente erróneas y contradictorias [VEL95].

Las mediciones de calidad bivariantes, se utilizan frecuentemente en aplicaciones de transmisión de imágenes, debido a que proporcionan información referente a la imagen antes y después de ser codificada. La medida bivariante más común es *el error cuadrático promedio* (MSE). El MSE comúnmente es normalizado por la energía o el pico de magnitud de la imagen de entrada. En el dominio discreto el MSE normalizado entre una imagen de entrada $F(j,k)$ y una imagen reconstruida de la codificación $\hat{F}(j,k)$ está definido como :

$$\varepsilon_N = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} \left[F(j,k) - \hat{F}(j,k) \right]^2 / \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} \left[F(j,k) \right]^2$$

El error de amplitud normalizado se define como:

$$\varepsilon_A = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} \left[F(j,k) - \hat{F}(j,k) \right]^2 / JKA^2$$

donde A denota la amplitud máxima de $F(j,k)$ [VEL95].

A pesar de que estos métodos de evaluación son eficaces, por otra parte no en todos los casos nos brindan una forma adecuada de evaluar el desempeño de un codificador. Existen otras técnicas de evaluación, pero aún estas no están libres de cometer errores. Es por esto que aún se realizan investigaciones para poder crear nuevas herramientas, y así medir de mejor forma las distorsiones en una imagen .

CAPÍTULO III

CLASIFICACIÓN DE LAS TÉCNICAS DE COMPRESIÓN

CAPÍTULO III.

CLASIFICACIÓN DE LAS TÉCNICAS DE COMPRESIÓN

Es un punto complicado de generalizar, pues existen diferentes formas de clasificarlas. Algunos autores mencionan que: en la forma como sean manejados los parámetros (Adaptativas o Estáticas) [HEL91], de acuerdo a los resultados de la información (Sin pérdida o con pérdida) [GON93], y según la aplicación (Física o Lógica) [HEL91]. Sin embargo, cada una de ellas caen en dos grupos fundamentales: (compression lossless y compression lossy) *compresión sin pérdidas y compresión con pérdidas* respectivamente.

En la compresión sin pérdidas la imagen recuperada, es numéricamente igual a la imagen original, pixel por pixel. Y en la compresión con pérdidas, la imagen recuperada contiene distorsiones con respecto a la imagen original. Aunque en términos generales, se consigue una mayor compresión, es importante subrayar que las distorsiones que se obtienen a consecuencia de la compresión, pueden ser visibles o no.

En algoritmos de compresión con pérdidas, pueden obtenerse resultados excelentes en los cuales las pérdidas no son visibles bajo condiciones normales de visibilidad. Es decir que el SVH no logra percibir esas distorsiones, por lo que se debe tener cuidado al interpretar dicha imagen.

3.1 CLASIFICACIÓN DE LAS TÉCNICAS DE COMPRESIÓN:

Dependiendo de la aplicación, existen dos grandes grupos dentro de la clasificación de las técnicas de compresión, nos dice [HEL91], que son:

> *La compresión Lógica*: Básicamente se requiere en el diseño de estructuras de datos adecuadas para una cierta aplicación, por ejemplo, la substitución de campos en las bases de datos, etc.

La compresión Física: Su objetivo principal es disminuir efectivamente la cantidad de bits requerida para almacenar o transmitir la información utilizando, para esto la correlación de los datos, la eliminación de redundancia, etc.

Para esta tesis nos enfocaremos a la descripción de las técnicas de *compresión física* de datos. En la **fig. 3.1** [HEI.91], se presenta una división de las técnicas de compresión más utilizadas, en la transmisión y en el almacenamiento de imágenes digitales.

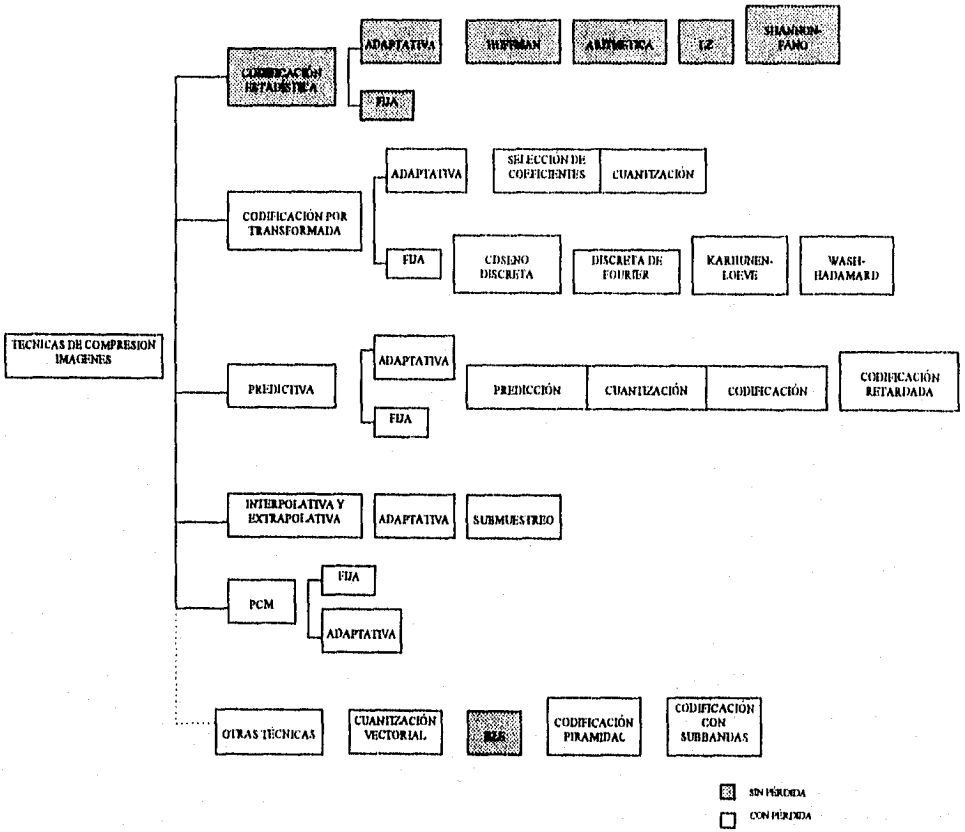


Figura 3.1 Clasificación de los Algoritmos de Compresión de Imágenes.

En base a la manera, en que los datos son codificados se dividen en: codificación estadística, codificación por transformada, codificación predictiva, codificación extrapolativa e interpolativa, y codificación por modulación de pulsos. Estas técnicas pueden ser aplicadas en el almacenamiento y transmisión de imágenes a color, tonos de gris, y binarias. Existe otro grupo más, en las cuales las técnicas de compresión no caen dentro de ninguna de las divisiones mencionadas, pero que se utilizan en la compresión de ciertas imágenes; por ejemplo, los algoritmos RLE (Run Lenght Encoded ó codificación de la longitud corrida) son populares y eficientes en el almacenamiento o transmisión de facsímiles. Así también existen técnicas de compresión fijas y técnicas de compresión adaptativas, teniendo en cuenta que los parámetros sean *fijos* o cambien de acuerdo, al tipo de datos que están codificando (*adaptativos*), respectivamente.

3.1.1 COMPRESIÓN LÓGICA

Este tipo de compresión, es aplicada para diseñar bases de datos. Debido a que se requiere la reducción de información redundante, y la representación de campos con indicadores lógicos tan pequeños como sea posible, dentro de ella. Dependiendo del análisis de los datos, los métodos y la compresión pueden variar. A continuación el ejemplo ilustra esta técnica de compresión, [HEL91].

Supongamos que una escuela imparte 40 materias diferentes. En la base de datos de la escuela, cada boleta debe contener las diferentes materias que cada alumno cursa. Sí tomamos en cuenta, que el campo que denota la materia es fijo y de 30 caracteres, valores tales como "matemáticas" desperdiciarían un total de 19 caracteres por campo. Sí en la escuela hay un total de 1000 alumnos que llevan matemáticas, se estaría desperdiciando el espacio equivalente a 19000 caracteres, sólo en esta materia.

Esto se puede solucionar utilizando un campo numérico, en donde cada uno de éstos representa una materia. Así el total de espacio que se necesitaría, es el que corresponde a 1000 campos numéricos. En este caso hablaríamos de una reducción del 95 % aproximadamente.

3.1.2 COMPRESIÓN FÍSICA

Este tipo de compresión, reduce la cantidad de datos antes de su transmisión o de su almacenamiento y los recupera en el canal receptor o al extraerlos del medio de almacenamiento. Se basa en las probabilidades de ocurrencia de caracteres o grupos de caracteres en la imagen; los caracteres que ocurren más frecuentemente se representan con códigos cortos mientras que los que ocurren raras veces utilizan códigos largos. Algunas técnicas reemplazan las cadenas de caracteres repetidos, con un carácter especial indicador de compresión y un carácter de conteo [HEL91].

Ambos tipos de compresión, lógica y física, tienen un objetivo en común que es el reducir la información, pero hay diferencias entre ellas. Por una parte la compresión lógica se utiliza normalmente para diseñar las bases de datos, no considerando la frecuencia de las ocurrencias de caracteres o grupos de caracteres como lo hace la compresión física.

Al igual que la compresión lógica existen diversas técnicas de compresión física (mostradas en el esquema de la **fig. 3.1** [HEL91]). La mayoría de estas técnicas se discuten en este capítulo; en cuanto a la compresión lógica, se ha presentado como una introducción a la clasificación de las técnicas de compresión.

3.1.3 COMPRESIÓN CON PÉRDIDA Y SIN PÉRDIDA DE INFORMACIÓN

Del hecho de que la imagen original, pueda ser o no recuperada perfectamente apartir de la imagen codificada; podemos clasificar las técnicas de compresión, en técnicas con pérdida de información y técnicas sin pérdida de información. Del diagrama de la **Figura 3.1** [HEL91], los siguientes grupos caen dentro de las técnicas con pérdida de información: codificación por modulación de pulsos, codificación predictiva, codificación por transformada, codificación extrapolativa e interpolativa. El grupo que se encuentra dentro de las técnicas sin pérdida de información, es la codificación estadística y los algoritmos RLE. Los demás métodos de compresión con pérdida de información, se explicarán en las siguientes secciones.

3.1.4 COMPRESIÓN ADAPTATIVA Y COMPRESIÓN ESTÁTICA

Esta subdivisión se basa en que los parámetros sean *fijos* o cambien de acuerdo al tipo de datos que se están codificando (*adaptativos*). La compresión adaptativa la definimos, cuando el diseño de la *tabla de look-up* utilizada en la codificación de la imagen es variable dependiendo de los datos que se van a codificar. Y en la compresión estática o no adaptativa, logre que la imagen sea codificada y recuperada perfectamente utilizando una *tabla de look-up* fija, [HEL91].

Tal es el caso de las técnicas con un paso, conocidas como *estáticas*, en las cuales no pueden adaptarse a los datos inesperados y las que utilizan dos pasos son las llamadas *semi-adaptativas*, las cuales requieren dos pasos sobre el mensaje lo cual no es recomendable para las líneas de comunicación. Estas dos técnicas no son adecuadas para la compresión general de datos. Por ejemplo, las técnicas de compresión RLE y métodos de diccionario en general utilizan estadísticas fijas y efectúan dos pasos en el mensaje, el primer paso es para determinar las estadísticas y la segunda para codificar el mensaje (utilizando las estadísticas previamente determinadas en la tabla de look-up).

Las técnicas *adaptativas* combinan lo mejor de las técnicas estáticas y semi-adaptativas efectuando un solo paso sobre el mensaje y adaptándose de acuerdo a éste. En cada paso la siguiente parte del mensaje se almacena utilizando un código construido de la compresión en la tabla de look-up; esto es posible debido a que tanto el codificador como el decodificador tienen acceso a dicha tabla y pueden construir independientemente el código usado, para almacenar la siguiente pieza del mensaje. La ventaja de adaptación durante un solo paso podría ser considerada razón suficiente, para no considerar a las técnicas estáticas y semi-adaptativa.

Una técnica estática siempre logrará mejor compresión, que una técnica adaptativa en aquellos datos por los cuales se ha mejorado la técnica estática; esto se debe a que la técnica adaptativa requiere de tiempo para reconocer lo que la técnica estática ya conoce de antemano, sin embargo, si la técnica estática utiliza datos inadecuados, producirá resultados ineficientes. Por otro lado, se ha demostrado que una técnica adaptativa siempre es mejor que una técnica semiadaptativa, o en el peor de los casos es igual [HEL91].

3.2 TÉCNICAS CON PÉRDIDA DE INFORMACIÓN.

Las técnicas con pérdida de información, se definen en una imagen cuando la intensidad original de los pixeles no se puede recuperar perfectamente, y su objetivo es minimizar la distorsión promedio para una razón de bits dada. Los rangos de compresión varían entre 4:1 a 32:1. Aunque estas técnicas se utilizan principalmente para conversiones analógicas-digitales de imágenes y transmisión de las mismas, muchos de sus principios y conceptos han servido de base, para el desarrollo de otras técnicas de compresión aplicadas en el almacenamiento de imágenes.

3.2.1 CUANTIZACIÓN DE UNA IMAGEN MONOCROMÁTICA (PCM)

La cuantización de una imagen monocromática, se define como la digitalización de la imagen o bien la *conversión analógica-digital* de la misma. En la cual se genera una representación discreta en amplitud y tiempo de los pixeles, sin eliminar la redundancia estadística o perceptual de la señal. Muestreando la señal a la *razón de Nyquist*, se origina el tiempo discreto; mientras que la amplitud discreta se crea utilizando un número grande de niveles de cuantización, de manera que la distorsión debida a los errores de cuantización sea tolerable. A este proceso se le conoce como codificación PCM (*Codificación por Modulación de Pulsos*). Se ha utilizado en la digitalización de vídeo para su transmisión o almacenamiento, o como técnica básica antes de aplicar otras técnicas de codificación .

El PCM básico se muestra en la **figura 3.2,** [SCH93], el cual consta de tres partes muy importantes para su proceso. La primera parte es el muestreo (usualmente a la razón de Nyquist) de una señal. En la segunda parte es cuantizada cada muestra utilizando 2^k niveles. En caso de que se generen distorsiones del efecto escalera (*aliasing*), se utiliza un prefiltro adecuado antes del muestro con una razón muy cercana a Nyquist. Y en la última parte se realiza la asignación de códigos binarios, en donde cada nivel representa una palabra binaria de k bits; usualmente la palabra binaria está relacionada con el nivel, de manera que las operaciones aritméticas sean más fáciles.

En el decodificador, estas palabras binarias se convierten en una secuencia de las amplitudes discretas de los niveles, los cuales se hacen pasar por un filtro pasobajas. El PCM básico es simple sin embargo, es ineficiente debido a que no utiliza la redundancia de información que se presenta.

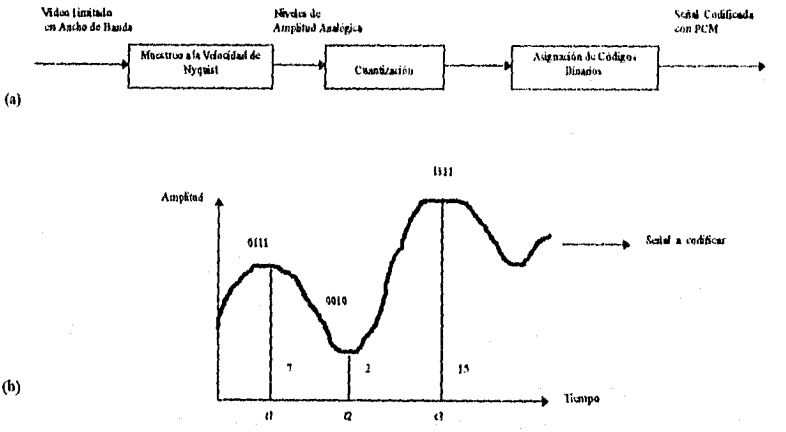


Figura 3.2 Codificación PCM, (a) Componente de un Codificador PCM, (b) Representación binaria en 4 bits para niveles de amplitud entre 0 y 15.

Se busca reducir el error en la cuantización y de que ése error no sea visible en la imagen debido a la separación de los niveles de cuantización al ser seleccionados. Existen varias técnicas para reducirlo. Una técnica es utilizar pre y post filtros (óptimos de acuerdo al criterio del *error cuadrático promedio),* en el cuantizador mostrado en la figura 3.2 [SCH93], haciendo que el cuantizador se comporte como una fuente de ruido aditivo.

El resultado de estos filtros da imágenes libres de contornos artificiales; sin embargo, estos filtros reducen también la resolución de las imágenes reproducidas. Una de las técnicas utilizadas para el filtrado es el *dither*, con esta técnica se añade ruido pseudoaleatorio a la imagen antes de su cuantización, y después el decodificador substrae el mismo ruido de la imagen cuantizada; estas técnicas de *dither* también se aplican en la conversión de imágenes en tonos de grises a blanco y negro [SOR95].

3.2.2 CODIFICACIÓN PREDICTIVA (DPCM)

La codificación predictiva, tiene la finalidad de remover la redundancia mutua entre pixeles muy cercanos, extrayendo y codificando sólo la *nueva información* de cada pixel. Tal información será la diferencia entre el valor actual y el valor previo transmitido de ese pixel, la cual se cuantiza en un conjunto de L niveles de amplitud discreta; y cada nivel se representa con palabras binarias de longitud fija o variable, las cuales se envían por el canal de transmisión, [SCH93].

La codificación predictiva es también conocida como *codificación con modulación de pulsos diferenciales*. Contiene 3 componentes básicos (como se muestra en la siguiente figura [SCH93]):

1. *Predictor.*
2. *Cuantizador,*
3. *Codificador.*

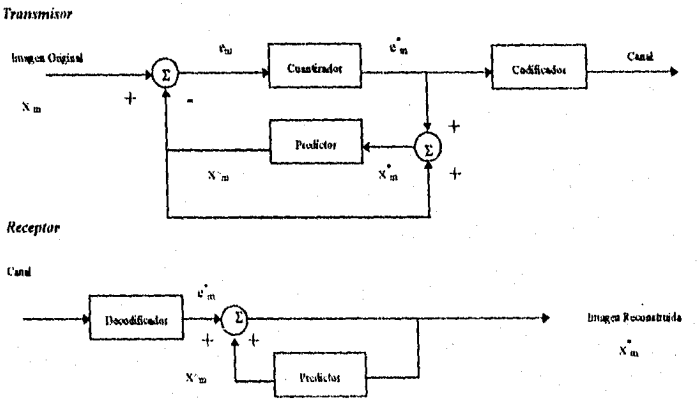


Figura 3.3 Diagrama de Bloques de la Codificación DPCM

La codificación predictiva más sencilla, es la modulación delta (DM *Delta Modulation*), se utiliza una función de retraso de un paso como predictor y la diferencia entre esta predicción y la señal se cuantiza en un bit. Puesto que utiliza dos niveles de cuantización, para obtener una calidad en la imagen es necesario muestrear a una velocidad mucho más alta que la *razón de Nyquist,* lo cual reduce la tasa de compresión. Aunque la modulación delta se ha utilizado efectivamente en la transmisión de voz, no ha tenido gran uso en la codificación de imágenes, debido quizá, a la gran velocidad de muestreo requerida; por otro lado, los codificadores DPCM si han tenido mucha utilidad en la codificación de imágenes, (para más en detalle, de los codificadores DPCM, [SCH93]).

Esta técnica se puede hacer adaptativa cambiando los parámetros de la predicción, de acuerdo con las estadísticas de la imagen, variando los niveles de cuantización con base a criterios perceptibles, o no transmitiendo el error cuando este se encuentre debajo de un valor límite establecido.

Otra posibilidad es retardar la codificación de un pixel hasta que sea posible observar la tendencia futura de la señal y de esta manera utilizar dicha tendencia para la predicción del pixel actual (ver la **figura 3.4** [SCH93]), se presenta el esquema general de la codificación DPCM, cuando se requiere ser retardada; en ella, los *N-1* pixeles previamente transmitidos se utilizan para calcular la predicción $\hat{b}_N$ del pixel b_N, tanto en el codificador como en el decodificador.

La señal diferencial $b_N - \hat{b}_N$ se codifica usualmente con una palabra binaria de longitud variable.

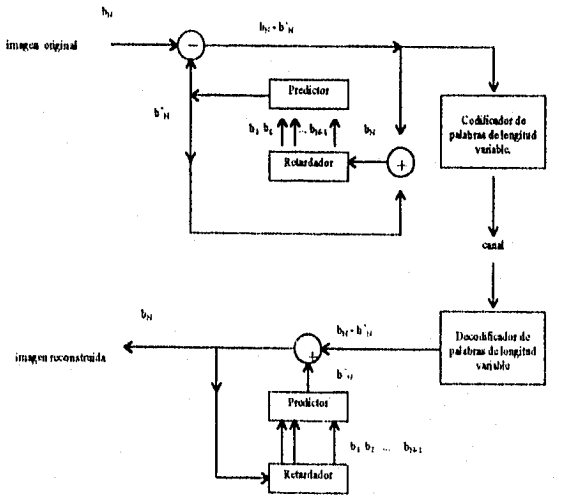


Figura 3.4 Esquema General de la Codificación DPCM retardada.

En cuanto a la codificación DPCM del color y la creación de mapas de colores, se aplican las mismas técnicas descritas para la codificación PCM.

Debido a que gran parte de la redundancia entre pixeles es eliminada por los procesos de predicción y diferenciación, la distribución de probabilidad de los niveles de cuantización, no es uniforme; esto da la representación de éstos, utilizando palabras binarias de longitud variable (por ejemplo, los códigos de Huffman, los cuales los veremos más adelante) en vez de usar longitudes fijas. La razón promedio de bits, de tales códigos es usualmente cercana a la entropía de la señal de salida del cuantizador.

Por ejemplo un código típico de longitud variable para un codificador DPCM con 16 niveles, se muestra en la **figura 3.5** [SCH93] , los niveles más internos que ocurren más frecuentemente se representan con códigos más cortos.

Nivel No.	Longitud del Código	Código
1	12	100101010101
2	10	1001010100
3	8	10010100
4	6	100100
5	4	1000
6	4	1111
7	3	110
8	2	01
9	2	00
10	3	101
11	4	1110
12	5	10012
13	7	1001011
14	9	100101011
15	11	10010101011
16	12	100101010100

Figura 3.5 Códigos típicos de longitud variable para una señal codificada con DPCM, con 16 niveles de cuantización.

3. 2. 3 CODIFICACIÓN POR TRANSFORMADA.

La codificación por transformada es también llamada de bloques, la cual divide generalmente una imagen de tamaño N x N, en bloques de M x M, donde M<N y es normalmente una potencia de 2. Se aplica una transformación lineal a cada bloque de valores (pixeles), estadísticamente dependientes, transformándolos en otro bloque de datos, llamados *coeficientes*, "más independientes". Finalmente, cuantizando y codificando los coeficientes seleccionados, para la transmisión (como se muestra en la **figura 3.6** [EMB90]).

Para obtener un buen desempeño en la codificación por transformada, es necesario tener en cuenta lo siguiente:

- *Tamaño de los bloques en la imagen*
- *Tipo de transformada que se desea aplicar según la imagen*
- *Selección de los coeficientes a ser cuantizados.*
- *Asignación de bits.*

Cada uno de estos puntos ayudan a que la imagen sea recuperada eficientemente.

La compresión se lleva a cabo, sólo cuando se seleccionan los coeficientes de máxima energía para efectuar su cuantización y su codificación, por lo que es necesario contar con una transformada que compacte la máxima energía de la imagen en el menor número de coeficientes que sea posible y que además sea fácil de implementar. Una transformada que cumple con estas necesidades es la transformada llamada Karhumen - Loeve sin embargo, es difícil de calcular puesto que depende de la estadística de la imagen.

Por tal motivo se utilizan otras transformadas que redistribuyen la energía de los pixeles en un número pequeño de coeficientes y además son fáciles de implementar. Entre ellas se encuentran: la transformada de Fourier, Coseno, Hadamard. Por medio de esta técnica se alcanzan razones de compresión de 10:1. Y cuando es adaptativa puede ser incrementada la eficiencia de un 25 a un 30 %, cambiando la transformación, para ajustarse a las estadísticas de una imagen en particular o cambiando el criterio de selección y cuantización de los coeficientes, para ajustarse a criterios cualitativos y perceptibles.

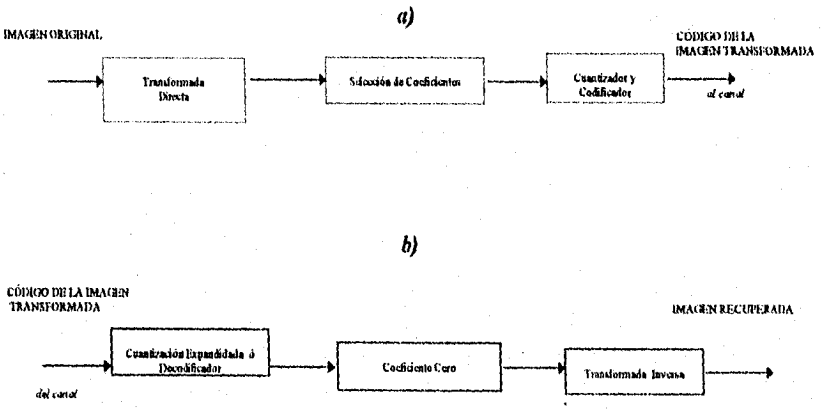


Fig. 3.6 Sistema de Codificación por transformaciones de bloques.

