UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DFT vs DCT: un ejemplo visual de uso mediante OpenCV

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores**
Centro	**Escola Tècnica Superior d'Enginyeria Informàtica** Universitat Politècnica de València

1 Resumen

Muchas veces leemos sobre el caso 1D de correspondencia de representación de señales en el dominio del tiempo y de la frecuencia. Es un tópico ineludible del procesado digital de señal ([1] y [2]) que existe una correspondencia entre ambas representaciones y que es posible calcularla con herramientas matemáticas en el ámbito discreto como la Transformada Discreta de *Fourier* o la Transformada Discreta del Coseno (que en adelante notaremos por la siglas de los términos en inglés, DFT y DCT, respectivamente).

Estas transformaciones también se utilizan al trabajar con imágenes, pero no es trivial explicarlo o dibujarlo. En este trabajo queremos ver y explorar la correspondencia entre el dominio espacial y el frecuencial en el caso bidimensional, sin entrar en los detalles de derivación de la formulación, para lo que la bibliografía clásica suele cubrir ampliamente el paso en ambos sentidos:

- Directo (DFT o DCT), que obtienen la representación frecuencial a partir de la espacial.
- Inverso (representados con las siglas IDFT o IDCT), que permite calcular la representación espacial a partir de la frecuencial.

¿Cómo lo haremos? Veremos un ejemplo de cálculo de estas transformaciones, las componentes que se obtienen y que el proceso es reversible. Veremos imágenes con diferentes contenidos y como su distribución espacial (repetición u orientación) se ven reflejados en las transformadas. Para terminar observando visualmente cómo se recupera una imagen, un tanto diferente de la original, si se manipulan esos elementos que constituyen la representación en frecuencia.

De una forma muy simplista, la DCT obtiene la respuesta clásica de qué frecuencias están presentes en el contenido de una imagen de una forma explícita y directa: en una sola componente de la misma dimensión que la imagen y de valores reales. Por eso la podemos encontrar habitualmente utilizada en el campo de la compresión.

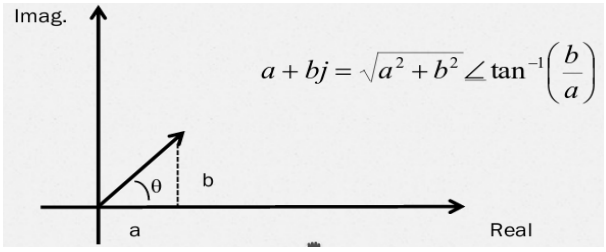


Figura 1: Representación vectorial de un número complejo. La parte real y la imaginaria (a+bj) son equivalentes a la magnitud y fase del vector que representa el número complejo. Imagen de <http://ccc.inaoep.mx/~pgomez/cursos/pds/slides/S5-DFT.pdf>.

Por su parte la DFT obtiene una representación en valores complejos, lo que se traduce en dos elementos: la parte real y la imaginaria, cada uno de la dimensión de la imagen de partida. La DFT se utiliza habitualmente en aplicaciones de detección de simetrías en imágenes, filtros de procesado de señal. Lo más habitual es ver la representación combinada de estos dos elementos en forma vectorial (véase la Figura 1), expresada en términos de magnitud y fase. La magnitud corresponde con el módulo del vector, esto es con la valoración de cuan presente está cada componente de frecuencia y, por ello, es comparable al resultado de la DCT. Pero, además, el ángulo sobre el eje horizontal nos cuenta acerca de la fase (relativa) de cada componente de

frecuencia presente en la señal. Esto es una valoración de dónde se da esa frecuencia en la imagen.

2 Objetivos

Una vez que el lector haya explorado los contenidos de este documento y experimentado con el código de ejemplo que se comenta, será capaz de:

- Asociar el uso de las transformaciones DFT y DCT con aplicaciones tanto de cuestiones de procesado, como de análisis de imagen.
- Experimentar con un ejemplo de código que permite visualizar la representación frecuencial que obtiene la DFT y la DCT, familiarizándose con su funcionamiento interno.
- Experimentar con un ejemplo de código para observar la relación entre la transformación directa y la inversa

Para hacer posible esta experimentación vamos a utilizar OpenCV [3] para mostrar de manera visual, sin entrar en los detalles de todo el proceso matemático interno, la DFT y la DCT. OpenCV es una librería de funciones que es un referente para la implementación de aplicaciones de Visión por Computador.

