Instituto Tecnológico de Buenos Aires

22.48 - Procesamiento de imágenes

Trabajo Práctico N°2

Guía 2

Grupo 1:

Matías Agustín
Larroque
Leg. 56597

Tomás Agustín
González Orlando
Leg. 57090

Manuel Fernando
Mollón
Leg. 58023

Profesores:

Daniel Jacoby

Entregado: 3 de Septiembre de 2020

Representación de imagenes y sus espectros

1 Introducción

Este trabajo práctico tiene como objetivo proponer códigos de ejemplos, que implementen la respresentación de imagenes y sus espectros, y algunos efectos que pueden sufrir estas imagenes al ingresar a sistemas típicos para el procesamientode imágenes.

A continuación se plantean los ejemplos.

Este informe tan solo pretende acompañar la práctica de programación realizada y la interpretación de resultados. Cada ejercicio de este artículo tiene su código ".py" asociado.

1.1 ex1.py

En este ejercicio se propone convolucionar una delta con un sistema 'h', y ver que efectivamente se logra ver el espectro del sistema h, ya que la resultante de la convolución con una delta es ella misma. Además se pretende notar la respuesta del sistema. A continuación se muestra el código para implementar lo propuesto.

```
from scipy import signal
from numpy.fft import fft2, fftshift, ifft2
from numpy import log
import cv2 as cv
import numpy as np
h = np.array([[0, 1/6, 0], [1/6, 1/3, 1/6], [0, 1/6, 0]])

# construyo una imagen (que es una delta en 0)
# suficientemente grande para poder bien el espectro

N = 512
big = np.zeros((N,N))   #make a big image
big[int(N/2)][int(N/2)] = 1     #unit impulse

h1 = signal.convolve2d(big, h) # conv with the filter
S = fft2(h1) # Spectrum
SM = abs(S) # Modulo
IMd = log(1+abs(SM))/log(10)

cv.imshow(mat = fftshift(SM/SM.max()), winname = 'Spectrum')
cv.waitKey(0)
cv.imshow(mat = fftshift(IMd/IMd.max()), winname = 'Spectrum Log Scale')
cv.waitKey(0)
```

Figure 1: Código propuesto ex1.py

La respuesta en frecuencia obtenida fue la siguiente:

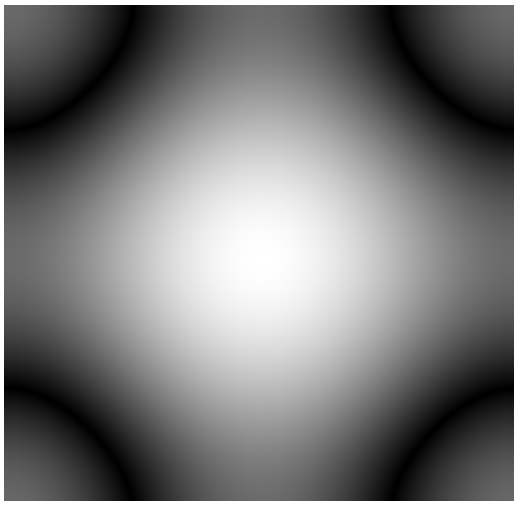

Figure 2: Respuesta en frecuencia - Escala lineal

Esta respuesta en escala lineal (que no presenta diferencias notables con la escala logarítmica), demuestra ser un pasabajos, ya que se nota el moy brillo en el centro de la imagen (frecuencias bajas).

Además, se pueden notar satisfactoriamente en los arcos negros del espectro, los ceros que se impusieron en las esquinas de la "h".

Finalmente, vale aclarar que fue necesario convolucionar con la delta para poder visualizar el espectro de "h", ya que si se hubiera aplicado la transformada de Fourier al "Kernel" de dimensiones disminuidas que conforma la respuesta del sistema, se obtendría una imagen tan pequeña que no se apreciaría visualmente.

1.2 ex2.py

En este ejercicio se utiliza un sistema "h" pasabajos utilizado anteriormente en el código "ex1.py". Se ingresa a este sistema una imagen (la imagen utilizada es la de "Bárbara") y se pretenden desribir los efectos visuales que ocasiona un pasabajos sobre una foto/imagen.

A continuación se muestra el código para implementar el ejercicio:

```python
import cv2 as cv
from scipy.ndimage import correlate
import numpy as np

h = np.array([[0, 1/6, 0], [1/6, 1/3, 1/6], [0, 1/6, 0]])
h = h/np.sum(h)
# h = fspecial('unsharp')
# h = fspecial('disk')
img = cv.imread(filename = '../data/images/barbara.png', flags = cv.IMREAD_GRAYSCALE)
print(img.shape)

h1 = correlate(img, h) # make conv with the filter
new_mat = np.hstack((img, h1))

cv.imshow(mat = new_mat, winname = 'Original Image')
cv.waitKey(0)
```

Figure 3: Código propuesto ex2.py

Las imágenes de Bárbara, original y filtrada, son las siguientes:

Figure 4: Imagen de Bárbara original

Figure 5: Imagen de Bárbara filtrada

Los efectos sobre la imagen de Bárbara, se pueden notar en zonas de la imagen donde se manifiestan "altas frecuencias", como por ejemplo en los bordes: bordes de las líneas del pantalón, mantel de la mesa, pañuelo de la cabeza, libros en la biblioteca del fondo. Sobre estas partes de la imagen, se notan cambios menos intensos, incluso con menos brillos. Sin embargo se siguen notando estos bordes, ya que se vió en el espectro del sistema que este anula solo ciertas ciertas frecuencias, mientras que otras frecuencias altas solo las atenúa.