En imágenes monocromáticas utilizando alguna de las técnicas de compresión por transformada se han logrado tasas de bit de 1.0 a 1.5 bits/pixel con un error cuadrático promedio menor del 0.5% [PRA79].

Con transformaciones lineales, cada bloque de pixeles a transformarse se arregla en un vector columna b de longitud N, al cual se aplica una transformada lineal ortonormal (también llamada *unitaria*).

$$c = T\,b$$
$$b = T^{t}\,c$$

donde T es una matriz de transformación de tamaño NxN, c es el vector columna de los coeficientes de transformación, y el apóstrofe indica conjugada transpuesta.

El vector b puede construirse de N pixeles sucesivos, en una línea *raster* y en tal caso la codificación con transformaciones explota la correlación unidimensional, entre los pixeles de una misma línea. Así también se puede explotar la correlación horizontal y vertical en la imagen, construyendo el vector b, considerando un arreglo bidimensional de LxL pixeles de la imagen de tal manera que se construya un vector de longitud $N = L$ [RAO90].

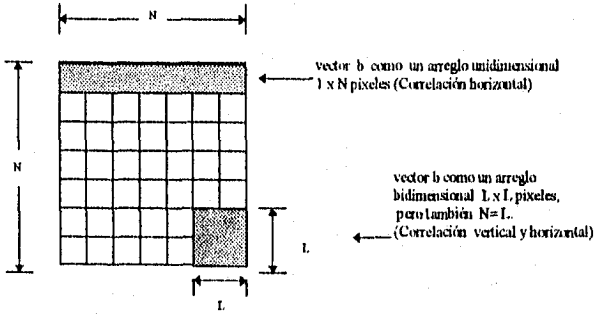


Figura 3.7 Diferentes tipos de correlación dentro de una imagen de (N x N) pixeles.

Otra posibilidad es denotar un arreglo de LxL pixeles con la matriz cuadrada $B = [b_{ij}]$ y separar la transformación en dos pasos; el primer paso es transformar los renglones de B, con longitud L para aprovechar la correlación horizontal; después se transforman las columnas de B para explotar la correlación vertical.

La operación combinada puede escribirse como:

$$c_{mn} = \sum_{i=1}^{L}\sum_{j=1}^{L} b_{ij} t_{ni} t_{mi}$$

donde las *t's* son los elementos de la matriz de transformación T, y $C = [c_{mn}]$ es la matriz de tamaño LxL de coeficientes de transformación.

En la **figura 3.8** se muestra el diagrama general de la codificación con transformaciones de bloques [EMB90].

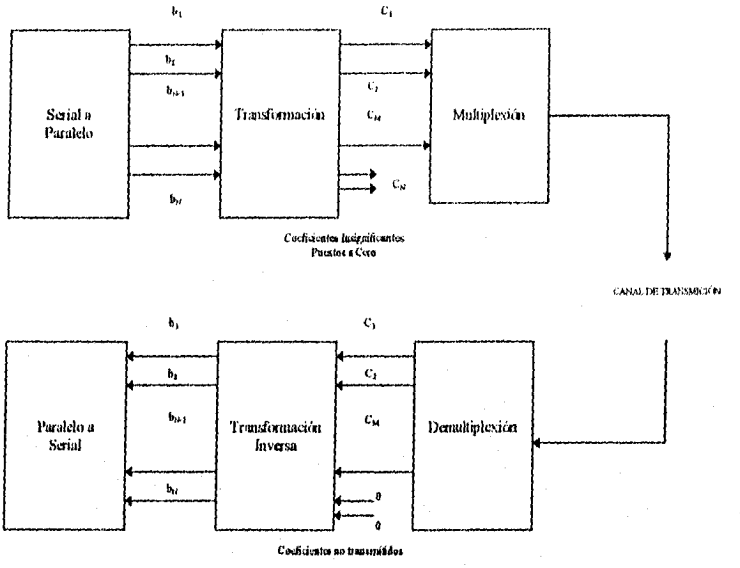


Fig. 3.8 Codificación con Transformaciones. Cada bloque de N pixeles se transforma (usualmente con una matriz lineal ortonormal) en un bloque de coeficientes de transformación y los coeficientes insignificantes se eliminan. Los restantes $M=pN$ coeficientes se codifican y transmiten al receptor, en el cual se efectúa la operación inversa.

En la matriz de transformación T, se han utilizado las principales transformadas como son: la transformada discreta de Fourier, la transformada de Walsh-Hadamard, la transformada de Karhumen-Loeve y la transformada Coseno Discreta, las cuales se explicarán a continuación.

3.2.3.1 TRANSFORMADA DISCRETA DE FOURIER.

La matriz unitaria T para la transformada discreta de Fourier (DFT *Discrete Fourier Transform*) tienen los coeficientes:

$$t_{mi} = \frac{1}{\sqrt{N}} \exp\left[-\frac{2\pi}{N}\sqrt{-1}(i-1)(m-1)\right] \quad i, m = 1...N$$

Para $N=8$ la matriz T se muestra en la **figura 3.9,** [GON93]; puesto que en este caso el vector b es real, los coeficientes complejos de c tienen una simetría conjugada, es decir:

$$c_{2+p} = c'_{N-p}; \qquad 0 \le p < N-2$$

De esta manera, la reconstrucción de los N pixeles de b solamente requiere la transformación de N valores reales.

Parte Real

1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.000	0.707	0.000	-0.707	-1.000	-0.707	0.000	0.707
1.000	0.000	-1.000	0.000	1.000	0.000	-1.000	0.000
1.000	-0.707	0.000	0.707	-1.000	0.707	0.000	-0.707
1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000
1.000	-0.707	0.000	0.707	-1.000	0.707	0.000	-0.707
1.000	0.000	-1.000	0.000	1.000	0.000	-1.000	0.000
1.000	0.707	0.000	-0.707	-1.000	-0.707	0.000	0.707

Parte Imaginaria

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.707	-1.000	-0.707	0.000	0.707	1.000	0.707
0.000	-1.000	0.000	1.000	0.000	-1.000	0.000	1.000
0.000	-0.707	1.000	-0.707	0.000	0.707	-1.000	0.707
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.707	-1.000	0.707	0.000	-0.707	1.000	-0.707
0.000	1.000	0.000	-1.000	0.000	1.000	0.000	-1.000
0.000	0.707	1.000	0.707	0.000	-0.707	-1.000	-0.707

Figura 3.9 Matriz para la transformación discreta de fourier con $N=8$.

Una ventaja de la DFT es el algoritmo, llamado *Transformada rápida de Fourier* (FFT *Fast Fourier Transform*). Mientras que una multiplicación directa de la matriz DFT, requiere aproximadamente N^2 sumas y multiplicaciones complejas, la FFT requiere cerca de $N \log_2 N$ operaciones si N es potencia de 2 [GON93].

Otra ventaja es la reducción de los requerimientos de almacenamiento y errores pequeños de redondeo.

3. 2. 3. 2. TRANSFORMADA WALSH-HADAMARD

La transformada Walsh-Hadamard (WHT) se describe más fácilmente con recursividad. Sea $H_1 = 1$, y para $N = 2^n$ se define [RAB91]:

$$H_{2N}\begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

Entonces, la matriz de transformación WHT unitaria simétrica está dada por:

$$T = \frac{1}{\sqrt{N}} H_N = T'$$

Para $N = 4$ y $N = 8$ las matrices T se muestran en la **figura 3.10,** [RAB91]. La principal ventaja de la WHT es que aparte del factor $(1/\sqrt{N})$, el cálculo requiere de sumas y restas, en cambio otras transformaciones requieren además de multiplicaciones. También existe un algoritmo rápido que requiere aproximadamente $N \log_2 N$ operaciones [RAB91].

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Figura 3.10 Matrices para la transformada de Walsh-Hadamard con N=4 y N=8.

3.2.3.3 TRANSFORMADA KARHUNEN-LOEVE.

La transformada Karhunen-Loeve (KLT), es una transformada ortonormal que elimina la correlación estadística (no la dependencia estadística) de bloques de pixeles. Es decir, los coeficientes de transformación de la transformada KTL satisfacen (para $m \neq n$) :

$$\sum c_m c_n p(c_m, c_n) = \sum c_m p(c_m) \bullet \sum c_n p(c_n)$$

$$c_m c_n \qquad c_m \qquad c_n$$

donde la sumatoria se efectúa sobre todos los valores posibles. Esto se escribe utilizando el operador del promedio estadístico E, como:

$$E(c_m\ c_n\ ) = Ec_m\ \ *\ \ Ec_n\ \ ;\ m \neq n$$

o bien como:

$$E[\ (c_m - E\ c_m\ )\ (c_m\ -\ Ec_n\ )\ ] = \lambda_m\ \ \delta_{mn}\ \ ;\qquad \forall\ m,\ n$$

donde λ_m es la varianza de c_m. Se debe notar que la independencia estadística implica no correlación, pero lo inverso no es generalmente cierto.

La KLT se deriva ajustando primero los vectores de pixeles b de tal forma que tengan una media igual a cero, es decir:

$$Eb = 0, \quad \text{y por lo tanto} \quad Ec = 0.$$

La matriz NxN de correlación para los pixeles está dada por:

$$R = E(b\ b') = E\ (T'c\ c'T) = T'E(cc')T$$

y tomando en cuenta la condición de ortonormalidad:

$$RT' = T'E(cc')$$

$E(cc')$ es una matriz diagonal de NxN con varianzas c_m en la diagonal principal; de esta manera, en términos de vectores columna de T', la ecuación anterior se convierte en :

$$Rt_m\ = \lambda_m\ t_m\ ; \qquad m = 1,...,N$$

Además de la ventaja de eliminar la correlación de los coeficientes, la transformada KLT tiene la propiedad de maximizar el número de coeficientes, que son muy pequeños para no tomarlos en cuenta; estadísticamente, esto significa que la KLT minimiza el error cuadrático promedio [RAO90].

3. 2. 3. 4 TRANSFORMADA COSENO DISCRETA.

Probablemente, la más común en compresión de imágenes, es la Transformada Coseno Discreta, muchas transformadas, han sido utilizadas en la compresión y su funcionamiento puede ser comparado, por su fidelidad al recuperar la imagen original [GON93].

El resultado de la imagen muestreada al transformarla, es un número de coeficientes seleccionados del total, y los coeficientes restantes, son recuantizados con menos bits que la original. Para la recuperación de la imagen, el proceso de funcionamiento es contrario. Los coeficientes, son recuantizados al número original de bits, los coeficientes pérdidos, son reemplazados por valores fijos (por ejemplo, 0) y la transformada inversa es ejecutada.

La fidelidad del proceso, puede ser medida tomando la diferencia en los niveles de intensidad entre la imagen original y la recuperada en cada punto del arreglo [EMB90]. Si la original es f_org[m,n] y la recuperada es f_rec[m,n] (ambas matrices de NxN), entonces el arreglo de la diferencia es

$$f\_diff[m,n] = f\_rec[m,n] - f\_org[m,n].$$

El error cuadrático promedio, es una medida de fidelidad, que puede ser derivada del arreglo de diferencias [EMB90]:

$$MSE = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f\_diff[m,n]^2$$

Por que la Transformada Coseno Discreta (DCT), da un resultado MSE cercano al límite teórico de la transformada de Karhuenen-Loeve [ROS82], la DCT es muy popular, para la compresión de imágenes.

La definición de la DCT para una imagen N x N, [EMB90], es

$$F[u,v] = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f[m,n]\cos\left[\frac{(2m+1)u\pi}{2N}\right]\cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

Donde:

u,v = variables de frecuencia discreta (0,1,2,....N-1)
f[m,n] = imagen de NxN pixeles (0,1,2,...N-1)
F[u,v] = transformada DCT resultante

La transformada inversa es definida como, [EMB90]:

$$\hat{f}[m,n] = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c[u]c[v]F[u,v]\cos\left[\frac{(2m+1)u\pi}{2N}\right]\cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

donde:

m,n = índices (pixeles) de la imagen resultante (0,1,2,...N-1)
F[u,v] = NxN DCT resultante que va a ser antitransformada

$$c[\lambda] = \begin{cases} 1, & \text{para } \lambda = 0 \\ 2, & \text{para } \lambda = 1,2,3,\ldots N-1 \end{cases}$$

$\hat{f}[m,n] = N \times N$ DCT inversa resultante.

Coeficientes de Cuantización en la Imagen Comprimida.

El propósito del funcionamiento de la transformada Coseno Discreta, es desarrollar una serie de números, los cuales representan a la imagen, pero cuyos valores no son correlacionados (por ejemplo, cada número en el arreglo da información nueva, no dada por otros números). Desde entonces, la misma información contenida, es para ser representada en los arreglos originales y transformados, algunos arreglos transformados, da poca o ninguna información acerca de la imagen original y pueden ser descartados.

La necesidad de un coeficiente dado depende de la varianza sobre una serie de imágenes. Si un coeficiente (por ejemplo, el coeficiente del renglón 45, columna 29 en una imagen transformada de 256 x 256), mantiene el mismo valor sobre una serie de imágenes, entonces este, no nos da mucha información, acerca de la diferencia en las imágenes de esta serie. Por lo tanto, este puede ser reemplazado con una constante en la compresión, sin con esto dañar la fidelidad. Por el contrario, si un coeficiente tiene una alta varianza sobre la serie, no puede ser descartado, ya que se tendrían serias consecuencias al recuperar la imagen.

En la DCT, variauzas sobre series de imágenes típicas, tienden a tener contornos de varianzas constantes, como muestra la fig. 3.11, [EMB90]. Los coeficientes de alta varianza, tienden a estar cerca del origen, y a los ejes "u" y "v". Una forma de usar estas características, es asignar un número de niveles de cuantización, basados en la varianza del coeficiente.

A esto se le llama "*Codificación por zona*" [PRA79] y una asignación típica, en una imagen transformada de 16 x 16, se muestra en la **fig. 3.12,** [EMB90]. Esta asignación reduce el número de bits requerido, para representar la imagen de bits por pixel, a 1.5 bits por pixel.

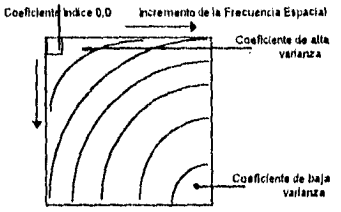


Fig. 3.11 Líneas de Coeficientes de Varianza Constante en una Imagen Transformada Típica.

Codificación por Bloque.

El número de cálculos para la DFT, es de orden de N^2, donde N, es la longitud de la transformada. La transformada FFT reduce este número a $N \log_2 (N)$. Sin embargo, crece rápidamente el número N, conforme aumenta la longitud de la transformada. La FFT reduce este número de orden, a $N \log_2 (N)$. La Transformada Coseno Discreta, tiene un tiempo de ejecución del orden de N^4 , donde N es la longitud del arreglo de un lado de la imagen.

Las técnicas de la transformada rápida (FFTs), puede reducir este, a un orden de $[N \log_2 (N)]^2$. En cualquier caso, como la longitud de uno de los lados de la imagen incrementa, el tiempo requerido para la DCT se incrementa más rápidamente. En la codificación por bloque podemos sacar ventaja de esto. Esta técnica divide una imagen en varias subimágenes. A cada subimagen, se le aplica una transformación y los coeficientes son cuantizados, como si cada uno fuera una imagen separada.

8 8 8 7 7 7 5 5 4 4 4 4 4 4 4 4
8 8 7 6 5 5 3 3 3 3 3 2 2 2 2 2
8 7 6 4 4 4 3 3 2 2 2 2 2 2 2 2
7 6 4 3 2 2 2 2 1 1 1 1 0 0 0 0
7 5 4 2 2 2 2 1 1 1 0 0 0 0 0 0
7 5 4 2 2 2 1 1 0 0 0 0 0 0 0 0
5 3 3 2 2 1 1 0 0 0 0 0 0 0 0 0
5 3 3 2 1 1 0 0 0 0 0 0 0 0 0 0
4 3 2 1 1 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 1 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0
4 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
4 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
4 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
4 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0

Fig. 3.12 a) Asignación de Bits a los coeficientes para un bloque de 16 x 16 con un compresión de 8 a 1.5 bpp.

8 8 4 4 4 4 4 4
8 8 4 4 4 4 4 0
4 4 4 4 0 0 0 0
4 4 4 4 0 0 0 0
4 4 0 0 0 0 0 0
4 4 0 0 0 0 0 0
4 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0

Fig. 3.12 b) Asignación de Bits a los coeficientes para un bloque de 8 x 8, con una compresión de 4 a 1 bpp.

Sin embargo, existe un costo en cuanto a la rapidez de computación. Cuando el tamaño del bloque es demasiado grande, la distorsión debido a la eliminación de coeficientes, se extiende en un espacio amplio de la imagen. Como el tamaño del bloque se reduce, la distorsión llega a ser desigual entre bloques y los límites de cada bloque, mostrándose en la imagen recuperada.

El sistema de visión humano, es más sensible a patrones regulares de interferencia, como en el caso, en que los bordes del bloques (interferencia dispersa). Como resultado, la calidad de la imagen percibida es menor. Además, porque la correlación entre valores de pixel, es más grande para pixeles los cuales son cercanos en una imagen, se pierde algo de la independencia de los coeficientes. Así la medidas cuantitativas de la fidelidad de la imagen, también se distorsionan.

Tamaño del bloque	Factor del Tiempo en Transformarla	Número requerido	Total	Factor de velocidad
64 X 64	17 X 10^6	16	3 X 10^8	16
32 X 32	1 X 10^6	64	64 X 10^6	64
16 X 16	65536	256	16 X 10^6	256
8 X 8	4096	1024	4 X 10^6	1024
4 X 4	256	4096	1 X 10^6	4096

TABLA 3.1 . Comparación de tiempo en transformar una imagen completa (tamaño de 256 X 256)

Para determinar, el tamaño óptimo del bloque, debemos conocer la correlación, dentro de cada bloque de pixeles adyacentes en la imagen original. En muchas de las imágenes típicas, la correlación significativa existe solo alrededor de 20 pixeles adyacentes [GON87]. Por lo tanto, como el tamaño del bloque, se incrementa alrededor de 16 x 16, la fidelidad incrementada es menor y menos significativa. Frecuentemente los bloques más grandes de 16 x 16 no son garantizados. Ver la **tabla 3.1**, donde se muestra un ejemplo de una imagen de 256 x 256, utilizando diferentes tamaños de bloques, y el tiempo que se requiere para calcular la transformada Coseno Discreta.

El uso de la DCT se ha incrementado en los últimos años debido a que bajo ciertas circunstancias su desempeño es muy cercano al de la KLT, y además al igual que la transformada discreta de Fourier, existe un algoritmo rápido para el cálculo de la DCT.

3. 2. 4 CODIFICACIÓN EXTRAPOLATIVA E INTERPOLATIVA.

La finalidad de este tipo de codificación es elegir un subconjunto de pixeles para su transmisión, por medio de cualquiera de las técnicas descritas previamente (PCM, DPCM, transformación de bloques, etc.); los pixeles que no son transmitidos, se reproducen en el receptor por medio de interpolación o extrapolación utilizando la información de los pixeles transmitidos.

En la codificación interpolativa, se interpolan los pixeles que no son transmitidos, mientras que en la codificación extrapolativa, los pixeles a transmitir se extrapolan de los pixeles anteriores, uno por uno; el proceso de extrapolación termina cuando se encuentra un pixel cuyo error de extrapolación sobrepasa un valor de umbral predefinido, después de lo cual dicho pixel se transmite y los siguientes pixeles son extrapolados otra vez como se hizo anteriormente.

Un ejemplo de codificación interpolativa se muestra en la **figura 3.13,** [TIM90].

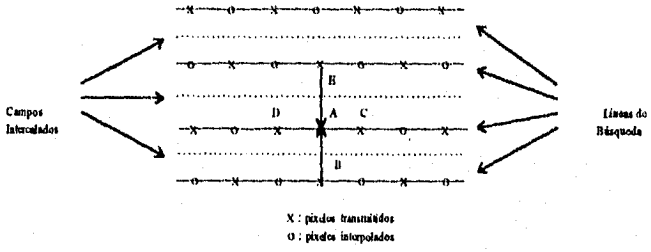


Figura 3.13 Ejemplo de codificación interpolativa utilizando un muestreo 2:1, escalonado de una línea a la siguiente.

Al adaptar este tipo de codificación se requiere de: cambiar el criterio de selección de las muestras que se transmitirán, así también la estrategia de extrapolación e interpolación de las muestras que aún no se han transmitido.

3. 2. 5 OTRAS TÉCNICAS DE COMPRESIÓN.

Existen otras técnicas de compresión de imágenes con pérdida de información, aparte de las ya mencionadas. Se explicarán brevemente cada una de estas técnicas mencionando sus características más importantes.

3.2.5.1 CODIFICACIÓN EN MULTIRRESOLUCIÓN.

La codificación en Multirresolución tiene como finalidad codificar una imagen a diferentes intensidades y resoluciones espaciales, usualmente en una forma anidada. Este proceso, es también conocido como *codificación piramidal*, el cual genera una representación piramidal laplaciana de la imagen por medio de muestreo e interpolación, y arreglando los residuos en capas repetidas.

Codificación Piramidal.

Se representa como una serie de imágenes pasa-bandas, cada una muestreada con razones de bits sucesivamente menores, el ejemplo más conocido, son los códigos piramidales de *Burt y Adelson*. Una implementación de esto, es construir primero, una secuencia de imágenes con filtros paso-bajas, $\{b_k(x_j, y_i)\}$, $k = 1...n$, tal que, [RAB91]:

$$b_{k+1}(x_j, y_i) = \sum_{m=-p}^{+p} \sum_{n=-p}^{+p} h(m,n) b_k(x_{2j+m}, y_{2i+m})$$

donde las funciones $h(m,n)$, se escogen para ser aproximadamente igual a una función Gaussiana, y $b_0(x_j, y_i)$ es la imagen original. Se calcula $h(m,n)$ en forma separada como:

$$h(m,n) = h(m)\ h(n)$$

Para p = 2, las funciones, que se toman son $h(0) = 0.4$, $h(1) = h(-1) = 0.25$ y $h(2) = h(-2) = 0.05$. En este caso $h(m,n)$, es un producto cartesiano de dos funciones triangulares idénticas y simétricas; la imagen $b_{k+1}(x_j, y_i)$ es una versión filtrada de $b_k(x_j, y_i)$, de esta manera:

$$L_k(x_j, y_i) = b_k(x_j, y_i) - b_{k+1,e}(x_j, y_i)$$

donde los pixeles interpolados de la versión expandida $b_{k+1,e}(x_j, y_i)$ están dados por:

$$b_{k+1,e}(x_j, y_i) = 4 \sum_{m=-2}^{+2} \sum_{n=-2}^{+2} h(m,n) b_{k+1}(x_{(j-n)/2}, y_{(i-m)/2})$$

En la ecuación anterior, solamente se incluyen en la sumatoria los enteros de x y y.

De esta manera, se han creado una serie $L_k(x_j , y_i)$ de imágenes filtradas con pasa-bajas; si cada una de estas imágenes se colocan unas sobre otras, el resultado es una estructura de datos piramidal. La compresión de la imagen se efectúa cuantizando $b_n(x_j , y_i)$ seguido de $L_k(x_j , y_i)$ y después codificar la salida cuantizada; de hecho, la imagen $b_n(x_j , y_i)$ puede consistir solamente de un pixel. En el receptor, se lleva a cabo la operación inversa [RAB91].

La principal ventaja de esta técnica es que los cálculos son simples, y pueden ser efectuados en paralelo; además, los mismos cálculos son iterados para construir la secuencia, que constituye a la pirámide.

Codificación con Sub-bandas.

En el procedimiento descrito en la codificación piramidal, las $L_k(x_j , y_i)$ pueden calcularse como la salida de n filtros pasa-bandas paralelos, en vez de obtenerse iterativamente; en este caso el algoritmo de compresión es conocido como *Codificación con Sub-bandas*. Al igual, que en la codificación piramidal, las $L_k(x_j , y_i)$ se cuantizan y codifican separadamente, quizás utilizando una codificación RLE para las cadenas con valor de cero; la imagen decodificada se obtiene simplemente añadiendo las imágenes pasa-bandas recibidas en cada paso de la transmisión.

La codificación con sub-banda, consiste en una convolución de la señal de entrada con un conjunto de filtros pasa-banda y decimando los resultados. Cada una de las señales sub-banda se situa en una porción particular del espectro de frecuencia, que corresponde a la información que se esta presentando en una escala espacial particular. Para la reconstrucción de la señal, las señales sub-banda son sobremuestreadas, filtradas y posteriormente combinadas aditivamente. Para propósitos de codificación, las transformadas sub-banda pueden ser utilizadas para controlar el error de cantidades relativas en diferentes partes del espectro de frecuencia [WOO91].

