Práctica 1
Visión por Computador

Filtrado y Detección de Regiones

Paula Iglesias Ahualli
Octubre de 2019

Paula Iglesias Ahualli
Octubre de 2019

CONTENIDO

Paula Iglesias Ahualli
Octubre de 2019

PRÁCTICA 1 VISIÓN POR COMPUTADOR

1. EJERCICIO 1

1.1. APARTADO A

El cálculo de la convolución de una imagen con una máscara 2D.

Usar una Gaussiana 2D *(GaussianBlur*) y máscaras 1D dadas por *getDerivKernels*).

Mostrar ejemplos con distintos tamaños de máscara, valores de sigma y condiciones de contorno. Valorar los resultados.

CONVOLUCIÓN

QUÉ SE CALCULA

Según [Forsyth, 137] una convolución es el proceso de aplicar un filtro a una imagen, dado un kernel H.

$$R_{i,j} = \sum_{u,v} H_{i-u,j-v} F_{u,v}$$

CÓMO SE CALCULA

La convolución la he implementado en la función:

def convolucion(imagen, kernelx, kernely, borde = cv2.BORDER_DEFAULT, normalize=True)

que a una imagen le aplica la convolución de un kernelx y un kernel y por filas y columnas, respectivamente.

CONVOLUCIÓN GAUSSIANA

QUÉ SE CALCULA

El filtro más conocido de alisamiento es el gaussiano, siendo el kernel de la siguiente forma [Szeliski, 104]:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

CÓMO SE CALCULA

Lo calculamos en la función:

def gaussiana(imagen, sigma, borde=cv2.BORDER_DEFAULT):

que internamente llama a la función de OpenCV

cv2.getGaussianKernel(tam, sigma)

que no hace sino calcular el kernel gaussiano.

Como parámetro necesita el tam (tamaño) que se calcula dependiendo de sigma y teniendo en cuenta que queremos que el mayor peso lo tenga el píxel central, por tanto:

$$tam = 6 * \sigma + 1$$

Ya que tenemos el kernel, que será el mismo para las filas y las columnas, sólo nos queda realizar la convolución con él.

RESULTADOS

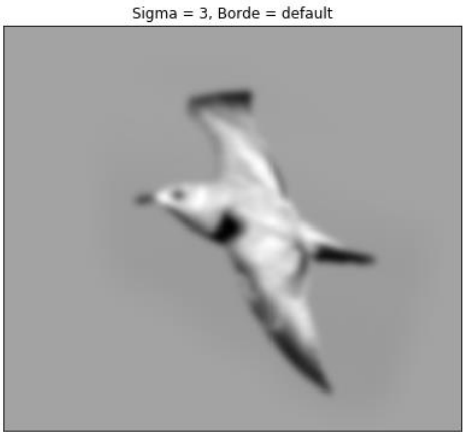
He aplicado la gaussiana con dos parámetros distintos:

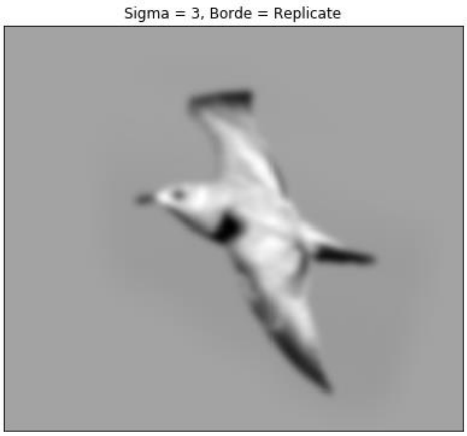
- Sigma
  - Sigma = 3 (arriba)
  - Sigma = 5 (abajo)
- Borde
  - Borde = default (izquierda)
  - Borde = Replicate (derecha)

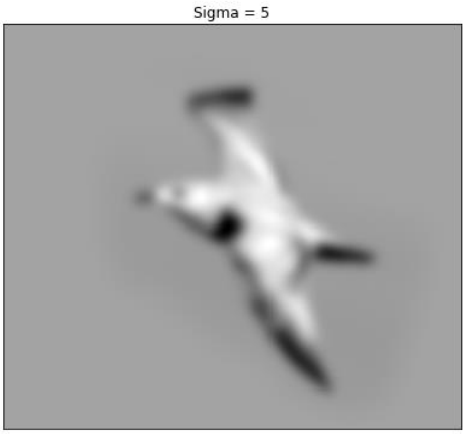
Con la variación de sigma se puede comprobar como el difuminado es mayor cuanto mayor es sigma, pues los píxeles vecinos tienen mayor peso en la suma ponderada (se tienen "más en cuenta").