Aunque podemos leer este documento y ver si nos convencen las imágenes, la diversión está en probar los ejemplos y modificarlos, Será la mejor manera de aprender de ellos. Solo necesitamos un terminal y un editor de código. Este trabajo se centrará en la plataforma *Linux/Unix*, pero podremos utilizar sus contenidos en otros sistemas operativos: el haber optado por OpenCV para desarrollar el código del ejemplo es para que sea totalmente portable.

3 Introducción

Sin entrar en la teoría matemática [1], revisaremos en este punto cómo se ve la representación en frecuencias de una señal en el caso 1D y 2D.

3.1 Caso unidimensional

En el caso unidimensional, una señal se puede expresar [2] como una suma infinita de señales periódicas, como expresa la Ec. 1.

$$f = a_n \cos(nx) + b_n \sin(nx) \qquad \textit{Ec. 1.}$$

La definición de DFT para 1D en [2] viene acompañada de una muy interesante animación que muestra la relación entre el dominio del tiempo y de la frecuencia utilizando la transformada de *Fourier*. La Figura 2, está hecha con capturas de instantes de esa animación y se puede ver como la función *f* (en rojo) en el tiempo, se descompone (Figura 2 izquierda) en seis funciones (en azul) en frecuencias. La función *f* se puede representar de forma equivalente (Figura 2 derecha), tanto como una que varía en el tiempo o como una que varía en el dominio de la frecuencia y formada por los seis valores de frecuencia que están presentes en la misma.

La DFT relaciona*, Figura 3, una señal en el dominio del tiempo, f(n), con su equivalente en el dominio de la frecuencia, F(u): o*bservemos de las expresiones[1] que la DFT y la IDFT son la misma expresión con el signo cambiado y una constante. La DFT de una función con N puntos da como resultado N

1 Extraídas de <http://ccc.inaoep.mx/~pgomez/cursos/pds/slides/S5-DFT.pdf>.

valores complejos que pueden dividirse en dos componentes: una parte real y una imaginaria.

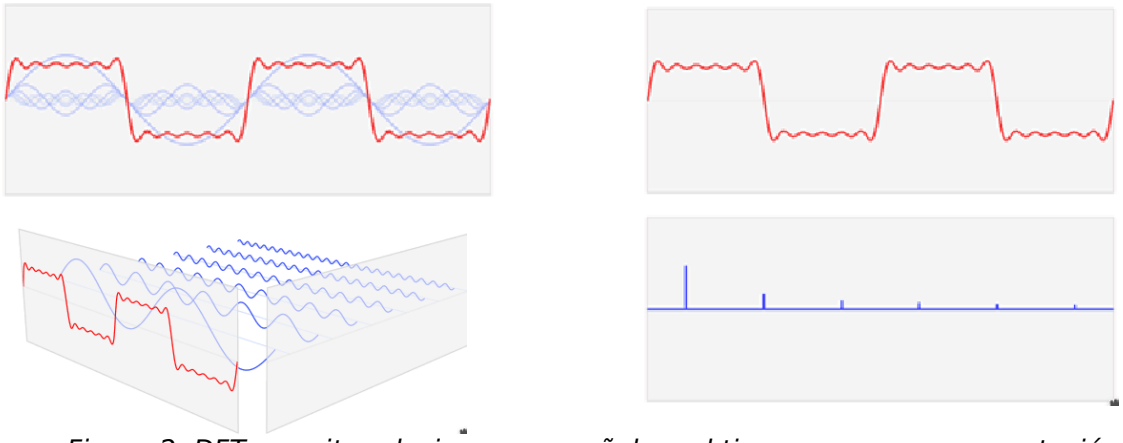


Figura 2: DFT permite relacionar una señal en el tiempo con su representación en frecuencias. Imágenes extraídas de [2].

