Visión por Computador

Práctica 1

Curso 2020-2021

Javier Gálvez Obispo
javiergalvez@correo.ugr.es

1. Ejercicio 1

Apartado A

Implementamos una función *gaussiana1D*(*x, sigma*) para el cálculo de la Gaussiana en el caso 1D y que nos devuelve el resultado de la siguiente función:

$$f(x) = e^{-\frac{x^2}{2\sigma^2}}$$

Derivamos *f(x)* para poder implementar la función que nos calcule la primera derivada de la Gaussiana:

$$\frac{\partial}{\partial x} f(x) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

Volvemos a derivar para obtener la segunda derivada de la Gaussiana

$$\frac{\partial}{\partial x^2} f(x) = \left(-\frac{1}{\sigma^2} + \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2}{2\sigma^2}}$$

Una vez tenemos implementadas éstas tres funciones, sólo necesitamos una función genérica para calcular máscaras, *generar_mascara*(*val, deriv, usar_sigma*), a la que podemos pasarle un valor de sigma o el tamaño deseado de la máscara e indicarle cuál de éstas funciones debe usar para generar dicha máscara.

Generamos una máscara para la Gaussiana 1D utilizando σ = 1 y la representamos

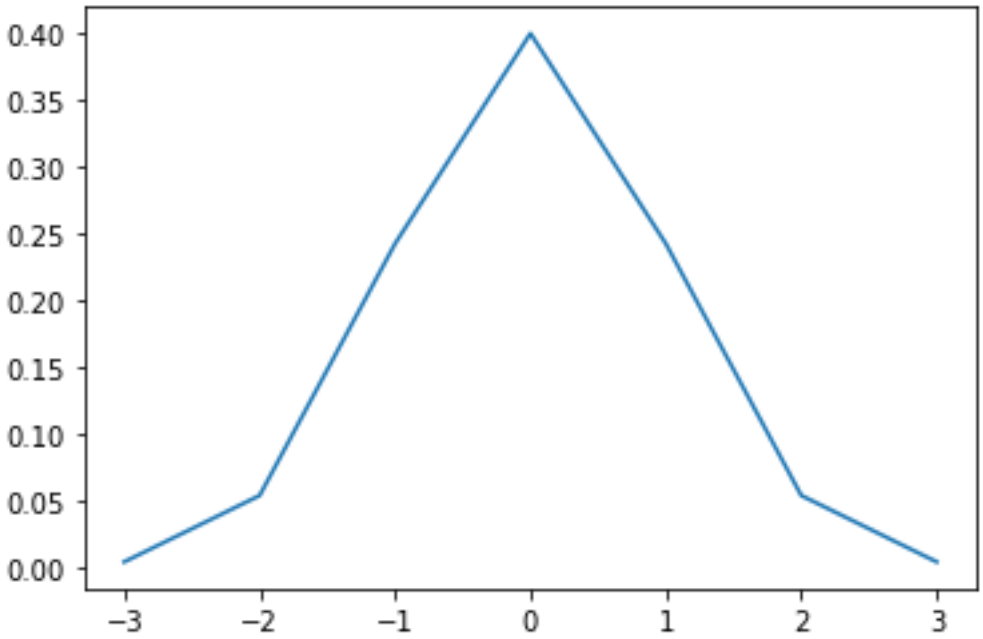


Figure 1: Representación de una máscara Gaussiana 1D con σ = 1

Hacemos lo mismo para la primera y la segunda derivadas

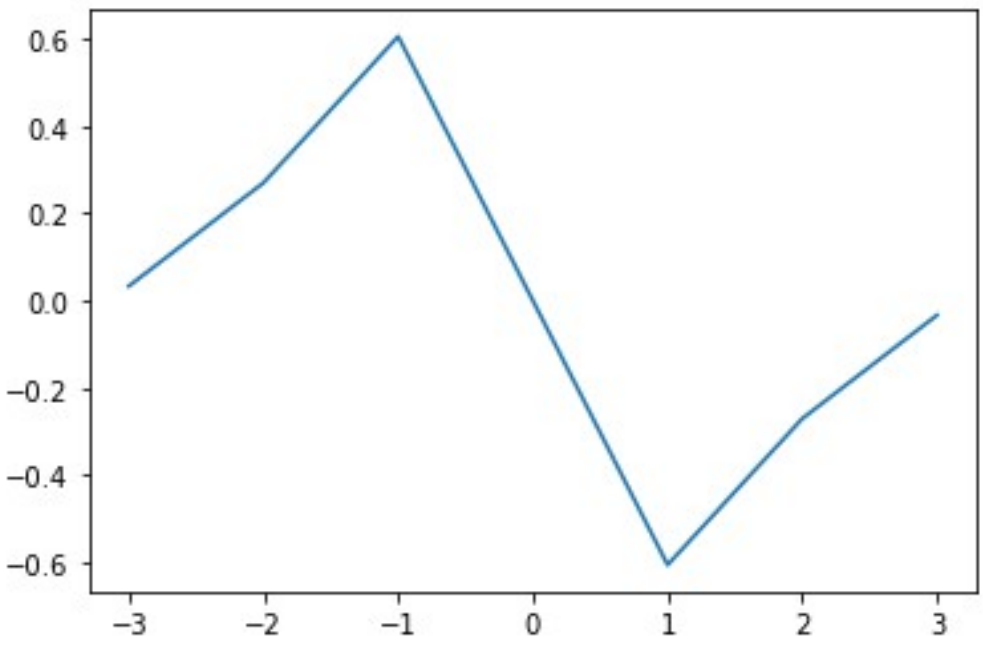


Figure 2: Representación de una máscara de la primera derivada de la Gaussiana 1D con $\sigma = 1$

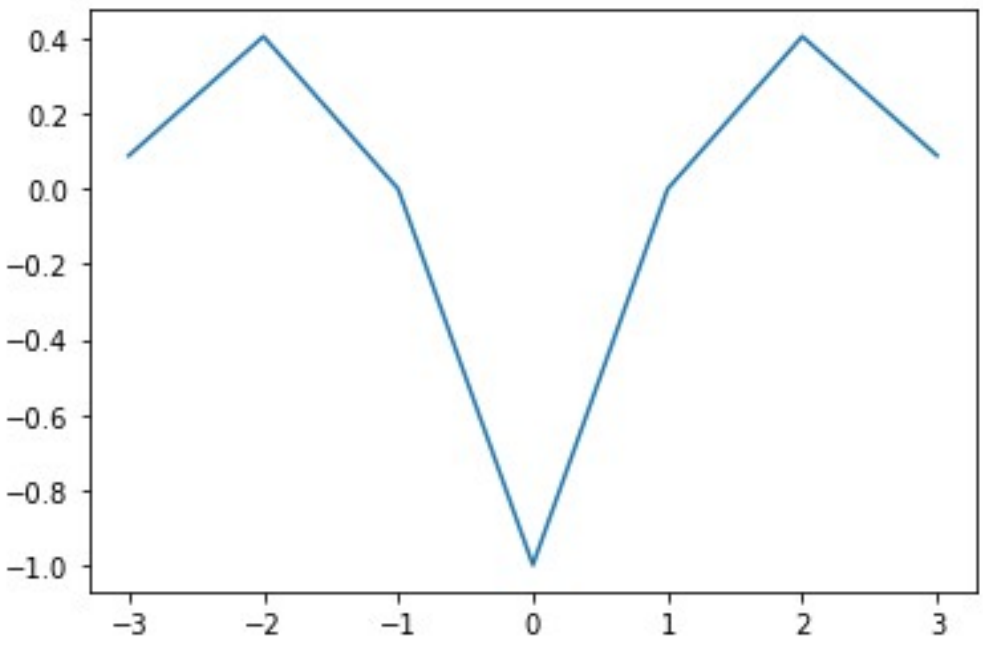


Figure 3: Representación de una máscara de la segunda derivada de la Gaussiana 1D con $\sigma = 1$