Esta "atenuación", se nota claramente en el sutil efecto "blur" (o empañado, o difuminado) que se manifiesta en la imagen filtrada.

2 Alliasorg.m

En este ejercicio se propuso el siguiente código para generar una imagen que recorre periódicamente los tonos de grises de forma senoidal:

```
import cv2 as cv
import numpy as np

from numpy import pi, cos, sin, log
from numpy.fft import fft2, fftshift

xsize = 1024
ysize = 1024

alpha1 = 0  # Rotationswinkel 1
alpha2 = pi/4  # Rotationswinkel 2
f1 = 180     # Frequenz 1
f2 = 8     # Frequenz 2
a1 = 0      # Amplitude 1
a2 = 1         # Amplitude 2
phase1 = pi/2
phase2 = 0

[X,Y] = np.meshgrid(np.arange(xsize)/xsize, np.arange(ysize)/ysize)
grid1 = cos(alpha1)*X + sin(alpha1)*Y
grid2 = cos(alpha2)*X + sin(alpha2)*Y
im = a1*sin(2*pi*f1*grid1 + phase1) + a2*sin(2*pi*f2*grid2 + phase2)
print(((im+1)*255/2).astype(np.uint8))
cv.imshow(mat=((im+1)*255/2).astype(np.uint8), winname='IMAG 1')
cv.waitKey(0)

IM = fft2(im)
IMd = log(1+abs(IM))/log(10)
cv.imshow(mat=fftshift(IMd/IMd.max()), winname='IMAG 2')
cv.waitKey(0)
```

Figure 6: Código propuesto

Se puede notar que la imagen depende de variables definidas en el código, como lo son la frecuencia, la fase y la rotación. Estos pueden ser considerados parámetros para observar los cambios espaciales en la imagen y también los cambios espectrales. Con la configuración que se muestra en la figura (frecuencia = 180, fase = 90° y rotación = 45°) se obtuvo la imagen que se muestra a continuación, junto con su espectro:

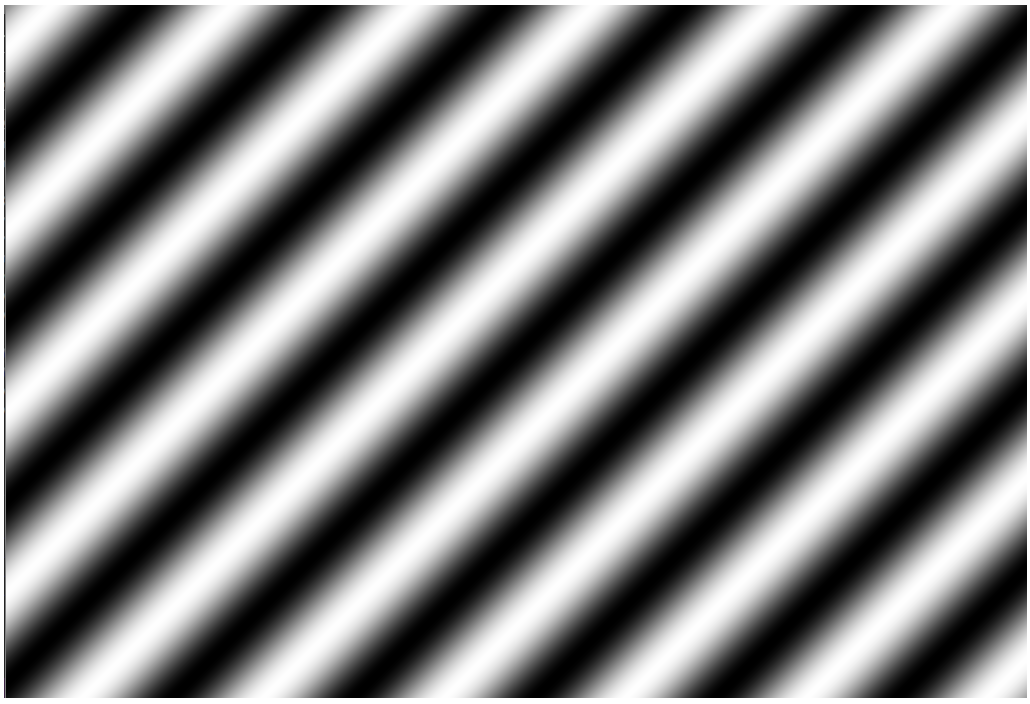

Figure 7: Imagen rotada 45° - “degrade” senoidal

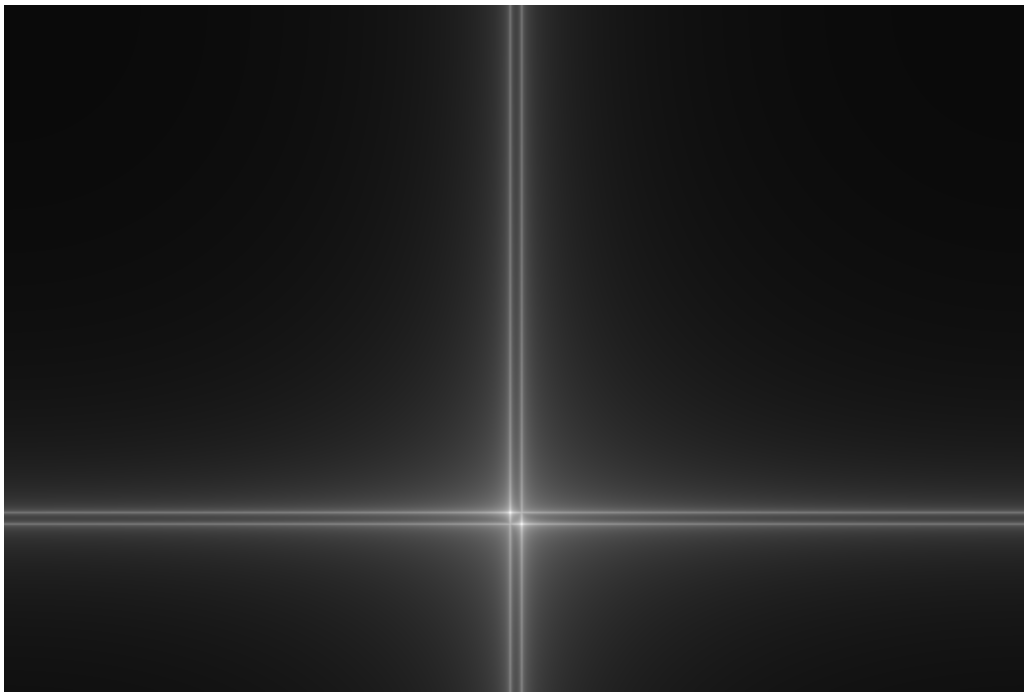

Figure 8: Espectro del degrade senoidal rotado

Para interpretar el espectro de estas señales de una manera gráfica, se generará nuevamente la imagen, pero con rotación de 90 grados y también sin rotación:

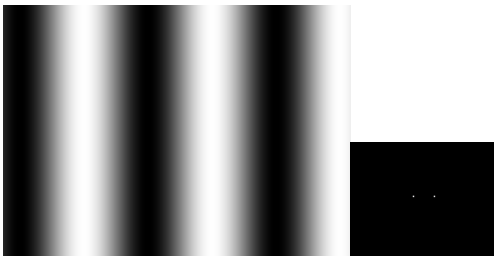

Figure 9: Imagen Líneas Verticales - “degrade” senoidal Horizontal

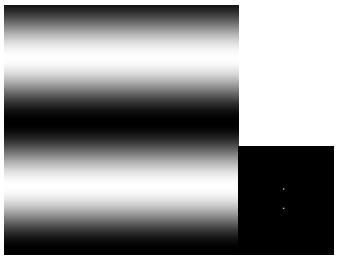

Figure 10: Imagen Líneas Horizontales - “degrade” senoidal Vertical

Lo que se logra apreciar en el espectro de las imagenes, es que existen un par de “deltas” simétricas con respecto al origen (definido por el centro de la imagen), y sobre un eje, el cual es el eje en el que la imagen sufre el cambio senoidal a una frecuencia definida.

Se cumple esto para todas las rotaciones, es decir que al rotar la imagen también rota el espectro.

Una novedad que puede verse es que el espectro de la imagen rotada a 45°, no solo cuenta con las deltas de dirac ya mencionadas, si no que además presenta líneas que parecen continuas y se intersectan formando 4 cruces. Sobre esto, se pudo corroborar que no es un efecto visible en un especrtro con escala lineal, si no que es generado debido a una distorsión introducida por la escala logarítmica, la cual, por cuestiones de saturación de funciones como “imshow”, es necesaria graficarla realizando log(1+FFT), en lugar de log(FFT) como se haría normalmente para análisis de señales temporales y de una sola dimensión.

2.1 myallias.m

Se pretende estudiar los efectos del aliasing en las imágenes.

Para cumplir con el Teorema de Shannon, para muestrear a una señal sin aliasing debe necesariamente contarse con una señal limitada en banda. Es por esto que, previo a realizar el muestreo, se suele aplicar un filtro pasabajos o de blurring para limitar en banda a la señal a muestrear y así disminuir los efectos de aliasing.

Los efectos de aliasing se pueden apreciar en los patrones de Moiré:

Cuando se cuenta con una señal periódica de una frecuencia al menos dos veces menor a la frecuencia de sampleo, no hay aliasing en la imagen. Cuando se superponen dos imágenes periódicas y hay un "quiebre" en la perioricidad de una imagen debido a la presencia de la otra, se aprecian los patrones de Moiré, en los cuales aparecen frecuencias extra en la imagen.