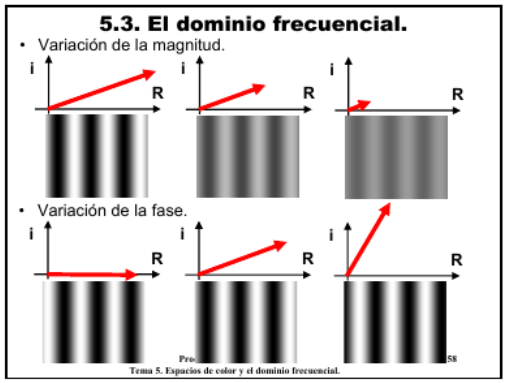
$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}un} F(u)$$

para n = 0,1 .. N-2

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

$u = 0,1,2, \ldots, N\text{-}1$

a)



b)

Figura 3: Formulación matemática de la DFT y de la IDFT (a) y visualización [4] de los elementos que definen la representación frecuencial (b).

3.2 Transformada bidimensional

El trabajo de García-Medrano [4] ofrece una perspectiva muy completa del dominio frecuencial aplicado al caso bidimensional de las imágenes, Figura 4, y además acompaña de forma gráfica la explicación de los entresijos de la representación frecuencial. Tanto en la versión que obtiene la transformación de *Fourier* (la DFT), como en la debida a la transformada del coseno (la DCT). Cabe destacar que:

- De las componentes reales e imaginarias se derivan los valores de magnitud y fase que habitualmente vemos representados como en la Figura 4 izquierda.
  Observe que el valor negro en la componente de Magnitud representa valores altos y, en la de Fase, valores cercanos a cero.
- La definición 1D de la DFT se extiende al caso 2D, Figura 4 derecha, generalizándola al considerar que el número de puntos depende ahora de las dimensiones de la imagen.

En este trabajo, también utilizaremos versiones en gris de las imágenes por brevedad en el código que mostraremos como parte de la implementación. En [4] podemos encontrar la validación de estas simplificación y la generalización para una imagen en color (RGB)

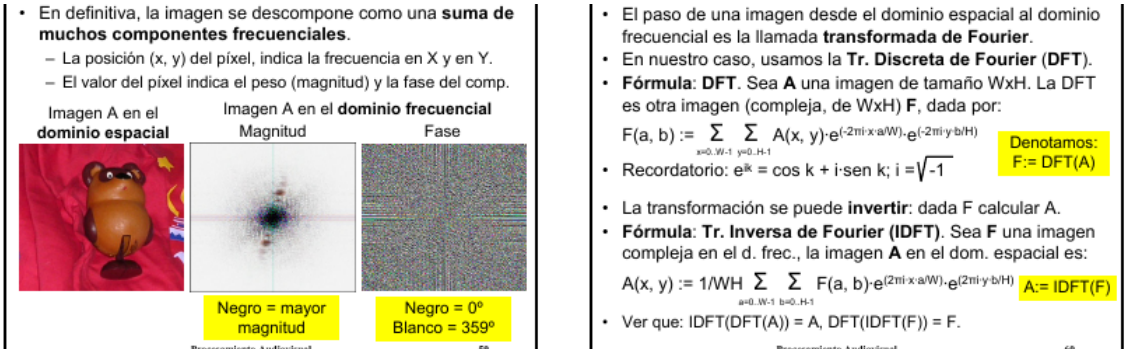
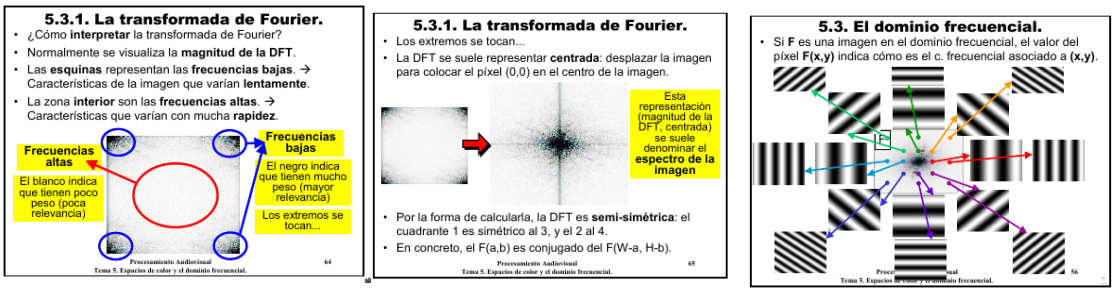


Figura 4: Visualización de los elementos que definen la representación frecuencial y su contextualización en el caso 2D. Imágenes extraídas de [4].

