

Práctica 1  
Visión por Computador  
Grupo de Prácticas 2

Juan José Herrera Aranda - 26516011-R  
[juanjoha@correo.ugr.es](mailto:juanjoha@correo.ugr.es)

27 de octubre de 2021



# Índice

<b>1. Ejercicio 1</b>	<b>3</b>
1.1. Ejercicio 1A . . . . .	3
1.2. Ejercicio 1B . . . . .	7
1.3. Ejercicio 1C . . . . .	8
1.4. Ejercicio 1D . . . . .	12
<b>2. Ejercicio 2</b>	<b>15</b>
2.1. Ejercicio 2A . . . . .	15
2.2. Ejercicio 2B . . . . .	19
2.3. Ejercicio 2C . . . . .	25
<b>3. Bonus</b>	<b>27</b>
3.1. Bonus A . . . . .	27
3.2. Bonus B . . . . .	40
3.3. Bonus C . . . . .	55
<b>4. Bibliografía</b>	<b>67</b>
<b>5. Anexo</b>	<b>68</b>

## 1. Ejercicio 1

### 1.1. Ejercicio 1A

Enunciado:

Considere la función Gaussiana 1D de media 0 y desviación típica  $\sigma$ . Calcular las máscaras discretas 1D de la función Gaussiana, la derivada de la Gaussiana y la segunda derivada de la Gaussiana. La función implementada debe considerar que tanto el valor de  $\sigma$  como el tamaño de la máscara son posibles entradas alternativas a la función. Represente en ejes cartesianos las máscaras obtenidas como funciones 1D y compare sus formas con las máscaras dadas por la función de OpenCV getDerivKernels para los mismos tamaños de máscara. Use los tamaños 5, 7 y 9.

Resolución:

Empezamos considerando la función gaussiana 1D de media  $\mu = 0$  y  $\sigma$

$$f(x) := e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

su derivada

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

y su segunda derivada

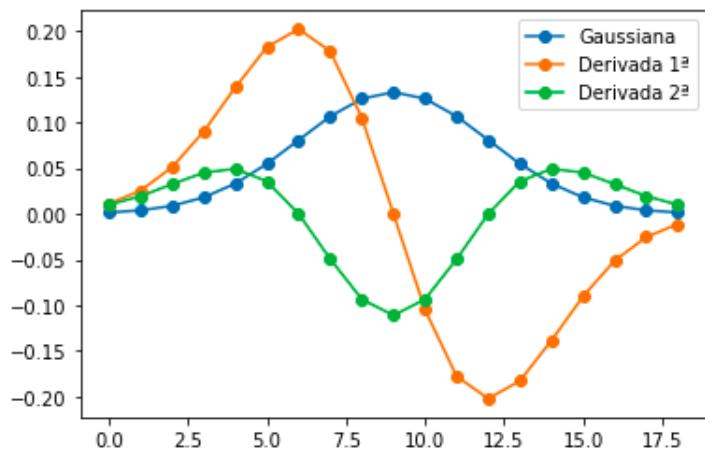
$$f''(x) := e^{-\frac{x^2}{2\sigma^2}} \frac{x^2 - \sigma^2}{\sigma^4} \quad (3)$$

Notamos que en la función gaussiana que hemos considerado hemos omitido el factor normalizador  $\frac{1}{\sqrt{2\pi\sigma^2}}$ , el cual nos asegura que  $\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} f(x) dx = 1$ , ya que al discretizar la máscara puede ser ignorado debido a que la normalización discreta se consigue imponiendo que la suma de los pesos del vector valga la unidad.

Se ha construido una función que admite como parámetros o la desviación típica o el tamaño del kernel a construir y en función de un parámetro u otro calculamos la máscara. La fórmula empleada es  $tam = 2(3\sigma) + 1$  debido a que más del 98.76 % del área de  $f(x)$  está comprendida en el intervalo  $[-3\sigma, 3\sigma]$  y a que queremos que el núcleo tenga un número impar de elementos para que de este modo tenga un único centro. La función devuelve, la máscara así como la desviación típica o el tamaño.

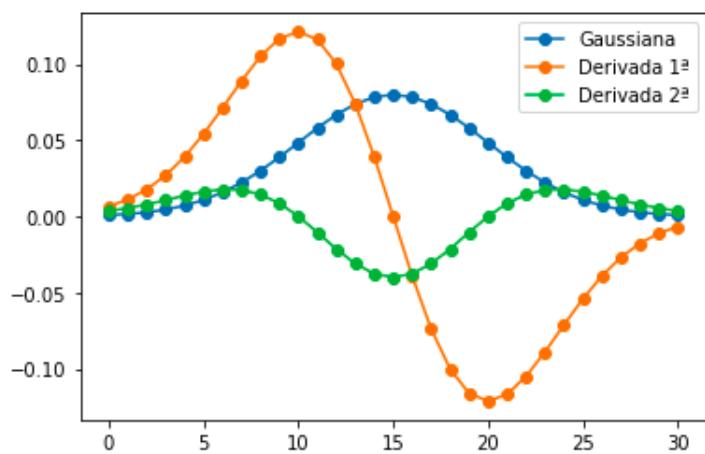
El cálculo de la máscaras se ha hecho usando las funciones analíticas que hemos presentado arriba. Veamos a continuación unos ejemplos gráficos

Presentamos las máscaras Gausiana, su primera y segunda derivada de sigma 3



Las tres presentan una tamaño de 19

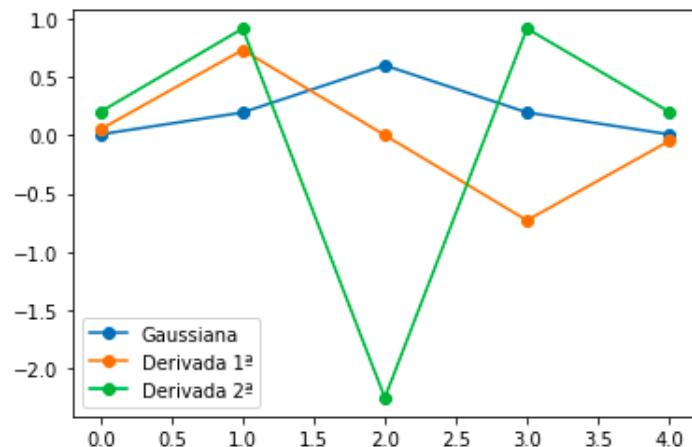
Presentamos las máscaras Gausiana, su primera y segunda derivada de sigma 5



Las tres presentan una tamaño de 31

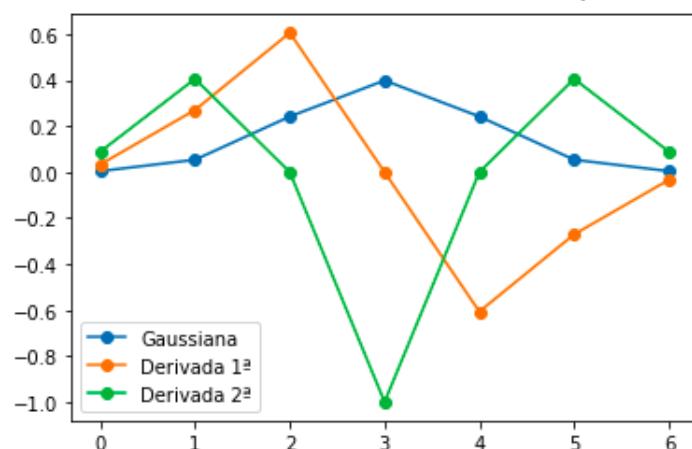
Figura 1

Presentamos las máscaras Gaussiana, su primera y segunda derivada de longitud 5



Las tres presentan una sigma de  $0.6666666666666666$

Presentamos las máscaras Gaussiana, su primera y segunda derivada de longitud 7



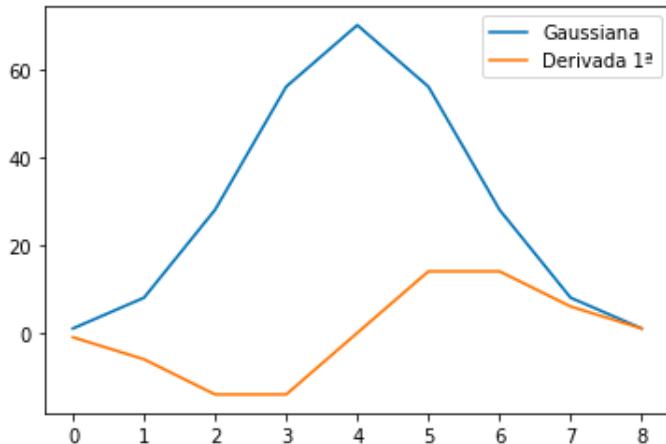
Las tres presentan una sigma de 1.0

Figura 2

Notamos que en la implementación el cálculo de la máscara Gaussiana se tiene en cuenta su normalización, sin embargo en la máscara de la derivada primera no se ha tenido en cuenta debido a que la suma de los pesos se anula y no se puede dividir en ese caso. Se podría haber realizado otro proceso de normalización para este caso como dividir por la suma del valor absoluto de los pesos por ejemplo o multiplicar por  $\sigma$  y  $\sigma^2$  en el caso de la primera derivada y segunda respectivamente.

Veamos a continuación una representación en ejes cartesianos de las máscaras obtenidas con la función gerDerivkernels presentada en el anexo 5 y una comparativa.

GET DERIV KERNELS



Filtro Gaussiano propio y filtro obtenido con getDerivKernel Normalizados

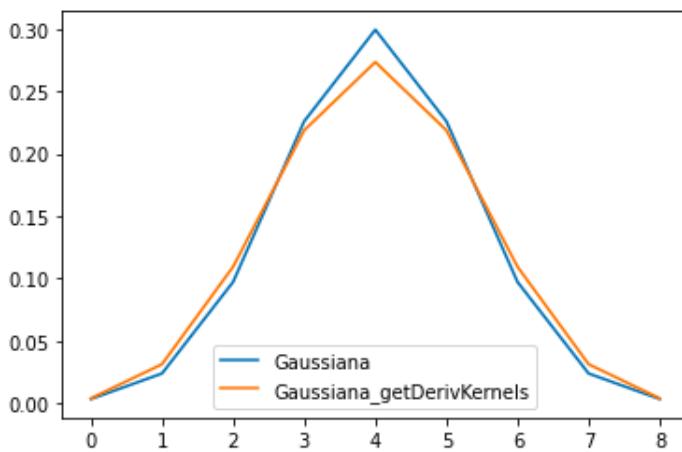


Figura 3: En la imagen superior se muestra gráficamente la salida de getDerivKernel con parámetros kszie="9", dx=1, dy=0. En la imagen superior notamos que la función derivada no se corresponde realmente con la pendiente de la recta tangente a la gráfica azul ya que está invertida. En la imagen inferior se hayan las dos máscaras Gaussianas superpuestas

Notamos que pese a que las máscaras tienen el mismo tamaño, son ligeramente distintas, implicando que las derivadas también sean ligeramente distintas. En la imagen observamos que la máscara calculada usando la función gaussiana tiene una amplitud ligeramente inferior en la ladera y una altura suavemente superior. La diferencia fundamental entre las dos máscaras es en la hora de calcular los coeficientes. Por un lado, la máscara Gaussiana que hemos calculado ha sido usando analíticamente su función de densidad, así como la función primera y segunda derivada. Por otra parte, getDerivKernel usa para el cálculo una aproximación aplicando el operador de Sobel priorizando así eficiencia a exactitud, aunque como se muestra en la imagen, la diferencia no es ligera pero tampoco muy pronunciada. Como curiosidad, los operadores standards de Sobel son:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

## 1.2. Ejercicio 1B

Calcule las máscaras discretas 1D de longitud 5 y 7 tanto de alisamiento como de derivada de primer orden generadas a partir de la aproximación binomial de la Gaussiana y la máscara derivada de longitud 3. Ejecute la función `cv2.getDerivKernels(0,1,9)`. Observe los vectores de salida y compárelos con los previamente calculados. ¿Qué relación hay? ¿Qué conclusiones extrae sobre la aproximación de OpenCV al cálculo de las máscaras de derivadas de primer orden?

Para este apartado se ha hecho uso de la propiedad conmutativa y asociativa de la convolución. Partimos del vector  $\frac{1}{2}[1, 2, 1]$  y tenemos que

$$\frac{1}{2}[1, 2, 1] * \frac{1}{2}[1, 2, 1] = \frac{1}{4}[1, 4, 6, 4, 1]$$

obtenemos así la máscara de longitud 5, por otra parte haciendo

$$\frac{1}{4}[1, 4, 6, 4, 1] * \frac{1}{2}[1, 2, 1] = \frac{1}{8}[1, 6, 15, 20, 15, 6, 1]$$

obteniendo la máscara de longitud 7.

Para el cálculo de la máscara derivada simplemente hacemos  $\frac{1}{2}[1, 2, 1] * [-1, 0, 1]$  para la máscara de longitud 5 y  $\frac{1}{2}[1, 2, 1] * [-1, 0, 1] * \frac{1}{2}[1, 2, 1]$  para la máscara de longitud 7 ( $\frac{1}{4}[-1, -2, 0, 2, 1]$  y  $\frac{1}{8}[-1, -4, -5, 0, 5, 4, 1]$  respectivamente).

```
Máscara discreta de longitud 5: [1 4 6 4 1]
Máscara discreta de longitud 7: [ 1   6 15 20 15   6   1]
Máscara discreta derivada de longitud 5: [-1 -2   0   2   1]
Máscara discreta derivada de longitud 7: [-1 -4 -5   0   5   4   1]
=====
Máscara discreta de longitud 9: [ 1   8 28 56 70 56 28   8   1]
GetDerivKernel smooth: [[ 1.   8. 28. 56. 70. 56. 28.   8.   1.]]
Máscara discreta derivada de longitud 9: [-1 -6 -14 -14   0   14 14   6   1]
GetDerivKernel dx: [[ -1. -6. -14. -14.   0. 14. 14.   6.   1.]]
```

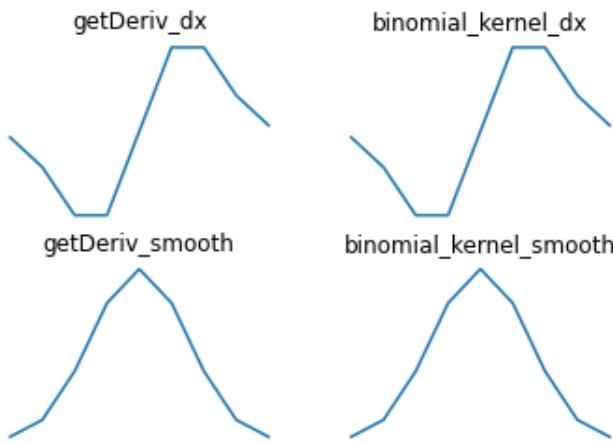


Figura 4: Gráficas de la salida de la función `getDerivKernel` de parámetros `dx=1`, `dy=0`, `ksize=9` y la máscara gaussiana y su derivada obtenida por la aproximación binomial SIN NORMALIZAR. El kernel resultante de `getDerivKernel` es  $[1, 8, 28, 56, 70, 56, 28, 8, 1]$  y se obtiene convolucionando el kernel de longitud 7 que hemos calculado antes con el vector  $[1, 2, 1]$ .

Como bien se observa en la imagen de arriba, las máscaras gaussianas obtenidas por la aproximación binomial sin normalizar y la dada por la función `getDerivKernel` son las mismas.

Hemos decidido no normalizar simplemente para que en la imagen sea más fácil ver a simple vista que son iguales. De hecho, `getDerivKernel` tiene una opción para normalizar también. Por tanto como conclusión podemos decir que `getDerivKernel` realiza una aproximación binomial para calcular las máscaras. La eficiencia es de  $O(w)$  siendo  $w$  el tamaño del vector por lo que se prima la eficiencia. La máscara implementada usando la función gaussiana ofrece la máxima exactitud mientras que el núcleo de la binomial es más amplio de lo necesario para valores crecientes de  $\sigma$ . Por ejemplo, para  $\sigma = 1,41$  una anchura de  $w=7$  capturaría casi el 98.76 % de la curva (ya que  $5\sigma \approx 7$ ), pero el triángulo binomial utiliza  $w=9$  para representar esta gaussiana de ahí que tenga menor altura para no capturar más área de la que debiera..

### 1.3. Ejercicio 1C

**Implementar la convolución de una imagen con una máscara 2D de dimensiones inferiores a la imagen suponiendo la propiedad de separabilidad de la máscara e imponiendo bordes reflejados. La entrada a la función serán las máscaras 1D descomposición de la máscara 2D. Comparar su funcionamiento con:**

- **La salida de `cv2.GaussianBlur` para una máscara de entrada Gaussiana con iguales parámetros en ambos casos. Mostrar ambos resultados.**
- **Usar las máscaras del punto A para calcular la imágenes derivadas respecto de x e y de una imagen dada. Mostrar los resultados**



Figura 5: Imagen que muestra los dos tipos de padding empleados en la práctica. El radio del padding es 30

El objetivo es aplicarle a una imagen un filtro 2D gaussiano ya sea de alisamiento o de derivada. Gracias a la separabilidad de los kernels gaussianos, podemos descomponer el filtro como convolución de dos kernels 1D de la misma dimensión, pero traspuestos. Haciendo ahora uso de la propiedad asociativa de la convolución, el problema se reduce en hacer una convolución por filas de la imagen con un vector fila y al resultado aplicarle una convolución por columnas con el otro vector columna. Notemos que lo anterior no es válido para filtros no Gaussianos, ya que no tienen por qué ser separables.

Para la resolución del ejercicio primeramente se ha creado una función para realizar el padding de una imagen, en particular de una matriz. Podemos elegir el modo de padding (Fig 5) como parámetro, si queremos un padding negro (parámetro `modo_padding = "negro"`) o un padding reflejado, es decir, como un espejo, (`modo_padding = 'reflejado'`) Así pues, dada una matriz de dimensión  $N \times N$  se devuelve una matriz de dimensión  $(N + 2k) \times (N + 2k)$  siendo  $k$  el

número de píxeles con un determinado valor que se añaden a todas las columnas y filas de la matriz. Veamos un ejemplo de como lo tenemos implementado, sea

$$B = \begin{pmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{pmatrix}$$

una matriz 3x3 y consideremos  $k = 1$ , entonces la matriz resultante quedaría de la forma

$$Bp = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 20 & 30 & 0 \\ 0 & 40 & 50 & 60 & 0 \\ 0 & 70 & 80 & 90 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Después, se ha creado una función que recibe como argumento a la imagen y dos máscaras, y realiza una correlación por columnas y a la salida le aplica otra correlación pero por filas. Veámoslos con un ejemplo

Consideramos la matriz anterior una vez realizado el padding,  $Bp$ . Vamos a realizar una convolución por columnas con el vector  $kv = [1, 2, 1]^t$ . Primero extraemos de  $Bp$ , la submatrix de dimensión 3x5

$$Bp_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 20 & 30 & 0 \\ 0 & 40 & 50 & 60 & 0 \end{pmatrix}$$

y aplicamos el producto escalar con el vector  $kv$  columna por columna,

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 20 & 30 & 0 \\ 0 & 40 & 50 & 60 & 0 \end{pmatrix} < * >_{col} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = (0 \ 60 \ 90 \ 120 \ 0)$$

Repetimos ahora el proceso con las submatrices

$$Bp_3 = \begin{pmatrix} 0 & 10 & 20 & 30 & 0 \\ 0 & 40 & 50 & 60 & 0 \\ 0 & 70 & 80 & 90 & 0 \end{pmatrix} \text{ y } Bp_3 = \begin{pmatrix} 0 & 40 & 50 & 60 & 0 \\ 0 & 70 & 80 & 90 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

y concatenando los resultados obtenemos la matriz convolucionada por columnas

$$Bp_{col} = \begin{pmatrix} 0 & 60 & 90 & 120 & 0 \\ 0 & 160 & 200 & 240 & 0 \\ 0 & 180 & 210 & 240 & 0 \end{pmatrix}$$

Ahora, para realizar la convolución por filas se procede de forma análoga trasponiendo tanto la matriz como el kernel. Finalmente trasponemos el resultado para obtener la matriz convolucionada.

$$Bp_{convolucionada} = \begin{pmatrix} 210 & 360 & 330 \\ 520 & 800 & 680 \\ 570 & 840 & 690 \end{pmatrix}$$

Nos damos cuenta de que más que una convolución, hemos realizado una correlación, esto se debe a que los kernels son simétricos. En caso de que no lo fueran, se procedería de forma análoga pero con los elementos de la máscaras invertidos de orden.

Así pues nuestra función de suavizar recibirá dos filtros gaussianos, y aplicará la convolución descrita anteriormente a una imagen de dimensión mayor y usando bordes reflejados como modo de padding. Veamos ejemplos a continuación.

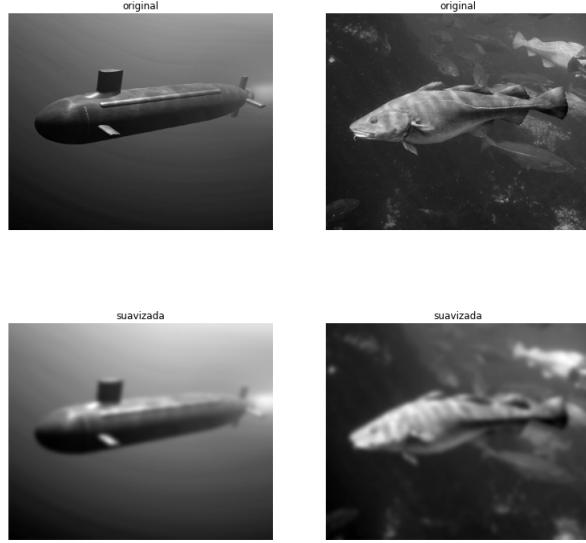


Figura 6: En la parte superior se presentan las imágenes originales y en la inferior tenemos las imágenes tras aplicar un suavizado con  $\sigma = 3$  y usando bordes reflejados como método de padding

Aplicamos ahora la continuación a la imagen la función proporcionada por opencv GaussianBlur detallada en el anexo (5). Tal y como muestra la Fig. (7) observamos que visualmente las imágenes son idénticas. Sin embargo, haciendo un estudio más detallado vemos que al calcular la diferencia (Fig. 8) entre las imágenes suavizadas y sus respectivas versiones de GaussianBlur el resultado es la silueta del pez y del submarino, correspondiente a las frecuencias altas. Si las imágenes fueran iguales, veríamos esas ventanas en negro ya que la diferencia en ese caso produciría una matriz de ceros. Por tanto, ya tenemos evidencias para justificar que las imágenes no son iguales. De hecho, se ha calculado (Fig. 9) la diferencia entre los píxeles con mayor valor y menor valor entre las imágenes suavizadas por nuestra implementación y la de openCV y vemos también que esa diferencia es no nula. También se ha calculado el error cuadrático medio y observamos que es también no nulo tanto para el pez como para el submarino.

Mostramos también las imágenes resultantes tras el filtro derivada tanto de x como de y calculado en el apartado 1A a la imagen de un gato y de un pez.