La mayoría de los filtros diseñados para codificadores sub-banda tratan de minimizar el *"aliasing"*, que se da como resultado del proceso de submuestreo. En el dominio espacial, el *"aliasing"* aparece en la imagen de salida como evidencia de la estructura de muestreo. En un sistema de sub-banda ideal se incorporan filtros pasabanda que evitan que se presente el "aliasing". Sin embargo, los filtros producen resonancia (fenómeno de Gibbs) en el dominio espacial lo cual es indeseable perceptualmente, [VEL95].

3. 2. 5. 2 CODIFICACIÓN CON TRUNCAMIENTO DE BLOQUES.

La codificación con truncamiento de bloques, divide la imagen de $N \times N$ en bloques de LxL pixeles (casi siempre de 4 pixeles), después cuantiza los pixeles de cada bloque conservando el promedio (media) y la varianza; la complejidad del cuantizador dependerá del número de niveles de cuantización. El promedio y la varianza del bloque se calcula con las siguientes ecuaciones, [SHI89]:

$$\hat{b} = \left(\frac{1}{L^2}\right)\sum_{i,j=1}^{L} b_{ij} \qquad \textit{promedio o media}$$

$$\sigma^2 = \left(\frac{1}{L^2}\right)\sum_{i,j=1}^{L}\left(b_{ij} - \hat{b}\right)^2 \qquad \textit{varianza}$$

donde la media se define como el promedio de brillantez del bloque; mientras que la varianza representa la actividad de la textura, ambos son cuantizados y transmitidos. Después, los pixeles del bloque se cuantizan a dos niveles, es decir, dependiendo de que b_{ij} sea mayor o menor que $\hat{b}$, se envía 1 ó 0 respectivamente.

Después de la transmisión, en el receptor cada pixel se decodifica a L_0 o L_1 dependiendo si se recibió un 0 ó 1, respectivamente. Evaluando q, como el número de 1's recibidos para el bloque N; y el número de 0's, se calcula como $p = L^2 - q$. Finalmente se evalúan los dos niveles del cuantizador conservando el promedio y la varianza:

$$L_0 = \hat{b} - \sigma\sqrt{\frac{q}{p}}$$

$$L_1 = \hat{b} - \sigma\sqrt{\frac{p}{q}}$$

Dependiendo del número de niveles de cuantización y de las imágenes, al utilizar esta técnica, puede haber resultados buenos ó ineficientes. Para imágenes con los bordes y los objetos grandes se reproducen muy bien con pocos niveles de cuantización; sin embargo, en áreas con poco detalle de la imagen donde la brillantez y/o color son pequeños, existen errores de discontinuidad y bordes fantasmas, debido a los pocos niveles de cuantización [GON93].

3.3 TÉCNICAS SIN PÉRDIDA DE INFORMACIÓN.

Es una división que generaliza la clasificación de las técnicas de compresión, debido a que todas las técnicas se pueden encontrar dentro de este grupo. La característica principal de este grupo, es que la información original puede ser perfectamente recuperada de la representación digital, sin ningún tipo de distorsión ó ruido. Es aplicable a imágenes ya digitalizadas y requiere técnicas de códigos con longitud variable. Considerando que este código, pueda utilizar códigos cortos para palabras más comúnes; diseñados de tal manera que el número de bits para cada pixel sea lo más pequeño.

Las técnicas más comúnes para la compresión sin pérdida de información son: los códigos Huffman, los códigos Huffman modificados, algoritmos LZ (Ziv-Lempel) ó de diccionario y los eódigos aritméticos. Estas técnicas tienen razones de compresión que varían desde 1.7:1 hasta 4:1.

La importancia de estas técnicas es sumamente grande, sobre todo dependiendo de la aplicación que se desee. En datos muy estrictos como en la medicina, ciencia, ingeniería, química, arquitectura, etc., donde se requiere que los datos que se transmitan o almacenen sean exactamente iguales. Aunque también es cierto, que se buscan aproximaciones, razón por la cual se utilizan las técnicas con pérdida de información, las cuales analizamos en detalle en este capítulo. Vimos que se puede lograr que la información sea exactamente parecida, dependiendo de varios factores, tanto de la información que se transmite como de la técnica empleada.

CAPÍTULO IV.

ALGORITMOS DE COMPRESIÓN PARA IMÁGENES BINARIAS

CAPÍTULO IV.

ALGORITMOS DE COMPRESIÓN PARA IMÁGENES BINARIAS

4. 1 CÓDIGOS DE HUFFMAN.

Los códigos de Huffman, fueron descritos por D. A. Huffman en 1952. Estos códigos, son una técnica de compresión de información más utilizada por imágenes binarias, debido a que reduce la lougitud promedio de los códigos, que se utilizan para representarlas. Las imágenes binarias , son aquéllas que tienen únicamente un color (blanco o negro). Representando así el blanco como un "1" y el negro como un "0" , por lo tanto tienen un bit por pixel, algunos ejemplos estándares de ellas se encuentran en el capítulo 5.

La finalidad de esta técnica es reducir la longitud promedio de la codificación estadística; siempre y cuando no se duplique, esto significa que si un carácter se representa con la combinación 1100, entonces 110001 puede ser el código de otra letra, puesto que buscado en la cadena de bits, de izquierda a derecha, el algoritmo podría interpretar los 6 bits, como el código 1100 seguido de 01.

La codificación del código de Huffman, se genera en forma de una estructura de árbol tal como se muestra en la **figura 4.1** [HEL91]. Los símbolos (caracteres) y la frecuencia de ocurrencias (probabilidad), se ordenan en tal forma que los caracteres con las frecuencias más pequeñas (X_3 y X_4), se combinan en un nuevo nodo, resultado de sumar ambas probabilidades, después este nodo se une con la siguiente probabilidad más baja de la ocurrencia de otro carácter o par de caracteres.

En la **figura 4.1** [HEL91], el par X_3X_4 se une con X_2 para producir un nodo, con probabilidad de 0.350; finalmente, el nodo que representa las probabilidades de X_2,X_3 y X_4 se une con X_1 dando como resultado un nodo cuya probabilidad es uno.

Este último nodo representa la probabilidad de ocurrencia, de todos los caracteres del grupo, después se van asignando valores binarios de 0's y 1's a cada segmento que parte de cada nodo, deduciendo así el código de Huffman para cada carácter. El código se obtiene trazando una ruta, desde el nodo con probabilidad uno, a cada carácter escribiendo cada uno de los 0's o 1's encontrados. El promedio de bits por carácter, se obtiene multiplicando la longitud del código de Huffman por su probabilidad de ocurrencia.

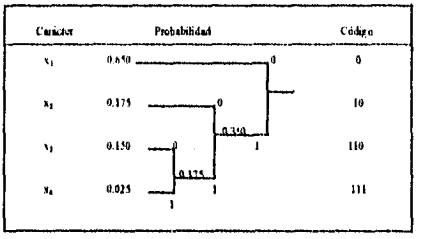


Figura 4.1 Desarrollo del Código de Huffman utilizando una estructura de árbol.

La decodificación es un proceso después de la codificación. En donde el código es decodificado al momento, en que los bits se leen de izquierda a derecha de la cadena de datos comprimidos, sin esperar a que se presente el fin de bloque. Se muestra en la **figura 4.2** [HEL91], este proceso.

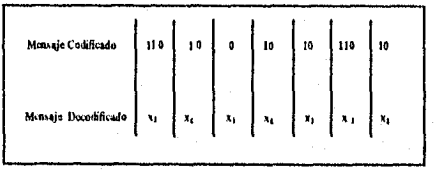


Figura 4.2 Decodificación del código de Huffman.

A continuación se muestra un ejemplo del desarrollo de los códigos de Huffman, dentro de la **figura 4.3.** El cual se desarrolló de la siguiente manera:

El primer paso es la reducción de fuentes, y generación del árbol binario.

a) Se forma una columna a la izquierda con todos los caracteres, a su lado otra columna con sus respectivas probabilidades de ocurrencia, considerando que las probabilidades de cada carácter se encuentren en orden descendente.

b) Posteriormente, se dibujan líneas horizontales para cada probabilidad del carácter. Y las líneas con las dos probabilidades menores se unen sumándose, para obtener una probabilidad compuesta, apartir de esta probabilidad se traza un línea horizontal más.

c) El proceso de combinar, las dos probabilidades menores continúa hasta que se han unido todas las líneas.

El segundo paso en el procedimiento Huffman es el codificar cada fuente reducida, empezando con la fuente más pequeña y trabajando hacia atrás de la fuente original. Es decir después de que se ha construido el árbol, el código de Huffman de cada carácter, se asigna colocando un bit 0, en uno de los lados del nodo y un bit 1 en el otro[2]. La secuencia apropiada de bits para cada carácter, se determina trazando la ruta del último nodo al nodo correspondiente al carácter, utilizando los 0's ó 1's que se encuentren en dicha ruta. Los códigos de Huffman de cada carácter se muestran en la **figura 4.3**.

FUENTE ORIGINAL		REDUCCIÓN DE FUENTES (Generación del árbol binario)				
CARÁCTER	PROBABILIDAD	1	2	3	4	CÓDIGO
A	0.4	0.4	0.4	0.4 *0*	**0.6** *0*	*1*
C	0.3	0.3	0.3 *0*	0.3	0.4 *1*	*00*
E	**0.1**	0.1 *0*	**0.2**	**0.3** *1*		*110*
R	0.1 *0*	0.1	**0.1** *1*			*0010*
O	0.06	0.1 *1*				*01010*
T	0.04 *1*					*11010*

Figura 4.3 Ejemplo de la generación de un Código Huffman

Por ejemplo veamos la ruta que sigue el carácter A con las probabilidades más negritas, en el segundo paso es sumado con 0.2 (tiene asignado el código 1), en el tercer paso es sumado con 0.3 (tiene asignado el código 1), dando la probabilidad de 0.6 (tiene asignado el código 0) en el cuarto paso. Si lo leemos tal como fueron asignados de derecha a izquierda tendríamos el código 110.

La medida promedio de este código es

$$\text{Lavg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2 \text{ bits/carácter}$$

y la entropía de la fuente es 2.14 bits/carácter, y la eficiencia del código es 0.973.

[2] La asignación de que lado se colocan los 0's ó los 1's es arbitraria, pero una vez que se escoge un orden de asignación se debe conservar para todos los nodos del árbol.

El número de bits requerido para codificar una letra usando la técnica de Huffman puede ser determinado con la fórmula:

$$b = f(-\log_2 P)$$

donde P = probabilidad de ocurrencia de la letra.
$f(x)$ = el entero más cercano, más grande o igual que x.

Puesto que la probabilidad de A es 0.4 y $-\log_2 0.4$ es 1.32, entonces el entero más grande o igual a 1.32 es 1, de manera que se requiere 1 bit para codificar la letra A.

Después de que se ha creado el código, en sí es el único código decodificable. Instantáneo, porque cada información es leída en una cadena de caracteres clave que son decodificados sin referir caracteres sucesivos.

Es decodificable únicamente, porque alguna cadena del código puede ser decodificados en un solo paso. Así, alguna cadena de caracteres codificados Huffman puede ser decodificado por examinación de izquierda a derecha de los caracteres individuales,. Para el código 010100111100 revela que el primer código es 01010 que corresponde a O, por lo que la palabra codificada es OEAAC.

Algunos resultados obtenidos se muestran en la **tabla 5.1** del capítulo 5, aplicado a imágenes binarias. Donde se obtuvo una tasa de compresión promedio de 4.92 : 1. De los resultados obtenidos concluimos que el método de compresión no es muy eficiente para imágenes en las cuales no predomina un tono o grupo de tonos en particular.

4.1.1 REQUERIMIENTOS PARA EL CÓDIGO HUFFMAN.

El requisito necesario del código de Huffman, es conocer la *distribución de probabilidad (ó distribución de frecuencia)* de cada carácter o símbolo a codificar, para poder calcular la longitud promedio del código.

Debido que la distribución de frecuencia de una cadena de datos es proporcional al uso final de la cadena, es factible utilizar una distribución preseleccionada de frecuencias para desarrollar los códigos de Huffman óptimos, en ciertas secuencias de transmisión. Como un ejemplo, la distribución de frecuencias de documentos como un carta y una factura son diferentes, en ambos hay letras y números, pero la factura tiene mayor ocurrencias de números.

Para compensar las diferencias de distribución de frecuencias, existen varios métodos de codificación.

Un primer método, es considerar una *codificación adaptativa de Huffman* (se explica en la siguiente sección). Esta técnica requiere del análisis de un bloque de datos grande, el cual puede ser codificado en base a su distribución. Antes de la transmisión de datos codificados, se debe almacenar la tabla de código de Huffman para cada símbolo, para que éstos datos puedan ser decodificados. Uno puede darse cuenta de que las frecuencias cambiantes de las cadenas de datos, requieren del almacenamiento de numerosas tablas junto con la información codificada; estas tablas son una sobrecarga y hacen que la razón de compresión disminuya.

Un segundo método para compensar las diferencias en las distribuciones de frecuencias es el uso de un *código de texto plano*. El código de texto plano, se utiliza para indicar que el carácter debe reproducirse tal como se recibe, esto permite a los caracteres que aparecen pocas veces, ser excluidos del proceso de codificación. En este esquema se agrupan a todos los caracteres de baja ocurrencia y se les asigna un código huffman que representa la suma de las probabilidades individuales; éste podría ser el código de texto plano, para indicar que los 8 bits siguientes representan un carácter no codificado. Por ejemplo sí el código de texto plano fuera de cuatro bits de longitud, cuando mucho 12 bits pueden ser requeridos para representar a cualquier carácter de baja frecuencia. Sin el uso del código de texto plano, se necesitarían cadenas largas de caracteres de 20 ó más bits, para representar los caracteres con baja frecuencia de ocurrencia [HEL91].

4.1.2 CÓDIGOS DE HUFFMAN MODIFICADOS

Es una variante del código de Huffman, apropiada de tal forma que la representación de caracteres tenga un promedio menor de bits por símbolo. Es importante señalar que el área que más se ocupa es en la transmisión de Documentos, debido a que podemos ver cada renglón como agrupaciones intercaladas de pixeles negros y blancos. Si conocemos el elemento inicial en cada renglón, entonces podemos conocer el tipo de agrupaciones que seguirán a esta. Así podemos aplicar el algoritmo de Huffman modificado, asignando códigos cortos a agrupaciones que ocurren con mayor frecuencia.

De donde podemos observar que para cada documento, difiere la probabilidad de ocurrencia de cada agrupación, por lo cual un código puede ser apropiado para algunos documentos, pero no para otros. Sí a esto agregamos la necesidad de realizar procesamiento en tiempo real, el problema se vuelve mayor.

Para disminuir los requerimientos de procesamiento en tiempo real, se utiliza una tabla de *lookup*. Cada uno de los estándares de la CCITT[3] requieren 1728 pixeles por línea, y 2376 líneas, por lo que el uso de la técnica de la tabla de *lookup*, requiere almacenamiento para 1728 localidades de longitud variable, para cada máquina de facsímil; cada localidad contiene un código binario que corresponde a una longitud de corrida en particular [HEL91].

[3] Comité Consult International Téléphonique e Comité Consultor Internacional de Telegrafía y Telefonía; es un comité europeo para la estandarización de formatos de transmisión y comunicaciones, principalmente de facsímiles.

En el desarrollo de una técnica con códigos de Huffman modificados para aplicaciones de facsímil se ha hecho un cambio, el cual aunque raramente permite que la longitud promedio para codificar un símbolo, se acerque a su entropía, permite una compresión aceptable minimizando los requerimientos de hardware y del tiempo de procesamiento. En este algoritmo de probabilidad de ocurrencia, de diferentes longitudes de corridas de pixeles fue calculada, para todas las longitudes de corridas de blancos y negros, con base en las estadísticas obtenidas del análisis de 8 documentos típicos recomentados por la CCITT [HEL91]. *Los documentos fueron obtenidos, del netscape, la dirección es:* http: //nic.funet.fi/pub/graphics/misc/test-images), *para aplicación del sistema de compresión propuesto en esta tesis (ver el siguiente capítulo).*

Los códigos de Huffman fueron truncados, con la creación de una representación en base 64 para cada longitud de corrida y la utilización de dos tablas de códigos para reducir el tamaño total de la tabla en comparación, con el tamaño que se hubiera requerido si solamente se usará una tabla. Se reducen los requerimientos de almacenamiento de la *tabla de Lookup.* Con las probabilidades de la longitud de las corridas de 8 documentos típicos, se han diseñado tablas de códigos con corridas que varían desde 0 a 63 pixeles, como se muestra en la tabla **figura 4.4** [HEL91].

Longitud de la Corrida	Código de Terminación(TC) Blanca	Representación en Base 64	Código de Terminación(TC) Negra
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	a	0000100
11	01000	b	0000101
12	001000	c	0000111
13	000011	d	00000100
14	110100	e	00000111
15	110101	f	000011000
16	101010	g	000001011
17	101011	h	000001000
18	0100111	i	000001000
19	0001100	j	000001100111
20	0001000	k	000011101000
21	0010111	l	000001101100
22	0000011	m	000001110111
23	0000100	n	000001101000
24	0101000	o	000000101111
25	0101011	p	000000011000
26	0010011	q	000011001010
27	0100100	r	000011001011
28	0011000	s	000011001100
29	00000010	t	000011001101
30	00000011	u	000001101000
31	00011010	v	000001101001
32	00011011	w	000001101010
33	00010010	x	000001101011
34	00010011	y	000011010010
35	00010100	z	000011010011
36	00010101	A	000021010100
37	00010110	B	000011010101
38	00010111	C	000011010110
39	00101000	D	000011010111
40	00201001	E	000001101100
41	00101010	F	000001101101
42	00101011	G	000011011010
43	00101100	H	000011012011
44	00101101	I	000001010100
45	00000100	J	000001010101
46	00000101	K	000001010110
47	00001010	L	000001010111
48	00001011	M	000001100100
49	01010010	N	000001100101
50	01010011	O	000001010010
51	01010100	P	000001010011
52	01010101	Q	000000100100
53	00200100	R	000000110111
54	00100101	S	000000111000
55	01011000	T	000000100111
56	01011001	U	000000101000
57	01011010	V	000001011000
58	01011011	W	000001011001
59	01001010	X	000001010011
60	01001011	Y	000000101100
61	00110010	Z	000001011010
62	00110011	*	000001100110
63	00110100	#	000001200111

Figura 4.4 Códigos de los dígitos menos significativos para el proceso de Huffman Modificado.

La figura anterior contiene la diferencias de ambas corridas (negras y blancas); el código en esta tabla se conoce como *código de terminación* (TC Terminating Code) y representa el dígito menos significativo LSD de la palabra de código.

Para permitir la codificación de corridas que pasan de 63 pixeles, se debe emplear una segunda tabla para manejar corridas que van de 64 pixeles a 1728 pixeles. Estos códigos se encuentran en la **figura 4.5** [HEL91], donde se conocen cómo códigos maestros y representan el dígito más significativo MSD.

Longitud de la Corrida	Código de Terminación (TC) Blanca	Representación en Base 64	Código de Terminación (TC) Negra
64	11011	1	0000001111
128	10010	2	000001001000
192	010111	3	000001001001
256	0110111	4	000001011011
320	00110110	5	000000110011
384	00110111	6	000000110100
448	01100100	7	000000110101
512	01100101	8	000000101100
576	01101000	9	000000101101
640	01100111	a	000000101010
704	011001100	b	000000101011
768	011001101	c	000000100100
832	011010010	d	000000100101
896	011010011	e	000000111010
960	011010100	f	000000111011
1024	011010101	g	000000111100
1088	011010110	h	000000111101
1152	011010111	i	000000111010
1216	011011000	j	000000121011
1280	011011001	k	000000101010
1344	011011010	l	000000101011
1408	011011011	m	000000101100
1472	010011000	n	000000101101
1536	010011001	o	000000101010
1600	010011010	p	000000101011
1664	011000	q	000000110100
1728	010011011	r	000000110101
EOL	000000000		000000000

	Código de Terminación (TC)		Código de Terminación (TC)
1792	00000001000	2240	000000010110
1856	00000001100	2304	000000010111
1920	00000001101	2368	000000011100
1984	000000010010	2432	000000011101
2048	000000010011	2496	000000011110
2112	000000010100	2560	000000011111
2176	000000010101		

Figura 4.5 Códigos de los dígitos más significativos para el proceso de Huffman Modificado.

Cuando se encuentra una corrida con 63 ó menos pixeles, se accesa el tipo de código TC para obtener un código en base 64. Para codificar una corrida de más de 64 pixeles, se deben usar dos códigos en base 64, siguiendo el procedimiento: primero, se obtiene el código maestro de tal manera que N* 64 (para $1 \leq N \leq 27$) no exceda la longitud de la corrida, después se calcula la diferencia entre la longitud de la corrida y N*64, y el dígito menos significativo LSD se accesa de la tabla de códigos TC apropiada, [HEL91].

La figura 4.6 [HEL91], muestra un ejemplo de las operaciones en una *tabla de lookup*, para una secuencia de corridas blancas y negras de varios tamaños; en la parte superior de esta ilustración se tabula la relación entre una serie de datos de vídeo original y su representación en los códigos de Huffman Modificados.

Datos originales de video	Código de Huffman Modificado, en base 64	Código de Huffman Modificado, en base 2	
		Código Maestro MSD	TC LSD
5 pixeles negros	5(negro)	NA	0011
17 pixeles blancos	h(blanco)	NA	10101)
32 pixeles negros	w(negro)	NA	000001101010
32 pixeles blancos	w(blanco)	NA	00011011
728 pixeles negros	b0(negro)	000000100101	00000010111
1728 pixeles blancos	r0(blanco)	010011011	00110101
64 pixeles negros	10(negro)	000000111	0000110111
55 pixeles blancos	T(blanco)	NA	01011000
1028 pixeles blancos	g2(blanco)	011010101	0111

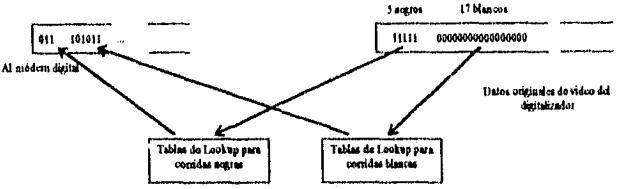


Figura 4.6 Codificación usando los códigos de Huffman Modificados, para una secuencia de referencias de corridas negras y blancas.

En estas técnicas los códigos no contienen información inherente de la posición, la cual es necesaria para la sincronización; esto se compensa estableciendo la regla de que la *primera corrida* de todas las líneas debe ser *blanca*, aún si debe ser una corrida de longitud cero, de esta manera las corridas alternan entre blanco y negro.

Para denotar el principio de cada línea, es necesario emplear un delimitador de fin de línea llamado *código de fin de línea (EOL, End Of Line)*, una vez que la línea se codifica se puede rellenar con 0's, si fuera necesario, antes de colocar el carácter EOL para propósitos de sincronía. El resultado de aplicar estas reglas se presenta en la **figura 4.7** [HEL91].

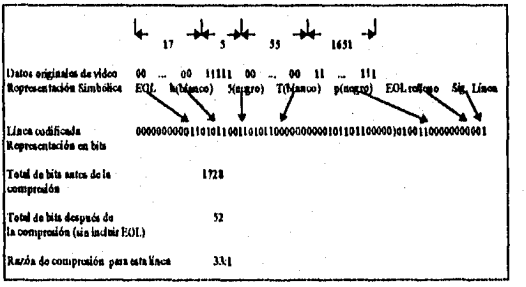


Figura 4.7 Reglas que definen el formato de Línea., para representar el principio y fin de cada línea común, se utiliza el código de EOL (fin de línea).

4.2 ALGORITMOS DE LA CCITT.

Los estándares más utilizados para la compresión de imágenes binarias, son por parte de la CCITT, el grupo 3 y 4. Estos algoritmos, llamados *Grupo (G3)* se utilizan principalmente para transmisión de documentos en redes de comunicación. Este algoritmo, elimina la redundancia que se genera en la transmisión de información, por los códigos especiales al final de cada línea y códigos comprimidos unidimensionalmente cada dos o cuatro líneas; ejemplos de este grupo: es el algoritmo RLE unidimensional y el otro elimina la correlación bidimensional en los datos.

En cuanto al siguiente *Grupo 4* (G4) se aplica a imágenes y líneas de comunicación más generales. Como ejemplo de esto, para las comunicaciones en redes digitales con control de errores, los algoritmos G4 se obtienen a partir de los G3, removiendo las dos redundancias mencionadas previamente, puesto que la red provee una transmisión libre de errores. Estos algoritmos de compresión de la CCITT, se desarrollaron principalmente para la transmisión de facsímiles (FAX), sin embargo, algunos formatos comerciales de almacenamiento de imágenes utilizan estos algoritmos (por ejemplo, el formato TIFF) [GON93].

4.2.1 CODIFICACIÓN UNIDIMENSIONAL DE LA CCITT

Esta técnica de codificación trata que cada línea del documento se codifique asignando palabras a las corridas de pixeles blancos o negros; asumiendo que estas líneas comienzan con una corrida blanca, y en caso de lo contrario, si la primera corrida es negra se agrega una corrida blanca de longitud 0. Se utilizan tablas de códigos por separado para representar corridas blancas y negras, basadas en los códigos modificados de Huffman y en estadísticas de las 8 imágenes estándares de la CCITT.