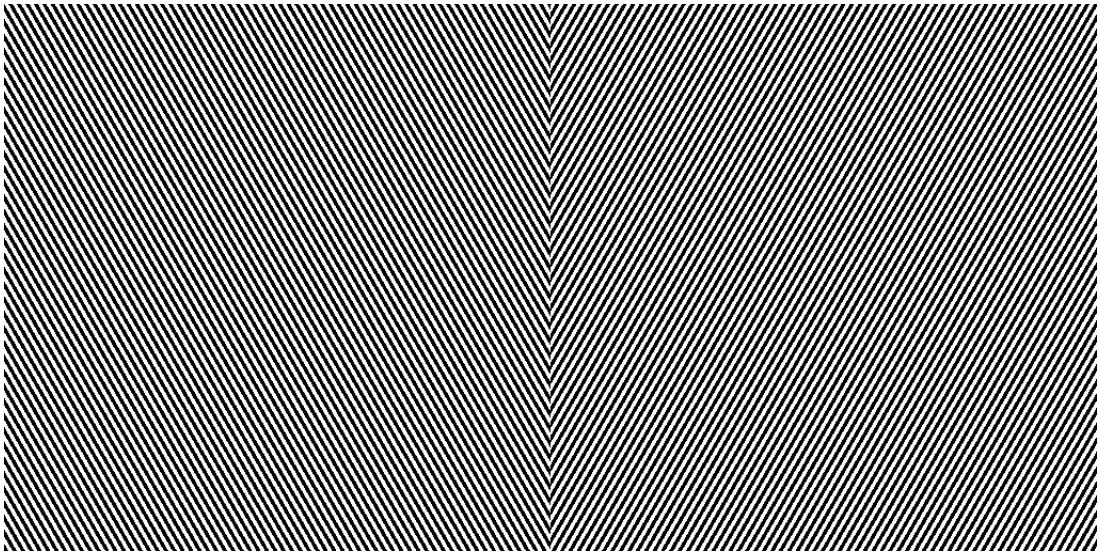

Figure 11: Dos imágenes periódicas

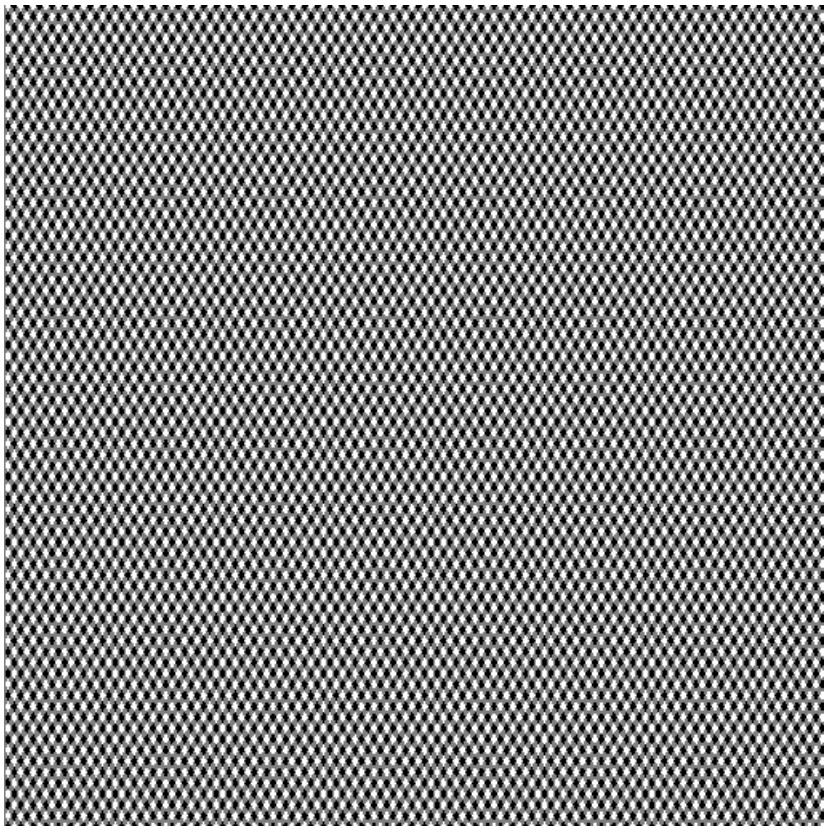

Figure 12: Dos imágenes periódicas superpuestas. Patrones de Moiré

Se muestra a continuación el espectro de las imágenes por separado:

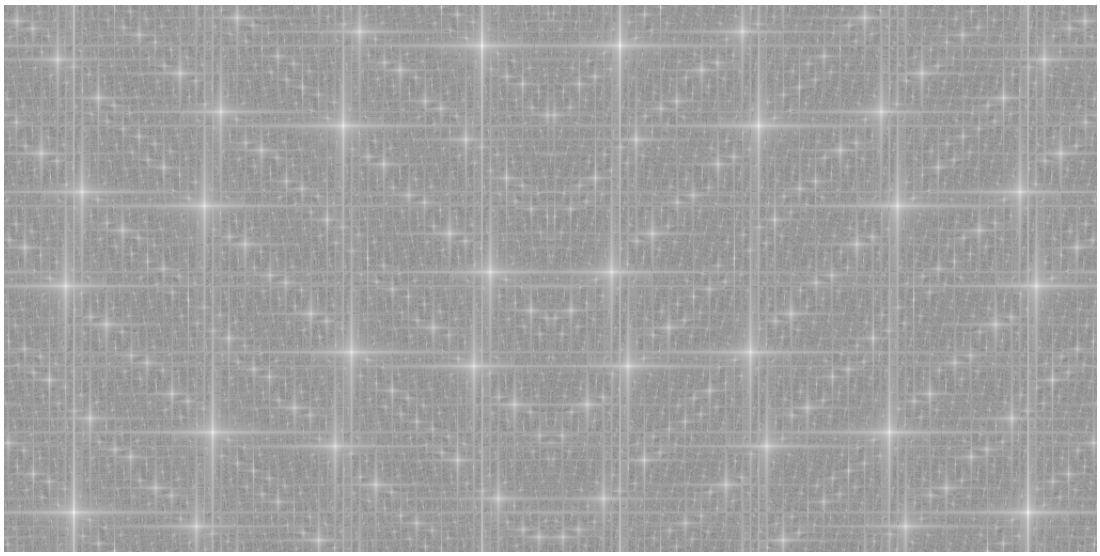

Figure 13: Dos imágenes periódicas superpuestas. Patrones de Moiré

Se muestra a continuación el espectro de las imágenes superpuestas. Aparecen las componentes en frecuencia superpuestas:

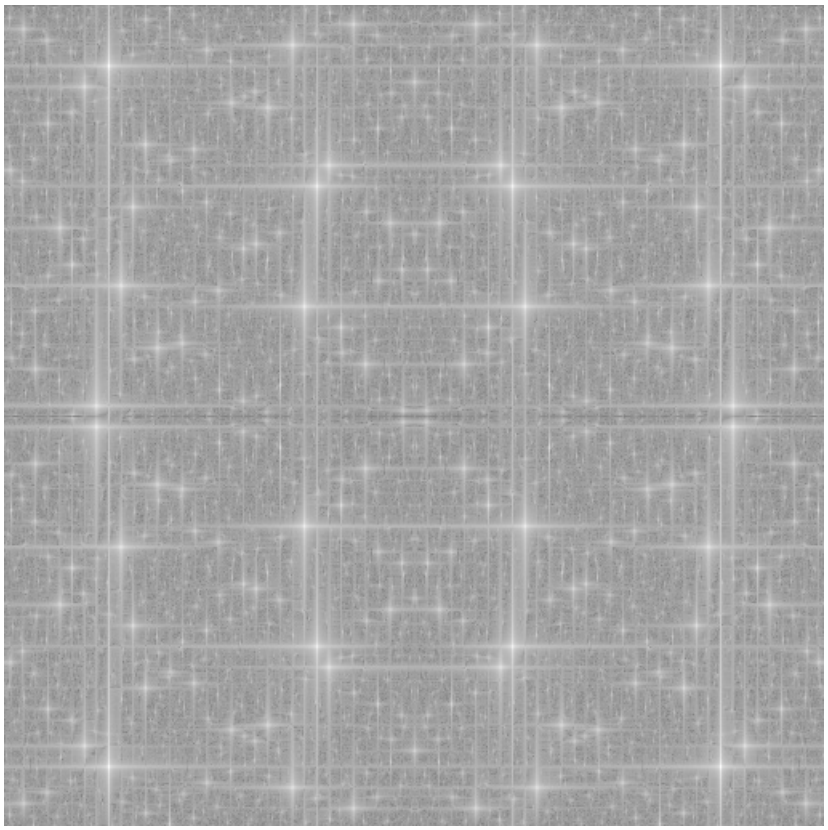

Figure 14: Dos imágenes periódicas superpuestas. Patrones de Moiré

3 Interpol.m

3.1 Downsample

Se realiza un downsampling con dos métodos de interpolación Distintos (Nearest Neighbours e interpolación Bicúbica) distintos utilizando la librería opencv. Se prueba con distintos valores de m (factor de downsampling). Se utiliza el tamaño de imagen original (720x580) y luego se realiza sobre la misma el downsample:

1. m=1 (Sin downsample, imagen original)

Figure 15: Imagen original sin downsamplear

De la imagen anterior, se aprecian detalles carácterísticos como las rayas de los pantalones, las rayas del mantel y del pañuelo de la mujer. Se aprecia por otro lado la suavidad de la cara y de la piel en conjunto con el piso.

1. m = 2

Figure 16: Izquierda: NN, Derecha:Bicúbica

Con un factor de downsample de 2, se aprecian efectos de aliasing (aparecen los efectos de moiré en el pantalón de forma circular). Esto se debe a que no se está cumpliendo con el teorema de sampleo de Shanon de forma notable.

Se compara el Nearest Neighbours con la bicúbica y parecería que los efectos del aliasing son menos profundos con la interpolación bicúbica.