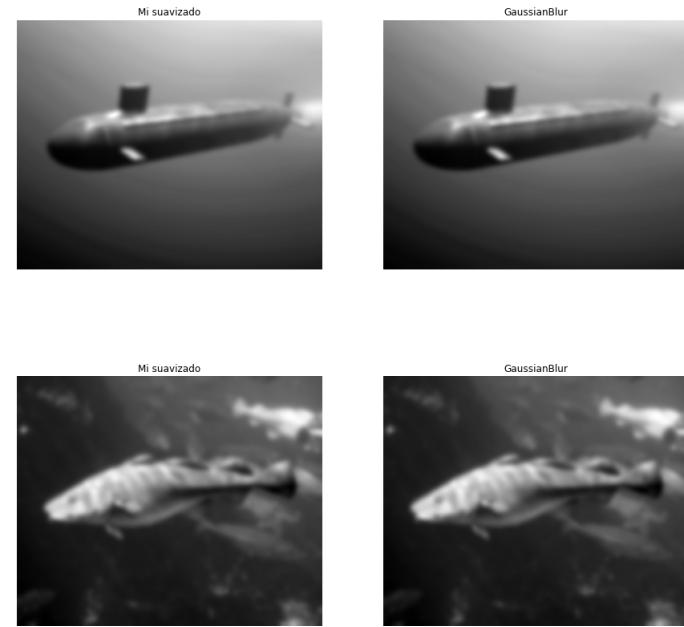


Figura 7: La parte izquierda de la ventana corresponde a las imágenes suavizadas a través de nuestra implementación mientras que en la parte de la derecha corresponde a las imágenes suavizadas por medio de la función GaussianBlur. En todas se ha usado  $\sigma = 3$ .

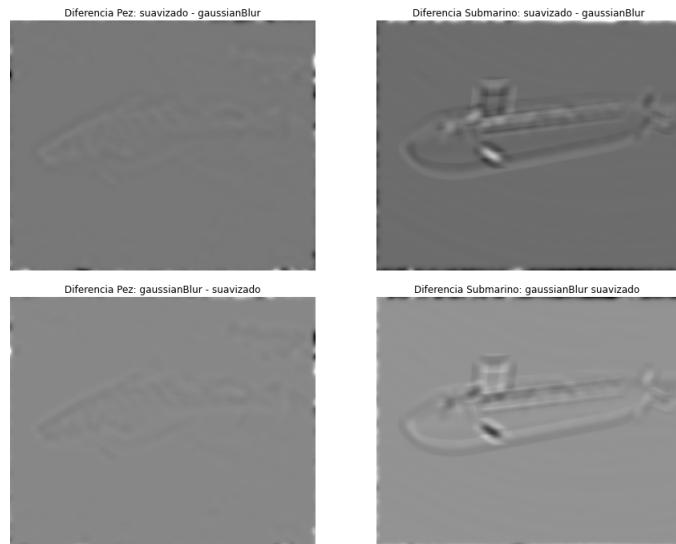


Figura 8: Estas imágenes representan la diferencia entre la imagen del pez (resp. submarino) suavizada por el método que hemos implementado y la producida por el filtro GaussianBlur y la misma diferencia pero permutando el minuendo de la diferencia por el sustraendo.

```

Error Cuadrático Medio:
Pez          --> 1155.0203175824006
Submarino    --> 50.87655729373286
=====
Diferencia entre los valores máximos:
Pez          --> 0.09640354057401623
Submarino    --> 0.11133431421802698
Diferencia entre los valores máximos:
Pez          --> -0.09246471992460314
Submarino    --> -0.11145127185596237

```

Figura 9: Calculos realizados para la comparación de imágenes.



Figura 10: Imagen Original, derivada con respecto x e y. Se ha usado  $\sigma = 3$ .



Figura 11: Imagen Original, derivada con respecto x e y. Se ha usado  $\sigma = 3$

#### 1.4. Ejercicio 1D

Usar las implementaciones del punto A para calcular máscaras normalizadas de la Laplaciana de una Gaussiana. Mostrar ejemplos de funcionamiento usando bordes reflejados y dos valores de sigma (1 y 3) con alguna imagen. Compare los resultados con la salida dada por OpenCV con la función Laplaciana. Discuta pros y contras.

Para la realización de este ejercicio se ha creado una función llamada `laplacian_of_gaussian` que recibe como argumento una imagen y un valor de sigma y devuelve la imagen laplaciana de la gaussiana. Para ello calculamos a partir del sigma dado de argumento las máscaras discretas de alisamiento y segunda derivada usando el ejercicio 1. Convolucionamos la imagen con la máscara de la derivada segunda por filas y el alisamiento por columnas y viceversa para obtener

la segunda derivada de la imagen respecto de la dirección del eje X y del eje Y. Finalmente sumamos el resultados y lo normalizamos multiplicando por  $\sigma^2$ .

La función usada para el calculo de la laplaciana en opencv es `cv2.Laplacian` detallada en el anexo 5. Esta función como utiliza el gradiente de las imágenes, llama internamente al operador Sobel para realizar su cálculo. Y como ya sabemos, el operador Sobel es una aproximación por lo que el cálculo no ser hará de forma exacta, esto puede ser una **desventaja** si precisamos de máxima exactitud. Por otra parte, como gran **ventaja** que presenta es la eficiencia ya que como se acaba de comentar, se usa el operador Sobel ya que por dentro usa la función `getDerivKernels` otorgándole eficiencia.

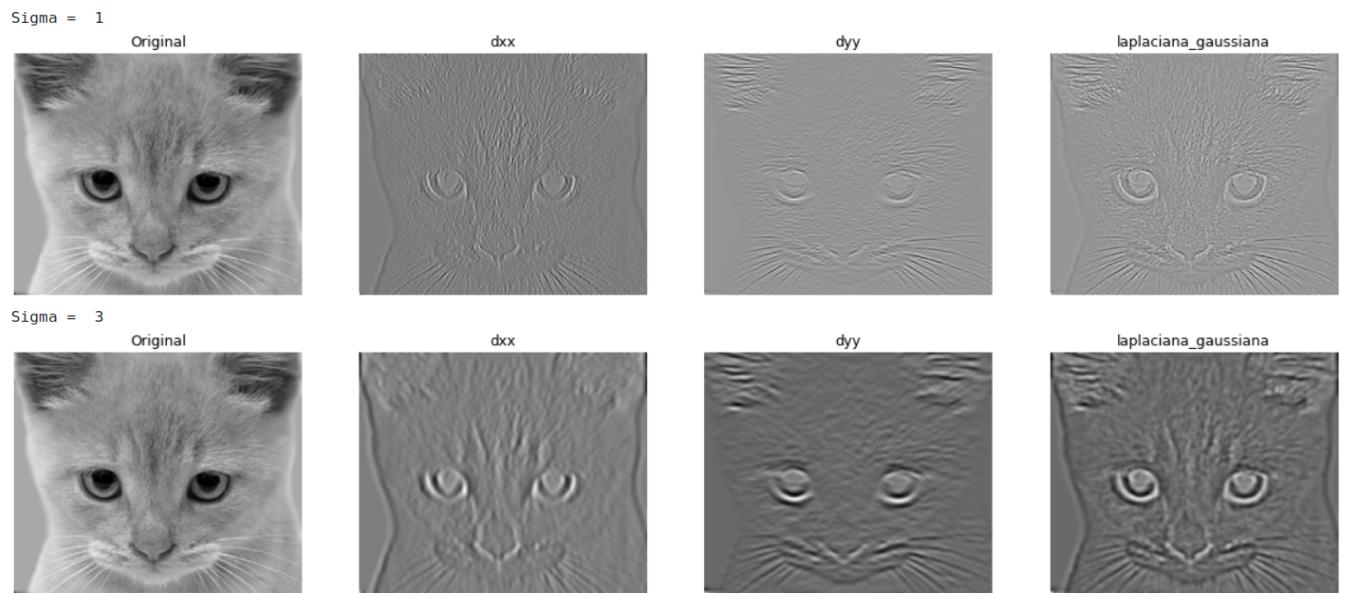


Figura 12: Se presenta la imagen del gato junto con su segunda derivada en la dirección del eje X y del eje Y y la laplaciana de la gaussiana

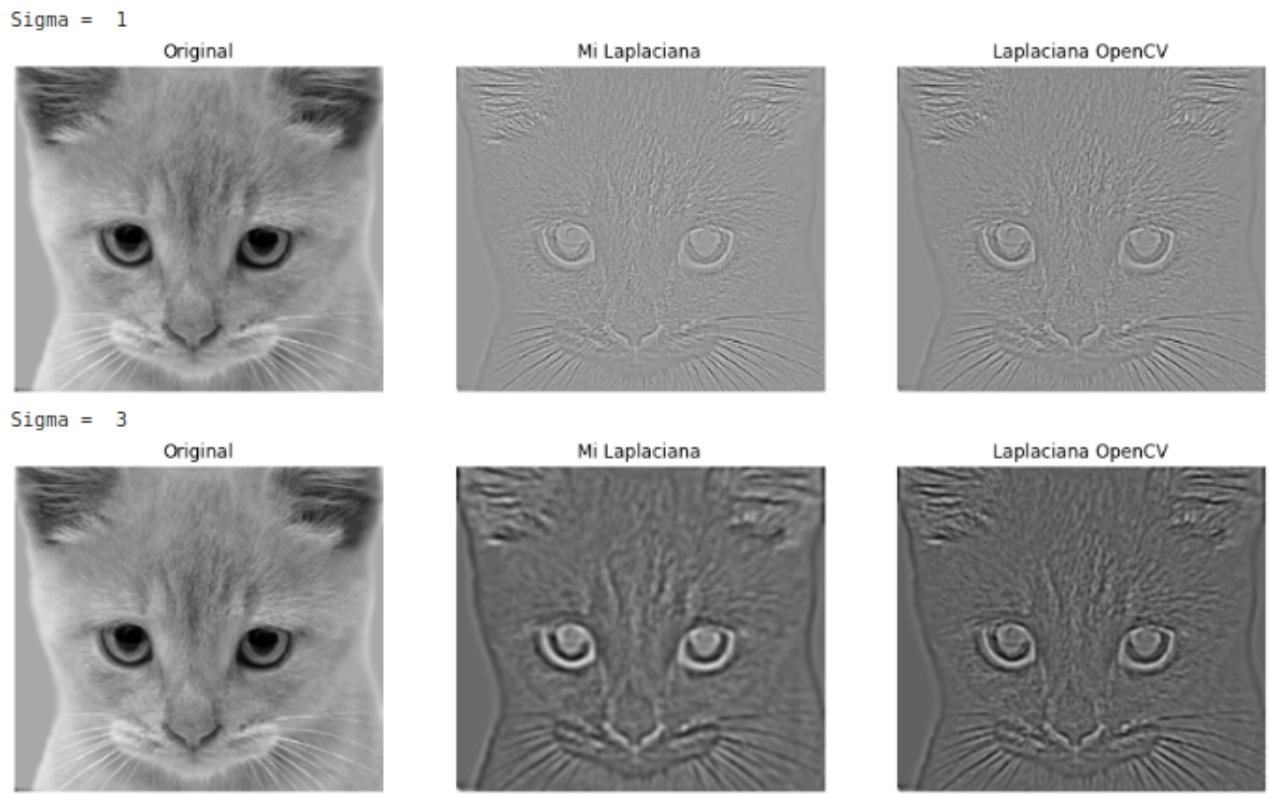


Figura 13: Se presenta la imagen original, nuestra LoG y la LoG usando la función Laplacian de OpenCV

## 2. Ejercicio 2

IMPLEMENTAR funciones para las siguientes tareas (4 puntos). Usar funciones implementadas en el ejercicio 1. No se permite el uso de funciones específicas de OpenCV.

### 2.1. Ejercicio 2A

Una función que genere una representación en pirámide Gaussiana de 4 niveles de una imagen. Mostrar ejemplos de funcionamiento usando bordes replicados o reflejados, y justificar la elección de los parámetros. Mostrar todos los niveles de la pirámide en una única imagen. Comparar los resultados con la pirámide obtenida usando las funciones OpenCV.

En este ejercicio vamos a realizar la construcción de una pirámide Gaussiana usando funciones propias y usando las funciones que trae OpenCv. Ésta se construye de forma iterativa suavizando la imagen y reescalándola a la mitad de su tamaño original. Se procede de la misma manera de forma sucesiva hasta cuántos niveles queramos.

Las pirámide Gaussianas son útiles por permitir optimizar la información ya que al reducir el tamaño de la imagen algunos algoritmos permiten operar de forma más rápida.

El proceso de submuestreo usado ha sido eliminar las filas y columnas impares quedándonos así con las pares. El suavizado se aplica con el mismo sigma a todos los elementos de la pirámide. Notar también que dependiendo de la forma en la que se haga el padding, obtenemos un resultado u otro. Algunos ejemplos son mostrados en Fig. 14 y Fig. 15.

Posteriormente se ha realizado la implementación de la pirámide Gaussiana con funciones propias de OpenCV, en particular, con la función *pyrDown* (detallada en el anexo 5). Esta función tiene la peculiaridad de que realiza el emborronado con la matriz

$$\frac{1}{256} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

obtenida realizando el outer product de la máscara gaussiana obtenida por la aproximación del método binomial  $\frac{1}{16}[1, 4, 6, 4, 1]$  consigo misma. No tenemos opciones de elegir nosotros nuestra propia máscara. Para poder compararla entonces con nuestra pirámide gaussiana debemos ajustar por lo menos el tamaño del kernel que usemos al tamaño del vector anterior. Realizando los cálculos pertinentes llegamos a que el tamaño del kernel 5 se corresponde con un  $\sigma \approx 0,66667$ . Notamos que van a existir diferencias entre las dos pirámides porque de nuevo usamos nosotros una expresión analítica para calcular el filtro mientras que la otra pirámide usa una aproximación. De hecho, nuestro filtro resultante es  $[0,00664603, 0,19422555, 0,59825683, 0,19422555, 0,00664603]$ . En la Fig. ?? se muestra un ejemplo visual.

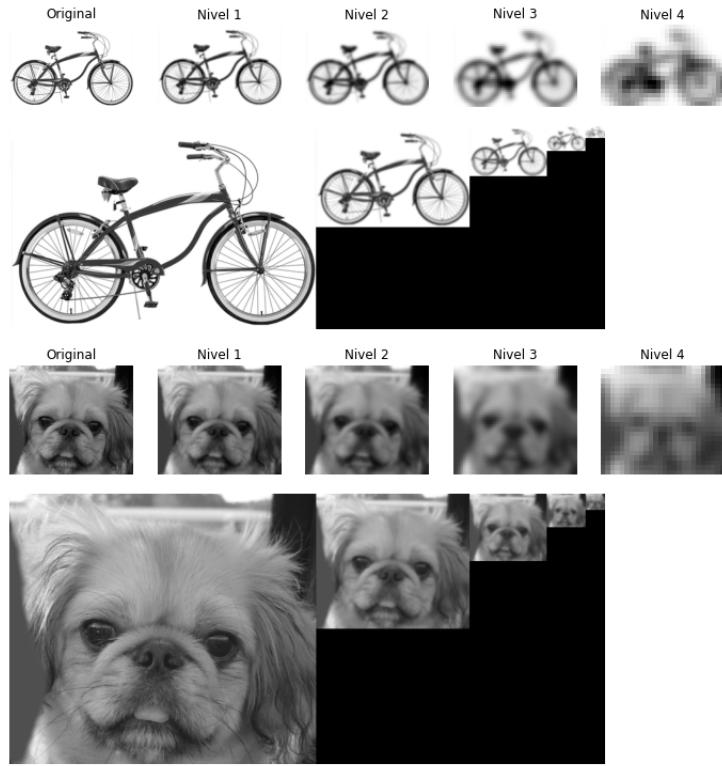


Figura 14: Pirámide gausiana de dos imágenes. Se muestra primero todos los niveles al mismo tamaño para resaltar el efecto del suavizado y a continuación apilada en forma de pirámide.  $\sigma = 3$

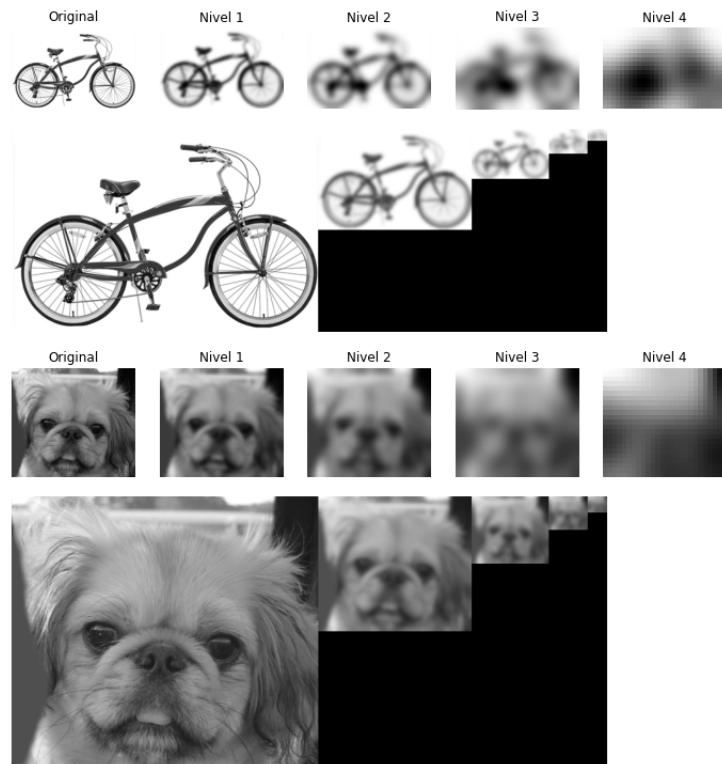


Figura 15: Pirámide Gaussiana con  $\sigma = 5$

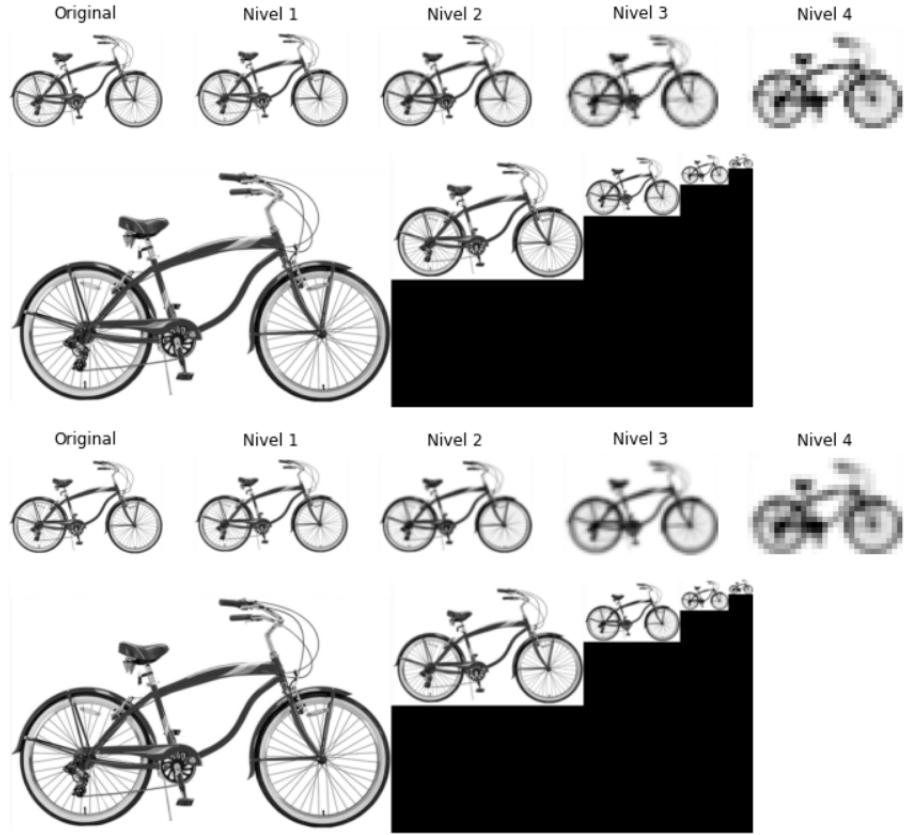


Figura 16: La pirámide de arriba corresponde con nuestra implementación mientras que la de abajo con la implementación de openCv que ya se ha comentado. Podemos observar en los últimos niveles de las pirámides algunas diferencias.

Finalmente, el ejercicio pedía comentar la elección justificada del parámetro. Pues bien, en la imagen de abajo observamos que para valores de sigma muy bajos los niveles de la pirámide aparecen cada vez más pixeladas. Esto se debe al bajo nivel de emborronamiento. Conforme aumentamos el valor de sigma se puede ver que cada vez aparecen las imágenes menos pixeladas y mas emborronadas. Un sigma muy bajo provoca que el suavizado se realice en un entorno casi inapreciable del punto y al hacer el reescalado, la eliminación de la mitad de filas y columnas se note mucho más. Mientras que para sigmas más grandes, tenemos que la máscara es mucho más amplia provocando que el emborronamiento se haga en una región del punto mucho mayor. No es bueno ni valores muy pequeños, ni valores muy grandes donde el nivel de emborronamiento es tal que no se puede distinguir ya la imagen.



Figura 17: Caption



Figura 18: Caption

## 2.2. Ejercicio 2B

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

**Mostrar todos los niveles de la pirámide en una única imagen. comparar los resultados con la pirámide obtenida usando las funciones OpenCV.**

**Nota:** En las visualizaciones de las imágenes se ha considerado la imagen del último nivel gaussiano con la etiqueta de 'original' por ser la imagen de partida para la la pirámide laplaciana y la etíiqueta de 'nivel i' para la construcción del i-esimo elemento de la pirámide.

En este apartado vamos a realizar la implementación de una pirámide Laplaciana usando funciones propias y usando las funciones que trae OpenCv. Para el cálculo de la pirámide, partimos de la pirámide Gaussiana y procedemos de forma iterativa expandiendo el nivel ' $i+1$ ' al tamaño del nivel ' $i$ ' - ya que la pirámide gaussiana es decreciente - y realizamos una diferencia entre la expansión realizada y el nivel ' $i$ ' de la pirámide Gaussiana.