La tabla de códigos que se utiliza en esta técnica, es la misma tabla de códigos de Huffman modificados descrita en la sección anterior. La tabla de códigos, contiene códigos para corridas, de hasta 1728 pixeles por línea. Estos códigos son de dos tipos: *códigos de terminación (TC Terminating Codes)* y *códigos maestros*. Cuando las corridas son entre 0 y 63 se transmiten utilizando una sola palabra de código LSB, y las corridas entre 64 y 1728 se transmiten con un código maestro MSB seguido de un TC; el código maestro representa una longitud de *64xN* (donde *N* es un entero entre 1 y 27), la cual es igual o menor que la longitud de la corrida que se transmite. La diferencia entre el código maestro y la longitud actual de la corrida se especifica por el TC; a cada línea codificada se le agrega un EOL, el cual es una secuencia única, que no puede ocurrir dentro de los datos codificados, el código utilizado es 000000000001, 10 0's seguido de 1 [GON93].

Si el número de bits codificados en una línea es menor que un cierto valor, se agrega un relleno de 0's entre el fin de los datos codificados y el código de fin de línea, esto asegura que cada línea codificada requiera de un tiempo fijo de transmisión, de manera que el transmisor y el receptor puedan sincronizarse; el estándar mínimo recomendado, es 96 bits/línea a una razón de transmisión de 4800 bits/seg, con opciones para (48, 24 y 0) bits/línea, de esta manera los bits de relleno son reconocidos y descartados fácilmente por el receptor.

La razón de compresión del algoritmo RLE, para 8 documentos estándares de la CCITT se muestra en la **figura 4.8** [GON93], de donde el promedio de compresión es de 9.7 .

Doc.	Promedio Corrida Blanca	Promedio Corrida Negra	Entropía Corrida Blanca	Entropía Corrida Negra	Razón Máx. Comp.	Razón Comp.
1	134.60	6.790	5.230	3.592	16.02	15.16
2	167.90	14.02	5.989	4.457	17.41	16.67
3	71.50	8.468	5.189	3.587	9.112	8.35
4	36.38	5.673	4.574	3.126	5.461	4.911
5	66.41	6.996	5.200	3.339	8.513	7.927
6	90.65	8.001	5.063	3.651	11.32	10.78
7	39.07	4.442	5.320	3.068	5.188	4.99
8	64.30	60.56	4.427	5.310	11.52	8.665

Figura 4.8 Desempeño y estadísticas del algoritmo unidimensional, para los 8 documentos estándares de la CCITT. La razón de compresión es calculada utilizando los códigos de Huffman, sin el fin de línea y sin los bits de relleno.

4.2.2 CODIFICACIÓN BIDIMENSIONAL DE LA CCITT

Es una técnica de codificación propuesta por la CCITT en 1979. Conocida como *Código READ Modificado (READ Relative Element Address Designate o Dirección Designada del Elemento Relativo)*. Esta técnica codifica la posición de cada elemento en cada línea; es decir, la posición de cada *elemento cambiante* en la línea presente, se codifica con respecto a, la posición correspondiente de un *elemento cambiante* en la línea de referencia, la cual se encuentra inmediatamente arriba de la presente línea, o bien con respecto al *elemento cambiante* precedente en la misma línea presente; después de que la línea presente se ha codificado, pasa a ser la línea de referencia para la siguiente línea. El *elemento cambiante,* es un elemento de color diferente al elemento anterior, a lo largo de la misma línea.

El procedimiento de codificación utiliza 5 elementos cambiantes, definidos como se muestra en la **figura 4.9,** [GON93].

a_0 : El primer elemento cambiante de la línea presente.

a_1 : El siguiente elemento cambiante de la línea presente, de acuerdo a la definición, tiene un color opuesto a a_0.

a_2 : El elemento cambiante de la línea presente, siguiente a a_1.

b_1 : El elemento cambiante de la línea de referencia a la derecha de a_0 con el mismo color de a_1.

b_2 : El elemento cambiante de la línea de referencia, siguiente a b_1 .

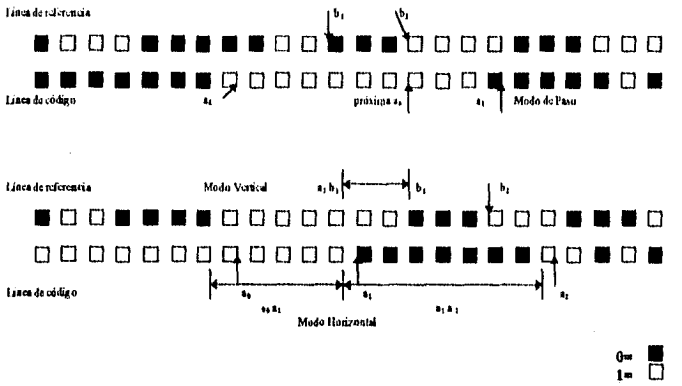


Figura 4.9 Ejemplos de modos de codificación para el esquema bidimensional. La configuración superior muestra el modo de paso, la central y la del fondo muestran los modos vertical y horizontal respectivamente.

El codificador opera un modo diferente, dependiendo de la posición relativa del elemento cambiante que está siendo codificado:

Modo de Paso

Esto ocurre cuando la corrida blanca o negra en la línea de referencia, no es adyacente a la correspondiente corrida blanca o negra en la línea presente. En este caso el elemento b_2 se encuentra horizontalmente a la izquierda de a_1. El modo de paso se representa con una palabra de código.

Modo Vertical.

En este caso el elemento a_1 está lo suficientemente cerca a b_1, y de esta manera se codifica relativamente con la posición de b_1. Se utiliza solamente si a_1 está a la izquierda o a la derecha de b_1 al menos 3 pixeles, y por lo tanto la distancia relativa $a_1 b_1$ puede tomar cualquiera de los 7 valores V(0), $V_D(1)$, $V_D(2)$, $V_D(3)$, $V_I(1)$, $V_I(2)$, $V_I(3)$; el subíndice de D se utiliza si a_1 está a la derecha de b_1 e I si se encuentra a la izquierda; el número entre el paréntesis indica la distancia $a_1 b_1$ en pixeles.

Modo Horizontal.

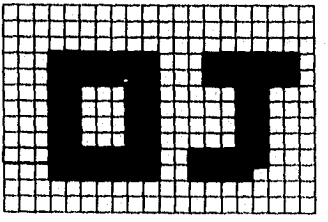
Si a_1 no está lo suficientemente cerca a b_1 entonces su posición debe codificarse en modo horizontal. De esta manera, las corridas a_0a_1 y a_1a_2 se codifican utilizando la concatenación de las tres palabras de código huffman, M(a_0 a_1) y M(a_1 a_2). La palabra de código huffman, tomada como 001, sirve como prefijo o bandera, y M(a_0 a_1) y M(a_1 a_2) se toman de la tabla de código para representar colores y valores de las corridas a_0a_1 y a_1a_2 , esta tabla se basa en los códigos de Huffman modificados igual que lo hacia el esquema unidimensional de la CCITT [GON93].

4.3 ALGORITMOS RLE

Los algoritmos RLE (*Run Length Encoding o Codificación de la longitud de corrida*), son una técnica de codificación, normalmente para imágenes monocromáticas, aunque su uso se ha extendido a imágenes de 16 ó 256 colores. La codificación RLE se utiliza en conjunto con el algoritmo modificado de Huffman así como en los algoritmos de compresión de la CCITT [HEL91].

4.3.1 CODIFICACIÓN RLE

Esta codificación tiene como objetivo, representar con un simple valor ("0" ó "1") y un contador (que lleva la cuenta de las ocurrencias del dato) a cada longitud de corrida de pixeles blancos o negros. Este tipo de compresión se da frecuentemente cuando se tienen grandes espacios en blanco o grandes regiones rellenas de un solo color. Por ejemplo en la **figura 4.10** la longitud de cada corrida (pixeles negros o blancos) se representa con palabras binarias; puesto que las corridas de pixeles blancos alternan con corridas negras, excepto para el comienzo de cada línea, el color de la corrida no necesita almacenarse (en el caso de una imagen a color, se debe almacenar explícitamente el color de la corrida). Las estadísticas de la longitud de corrida varían de documento a documento.



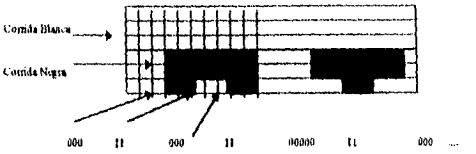


Figura 4.10 Representación de una imagen en blanco y negro digitalizada en una cuadrícula de pixeles y su representación en corridas horizontales.

Se muestran algunas estadísticas importantes de la eficiencia del algoritmo RLE para imágenes estándar de la CCITT, en la tabla 5.2 del cap. 5. Es claro que para documentos que tienen información negra sobre fondo blanco el promedio de la longitud de corrida blanca es mucho más grande, que el promedio de la corrida negra. La razón de compresión varía dependiendo del documento, en promedio, la razón de compresión es de 7 : 1 aproximadamente, para este tipo de imágenes. De donde podemos concluir que esta técnica es más apropiada a imágenes en las cuales existan corridas largas de pixeles de un solo color o tono (blanco o negro), porque se puede lograr una razón de compresión mayor.

4.4 ALGORITMOS LZ

Este algoritmo de Ziv y Lempel es una técnica adaptativa, la cual va construyendo un diccionario añadiendo a este cada palabra o grupo de palabras que se ha almacenado, y con él van codificando el mensaje por lo que deben encontrar la forma de almacenar el diccionario, para que el decodificador pueda efectuar su función. Por otro lado, anteriormente mencionamos que las técnicas estáticas utilizan un diccionario que no necesita almacenarse, porque ya se encuentra definido implícitamente en la técnica, y las técnicas semiadaptativas necesitan almacenar el diccionario antes de enviar el mensaje.

Los algoritmos LZ, durante la codificación usan el diccionario actual en cada punto, para codificar la siguiente porción del mensaje, cambiando una subcadena del mensaje por una referencia a otra subcadena en alguna parte anterior del mensaje. A continuación se describen dos algoritmos LZ.

4.4.1 ALGORITMO LZ77

El algoritmo LZ77, se basa en un buffer de desplazamiento de longitud N a través del cual se pasa el mensaje de derecha a izquierda (figura 4.11 [TIM91]). Consta de dos parámetros fundamentales:

- N ($1 \leq N \leq \infty$), la longitud del buffer utilizado en la compresión.
- F ($1 \leq F \leq N-1$), la máxima longitud de la cadena coincidente, con $F \leq N$.

Los valores típicos que se utilizan en la práctica son 2^{13} para N y 2^4 para F.

Los elementos del buffer se numeran consecutivamente con el 1 a la izquierda y N a la derecha, los N-F elementos de izquierda a derecha son las subcadenas que se van encontrando y definiendo y los F elementos más a la derecha son las posibles subcadenas.

La sección A contiene los N-F datos transmitidos previamente y la sección B almacena los siguientes F datos que se codificarán.

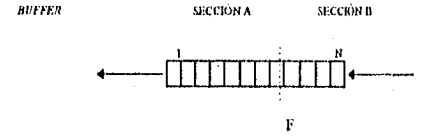


Figura 4.11 Representación del algoritmo LZ77.

El algoritmo inicializa la sección A con una cadena predefinida, colocando el inicio del mensaje en la sección A, la codificación se lleva a cabo encontrando la subcadena más larga en el buffer cuyos elementos más a la izquierda se encuentran en la sección A y coincidan primero con cero o más datos en la sección B y transmitiéndola (junto con el siguiente dato) como un conjunto de triadas (s,l,a) donde:

s, es la posición en la sección A donde comienza la cadena coincidente, en el rango ($1 \leq s \leq N-F$)
l, es la longitud de la cadena coincidente en el rango ($0 \leq s \leq F$) y
a, es el dato siguiente a la cadena coincidente.

Después, el mensaje se desplaza en el buffer desde la derecha hasta que el siguiente dato a ser codificado, se encuentre en el elemento más a la izquierda de la sección B, es decir, el elemento N-F+1 [TIM91].

Al decodificar es rápido, porque utiliza un buffer idéntico al codificador y copia repetidamente las subcadenas, especificadas por el conjunto de triadas, de la sección A a la sección B.

4.4.2 ALGORITMO LZ78

El algoritmo LZ78 es muy similar al LZ77 excepto que la sección A se reemplaza con un diccionario creciente de n frases (n en el rango de 0 a ∞) numeradas de 0 a n-1, no se limita la longitud de la sección B y el algoritmo no tiene parámetros.

El algoritmo inicializa el diccionario a una sola frase que consiste de la cadena vacía e inicializa n a 1. En cada paso, el algoritmo transmite una nueva frase $p \in S$ en la sección B, la cual consiste de la frase coincidente más larga $m \in S$ en el diccionario, más el siguiente elemento a. De esta manera $p = ma$ y la sección B es igual a "p...", p se transmite como un índice (con $log_2\ n$ bits) de entrada al diccionario de m's seguido por el elemento a. La nueva frase p se inserta en el diccionario, después, otra parte más del mensaje se desplaza a la sección B del buffer y el proceso continúa. Este proceso lleva a dividir la entrada en frases cada una de las cuales consiste de la frase previa más larga, más un elemento.

Como se muestra en la **figura 4.12** [TIM91], el algoritmo LZ78 construye un diccionario de frases y almacena repetidamente el número de la entrada más larga al diccionario que coincide la sección B, seguido de un elemento. Después de que cada frase se almacena se añade una nueva frase al diccionario, la nueva frase es la frase almacenada más el elemento siguiente. A continuación, se muestra el estado del algoritmo a mitad del procesamiento de la cadena "wooloomooloo".

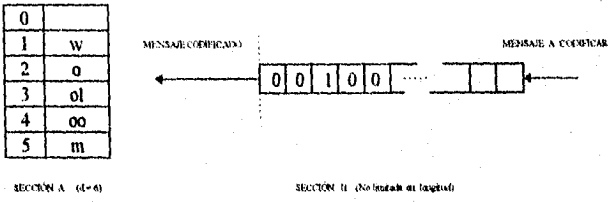


Figura 4.12 Representación del algoritmo LZ78.

Las características más importantes de este algoritmo son:

- En este algoritmo el diccionario se puede implementar utilizando un árbol de búsqueda (**figura 4.13** [TIM91]). Al analizarlo, consiste en ir de la raíz del árbol al nodo correspondiente a m y después añadiendo un nuevo nodo p a m utilizando una rama a.

- Tal como se define el algoritmo, el tamaño del diccionario se incrementa infinitamente, y la memoria puede agotarse. Para evitar esto, se puede vaciar el árbol y continuar el proceso [TIM91].

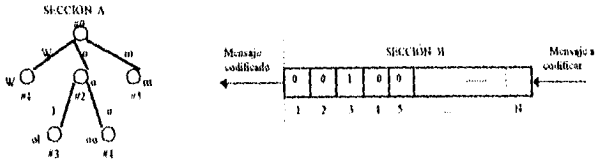


Figura 4.13 El algoritmo LZ78 implementado con un árbol.

4.4.3 ALGORITMO LZW

El algoritmo LZW (Lempel-Ziv-Welch), es una técnica de codificación por medio de una tabla de cadenas la cual contiene 2^j cadenas; esta tabla se inicializa con el conjunto de los 2^k valores posibles de los pixeles, y se logra una mejor compresión si $k<<j$, dependiendo de la cantidad de redundancia en los datos. La codificación define una cadena actual A con el primer pixel de la imagen; para formar la tabla de cadenas. Y continua de la siguiente forma, veasé en la **fig. 4.14** un ejemplo:

1. Si no existen más pixeles (fin de la imagen), escribe el código j-ésimo para A y termina. Si no continúa con el paso 2.
2. Se toma el siguiente pixel B y se agrega a A para formar la cadena AB.
3. Si AB se encuentra ya en la tabla de cadenas se hace $A = AB$ y continúa con el paso 1. Si no continúa con el paso 4.
4. Escribe el código j-ésimo para A.
5. Añade AB a la tabla de cadenas si aún existe espacio en ésta.
6. Se hace $A=B$ y continúa con el paso 1.

Entrada: a a b ab aba aa
Salida: 0 0 1 3 5 2

Tabla de cadenas:

No. de la cadena	Cadena	Derivada de la cadena
0	a	inicial
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a

Fig. 4.14 Codificación LZW de la cadena "aababababaa".

Durante la primera parte de la codificación puede haber mayor compresión, pero no después de que vaya creciendo el tamaño de la longitud de la cadena; en este caso, se puede utilizar una medida de compresión de tal manera que, cuando esta cae debajo de cierto valor prefijado, la tabla de cadenas debe ser reinicializada y reconstruida considerando los cambios en las estadísticas .

4.5 CÓDIGOS ARITMÉTICOS.

La idea de la codificación aritmética fue presentada por Shannon en 1942, en un seminario sobre la teoría de la información.

Un mensaje del código aritmético se representa por un intervalo de números reales entre 0 y 1. Como el mensaje llega a ser muy grande, el intervalo necesario para representarlo debe ser muy pequeño, y el número de bits necesarios para especificar el tamaño del intervalo. Los símbolos sucesivos del mensaje reducen el tamaño del intervalo de acuerdo con las probabilidades de los símbolos generados por el modelo. Los símbolos regulares reducen el rango menor que los símbolos irregulares, y así se adicionan menos bits al mensaje.

Un modelo exacto para la probabilidad de ocurrencia de cada símbolo en cualquier punto de la imagen que se desee comprimir, es codificar un símbolo con la longitud mínima promedio de los códigos, la cual se define como $-\log p$ bits cuya probabilidad de ocurrencia es p *(ver cap. 2, entropía)*.

Antes de que algo sea transmitido el rango para el mensaje es el íntegro intervalo semiabierto de 0 a 1, [0,1). Como cada símbolo es procesado, el rango se disminuye a tal porción de que el símbolo este bien distribuido, de la siguiente manera:

1. Utilizamos un intervalo [L,H) actual, inicializando a [0,1).

2. Para cada símbolo de la imagen se ejecutan los pasos siguientes, **figura 4.15 [TIM91]:**

   a) Se subdivide el intervalo actual en intervalos, cada uno de ellos corresponde a un posible símbolo del alfabeto. El tamaño del subintervalo para un símbolo es proporcional a la probabilidad estimada de que ese símbolo será la siguiente entrada, de acuerdo al modelo probabilístico utilizado.

b) Se selecciona el subintervalo correspondiente al símbolo que se presenta en la entrada, y se hace el intervalo actual.

3. Se escriben los bits necesarios para distinguir el intervalo actual final de los otros posibles subintervalos finales.

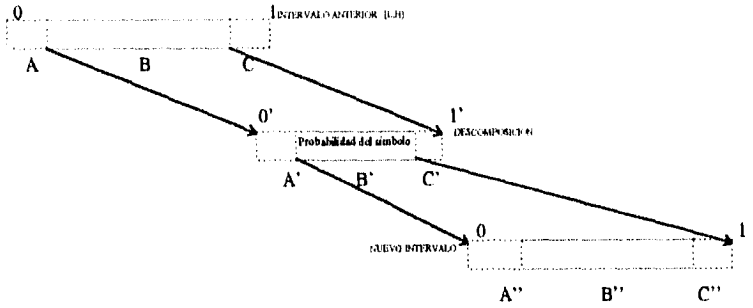


Figura 4.15 Subdivisión del intervalo actual basada en la probabilidad del siguiente símbolo de entrada.

La longitud del intervalo final es igual al producto de las probabilidades de los símbolos individuales, la cual es la probabilidad *p* de la secuencia particular de símbolos en el archivo. Y la subdivisión final utiliza *-log p* bits para distinguir al archivo de otros posibles archivos. Además de indicar el fin de archivo de la imagen, ya sea un símbolo especial de fin de archivo o alguna indicación externa de la longitud del archivo.

A continuación se presenta un ejemplo con modelo probabilístico no adaptativo para comprimir un archivo, que contiene los símbolos *bbb*, utilizando probabilidades arbitrarias $p_a = 0.4$, $p_b = 0.5$ y $p_{EOF} = 0.1$. El proceso de codificación es el siguiente:

Intervalo Actual	Acción	Subintervalos a	b	EOF	Entrada
[0.0, 1.0]	subdivide	[0.00, 0.40]	[0.40, 0.90]	[0.90, 1.00]	b
[0.40, 0.90]	subdivide	[0.40, 0.60]	[0.60, 0.85]	[0.85, 0.90]	b
[0.60, 0.85]	subdivide	[0.60, 0.70]	[0.700, 0.825]	[0.825, 0.850]	b
[0.700, 0.825]	subdivide	[0.70, 0.75]	[0.750, 0.812]	[0.812, 0.825]	EOF
[0.812, 0.825]					

El intervalo final (sin redondeo) es [0.8125, 0.825], el cual en representación binaria es aproximadamente (0.11010000, 0.11010011); se puede identificar este intervalo en forma única asignando la cadena 110100. De acuerdo al modelo fijo, la probabilidad *p* de este archivo es el tamaño del intervalo final 0.825-0.8125 = 0.0125, y la longitud del código (en bits) es $-\log p = -\log 0.0125 = 6.322$; en la práctica se utilizan 6 bits.

La entropía del mensaje bbb es

$$-\log 0.5 - \log 0.5 - \log 0.5 = -\log 0.0125 \text{ aproximadamente } 6.322.$$

Algunas características importantes del algoritmo [TIM91]:

- Incrementar la transmisión y recepción . El algoritmo codificado como se describe no transmite nada hasta que se codifique todo el mensaje, ni tampoco el algoritmo decodificado es decodificado hasta que se recibe la transmisión completa.

- El deseo de usar un punto aritmético fijo. La precisión requerida para representar el tamaño del intervalo, [L,H) con la longitud del mensaje.

- Representación del modelo. La representación usada para el modelo minimizaría el tiempo necesario para decodificar el algoritmo e identificar el próximo símbolo. Sin embargo, un modelo adaptativo se organizaría para minimizar el tiempo para mantener las frecuencias acumulativas.

- La característica importante de la codificación aritmética, es su optimización; en el sentido de codificar utilizando la longitud mínima promedio de los códigos. Por ejemplo, los códigos de Huffman, aunque sean igual de eficientes, cuando la probabilidad de un símbolo es cercana a 1, la codificación aritmética logra razones de compresión mucho mayores que otras técnicas.

4.6 CUANTIZACIÓN VECTORIAL

La Cuantización vectorial (VQ), es un algoritmo de compresión, el cual mapea vectores de tonalidades de pixeles, dentro de índices de vectores binarios, los cuales son de un número limitado.

Tomando como base los principios de cuantización escalar [RAB91],:

1) Divide el rango de posibles valores de entrada en un conjunto finito de subconjuntos, es decir, establece niveles de cuantización; y

2) Para cada subconjunto, escoge un valor representativo de salida, cuando la entrada se encuentra dentro de ese subconjunto.

Pero estos ya no tienen lugar en un espacio unidimensional, sino en un espacio vectorial de dimensión *N.*

De esta manera, el espacio se divide en 2^{2R} subconjuntos, y cada uno de ellos con su correspondiente valor representativo o *código vectorial*, entonces los bloques b de N pixeles se pueden codificar con RN bits por bloque o R bits por pixel.

VQ redondea (o cuantiza) grupos de números unidos, en lugar de uno a la vez. Estos grupos de números son llamados *vectores de entrada,* y los niveles de cuantización son llamados *vectores de reproducción.* Para especificar un cuantizador vectorial, necesitamos la serie de posibles vectores de reproducción y una regla para mapear los vectores de entrada, a los vectores de reproducción. El elegir un vector de reproducción Y, (el índice i de Y_i es binario) ó *palabra de código,* es usualmente un bloque de pixeles en tonos de gris de la misma dimensión como el bloque de entrada.

En la siguiente figura se muestra que X y $\hat{X}$ son insignificantemente diferentes, de donde demostramos que VQ es un algoritmo de *compresión Lossy* (con pérdida) [COS95]:

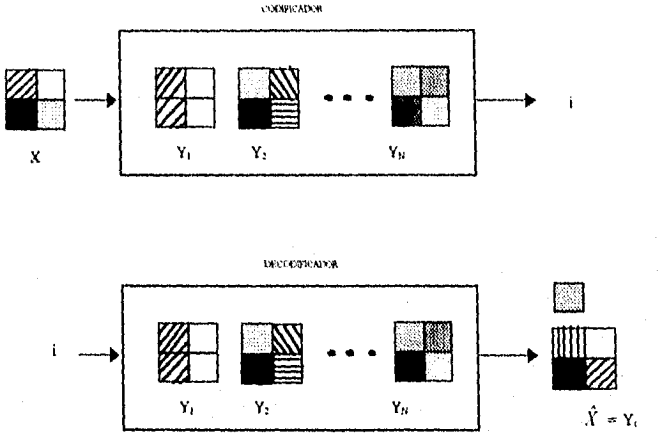


Figura. 4.16 Cuantización Vectorial.