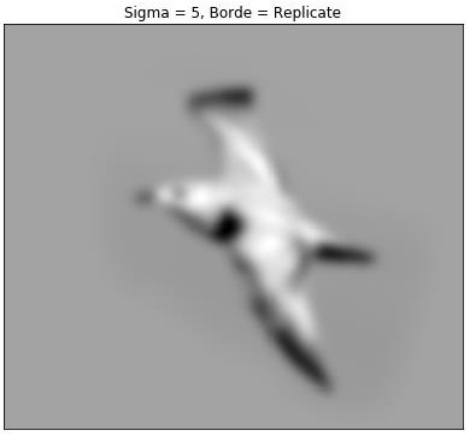
El borde replicate repite los píxeles de la esquinas indefinidamente [Sezeliski, 101] por tanto es difícil de apreciar la diferencia con el borde por defecto.

Paula Iglesias Ahualli
Octubre de 2019









DERIVADAS

QUÉ SE CALCULA

Las derivadas de las imágenes.

CÓMO SE CALCULA

Utilizamos la función

def derivada(imagen, tam, dx, dy, borde=cv2.BORDER_DEFAULT):

que realiza la convolución con los kernels obtenidos de

cv2.getDerivKernels(dx,dy,tam, borde)

que devuelve para los órdenes de derivadas dx y dy los respectivos kernel.

RESULTADOS

Aplicando los siguientes parámetros

- Sigma
  - Sigma = 3 (arriba)
  - Sigma = 5 (abajo)
- Borde
  - Borde = default (izquierda)

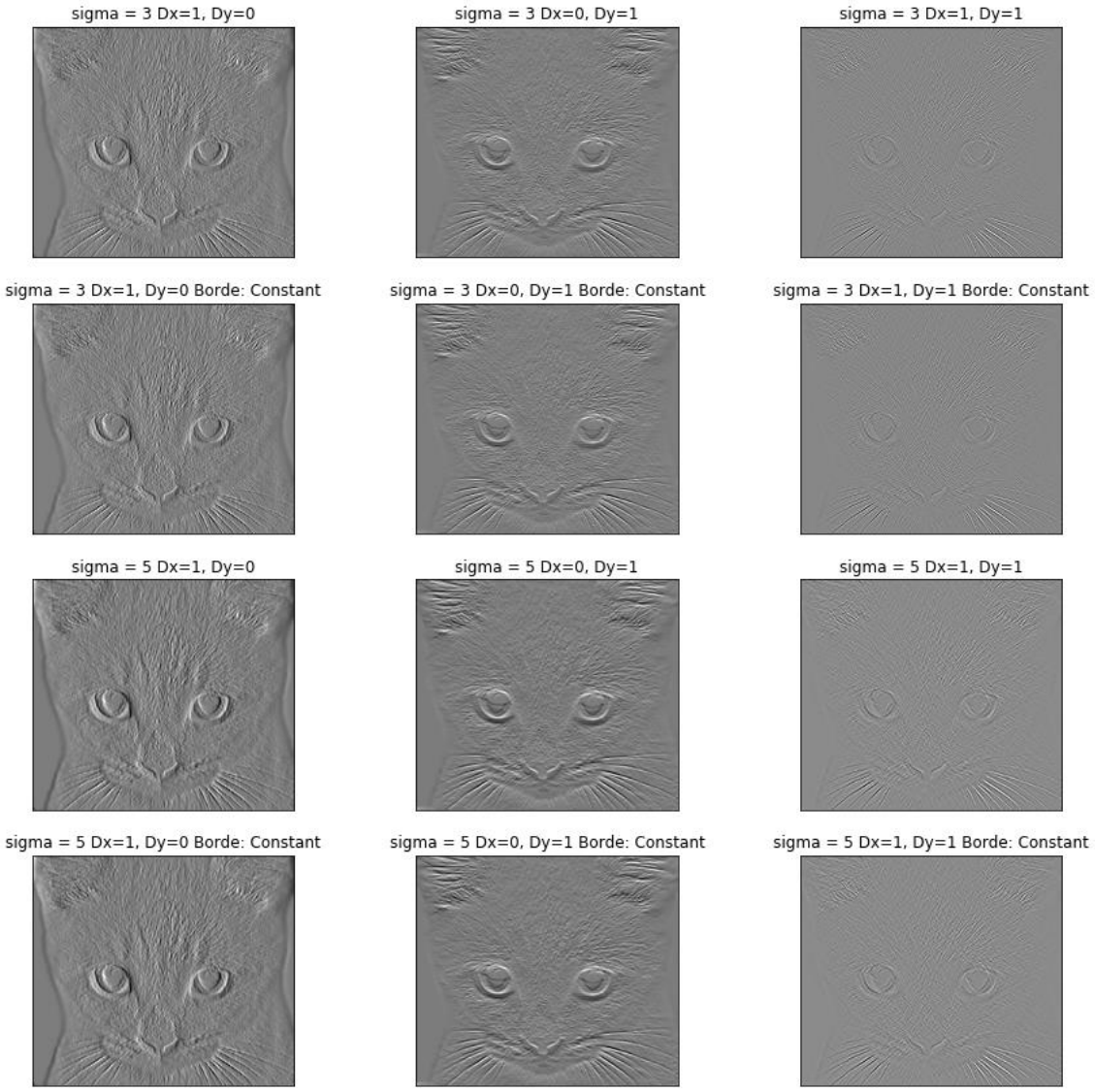
- Borde = constant (derecha)