Veamos unos ejemplos a continuación

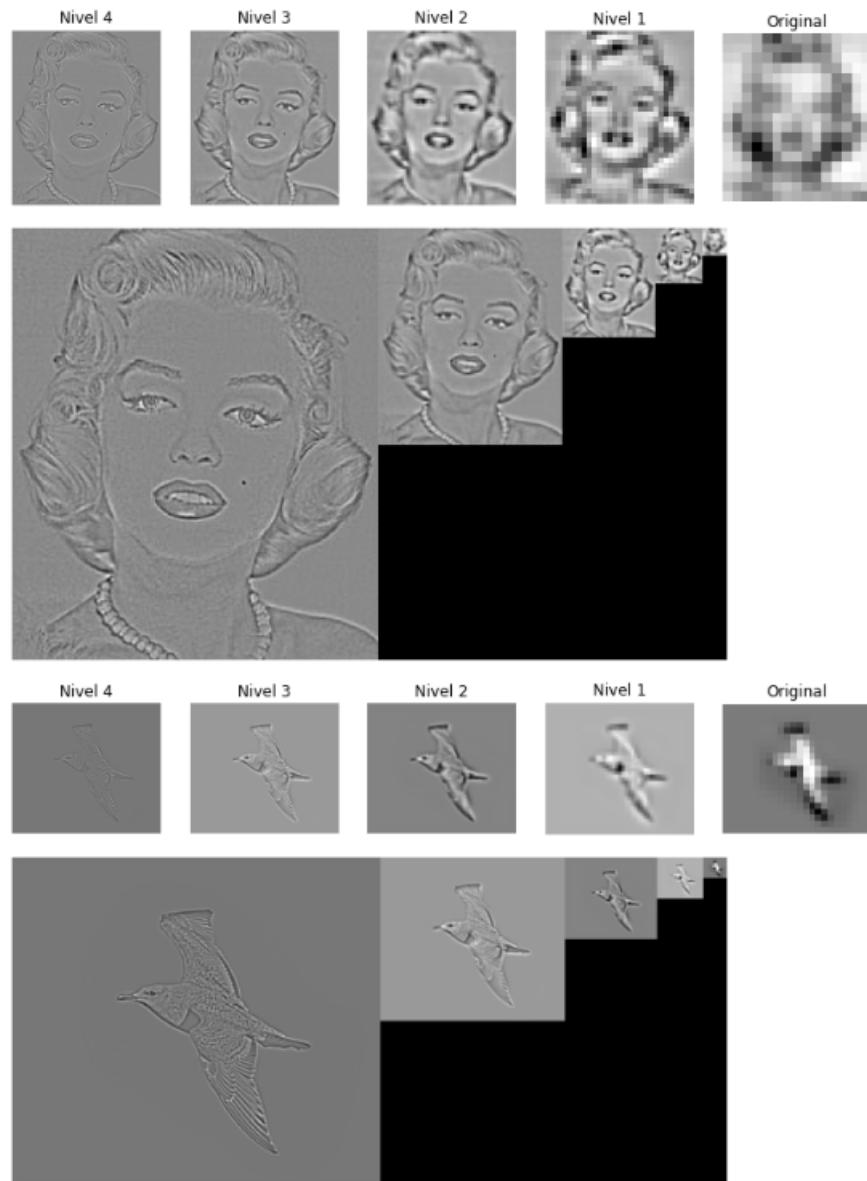


Figura 19:  $\sigma = 2$

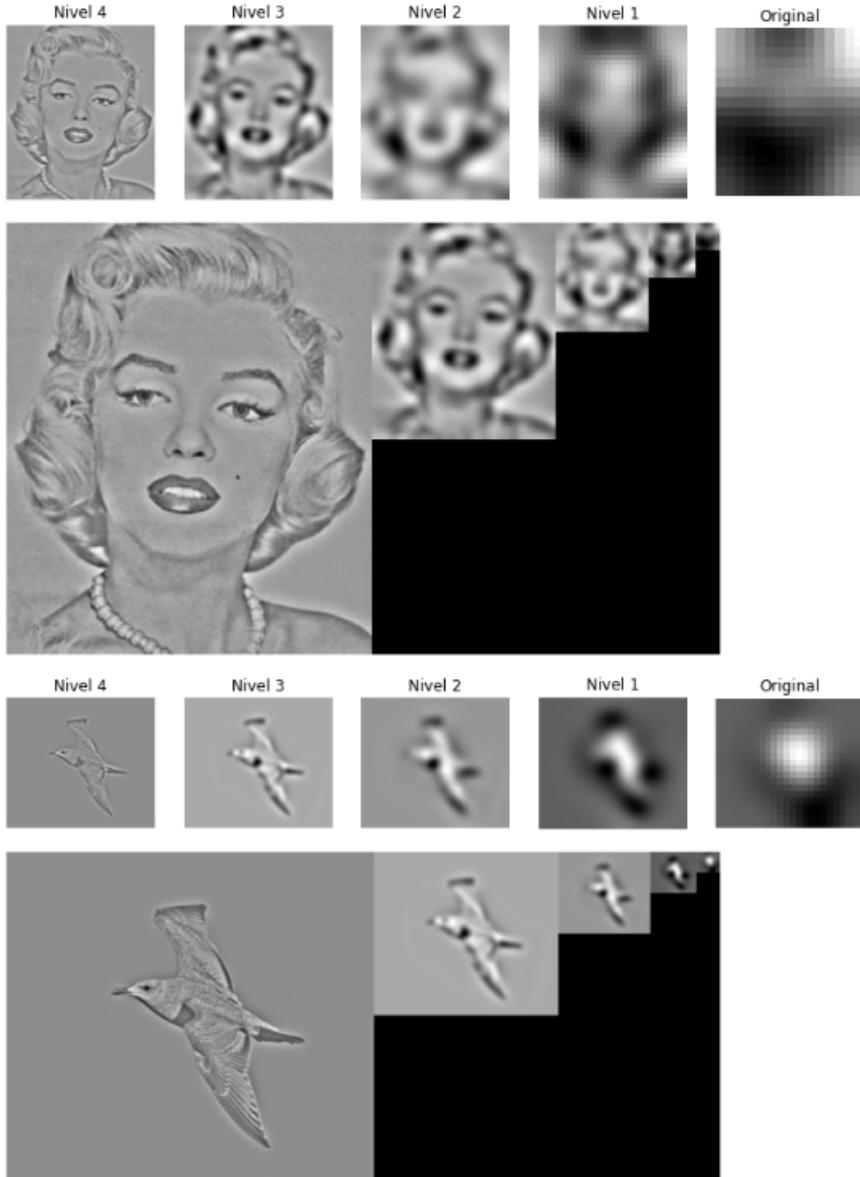


Figura 20:  $\sigma = 5$

Como en el apartado anterior, se ha implementado una manera alternativa de calcular la pirámide laplaciana con funciones de OpenCV. En este caso usaremos la función `pyrUp` (detallada en el anexo 5) la cual tiene la peculiaridad de que aplica un alisamiento a la imagen después de expandir, por lo que aquí tenemos la primera diferencia con respecto a nuestra implementación. Ésto lo hace porque la opción que tiene por defecto para expandir es insertar ceros y de este modo suaviza el efecto del negro. Ésto último constituye otra diferencia con respecto a nuestro método ya que nosotros aplicamos la interpolación por defecto que viene con `cv.resize` al expandir la imagen. Por otra parte, a nuestra función le pasamos como argumento la pirámide gaussiana hecha a partir de nuestros procedimientos y a la otra se le pasa la pirámide Gaussiana construida a partir de `opencv`.

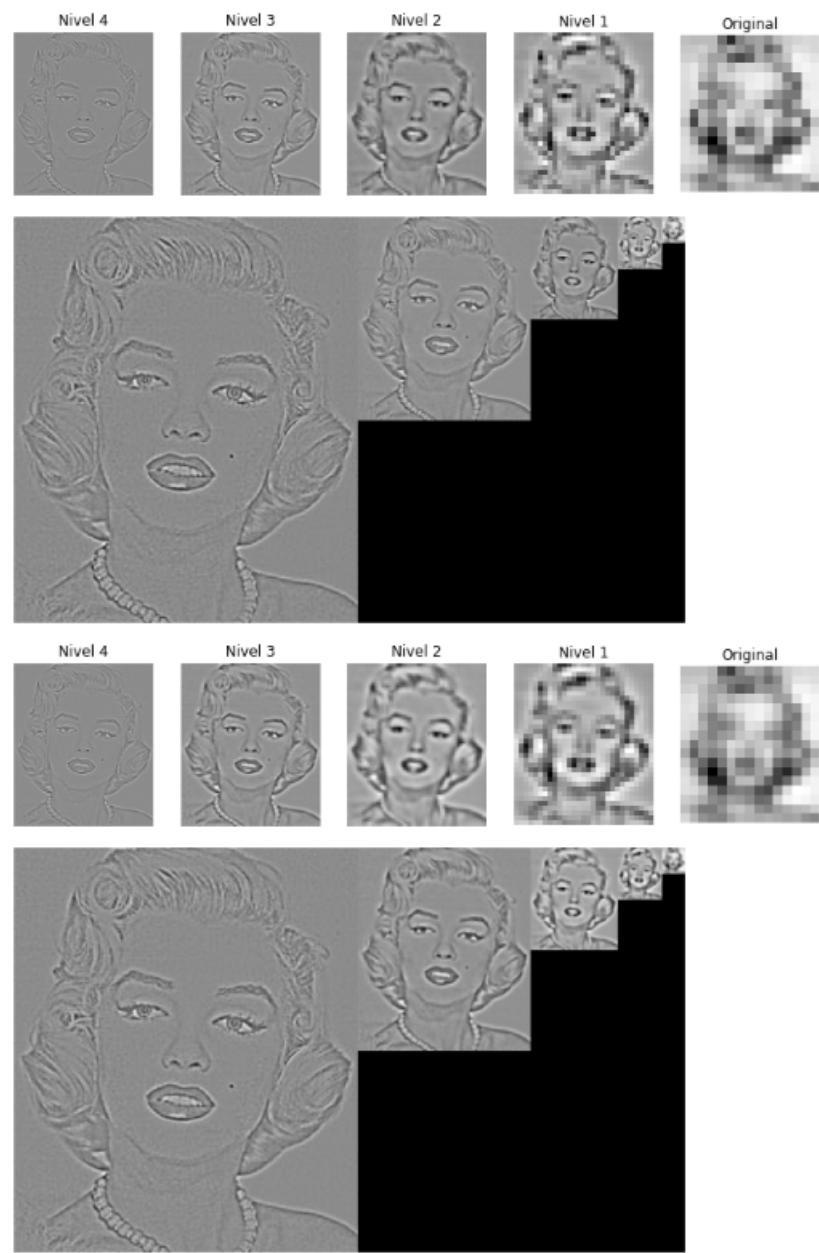


Figura 21: En la parte superior de la imagen se muestra la pirámide laplaciana correspondiente a nuestra implementación mientras que en la parte e la izqueirda se muestra la pirámide laplaciana usando la función `pyrUp` de OpenCV

A simple vista, tal y como se observa en la Fig. 21 podemos pensar que ambas son idénticas. Sin embargo, al observarlas de forma analítica aplicando algunas métricas vemos que son distintas, todo ésto se muestra en la figura Fig. 22

```
Error Cuadrático Medio : 156756.94855861706
Media grises OpenCV: -0.0009779956105870021
Media grises nuestra implementación: -0.00988200727413935
```



Figura 22: En la figura observamos que las dos imágenes producidas por las distintas pirámides laplacianas no son iguales. Primero presentamos el error cuadrático medio que es distinto de cero por lo que las imágenes ya sabemos que no son iguales. Por otra parte tenemos la media de intensidad de gris en ambas imágenes, que también difieren. Esta segunda métrica no es muy precisa porque puede darse el caso de que las imágenes sean distintas pero que coincidan en medida. Finalmente presentamos como evidencia final la diferencia de imágenes permutando en una el minuendo con el sustraendo. Nos fijamos en algunos detalles como por ejemplo a saturación en blanco de la imagen de la derecha en características como el lunar de la señora o en los fosas nasales, mientras que en la de la izquierda está en negro. Esto es debido al hecho de realizar una diferencia u otra.

Finalmente, para acabar este apartado del ejercicio, estudiemos cómo varía la pirámide laplaciana implementada por nosotros según distintos valores de sigma.

Gracias a la Fig. 23 y Fig. 24 observamos que para empezar, partiendo de un valor muy bajo de sigma, empezamos con una imagen muy pixelada de partida debido a que proviene de la pirámide gaussiana. Además al ser el emborronamiento en la gaussiana en un entorno pequeño provocará que no haya mucha diferencia al computarla. Entonces las imágenes laplacianas en los siguientes espacios de escalas resaltan muy poco los bordes. Conforme va creciendo el sigma los bordes en el último nivel de la laplaciana se van distinguiendo un poco más y se puede apreciar con más claridad la figura del perro. Finalmente para valores muy altos de sigma tenemos que la laplaciana final nos proporciona muy detalladamente los bordes de la figura.

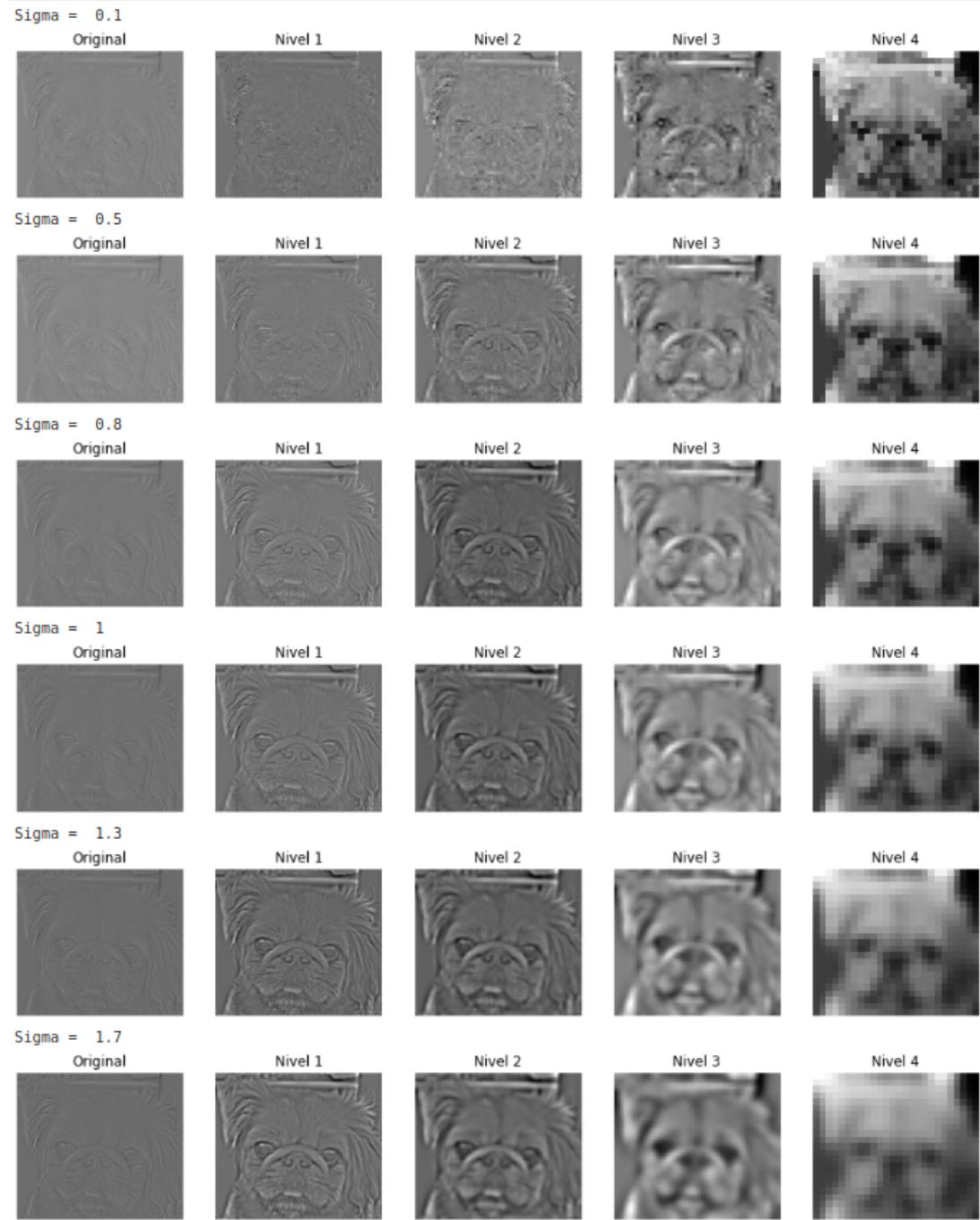


Figura 23: Pirámide laplaciana correspondiente a sigmas con valores bajos

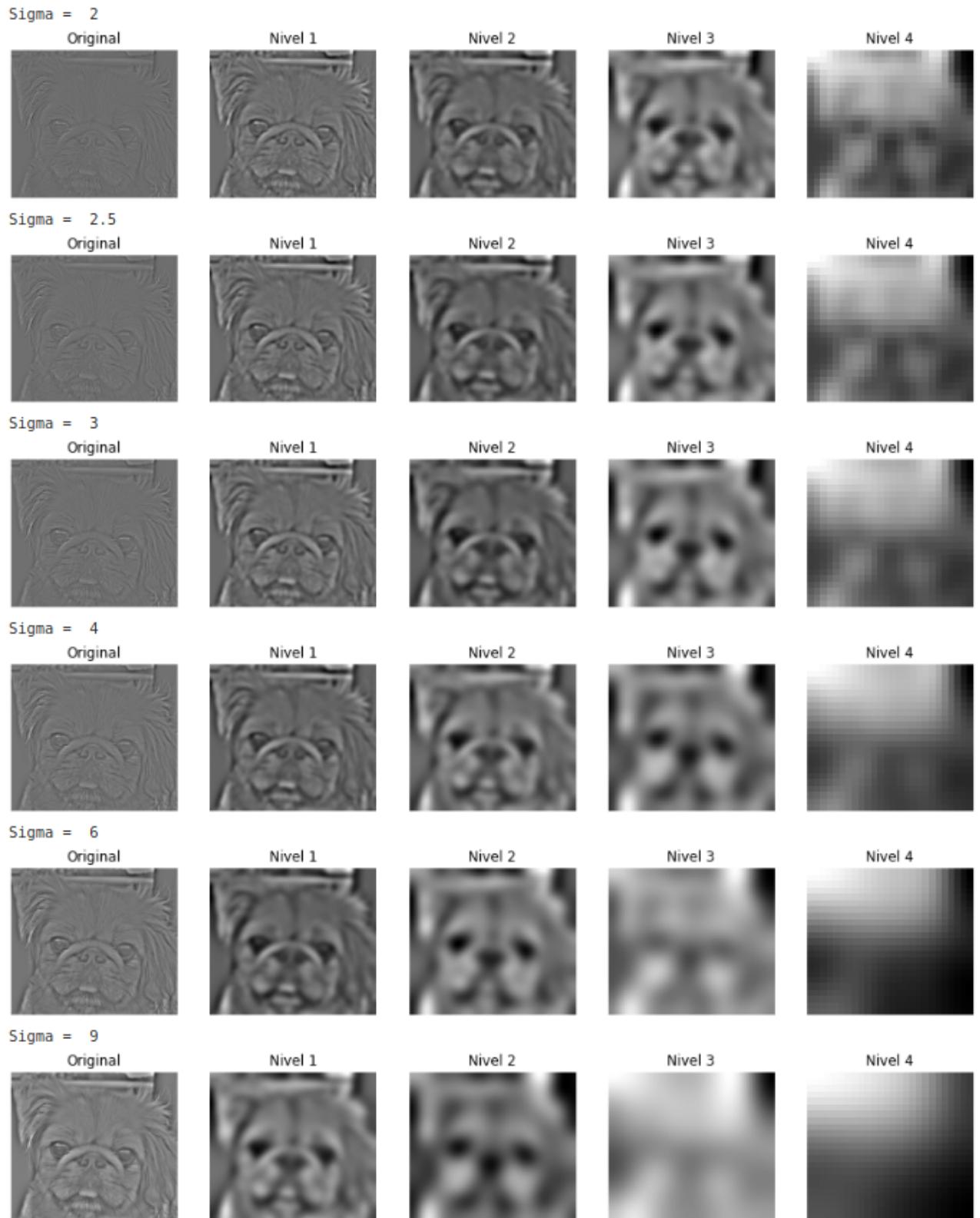


Figura 24: Pirámide laplaciana correspondiente a sigmas con valores altos

### 2.3. Ejercicio 2C

Verificar el correcto funcionamiento de la pirámide Laplaciana por medio de mostrar su capacidad para recuperar la imagen original. Se debe mostrar el error

obtenido en la aproximación, en términos de distancia Euclídea entre los niveles de gris de la imagen original y la imagen reconstruida.

En éste último apartado del ejercicio, vamos a partir de una imagen, calculamos la pirámide gaussiana y la placiana y a partir de éstas dos pirámides vamos a reconstruir la imagen tal cual la original. Para ello vamos a expandir el nivel ' $i$ ' de la pirámide laplaciana al tamaño del nivel ' $i+1$ ' y lo sumamos con el correspondiente nivel de la pirámide Gaussiana.

Tras computar la distancia euclídea tanto a la imagen original como a la reconstrucción tenemos que vale cero. Por lo que podemos concluir que las dos imágenes son iguales ya que la diferencia de cada píxel es nula.

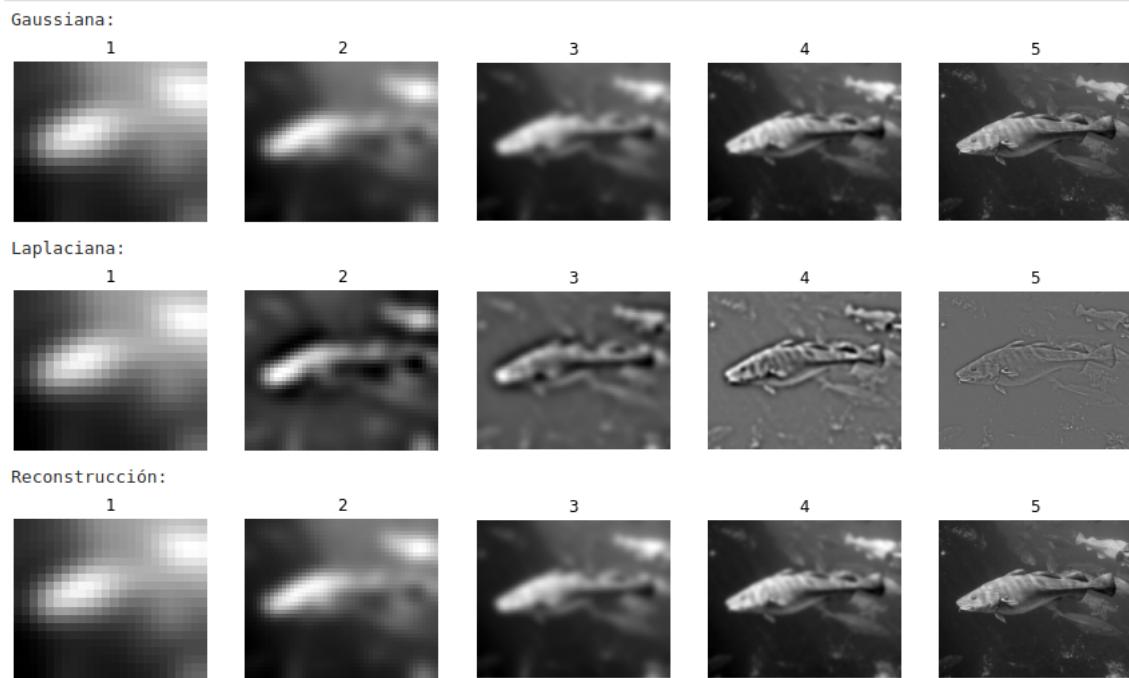


Figura 25: Reconstrucción de la imagen del pez

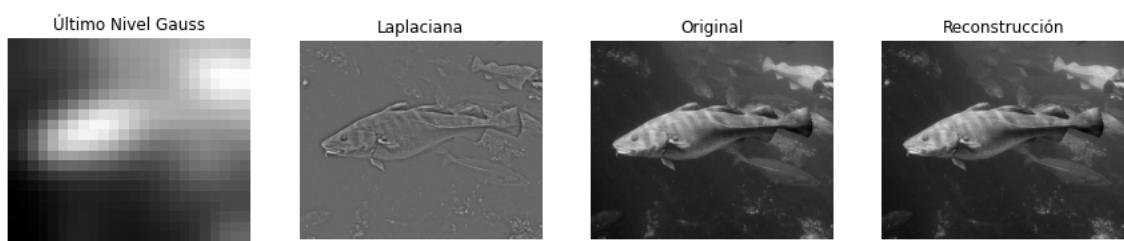


Figura 26

### 3. Bonus

#### 3.1. Bonus A

(2 puntos) 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). La imagen a usar para las bajas frecuencias y las altas respectivamente es importante para la calidad final. El valor de sigma (frecuencia de corte) más adecuado para cada imagen en cada pareja hay que encontrarlo por experimentación de prueba y error (el artículo propone una técnica)

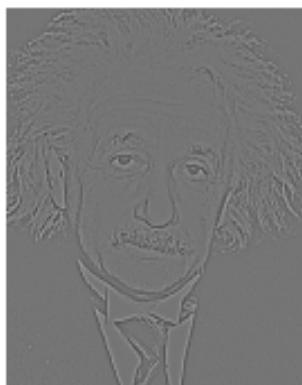
- Escribir una función que muestre las tres imágenes (alta, baja e híbrida) en una misma ventana. Recordar que las imágenes después de una convolución contienen número flotantes que pueden ser positivos y negativos.
- Realizar la composición con, al menos, 3 de las parejas de imágenes
- Construir pirámides Gaussianas de, al menos, 4 niveles con las imágenes resultado. Expliar el efecto que se observa.