Respecto a los cálculos que muestra la Figura 4, hay que recalcar que se asumen ciertas cuestiones:

- No son los valores que se obtienen en primera instancia de la transformada, que son valores complejos y tienen, por tanto, parte real y parte imaginaria. La elaboración de estos valores en la pareja magnitud y fase los hace más cercanos a una interpretación física de los valores obtenidos.
- Como valores derivados que son la magnitud y la fase, implican un proceso de elaboración de la información. Así, es costumbre representar la componente de magnitud como una imagen; no como se obtiene inicialmente del cálculo del módulo del valor complejo (Figura 5a), sino que se suele modificar la disposición original de los cuadrantes de la misma (Figura 5b), para ofrecer una interpretación física de sus valores (Figura 5c). La Figura 6 muestra otros ejemplos de la magnitud o espectro de frecuencias.



a) b) c)

Figura 5: Componentes derivados del análisis frecuencial: el caso de la magnitud o espectro de frecuencias. Imágenes extraídas de [4].

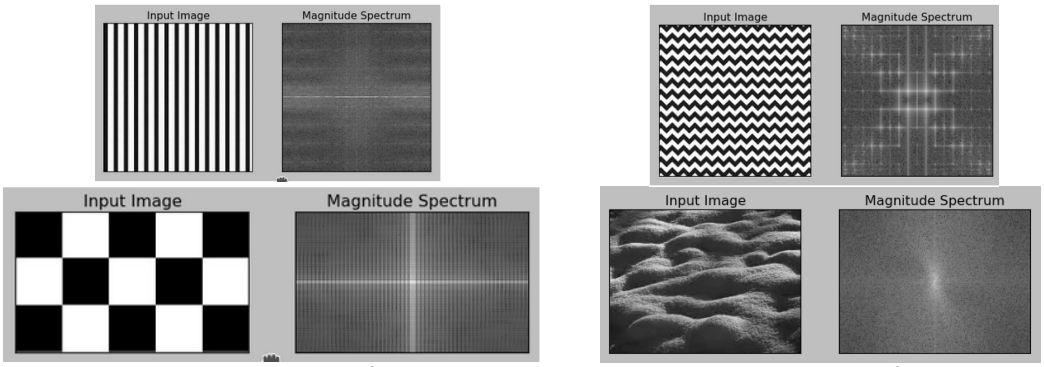
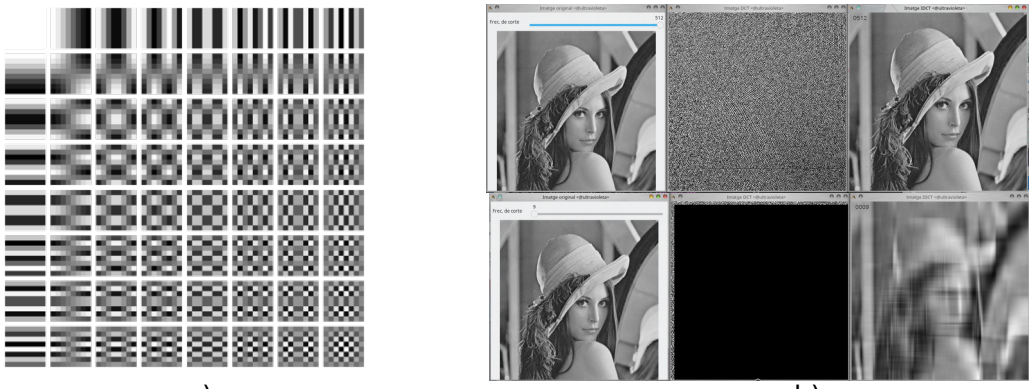


Figura 6: Ejemplos de imágenes y componente de magnitud. Imágenes de [5].

