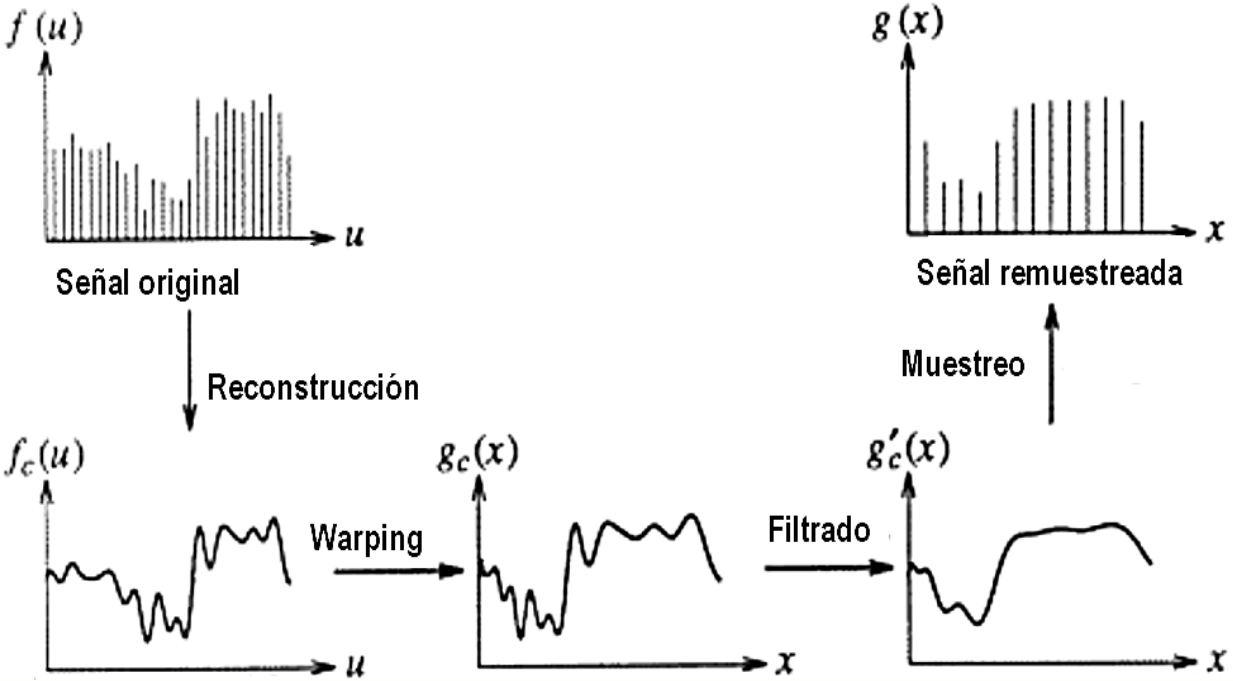
Procesamiento Digital de Imágenes

(Re)muestreo, reconstrucción, cuantización, compresión

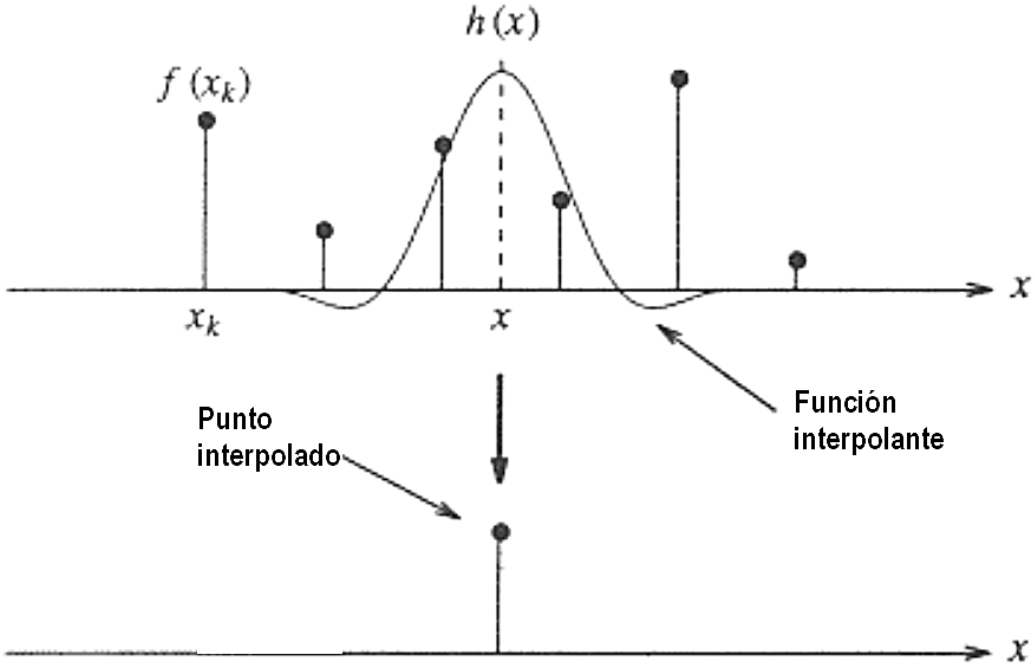
PDI – Remuestreo y reconstrucción

Si asumimos que el (hipotético) continuo puede modelarse como una señal (o imagen) de frecuencia de muestreo que tiende a infinito, entonces las operaciones de muestreo y reconstrucción solamente implican una reconversión de la frecuencia de muestreo.



PDI – Remuestreo y reconstrucción

En muchos casos, estas operaciones pueden ser subsumidas en una única operación de ajuste, donde se ajusta un conjunto finito de muestras de la señal original a una función interpolante dada, y se evalúa dicha función en el punto correspondiente a la remuestra.



PDI – Remuestreo y reconstrucción

El caso más sencillo consiste en utilizar una función interpolante constante a trozos, donde el valor de la función se mantiene constante entre muestras:

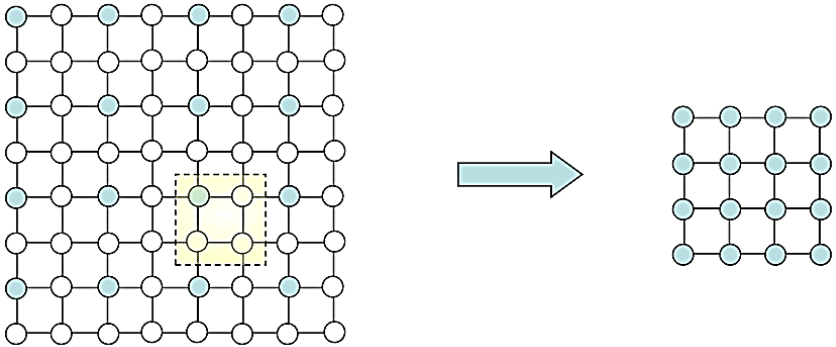
$$f(x) = f(x_k), \qquad \frac{x_{k-1} + x_k}{2} < x \leq \frac{x_k + x_{k+1}}{2}$$

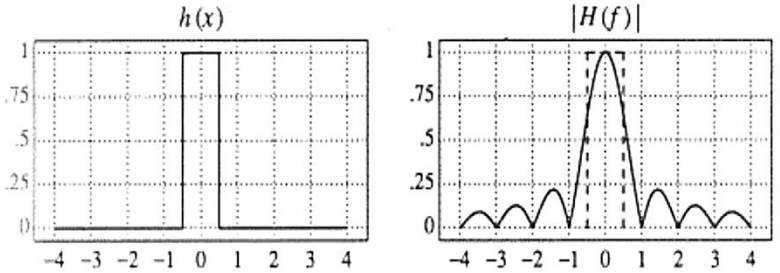
$$h(x) = \begin{cases} 1 & 0 \leq |x| < 0.5 \\ 0 & 0.5 \leq |x| \end{cases}$$

Este remuestreo es conocido como “sample and hold” (por sus reminiscencias electrónicas) y también como “vecino más cercano”. Equivale a considerar cada muestra como un impulso localizado en el centro de la muestra o del pixel.

PDI – Remuestreo y reconstrucción

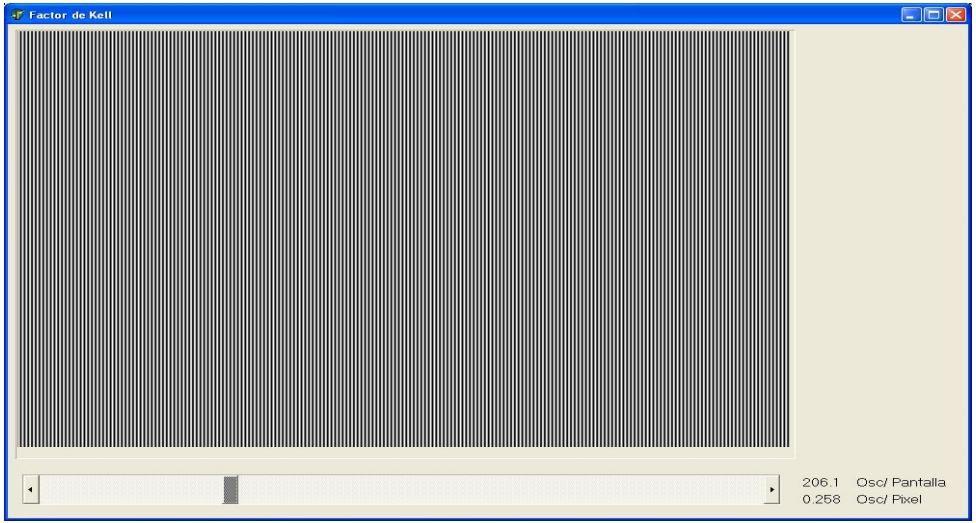
En imágenes, es el caso más frecuente: así funciona el muestreo de los sensores que ya vimos. El ejemplo más sencillo en imágenes digitales es el downsample a la mitad de la resolución. La función de ajuste constante a trozos tiene la respuesta en frecuencia ya conocida:





PDI – Remuestreo y reconstrucción

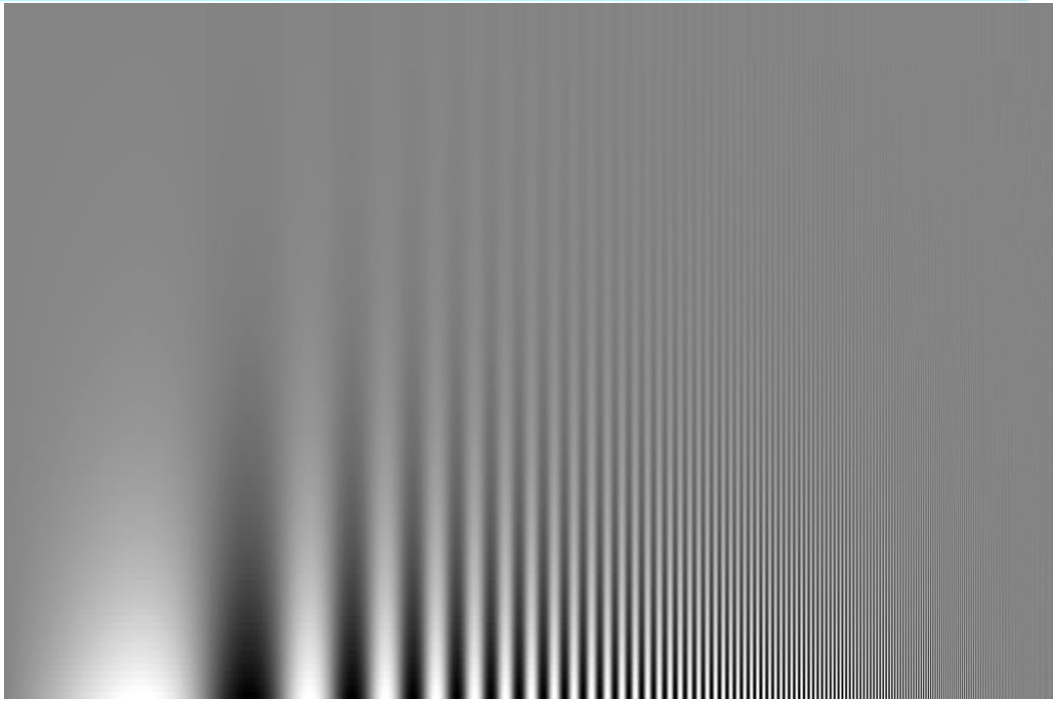
También, “implícitamente” el S&H es utilizado por los dispositivos de reproducción de imágenes (pantallas, impresoras, proyectores). Puede una mala reconstrucción generar aliasing?



PDI – Remuestreo y reconstrucción

Antes de hacer algunas experiencias tenemos que tener en cuenta otro factor de la percepción visual humana que es la respuesta en frecuencia (geométrica).

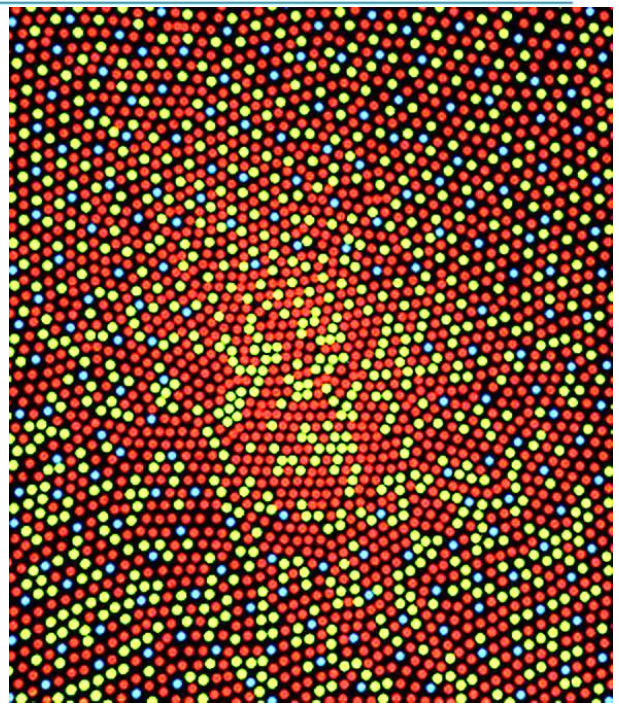
Esta imagen intenta «mostrar» empíricamente que hay una región donde nuestra capacidad es mayor, o sea, que nuestro ojo es un pasabanda.



PDI – Remuestreo y reconstrucción

Por qué ocurre eso (y sobre todo, por qué el ojo humano no tiene aliasing)? La ubicación de los conos en la retina no es una grilla, sino una red cuasi-hexagonal perturbada que se puede modelar con una distribución de Poisson.

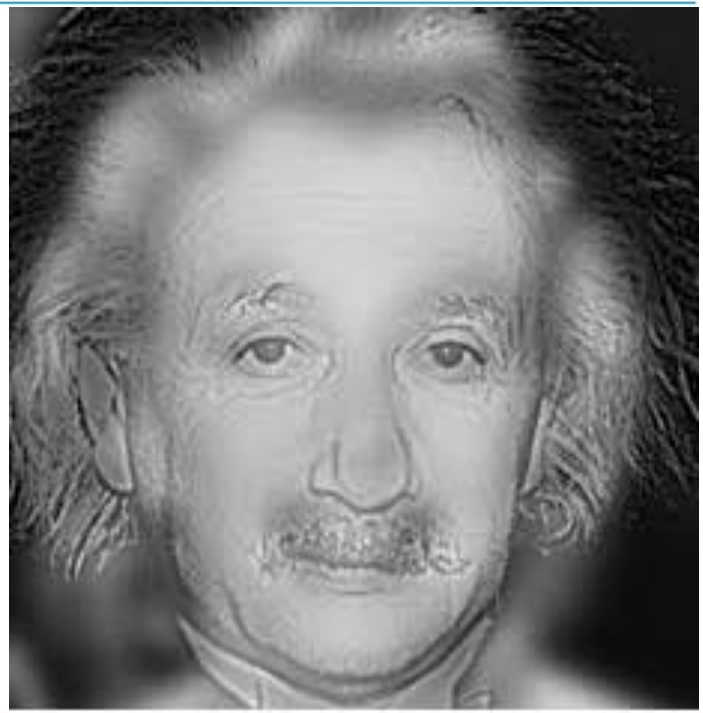
Eso “implementa” un muestreo estocástico (en geometría), el cual no tiene aliasing porque convierte la alta frecuencia en ruido. La TF de la distribución de Poisson es un pasaaltos, y todos los efectos ópticos (desenfoque, dispersión, deformación anatómica) son pasabajos.



PDI – Remuestreo y reconstrucción

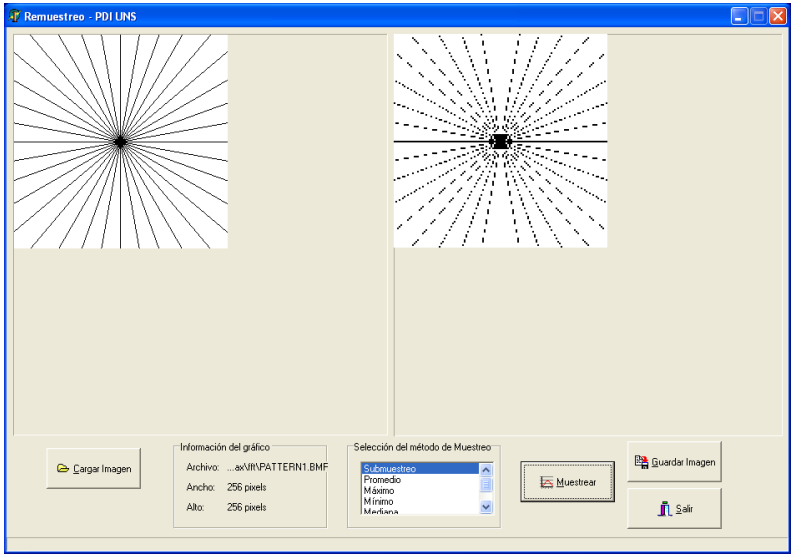
Esos efectos (Poisson más «Gaussianización») determinan que haya una banda de paso (aprox. 44 min de arco) donde nuestra percepción geométrica es máxima.

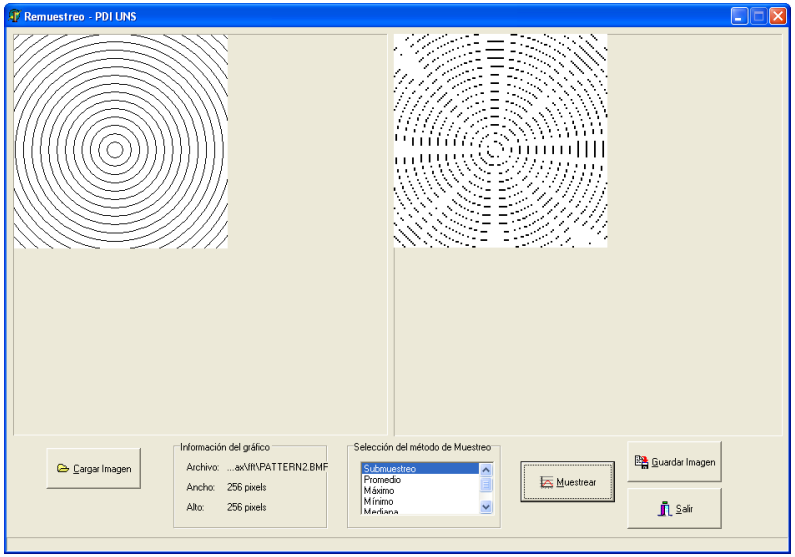
Por lo tanto a la hora de evaluar el efecto del aliasing y demás resultados del remuestreo, es necesario representar las imágenes con un mismo tamaño (aunque tengan diferente resolución) para garantizar que nuestra percepción sea adecuada.