Para este ejercicio hemos implementado una función que es la que realiza la hibridación. Básicamente le pasamos como argumento las imágenes junto con sus respectivos sigmas. Calculamos la imagen a bajas frecuencias de una de ellas suavizándola y la imagen de alta frecuencias será la diferencia entre ella y su suavizado. Recalcamos que la construcción de la imagen de altas frecuencias se ha hecho de esta manera, asemejándose el proceso de construcción de la laplaciana en la pirámide ya que si calculabamos directamente la imagen laplaciana usando la función que juega por dentro con las segundas derivadas el resultado no es del todo el deseado.

Para la hibridación se ha podido seguir el método que se proponía en el paper que nos daban como referencia, sin embargo, se ha elegido el método de pruebas y error. Se han realizado numerosos pruebas para cada par de imágenes de las cuales pondremos unas pocas junto con el mejor resultado. La pirámides Gaussianas se han construido con  $\sigma = 1$  y con 5 niveles.

Sigma high: 1 - Sigma low: 5

Freuencia alta



Freuencia Baja

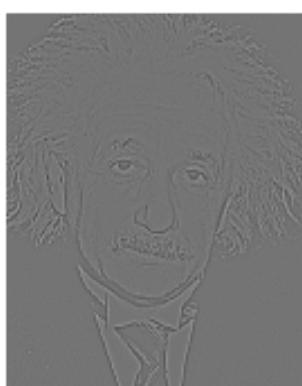


Hibridación



Sigma high: 0.8 - Sigma low: 2.5

Freuencia alta



Freuencia Baja

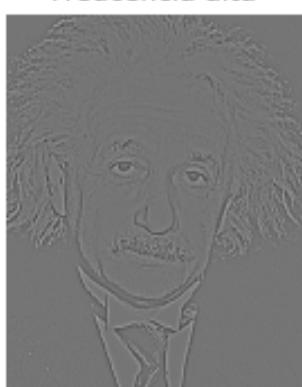


Hibridación



Sigma high: 1 - Sigma low: 3

Freuencia alta



Freuencia Baja



Hibridación



Figura 27: Experimentos para la hibridación de imágenes

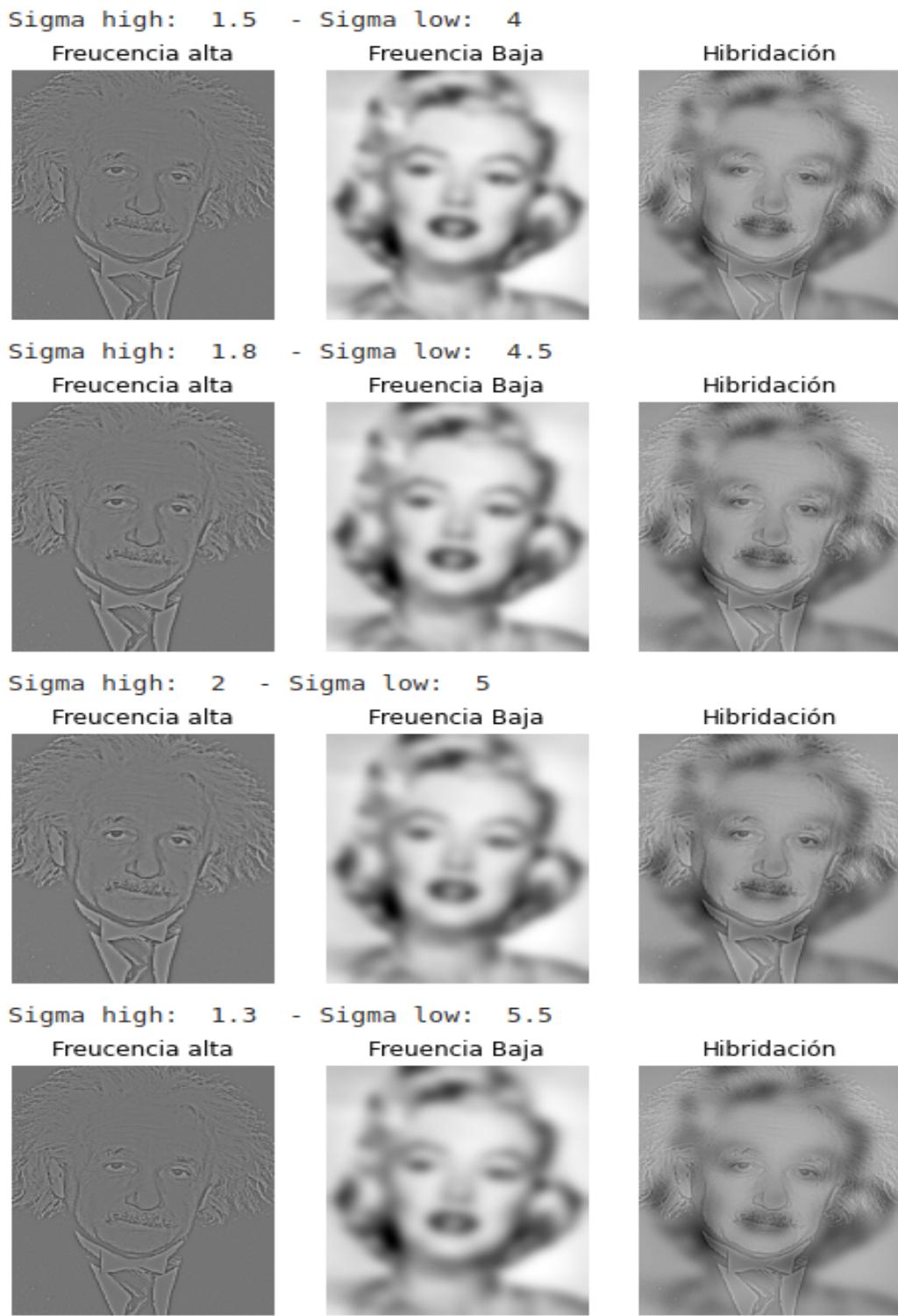


Figura 28: Experimentos para la hibridación de imágenes

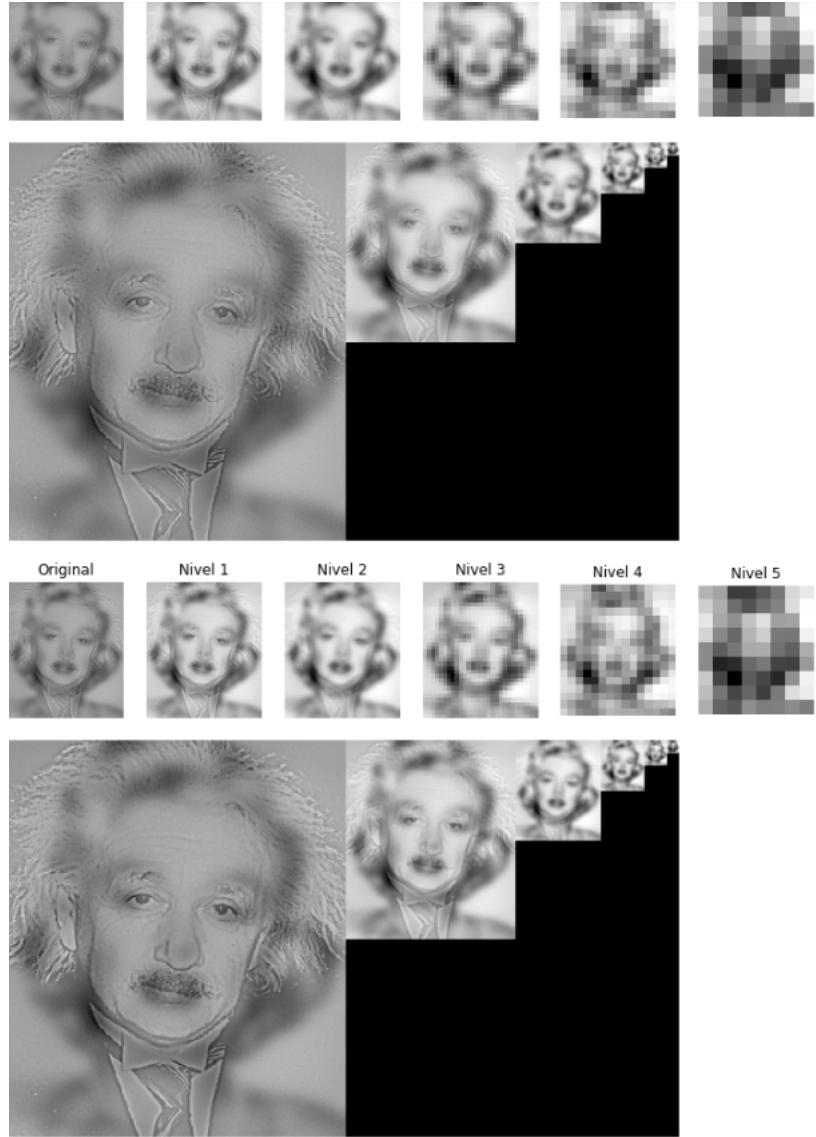


Figura 29: Mejores resultados.  $\sigma_{low} = 5,5, \sigma_{high} = 1,3$  (arriba)  $\sigma_{low} = 4,5, \sigma_{high} = 1,8$  (abajo)

En las imágenes de arriba podemos ver el proceso de hibridación de una imagen de Albert Einstein y otra imagen de Marylin Monrrow. Las Fig. 27 y Fig. 28 describen todo el proceso de experimentación. Finalmente los mejores resultados nos lo dan las Fig. 29. Podemos observar que para una frecuencia alta poco atenuada y una frecuencia baja poco emborronada, la imagen resultante no se corresponde con una buena hibridación. En este caso, debemos acentuar más las frecuencias bajas de Einstein y buscar un emborronado algo más fuerte de Marylin para obtener una buena hibridación. En la pirámide Gaussiana se puede apreciar como en los primeros niveles tenemos a Albert Einstein y en los siguientes poco a poco la figura de Einstein se va desvaneciendo para dar lugar a la cara de Marilyn. La imagen de Albert Einstein la hemos elegido para la frecuencias altas por tener una superficie más rugosa y más marcada ya que la cara de la famosa actriz es más lisa y más pulida que la del científico.

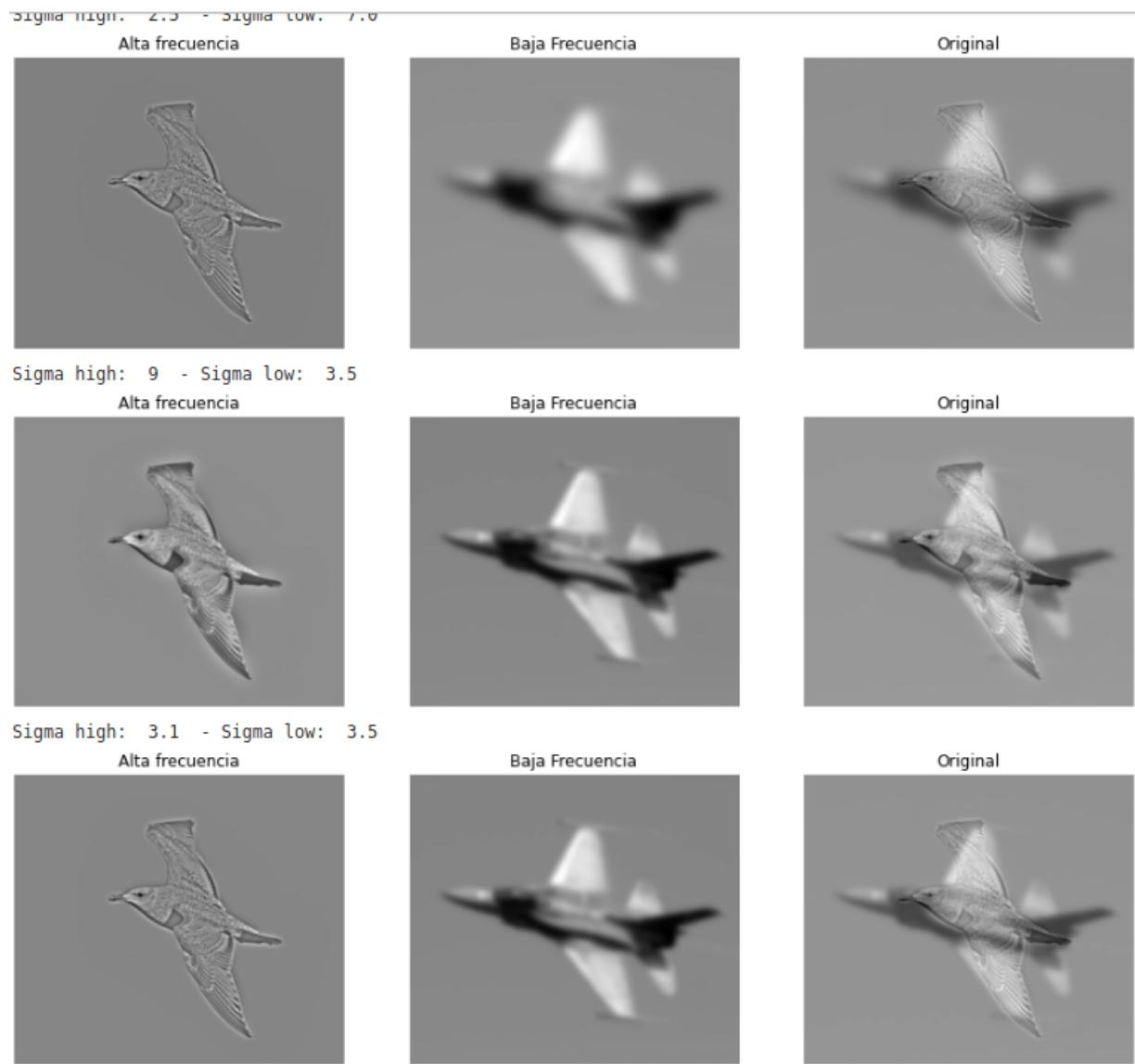


Figura 30: Experimentos para la hibridación de imágenes

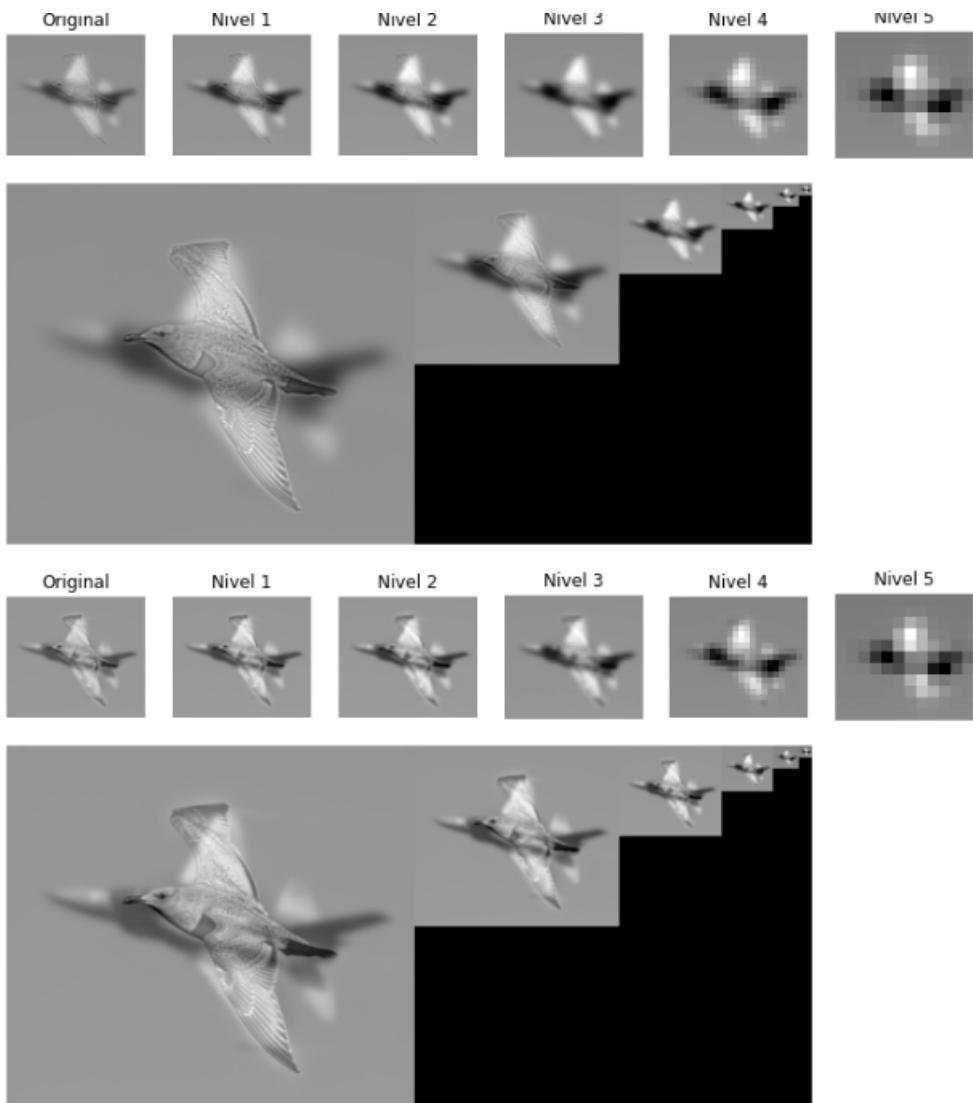


Figura 31: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

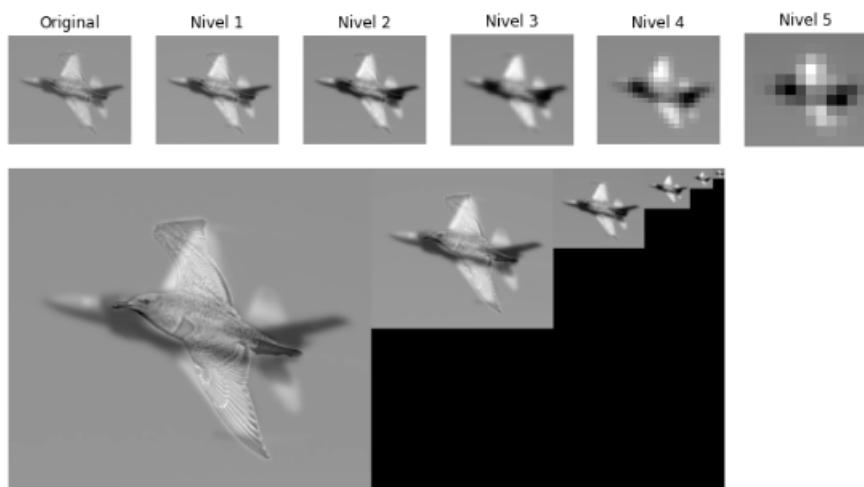


Figura 32: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En las imágenes de arriba vemos los distintos experimentos hechos para el pájaro y el avión.

Se ha elegido la imagen del pájaro para las frecuencias altas ya que presenta una superficie llena de matices con contornos más marcado que la del avión, que se ha usado para las frecuencias bajas. Debido a que la superficie del avión en la imagen es algo más alargada que la del pájaro, la hibridación no queda bien del todo. Dentro de las tres experimentaciones, observamos en el primer nivel de la pirámide gaussiana la que más aspecto tiene de pájaro es el segundo experimento por haber usado un elevado valor del sigma. El resto conservan el aspecto de pájaro pero bastante menos acentuado haciendo que sobresalga más la figura del avisón. Conforme avanzamos niveles en la pirámide Gaussiana se puede observar que la silueta el pájaro se va desvaneciendo y se nota más la figura del avión. Nos quedamos con la segunda opción por ser la que más fielmente se asemeja a una hibridación.