El caso de la DCT no es menos complejo conceptualmente, pero su resultado es un solo elemento, Figura 7a; por lo que, quizá, resulte más obvio en un primer contacto. La Figura 7b muestra la visualización de la imagen original y la reconstruida arropando a la representación en frecuencias (la DCT que está en el medio). Modificar esta DCT, como se ve en la fila inferior de la Figura 7b influye en el resultado: al borrar elementos de la DCT, la versión reconstruida (IDCT) pierde información respecto a la imagen original.



a) b)

Figura 7: El uso de DCT: (a) representación visual de las componentes frecuenciales que utiliza la DCT y (b) secuencias de imagen original, DCT e imagen reconstruida (IDCT).

4 Desarrollo

Para ver como también existe esa dependencia entre la DFT y su versión reconstruida, vamos a ver el proceso análogo a lo que como hemos visto en la DCT en la Figura 7. Para ello nos hemos basado en ejemplos de código ya existentes. Por un lado, *OpenCV* [3] ofrece un ejemplo de cómo aplicar la DFT al caso 2D, pero no calcula ni visualiza la fase o la IDFT. Y por otro, Toennies [6] (que utiliza [3] y tampoco hace uso de la fase, ni de la IDFT) que introduce la comparación de la DFT y la DCT y muestra cómo la modificación de una parte de la representación frecuencial corresponde con la pérdida de detalles en la imagen y se hace visible al comparar la original con la reconstruida (IDCT).

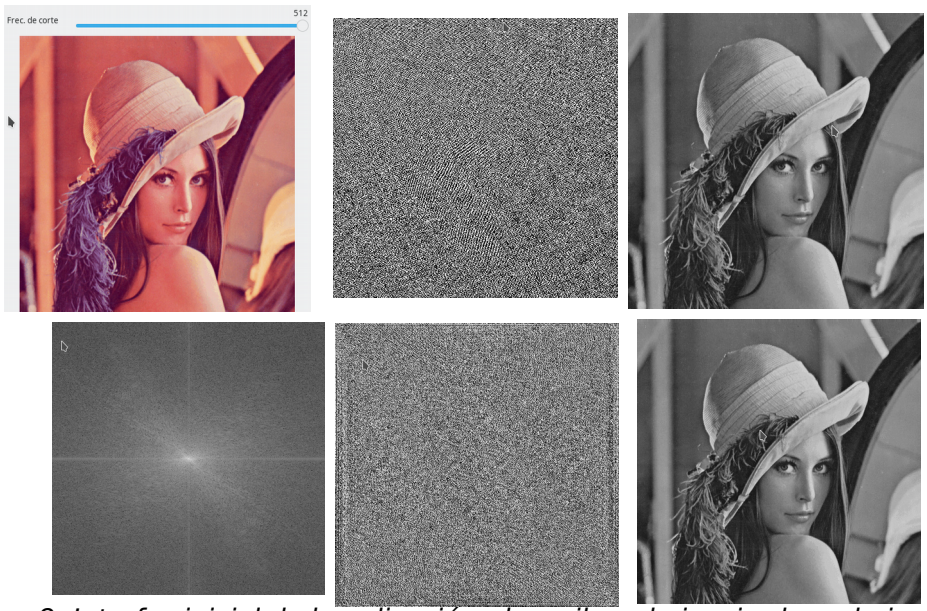


Figura 8: Interfaz inicial de la aplicación, de arriba a la izquierda a abajo a la derecha: imagen original, DCT e IDCT, magnitud y fase de la DFT y IDFT.

La Figura 8 muestra las ventanas de la aplicación desarrollada. En la parte de arriba se muestra desde la imagen original hasta la reconstruida y, en medio, la matriz de valores de frecuencias (DCT) calculada. Para ello, el Listado 1 muestra cómo [6] elabora una visualización de la matriz DCT a partir de los métodos directo e inverso de esta transformación que ofrece *OpenCV*. Observe que se hacen ajustes en la dimensión y tipo de la imagen original, porque la DCT espera que sean valores pares y que sea valores de tipo real, en lugar de los 8 bits por píxel de la representación espacial en escala de grises.