1. m = 3

Figure 17: Izquierda: NN, Derecha:Bicúbica

Lo mencionado para la imagen anterior se cumple también para esta imagen y para la imagen con un mayor factor de downsample. Se observa, como era de espear, que a medida que aumenta el factor de downsample, también lo hace el aliasing apreciable. Parecería ser que la bicúbica sigue teniendo mejor performance que NearestNeighbour.

1. m = 5:

Figure 18: Izquierda: NN, Derecha:Bicúbica

3.1.1 Método de Nearest Neighbours

El método de Nearest Neighbours para realizar downsample como caso general consiste en, conceptualmente, superponer una grilla con cantidad de agujeros en cada eje iguales a la dimensión de la imagen final en su respectivo eje. Como se está realizando downsample, queda claro que más de un pixel entrará en cada uno de los agujeros de la grilla, y se elige el valor del píxel que se considera más cercano al agujero de la grilla.

Vale mencionar que la analogía inversa vale para cuando se está intentando realizar un upsampling de la imagen a través de interpolación Nearest Neighbours

Así se obtiene la nueva imagen. Otra posible implementación del downsample a través de Nearest Neighbours surge en el caso particular en que el factor m de downsample sea un número entero (como son los casos mostrados anteriormente), en el cual basta con eliminar 1 de cada m píxeles de la imagen original para obtener la nueva imagen.

3.1.2 Soluciones al problema de aliasing

Para reducir los efectos de aliasing al realizar downsample, se suele aplicar un efecto de blur antes de realizar el downsample.

Cabe destacar que específicamente para downsampling, la librería openCV recomienda el uso del método de interpolación INTER_AREA que realiza un resample usando la relación en área de pixel, el cual da resultados libres de moiré, es decir, libres de aliasing.

3.2 Filtrado

3.2.1 Filtrado Gaussiano

El filtro gaussiano es un filtro pasabajos en el cual sus pesos derivan de la función gaussiana, lo que termina siendo un filtro pasabajos implementado a partir de un promedio ponderado, de mayor influencia por parte del píxel central del kernel y menor influencia a medida que me alejo de dicho pixel en el kernel.

Se utilizó el filtrado Gaussiano de la librería opencv. Dicha función no acepta kernels de tamaño par (no acepta realizar un filtrado, por ejemplo, con un kernel gaussiano de 8x8, pero sí con uno de 9x9).

Para discretizar la función gaussiana, usamos el hecho de que el 99.3% de la distribución cae dentro de los 3 devíos estándar alrededor de la media, luego este intervalo los valores de la gaussiana serán muy cercanos a cero. Es por esto que la función de opencv se limita a tomar valores en este rango.

Al discretizar a esta función de dominio infinito, se está haciendo que la función deje de integrar 1, por lo que se debe normalizar al kernel (dividiendo al kernel por la suma de todos los elementos del mismo) para que la imagen no se vuelva más brillante o más oscura que lo que era originalmente.

Hay algunas propiedades del filtro gaussiano que son de interés a la hora de hacerlo más eficiente:

- El kernel Gaussiano es linearmente separable. Esto significa que podemos separar al filtro h de 2 dimensiones en dos filtros 1D, h_1 y h_2. Gracias a esto, complejidad computational se ve reducida de $O(n^2)$ a $O(n)$. El filtro gaussiano es el único filtro que es simétrico y linealmente separable.

$$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \frac{1}{16}\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

Figure 19: Cómo pasar de un kernel Gaussiano 1D a uno 2D

- Aplicar sucesivamente kernels Gaussianos es equivalente a aplicar un único blur Gaussiano más grande, cuyo radio es la raíz cuadrada de la suma de los cuadrados de los radios de los múltiples kernel gaussianos. Usando esta propuedad, podemos aproximar un filtro no separable como una combinación de múltiples filtros separables.

- Los pesos del kernel Gaussiano 1-D pueden ser obtenidos rápidamente usando el Triángulo Pascal. Podemos apreciar como la última fila del triángulo corresponde a los pesos de la matriz mostrada en la imagen anterior.

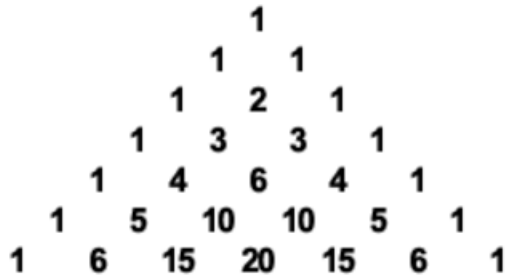

Figure 20: Triángulo de Pascal

Debido a estas propiedades, el filtro gaussiano es el más eficiente y el más comunmente usado de los filtros pasabajos/de blurring.

OpenCv permite obtener el filtro lineal 1D gaussiano para aplicarlo de forma separable así como también tiene una función específica para aplicar filtros gaussianos.