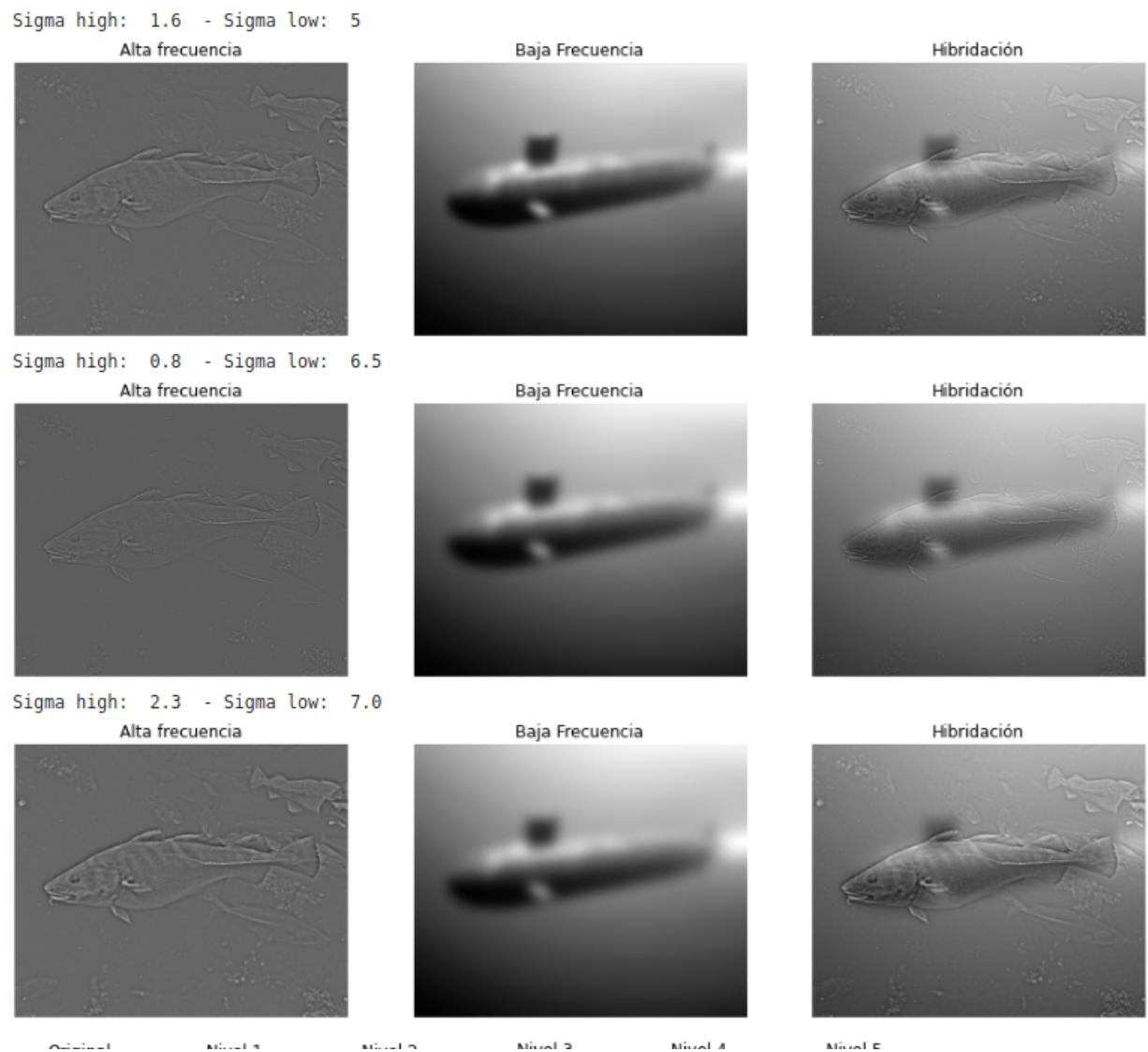


Figura 33: Experimentos de hibridaciones

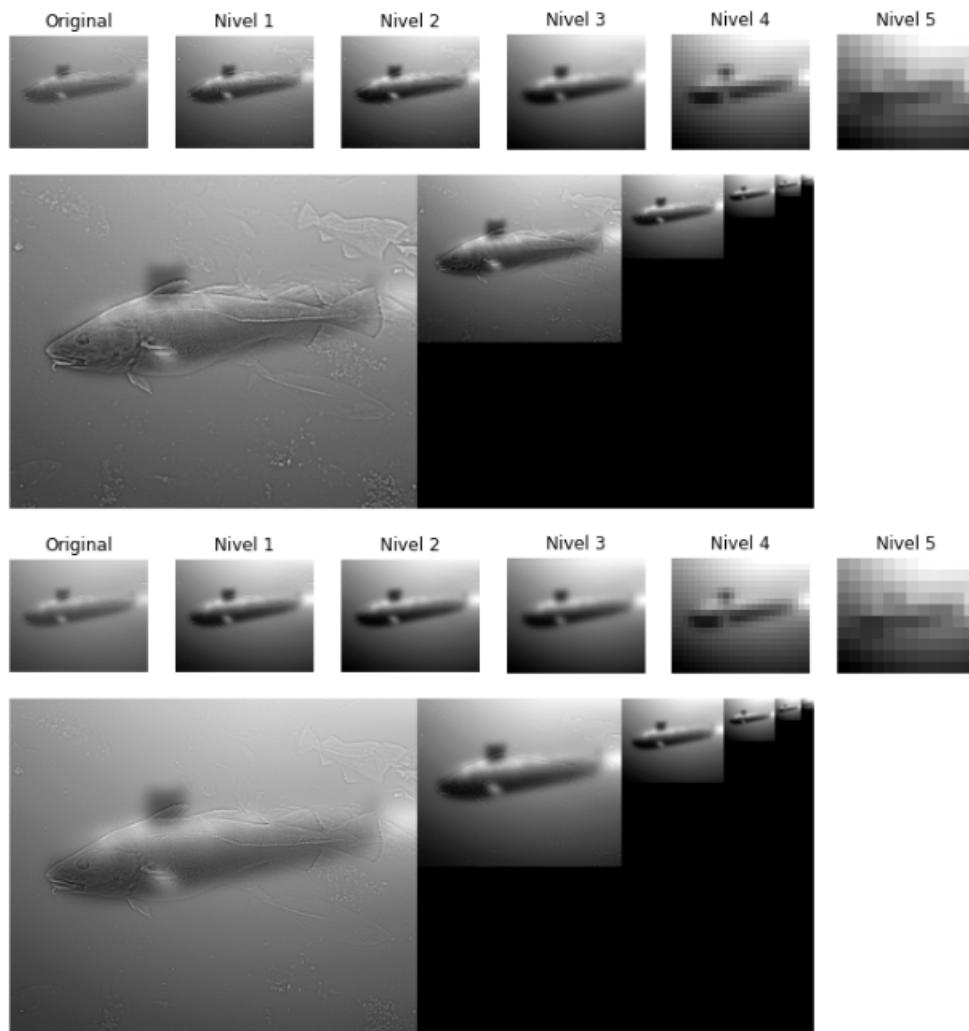


Figura 34: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

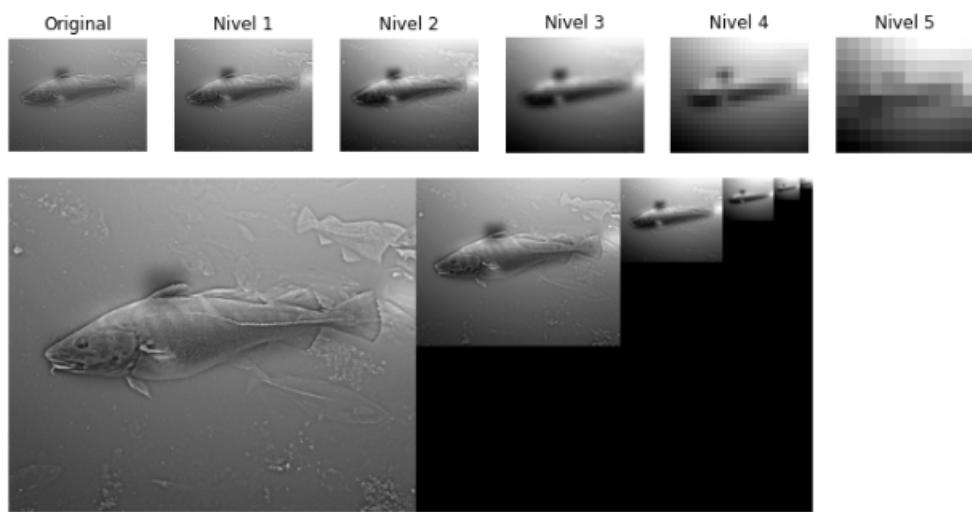


Figura 35: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En las imágenes de arriba se presenta la hibridación del submarino y del pez. Para las frecuencias altas hemos usado la imagen del pez por ser la que más detalles presenta. En la experimentación hemos probado resaltando poco, mucho y un término medio las frecuencias altas

del pez. Como resultado tenemos que para una frecuencia alta con un sigma pequeño, la imagen resultante es prácticamente el submarino sin apreciarse el pez. Para  $\sigma = 1,6$  tenemos que en la hibridación, la imagen del pez empieza a tener más presencia. Finalmente para  $\sigma = 2,3$ , el más alto, la imagen del pez cobra mucha más relevancia en la imagen. En la pirámide Gaussiana se puede observar que la peor hibridación es la de  $\sigma = 0,8$  en las frecuencias altas. La mejor es la que presenta un sigma mayor en las frecuencias altas ya que en el primer nivel de la Gaussiana es el pez quien posee todo el protagonismo, aunque la hibridación restante también es buena.

**Sigma high: 3 - Sigma low: 9**

Alta frecuencia



Baja Frecuencia



Hibridación

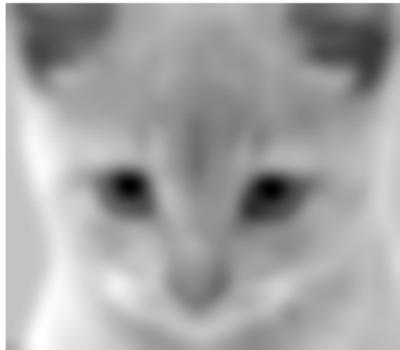


**Sigma high: 4.5 - Sigma low: 8**

Alta frecuencia



Baja Frecuencia



Hibridación



**Sigma high: 6 - Sigma low: 10**

Alta frecuencia



Baja Frecuencia



Hibridación



Figura 36: Experimentos de hibridaciones

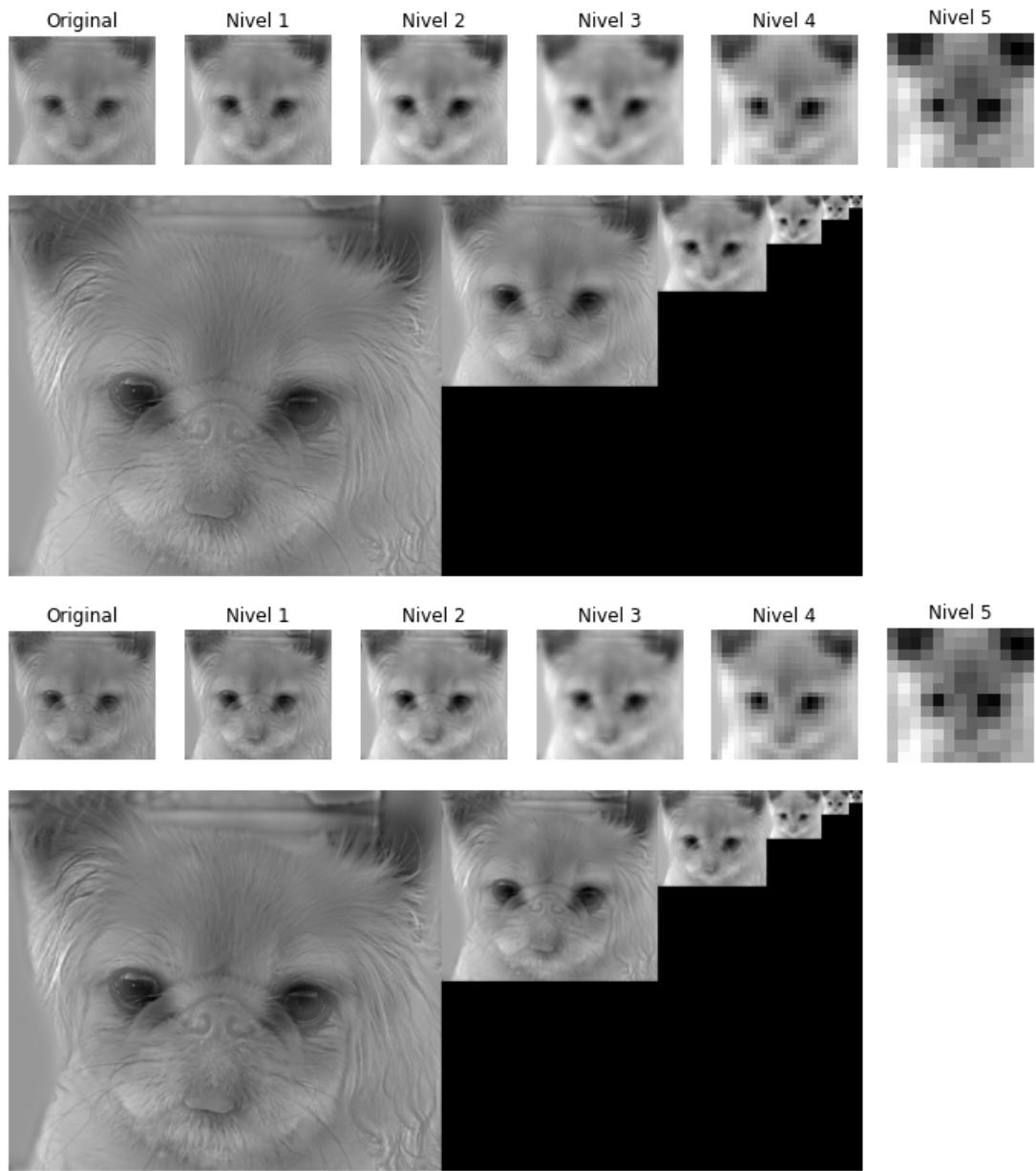


Figura 37: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

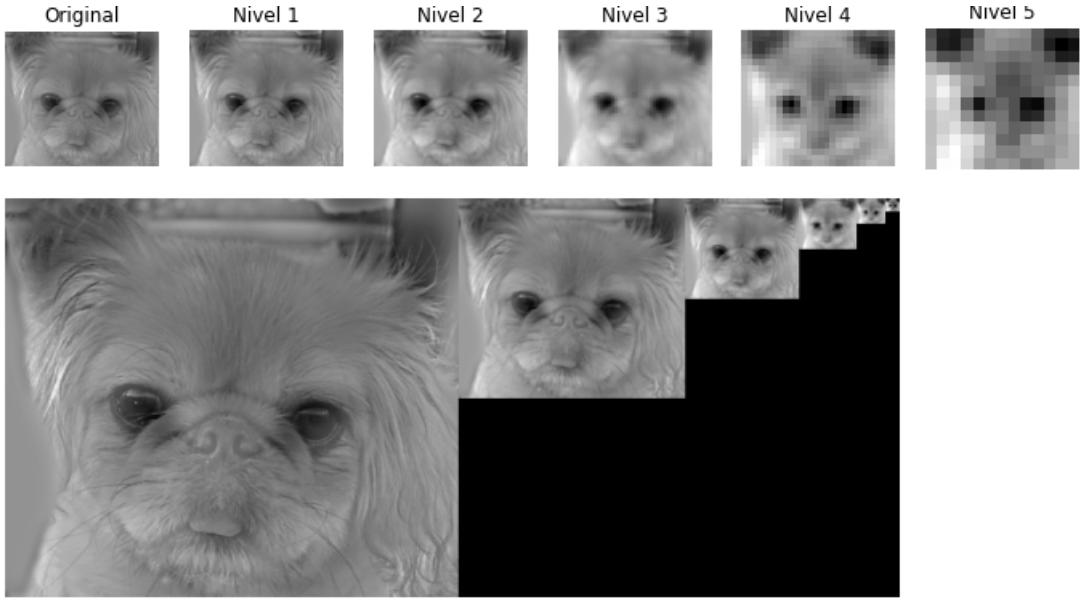


Figura 38: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En la Fig. 36 mostramos los distintos experimentos para la hibridación del perro y el gato. La imagen del perro se ha utilizado para las frecuencias altas ya que tiene unos contornos y unas características faciales mucho más marcadas que la del gato. En los tres experimentos hemos emborronado al gato con un sigma entre 8 y 10 pero el papel clave lo ha jugado el sigma de las frecuencias altas. Observamos que para  $\sigma = 3$  apenas se nota el perro en la hibridación. Al subir una unidad y media el sigma, observamos que la figura del perro toma más forma en la hibridación. También se ha probado para  $\sigma = 6$  y podemos ver que el gato ya no se aprecia en la hibridación y cobra total protagonismo el perro.

En la pirámide Gaussiana observamos que para la primera experimentación, solo se aprecia el gato. Mientras que en la segunda podemos ver en el primer nivel de la pirámide al perro y a partir del tercero ya se puede observar al gato, lo mismo ocurre en la tercera experimentación. Por tanto de entre las tres, nos quedamos con la tercera ya que el perro es más notorio en el primer nivel que en la segunda. Aún así la segunda experimentación también es buena.

Sigma high: 1.8 - Sigma low: 5

Alta frecuencia



Baja Frecuencia



Hibridación



Sigma high: 1.3 - Sigma low: 4.5

Alta frecuencia



Baja Frecuencia



Hibridación



Sigma high: 2.8 - Sigma low: 6

Alta frecuencia



Baja Frecuencia



Hibridación



Sigma high: 3.5 - Sigma low: 6

Alta frecuencia



Baja Frecuencia



Hibridación



Figura 39: Experimentos de hibridaciones

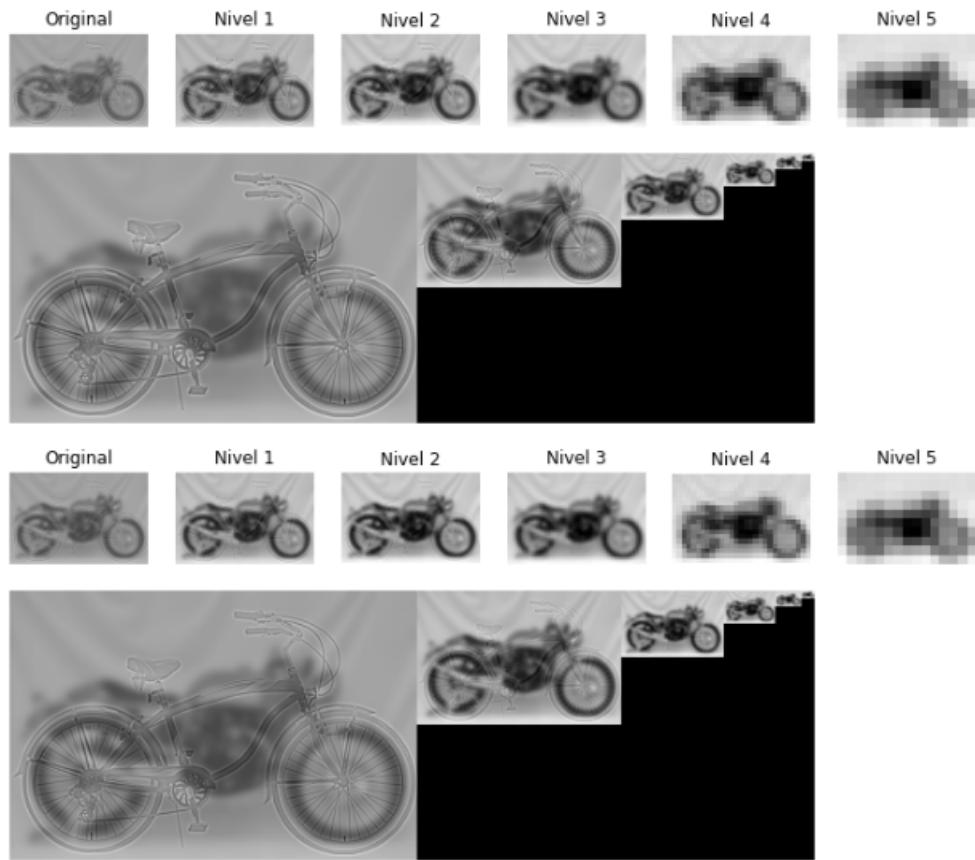


Figura 40: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

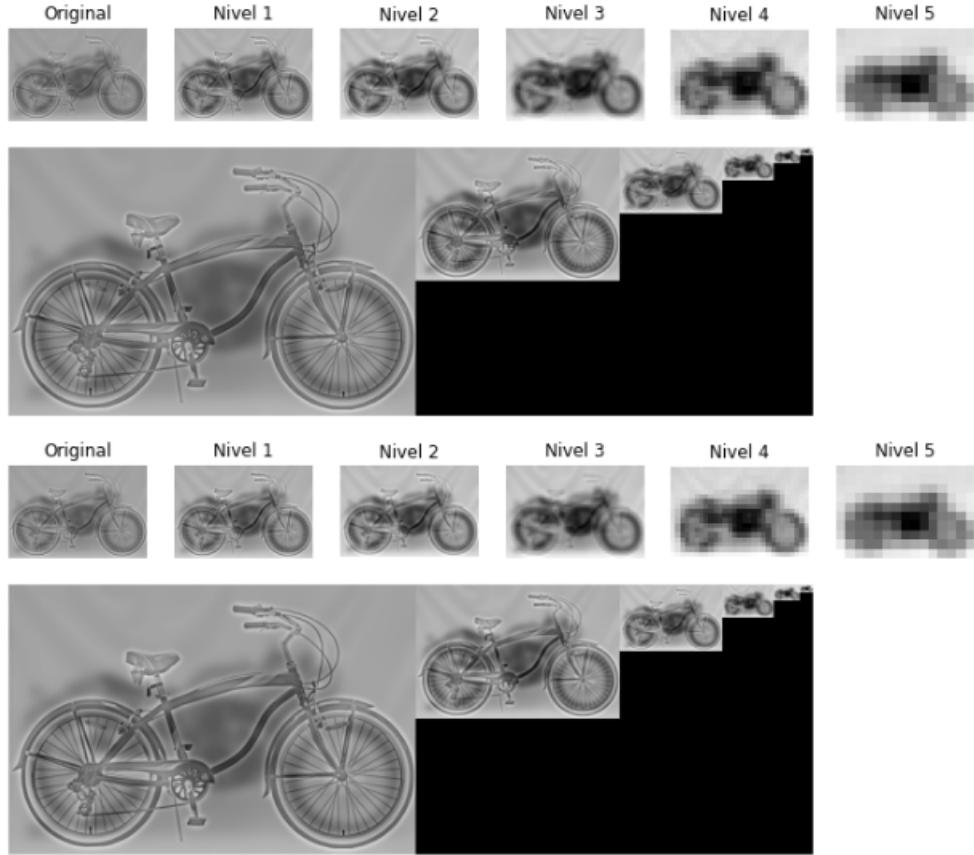


Figura 41: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

Ésta última pareja de imágenes ha costado más trabajo ya que no ha sido fácil la decisión sobre qué imagen tomar para las frecuencias bajas y para las altas. Se ha optado finalmente por utilizar para las frecuencias bajas la moto y para las altas la bicicleta. La decisión se ha llevado así ya que hemos considerado que la bicicleta tiene contornos y bordes que resaltan más que la moto. Por otra parte la hibridación tampoco ha sido fácil. Presentamos distintos experimentos para los cuales hemos el sigma de las frecuencias bajas entre 4.5 y 6 y el de las frecuencias altas entre 1.3 y 3.5. Notamos que para un sigma bajo en las frecuencias bajas la bicicleta apenas se aprecia en la hibridación, por ese motivo hemos ido poco a poco aumentando el sigma hasta quedarnos con los dos últimos experimentos.

En cuanto a las pirámides Gaussianas, nos hacen ver que las dos primeras nos ofrecen una hibridación un poco pésima por no resaltar la bicicleta. Los dos últimos experimentos son los mejores ya que la bicicleta en el primer nivel es la que predomina y a partir del tercero ya se empieza a notar la moto. Cualquiera de las dos últimas imágenes de la moto nos serviría de hibridación.

### 3.2. Bonus B

Es este apartado presentaremos todas las hibridaciones hechas anteriormente pero ha color. Se han tenido que probar otros experimentos con otros valores de sigmas diferentes para estas hibridaciones ya que el color influye bastante. Además se han usado las mismas imágenes para altas y baja frecuencias que en la versión en gris. Por otra parte, para poder realizar éste apartado se ha tenido que modificar algunas funciones como por ejemplo el padding la cual consiste

en hacer el padding a los tres canales en color por separado, añadir un nuevo eje a cada matriz para finalmente concatenarlas. El método para concatenar las imágenes en forma de pirámide también ha recibido una pequeña modificación, aunque éste último método carece de interés por eso no se darán más detalles.