Esta compresión tiene una razón de compresión de 32:8 ó 4:1. El decodificador también es una copia de la *tabla de códigos*, y opera como una simple *tabla de look-up*. Cuando recibe un índice i, el decodificador da la salida de la palabra almacenada Y_i.

La operación del codificador requiere una elección de la regla de mapeo, se muestra en la **figura 4.17,** un diagrama de bloques de la Cuantización Vectorial (VQ).

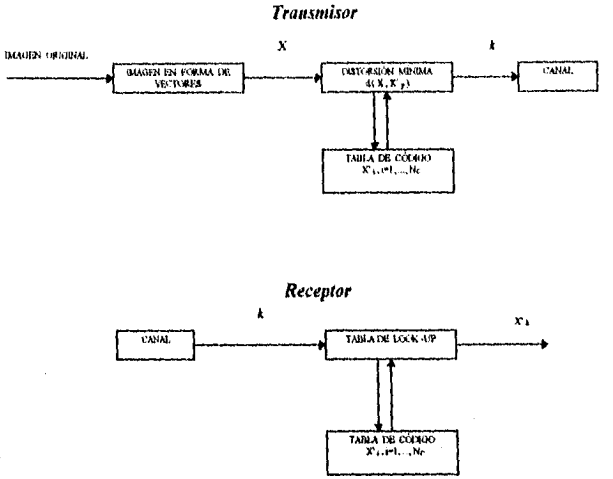


Figura 4.17 Diagrama de bloques de la Cuantización Vectorial.

Asumimos que $d(X, \hat{X}) \geq 0$ mide la distorsión o el costo de reproducir un vector de entrada X en uno de reproducción $\hat{X}$, y la distorsión del sistema es medida por un promedio de distorsión, el codificador óptimo para una tabla de código dada, elige el vector Y_i si

$$d(X,Y_i) \leq d(X,Y_j), \text{ para toda j}$$

Si no se asumen errores de otro tipo es posible definir una medición de distorsión $d(X-\hat{X})$ entre el valor de entrada X y el vector codificado $\hat{X}$. De esta manera, el problema de la cuantización vectorial es dividir y los códigos vectoriales de tal forma que minimicen la distorsión para el tipo de imágenes manejadas. La medición de distorsión más utilizada, al igual que en otras técnicas, es el error cuadrático promedio (MSE), el cual se define como:

$$d = \frac{1}{N}(X-\hat{X})'(X-\hat{X}) \text{, o bien } d(X,\hat{X}) = \frac{1}{N}\sum_{i=1}^{N}\left(X_i,\hat{X}_i\right)^2$$

y la distorsión es simplemente el valor esperado:

$$D = E(d)$$

Este método es usado ampliamente, debido a que es sencillo de calcular en la computadora. Una medida más útil en ciertas aplicaciones es el MSE, definido como:

$$d_w(X,\hat{X}) = \frac{1}{N}\sum_{i=1}^{N} w_i (X_i - \hat{X}_i)^2$$

donde las w_i's son los factores ponderados aplicados a las diferencias de los componentes del vector.

Después de haber elegido la palabra de código con la mínima distorsión, su índice k es transmitido usando $log_2 N_c$ bits, donde N_c es el número de pixeles. En el receptor, este índice es usado como una entrada a una tabla de código duplicada (*tabla look-up*) para reproducir $\hat{X}_k$. El decodificador VQ tiene una estructura muy simple, que consiste en una procedimiento de búsqueda, como se muestra en la **fig. 4.17.**

El diseño de cuantizadores vectoriales (VQ *Vectorial Quantizers*), generalmente requiere de imágenes de *entrenamiento,* puesto que todavía no hay disponibles estadísticas exactas. Los códigos vectoriales VQ y el particionamiento se determinan iterativamente repitiendo el VQ, un bloque *b* de pixeles se puede codificar utilizando la regla del *vecino más cercano*, es decir, se debe escoger el código vectorial $\hat{X}$ que minimice $d(X-\hat{X})$ y se transmite un índice codificado de *NR* bits, para indicar al receptor cual palabra de código debe utilizar; finalmente, el decodificador despliega el vector $\hat{X}$ como la representación codificada del vector *X*.

El diseño del código esta basado en una serie de datos conocidos y típicos, más que en modelos matemáticos de los datos. Estas imágenes son divididas en los vectores de entrenamiento y el algoritmo es ejecutado con esta serie de datos.

La principal ventaja de la VQ es la estructura en el receptor, la cual solamente consiste de una tabla de códigos que contiene 2^{NR} códigos vectoriales. La desventaja es la complejidad del codificador, y el hecho de que las imágenes diferentes al conjunto de entrenamiento no se representan correctamente en la tabla de códigos [RAB91]

4.6.1 DISEÑO DE LA TABLA DE CÓDIGOS.

En el diseño de la tabla de código se utiliza el algoritmo de Lloyd-Max, también conocido como algoritmo LGB (*Linde-Buzo-Gray*) ó algoritmo de los k-promedios, para cuantización escalar, el cual requiere de una tabla inicial de códigos y procede como sigue:

1. Mapea los bloques de entrenamiento , en códigos utilizando la regla del *vecino más cercano*. Si la distorsión total de este mapeo, o la cuantización es muy pequeña, termina; de lo contrario, sigue con el paso 2.

2. Para cada código vectorial $\hat{X}$, se determina el subconjunto de bloques de entrenamiento que se mapearon en éste; después, se remplaza $\hat{X}$ con otro código que reduzca (idealmente minimice) la distorsión total para dicho subconjunto de bloques de entrenamiento. Continua con el paso 1.

En el paso 2, para el error cuadrático promedio, lo mejor es remplazar el código con el promedio del correspondiente subconjunto de bloques de entrenamiento; el algoritmo termina cuando la distorsión total se ha minimizado al máximo posible, para una tabla inicinl de códigos dada, el algoritmo LGB produce una distorsión local mínima [RAB91].

Escoger una tabla inicial de códigos es un problema en sí; una selección adecuada es que la tabla inicial sea un subconjunto de vectores de entrenamiento, y que cada uno de ellos sea lo menos similar entre sí; otra tabla inicial de tabla de códigos podría obtenerse, simplemente utilizando un cuantizador con pocos niveles en cada uno de los componentes individuales de los vectores.

Otro método, se llama *técnica de particionamiento*, la cual comienza con un sólo código igual al promedio de los vectores de entrenamiento, después se genera el siguiente código introduciendo un pequeño factor diferencial y se aplica el algoritmo LGB para optimar estos dos códigos. En la **(figura 4.18 [RAB91])**, se muestra el proceso, en el cual un nivel K dado, a cada uno de los códigos vectoriales, se le agrega el factor diferencial para obtener el conjunto de $2K$ códigos vectoriales, los cuales se optimizan con el algoritmo LGB que produce la tabla de código al nivel $2K$; finalmente, los 2^{RN} códigos vectoriales.

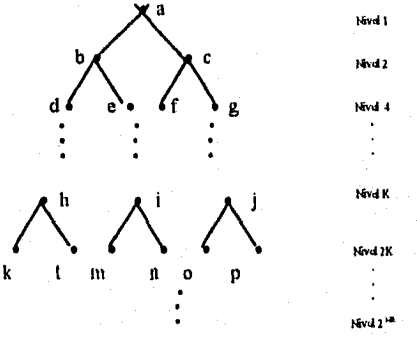


Figura 4.18 El método de particionamiento para generar la tabla de códigos VQ.

Los nodos de la figura 4.18 [RAB91] son:

$$a = C_1^i \quad b = C_2^1 \quad c = C_2^2 \quad d = C_4^1 \quad e = C_4^2 \quad f = C_4^3 \quad g = C_4^4 \quad h = C_k^{i-1} \quad i = C_k^i$$
$$j = C_k^{i+1} \quad k = C_{2k}^{2i-3} \quad l = C_{2k}^{2i-2} \quad m = C_{2k}^{2i-1} \quad n = C_{2k}^{2i} \quad o = C_{2k}^{2i+1} \quad p = C_{2k}^{2i+2}$$

Este árbol también se utiliza en el proceso de codificación; en cada nodo C_k^i, el vector de entrada se compara con los códigos vectoriales C_{2k}^{2i-1} y C_{2k}^{2i} y el más cercano se escoge como el siguiente código.

Un problema con el algoritmo LGB y sus variantes, es el tiempo de convergencia en la construcción de la tabla de códigos; sin embargo, las técnicas para reducir este tiempo, son casi siempre menos efectivas. Existe una técnica que reduce el tiempo de construcción de la tabla, conocida como *tabla de códigos en árbol*, la cual comienza con un código y procede al igual que la técnica de particionamiento. En cada etapa K el conjunto de vectores de entrenamiento se divide en subconjuntos de vectores de entrenamiento, que se mapean al mismo código vectorial; después, en la siguiente etapa, como cada código se multiplica por el factor diferencial para producir otro código vectorial, para optimar los dos vectores resultantes solamente se utilizan aquellos vectores de entrenamiento que se encuentran en el subconjunto correspondiente [RAB91].

4.6.2 VQ PARA PROCESAMIENTO DE IMÁGENES DIGITALES

El procesamiento de imágenes digitales puede ser dividido en diferentes clases de aplicaciones, incluyendo representación y modelación, mejoramiento, restauración, análisis y reconstrucción. En la siguiente figura se muestra un diagrama de la secuencia de pasos del procesamiento y decodificación con la generación de un archivo intermedio. En él, los datos originales son primero comprimidos y almacenados como una lista de índices de palabras de código. El decodificador lee los índices y genera una serie de datos decomprimidos (imagen reconstruida), de este modo se expande el respaldo del archivo comprimido al tamaño original.

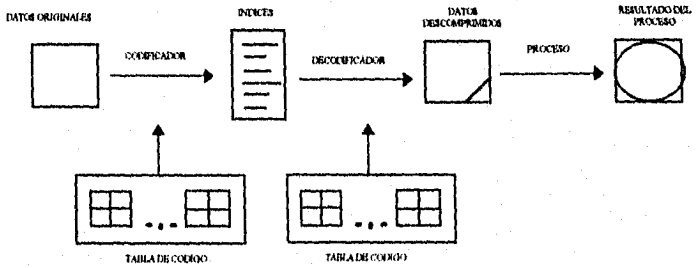


Fig. 4.19 Secuencia de pasos de procesamiento y decodificación con la generación de un archivo intermedio

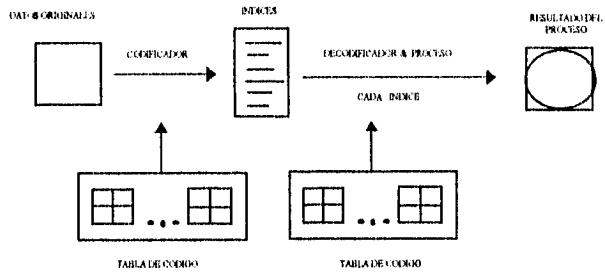


Fig. 4.20 Secuencia de pasos del procesamiento y decodificación sin archivo intermedio.

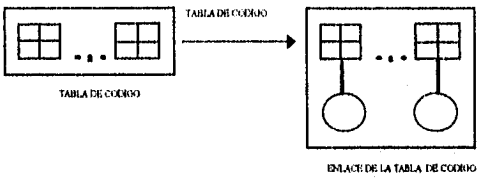


Fig. 4.21 Funcionamiento del procesamiento de la tabla de código (codebook).

El procesamiento termina en el tiempo de diseño de la tabla de código, y el decodificador de la tabla de código contiene las palabras y resultados procesados ligados. Cuando el sistema descomprime una imagen, el decodificador puede dar como salida las palabras de código, el resultado procesado, o ambos, sin necesidad de tiempo adicional.

Un ejemplo trivial de la combinación de VQ y procesamiento de una imagen es *thresholding* (umbral o comparación con un umbral), el cual opera en una imagen de tonos de gris. El VQ es designado de tal forma que, cada pixel en cada palabra es comparado para producir una tabla de códigos ligada. Cuando la tabla es usada para descompresión, el decodificador puede seleccionar la producción de una u otra, la original o las palabras umbrales (thresholded codewords) **[COS95].**

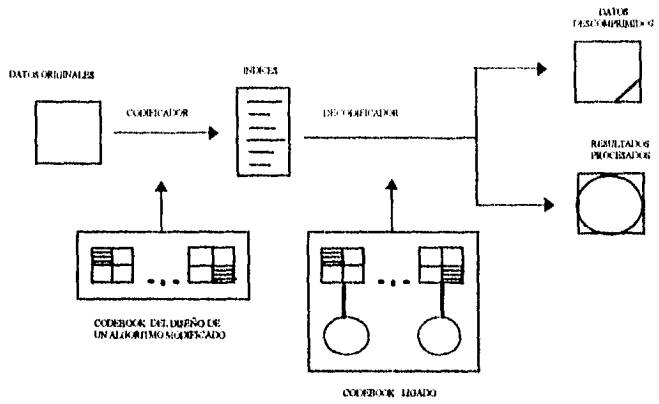


Fig. 4.22 Pasos de Procesamiento y decodificación, donde el codificador del codebook ha sido modificado para conocimiento del procesamiento.

El objetivo del mejoramiento es para acentuar ciertas características o rasgos de la imagen para un subsecuente análisis o despliegue de la misma. Ejemplos que incluyen mejoramiento de contraste, pseudocoloración, filtración de ruido, etc. VQ tiene la habilidad innata para remover cierto tipo de ruido (speckle noise), debido al alisamiento o ejecución promedio para la operación del centroide.

4.6.3 HALFTONING.

Halftoning es el proceso de representar imágenes en escala de grises (normalmente de 8 bpp) para imprimirlos en un dispositivo binario (blanco y negro) tales como impresoras láser. Aunque los pixeles en una imagen halftoning son solo blancos o negros, la ilusión de sombras continuas de gris, puede ser creada por medio de una apropiada elección del porcentaje y patrón de pixeles negros en cada región de la imagen. El porcentaje de datos de una imagen halftoning no comprimida es de 1 bpp, lo cual es relativamente menor que el porcentaje en una imagen en escala de grises. Sin embargo, las imágenes halftoning muy grandes aún requieren de varios Mb de datos para ser transmitidas a una impresora [SOR95].

La compresión de datos puede ser usada para reducir los tiempos de transmisión y los requerimientos de memoria necesarios.

Ordered Dither es una técnica de halftoning simple, que opera independientemente en bloques. En la siguiente ecuación, X es una matriz dither de 4x4 para la aplicación de este proceso a imágenes de 8 bits en escala de grises.

$$X = \begin{pmatrix} 8 & 136 & 40 & 168 \\ 200 & 72 & 232 & 104 \\ 56 & 184 & 24 & 152 \\ 248 & 120 & 216 & 88 \end{pmatrix}$$

La imagen inicial es escaneada en forma no superpuesta con la matriz dither, y cada pixel en escala de gris es comparado con el apropiado valor de umbral en la matriz. Si la intensidad del pixel es mayor al umbral, este es puesto en blanco, en caso contrario es puesto en negro. De esta forma, la decisión para cada pixel depende de su posición en el bloque tanto como el valor de intensidad en escala de grises. La calidad de la imagen halftoning resultante depende del tamaño y los valores de la matriz dither.

Debido a que cada bloque es procesado independientemente, Ordered dither puede ser implementado fácilmente con VQ, si la matriz dither y los bloques VQ son del mismo tamaño (o el tamaño del bloque VQ es un entero múltiplo del tamaño de la matriz dither). Uno aplica la matriz dither a cada vector en la tabla de códigos, para formar tablas de código VQ ligadas, como en la **Fig. 4.22** [COS95], la tabla de código principal contiene los vectores en escala de gris, y la tabla de código procesada contiene vectores binarios.

Difusión de error es un proceso de halftoning de vecindad. Este es más complejo que Ordered dither, pero usualmente presenta una mejor calidad en la imagen cuando la resolución de la impresora es baja (por abajo de 300 dpi). El error entre el pixel de entrada en escala de gris y su salida binaria es esparcido sobre los pixeles vecinos, sumando a la salida una combinación ponderada de los errores de salida provenientes de los pixeles arriba y a la izquierda de la entrada. La entrada modificada es comparada con el umbral fijo para hacer la decisión (blanco o negro). Combinar Difusión de error con VQ puede ser un reto, ya que este es un proceso de vecindad. Sin embargo, este proceso puede conseguir un decremento significativo en la complejidad computacional. Por lo que aún se sigue explorando el problema.

Vander Kam et al, presenta un algoritmo para un cuantizador de unión vectorial y el diseño de un halftoner. Una medida de distorsión de frecuencias ponderadas basado en la respuesta visual humana, es usado tanto en el diseño del VQ como en elegir la reproducción binaria para un vector en escala de grises. El VQ de búsqueda completa es usado para seleccionar el vector binario que medirá la baja distorsión de frecuencia comparada al bloque de salida en escala de gris [COS95].

El algoritmo halftoning esta basado en bloques (como Ordered Dither), así que es fácilmente compatible con VQ, pero es más complejo y de mejor calidad. La calidad de la imagen halftoning comprimida es, por lo tanto, mejorada por medio de incluir información contextual que permita más reproducciones de salida. Vander Kam, encontró que su VQ/halftoner tiene mejores resultados que otros sistemas que tienen el compresor y el halftoner separados. En otros trabajos, mejoraron el funcionamiento del sistema, remplazando el VQ de búsqueda completa por un VQ de entropía restringida. obtuvieron una calidad de imagen muy buena, con porcentajes menores a 0.1 bpp [COS95].

4.6.4 LA CALIDAD DE LA IMAGEN EN LOS CÓDIGOS VQ.

Depende de muchos factores, incluyendo el tamaño del bloque, la cantidad de correlación entre los pixeles, el número de niveles de cuantización, lo adecuado de la tabla de códigos para las imágenes que se codificarán, etc.

En cuanto a la calidad subjetiva de la imagen, utilizando VQ no está definida actualmente; el error cuadrático promedio es el único criterio considerado en la codificación con VQ. Sin embargo, se ha propuesto a la VQ, como una técnica ideal para la codificación y reproducción de imágenes con diferentes características de detalles, texturas, bordes, etc.

Nuestro objetivo fue describir ideas fundamentales de la Cuantización Vectorial y examinar procedimientos en el diseño de algoritmos, los cuales son basados, en el agrupamiento y clasificación de árboles que pueden ser modificados para incorporar el procesamiento de imágenes dentro de VQ. De acuerdo con las funciones principales de estos algoritmos, como son reducir la complejidad del procesamiento, que sigue el paso de descompresión o para proveer una mejor calidad y optimización en las dos operaciones (compresión y descompresión).

De acuerdo con la información investigada, hemos propuesto una aplicación que podría ser usada en la compresión de imágenes binarias. El primer paso para crear dicha aplicación, sería ver que tan viable es crear vectores de pixeles en estas imágenes. Recordemos que en el caso de este tipo de imágenes, cada pixel es representado por un bit. Por esto, la creación de vectores que agrupen bytes, influirían de una manera amplia en la reconstrucción de la imagen. En algunas imágenes de este tipo, se tienen cambios de corridas de bits con bastante frecuencia, lo que dificultaría la creación de la tabla de código.

Se requerirá de un bloque auxiliar en el decodificador, para que la reconstrucción de las imágenes con cambios demasiado bruscos en las corridas de bits, se haga en forma adecuada. Esto es debido a que al perder información, se necesita que esta sea de alguna forma sustituida, para que la pérdida no afecte demasiado a la imagen. En forma general, la modificación al algoritmo planteado, estaría en la etapa auxiliar, que sustituiría la pérdida de información necesaria. En el siguiente capítulo analizaremos el sistema de compresión propuesto por nosotros, aplicado a imágenes binarias y en particular a documentos digitalizados.

CAPÍTULO V.

ANÁLISIS DEL SISTEMA DE COMPRESIÓN

CAPÍTULO V.

ANÁLISIS DEL SISTEMA DE COMPRESIÓN.

5.1 IMPLEMENTACIÓN DEL SISTEMA DE COMPRESIÓN PARA IMÁGENES BINARIAS.

Conociendo la importancia que existe en el manejo de documentos, diseñamos un Sistema de Compresión para imágenes binarias, en particular a documentos digitalizados. Que fuese eficiente, rápido y compatible para el almacenamiento de esta clase de imágenes.

La implementación del Sistema de Compresión, se desarrolla principalmente en los conocimientos teóricos de las principales Técnicas de Compresión, descritas para éste tipo de imágenes.

El sistema consta de dos fases de desarrollo:

La primera, se le llama *Fase de Compresión de la Imagen*, en la cual la imagen en formato raster (.raw) es comprimida a través de un proceso de Codificación, en él se utilizan tres algoritmos los cuales son Cuantización Vectorial, Run-Length y Codificación por Huffman. El módulo de cuantización divide en bloques a la imagen (formando vectores), y toma los más significativos para representar a esta, asignándoles un índice, que serán finalmente los que representen a la imagen, al cambiar vectores por índices. El proceso de codificación se utiliza para asignar un código con el menor número de bits, los necesarios para codificar la imagen. Puesto que hay imágenes (documentos) donde existen muchos espacios del mismo color (en este caso blancos o negros), éstos se pueden reducir dando un formato (RLE). Más tarde se les asignará el código de Huffman, siempre que tengan una probabilidad de ocurrencia mayor que cero.

En la segunda fase, se procede a recuperar la imagen original, que fue comprimida en la fase anterior para ser almacenada. En esta fase se restablecen cada uno de los pixeles que fueron comprimidos por medio de VQ, Código de Huffman y el formato de RLE (**figura 5.1**) para recuperar la imagen original. Por lo tanto, se le llama *Fase de Expansión o Recuperación* de la imagen.

A continuación se muestra un diagrama de bloques de las fases que se desarrollan en este sistema.

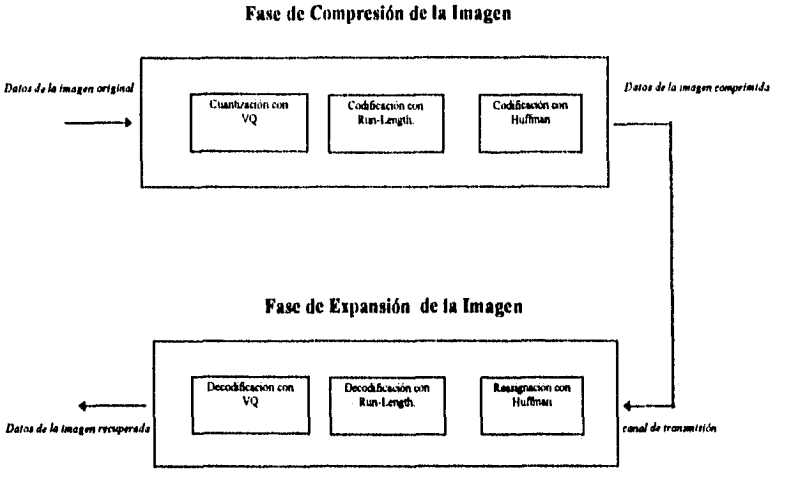


Figura 5.1 Diagrama General del Sistema de Compresión

5.2 MÓDULO DE CUANTIZACIÓN VECTORIAL

El algoritmo de VQ, es considerado como un buen método aplicado al campo de la compresión de imágenes. A continuación se planteará una aplicación para comprimir imágenes de tipo binario. Posteriormente, esta aplicación fue incluida dentro del sistema de compresión propuesto.

El diagrama de bloques del subsistema es el siguiente:

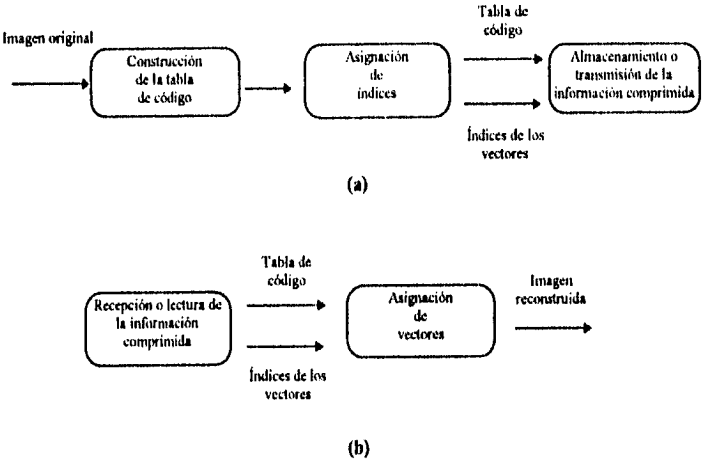


Figura 5.2 Diagrama de bloques del Subsistema.

El módulo VQ a seguir en la realización de este subsistema, es el mismo que el explicado con anterioridad. Con la siguientes especificaciones.

- La dimensión de los vectores es de 8x1 pixeles y el tamaño de los índices es de 4 bits.
- Debido a que se tienen imágenes con un formato de 8 pixeles por byte, se espera una compresión de 2:1.
- Para generar la tabla de código se utilizó el método de seccionamiento por árboles binarios de 4 niveles, y teniendo por lo tanto 16 niveles de cuantización.
- Una mejora que se le hizo fue incluir directamente a la tabla de código, aquellos vectores cuya probabilidad de ocurrencia fuera mayor a 0.25.
- La asignación de índices se hace en forma secuencial.