A mayor sigma, los bordes que se detectan se ven más definidos.

Al hacer la primera derivada en x, podemos observar que ofrece muy buena localización de los contrastes verticales (los pelos en vertical, por ejemplo), mientras que con la primera derivada en y se aprecian mucho mejor los horizontales y los verticales pierden relevancia.

Si aplicamos la primera derivada tanto en x como en y podemos ver que se notan mucho los diagonales, como los bigotes.

El borde que he utilizado es *constant*, que a todos los píxeles exteriores de la imagen aplica un valor concreto. Sin embargo, resulta difícil de apreciar su diferencia con el borde por defecto.



Paula Iglesias Ahualli
Octubre de 2019

1.2. APARTADO B

> Usar la función Laplacian para el cálculo de la convolución 2D con una máscara normalizada de Laplaciana-de-Gaussiana de tamaño variable. Mostrar ejemplos de funcionamiento usando dos tipos de bordes y dos valores de sigma: 1 y 3.

Para aplicar la Laplaciana de Gaussiana, primero veamos cómo se calcula la Laplaciana

LAPLACIANA

QUÉ SE CALCULA

El operador laplaciano queda definido como la segunda derivada de la imagen [Szeliski, 104]:

$$\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 y}{\delta y^2}$$

CÓMO SE CALCULA

Aprovechamos la función *derivada* que hemos definido en el apartado anterior para crear la función

laplaciana(imagen, tam, borde=cv2.BORDER_DEFAULT):

Simplemente calculamos la suma de las segundas derivadas, como se indica en la fórmula.

LAPLACIANA DE GAUSSIANA

QUÉ SE CALCULA

Laplaciana de Gaussiana es equivalente a primero aplicar el filtro de la gaussiana y seguidamente el de laplaciana a una imagen.

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} + \frac{2}{\sigma^2} \right) G(x, y; \sigma)$$

CÓMO SE CALCULA

Definimos la función

laplacianaGaussiana(imagen, tam, sigma, borde=cv2.BORDER_DEFAULT):

que primero aplica la función *laplaciana*, y seguidamente *gaussiana*.

RESULTADOS

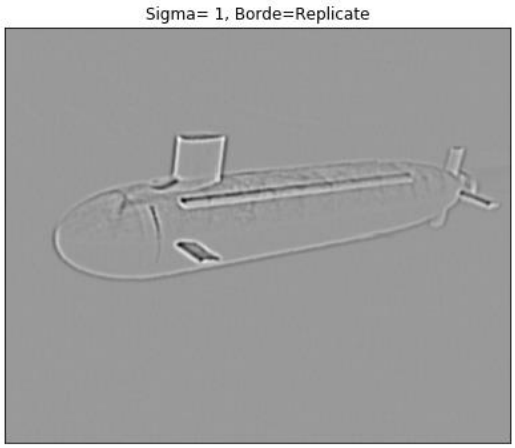
Aplicando los siguientes parámetros

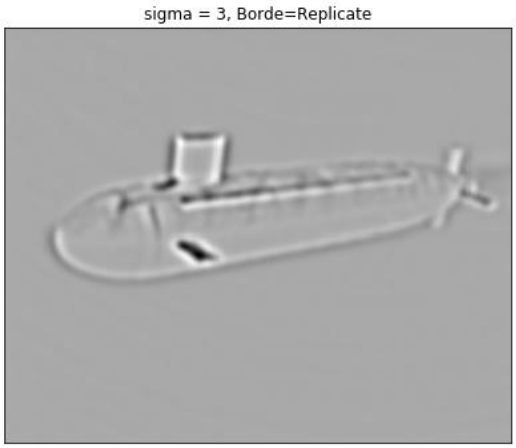
- Sigma
  - Sigma = 1 (izquierda)
  - Sigma = 3 (derecha)
- Borde
  - Borde = replicate (arriba)