En las figuras de abajo presentamos la hibridación a color del gato y el perro. El emborronado del gato ha sido elevado, con un sigma de 9 o 10 y se observa que si no se destaca las frecuencias altas del perro como en el primer experimento, la silueta no se marca tanto como se desearía. Por ese motivo se ha acentuado más en los siguientes experimentos. En el segundo, vemos que quizás el perro esté demasiado marcado en la hibridación y por ese motivo se ha disminuido un poco el sigma en el último.

Las pirámides Gaussians nos dicen que el primer experimento es una mala aproximación. En el segundo experimento se nota el gato a partir del cuarto nivel y en el último experimento el gato se aprecia algo más en el tercer nivel. De entre los tres experimentos el segundo quizás se necesite visualizar algo más de lejos la imagen para apreciar el gato que en el tercero, por ese motivo, nos quedamos con la tercera prueba. (Aunque ambas son igualmente válidas)

Sigma high: 3 - Sigma low: 9

Alta frecuencia



Baja Frecuencia



Hibridación



Sigma high: 10 - Sigma low: 10

Alta frecuencia



Baja Frecuencia

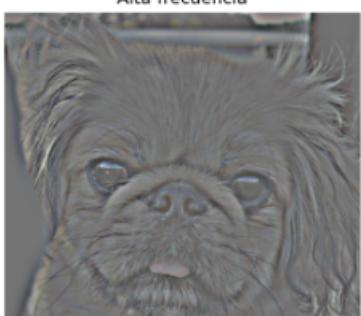


Hibridación



Sigma high: 6 - Sigma low: 10

Alta frecuencia



Baja Frecuencia



Hibridación



Figura 42: Experimentos de hibridaciones

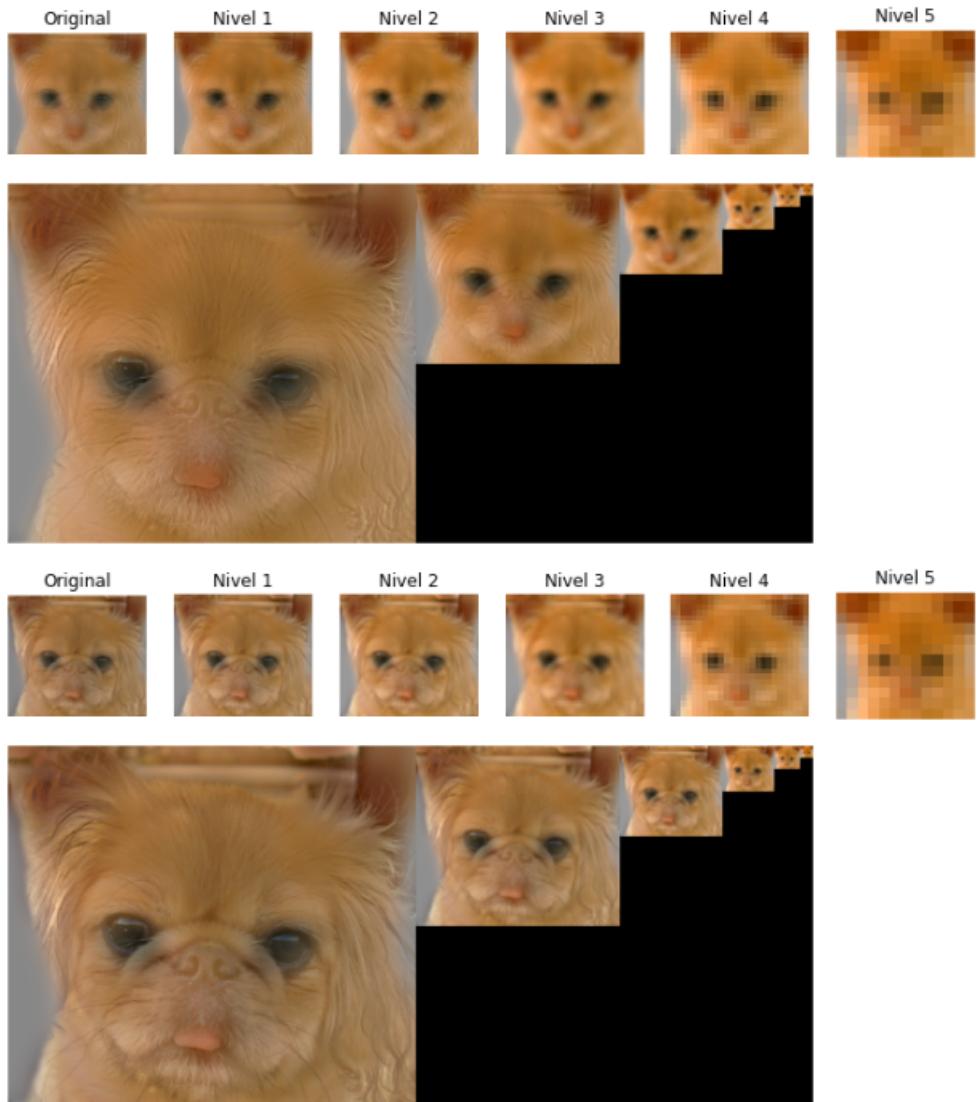


Figura 43: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

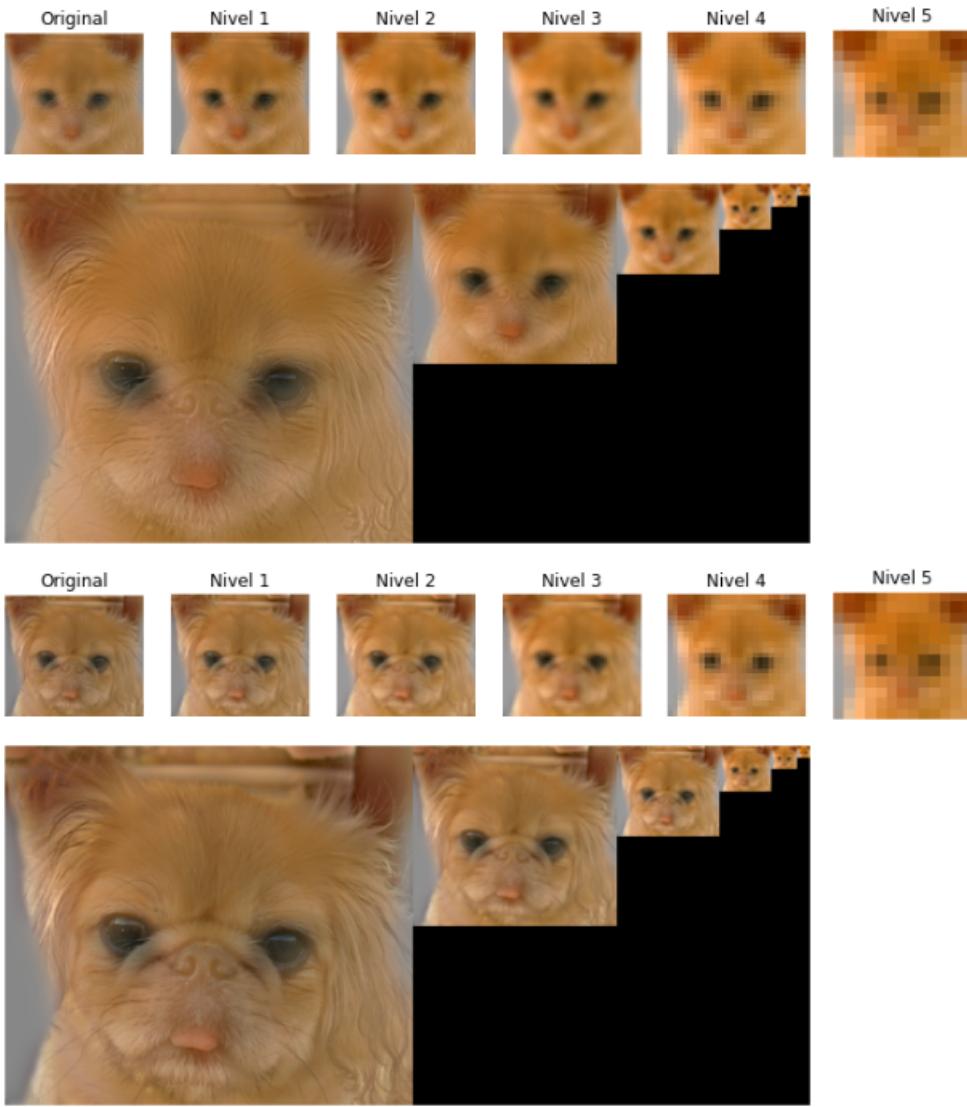


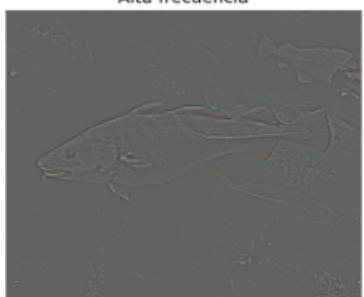
Figura 44: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En las imágenes de abajo presentamos la hibridación a color del pez y el submarino. En el primer experimento donde el pez prenta unas frecuencias altas poco acentuadas observamos que en la imagen híbrida apenas es apreciable. Por lo que se decidió a aumentar esas frecuencias en el pez y a emborronar más la imagen del submarino. Como resultado tenemos el experimento número dos donde tenemos una hibridación bastante mejor que en la primera. En el tercer experimento queríamos que el submarino resaltase más y por ello se emborronó menos y se disminuyeron un poco las frecuencias altas de la imagen del pez.

La pirámide Gaussiana nos dice que el primer experimento es una mala aproximación. El segundo experimento es el mejor ya que el pez es más nítido en el primer nivel y a partir del tercero se aprecia la imagen del submarino. El tercer experimento es mejor que el primero pero peor que el segundo ya que el pez se nota menos.

Sigma high: 1.3 - Sigma low: 5

Alta frecuencia



Baja Frecuencia



Hibridación



Sigma high: 2.8 - Sigma low: 6.5

Alta frecuencia



Baja Frecuencia

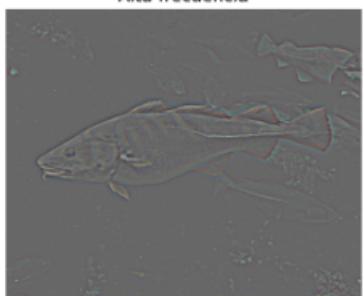


Hibridación



Sigma high: 2 - Sigma low: 4.0

Alta frecuencia



Baja Frecuencia



Hibridación



Figura 45: Experimentos de hibridaciones

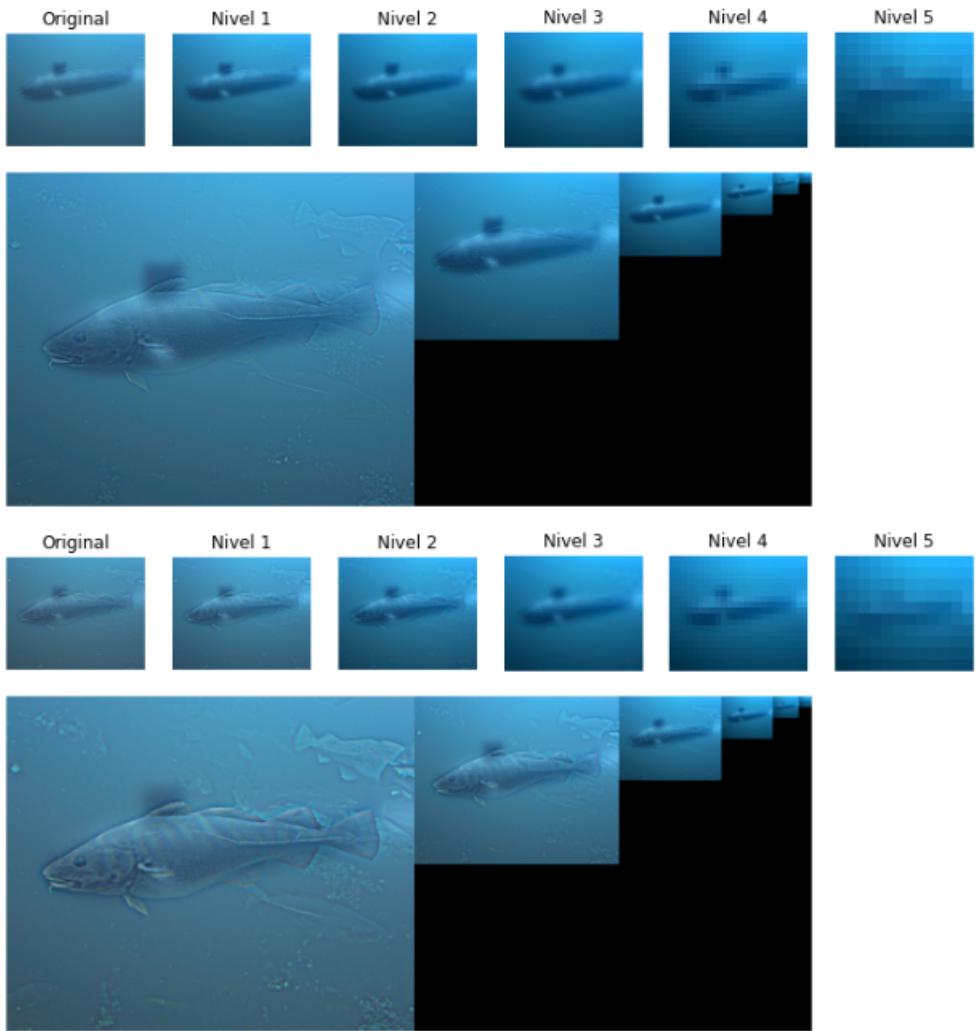


Figura 46: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

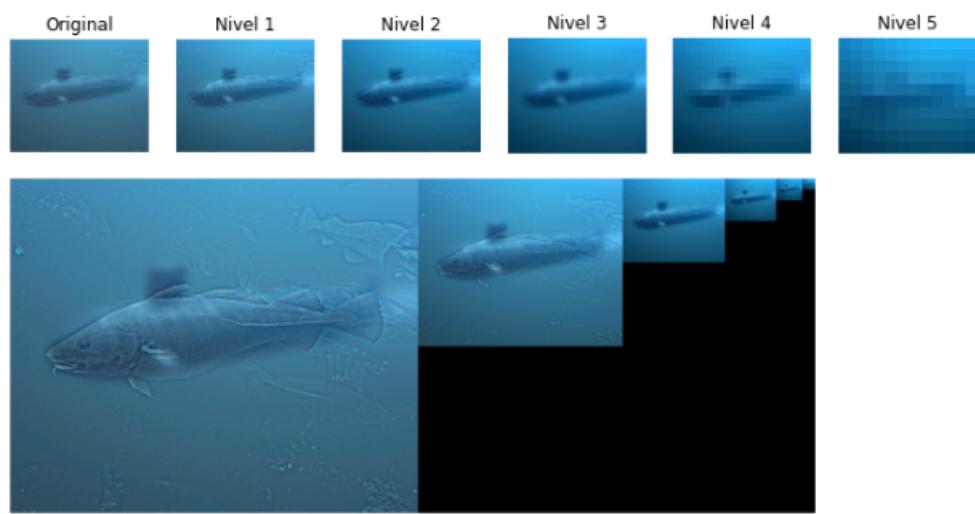


Figura 47: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En las imágenes de abajo presentamos la hibridación a color del avión y del pájaro. Ésta hibridación ha costado algo más de trabajo. Observamos que en el primer experimento hemos

tratado de hacer una hibridación acentuando poco las frecuencias altas del pájaro y emborronando de forma pobre el avión. De resultado nos da una imagen en la que solo se aprecia el avión. En el segundo experimento se ha tratado de paliar lo anterior y hacer que el pájaro se viera algo más, por ese motivo se ha aumentado el sigma tanto en las frecuencias altas como en las bajas para que destaque. El resultado sigue saliendo malo, aunque no tan nefasto como el primer caso. A continuación se ha incrementado el emborronado y acentuado las frecuencias altas. Como resultado ya tenemos una primera aproximación a una hibridación. En el último experimento hemos aumentado mucho las frecuencias altas y para paliar un poco el peso del pájaro en la hibridación se ha emborronado un poco menos el avión. Como resultado tenemos una imagen en la que el pájaro es el principal protagonista.

Fijándonos en las pirámides Gaussianas para comprobar la bondad de la hibridación, observamos que el primer y el segundo experimento son unas malas aproximaciones. El tercer experimento pese a que el avión se sigue notando en el primer nivel, tenemos que es el pájaro quien predomina. A partir del tercer nivel el pájaro deja de apreciarse. Finalmente en el último experimento tenemos que en el primer nivel de la gaussiana la imagen que más se ve es la del pájaro, en el segundo nivel ocurre igual. En el tercer nivel aún no se tiene del todo claro, aunque el avión empieza a resaltar más. Finalmente es el cuarto nivel cuando se puede apreciar más o menos un avión. El problema del cuarto experimento es que el pájaro desaparece cuando la imagen se ve de muy lejos, sin embargo en el tercer experimento no hace falta situarla muy lejos para empezar a ver el avión. Por este motivo nos quedamos con el tercer experimento como hibridación por mantener un equilibrio entre la hibridación y la distancia.

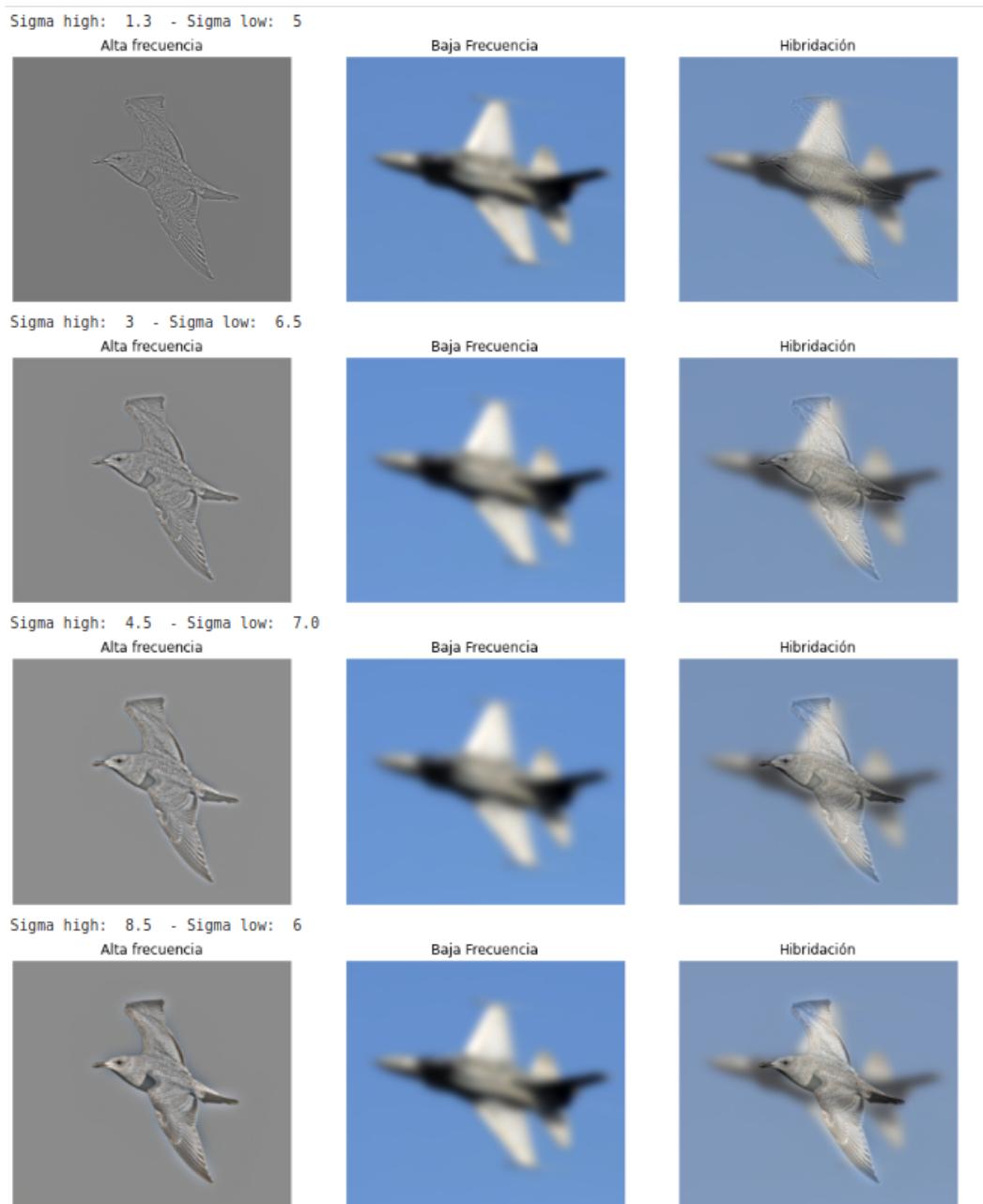


Figura 48: Experimentos de hibridaciones

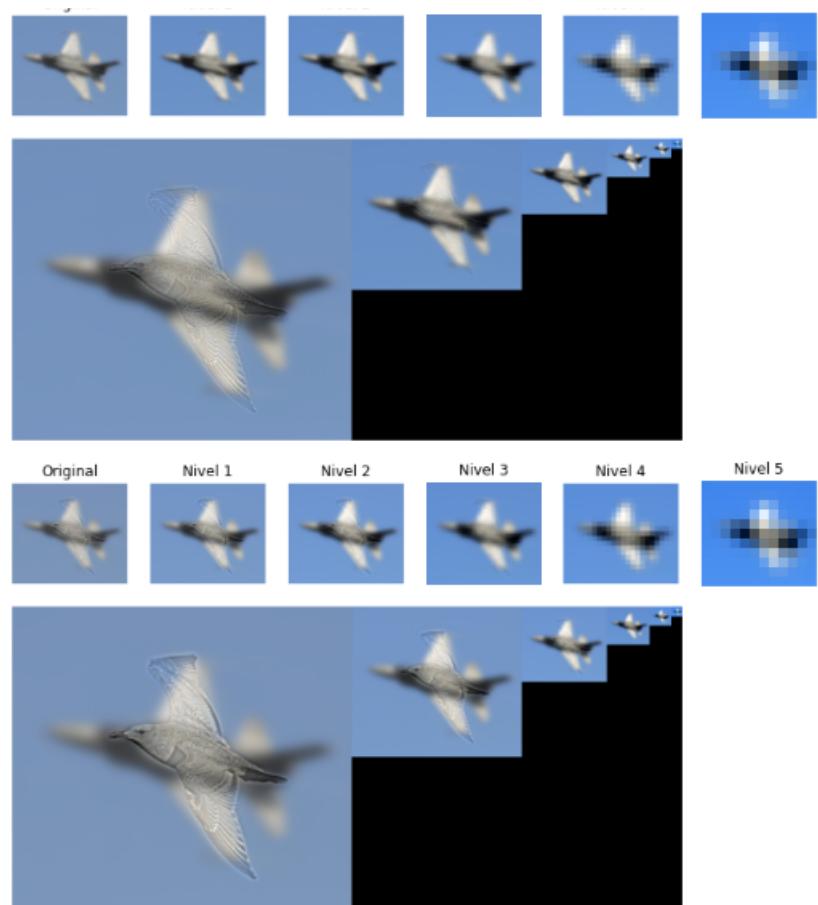


Figura 49: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

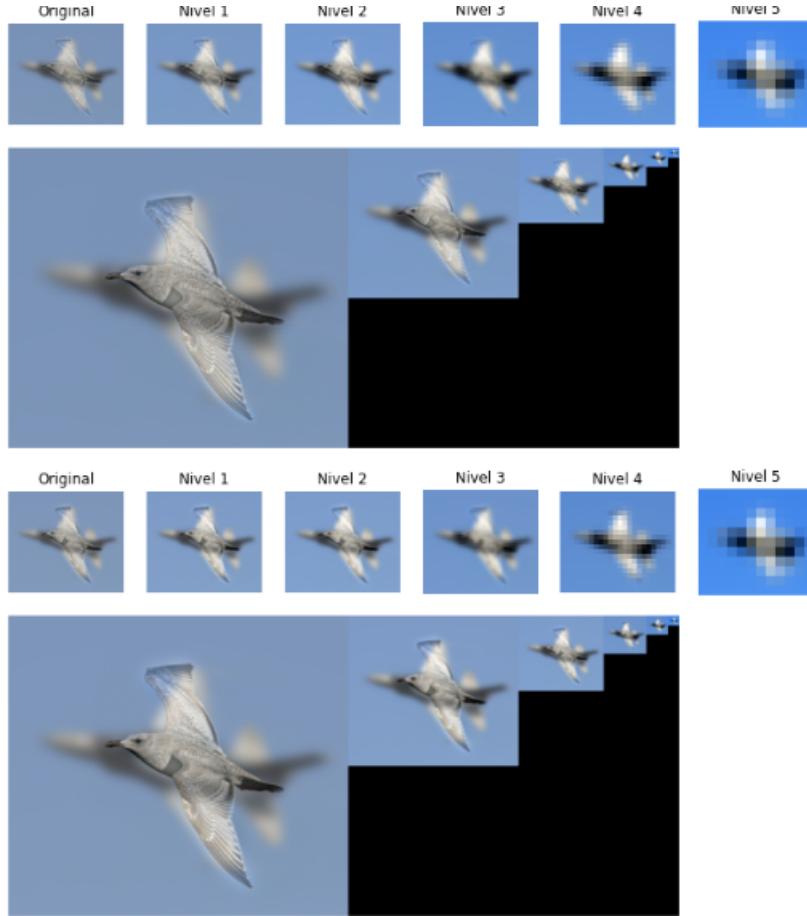


Figura 50: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

En las imágenes de abajo se muestran el proceso de construcción de la hibridación a color de la bicicleta y la moto. Ésta hibridación al igual que su versión en gris nos ha costado bastante trabajo realizarla. Empezamos con un primer experimento en el que el emborronamiento de la moto es muy marcado y las frecuencias altas de la bicicletas muy tenues. Como resultado tenemos una hibridación pésima. Tras ello disminuimos el emborronamiento de la moto y aumentamos muy poco las frecuencias altas. La hibridación resultante también es pésima. El tercer experimento es muy parecido al segundo por lo que no lo vamos a comentar. Finalmente el cuarto experimento aumentamos mucho las frecuencias altas de la bicicleta y emborronamos la moto con un sigma de 5 unidades. Como resultados tenemos que en la hibridación la bicicleta resalta bastante más.