Por su parte, las imágenes de la fila inferior de la Figura 8 muestran el trabajo utilizando la DFT. No se ha visualizado el resultado en forma compleja (esto es, las componentes reales e imaginarias que se obtienen como resultado de la aplicación de la transformada de *Fourier*), sino la elaboración de estas componentes en los valores de magnitud y fase, tras los cuales se muestra la imagen reconstruida por la aplicación de la IDFT sobre los valores iniciales de valores complejos.

```
…
// Read image
Mat img = imread(argv[0], CV_LOAD_IMAGE_GRAYSCALE);
// Make sure the both image dimensions are a multiple of 2
Mat img2, img3, freq, dst;
int w = img.cols, h = img.rows, w2,h2;
if (w % 2 == 0)     w2 = w; else     w2 = w+1;
if (h % 2 == 0)     h2 = h; else     h2 = h+1;
copyMakeBorder(img, img2, 0, h2-h, 0, w2-w, IPL_BORDER_REPLICATE);
// Grayscale image is 8bits per pixel, but dct()  wants float values!
img3 = Mat( img2.rows, img2.cols, CV_64F);
img2.convertTo(img3, CV_64F);
// Let's do the DCT now: image => frequencies
dct(img3, freq);
// Visualiza los coeficientes de la DCT
imshow(FDCT, freq);
//
// Aquí se puede modificar la matriz “freq” que es la que contiene la DCT
//
// Do inverse DCT: (some low) frequencies => image
idct(freq, dst);imshow(FIDCT, dst);
...
```

Listado 1: Parte central del código que implementa la transformación DCT [6].

La implementación de las operaciones relativas a la DFT se pueden ver en el Listado 2 y Listado 3. Aunque la DFT se puede aplicar para una imagen de cualquier número de filas y columnas, es recomendable utilizar el método *getOptimalDFTSize* para obtener una versión de la imagen original que optimice la aplicación de la transformada por si hay que ejecutarla con restricciones temporales.

La magnitud se ha obtenido, recuérdese la Figura 1, del ejemplo de [3] (el mismo que utiliza [6]) implementando el cálculo del módulo de la representación vectorial de un número complejo. Observe que tras el cálculo de la DFT sobre una variable compleja (*complexI*), esta es descompuesta en la

parte real (*planes[0]*) e imaginaria (*planes[1]*). La función *magnitude* de OpenCV obtiene el módulo del vector (M_0) y lo reescala en forma logarítmica porque el rango dinámico de *Fourier* no permitiría observar en pantalla el rango de valores obtenido. Este valor así calculado de la magnitud no es el que se muestra habitualmente (lo mencionábamos en la Figura 5), se dibuja con las bajas frecuencias en el centro y no en las esquinas como se obtiene inicialmente, por lo que se intercambian los cuadrantes para obtenerla.

```
...
 Mat I = imread(filename, IMREAD_GRAYSCALE);
 if( I.empty()) return -1;
 Mat padded; //expand input image to optimal size
 int m = getOptimalDFTSize( I.rows );
 int n = getOptimalDFTSize( I.cols ); // on the border add zero values
 copyMakeBorder(I, padded, 0, m - I.rows, 0, n - I.cols,
                BORDER_CONSTANT, Scalar::all(0));
 Mat planes[]={Mat_<float>(padded), Mat::zeros(padded.size(),CV_32F)};
 Mat complexI;
 merge(planes, 2, complexI); // Add another plane with zeros
 dft(complexI, complexI);
 // => log(1 + sqrt(Re(DFT(I))^2 + Im(DFT(I))^2))
 split(complexI, planes); // planes[0] = Re(DFT(I),
                                 // planes[1] = Im(DFT(I))
 magnitude(planes[0], planes[1], planes[0]);// planes[0] = magnitude
 Mat magI = planes[0];
 magI += Scalar::all(1);  log(magI, magI); // to logarithmic scale
 // crop the spectrum, if it has an odd number of rows or columns
 magI = magI(Rect(0, 0, magI.cols & -2, magI.rows & -2));
 // rearrange the quadrants of Fourier image so that the origin is at the
 // image center
 int cx = magI.cols/2;  int cy = magI.rows/2;
 Mat q0(magI, Rect(0, 0, cx, cy)); // Top-Left - a ROI per quadrant
 Mat q1(magI, Rect(cx, 0, cx, cy)); // Top-Right
 Mat q2(magI, Rect(0, cy, cx, cy)); // Bottom-Left
 Mat q3(magI, Rect(cx, cy, cx, cy)); // Bottom-Right
 Mat tmp; // swap quadrants (Top-Left with Bottom-Right)
...
 // Transform the matrix with float values into a
 // viewable image form (float between values 0 and 1).
 normalize(magI, magI, 0, 1, NORM_MINMAX);
 imshow("Input Image" , I ); imshow("spectrum magnitude", magI);
...
```