Apartado B

En éste apartado nos piden realizar la convolución de una imagen con una máscara Gaussiana 2D cuya función es:

$$f(x) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

A la hora de implementar la convolución, en vez de utilizar ésta función, podemos aprovechar la separabilidad de la Gaussiana 2D en el producto de dos funciones, una dependiente de *x* y la otra de *y,* es decir, la separamos en dos Gaussianas 1D.

$$f(x) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \; e^{-\frac{y^2}{2\sigma^2}}$$

Ésto implica que podemos pasar una máscara Gaussiana 1D por filas y a la imagen resultado le volvemos a pasar exactamente la misma máscara por columnas para realizar la convolución de una imagen con una máscara Gaussiana 2D.

Entonces, a la hora de implementar la convolución, tan solo tenemos que implementar la convolución de una imagen por filas dada una máscara y volvemos a pasar la misma máscara por la imagen transpuesta, o lo que es lo mismo, realizamos la convolución por columnas.

Antes de realizar la convolución, es necesario añadir bordes a la imagen original para que el resultado tenga la mismas dimensiones. Para ello, se ha implementado una función *generar_bordes*(*img, k, replicar*) que añade un borde, de ceros o replicados, con grosor *k* a la imagen dada.

Figure 4: Comparación de la imagen original (izquierda) con el resultado de la convolución con una máscara Gaussiana 2D usando σ = 1

Si comparamos el resultado con los obtenidos por la función *GaussianBlur* que nos ofrece **opencv** utilizando la misma máscara, no podemos notar ninguna diferencia a simple vista

Figure 5: Comparación del resultado de nuestra función (izquierda) con el resultado de GaussianBlur (izquierda) utilizando la misma máscara

Podemos comparar los valores que toman los píxeles para una misma región en las dos imágenes para ver las diferencias aunque no son muy destacables.

146.445	149.004	151.551	153.855	146	149	152	154
146.179	147.782	150.493	153.532	146	148	151	154
145.629	146.737	149.768	153.411	146	147	150	153
145.143	145.738	148.966	153.095	145	146	149	153

Table 1: Comparación de los píxeles para una misma región de los resultados de nuestra función (izquierda) y GaussianBlur (derecha)

Como podemos observar, los valores obtenidos por *GaussianBlur* son los mismos que obtenemos con nuestra propia función de convolución pero redondeados.

Apartado C

Opencv nos ofrece la función *getDerivKernels* para obtener las máscaras que utiliza para las derivadas de la Gaussiana. Si representamos las máscaras de tamaño 7 devueltas por ésta función obtenemos los siguientes gráficos:

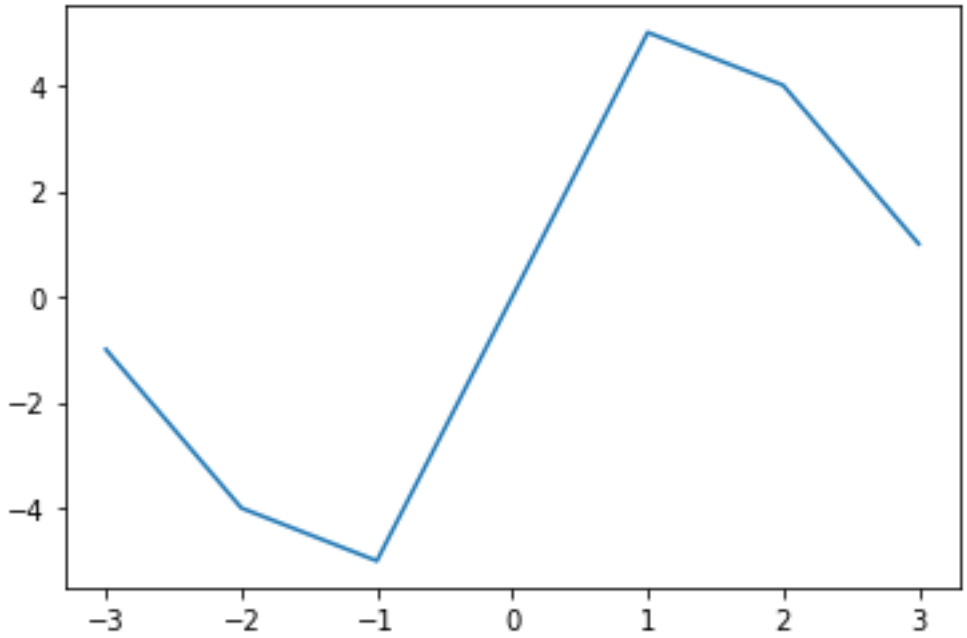


Figure 6: Máscara de tamaño 7 de la primera derivada obtenida con getDerivKernel

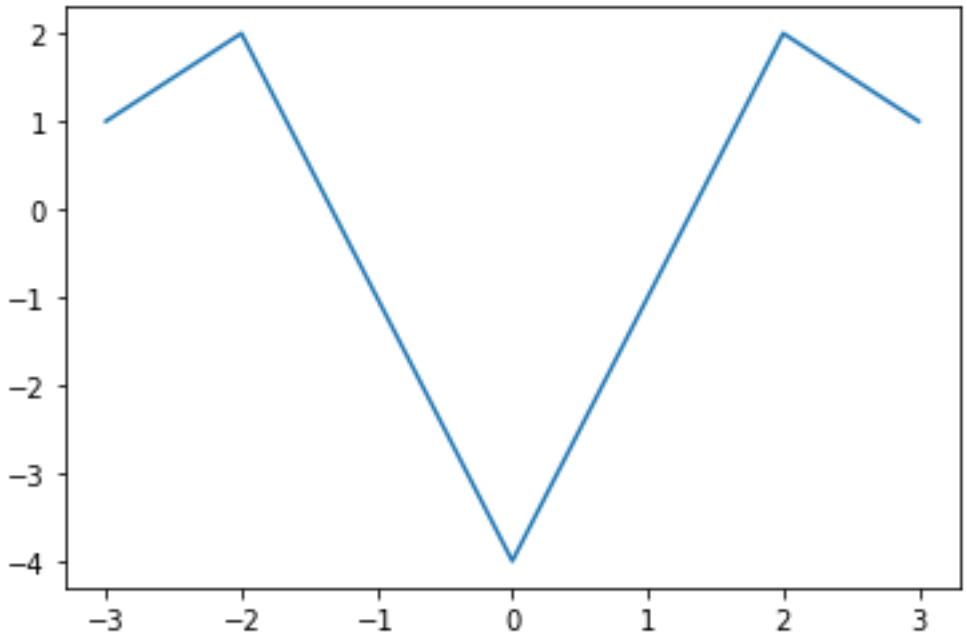


Figure 7: Máscara de tamaño 7 de la segunda derivada obtenida con getDerivKernel

Si las comparamos con las representaciones de las máscaras del apartado A, vemos que ambas tienen la misma estructura pero, las obtenidas por *getDerivKernels* toman valores mayores que las nuestras, lo cuál es mucho más destacable cuanto mayor es el tamaño de las máscaras.

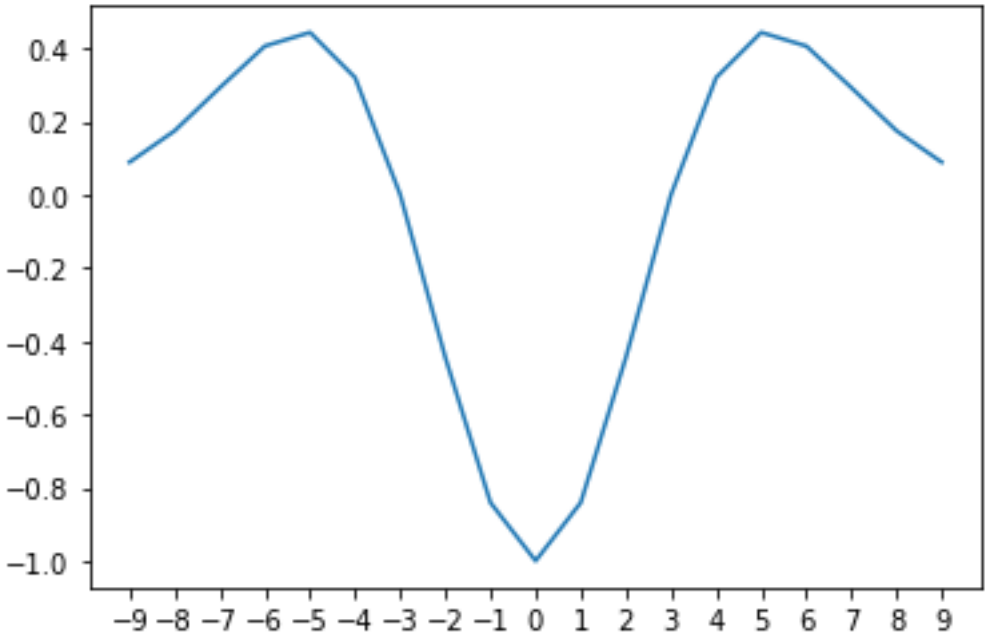


Figure 8: Máscara de tamaño 19 de la segunda derivada obtenida con getDerivKernel

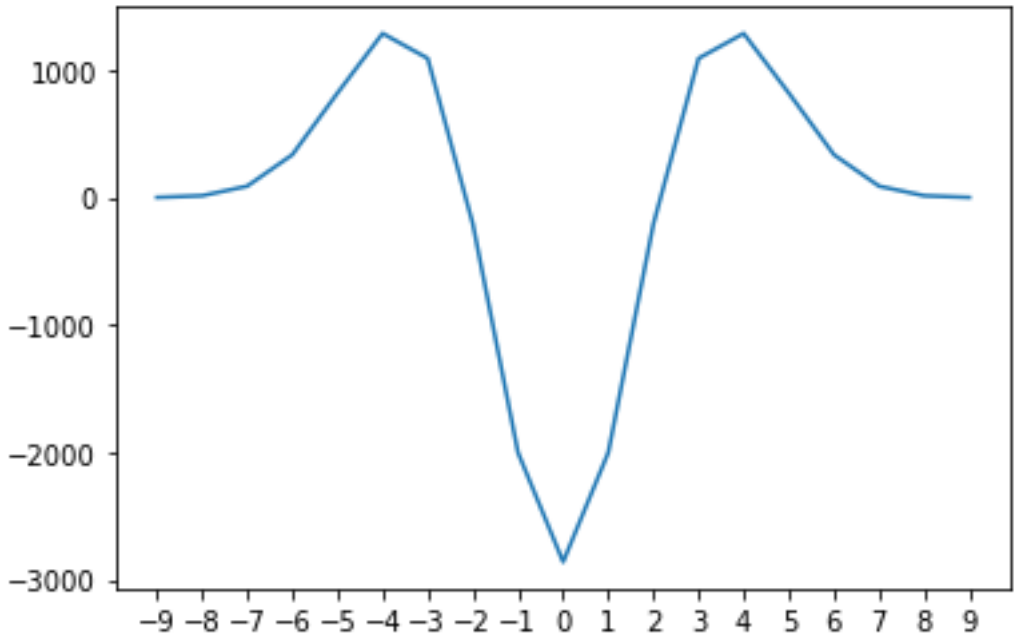


Figure 9: Máscara de tamaño 19 de la segunda derivada de la Gaussiana 1D

A pesar de la diferencia de valores, los resultados obtenidos por las máscaras de **opencv** y las nuestras van a ser similares, ya que ambas máscaras tienen la misma estructura.

Apartado D

Para poder calcular la Laplaciana de la Gaussiana tenemos que calcular las segundas derivadas parciales de la Gaussiana 2D. Como ya vimos en el apartado B la Gaussiana 2D se puede separar en el producto de dos Gaussiana 1D lo que simplifica la derivación de ésta. Calculamos la segunda derivada parcial respecto a *x*.

$$\frac{\partial}{\partial x^2} f(x) = \left(-\frac{1}{\sigma^2} + \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2}{2\sigma^2}} \quad e^{-\frac{y^2}{2\sigma^2}}$$

Si analizamos el resultado que hemos obtenido, podemos ver dos términos totalmente independientes como pasaba con la Gaussiana 2D. Ésto significa que podemos calcular la segunda derivada parcial aplicando dos máscaras 1D las cuales serían, la máscara de la segunda derivada de la Gaussiana 1D para las filas y la máscara de la Gaussiana 1D para las columnas.

Claramente, la segunda derivada parcial respecto de *y* es idéntica a la de *x* pero cambiando los roles de ambas variables. Entonces, para calcular la Laplaciana de la Gaussiana tan solo hay que pasar cuatro máscaras 1D, por un lado, pasamos la máscara de la segunda derivada por filas a la imagen original y después la de la Gaussiana 1D por columnas y, por otro lado, hacemos lo mismo cambiando filas por columnas. Por último sumamos ambos resultados para obtener la Laplaciana.