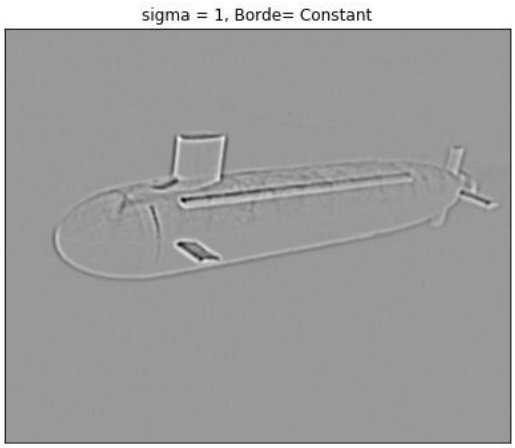
- Borde = constant (abajo)

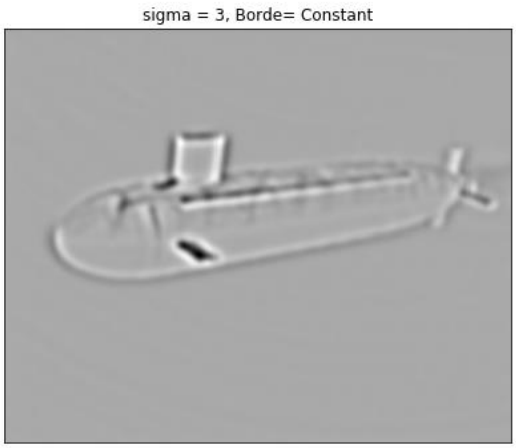
A mayor sigma, como pasaba en *gaussiana* el difuminado es mayor.

Ya he explicado anteriormente la diferencia entre los bordes replicate y constant, y aunque en estas imágenes no se aprecia mucho, podemos intuir que un borde constante rompe la imagen, mientras que el efecto replicate pasa más desapercibido y por tanto es más deseable éste último.









Paula Iglesias Ahualli
Octubre de 2019

2. EJERCICIO 2

2.1. APARTADO B

Una función que genere una representación en pirámide Gaussiana de 4 niveles de una imagen. Mostrar ejemplos de funcionamiento usando bordes y justificar la elección de los parámetros.

QUÉ SE HACE

Una pirámide de imágenes es una colección de representaciones de una imagen. Su particularidad es que cada nivel tiene la mitad de ancho y la mitad de alto que la anterior. En una pirámide gaussiana en cada nivel se aplica la convolución con un kernel gaussiano y se reescala al siguiente. El último nivel, por tanto, tendrá un difuminado muy fuerte [Forsyth, 165].

CÓMO SE HACE

Definimos la función

piramideGauss(imagen, niveles, sigma=1, borde=cv2.BORDER_DEFAULT):

que para cada nivel aplica la gaussiana y seguidamente submuestrea la imagen (reduciendo a la mitad tanto a filas como columnas).

RESULTADOS

Parámetros:

- Borde
  - Borde = default (arriba)
  - Borde = replicate (medio)
  - Borde = constant (abajo)

Si mostráramos las imágenes con el mismo tamaño podríamos ver como a medida que avanzan los niveles tienen menos píxeles, por lo que éstos se verían más grandes.

En la pirámide podemos apreciar como por ejemplo los pelos del bigote del perro apenas son distinguibles a partir del segundo nivel, debido al filtro gaussiano.

Paula Iglesias Ahualli
Octubre de 2019

Sin bordes

Borde = Replicate

Borde = Reflect

Paula Iglesias Ahualli
Octubre de 2019

2.2. APARTADO B

Una función que genere una representación en pirámide Laplaciana de 4 niveles de una imagen. Mostrar ejemplos de funcionamiento usando bordes.

QUÉ SE HACE

Se basa en aplicar recursivamente un filtro paso bajo y una reducción de tamaño a la imagen para obtener una serie de imágenes cada vez con menos detalle y menor dimensión a partir de dichas operaciones (coeficientes de aproximación o pirámide gaussiana) y guardar la diferencia entre esas imágenes y las originales (pirámide Laplaciana) [Burt &Kolczynski, 1993] Burt, P. J., & Kolczynski, R. J. (1993). Enhanced Image Capture Through Fusion.