PDI – Remuestreo y reconstrucción

Los resultados del S&H pueden ser insatisfactorios dado que “implícitamente” el filtrado que estamos aplicando genera aliasing (ilustrado aquí con dos patrones):

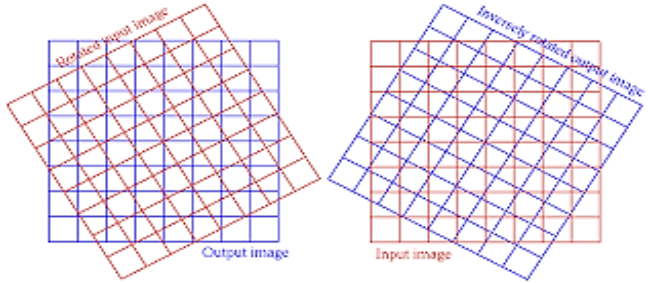




PDI – Remuestreo y reconstrucción

Un caso particular es el de la rotación de imágenes. Si bien la ecuación para rotar los píxels es sencilla:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta,$$

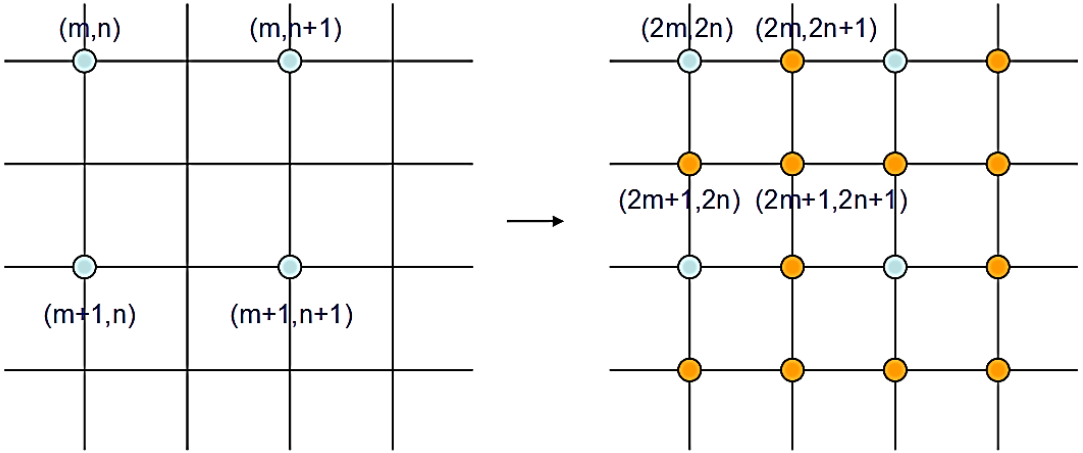


pero luego es necesario determinar en cada pixel de la imagen destino cuál es el pixel original correspondiente. Para ello es más sencillo utilizar la transformación inversa (que es la rotación por el ángulo contrario), y determinar en dicha inversa cuál es el pixel más cercano.

PDI – Remuestreo y reconstrucción

Otro caso particular es el “supermuestreo” (upsampling), en cuyo caso simplemente se repiten los valores de los pixels de la imagen original. En el sencillo caso del upsampling X2 obtenemos:

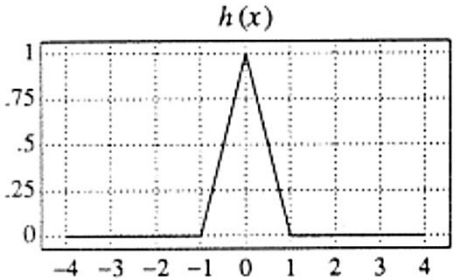
T(x, y) := S(x div 2, y div 2);

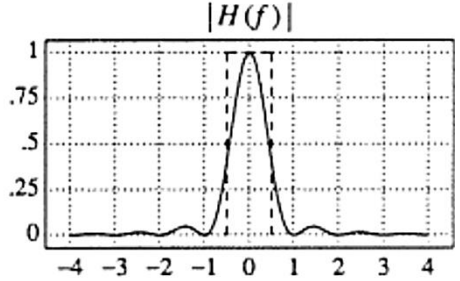


PDI – Remuestreo y reconstrucción

Luego podemos ir hacia funciones de interpolación de mejor calidad (funcionan como “kernels” con una respuesta en frecuencia más adecuada). El segundo caso más sencillo es la interpolación lineal a trozos.

$$f(x) = f_0 + \left(\frac{x - x_0}{x_1 - x_0} \right) \cdot (f_1 - f_0)$$



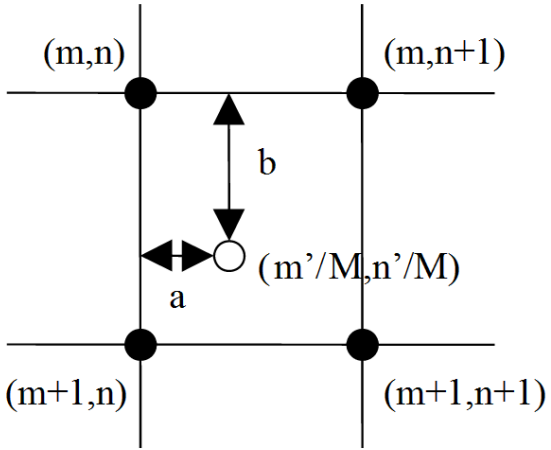


Esta función interpolante se conoce como filtro triangular, función carpa, ventana de Bartlett, etc. Equivale a considerar cada muestra como un vértice de una poligonal.

PDI – Remuestreo y reconstrucción

En el caso de imágenes, se aplica la descomposición por separabilidad. Para el upsampling vemos que a y b en la figura determinan la “altura” a la que estaría el círculo blanco, en función de las alturas de sus cuatro vértices que lo rodean.

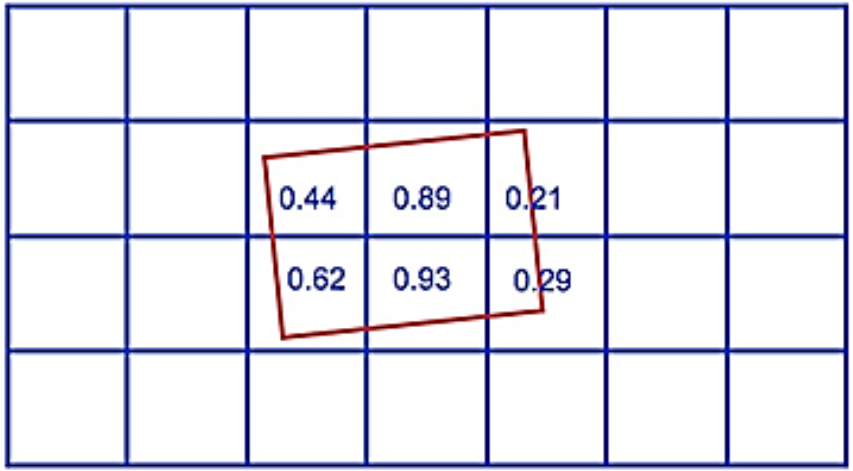
En el caso especial de upsampling X2, el pixel interpolado por fila es la media de los dos vecinos al igual que el pixel interpolado por columna, y el pixel “central” es la media de los cuatro vecinos.



PDI – Remuestreo y reconstrucción

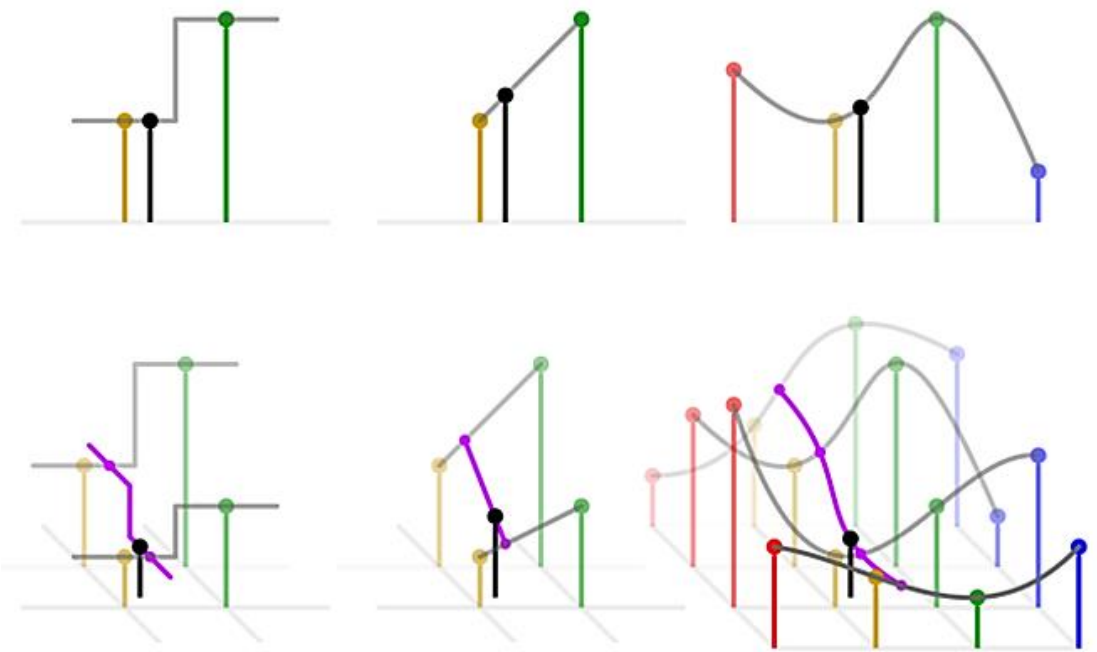
El downsampling y la rotación requieren un cuidado especial. En este caso conviene considerar como función ponderadora el área de cada pixel de la imagen de entrada que es visitada por cada pixel de la imagen de salida:

En el caso del downsampling X2, cada pixel de salida contiene cuatro de entrada, cada uno aportando toda su área, por lo cual la salida es el promedio de las cuatro.