Listado 2: Código de cálculo de la DFT y la magnitud de la DFT. Extraído de <https://docs.opencv.org/3.2.0/d8/d01/tutorial_discrete_fourier_transform.html>.

Falta por ver el cálculo de la fase y cómo obtener la inversa de la DFT. OpenCV nos ayudará con los métodos *phase* e *idft* que hacen[2] justo lo que necesitamos. El Listado 3 lo muestra de forma abreviada y resaltando las funciones principales.

```
...
 dft(complexImg, complexImg);
// => log(1 + sqrt(Re(DFT(I))^2 + Im(DFT(I))^2))
 split(complexImg, planes);
//
// Aquí se puede modificar los valores complejos de la DFT:
//
//   Re(DFT(img)) → planes[0] y Im(DFT(img)) → planes[1]
 magnitude(planes[0], planes[1], magnitut_dft );
 phase(planes[0], planes[1], fase_dft, true ); // true -> graus
// IDFT
 idft(complexImg, inverseTransform, DFT_REAL_OUTPUT);
// Es equivalente a hacer
// dft(complexImg, inverseTransform, DFT_INVERSE|DFT_REAL_OUTPUT);
// Para visulizar la IDFT
 normalize(inverseTransform, inverseTransform, 0, 255, CV_MINMAX);
 inverseTransform.convertTo( inverseTransform, CV_8UC1 );
// magnitud en rango logarítimico
 magnitut_dft += Scalar::all(1); log( magnitut_dft, magnitut_dft );
// rearrange the quadrants of Fourier image. como en el Listado 2
// Para visulizar la magnitud y la fase
 normalize(magnitut_dft, magnitut_dft, 0, 255, CV_MINMAX);
 magnitut_dft.convertTo(magnitut_dft, CV_8UC1);
 normalize(fase_dft, fase_dft, 0, 255, CV_MINMAX);
 fase_dft.convertTo( fase_dft, CV_8UC1 );
 imshow(FDFT_MAGNITUT, magnitut_dft ); imshow(FDFT_FASE, fase_dft);
 imshow(FIDFT, inverseTransform );
...
```

Listado 3: Código ampliado a cálculo de la fase, la inversa dela DFT y la posible modificación de los valores reales e imaginarios de la DFT.

En este listado hay un paso comentado que es donde introducimos qué componentes se pueden manipular en esta representación. Podríamos modificar la magnitud y la fase y después deberíamos hacer la inversa de estos valores para dejar los valores modificados en las variables que contienen la parte real y la imaginaria que son necesarias para aplicar la IDFT. En su lugar y por similitud con el código de la DCT, hemos modificado la variable *planes* y mostraremos la magnitud y fase resultantes de esos cambios. Hemos dejado indicado dónde hacerlo.

La Figura 9 muestra dos ejemplos de lo que se puede ver: a pesar de los recortes (las áreas en negro en la DCT y en la magnitud/fase), la imagen reconstruida mantiene mucha información y no parece verse afectada.