CÓMO SE HACE

Defino la función:

piramideLaplaciana(imagen, niveles, borde= cv2.BORDER_DEFAULT):

En primer lugar se construye una pirámide gaussiana usando la función del apartado anterior. Para cada nivel i restamos al nivel i de la gaussiana el nivel i+1 sobremuestreado. Para hacer esto ultimo precisamos de la función

sobremuestrearDuplicar(imagen)

que lleva el nombre "duplicar" porque conseguimos sobremuestrear a base de duplicar las filas y columnas de la imagen (también podríamos haberlo hecho añadiendo filas y columnas pares de ceros).

RESULTADOS

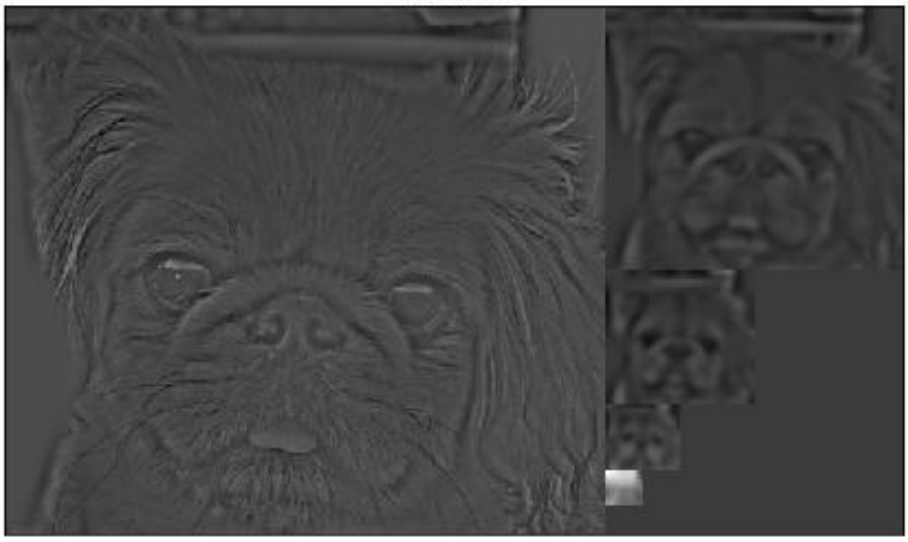
Parámetros:

- Borde
  - Borde = default (arriba)
  - Borde = replicate (medio)
  - Borde = constant (abajo)

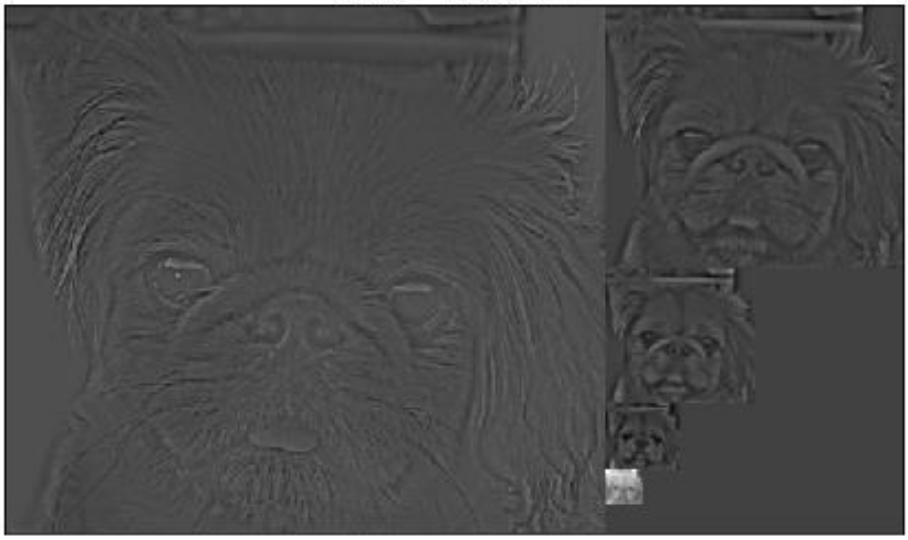
Podemos ver en cada nivel cómo se pierde definición, pero lo importante de esta pirámide es que si duplicáramos el tamaño de la imagen original y sumáramos la del nivel i+1 podríamos obtener la imagen en esa escala sin pérdida de información.