PDI – Remuestreo y reconstrucción

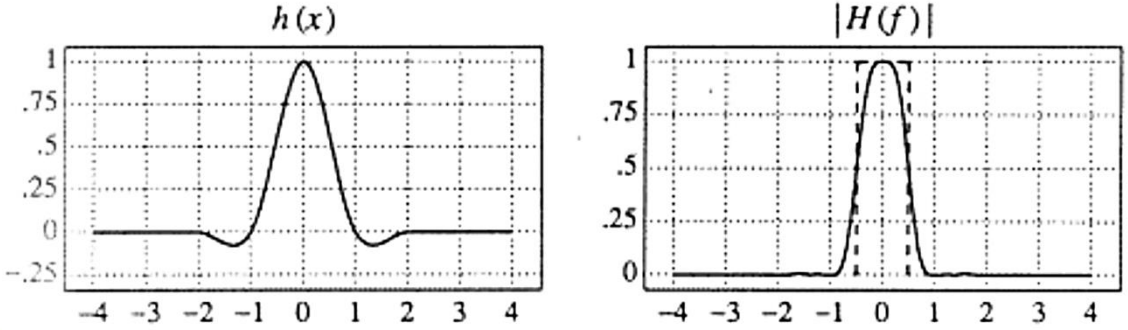
Existen funciones interpolantes de mayor orden, con la propiedad de separabilidad. Ello implica tener en cuenta un soporte que involucra una vecindad mayor. Por ejemplo para funciones (bi)cúbicas se requieren (4x)4 muestras de soporte.



PDI – Remuestreo y reconstrucción

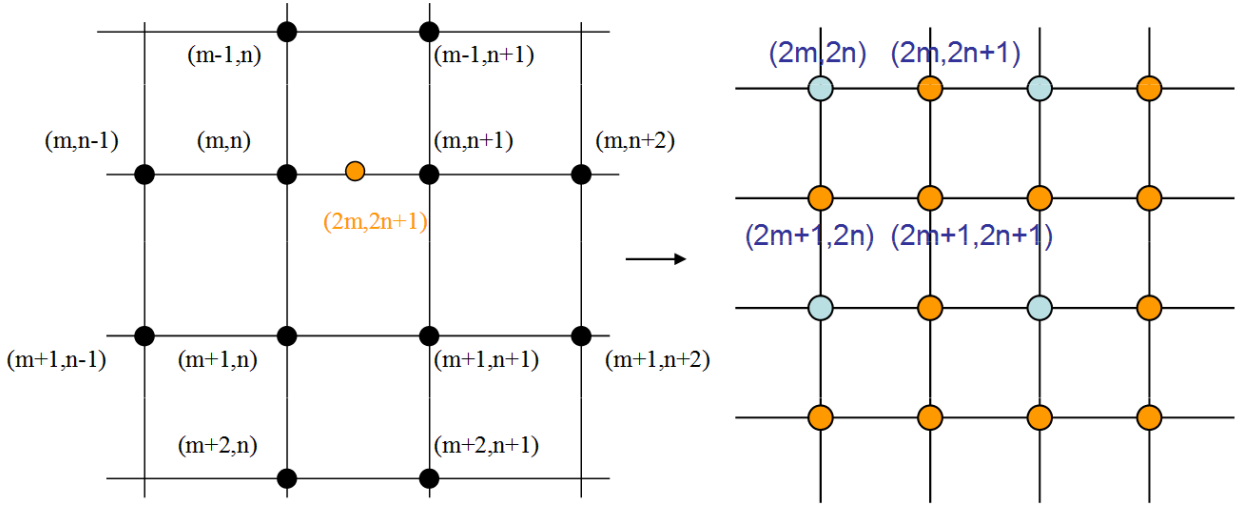
Una función muy utilizada es la convolución cúbica, donde a es un parámetro libre que permite aproximarla a una sinc truncada (-3 >a >0):

$$h(x) = \begin{cases} (a+2)\cdot|x|^3 - (a+3)\cdot|x|^2 + 1 & 0 \le |x| < 1 \\ a\cdot|x|^3 + 5a\cdot|x|^2 + 8a\cdot|x| - 4a & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$



PDI – Remuestreo y reconstrucción

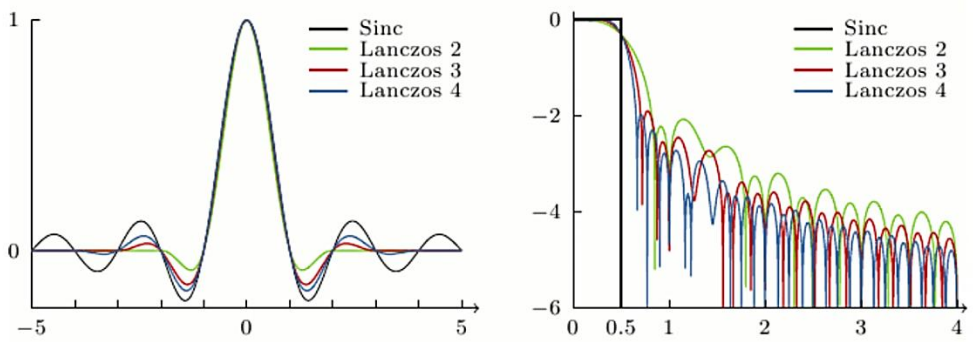
El caso particular del downsample y upsample X2 se resuelve para a = -1. El punto naranja en la figura de la izquierda es (-1/8, 5/8, 5/8, -1/8) por su fila de puntos negros.



PDI – Remuestreo y reconstrucción

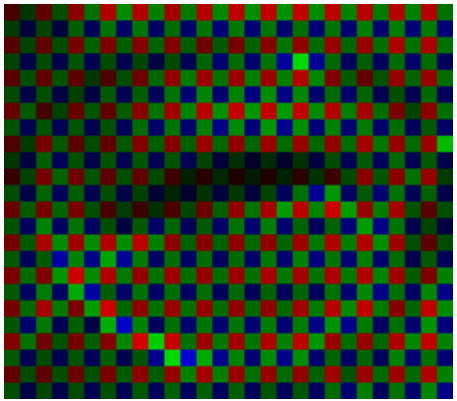
Si bien existen otras funciones polinomiales prácticas, en general se considera que la función de Lanczos es la mejor aproximación finita a la reconstrucción ideal:

$$L(x) = \begin{cases} \operatorname{sinc}(x)\ \operatorname{sinc}(x/a) & \text{si } -a < x < a \\ 0 & \text{caso contrario} \end{cases}$$

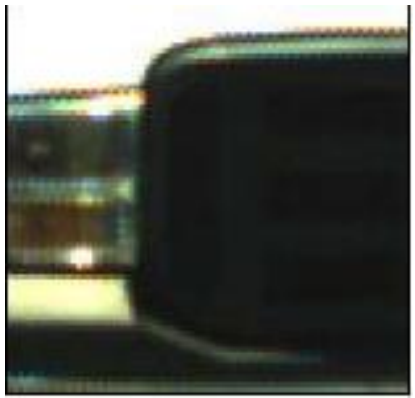


PDI – deBayering

Una aplicación usual de estos filtros de reconstrucción es en el deBayering. El nombre proviene de la grilla de Bayer que configura usualmente los sensores RGB, y que genera las imágenes por medio de interpolación bilineal, con los consiguientes artificios.



Grilla de Bayer



Artificios típicos

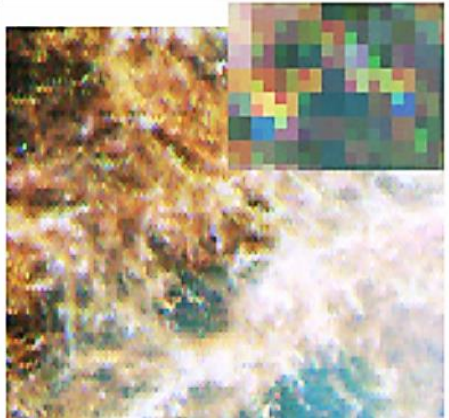
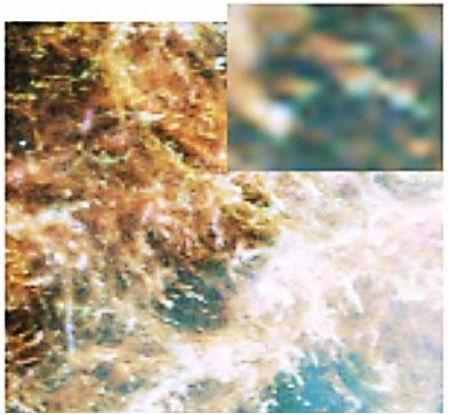


Imagen sin deBayer



Con deBayer

PDI – Cuantización

Consiste en elegir un posible conjunto V de *n* valores target de luminancia para representar la imagen. Todo valor original de luminancia es coercionado a alguno de ellos de acuerdo a diferentes criterios.

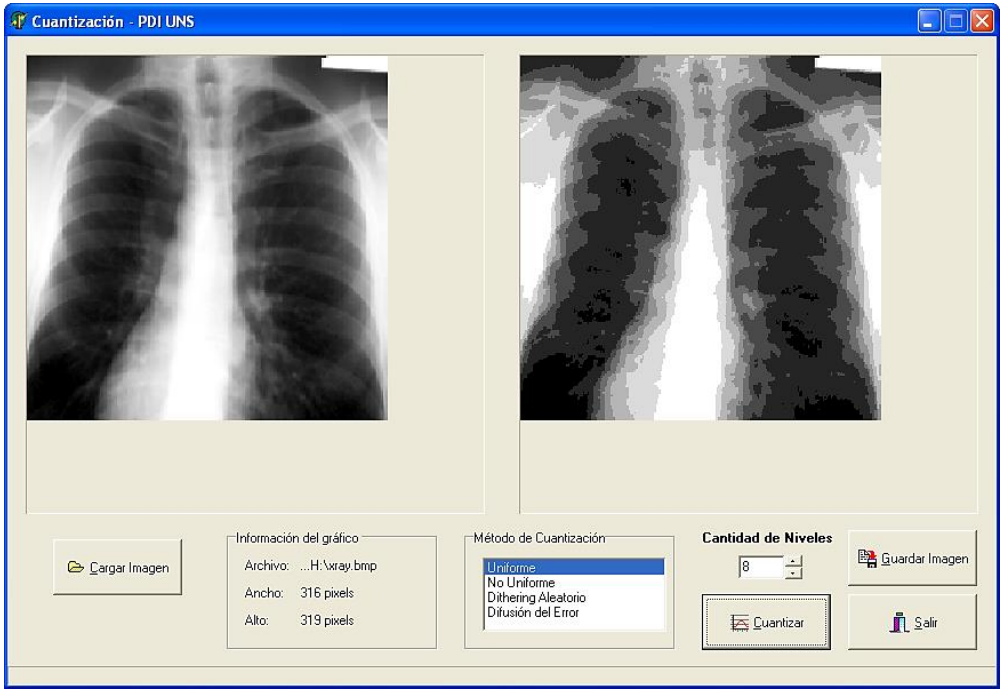
Los valores target suelen ser una partición uniforme del rango de luminancias (por ejemplo V = {0, 1/(*n*-1), 2/(*n*-1), …, (*n*-2)/(*n*-1), 1}). En momentos prehistóricos, V incorporaba también la corrección Γ.