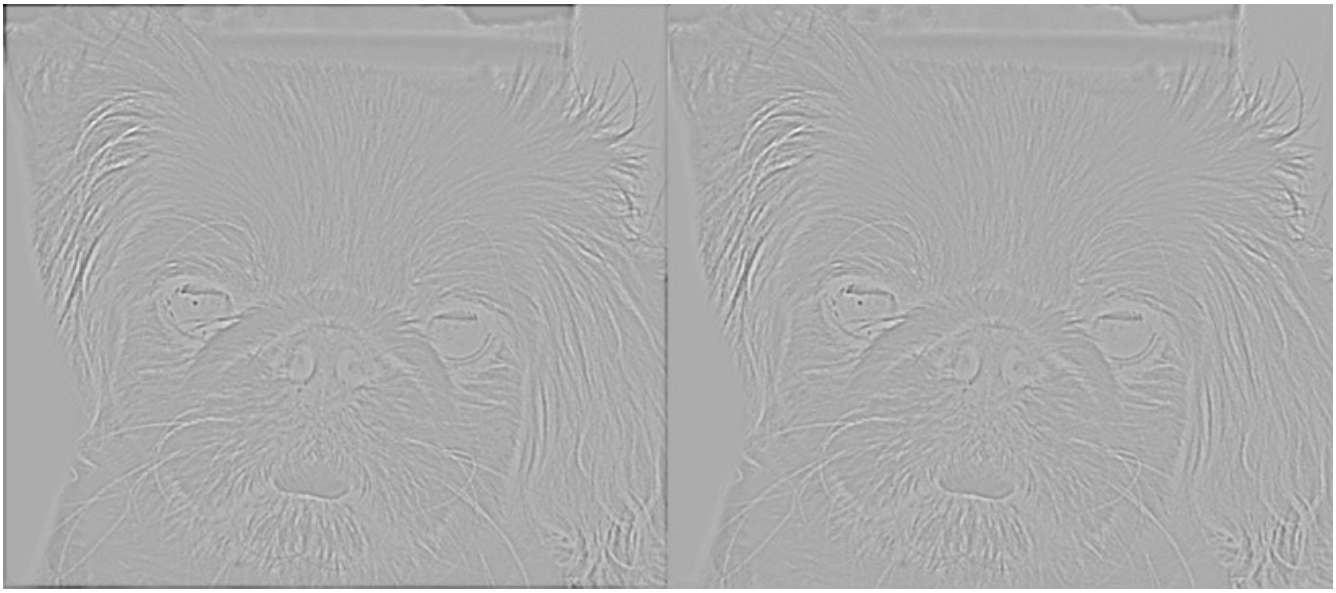


Figure 10: Comparativa de la Laplaciana con σ = 1 utilizando bordes de ceros (izquierda) y bordes replicados (derecha)

Hay una clara diferencia entre usar un tipo de bordes u otro, en el primero estamos introduciendo ruido cerca de los bordes ya que la imagen toma valores más cercanos al blanco en éstas regiones, lo que provoca un alto contraste. No ocurre lo mismo en el caso de utilizar bordes replicados.

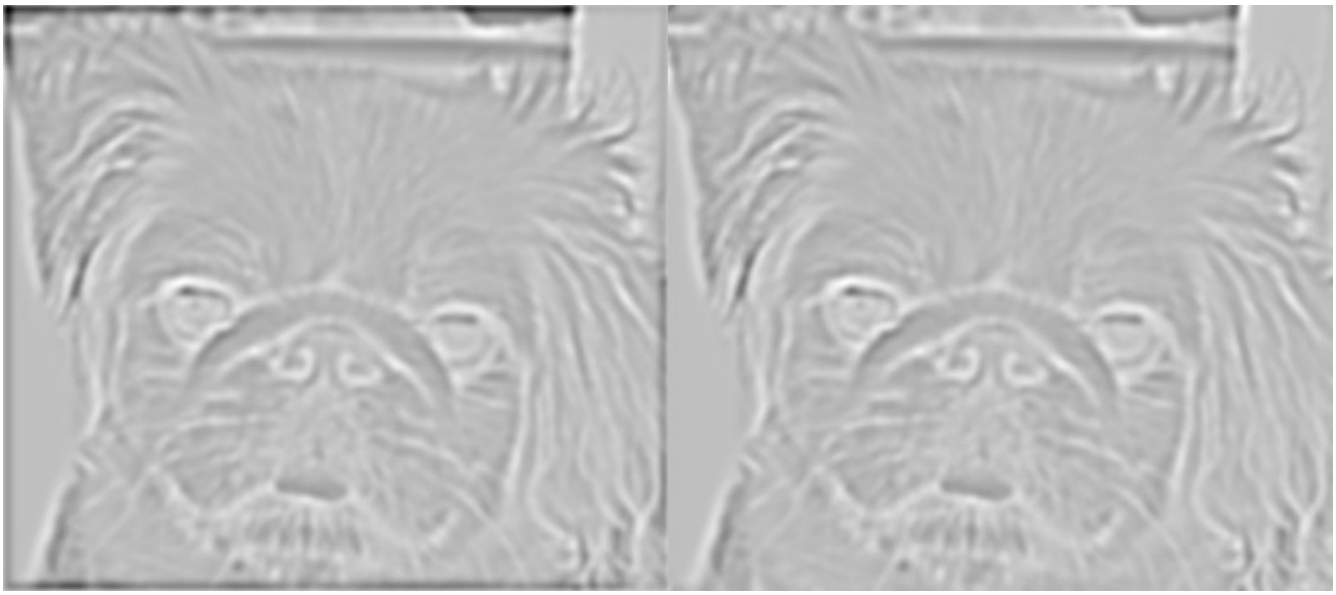


Figure 11: Comparativa de la Laplaciana con σ = 3 utilizando bordes de ceros (izquierda) y bordes replicados (derecha)

Si aumentamos el valor del σ utilizado la introducción de ruido cuando usamos bordes de ceros es mucho más notoria que antes. Está claro que a la hora de calcular la Laplaciana es mejor no usar éste tipo de bordes si se genera un alto contraste con la imagen original.

Ejercicio 2

Apartado A

Una vez hemos implementado todo el ejercicio 1, realizar ahora la pirámide Gaussiana de una imagen es muy sencillo. Nuestra función lo único que tiene que hacer es alisar la imagen con la que estamos trabajando calculando la convolución con la Gaussiana y después descartar la mitad de filas y columnas de la imagen alisada. Repetimos este proceso tantas veces como se indique y obtenemos la pirámide Gaussiana de una imagen.

