# OpenCV: Filtrado frecuancial

FICH, UNL - Procesamiento Digital de Imágenes
Walter Bedrij

28 de abril de 2016

## 1. Funciones para el cálculo de la TDF

dft() idft()

Recibe como parámetros la imagen de entrada y donde se almacenará el resultado, además de banderas que controlan el comportamiento.

Con dft se puede realizar tanto la transformada directa (flag DFT_SCALE) o inversa (flag DFT_INVERSE) que se encargará de realizar el escalado correspondiente.

Para operar con estas funciones se requiere una representación de punto flotante (CV_32F o CV_64F)

Si bien la transformada opera en el plano complejo, se puede forzar que el resultado sea real (si se está seguro de que la entrada es simétrica conjugada) mediante DFT_REAL_OUTPUT

Al realizar los filtros se trabaja en el espacio complejo, forzar con DFT_COMPLEX_OUTPUT

## 2. Relleno

Con el fin de realizar el cálculo de forma eficiente, se realiza un relleno con 0 a la imagen de entrada.

```
rows = cv::getOptimalDFTSize(image.rows);
cols = cv::getOptimalDFTSize(image.cols);
cv::copyMakeBorder(
  image,
  result,
  0, rows-image.rows, //borde abajo
  0, cols-image.cols, //y a la derecha
  cv::BORDER_CONSTANT,
  cv::Scalar::all(0)
);
```

Para esto se provee de la función pdi::optimum_size() image = pdi::optimum_size(image);

Se recomienda guardar el tamaño original de la imagen para poder eliminar el relleno luego de aplicar el filtro.

## 3. Definición de filtros en el plano frecuencial

Se pueden definir filtros al especificar la magnitud de su espectro, dejando la fase en 0. Luego, para aplicarlos simplemente se multiplican los espectros del filtro y la imagen. Debe considerarse que como se está operando en f, se trata de imágenes complejas.

```
cv::polarToCart(magnitud, phase, x[0], x[1]);
cv::merge(x, 2, filter);
mulSpectrums(image_f, filter, result, cv::DFT_ROWS);
```

Se proveen funciones para realizar filtros ideales, de butterworth y gaussiano.

## 4. Visualizado

La imagen de la transformación se encuentra descentrada y es compleja. Para centrarla, se la divide en cuadrantes y se intercambian los que tienen sólo un vértice en común. Así, numerando en sentido antihorario, se intercambiaría el cuadrante 1 con el 3, y el 2 con el 4. Esto lo realiza la función pdi::centre()

Como la imagen es compleja no puede visualizarse. Lo que se visualiza es el logaritmo de su espectro.

```
cv::Mat planes[2];
cv::split(image, planes);

cv::Mat result;
cv::magnitude(planes[0], planes[1], result);
result += 1; //evitar el cálculo de log(0)
cv::log(result, result);
```

El algoritmo se encuentra implementado en pdi::spectrum()

## 5. Proceso de filtrado

```
#include <opencv2/opencv.hpp>
#include "pdi_functions/pdi_functions.h"

int main(int argc, char **argv){
  if(argc not_eq 2) return 1;

  cv::Mat image = cv::imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
  //convertir a 32F o 64F
  image.convertTo(image, CV_32F, 1./255);
  //ajustar el tamaño para que sea más eficiente
  int
    original_rows = image.rows,
    original_cols = image.cols;
  image = pdi::optimum_size(image);

  //crear un filtro (descentrado)
```

```
    cv::Mat ideal = pdi::filter_ideal(image.rows, image.cols, 0.1);

    //aplicar el filtro
    cv::Mat result = pdi::filter(image, ideal);

    //eliminar el relleno
    result = result(
      cv::Range(0, original_rows),
      cv::Range(0, original_cols)
    );
    image = image(
      cv::Range(0, original_rows),
      cv::Range(0, original_cols)
    );

    //visualizado
    cv::namedWindow("image", CV_WINDOW_KEEPRATIO);
    cv::namedWindow("image_espectro", CV_WINDOW_KEEPRATIO);
    cv::namedWindow("filtrada", CV_WINDOW_KEEPRATIO);
    cv::namedWindow("filtrada_espectro", CV_WINDOW_KEEPRATIO);
    cv::imshow("image", image);
    cv::imshow("image_espectro", pdi::spectrum(image));
    cv::imshow("filtrada", result);
    cv::imshow("filtrada_espectro", pdi::spectrum(result));
    cv::waitKey();

    return 0;
}
```

pdi:: filter () supone un filtro de fase nula. Si se requiere especificar la fase, entonces el proceso *aplicar filtro* sería

- realizar la transformación de la imagen cv::dft(), pasando como flag DFT_COMPLEX_OUTPUT
- multiplicar los espectros de la imagen y del filtro con cv::mulSpectrums(). Debe tenerse especial cuidado en que tanto el espectro de la imagen como el filtro se encuentren descentrados.
- antitransformar cv:: idft () utilizando los flags DFT_SCALE y DFT_REAL_OUTPUT

Se recomienda analizar el código de pdi:: filter ()

En general, resulta más sencillo especificar el filtro en términos de su magnitud y fase en lugar de componente real e imaginaria. Sin embargo, mulSpectrums() espera este último formato. Debido a esto se provee la función pdi::polar_combine(), que devuelve una matriz compleja a partir de la magnitud y fase.