En la pirámides Gaussianas podemos comprobar que los dos primeros experimentos son fallidos. Que el tercer experimento empieza a acercarse más a lo que queremos pero sigue siendo un fracaso y que el cuarto experimento es el mejor. De hecho se aprecia la bicicleta hasta el tercer nivel a partir del cual es la moto quien tiene el protagonismo.

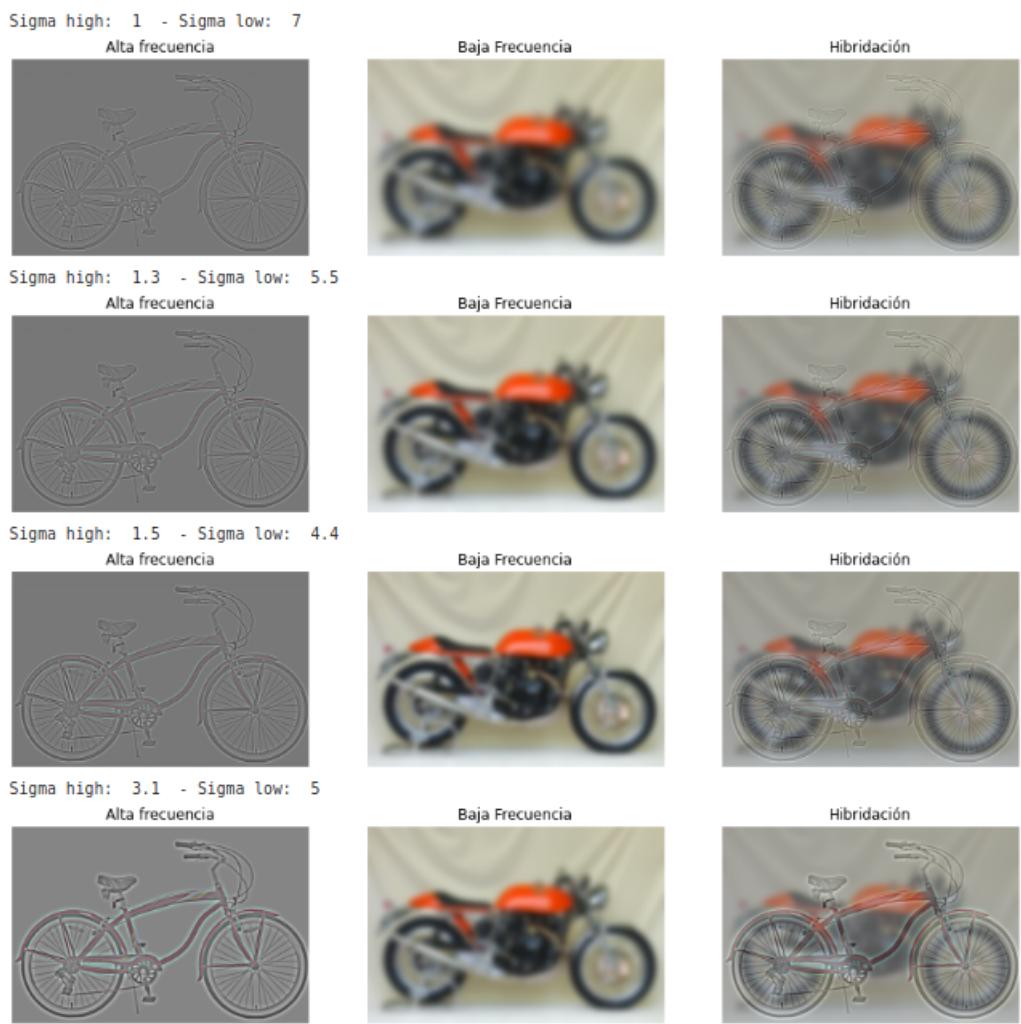


Figura 51: Experimentos de hibridaciones

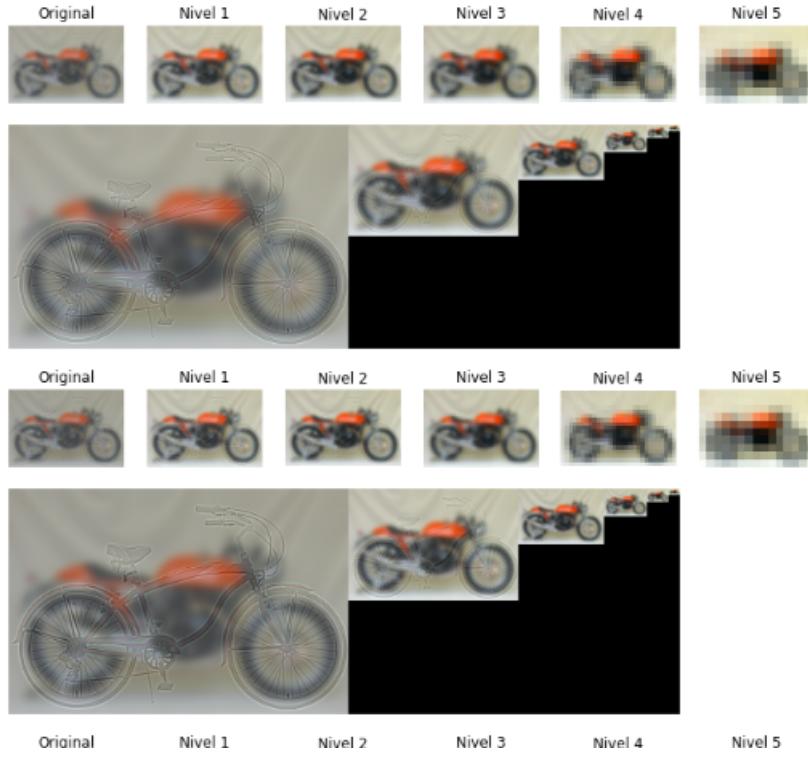


Figura 52: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

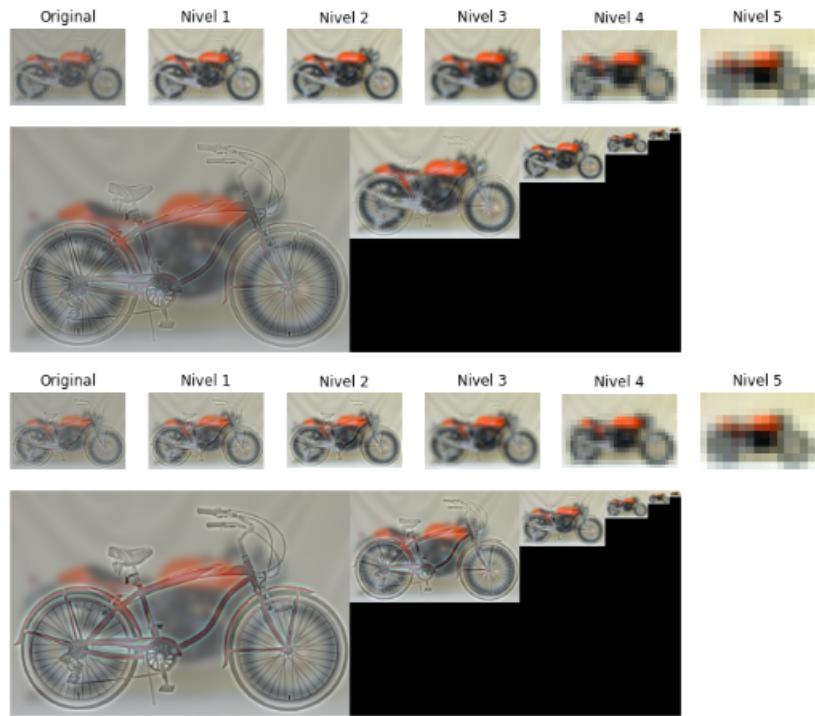


Figura 53: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

Finalmente presentamos la hibridación de Albert Einstein y Marilyin Monrrow. Aunque la hibridación ya se ha hecho, notamos que ahora las imágenes son tricanales, a diferencias de las imágenes monocanales del apartado anterior. Básicamente la forma de proceder ha sido como el resto de parejas, a base de prueba y error. Todos los experimentos producen buenas hibrida-

ciones a excepción del último, ya que en su pirámide Gaussiana apenas se aprecia la figura de Marylin debido a su mayor emborronamiento y a las acentuadas altas frecuencias de la imagen de Albert Einstein.

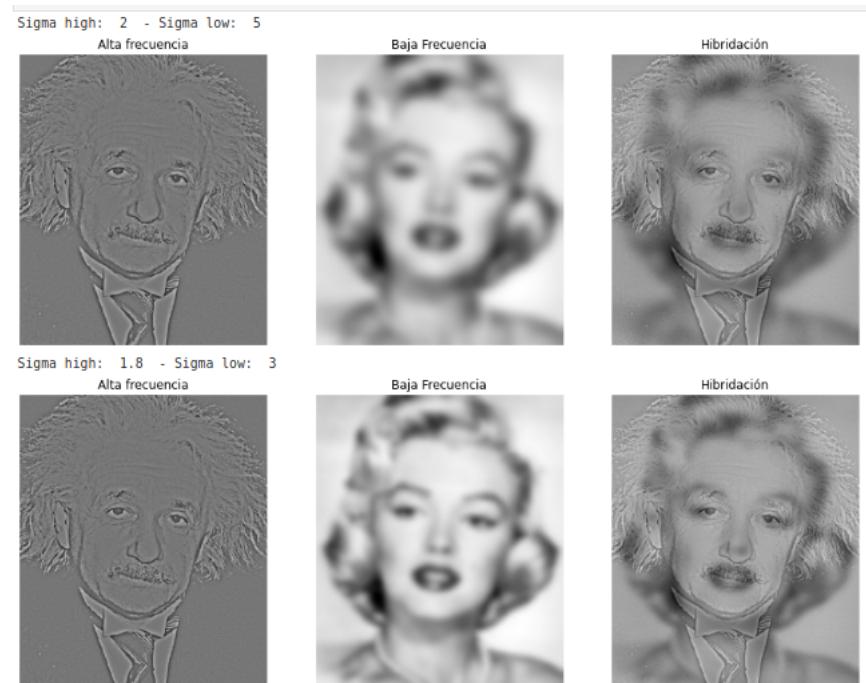


Figura 54: Experimentos de hibridaciones

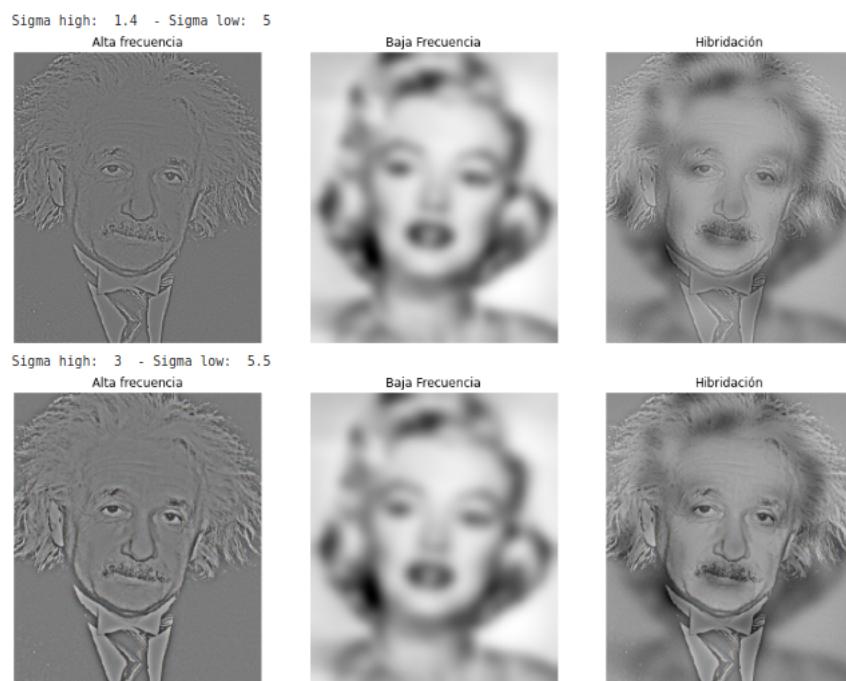


Figura 55: Experimentos de hibridaciones

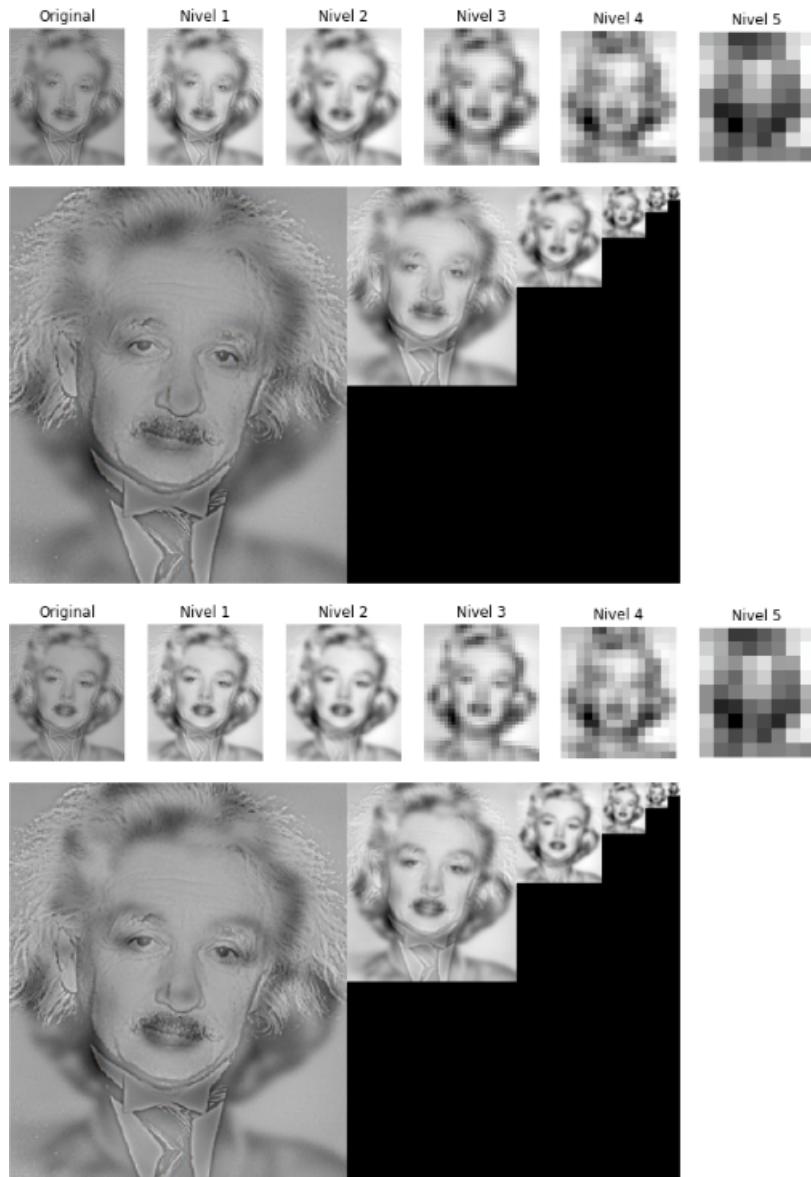


Figura 56: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

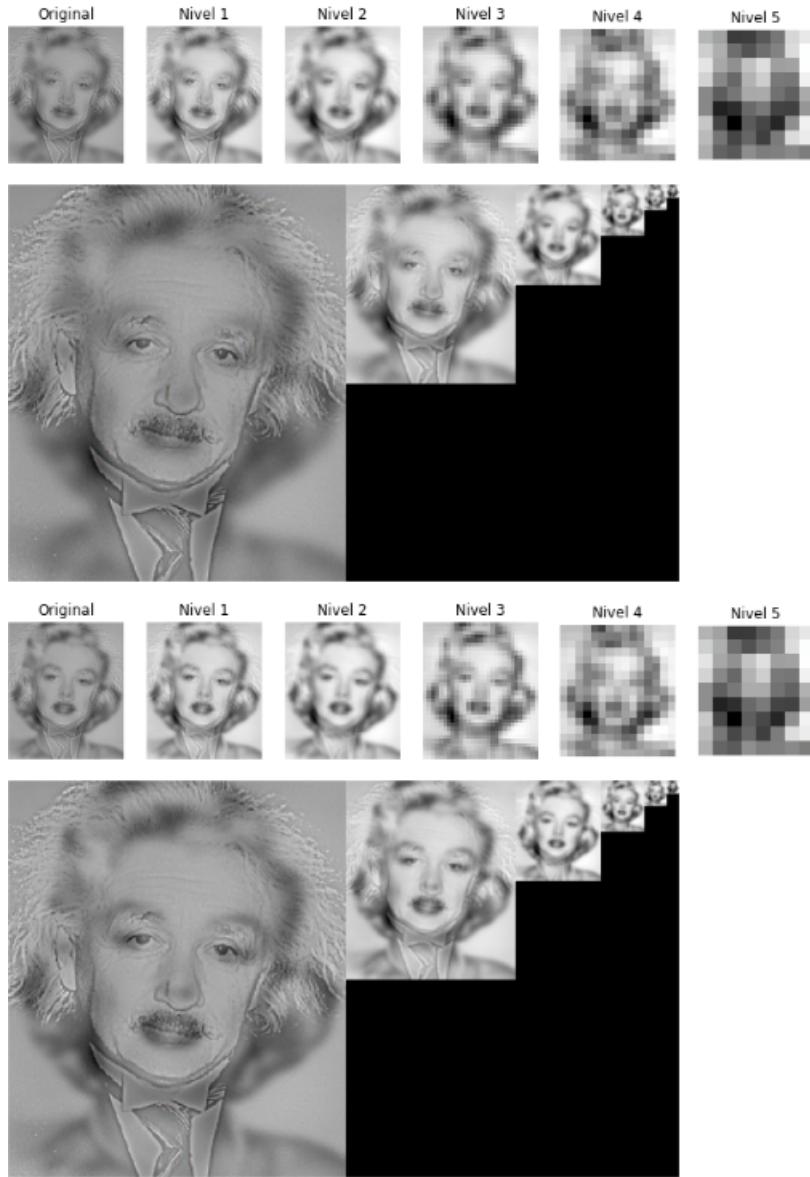


Figura 57: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

### 3.3. Bonus C

Finalmente, para acabar este apartado hemos realizado una hibridación a color de varias parejas de imágenes y vamos a estudiar brevemente por qué los resultados han salido satisfactorios o no. En primer lugar presentaremos una hibridación de nuestro querido profesor de teoría, Nicolás Pérez, junto con nuestra también querida rectora, Pilar Aranda. Tras ello se realizará la hibridación entre el famoso cuadro de Edvard Munch, El Grito, y el famosísimo cuadro del gran Leonardo Da Vinci, La Monalisa. Por último, se realizará la hibridación de una zapatilla y un pie.

Empezamos estudiando la hibridación de Nicolás Pérez y Pilar Aranda. En un primer experimento se optó por dejar a Pilar con las frecuencias altas y a Nicolás con las bajas, pero los resultados eran muy malos por lo que se permutaron estos papeles y al final Nicolás se ha quedado con las altas y Pilar con las bajas.

Como se observa en la imagen de abajo tenemos que a Nicolás le hemos puesto unas frecuencias altas elevadas ya que la figura de Pilar presenta fuertes contrastes de Colores y en la

hibridación lo que más llama a la vista son los colores de Pilar más que el contorno de Nicolás. Por otra parte, hemos emborronado mucho la figura de Pilar para paliar lo anteriormente explicado. Aún así hemos tenido que tener cuidado ya que en la pirámide Gaussiana el objetivo no es que Nicolás tape a Pilar, sino guardar un equilibrio entre distancia e hibridación. A partir del cuarto nivel es cuando Pilar empieza a apreciarse en la imagen ya que los contornos de Nicolás se van yendo.