Para que la pirámide que obtenemos tenga sentido, el valor de σ debe ser pequeño, ya que al reducir el número de filas y columnas de la imagen a la mitad en cada nivel, perdemos muchos detalles si, además, usamos un σ grande estaríamos eliminando aún más detalles de la imagen en cada alisado y al final tendríamos imágenes que no pueden ser interpretadas.

Figure 12: Pirámide Gaussiana de nivel 4 con σ = 1

Figure 13: Pirámide Gaussiana de nivel 4 con σ = 3

Apartado B

Para calcular la pirámide Laplaciana de una imagen primero debemos calcular su pirámide Gaussiana. Una vez la tenemos sólo hay que aplicar el algoritmo visto en clase, para calcular el nivel *i* de la pirámide Laplaciana le restamos al nivel *i* de la pirámide Gaussiana el nivel *i*+1 de la misma escalando su tamaño al del nivel *i*. Hay que destacar que el último nivel de ambas pirámides se corresponde con la misma imagen.

Figure 14: Pirámide Laplaciana de nivel 4

Para comprobar que hemos calcula la pirámide Laplaciana correctamente podemos reconstruir la imagen original a partir de ésta y comparar las dos imágenes.

Partiendo del último nivel de la pirámide Laplaciana, aumentamos la imagen al tamaño del nivel anterior y las sumamos. Escalamos la imagen obtenida al tamaño del nivel anterior y volvemos a sumar. Repetimos este proceso con todos los niveles y obtenemos una reconstrucción de la imagen original.

Figure 15: Comparación de la imagen original (izquierda) con su reconstrucción (derecha) a partir de una pirámide Laplaciana de nivel 4

Ejercicio 3

En éste apartado nos piden obtener una imagen híbrida que admita distintas interpretaciones dependiendo de la distancia a las que se observe. Para ello, se nos dan varías parejas de imágenes para las cuales debemos elegir de que imagen vamos a obtener las frecuencias bajas y cual utilizaremos para las frecuencias altas.

Para obtener las frecuencias bajas de una imagen pasamos una máscara de Gaussiana, cuánto mayor sea el σ utilizado más frecuencias altas eliminamos. En el caso de las frecuencias altas también vamos a convolucionar la imagen con una Gaussiana pero, el resultado obtenido se lo restamos a la imagen original eliminando así las frecuencias bajas de ésta y quedándonos sólo con las altas.

Para comprobar que hemos realizado bien la hibridación de las imágenes calculamos la pirámide Gaussiana de nivel 4 para cada imagen híbrida. Si la interpretación de la imagen cambia cuanto más avanzamos en los niveles de la pirámide entonces, habremos realizado correctamente la hibridación.

Mostramos a continuación las tres parejas que se han utilizado para generar imágenes híbridas junto con las pirámides Gaussiana de cada una de ellas. En todas las parejas se muestra a la izquierda la imagen utilizada para las frecuencias bajas y a la derecha la utilizada para las frecuencias altas.

Los sigmas se han obtenido probando distintas combinaciones de valores para cada uno hasta que se ha producido el efecto esperado en la pirámide Gaussiana de cada imagen híbrida.

Figure 16: Frecuencias bajas, perro, frecuencias altas, gato

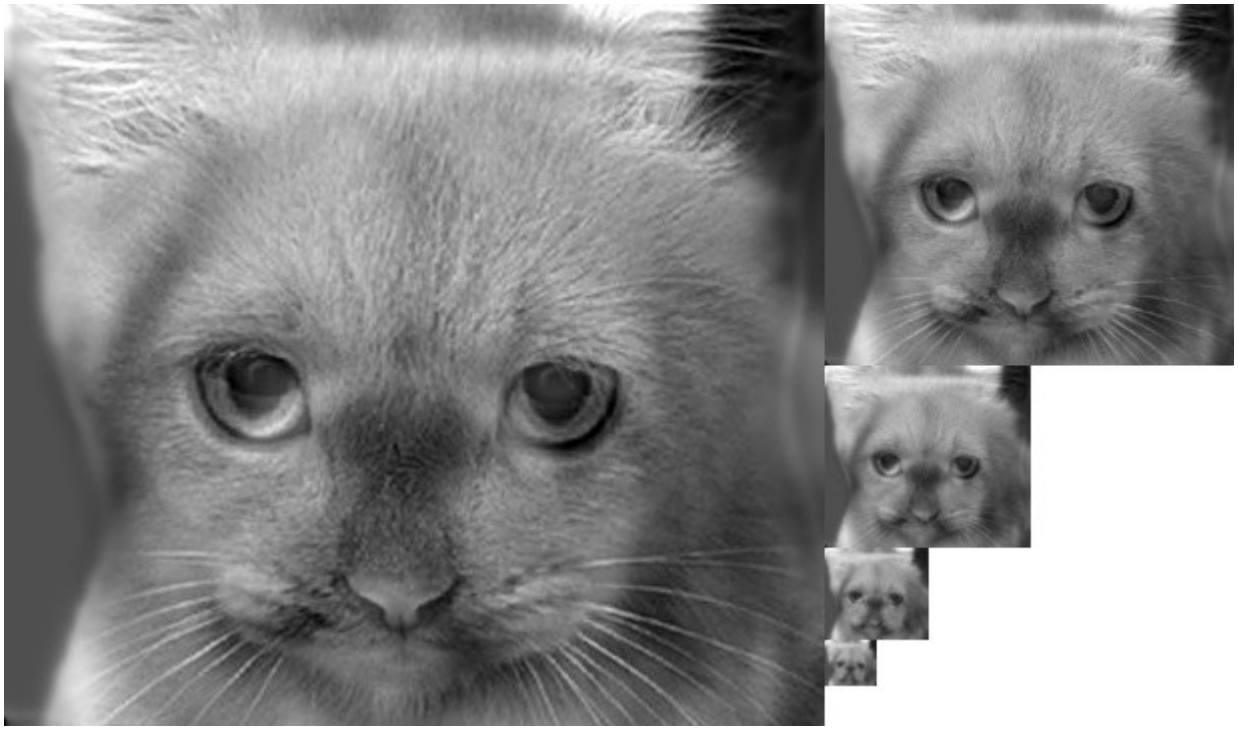


Figure 17: Pirámide Gaussiana de la imagen híbrida del perro (σ = 5) y el gato (σ = 5)

Podemos ver en los niveles de la pirámide como cambia la interpretación de la imagen. Primero vemos al gato pero cuando se hace más pequeña la imagen empezamos a ver el perro y en el último nivel ya no se observa ningún detalle del gato.