El método más simple consiste en elegir el valor de V más cercano, lo cual se denomina cuantización uniforme (o no-uniforme si V sigue una secuencia diferente).

PDI – Cuantización

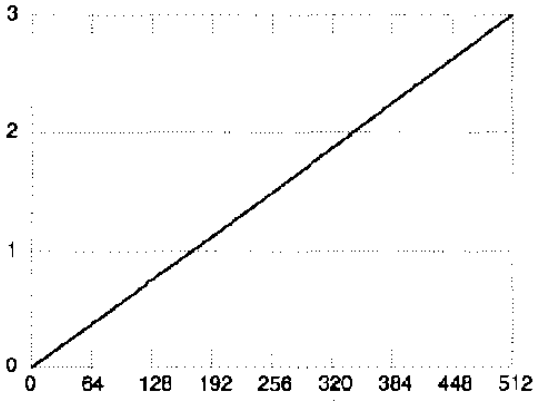
La cuantización se puede modelar como un proceso lineal, en particular como la suma de un error de cuantización.

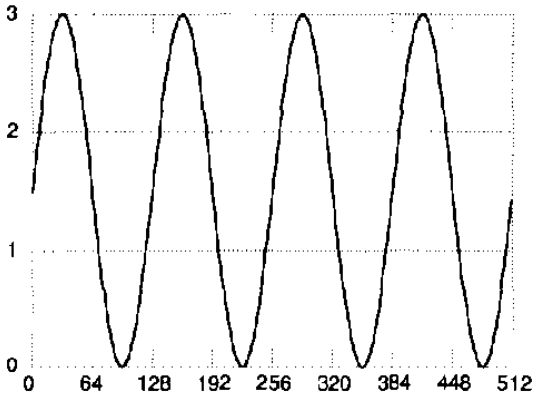
El deterioro en la calidad de la imagen debido a la cuantización es difícil de evaluar en términos del RMS del error.

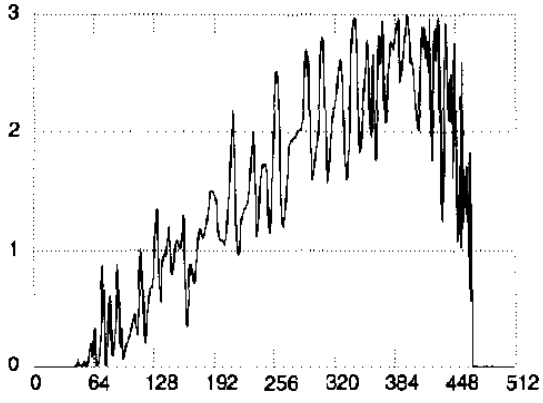


PDI – Cuantización

Una forma más adecuada de caracterizar el efecto pernicioso de la cuantización es analizar el espectro del error de cuantización, en términos de la percepción humana a las frecuencias espaciales. Consideramos tres “señales” típicas (scan-lines):



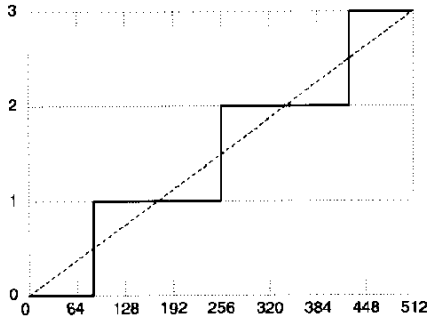


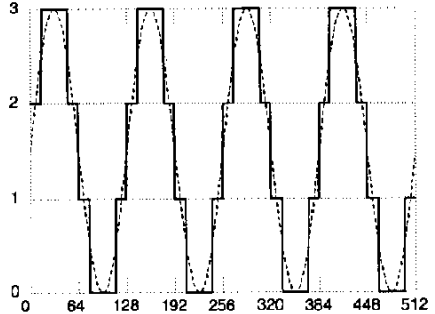


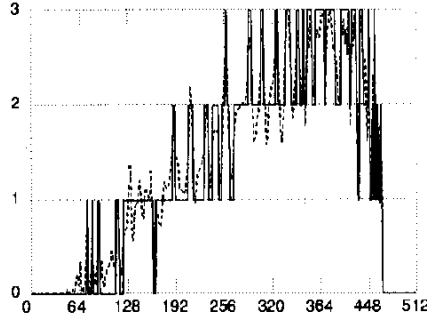
PDI – Cuantización

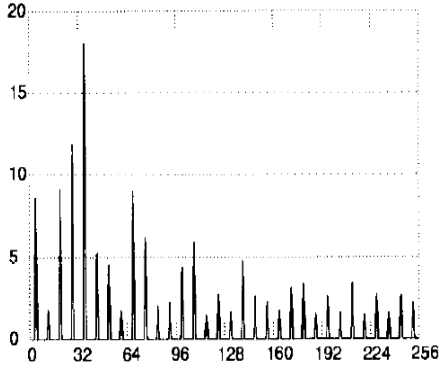
En el caso de tener n=4, se ilustra abajo el efecto de la cuantización y el espectro del error.

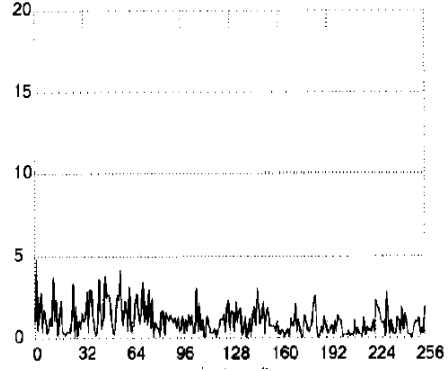
Recordar que la frecuencia espacial preferida por el ojo está alrededor de ~40 osc/scanline.







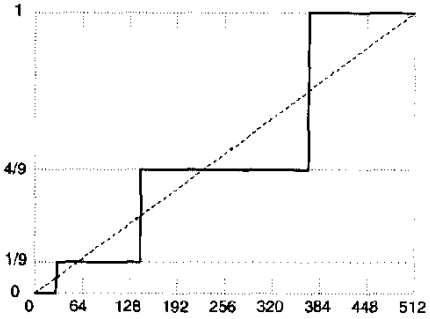


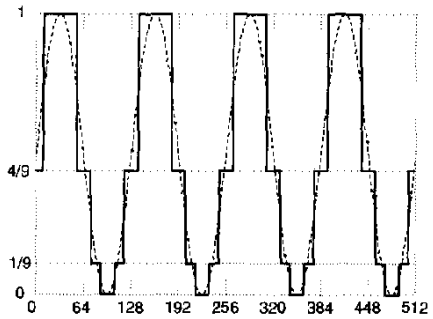


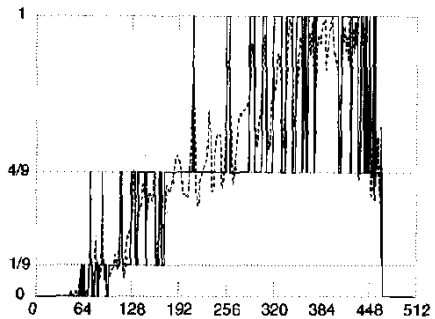
PDI – Cuantización

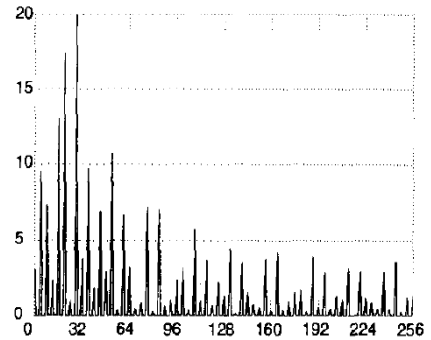
El uso de cuantización no uniforme no mejora el mal comportamiento espectral.

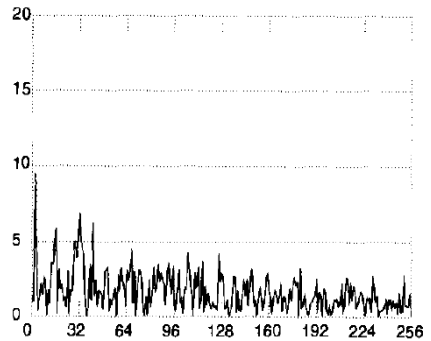
Recordemos que en formato sRGB la luminancia ya incluye la corrección Γ.







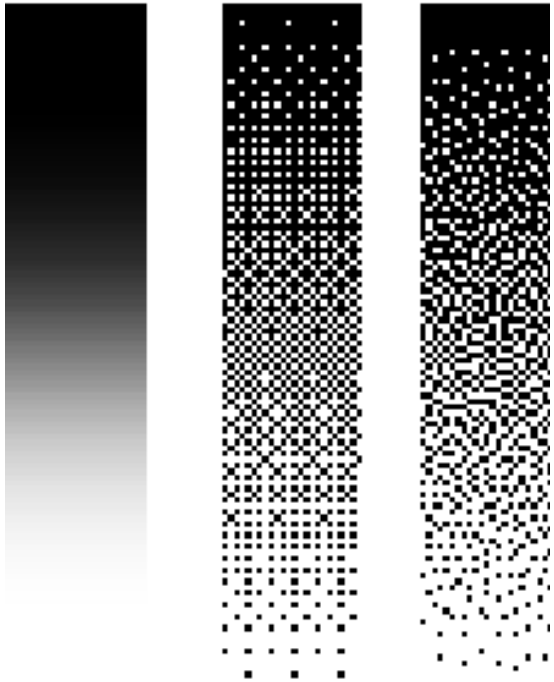




PDI – Cuantización

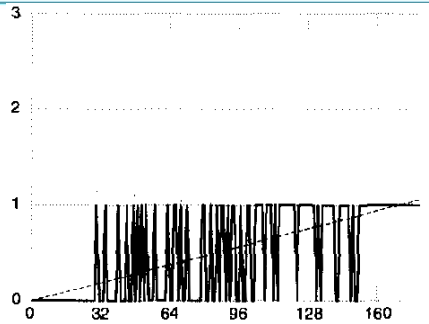
Una forma de reducir el error de cuantización es por medio del uso de tramas (dithering), las cuales pueden ser ordenadas o aleatorias.