Pasos del proceso.

En la **figura 5.3**, se muestra los pasos realizados en el proceso de cuantización.

El proceso inicia con la entrada de los datos que serán codificados (1), así como el número de renglones y columnas que contiene nuestra imagen. Se toman vectores de 8x1 pixeles, para encontrar la distribución de probabilidad de cada uno de ellos (2). Las probabilidades de cada vector son almacenadas en un arreglo, para después ser mandadas como parámetro a la función no recursiva del proceso (3).

En esta función, se recorre el arreglo (3.1) hasta encontrar vectores que tengan una probabilidad de ocurrencia mayor al 25%. Estos vectores son incluidos dentro de la tabla de códigos (3.2), ya que los consideramos elementos muy representativos en la imagen. El proceso cumple la tarea de balancear la distribución de probabilidad de la imagen, eliminando aquellos vectores excesivamente representativos.

La función recursiva del módulo, es la que utiliza el algoritmo de cuantización vectorial. El proceso inicia encontrando un punto medio (*centroide*) del total de la imagen (3.4.1), para posteriormente dividir en dos partes el arreglo de probabilidades a partir de este (3.4.3). Las partes obtenidas son, a su vez, pasadas como parámetro a esta misma función (3.4.4), para que mediante recursividad se obtengan centroides representativos de subgrupos de la imagen total, utilizando un árbol para vaciar cada uno de los centroides . Cuando se ha llegado a un determinado nivel del árbol, se incluyen los centroides obtenidos en este nivel, dentro de la tabla de código. El proceso concluye cuando se hayan generado todos los niveles de cuantización que se establecieron, obteniendo de esta forma la tabla de código completa.

Se toman uno a uno los vectores originales (4) y se rastrea en la tabla de código el vector que se adapte mejor a las características (5) del original. Se utiliza el error cuadrático promedio para asignar un determinado vector de la tabla a uno original. Ya que se tiene el vector asociado, se transmite o se guarda el índice de este, dependiendo de la aplicación (7).

El módulo concluye al terminarse el número de vectores originales, cerrando el archivo o finalizando la transmisión.

Como ya se ha explicado, el método de cuantización vectorial aplicado a la compresión, implica pérdida en los datos reconstruidos. Debido a esto, introduciremos un nuevo concepto para poder calificar el rendimiento del sistema realizado: *distorsión.*

En nuestro caso, la distorsión la consideraremos como la diferencia existente entre un pixel determinado contenido en la imagen original y su correspondiente en la imagen reconstruida. Para poder medir este parámetro utilizaremos la siguiente fórmula.

$$DT = \frac{i}{N_t} \sum_{i=0}^{N_t} \left(x_i - x'_i \right)^2$$

donde:

N_t = Número total de pixeles.
x_i = i-ésimo pixel en la imagen orlginal.
x'_i = i-ésimo pixel en la imagen reconstruida.

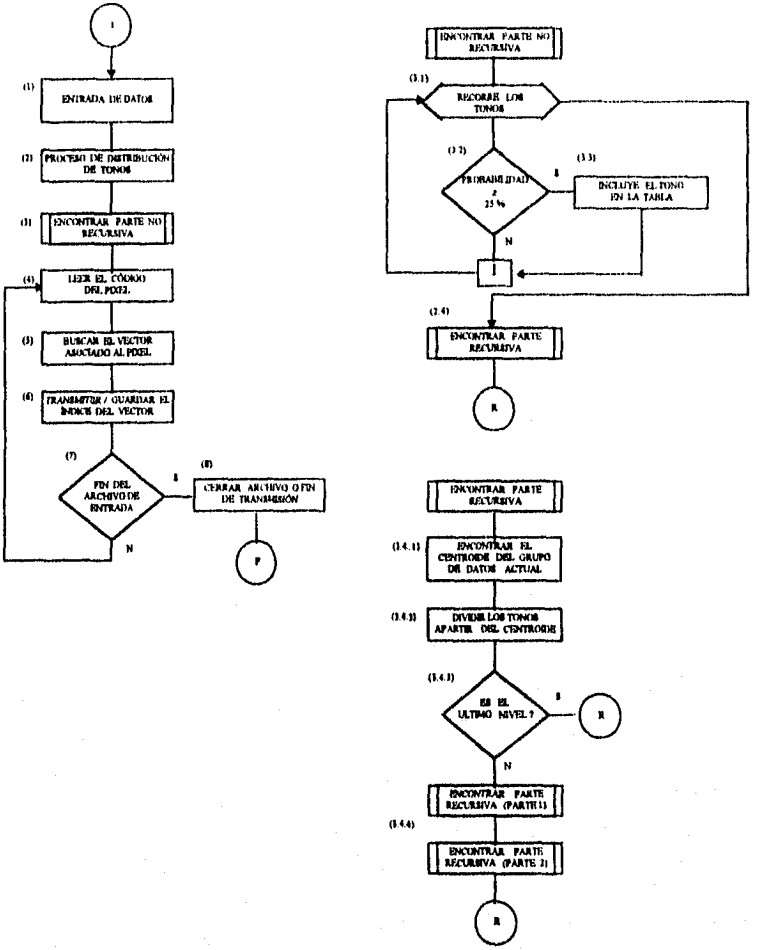


Fig. 5.3 Diagrama de la primera fase del sistema (Cuantización Vectorial).

5.3 MÓDULO DE CODIFICACIÓN RUN-LENGTH

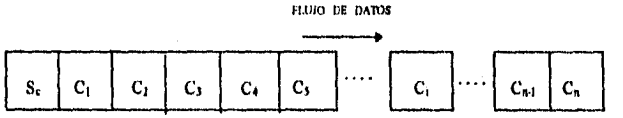
En el proceso de Codificación, se dará un formato RLE a todos los datos originales que son repetidos, en donde sí se tiene una cadena del tipo "11111100001100000", se cuantifica los datos en una lista de la siguiente forma:

6 "1"
4 "0"
2 "1"
6 "0"

resultando 61402160, es decir la cadena se sustituye por un simple valor (que representa la información 0 ó 1) y un contador (que lleva el número de ocurrencias del dato) como se muestra en el siguiente formato RLE **(figura 5.4)**.

Por lo que es más recomendado para cadenas largas de 1 ó 0 repetidos (números codificados en bits o bytes). Ejemplo de estas cadenas son grandes espacios en blanco ó grandes regiones rellenadas de negro.

Específicamente tiene la finalidad de reducir la redundancia de información de una manera muy eficiente.



S_c = Carácter que define con que dato inicia la secuencia (0 ó 1).
C_i = Contador, este contador es el número de tiempos de caracteres comprimidos que son repetidos
n = Número máximo de caracteres en la secuencia.

Figura 5.4 Formato RLE.

Pasos del proceso

En la **figura 5.5,** se muestra los pasos realizados en el proceso de codificación.

Un contador (1) y un identificador (2) son inicializados a cero. El identificador nos indicará de que tono es la corrida (1's ó 0's).

Después es obtenido un carácter de la cadena de datos original (3). Entonces el contador es comparado con 254 (4), ya que este es el número máximo para representar una corrida. De ser así, se verifica si el identificador corresponde al mismo tono del carácter que llegó (5). De ser verdad, se da por entendido que la corrida continua, y por lo tanto se almacena un carácter de continuación (255), que nos indica precisamente este hecho(6). De lo contrario, la corrida se almacena con un carácter de finalización (254), que nos especifica que la siguiente corrida es de un tono diferente(7). De esta forma también se cambia el identificador al tono contrario(8). En ambos casos se reinicializa el contador a cero.

Si el contador aún no ha llegado al máximo, entonces se pregunta por si existe un cambio de tono(9). Si este es al caso, se almacena el contador(10) y se cambia el identificador de tono.

Si aún existen caracteres para formatear, se incrementa el contador en uno (11). Después de esto se regresa a tomar un nuevo carácter. En caso contrario, se almacena el contador (12), terminando de esta forma el proceso.

En esta parte del algoritmo, en donde se guarda automáticamente las corridas de "1's" ó "0's", dado por establecido, que cada dato corresponde un cambio de color (blanco a negro ó viceversa). Como ya se vio, para los casos en los que el número de corrida rebasa el byte, se usa un código para representar tal situación (255).

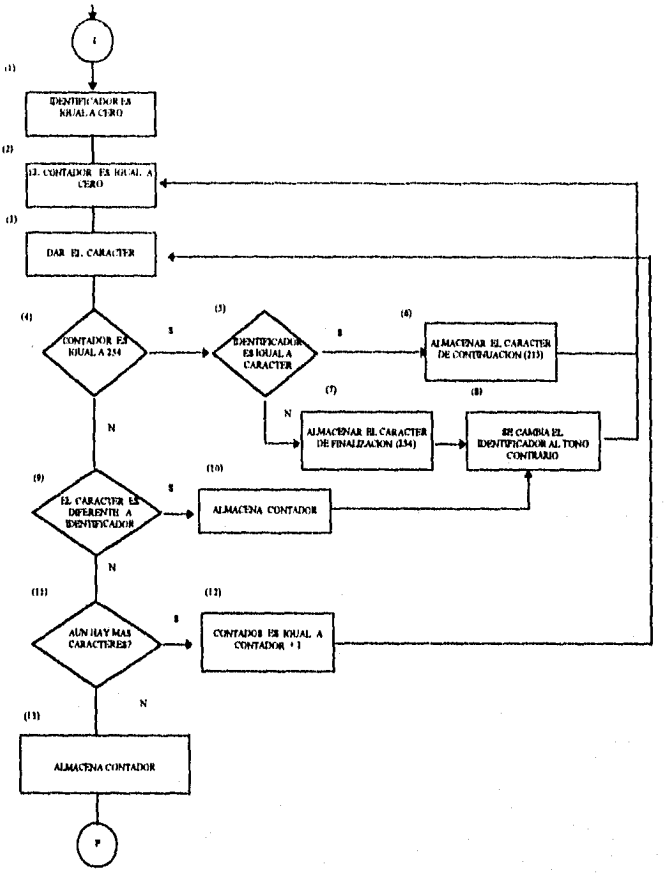


Figura 5.5 Diagrama de la segunda fase del sistema (Run Length).

5.4 MÓDULO DE CODIFICACIÓN POR HUFFMAN

La última etapa del sistema, consiste en codificar los datos obtenidos en el proceso de codificación Run-length, por medio del algoritmo de codificación de Huffman. El proceso tiene la finalidad de asignar códigos de pocos bits, a aquéllos paquetes de pixeles que tengan una alta probabilidad de ocurrencia en la imagen. En nuestro caso, utilizamos paquetes de 8 bits, a los cuales se les deberá asignar un determinado código de Huffman.

La forma de asignar los códigos es unir los paquetes que tienen las dos probabilidades de ocurrencia mas pequeñas, y acomodarlos como si fuera uno solo, dentro de la tabla de probabilidades. Este proceso se realiza recursivamente hasta que se tienen sólo dos probabilidades en la tabla. Después se desempaquetan estos últimos en sus componentes originales, asignando un bit al código de cada componente, cada que se regrese a un nivel anterior. Al final se tiene que los paquetes están representados por un código binario de longitud variable. Un ejemplo de este proceso se ve en la **Figura 5.6**

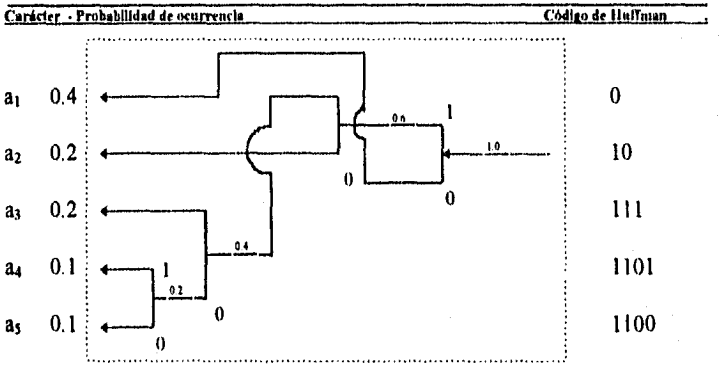


Figura 5.6 Tabla de Códigos y la generación del árbol binario para el Código de Huffman

Pasos del proceso

En la **figura 5.7**, se muestra los pasos realizados en el proceso de codificación.

El proceso inicia con la entrada de los datos a codificar, en paquetes de 8 bits (1). Se realiza un proceso para encontrar la probabilidad de ocurrencia de cada paquete (2). Aquellos paquetes con probabilidad mayor a cero, son guardados en una lista, la cual se ordena de forma ascendente (3). Cada nodo de la lista esta formado por el paquete de bits y su probabilidad.

Cuando ya se ha terminado de ordenar la lista, se suman las dos probabilidades más pequeñas, generando con ellas un subárbol, donde la raíz es un nodo que contiene el resultado de la suma y las hojas son los nodos cuyas probabilidades se sumaron (4). Este subárbol es ordenado en la lista en forma ascendente, tomando como nodo de ordenamiento la raíz (5). El proceso termina, cuando la lista queda convertida en un árbol, es decir cuando sólo hay un nodo en la lista, que será la raíz del árbol (6).

El paso siguiente es asignar el código de huffman a cada paquete (7), para lo cual se recorre el árbol en orden. Al bajar de nivel se le asigna un bit al código del paquete (si es una rama derecha se le asigna "1" (6.2), en caso contrario se asigna un "0" (6.4)). Al final del proceso se tienen los códigos para cada uno de los paquetes.

La siguiente función (8) vacía el contenido del árbol en una lista, la cual será nuestra tabla de códigos, que se guardará como encabezado del archivo, o bien se transmitirá por el canal.

Finalmente se toma uno a uno los paquetes que componen la entrada (9). Se busca en la tabla el código asociado a cada paquete (10), para posteriormente transmitirlo o guardarlo en el archivo de salida (11). Cuando todos los paquetes han sido codificados, se cierra el archivo o se finaliza la transmisión.

Después de que la imagen raster original (.raw) ha recorrido estas tres etapas, se obtiene una imagen comprimida contenida en un archivo con el mismo nombre y de extensión "**.ibc**" (*imagen binaria comprida*). El cual puede ser transmitido o almacenado en una unidad de memoria secundaria.

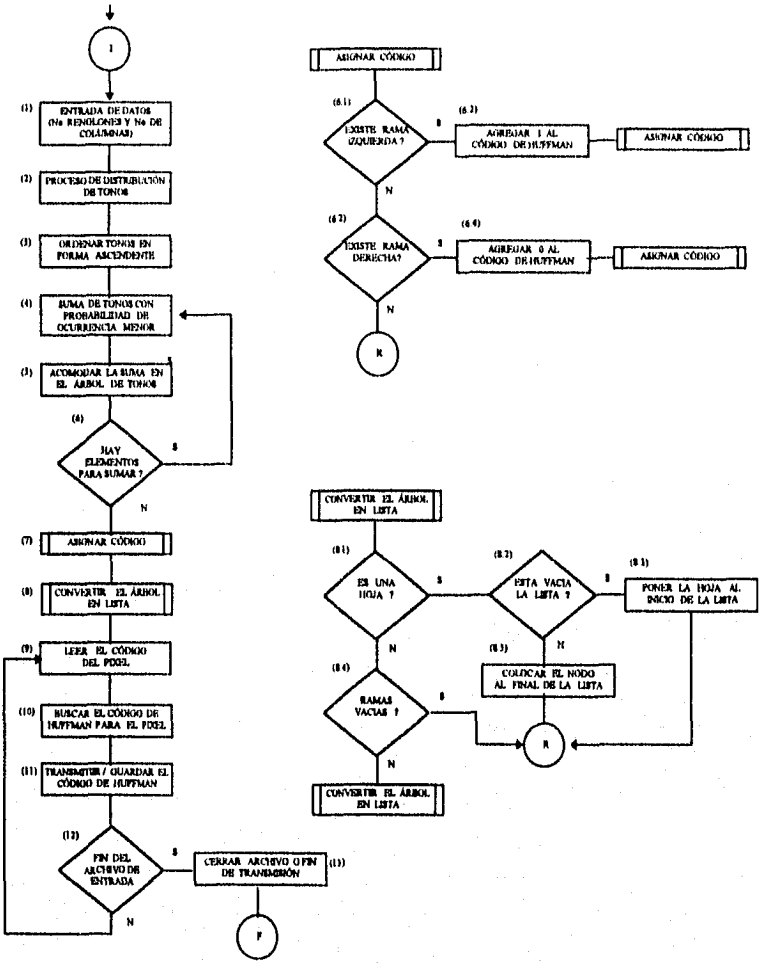


Figura 5.7 Diagrama de la tercera fase del sistema (Código de Huffman).

5.5 DESCOMPRESIÓN

En la fase de descompresión es inverso al anterior, es decir los datos comprimidos son expandidos por medio de un proceso de decodificación. Se van tomando los datos codificados, byte por byte, asignándoles, de acuerdo con Huffman, su correspondiente valor en la tabla de código. Los datos obtenidos de esta forma, son guardados en un archivo intermedio (con extensión *.rle). Después pasa por el segundo decodificador para restablecerlos con corridas de 1's ó 0's, entendiendo que a cada cambio de dato corresponde un cambio de tono. Finalmente se busca dentro de la tabla de código de cuantización vectorial, los vectores que corresponden a los índices que se tienen como resultado de la decodificación anterior. Después de esto, tendremos otra vez la imagen original con una cierta distorsión.

A continuación se muestra un diagrama en el que se representa el procedimiento de Descompresión

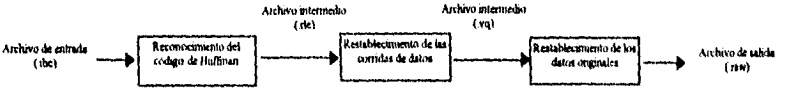


Figura 5.8 Proceso de decodificación del sistema

La simulación del algoritmo de compresión anteriormente presentado fue desarrollado usando como plataforma: estaciones de trabajo SUN, y lenguaje de programación C.

5.6 ANÁLISIS Y EVALUACIÓN DE RESULTADOS.

Los resultados obtenidos, nos permitirán llevar a cabo una evaluación del algoritmo en términos de compresión y de la calidad subjetiva de la imagen.

5.6.1 DOCUMENTOS ESTÁNDARES DE LA CCITT (como casos de estudio)

A continuación se muestran los estándares utilizados por la CCITT (**ver figuras 5.9a hasta 5.16b**, el inciso "a" será para imágenes originales, y el inciso "b" para imágenes comprimidas.). Las cuales serán utilizadas en la evaluación del sistema, siendo estas una aplicación más real a imágenes binarias.

Cada una, cuenta con un formato de 1728 x 2376 pixeles, bit map (mapa de bits), un tamaño de 513 216 bytes, se muestran en tamaño real a un documento, cada uno de ellos se distingue por la dispersión de blancos y negros, algunos son más concentrados en negros y otros son menos, según sea la imagen de la que se trate. También se distingue que se puede comprimir más, cuando hay mayor cantidad de negros o blancos, pues el sistema comprime cuando haya más corridas de negros o blancos.

Estás imágenes fueron obtenidas en la siguiente dirección de Internet:

http: //nic.funet.fi/pub/graphics/misc/test-images

Nombre de la Imagen	*Contenido de la imagen*
CCITT1	Ejemplo de carta
CCITT2	Diseño de un circuito
CCITT3	Factura
CCITT4	Párrafo de un escrito
CCITT5	Página de un libro
CCITT6	Gráfica estadística
CCITT7	Alfabeto Oriental
CCITT8	Memorándum / título

TABLA 5.1

THE SLEREXE COMPANY LIMITED

SAPORS LANE - BOOLE - DORSET - BH25 8ER

TELEPHONE BOOLE (0945 13) 51617 - TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

Registered in England: No. 00000
Registered Office: 00 Victoria Lane, Third Town.

Fig. 5.9 a Muestra la imagen binaria CCITT1 de 1728 x 2376 pixeles.

THE SILREXE COMPANY LIMITED

[illegible]

Our Ref. 350/PJC/EAC 18th January, 1972.

Mr. P.B. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronized with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

[illegible]

Fig. 5.9 b Muestra la imagen binaria CCITT1 recuperada, con una razón de compresión de 18.85 : 1.

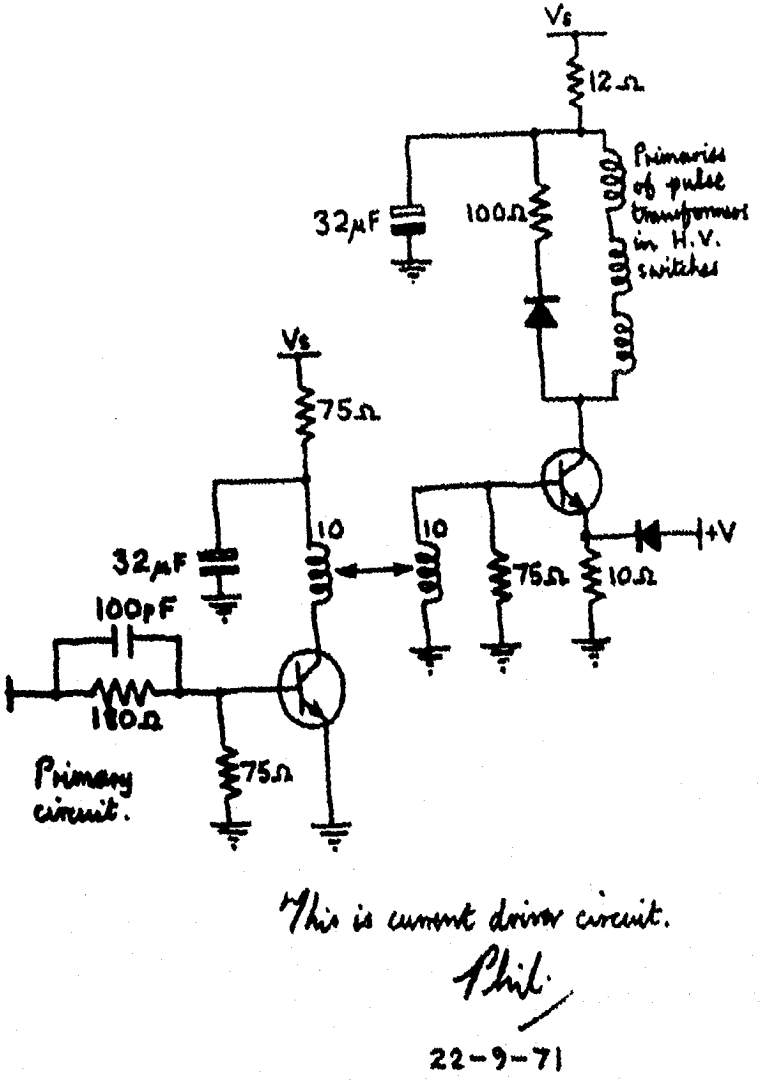


Fig. 5.10 a Muestra la imagen binaria CCITT2, de 1728 x 2376 pixeles.