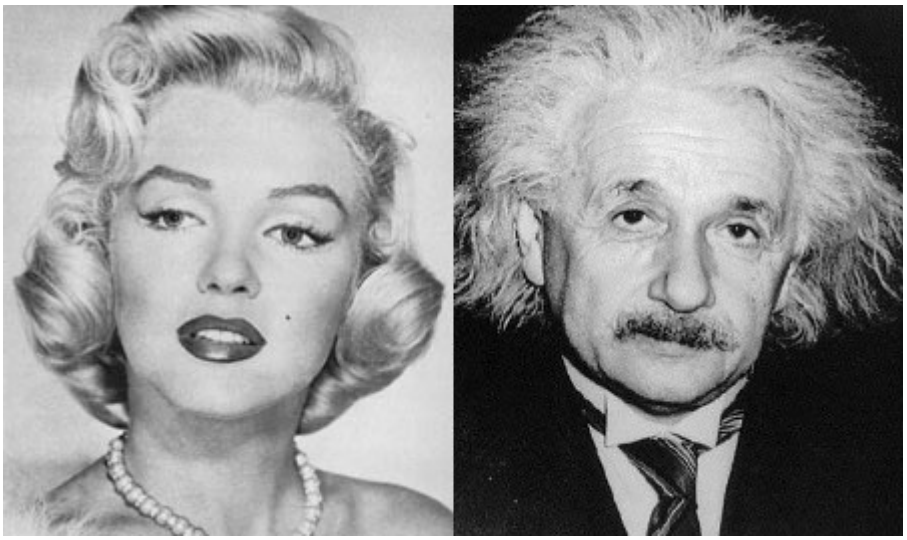


Figure 18: Frecuencias bajas, Marilyn, frecuencias altas, Einstein

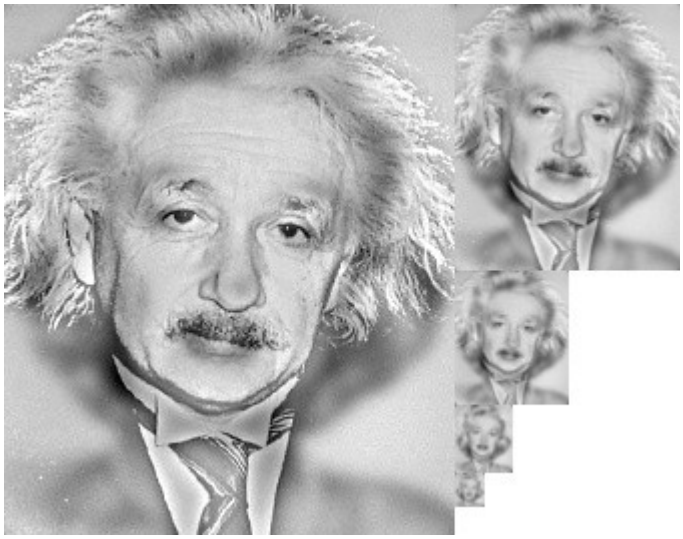


Figure 19: Pirámide Gaussiana de la imagen híbrida de Marilyn (σ = 4) y Einstein (σ = 4)

Aquí ocurre lo mismo que en la pareja anterior, vemos como la imagen cambia de la cara de Einstein en los primeros niveles a la cara de Marilyn en los últimos.

Figure 20: Frecuencias bajas, submarino, frecuencias altas, pez

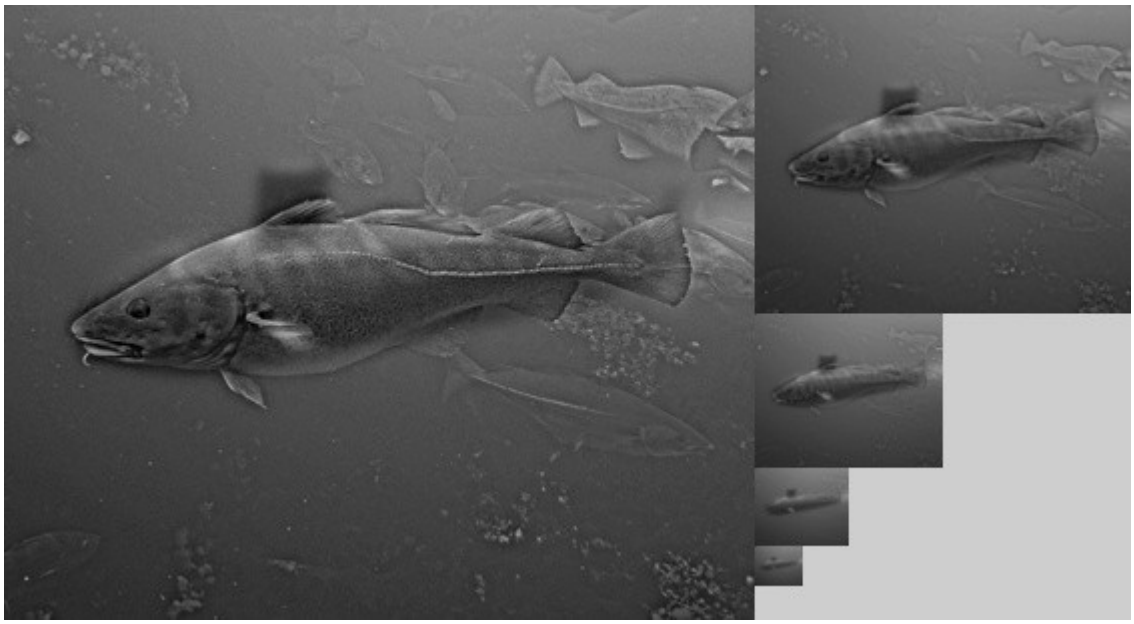


Figure 21: Pirámide Gaussiana de la imagen híbrida submarino (σ = 3) y pez (σ = 3)

Al igual que antes, la imagen híbrida se ha hecho correctamente. En los primeros niveles el submarino se ve como una sombra detrás del pez mientras que, para los últimos niveles, el pez deja de ser visible y sólo percibimos el submarino.

Bonus 1

Volvemos a realizar las imágenes híbridas para todas las parejas que nos han dado pero ahora en color, excepto las caras de Einstein y Marilyn que no están disponibles a color.

Para poder hacer éste apartado hay que modificar la función de convolución que tenemos implementada para que también acepte imágenes con 3 canales. Gracias a la potencia de **numpy** realizar ésta modificación no supone un gran problema. Además, la función que genera los bordes para una imagen también debe ser modificada para que añada bordes con 3 canales en el caso de que tenga color. También, ha sido necesario implementar una función que calcule la transpuesta de una imagen teniendo en cuenta los canales de la misma.

Volvemos a mostrar todas las parejas junto a las pirámides Gaussianas de la imágenes híbridas obtenidas.