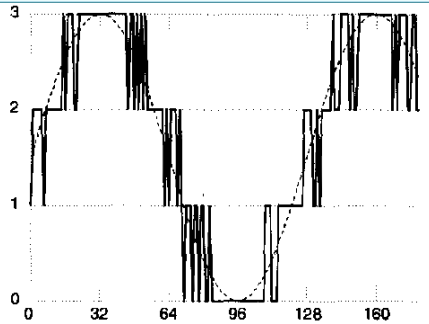
Aquí mostramos dithering entre dos niveles, pero puede generalizarse a más niveles.

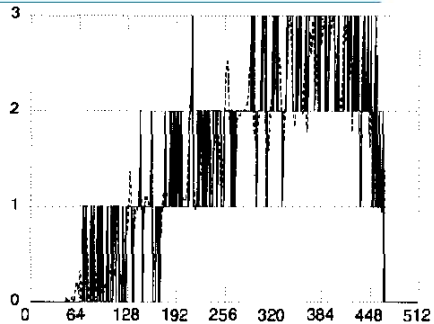


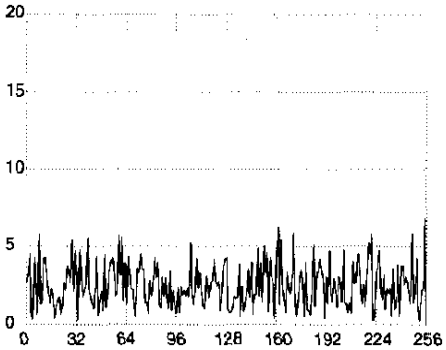
PDI – Cuantización

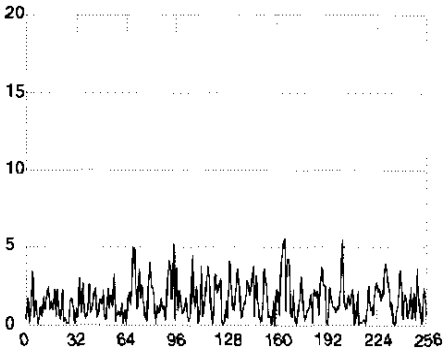
Comparado con la cuantización uniforme, el eror de cuantización por dithering es más isotrópico en frecuencias.







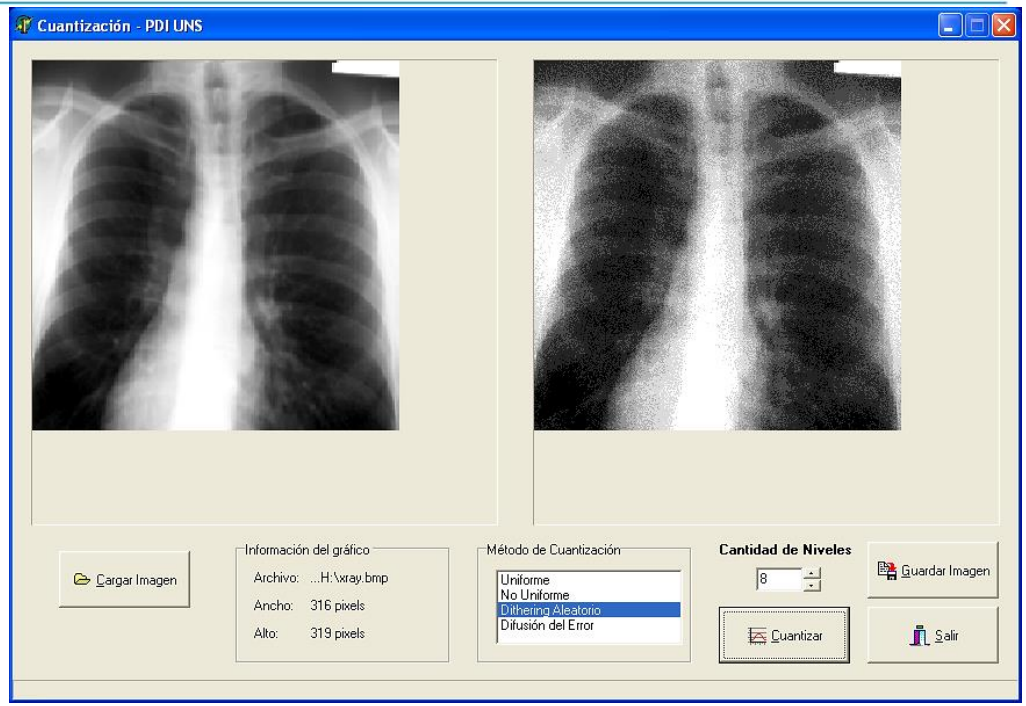




PDI – Cuantización

Aún con pocos niveles de gris, el dithering es capaz de reproducir en forma inteligible figuras en la imagen que tengan bajo contraste y bordes suaves.

El exceso de alta frecuencia es mitigable en muchos casos utilizando un pasabajos.



PDI – Cuantización

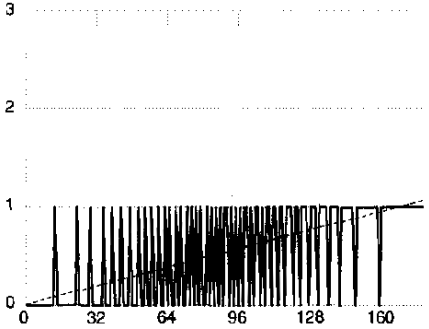
Una forma de reducir el error de cuantización es “difundir” el error entre los vecinos, de manera tal que se compensen localmente los errores por defecto con los errores vecinos por defecto (y viceversa).

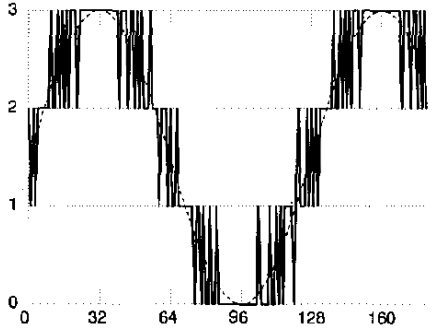
Esa es la idea subyacente al método de **difusión del error**. Una forma muy sencilla de implementar DE por scan-line es trabajar con un umbral flotante:

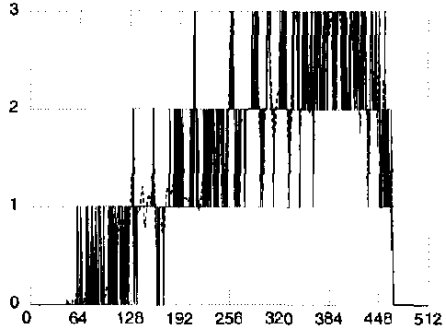
```
error := 0;                          {asumimos esto para toda fila j}
for i:= 0 to scanline.widht-1 do begin
   salida[i,j] := round( entrada[i,j] + error );
   error:= error + salida[i,j] - entrada [i,j];
end;
```

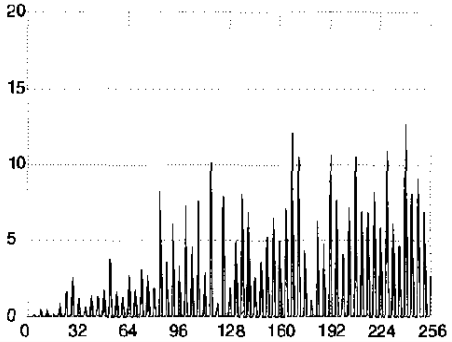
PDI – Cuantización

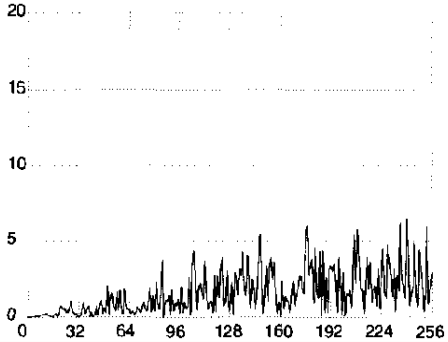
La difusión del error (aún la muy simple que mostramos) tiene la virtud de generar una señal error con menos baja frecuencia (que es la que distorsiona las figuras).







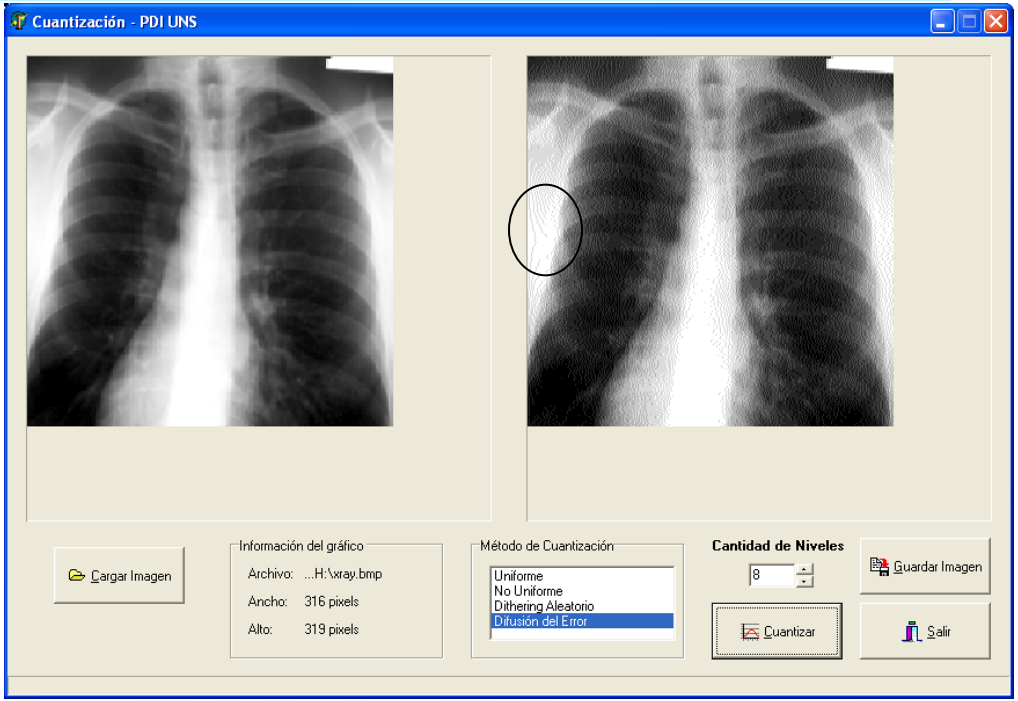




PDI – Cuantización

La difusión del error genera imágenes donde la forma es mucho más inteligible, y las diferencias de contraste quedan transformadas en diferencias de texturas que son fácilmente procesables con pasabajos.