2 Véase *Operations on arrays. Core functionality*, disponible en <https://docs.opencv.org/3.2.0/d2/de8/group__core__array.html#gaa708aa2d2e57a508f968eb0f69aa5ff1>.

¿Impresionante no? Bueno, es que solo hemos modificado altas frecuencias, esto es, cambios locales y poco apreciables visualmente. Pero si subimos los parámetros de la aplicación propuesta los veríamos.

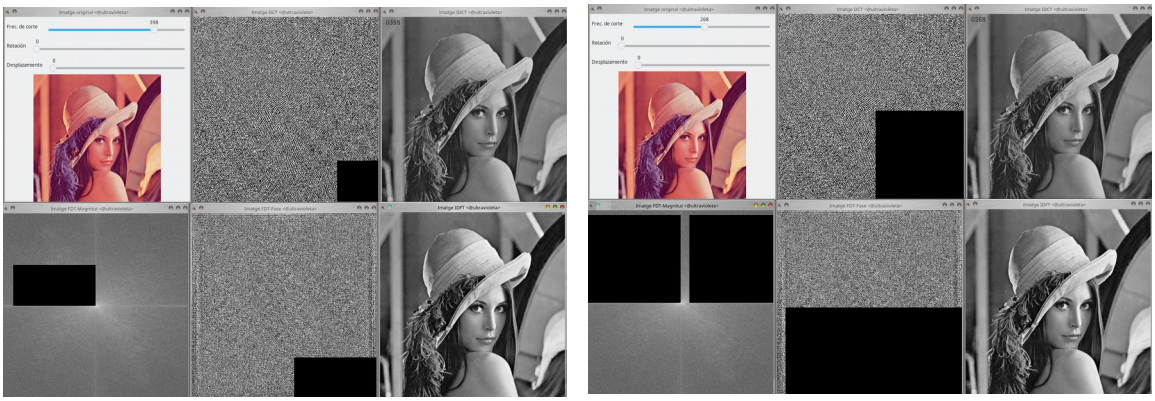


Figura 9: Ejemplos de edición de la DCT y de la magnitud/fase de la DFT. El valor reconstruido sigue mostrando una alta similitud con la imagen de partida.

5 Conclusión

Como se propone en los objetivos, hemos visto que no se modifica el contenido de una imagen al aplicar una transformación en el dominio frecuencial. Es equivalente al domino espacial habitual de los píxeles. Hemos visto cómo usar las transformaciones DFT y DCT. Que si modificamos los resultados de la aplicación directa, esto afectará en la inversa para recuperar la imagen de partida. El grado de degradación dependerá de lo que se hayan modificado los valores de frecuencias de cada transformación y que son muy diferentes en ese sentido. Y, sobre todo, hemos "visto" lo que sucede dentro de ellas y, así, comparar los resultados de la DFT y la DCT.

Si está interesado en el código completo para visualizar las etapas intermedias y explorar con los valores no tiene más que enviarme un correo electrónico y se lo haré llegar.

6 Bibliografía

[1] S.W. Smith, (1997). The Scientist and Engineer's Guide to Digital Signal Processing. ISBN 978-0966017632. Disponible en <http://www.dspguide.com/ >.

[2] *Transformation de Fourier discrète*. Wikipedia, la enciclopedia libre. Disponible en <http://fr.wikipedia.org/w/index.php?title=Transformation_de_Fourier_discr%C3%A8te&oldid=153039054>. .

[3] OpenCV (Open Source Computer Vision Library). Disponible en <https://opencv.org/>.

[4] G. García-Medrano. (2014). Procesamiento Audiovisual, Dept. de Informática y Sistemas, Universidad de Murcia. Disponible en <http://dis.um.es/~ginesgm/files/doc/pav/tema5.pdf>.

[5] K Hong. (). Signal Processing with NumPy II - Image Fourier Transform : FFT & DFT. Disponible en <https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Signal_Processing_with_NumPy_Fourier_Transform_FFT_DFT_2.php>.

[6] K. Toennies . (2012). DCT, DFT, and DST. Disponible en <http://www.juergenwiki.de/old_wiki/doku.php?id=public:dft_dct_dst>.