Figure 22: Frecuencias bajas, perro, frecuencias altas, gato

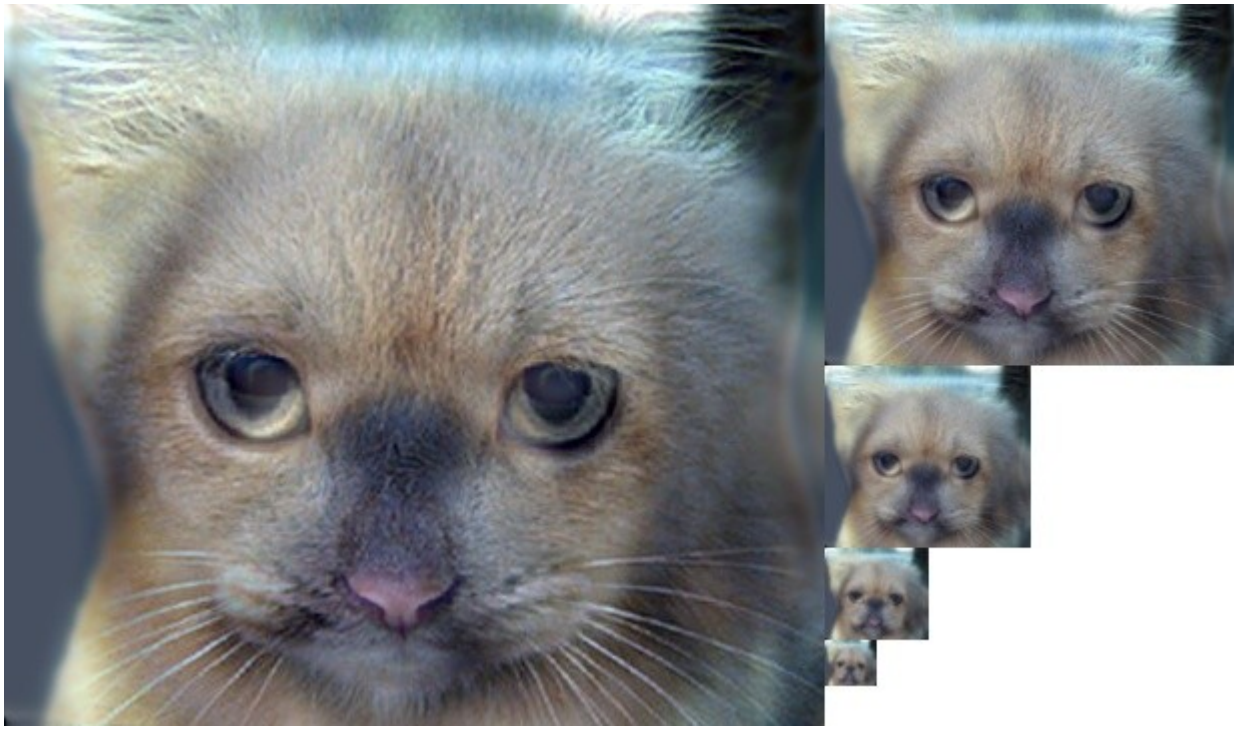


Figure 23: Pirámide Gaussiana de la imagen híbrida del perro (σ = 5) y el gato (σ = 5)

Obtenemos un resultado similar al caso sin color. En los primero niveles no se puede identificar al perro y, de hecho, los colores que introduce en la imagen parecen manchas del gato.

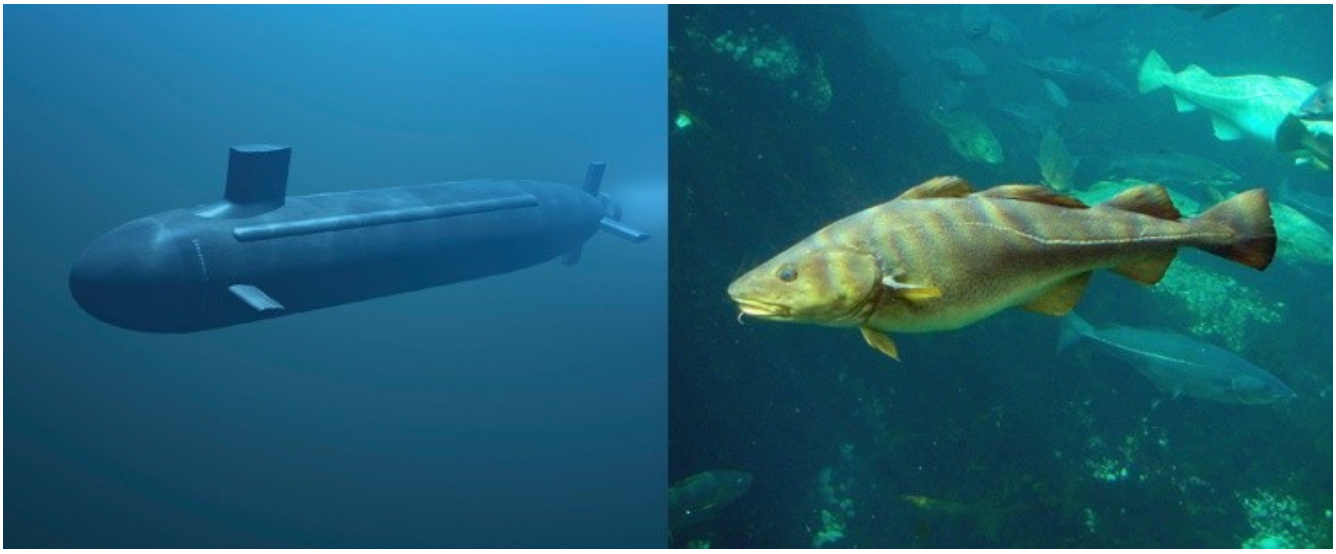


Figure 24: Frecuencias bajas, submarino, frecuencias altas, pez

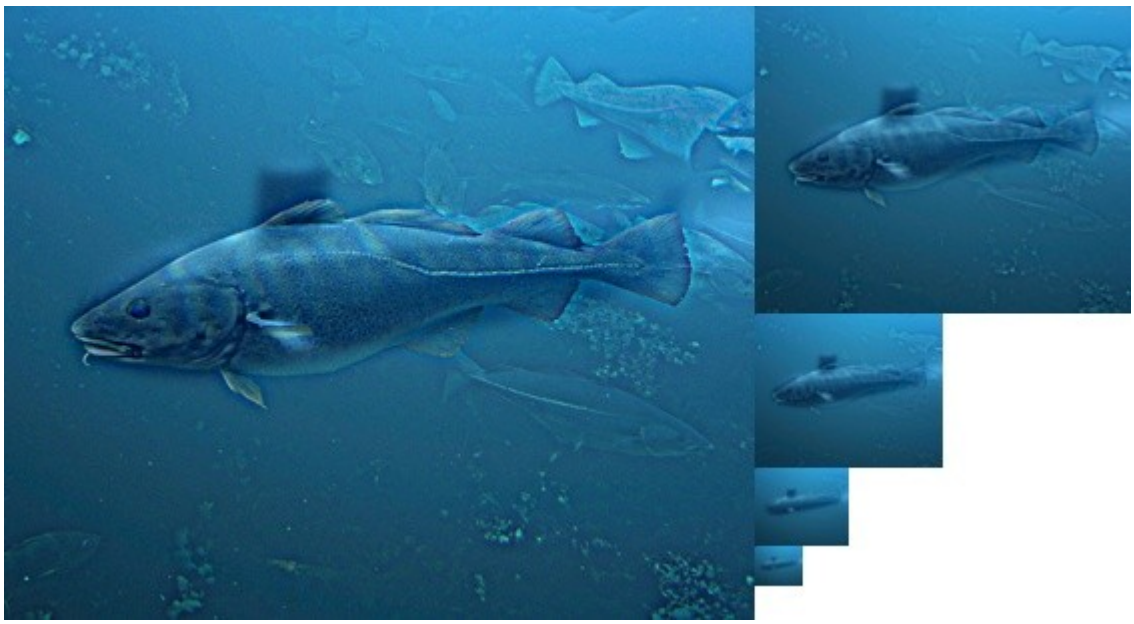


Figure 25: Pirámide Gaussiana de la imagen híbrida submarino (σ = 3) y pez (σ = 3)

En ésta imagen híbrida perdemos el color del pez. A parte de eso no hay ninguna diferencia con el resultado del ejercicio anterior.