Ver el efecto de “curvas de nivel” en la parte destacada. A veces puede servir para reconstruir los perfiles de luminancia con mucha precisión.



PDI – Cuantización

La difusión del error puede hacerse en 2D con mayor eficacia. Uno de los algoritmos más utilizados es el de Floyd-Steinberg. Se basa también en progresar por scan-line y distribuir el error en una vecindad, pero utiliza un kernel de 2X3 que considera el scan-line inferior (“.” es el pixel actualmente procesado, y X es el pixel procesado en el paso anterior):

$$\frac{1}{16}\begin{bmatrix} X & . & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

Hay variantes de esta idea con kernels más grandes, y también algoritmos que combinan la cuantización con los aspectos perceptuales del ojo humano, como por ejemplo el método retinex.

PDI – Cuantización en espacios cromáticos

Típicamente es necesaria en formatos como el .gif u otros ‘’paletizados’’.

El algoritmo tradicional denominado **populosity** básicamente hace un histograma en el espacio cromático 3D, eligiendo las 256 modas más populares. Esto claramente tiene desventajas, dado que hay ‘’minorías’’ no representadas que pueden tener un impacto muy importante en la imagen.

Un algoritmo más complejo pero de mejor resultado es el **median cut**, que básicamente particiona el espacio cromático en 256 regiones de tamaño variable, de manera que cada región tenga una cantidad similar de pixels. De esa manera, las minorías pierden precisión pero no representatividad.

PDI – Actividad práctica

Desarrollar un aplicativo que implemente las siguientes funciones:

- Downsampling X2 (mostrando la imagen resultado al doble de tamaño para comparar el efecto) utilizando kernel constante, bilineal, y bicúbico.
- Upsampling X2 (mostrando la imagen original al doble de tamaño para comparar el efecto) utilizando kernel constante, bilineal, y bicúbico.
- Cuantización a cantidad de niveles de gris variable, utilizando uniforme, dithering aleatorio, y difusión del error por scan-line.

PDI – Compresión

Es posible categorizar a los métodos de compresión de imágenes de acuerdo a varios criterios:

- Con pérdida vs. sin pérdida
- 1D vs. 2D
- Entropía vs. psico-visual
- Geométrico vs. espectral

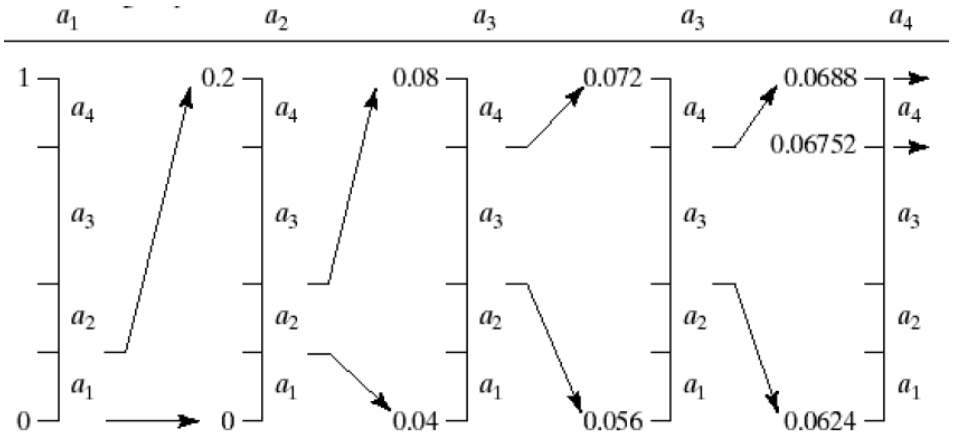
PDI – Compresión

Algunos de los métodos más usuales:

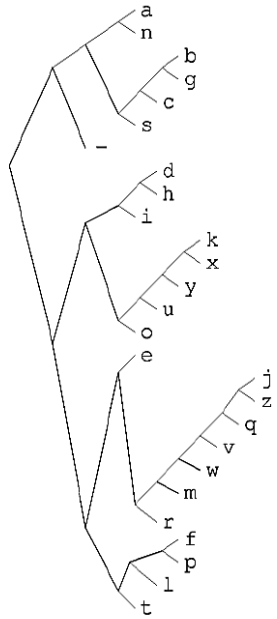
- Huffman - Aritmético (1D, sin pérdida, entropía, numérico).
- RLE (1D o 2D, sin pérdida, entropía, geométrico).
- Predictivos (VQ) (1D o 2D, con pérdida, entropía, numérico).
- Fourier (2D, con pérdida, psico-visual, espectral).
- JPeg (DCT) (2D, con pérdida, psico-visual, espectral).
- JPeg 2000 (Wavelet) (2D, con pérdida, psico-visual, espectral).
- CFB (2D, con pérdida, psico-visual, geométrico).
- Karhunen-Loève (2D, con pérdida, entropía, espectral).

PDI – Compresión

Los métodos de Huffman y aritmético permiten la compresión sin pérdida de un mensaje eliminando la redundancia:



a_i	p_i	$\log_2 \frac{1}{p_i}$	l_i	$c(a_i)$
a	0.0575	4.1	4	0000
b	0.0128	6.3	6	001000
c	0.0263	5.2	5	00101
d	0.0285	5.1	5	10000
e	0.0913	3.5	4	1100
f	0.0173	5.9	6	111000
g	0.0133	6.2	6	001001
h	0.0313	5.0	5	10001
i	0.0599	4.1	4	1001
j	0.0006	10.7	10	1101000000
k	0.0084	6.9	7	1010000
l	0.0335	4.9	5	11101
m	0.0235	5.4	6	110101
n	0.0596	4.1	4	0001
o	0.0689	3.9	4	1011
p	0.0192	5.7	6	111001
q	0.0008	10.3	9	110100001
r	0.0508	4.3	5	11011
s	0.0567	4.1	4	0011
t	0.0706	3.8	4	1111
u	0.0334	4.9	5	10101
v	0.0069	7.2	8	11010001
w	0.0119	6.4	7	1101001
x	0.0073	7.1	7	1010001
y	0.0164	5.9	6	101001
z	0.0007	10.4	10	1101000001
–	0.1928	2.4	2	01



PDI – Compresión

Ya vimos que la DFT tiene la propiedad de “empaquetar” mejor la información, en el sentido de tener la energía representada en menor cantidad de coeficientes relevantes.

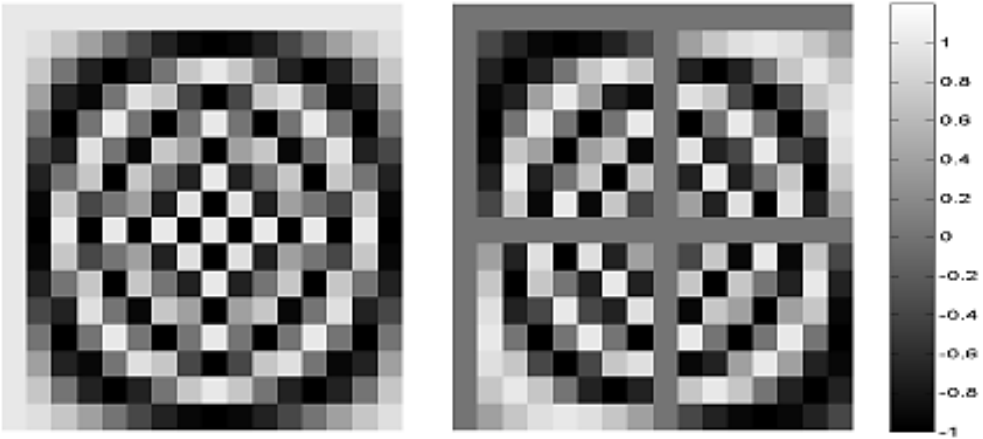
Una mirada más atenta a la DFT nos permite entender algunas de sus propiedades:

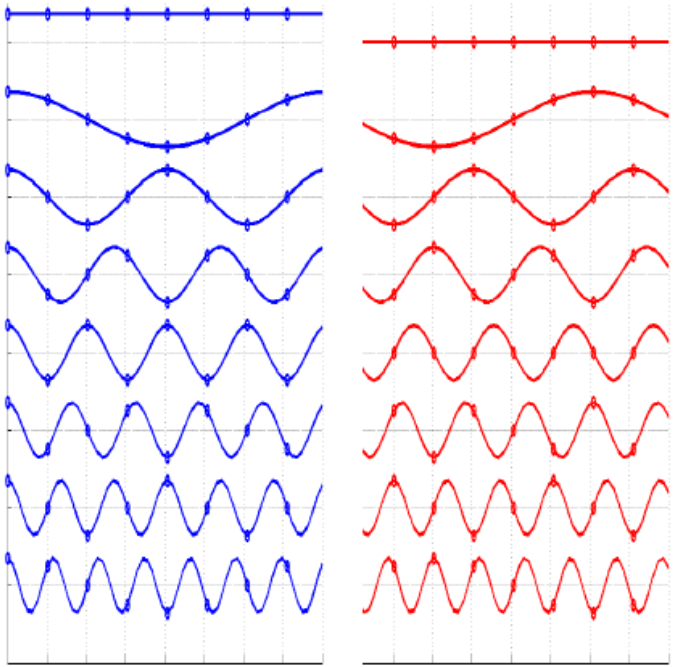
$$E(F) = \frac{1}{N} \sum_{n=0}^{N-1} f(n) e^{\frac{-2\pi i F n}{N}}$$

Cada F representa un producto escalar con los coeficientes $f(n)$ y una de las N bases. Se puede pensar entonces como un producto de vector por matriz.