Paula Iglesias Ahualli
Octubre de 2019

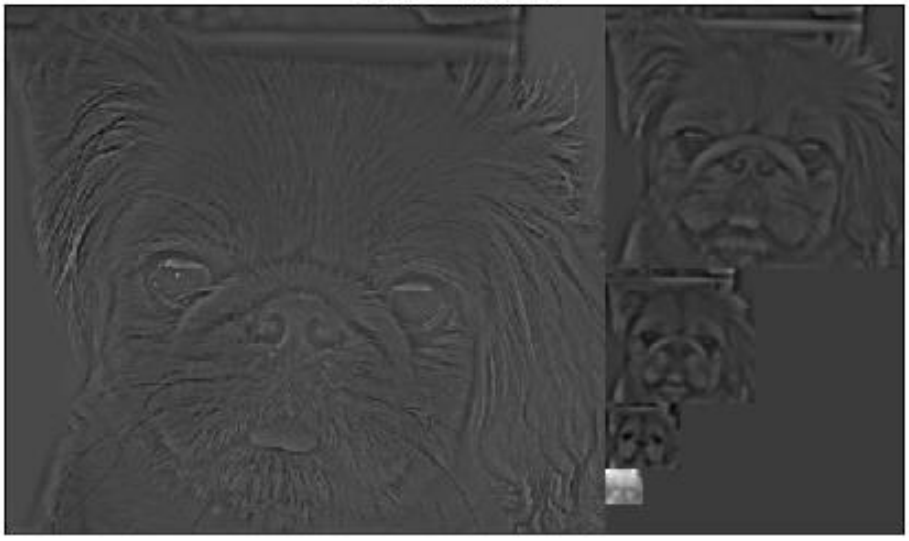
Si bordes



Borde = Replicate



Borde = Reflect



Paula Iglesias Ahualli
Octubre de 2019

3. EJERCICIO 3

Mezclando adecuadamente una parte de las frecuencias altas de una imagen con una parte de las frecuencias bajas de otra imagen, obtenemos una imagen híbrida que admite distintas interpretaciones a distintas distancias (ver hybrid images project page).

Para seleccionar la parte de frecuencias altas y bajas que nos quedamos de cada una de las imágenes usaremos el parámetro sigma del núcleo/máscara de alisamiento gaussiano que usaremos. A mayor valor de sigma mayor eliminación de altas frecuencias en la imagen convolucionada. Para una buena implementación elegir dicho valor de forma separada para cada una de las dos imágenes (ver las recomendaciones dadas en el paper de Oliva et al.). Recordar que las máscaras 1D siempre deben tener de longitud un número impar.

Implementar una función que genere las imágenes de baja y alta frecuencia a partir de las parejas de imágenes (solo en la versión de imágenes de gris) . El valor de sigma más adecuado para cada pareja habrá que encontrarlo por experimentación

1. Escribir una función que muestre las tres imágenes (alta, baja e híbrida) en una misma ventana. (Recordar que las imágenes después de una convolución contienen número flotantes que pueden ser positivos y negativos)

2. Realizar la composición con al menos 3 de las parejas de imágenes

3. Construir pirámides gaussianas de al menos 4 niveles con las imágenes resultado. Explicar el efecto que se observa.

QUÉ SE HACE

Una imagen híbrida H se obtiene combinando dos imágenes I1, I2.

I1 se filtra con un filtro de paso bajo (G1) e I2 con un filtro de paso alto(G2).

H se obtiene por tanto:

$$H = I1 * G1 + I2 * (1 - G2)$$

Elegiremos a la imagen con filtro de paso bajo como aquella que queremos que se vea desde la distancia, y a la de filtro de paso alto a la que se ve de cerca.

CÓMO SE HACE

Definimos la función

imagenesHibridas(imagen1, imagen2, lFreq, hFreq)

aplicando exactamente los pasos que se describen arriba, obteniendo I1, I2 y H con la suma de ambos.