Figure 26: Frecuencias bajas, moto, frecuencias altas, bici

Figure 27: Pirámide Gaussiana de la imagen híbrida de la moto (σ = 8) y la bici (σ = 4)

En principio en esta pareja parece que la moto sería mejor para las frecuencias altas por los contrastes que hay en el motor y lo simple que es la imagen de la bici. Aun así, se han obtenido mejores resultados cambiando el rol de las imágenes y, como podemos ver en la pirámide Gaussiana, la transición entre niveles permite cambiar la interpretación de la imagen híbrida, siendo al principio una bici con un fondo naranja detrás que luego desaparece cuanto más avanzamos en los niveles de la pirámide permitiéndonos ver la moto.

Figure 28: Frecuencias bajas, avión, frecuencias altas, pájaro

Figure 29: Pirámide Gaussiana de la imagen híbrida avión (σ = 8) y pájaro (σ = 4)

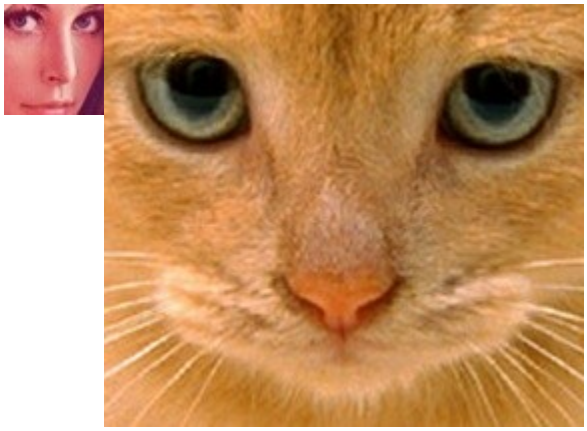
Para ésta pareja ocurre algo similar que en el caso anterior. Al contrario que para las dos primeras parejas que no era posible ver la imagen utilizada para las frecuencias bajas en los primeros niveles, en ésta, la imagen del avión se ve como una sombra borrosa detrás del pájaro debido a que tienen distintos tamaños.

También se ha probado realizar la imagen híbrida cambiando el rol de las imágenes pero no se han conseguido buenos resultados, no se dejaba de ver el avión en los niveles más altos de la pirámide.

Bonus 2

Para éste apartado se han utilizado la famosa imagen de Lena y la imagen del gato anterior. El objetivo es hacer una imagen híbrida de la cara de Lena con la cara del gato. Las imágenes originales son las siguientes:

y la regiones con las que vamos a trabajar son éstas:



La dificultad de éste apartado reside en la diferencia de las dimensiones de ambas imágenes siendo la cara de Lena una imagen de 55x50px y la del gato 211x240px. Vamos a resolver éste problema de tres formas diferentes.

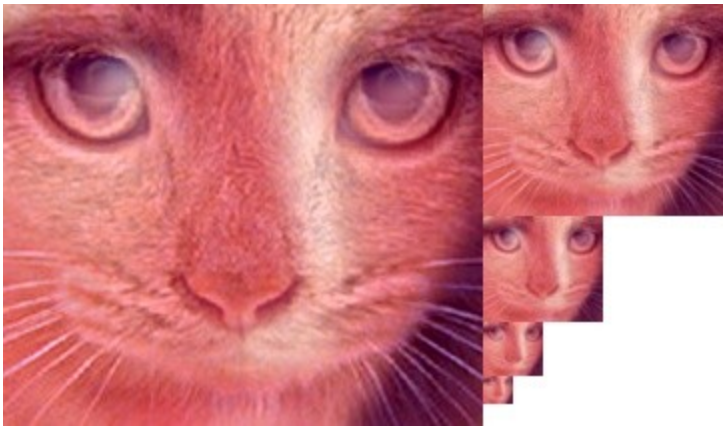
- Reduciendo las dimensiones de la cara del gato a las dimensiones de la cara de Lena
- Aumentando el tamaño de la cara de Lena a la cara del gato.
- Escalando las dos imágenes a un punto intermedio entre las dos

El motivo de hacer tres casos distintos se debe a la pérdida de detalles al modificar el tamaño de las imágenes. Al reducir las dimensiones del gato, la imagen que utilizamos para las frecuencias altas, perdemos detalles importantes. Si aumentamos la cara de Lena la deformamos perdiendo calidad de la imagen aunque, al ser ésta la que utilizamos para las frecuencias bajas puede no tener mucha importancia ya que, en principio, sólo deberíamos ser capaces de identificar su cara en los niveles más altos de la pirámide Gaussiana, cuando la imagen híbrida tiene dimensiones parecidas a la original. En el último caso sacrificamos detalles de ambas imágenes pero en menor proporción para ambas.

- Reducimos la cara del gato a la dimensión de la cara de Lena

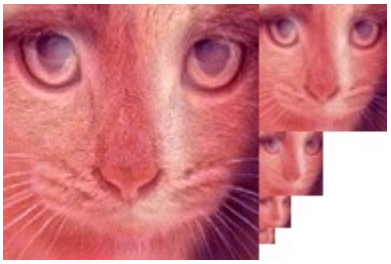
Debido al pequeño tamaño de la imagen es difícil comprobar que se ha realizado bien la imagen híbrida con la pirámide Gaussiana, ésta ha sido otra de las motivaciones para probar los otros dos casos descritos. Además, algunos detalles del gato, como los bigotes del lado derecho, no se pueden observar bien.

- Aumentamos el tamaño de la cara de Lena a la cara del gato.



Como se ha mencionado anteriormente, la cara de Lena no se puede identificar hasta los niveles 3-4 de la pirámide en lo cuales, la imagen tiene dimensiones similares a la original por lo que no se pierden muchos detalles. Aun así, la cara está deformada haciéndola más ancha que la original.

- Escalando las dos imágenes a un punto intermedio entre las dos



Posiblemente el mejor resultado de los tres casos, mantiene los detalles de la cara del gato sin deformar excesivamente ninguna de las caras.