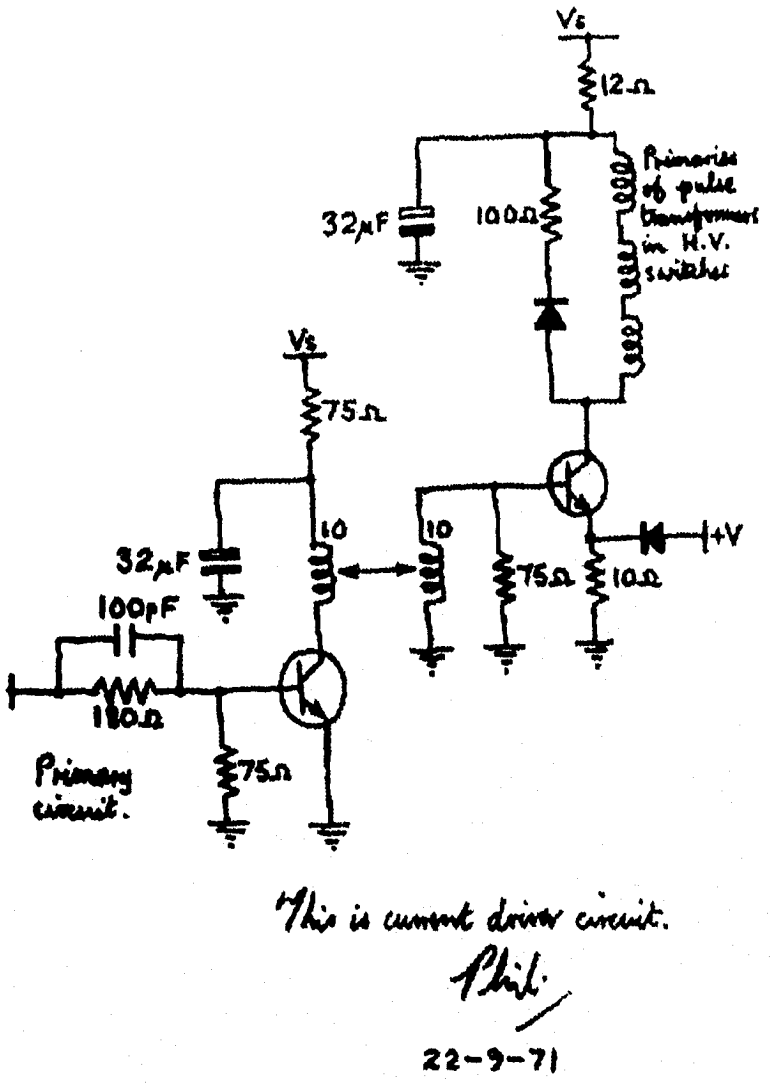
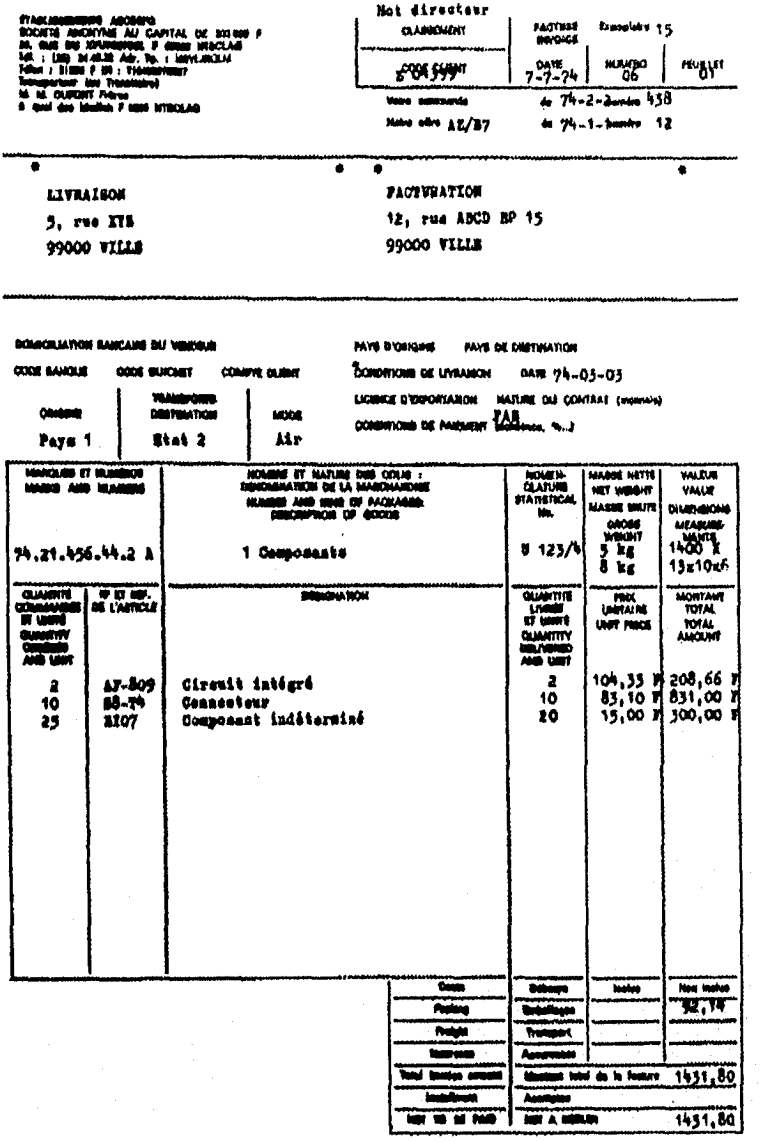


Fig. 5.10 b Muestra la imagen binaria CCITT2 recuperada, con una razón de compresión de 18.43 : 1.



Mot directeur

CLASSEMENT

CODE CLIENT 6 04 599

FACTURE ENVOYÉE — Exemplaire 15

DATE 7-7-74 — NUMÉRO 06 — FEUILLET 01

Votre commande du 74-2-2ème 438

Notre réf. AZ/B7 du 74-1-1ère 12

LIVRAISON
3, rue XYS
99000 VILLE

FACTURATION
12, rue ABCD BP 15
99000 VILLE

DOMICILIATION BANCAIRE DU VENDEUR — PAYS D'ORIGINE — PAYS DE DESTINATION

CODE BANQUE — CODE GUICHET — COMPTE CLIENT

CONDITIONS DE LIVRAISON — DATE 74-05-03

LICENCE D'EXPORTATION — NATURE DU CONTRAT (marché)

CONDITIONS DE PAIEMENT FAB (traites, ...)

Origine	Transporteur destination	Mode
Pays 1	Etat 2	Air

MARQUES ET NUMÉROS MARKS AND NUMBERS	NOMBRE ET NATURE DES COLIS ; DÉNOMINATION DE LA MARCHANDISE NUMBER AND KIND OF PACKAGES; DESCRIPTION OF GOODS	NOMENCLATURE STATISTICAL No.	MASSE NETTE NET WEIGHT / MASSE BRUTE GROSS WEIGHT	VALEUR VALUE / DIMENSIONS MEASUREMENTS
74.21.456.44.2 A	1 Composants	8 123/6	5 kg 8 kg	1400 F 15x10x6

QUANTITÉ COMMANDÉE ET UNITÉ QUANTITY ORDERED AND UNIT	RÉF. ET DÉSIGNATION DE L'ARTICLE	DÉSIGNATION	QUANTITÉ LIVRÉE ET UNITÉ QUANTITY DELIVERED AND UNIT	PRIX UNITAIRE UNIT PRICE	MONTANT TOTAL TOTAL AMOUNT
2	AF-809	Circuit intégré	2	104,33 F	208,66 F
10	88-74	Connecteur	10	83,10 F	831,00 F
25	B107	Composant indéterminé	20	15,00 F	300,00 F

Poste	Emballage	Inclus	Non inclus 92,74
Fret	Transport		
Assurance	Assurance		
Total frais annexes	Montant total de la facture		1431,80
Escompte	Acompte		
NET TO BE PAID	NET A RÉGLER		1431,80

Fig. 5.11 a Muestra la imagen binaria CCITT3, de 1728 x 2376 pixeles.

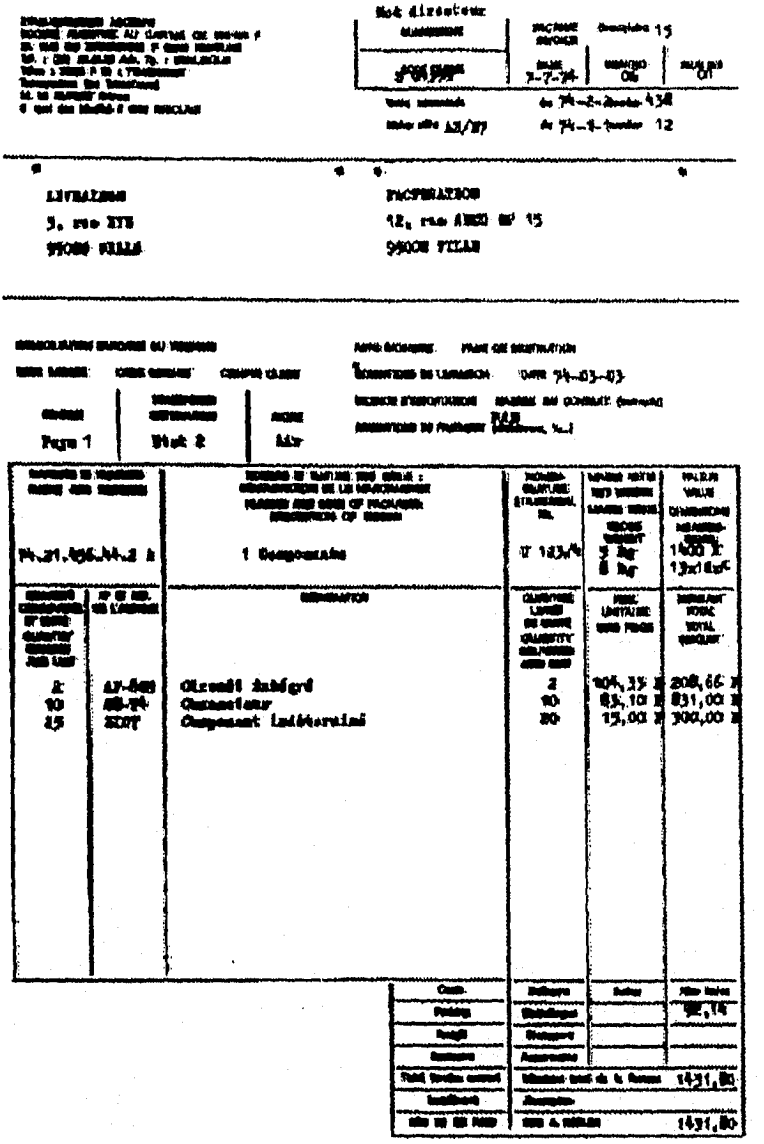


Fig. 5.11 b Muestra la imagen binaria CCITT3 recuperada, con una razón de compresión de 12.18 : 1.

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de construire ce système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, six en sont au stade de l'exploitation, six autres se sont vu donner la priorité pour leur réalisation.
Chaque application est confiée à un "chef de projet", responsable successivement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend des résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger le document de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet, et comporte obligatoirement un bon analyste affecté à l'application.

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 20 régions. Des calculateurs ont été implantés dans le passé au moins dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Massy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment ou sont encore en service dans les régions de Nancy, Nantes, Limoges, Poitiers et Rouen ; ce parc est essentiellement utilisé pour la comptabilité téléphonique.
A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.
Le volume total de caractères à gérer en phase finale sur un ordinateur ayant un charge quelconque 500 000 abonnés a été estimé à un milliard de caractères au moins. Au moins le tiers des données seront concernées par des traitements en temps réel.
Aucun des calculateurs énumérés plus haut ne permettait d'envisager de tels traitements.
L'intégration progressive de toutes les applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement nationaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets des services d'abonnement, les services de personnel etc.
L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation du système informatif. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.
L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informaticiens; et d'autre part le refus de créer des centres trop importants difficiles à gérer et à diriger, et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gérera" environ un million d'abonnés à la fin du VIème Plan.
La mise en place de ces centres a débuté au début de l'année 1971 : un ordinateur IRIS 80 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux.

Photo n° 1 - Document très dense lettre 1,5mm de haut -
Restitution photo n° 9

Fig. 5.12 a Muestra la imagen binaria CCITT4, de 1728 x 1376 pixeles.

- 54 -

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de concevoir un système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, cinq sont au stade de l'exploitation, six autres en sont en donner la priorité pour leur réalisation.

Chaque application est confiée à un "chef de projet", responsable essentiellement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend de ses résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger la documentation de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet,et comporte obligatoirement un bon analyste attaché à l'application.

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 28 régions. Des calculateurs ont été implantés dans le passé en même dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Nancy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment en cours d'amortissement et servaient dans les régions de Nancy, Rennes, Limoges, Poitiers et Rouen ; on peut les essentiellement utilisé pour la comptabilité téléphonique.

A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.

Le volume total de caractères à gérer en phase finale sur un ordinateur ayant en charge quelques 300 000 abonnés a été estimé à un milliard de caractères en moins. Au moins le tiers des données seront renouvelés par des traitements en temps réel.

Aucun des calculateurs dans le plus haut de permettait d'envisager de tels traitements.

L'intégration progressive de nouvelles applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement régionaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets de services d'abonnement, les services de personnel etc.

L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation de ces applications informatiques. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.

L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informaticiens; et d'autre part la notion de créer des centres trop importants difficiles à gérer et à diriger,et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gérera" environ un million d'abonnés à la fin du Vième Plan.

La mise en place de ces centres a débuté au début de l'année 1974 : un ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux.

Photo nº 1 - Document très dense lettre 1,5mm de haut -
Restitution photo nº 9

Fig. 5.12 b Muestra la imagen binaria CCITT4 recuperada, con una razón de compresión de 7.16 : 1.

Cela est d'autant plus valable que $T\Delta f$ est plus grand. A cet égard la figure 2 représente la vraie courbe donnant $|\phi(f)|$ en fonction de f pour les valeurs numériques indiquées page précédente.

Fig. 2

Dans ce cas, le filtre adapté pourra être constitué, conformément à la figure 3, par la cascade :

— d'un filtre passe-bande de transfert unité pour $f_0 \leqslant f \leqslant f_0+\Delta f$ et de transfert quasi nul pour $f < f_0$ et $f > f_0+\Delta f$, filtre ne modifiant pas la phase des composantes le traversant ;

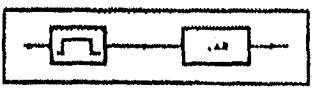


Fig. 3

— filtre suivi d'une ligne à retard (LAR) dispersive ayant un temps de propagation de groupe T_g décroissant linéairement avec la fréquence f suivant l'expression :

$$T_g = T_0+(f_0-f)\frac{T}{\Delta f} \quad (\text{avec } T_0 > T)$$

(voir fig. 4).

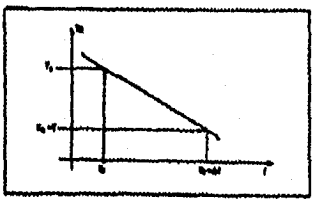


Fig. 4

telle ligne à retard est donnée par :

$$\varphi = -2\pi \int_0^f T_g \, df$$

$$\varphi = -2\pi\left[T_0+\frac{f_0 T}{\Delta f}\right] f + \pi \frac{T}{\Delta f} f^2$$

Et cette phase est bien l'opposé de $|\phi(f)|$,

à un déphasage constant près (sans importance) et à un retard T_0 près (inévitable).

Un signal utile $S(t)$ traversant un tel filtre adapté donne à la sortie (à un retard T_0 près et à un déphasage près de la porteuse) un signal dont la transformée de Fourier est réelle, constante entre f_0 et $f_0+\Delta f$, et nulle de part et d'autre de f_0 et de $f_0+\Delta f$, c'est-à-dire un signal de fréquence porteuse $f_0+\Delta f/2$ et dont l'enveloppe a la forme indiquée à la figure 5, où l'on a représenté simultanément le signal $S(t)$ et le signal $S_1(t)$ correspondant obtenu à la sortie du filtre adapté. On comprend le nom de récepteur à compression d'impulsion donné à ce genre de filtre adapté ; la « largeur » (à 3 dB) du signal comprimé étant égale à $1/\Delta f$, le rapport de compression est de $\frac{T}{1/\Delta f} = T\Delta f$

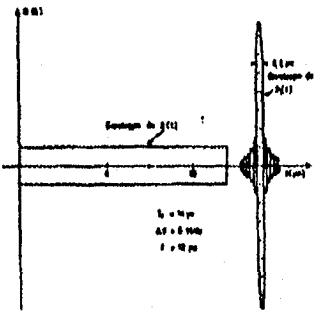


Fig. 5

On saisit physiquement le phénomène de compression en réalisant que lorsque le signal $S(t)$ entre dans la ligne à retard (LAR) la fréquence qui entre la première à l'instant 0 est la fréquence basse f_0, qui met un temps T_0 pour traverser. La fréquence f entre à l'instant $t = (f-f_0)\frac{T}{\Delta f}$ et elle met un temps $T_0-(f-f_0)\frac{T}{\Delta f}$ pour traverser, ce qui la fait ressortir à l'instant T, évidemment. Ainsi donc, le signal $S(t)$

Fig. 5.13 a Muestra la Imagen binaria CCITT5, de 1728 x 2376 pixeles.

Cela est d'autant plus valable que $T\Delta f$ est plus grand. A ces égard la figure 2 représente la seule courbe donnant $|\phi(f)|$ en fonction de f pour les valeurs numériques indiquées page précédente.

Fig. 3

Dans ce cas, le filtre adapté pourra être constitué, conformément à la figure 3, par la cascade :

— d'un filtre passe-bande de transfert nulle pour $f_0 < f < f_0 + \Delta f$ et de transfert quasi nul pour $f < f_0$ et $f > f_0 + \Delta f$, filtre ne modifiant pas la phase des composantes le traversant ;

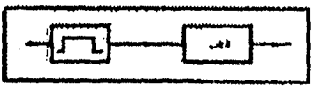


Fig. 3

— filtre retard d'une ligne à retard (LAR) dispersive ayant un temps de propagation de groupe T_0 décroissant linéairement avec la fréquence f suivant l'expression :

$$T_0 = T_0 + (f_0 - f)\frac{T}{\Delta f} \quad (\text{avec } T_0 > T)$$

(voir fig. 4).

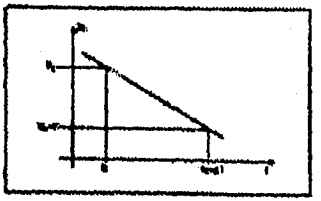


Fig. 4

Cette ligne à retard est donnée par :

$$\varphi = -\int_0^f T_0\, df$$

$$\varphi = -2\pi\left[T_0 + \frac{(f_0 T)}{\Delta f}\right]f + \pi\frac{T}{\Delta f}f^2$$

Et cette phase est bien l'opposé de $\arg C(f)$.

à un déphasage constant près (sans importance) et à un retard T_0 près (inévitable).

Un signal utile $S(t)$ traversant un tel filtre adapté obtient à la sortie (à un retard T_0 près et à un déplacement près de la fréquence) un signal dont la transformée de Fourier est nulle, constante entre f_0 et $f_0 + \Delta f$, et nulle en tout et d'autre de f_0 et de $f_0 + \Delta f$, c'est-à-dire un signal de fréquence pratique $f_0 + \Delta f/2$ et dont l'enveloppe a la forme indiquée à la figure 5, où l'on a représenté simultanément le signal $S(t)$ et le signal $S_1(t)$ correspondant obtenu à la sortie du filtre adapté. On comprend le rôle du récepteur à compression d'impulsion donné à ce genre de filtre adapté : la « hauteur » (à 3 dB) du signal comprimé étant égale à Δf, le rapport de compression est $\Phi = \frac{T}{1/\Delta f} = T\Delta f$

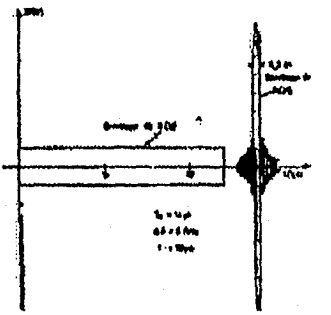
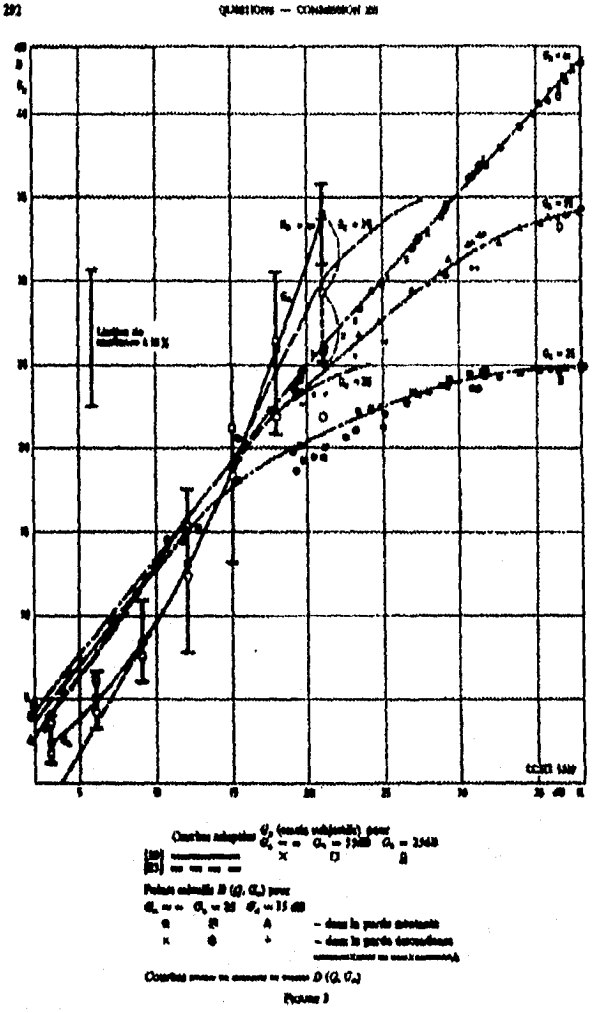


Fig. 5

On voit physiquement le phénomène de compression en réalisant que lorsque le signal $S(t)$ arrive dans la ligne à retard (LAR) la fréquence qui entre la première à l'instant 0 est la fréquence basse f_0, qui met un temps T_0 pour traverser. La fréquence f entre à l'instant $x = (f - f_0)\frac{T}{\Delta f}$ et elle met un temps $T_0 - (f - f_0)\frac{T}{\Delta f}$ pour traverser, ce qui la fait ressortir à l'instant T_0 finalement. Ainsi donc, le signal $S(t)$

Fig. 5.13 b Muestra la imagen binaria CCITT5 recuperada, con una razón de compresión de 11.19 : 1.



292 QUESTIONS — COMMISSION XII

TOME V — Question 10/XII, Annexe 6

Fig. 5.14 a Muestra la imagen binaria CCITT6, de 1728 x 2376 pixeles.

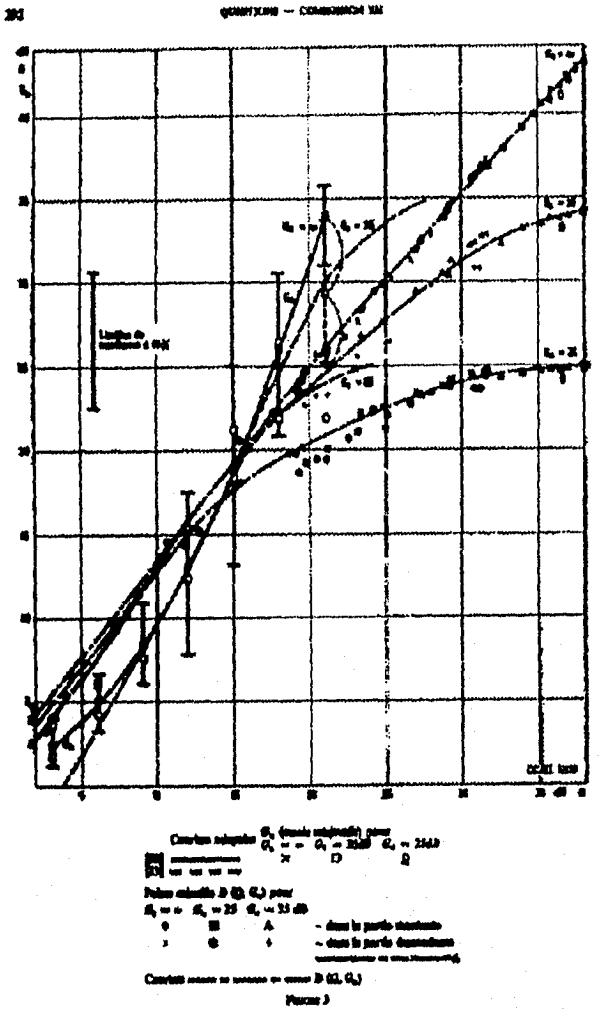


Fig. 5.14 bMuestra la Imagen binaria CCITT6 recuperada, con una razón de compresión de 13.51 : 1.

Fig. 5.15 a Muestra la Imagen binaria CCITT7, de 1728 x 2376 pixeles.

Fig. 5.15 bMuestra la imagen binaria CCITT7 recuperada, con una razón de compresión de 5.86 : 1.

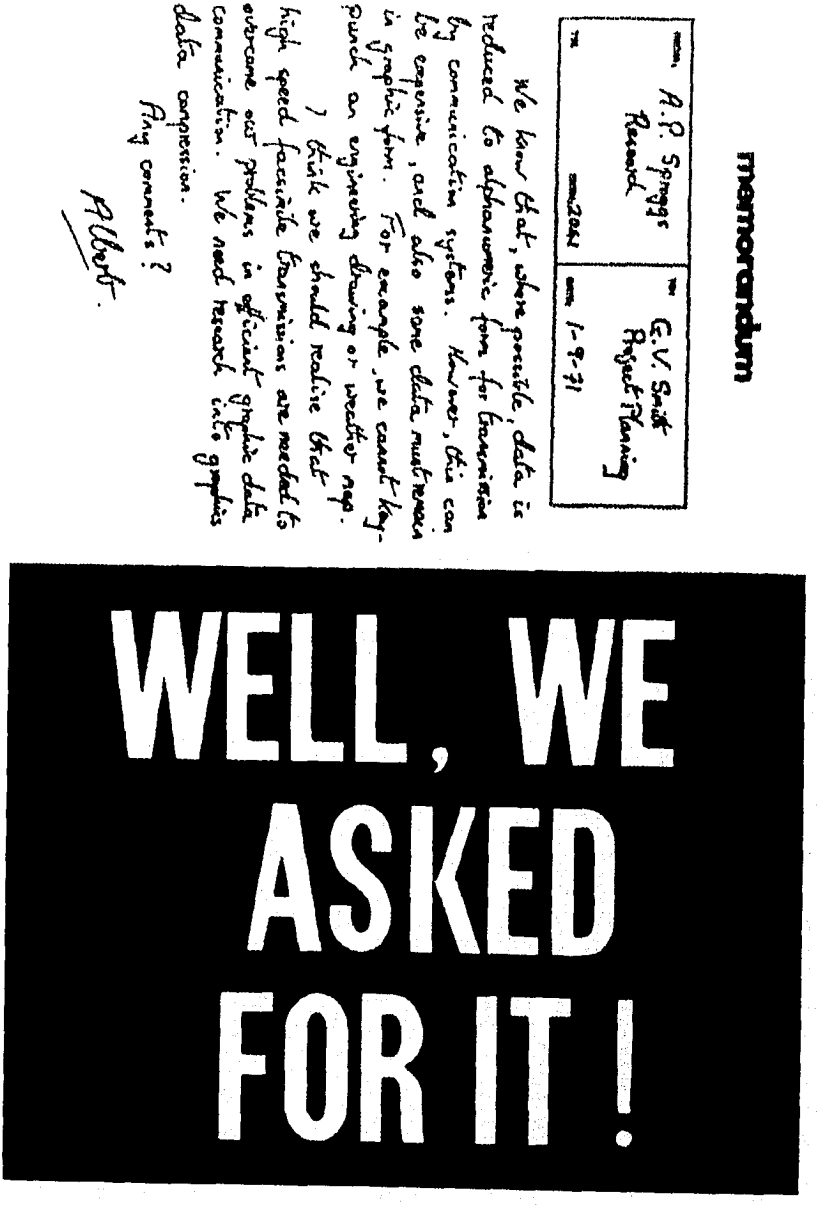


Fig. 5.16 a Muestra la imagen binaria CCITT8, de 1728 x 2376 pixeles.