PDI – Compresión

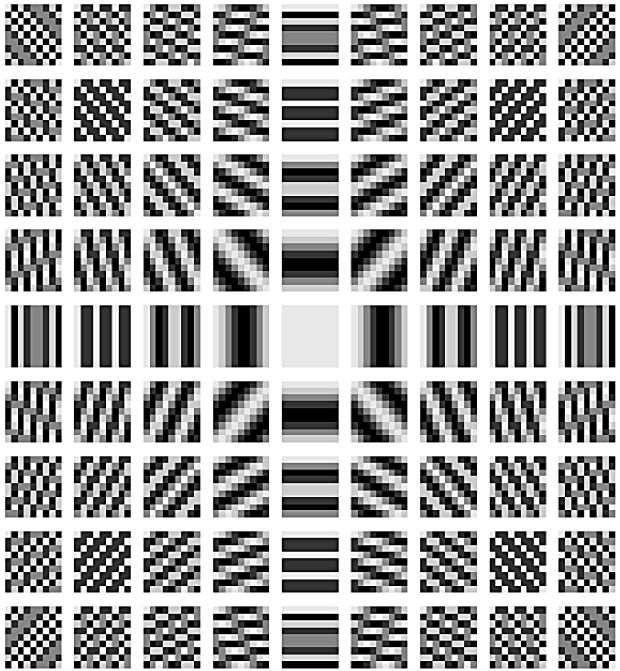
Dos posibles visualizaciones de la parte real e imaginaria de las 16 bases para $N = 16$.

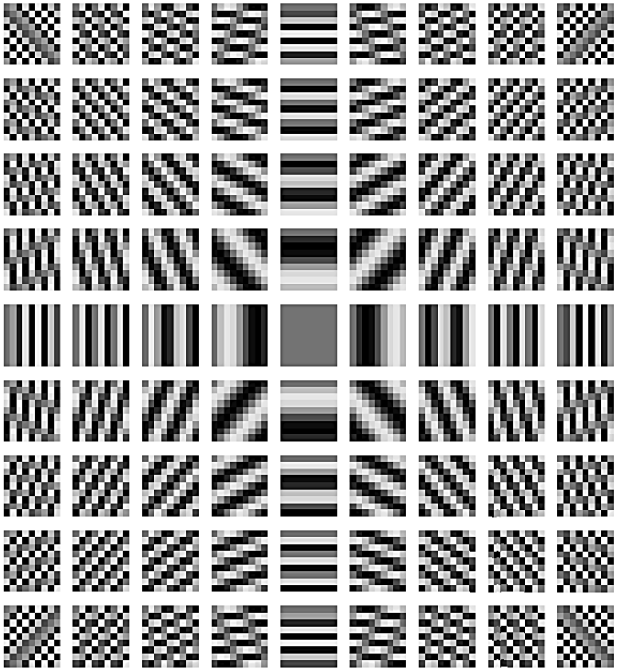




PDI – Compresión

En 2D tenemos el mismo concepto. Para cada frecuencia horizontal o vertical tenemos una matriz real y una imaginara de coeficientes que se multiplican con la imagen original (en este caso N=9).



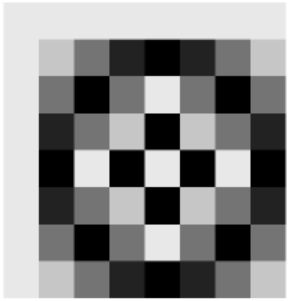


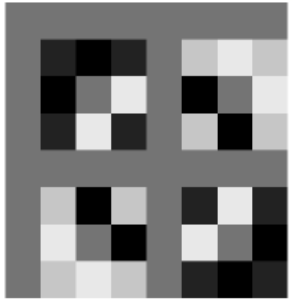
PDI – Compresión

En el caso de *N*=8 es más fácil visualizar que la parte real tiene solo cinco filas diferentes, y la parte imaginaria tiene tres. Es posible ver también que estas matrices son ortonormales e iguales a su transpuesta.

La matriz que se obtendría si utilizamos solo las ocho filas diferentes es ortonormal y su inversa es su transpuesta. Esta inversa es, además, la matriz que se obtendría si realizamos el mismo procedimiento con la DFT inversa.

La DFT es implícitamente una rotación!





$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & r & 0 & -r & -1 & -r & 0 & r \\ 0 & r & 1 & r & 0 & -r & -1 & -r \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & -r & 0 & r & -1 & r & 0 & -r \\ 0 & r & -1 & r & 0 & -r & 1 & -r \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

PDI – Compresión

En la práctica, en vez de utilizar esta última matriz, se utiliza la transformada coseno discreta (DCT):

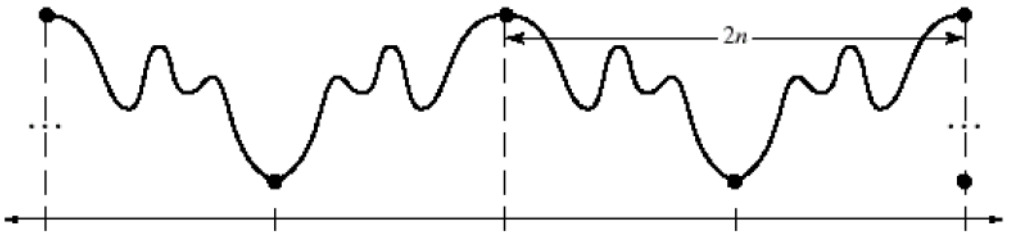
$$E(F) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} f(n) \cos\left[\frac{\pi(2n-1)F}{2N}\right] \qquad F = 1, \ldots, N-1$$

$$E(0) = \frac{1}{\sqrt{N}}$$

Es importante entender que la DCT **NO es** la parte real de la DFT. La matriz con los NxN coeficientes es ortonormal, y su inversa es su transpuesta: es una rotación.

PDI – Compresión

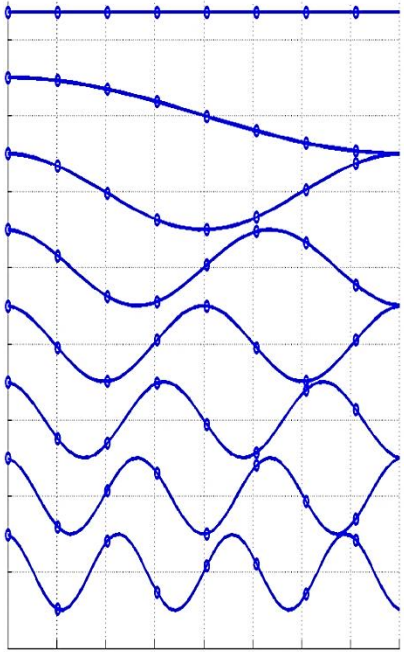
Una de las diferencias más importantes está en el término dentro del coseno: se toman los valores del coseno “a mitad de camino”, y se barre la mitad. Eso implica entre otras cosas que la periodicidad se hace por reflexión y no por repetición.

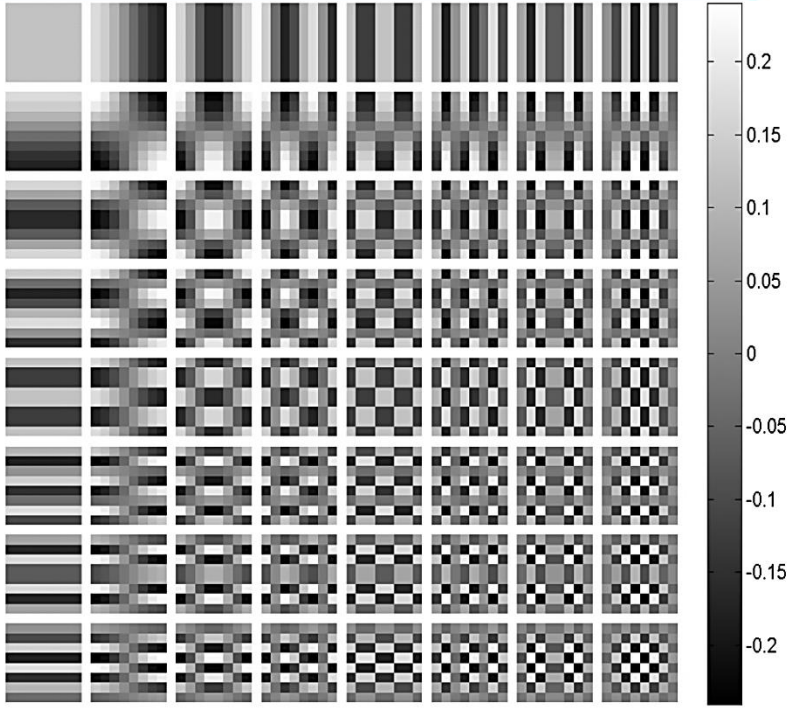


PDI – Compresión

Las bases DCT en 1D y 2D para N=8.

Observar que la 2D es el producto (separable) de dos bases 1D.





PDI – Compresión

La DCT es la base de la compresión JPEG (1987), MPEG (1988) y subsiguientes. Se basa en dividir la imagen en bloques de 8x8 pixels, los cuales son procesados independientemente. Los pasos completos del standard JPEG son:

- Conversión de RGB a YCrCb (similar a YIQ).
- Downsampling de los canales Cr y Cb (4:4:4, 4:4:0, 4:2:2, 4:2:0, 4:1:1).
- División de cada plano Y-Cr-Cb a bloques de 8x8.
- DCT independiente a cada bloque (a veces se sustrae 127 a cada valor).
- Cuantización de los coeficientes DCT resultantes (a 4, 5 o 6 bit).
- Recorrida en zig-zag de los coeficientes, reteniendo los más importantes. Run lenght.
- Compresión Huffman de los bloques resultantes.

PDI – Compresión

Otros métodos:

- JPEG-2000 (Wavelets)
- Karhunen-Loève (a.k.a., Hadamard, a.k.a. PCA)
- Compresión fractal en bloques
- Cuantización de vectores (compresión adaptativa)