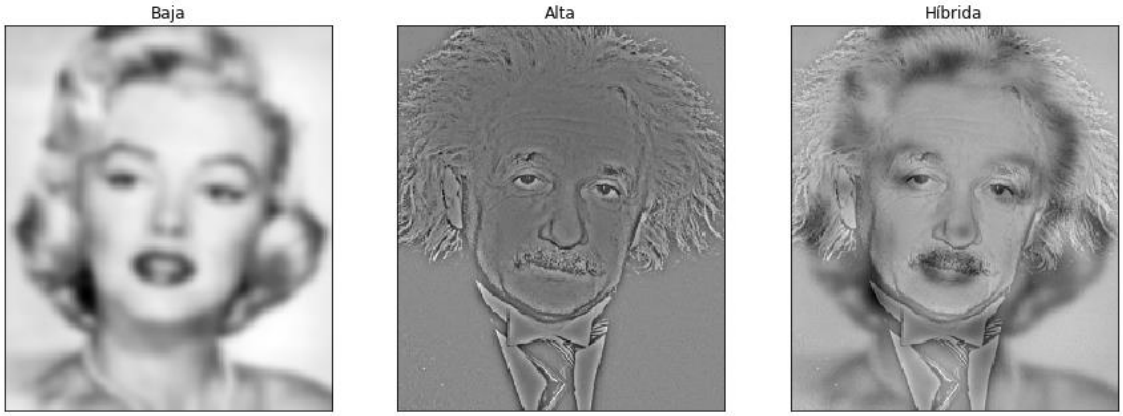
RESULTADOS

- Einstein y Marilyn

Siendo el pelo de Einstein muy característico he decidido utilizar su imagen como filtro de paso alto, mientras que Marilyn de paso bajo.

Experimentando he decidido los sigmas:

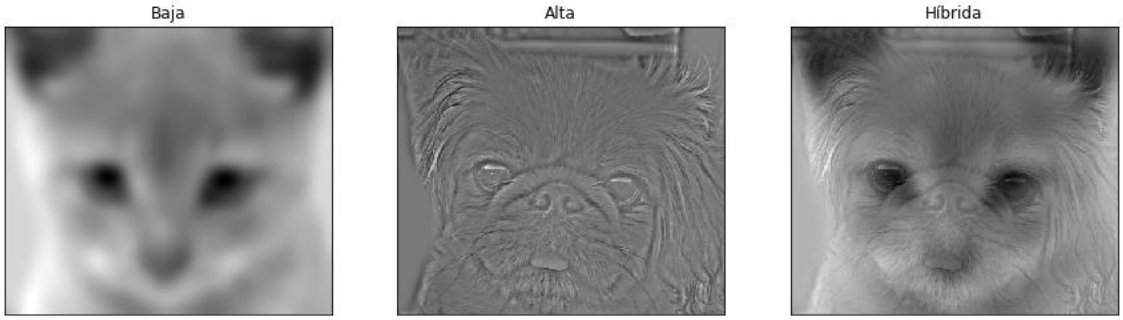
- Sigma paso bajo: 3
- Sigma paso alto: 3



- Perro y gato
  Tanto el pelaje del gato como del perro eran característicos por lo que no importaba realmente qué imagen se escogiera para qué filtro.

  Experimentando he decidido los sigmas:

  - Sigma paso bajo: 9
  - Sigma paso alto: 5



- Pez y submarino
  El contorno del pez es quizás más interesante que el del submarino, añadiendo esto al hecho de que el tamaño del submarino era mayor, he decidido usar el submarino con filtro de paso bajo y el pez con filtro de paso alto. Sin embargo la imagen no queda del todo bien porque los contornos de los demás peces también se detecta en el filtro de paso alto, y entorpece la imagen.

  Experimentando he elegido los sigmas:

  - Sigma paso bajo: 9
  - Sigma paso alto: 7

- Pájaro y avión
  He elegido el pájaro como imagen para filtro de paso bajo, ya que difuminada quedaba mejor que el avión, que se utiliza como filtro de paso alto.

  Experimentando he elegido los sigmas:

  - Sigma paso bajo: 5
  - Sigma paso alto: 3

PIRÁMIDES GAUSSIANAS

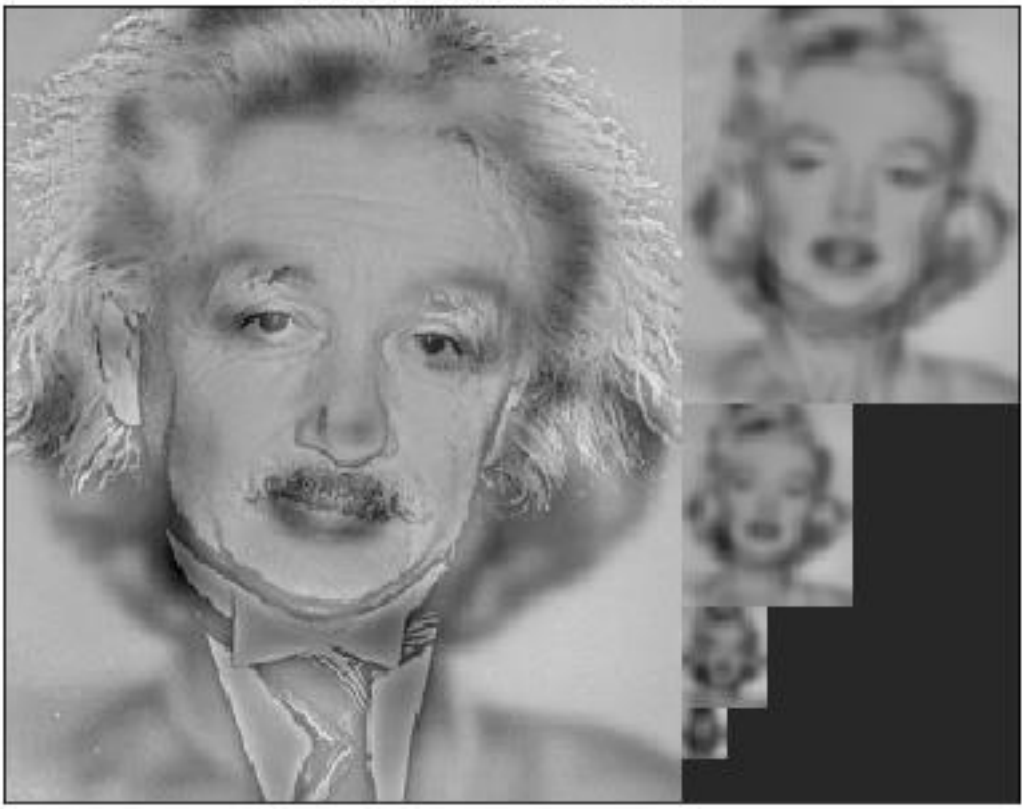
Para aplicar la pirámide gaussiana a una imagen híbrida nos valemos de la función recientemente implementada, imagenesHibridas, y la que se implementó anteriormente piramideGaussiana.

RESULTADOS

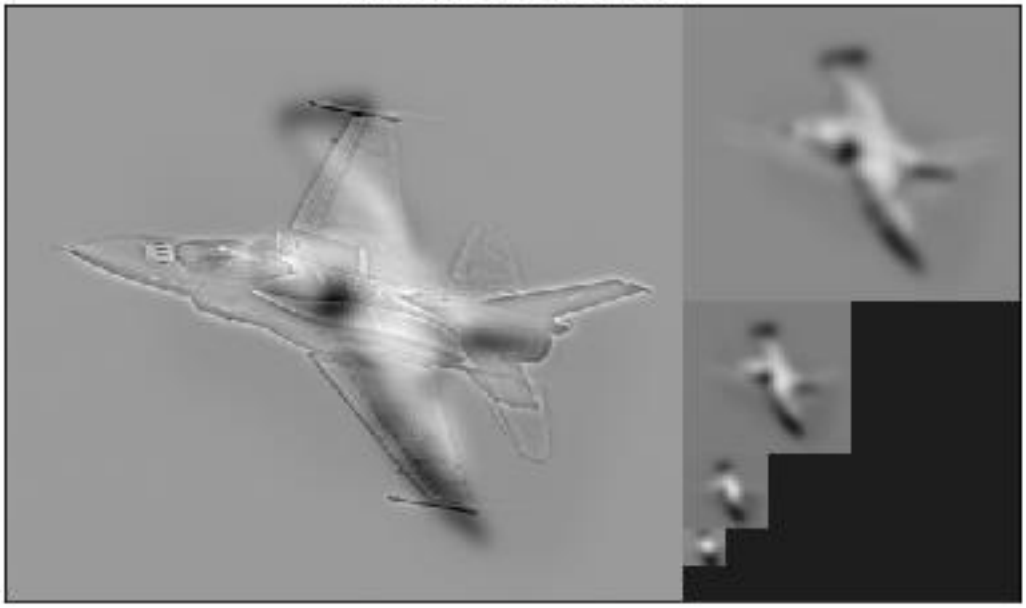
En todos los ejemplos podemos apreciar cómo al aplicar un filtro gaussiano (que es lo que hace *piramideGaussiana*) con sigma = sigma de paso bajo que usamos para hibridar, la imagen de paso alto desaparece, por lo que nos queda sólo la imagen de paso bajo original.

Paula Iglesias Ahualli
Octubre de 2019

Piramide Marilyn-Einstein



Piramide Pájaro-Avión



Paula Iglesias Ahualli
Octubre de 2019

Piramide Pez-Submarino

BIBLIOGRAFÍA

A. Oliva, A. Torralba, P.G. Schyns, in ***Proceeding ACM SIGGRAPH*** (2006), 527-532.

D.A. Forsyth, J. Ponce, ***Computer Vision: A modern Approach***, ISBN: 9780130851987, Ed. Pearson, New Jersey (2003).

R. Szeliski, ***Computer Vision: Algorithms and Applications***, ISBN: 978-1-84882-934-3, Ed. Springer, London (2010)