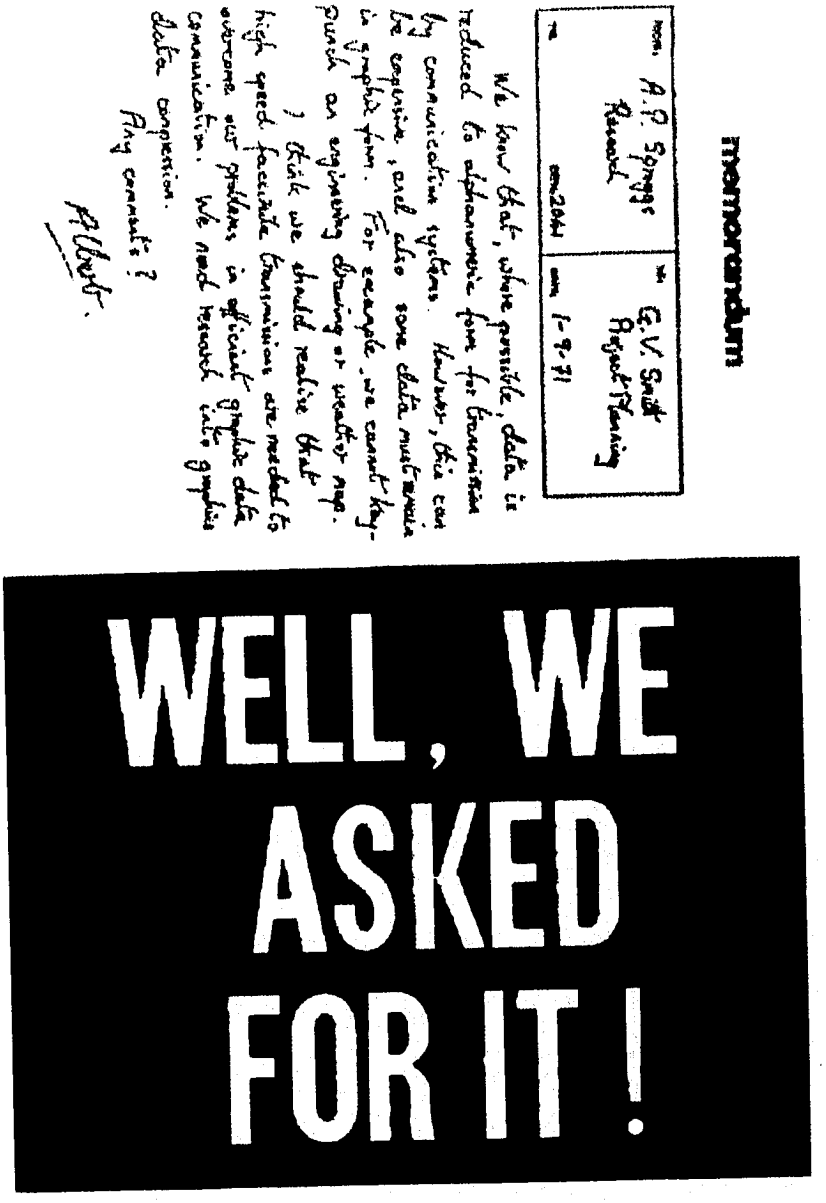


Fig. 5.16 b Muestra la imagen binaria CCITT8 recuperada, con una razón de compresión de 4.07 : 1.

5.6.2 RESULTADO DE LA APLICACIÓN DEL SISTEMA A IMÁGENES BINARIAS.

A continuación se muestran tres tablas de resultados (razón de compresión, tiempo de procesamiento y entropía) de los algoritmos por separado **(tablas 5.2 y 5.3)**, y una de la aplicación del sistema de compresión **(tabla 5.4)**.

	RUN_LENGTH		HUFFMAN		CUANTIZACION VECTORIAL	
Documento	*Razón de Compresión*	*Tiempo de procesamiento (seg.)*	*Razón de Compresión*	*Tiempo de procesamiento (seg.)*	*Razón de Compresión*	*Tiempo de procesamiento (seg.)*
CCITT1	8.59	13	5.91	90	2.0	26
CCITT2	13.66	10	6.38	61	2.0	24
CCITT3	5.85	15	4.89	86	2.0	28
CCITT4	2.72	26	3.54	99	2.0	39
CCITT5	5.08	15	4.77	154	2.0	28
CCITT6	8.52	16	5.64	85	2.0	29
CCITT7	3.08	24	3.82	129	2.0	35
CCITT8	8.97	14	4.46	61	2.0	27

* Tiempos de procesamiento, al comprimir y descomprimir

TABLA 5.2

Documento	*Entropía RLE (bpp)*	*Entropía HUFFMAN (bpp)*	*Entropía VQ (bpp)*
CCITT1	0.12	0.17	0.5
CCITT2	0.07	0.16	0.5
CCITT3	0.17	0.20	0.5
CCITT4	0.37	0.28	0.5
CCITT5	0.20	0.21	0.5
CCITT6	0.12	0.18	0.5
CCITT7	0.33	0.26	0.5
CCITT8	0.11	0.22	0.5

TABLA 5.3

SISTEMA DE COMPRESIÓN

Documento	*Razón de Compresión*	*Tiempo de Procesamiento (seg.)*	*Entropía (bpp)*	*Distorsión*
CCITT1	18.85	60	0.053	0.0086
CCITT2	18.43	51	0.054	0.0051
CCITT3	12.18	67	0.082	0.0235
CCITT4	7.16	111	0.139	0.0497
CCITT5	11.19	81	0.089	0.0212
CCITT6	13.51	59	0.074	0.0092
CCITT7	5.86	122	0.171	0.0366
CCITT8	4.07	205	0.246	0.0107

TABLA 5.4

En las tablas podemos observar los parámetros con los cuales calificaremos el desempeño del sistema, al utilizar diferentes tipos de imágenes binarias, con características propias. Los parámetros fueron obtenidos utilizando las siguientes fórmulas:

- Razón de compresión

$$RC = \frac{\text{Núm de pixeles en la imagen original.}}{\text{Núm de pixeles en la imagen recuperada}}$$

- Tiempo de procesamiento

TP = tiempo de terminación del proceso - tiempo de inicio del proceso

- Distorsión

$$DT = \frac{i}{N_i} \sum_{i=0}^{N_i} \left(x_i - x'_i\right)^2$$

donde:

N_i = Número total de pixeles.
x_i = i-ésimo pixel en la imagen original.
x'_i = i-ésimo pixel en la imagen reconstruida.

- Entropía

$$e = \frac{1}{RC}$$

De aquí podemos observar que:

En las imágenes CCITT1, CCITT2 y CCITT6, el sistema actúo de una manera muy satisfactoria, ya que se obtuvieron los mejores resultados posibles (razón de compresión alta, así como tiempo de procesamiento y distorsión bajas). Con lo cual podemos decir, que el sistema es el adecuado para la compresión de imágenes correspondientes a estas 3 clases de estándares.

En las imágenes CCITT3 y CCITT5, el desempeño del sistema es bueno, sin embargo, la distorsión obtenida fue algo alta. Con algunas modificaciones en el módulo de Cuantización Vectorial, se podría mejorar la calidad de la imagen recuperada, sin afectar el tiempo de procesamiento, ni la razón de compresión. De acuerdo con esto, podemos decir que el sistema sería también una buena opción para el manejo de esta clase de imágenes.

En las imágenes CCITT4, CCITT7 y CCITT8, los resultados obtenidos no son del todo satisfactorios. Los tiempo de procesamiento son hasta cierto punto altos, dando un desempeño no tan bueno. Las razones de compresión son menores que en los casos anteriores, aunque de cualquier forma son bastante aceptables. Cabe señalar que las distorsiones obtenidas, sobretodo para la imagen CCITT4, crecieron de manera considerable, dando como resultado una calidad muy baja.

Se tiene que la máxima distorsión obtenida fue de 0.0497 (ccitt1), es decir que el 4.97 % de la imagen fue alterada por el sistema. Sin embargo, se puede notar que la calidad de las imágenes es aceptable. Para las demás imágenes las razones de compresión varían entre 4.07 y 18, con las mínimas distorsiones. Por lo cual el sistema se comporta de una manera eficiente para imágenes con características similares a estas.

Si comparamos la razón de compresión obtenida por el sistema que proponemos, podemos observar que es mayor a la generada por los algoritmos separados. Run-length es la que presenta razones de compresión más altas, sin embargo, en todos los casos, al aplicar nuestro sistema a las imágenes de prueba, se obtiene en promedio una razón de compresión 50 % más grande.

Huffman presenta una compresión en promedio buena, sin embargo, el sistema que proponemos, comprime en mayor forma. Las razones de compresión obtenidas, rebasan en un 60 % las generadas por la Codificación Huffman. En consecuencia la entropía asociada a imágenes modificadas por este algoritmo, toma valores bastante elevados.

La ventaja de estos algoritmos es que no generan distorsión, por lo cual conviene considerar este hecho al utilizar el sistema que proponemos. Al hacer una evaluación sobre los beneficios implícitos en utilizar VQ como parte de la aplicación, obtuvimos que valía la pena arriesgar un poco de claridad en las imágenes, a cambio de obtener mayores razones de compresión. De esta manera diseñamos un algoritmo modificado de Cuantización Vectorial, el cual forma parte del sistema propuesto.

Los resultados obtenidos en la aplicación de nuestro sistema, a las imágenes mencionadas, nos hace considerarlo como un sistema recomendable y útil para el manejo de documentos.

CAPÍTULO VI.

CONCLUSIONES Y PERSPECTIVAS

CAPÍTULO VI.

CONCLUSIONES Y PERSPECTIVAS

Conclusiones de la aplicación del Sistema de Compresión Propuesto a Documentos Digitalizados.

El principal objetivo que se logra con la implementación de nuestro sistema, es la compresión de información de imágenes binarias en particular a documentos digitalizados. Además de que el sistema comprime al máximo, se obtienen imágenes con la mínima distorsión posible, considerando que las técnicas empleadas, sean las más apropiadas, para recuperar la imagen casi al 100%.

Las consecuencias que se obtuvieron del sistema propuesto, aplicándolo a imágenes de documentos estándar son: una razón de compresión promedio de 11.45 a 1, una entropía promedio de 0.1135 bpp (bit por pixel), un porcentaje de reducción del 88.65% y una distorsión promedio de 0.020575.

Los mejores resultados esperados, para identificar un algoritmo de compresión eficiente serían: que no hubiera distorsión, la entropía (número promedio de bits utilizados para codificar) menor de 1 bpp, el porcentaje de reducción entre 50 y 100%, y una tasa de compresión mayor de 1. Como se observa en la **tabla 5.4**, nuestros resultados se encuentran entre las mejores condiciones de compresión, además de que la imagen recuperada es casi igual a la original. Comprobando así que el sistema es eficiente y útil para el manejo de sistemas de información (documentos).

Durante el desarrollo de nuestro sistema se observaron detalles, los cuales se mencionan a continuación, para dar una mejor información de lo que se logra con ellos.

El módulo de Cuantización Vectorial da buenos resultados al ser aplicado como primera etapa en el sistema. A pesar de que este algoritmo nos introduce una distorsión, esta se puede considerar dentro de los límites para el manejo de las imágenes. La ventaja que se obtiene al utilizar VQ como parte del sistema, es una mayor compresión, la cual compensa la distorsión que pueda dar.

Se utiliza el número de niveles de cuantización más óptimo en este módulo, debido a que se toman los vectores de representativos necesarios para recuperar la imagen con el menor número de bits. Siendo 4 niveles de cuantización los que se utilizan, ya que de utilizarse menos, se generaría una distorsión mucho más alta. En caso de que introdujéramos más vectores representativos, necesitaríamos más bits para representar el número de índices requerido, por lo tanto se obtendrían razones de compresión menores.

Al aplicar en segunda instancia Run-length, se obtiene un archivo de tamaño similar al original. Se observa que las imágenes binarias se adaptan al módulo de Run-length, ya que no es necesario identificar el tono de la corrida, debido a la restricción de tener solo 2 tonos. Este módulo sirve más, para adaptar la imagen que va a ser comprimida, que como método de compresión.

En la última etapa de codificación Huffman, es la parte que aporta mayor compresión de los tres módulos, debido a que identifica a los pixeles más frecuentes con el menor número de bits, considerando siempre que el código no se repita para otros datos. De aquí observamos que da una cierta seguridad a la información, ya que, en caso de que ocurra alguna distorsión externa al transmitir la imagen, el algoritmo reconocerá solo las palabras que se encuentren dentro de la tabla de decodificación de Huffman.

Nuestros resultados cumplen con los objetivos de esta tesis, en cuanto a que:

- Se almacena información con la menor cantidad de bits, aprovechando para esto la correlación de los datos, y suprimiendo la información repetida, que sin lugar a duda son formas de redundancia muy usuales en este tipo de imágenes.

- En el sistema propuesto, los datos se comprimen antes de que se almacenen y se descomprimen cuando se recuperan, de acuerdo con la capacidad de los dispositivos de almacenamiento utilizados (el sistema fue implementado en estaciones de trabajo SUN de la DEPFI).

- A consecuencia del cumplimiento de los dos objetivos anteriores es obvio suponer que, comprimiendo la información en el transmisor y descomprimiéndola en el lado del receptor, la adaptamos al canal de comunicación, aumentando con esto las razones de transmisión.

El costo del sistema es una característica muy importante para definirlo como óptimo, compatible, eficiente, rápido, accesible, y económico. Veamos desde este punto de vista:

Si se implementa este sistema a una red de transmisión de documentos, se tendría 88.65% de espacio más en memoria, para disponer de este en otras aplicaciones, o bien para almacenar más imágenes. También se transmitiría en menos tiempo la mayor parte de ellas.

Si se pensaba en comprar un mejor equipo, se puede considerar que tan eficiente y costoso sería, en vez de distribuir mejor la capacidad de memoria, comprimiendo las imágenes.

Para dar cifras exactas de cuanto se ahorraría implantar un sistema así, necesitaríamos aplicarlo en un problema real. En nuestro caso se ha implementado en la red del Departamento de Eléctrica de la DEPFI, para comprobar el funcionamiento del Sistema de Compresión propuesto, del cual no se obtuvieron cifras económicas.

Brindar nuevos conocimientos, de gran utilidad sobre el tema de Compresión de información de imágenes binarias, es nuestro mejor anhelo. Por lo cual, creemos que esta aplicación puede modificarse, para generar un sistema global de manejo de documentos. Aún quedan algunas cosas, que mejorarían el funcionamiento de nuestro sistema, sin embargo, en términos generales cumple con las expectativas planteadas al inicio de la tesis.

GLOSARIO DE TÉRMINOS:

Aliasing: Término en inglés, nos indica las distorsiones tipo escalera que sufre una imagen cuando disminuye o aumenta su resolución espacial.

ASCII: *America Standar Code for Information Interchange;* Código Estándar Americano para el Intercambio de Información. Estándar para la representación de caracteres en modo texto.

Buffer: Localidad de memoria para el almacenamiento temporal de datos.

CAD: *Computer Assisted Design;* Diseño Asistido por Computadora. Son las técnicas matemáticas y software por computadora, para resolver problemas de diseño de ingeniería, arquitectura y otras áreas.

CCITT: Comitte Consult International Telegraphique et Telephonique; Comité Consultor International de Telegrafía y Telefonía. Es un comité europeo para la estandarización de formatos de transmisión y comunicaciones.

DCT: Discrete Cosine Transform; Transformada Coseno Discreta. Técnica matemática para transformar una señal discreta en el tiempo al dominio de la frecuencia discreta.

DEPFI: División de Estudios de Posgrado. Facultad de Ingeniería. Universidad Nacional Autónoma de México.

Dither: Método para disminuir los tonos de una imagen cuando el hardware no es capaz de reproducirlos todos; esta disminución de tonos busca que se reproduzcan todos los tonos combinado los colores disponibles. Un ejemplo de ello es disminuir una imagen de 256 colores a 16 colores.

DFT: *Discrete Fourier Transform;* Transformada Discreta de Fourier. Técnica matemática para transformar una señal discreta en el tiempo al dominio de la frecuencia discreta.

DM *Delta Modulation* Módulación delta. Es una técnica para la conversión Analógica Digital de señales e imágenes.

DPCM: *Diferencial Pulse Code Modulation;* Modulación de Pulsos Diferenciales. Es una técnica para la conversión Analógica / Digital de señales e imágenes, derivada del PCM.

Entropía: Es el número mínimo de bits requerido para representar un símbolo de un alfabeto dado.

Espacio de Color: Representación conceptual gráfica del color, los espacios de color más utilizados son RGB, HSV y HSL.

FFT: *Fast Fourier Transform;* Transformación Rápida de Fourier. Técnica matemática derivada de la Transformada Discreta de Fourier (DFT).

Frame : Imagen individual de una película (por ejemplo una fotografía). La sucesión rápida de frames genera una imagen animada.

GIF: Graphic Interchange Format Formato para el Intercambio de Gráficas, formato gráfico desarrollado por CompuServe.

Imagen: Conjunto de pixeles distribuidos en un plano, cada uno representa una intensidad de color.

Imagen Binaria: Imagen cuyos pixeles toman valores de 0 y 1 para denotar negro y blanco, respectivamente.

Imagen en Tonos de Gris: Imagen cuyos pixeles toman valores en un rango definido de grises (entre 0 y 255).

JPEG: Join Photographic Expert Group, Unión de Grupos Fotográficos de Expertos. Es un comité de estandarización de la CCITT para la compresión de imágenes fotográficas.

Luz: Forma de energía electromagnética definida por una longitud de onda específica.

LZ: Lempel Ziv Técnica adaptativa para la compresión de datos sin pérdida de información, el nombre proviene de las iniciales de los apellidos de sus creadores.

LZW: Lempel Ziv Welch : Técnica adaptativa para la compresión de datos sin pérdida de información el nombre proviene de los apellidos de sus creadores.

Histograma: Representación gráfica de la frecuencia de los pixeles de una imagen.

KLT: *Karhumen Laeve Transform;* Transformada de Karhumen Loeve Técnica matemática para la transformación de bloques de pixeles de un espacio a otro de tal manera que se elimine la correlación que existe entre ellos.

Mapa: Conjunto de valores que conforman a un sistema o técnica para representar una imagen y que se aplica a un dominio para convertirlo en un rango.

Nibbles: Conjunto de 4 bits, un byte contiene 2 nibbles.

Paleta: *Tabla de Lookup* para definir los valores RGB de un pixel , el valor de éste indica la entrada a la tabla.

PCM: *Pulse Code Modulation* , Codificación por Modulación de pulsos, es un técnica para la conversión Analógica/Digital de señales e imágenes.

PCX: Formato gráfico comercial desarrollado por Zsoft.

Procesamiento de Imágenes Digitales: Conjunto de técnicas para el mejoramiento, análisis, reconocimiento, síntesis de imágenes.

Raster: Representación de una imagen en la cual ésta se forma por medio de líneas, cada una de ellas sigue físicamente a la otra.

Razón de Compresión: Medida para la compresión de una imagen con respecto a su original.

RGB: *Red Green and Blue,* Rojo , verde y azul, Espacio de color con forma de cubo, un punto dentro de este cubo representa el color resultado de combinar los tres colores primarios.

RLE: *Run Length.* Codificación de Longitud de corridas. Técnica de compresión sin pérdida de datos, para imágenes monocromáticas y a color.

Tabla de Lookup: Tabla para el mapeo de valores, compuesto por índice de la tabla y el valor que éste representa. Un ejemplo son las paletas para imágenes de 256 colores.

TIFF: *Tag Image File Format,* Formato de Archivo de imágenes etiquetadas. Formato gráfico comercial desarrollado por Hewlett Packard, Microsoft.

VQ: *Vectorial Quantizers,* Cuantizadores vectoriales. Herramientas matemáticas para la compresión de imágenes por medio de bloques.

WHT: *Walsh Hadamard Transform* Transformada de Walsh Hadamard. Técnica para la representación de bloques de pixeles de un espacio a otro.

YIQ: Representación intermedia del color que utiliza los componentes de luminiscencia Y, intensidad Y y saturación Q (cuya combinación de estos dos últimos se conoce como cromaticidad). Es un estándar de vídeo compuesto para una mejor transmisión de las imágenes debido a los anchos de banda menores que ocupan estas señales.

BIBLIOGRAFÍA.

[AHM74] AHMED, T. NATARAJAN, AND K.R. RAO, *"Discrete Cosine Transform"*, IEEE Trans. Comput., vol C-23, pp 90-93, Jan 74.

[CAR95] CARRERA O. EDUARDO , *"Compresión de Imágenes con Técnicas Fractales"* , México, D. F., Facultad de Ingeniería, UNAM (Tesis de Licenciatura), 1995.

[COS95] COSMAN PAMELA C., L. OEHLER KAREN, *"Using Vector Quantization for Image Processing"*, Information processing and management, Aug. 95.

[CRO83] CROCHIERE, R. E., AND RABINER, l. R., *"Multirate digital signal processing"*. Englewood Cliffs, NJ, Prentice Hall, 1983.

[EMB90] EMBREE M. PAUL AND KIMBLE BRUCE, *"C LanguageAlgorithms for Digital Signal Processing"*. Prentice Hall, 1990.

[GON93] GONZÁLEZ C. RAFAEL AND WOODS E. RICHARD, *"Digital Image Processing"*, Addison-Wesley Pubishing company, Inc., 1993. 3a. Edition.

[GON87] RAFAEL C. GONZALEZ AND PAUL WINTZ, *"Digital Image Processing"*, Addison-Wesley Pubishing company, Inc., 1987. second Edition.

[HEL91] HELD GILBERT, *"Data Compression Techniques and Applications, Hardware and Software Considerations"*, 3a. Edition John Wiley & Sons LTD. 1991 New York.

[LIM90] LIM, J. S. *"Two dimenstional signal and image processing"*, Englewood Cliffs, NJ, Prentice Hall, 1990.

[PRA79] PRATT W. K.,*"Image Transmission Techniques"*. NJ, Academic, 1979.

[RAB91] RABBANI MAJID, PAUL W. JONES, *" Digital Image Compression Techniques"*, Eastman Kodak Company, Georgia Institute of Technology, 1991.

[RAO90] RAO, K. R.., P. YIP , *"Discrete Cosine Transform Algorithms, Advantages and Applications"*, Academic Press, Inc., 1990.

[ROS82] ROSENFELD AND A. C. KAK, *"Digital Picture Processing"*, Second Edition, Vol. 1, Academic Press, Inc. 1982.

[SCH93] SCHREIBER WILLIAM F., *"Fundamentals of Electronic Imaging Systems"*, 3a. Edición, Springer Verlag, 1993.

[SHI89] SHI-KOU CHANG, *"Principles of Pictorial Information System Design"*, Englewood Cliffs, NJ, Prentice Hall, 1989.

[SOR96] SORIANO OSCAR, *"Implantación de Técnicas Halftoning para el mejoramiento de la calidad de visualización y de la impresión de imágenes binivel"*, México, D. F., Facultad de Ingeniería, UNAM, 1996 (Tesis de Licenciatura).

[TIM91] BELL TIMOTHY C. AND CLEARY JOHN G. *"Text Compression"*, Prentice Hall, Advance Reference Series Computer Science, 1991

[VEL95] VELÁZQUEZ OSNAYA ALBERTO, *"Compresión de Imágenes utilizando una Descomposición Subbanda por medio de Filtros Espejo en Cuadratura (QMF) no separables"*. México, D. F., Facultad de Ingeniería, UNAM, 1995 (Tesis de Licenciatura).

[WOO91] WOODS, J. W. *"Subband image coding"*, Kluwer Academic Publishers, Inc., 1991.