Se muestra a continuación la forma y el espectro de un kernel gaussiano de 9x9 y de $\sigma = 0.8$

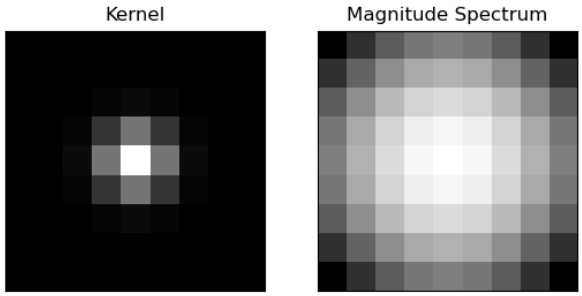

Figure 21: Forma y espectro de un kernel gaussiano de 9x9 y de $\sigma = 0.8$

Se muestra luego la superficie 3D de un kernel Gaussiano sin interpolar. Para ver la forma del filtro interpolada, ir a la próxima sección sobre filtros PillBox:

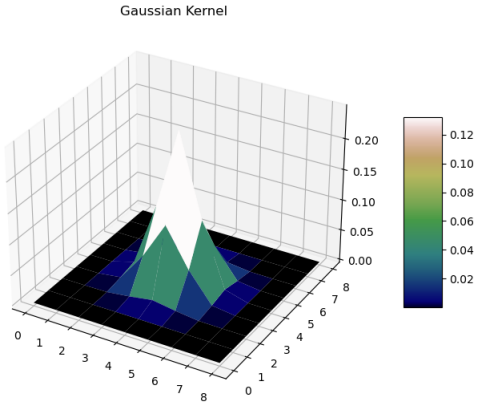

Figure 22: superficie 3D de un kernel Gaussiano de 9x9 y de $\sigma = 0.8$

Procedemos a mostrar el efecto del filtro gaussiano sobre la imagen downsampleada por un factor de dos:

Figure 23: Filtro Gaussiano de 9x9 y de $\sigma = 1$ aplicado a la imagen

Se aprecia cómo se reducen los efectos del aliasing, que siguen siendo existentes. Esto es porque el filtro gaussiano es un filtro pasabajos que elimina las frecuencias altas introducidas.

Figure 24: Filtro Gaussiano de 5x5 y de $\sigma = 1$ aplicado a la imagen

Figure 25: Filtro Gaussiano de 9x9 y de $\sigma = 5$ aplicado a la imagen

Figure 26: Filtro Gaussiano de 5x5 y de $\sigma = 5$ aplicado a la imagen

Figure 27: Filtro Gaussiano de 3x3 y de $\sigma = 5$ aplicado a la imagen

3.2.2 Filtrado Circular o PillBox

El filtro Pillbox no está implementado en openCv. Es más filoso que el gaussiano debido a ser constante en una zona y decaer muy rapidamente en los extremos. Tiene un tope circular y costados casi verticales

Se muestra el kernel circular/Pillbox:

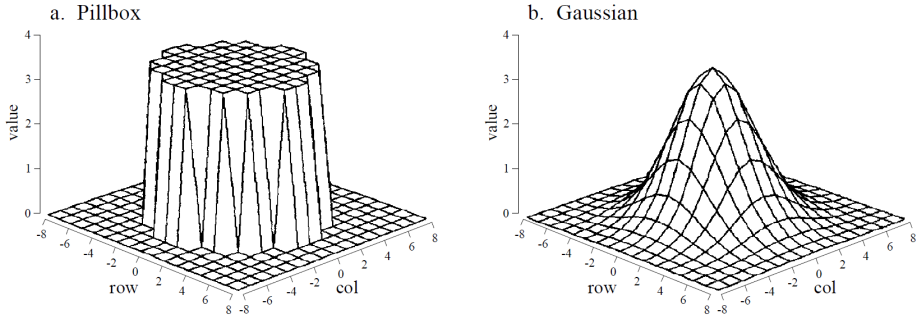

Figure 28: Filtro PillBox

Podemos también observar la forma matricial de dicho filtro (en este caso de radio 5):

0	0	0	0.0012	0.0050	0.0063	0.0050	0.0012	0	0	0
0	0.0000	0.0062	0.0124	0.0127	0.0127	0.0127	0.0124	0.0062	0.0000	0
0	0.0062	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0062	0
0.0012	0.0124	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0124	0.0012
0.0050	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0050
0.0063	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0063
0.0050	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0050
0.0012	0.0124	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0124	0.0012
0	0.0062	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0062	0
0	0.0000	0.0062	0.0124	0.0127	0.0127	0.0127	0.0124	0.0062	0.0000	0
0	0	0	0.0012	0.0050	0.0063	0.0050	0.0012	0	0	0

Figure 29: Filtro PillBox

Se muestra un kernel circular en frecuencia:

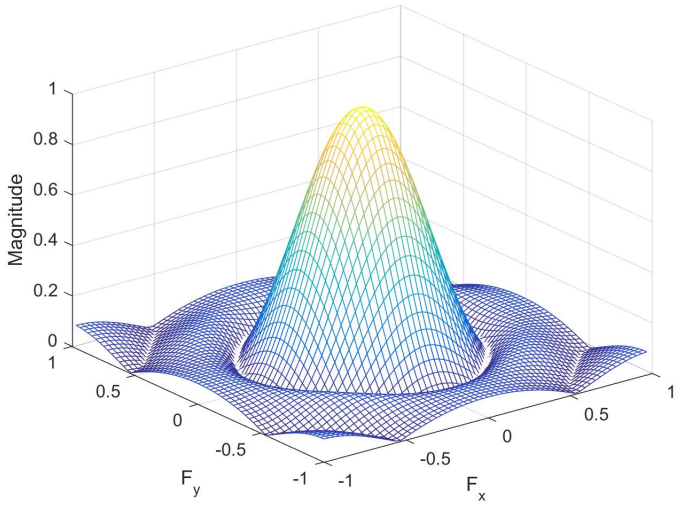

Figure 30: Filtro PillBox en Frecuencia

Si el lente de una cámara no está enfocado correctamente, cada punto en la imagen estará proyectado en un punto circular en el sensor de la imagen. El filtro pillbox es, por lo tanto, la función de un lente fuera de foco.

El filtro pillbox es un filtro que es simétrico pero no es linealmente separable, por lo que su implementación será necesariamente $O(n^2)$.

Vemos los efectos de aplicar un filtro pillbox de radio 5 a la imagen original:

Figure 31: Filtro PillBox en Frecuencia

4 spect.m

Se aprende a calcular el y a visualizarlo.

Se observa la imagen original a la cual se le calculará el espectro:

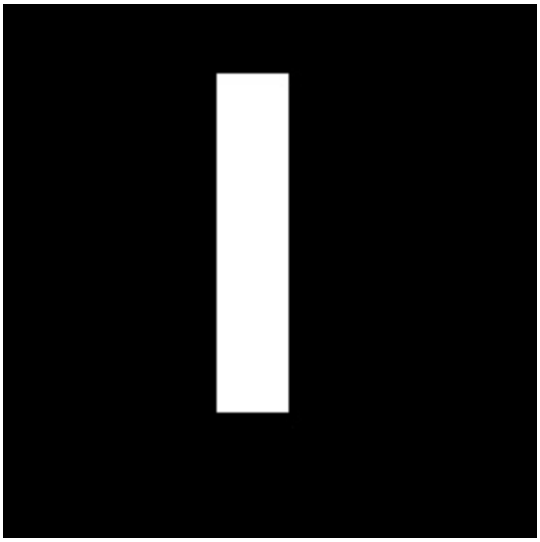

Figure 32: Imagen a la cual se le calculará el espectro

La imagen original es un pulso en x y un pulso en y. Es de notar que el pulso es más fino en la coordenada en x y más ancho en la coordenada y. Esta diferencia de duraciones debería poder notarse en los espectros calculados próximamente

Se observa el espectro de dicha imagen y su espectro en escala logarítmica:

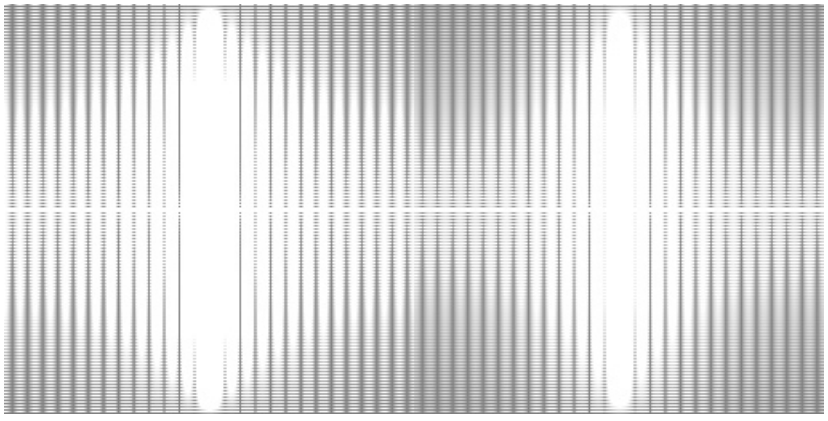

Figure 33: Espectro de la imagen(izquierda) y su versión en escala logarítmica

5 upsampling.m

Se busca realizar el upsampling de una imagen

El algoritmo a realizar, entonces, es el siguiente:

1. Se construye una imagen nula con las dimensiones que tendría la imagen upsampleada
2. Se completa a la nueva imagen con los valores de la imagen original rellenada con ceros entre los valores, haciendo efectivamente un zero padding de los valores originales (pero no en los bordes de la imagen, sino entre los valores)
3. Se realiza una fft 2D de la nueva imagen.
4. Luego, se realiza la ifft de la nueva imagen y se la filtra con un pasabajos, lo que es equivalente a interpolar entre los valores. El pasabajos es ideal, por lo que efectivamente se estaría usando una interpolación de tipo sin(x) / x entre los valores.

En la práctica, para lograr este pasabajos de tipo ideal y ser eficiente en tiempo, en vez de aplicar un pasabajos luego de realizar la ifft, se anulan los valores de frecuencias altas en el paso 3 luego de hacer la fft y después se hace la ifft sobre lo obtenido.

A continuación, se muestran los resultados obtenidos:

Figure 34: Imagen original

Figure 35: Imagen upsampleada por un factor de 2