A modo didáctico, hemos aprendido que hay que tener cuidado con las hibridaciones con grandes contrastes de colores ya que puede no salir del todo como se desea.

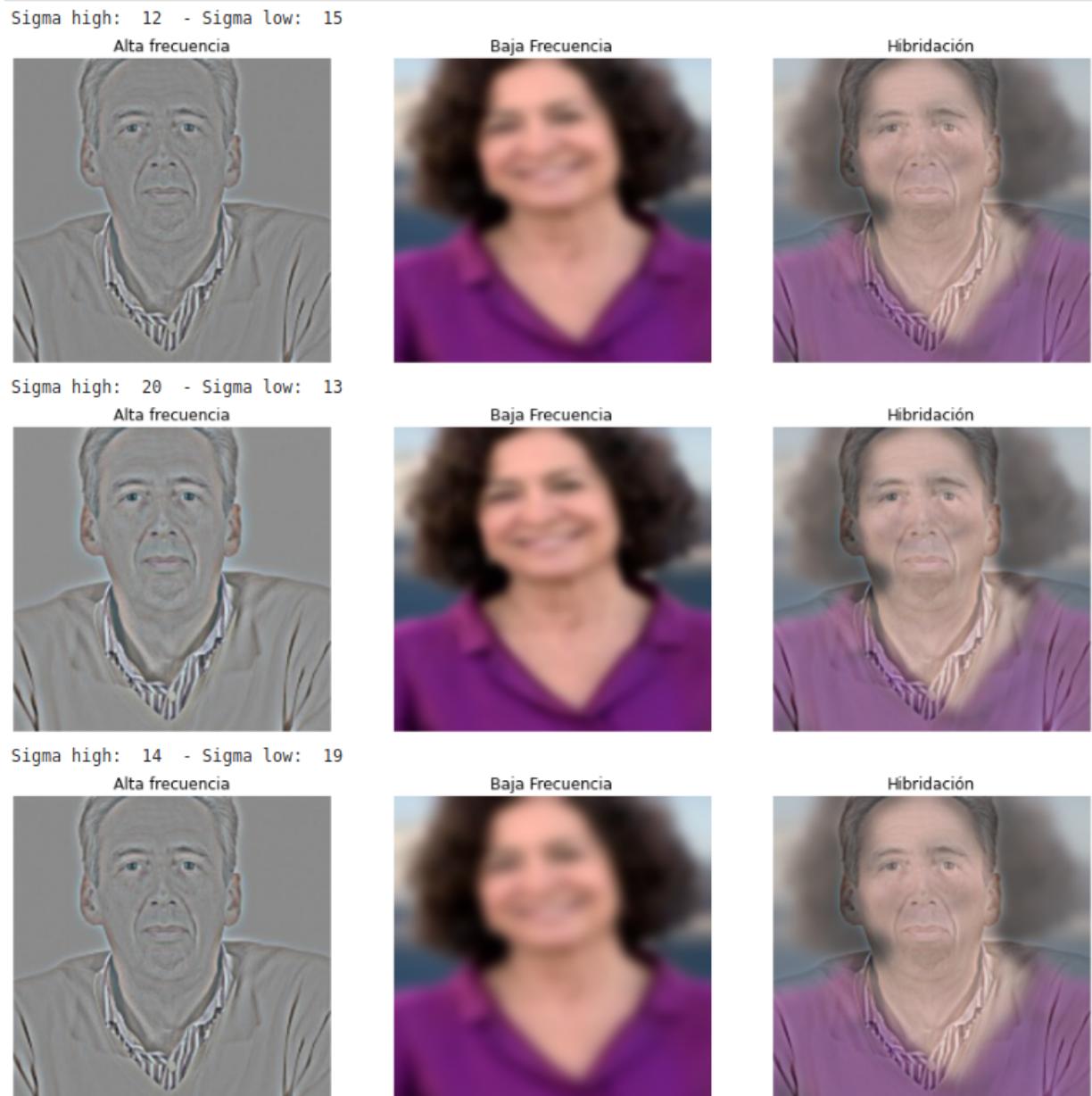


Figura 58: Experimentos realizados



Figura 59: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana



Figura 60: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

Presentamos ahora la hibridación de dos famosos cuadros. De esta hibridación se ha aprendido bastante ya que el método que usábamos para intuir que imagen tiene las frecuencias altas y cual las bajas no es exacto. En primera instancia la imagen del grito es la que más contornos y detalles presenta por lo que se ha elegido para las frecuencias altas y el otro cuadro para las bajas. Sin embargo, en la experimentación hemos comprobado que no ha sido tan buena idea y se ha permutado los papeles de las imágenes. Por lo que ahora La Monalisa tiene las frecuencias altas y El Grito las bajas.

Las imágenes de abajo corresponden al cuadro del Grito con las frecuencias altas y a La Monalisa con las bajas. Observamos que en el primer experimento, la hibridación es una mezcla extraña de las dos, de hecho, parece más bien un extraterrestre que una hibridación. El problema está en que La Monalisa resalta más que El Grito que ya de por sí está poco acentuado. En el siguiente experimento se le da más peso a las frecuencias altas y se decremente el nivel de emborronamiento en las bajas. El resultado tampoco no es tan bueno como queríamos. Finalmente, emborronamos bastante las frecuencias bajas y ecentuamos mucho las altas. El resultado es una hibridación en el que el Grito destaca más que La Monalisa. Hemos logrado lo que queríamos, sin embargo, en la pirámide Gaussiana La Monalisa se aprecia en los últimos niveles y el cambio de una imagen a otra no guarda un buen equilibrio con la distancia.

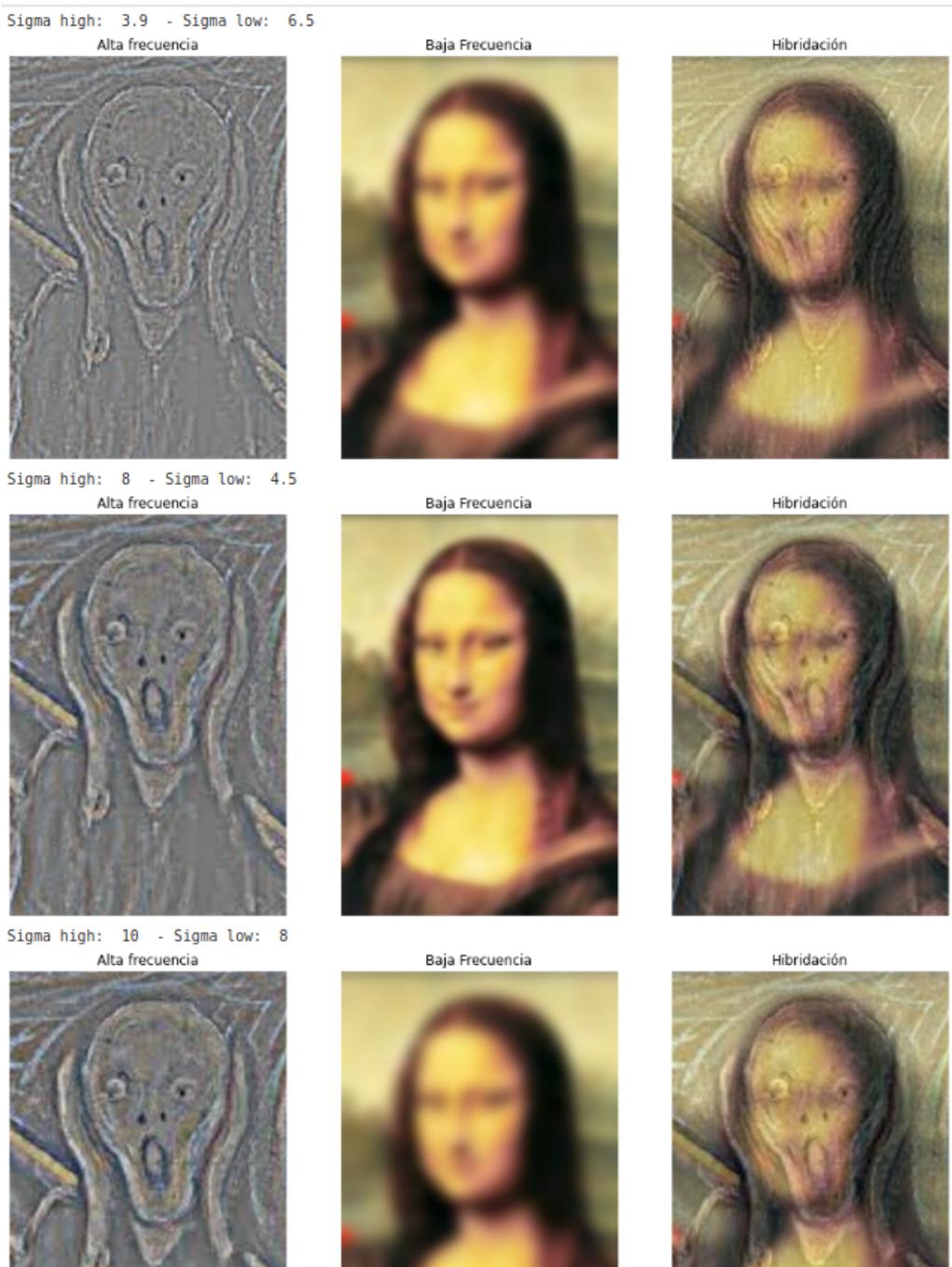


Figura 61: Experimentos realizados

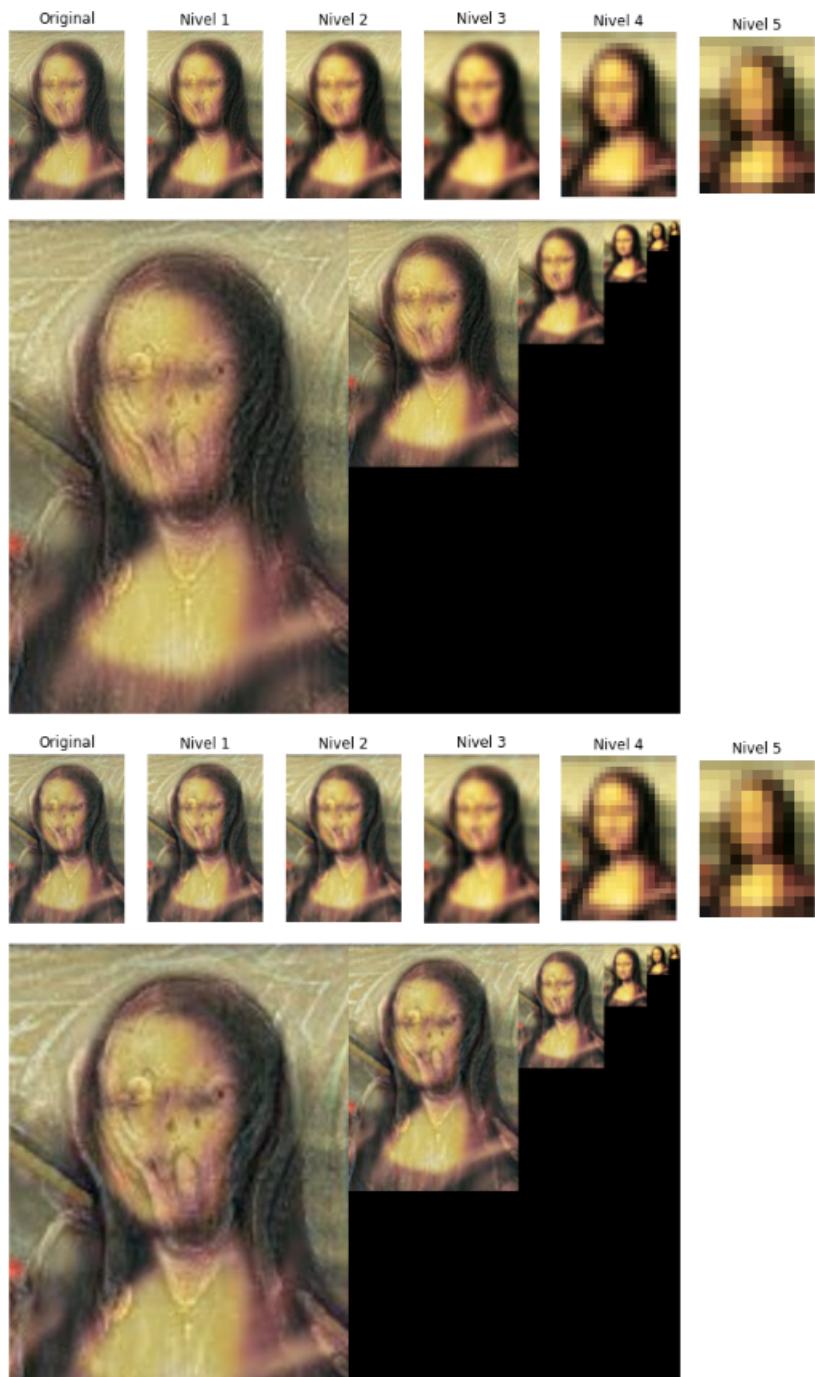


Figura 62: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

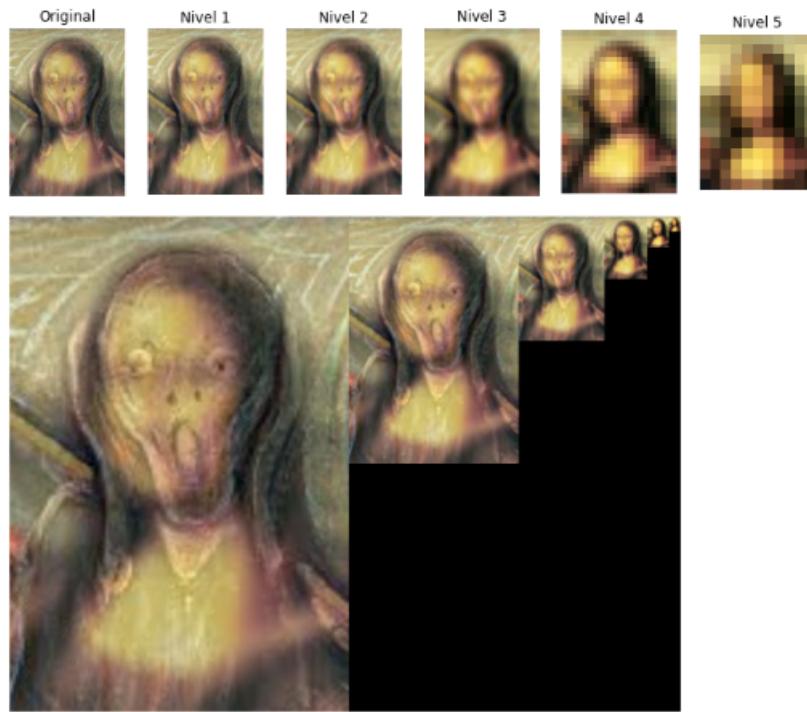


Figura 63: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

Un problema a la hora de hacer esta hibridación es que el pelo de color marrón de La Monalisa está superpuesto a las manos de El Grito. El pelo al ser de color oscuro provoca que las manos de el grito apenas se aprecien dificultando la hibridación. Al aumentar la intensidad de las frecuencias altas tenemos que en ese caso el grito destaca demasiado. Para paliar este problema pensamos la alternativa que ya se ha comentado. Así pues, se alternan ahora los papeles de ambas figuras en cuanto a frecuencias se refiere.

Presentamos a continuación los experimentos de la otra alternativa junto con las pirámides Gaussianas fijándonos únicamente en el tercero que es el que merece la pena. Notamos a que la imagen del grito tiene una cara y manos de color claro y que resalta sobre el oscuro fondo. También se puede distinguir que los rasgos faciales de La Monalisa son muy importantes, los ojos, pupilas, cejas, nariz y boca están muy marcados y al hacer la hibridación tenemos que al superponer esos rasgos en la cara blanca del grito se ven perfectamente. En los distintos niveles de la gaussiana el cambio de La Monalisa a El Grito lo marcan los colores de éste último cuadro.

A modo de conclusión decimos que pese a que el grito presente mas frecuencias altas que La Monalisa, los colores y el tono de éstos en la imagen también juegan un papel importante a la hora de hibridar.

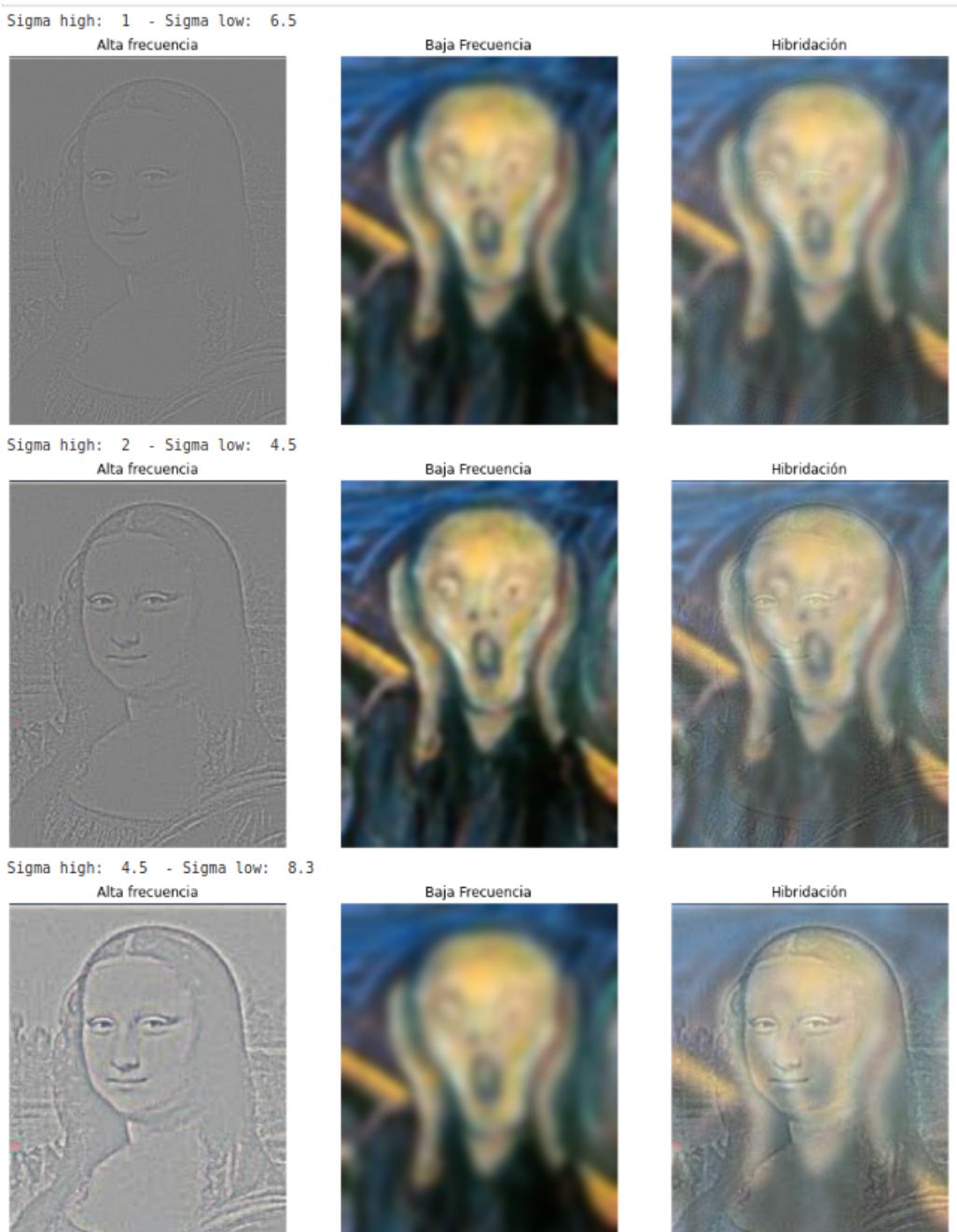


Figura 64: Experimentos realizados

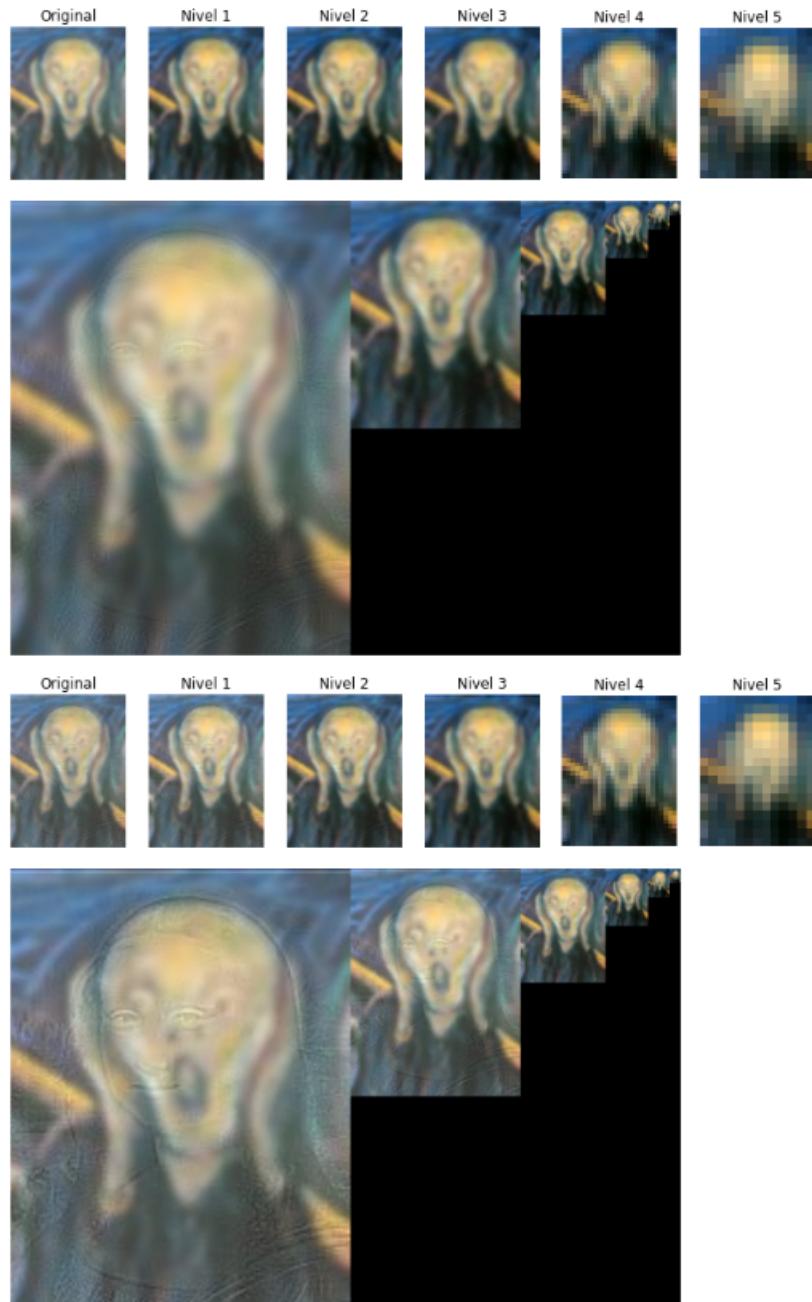


Figura 65: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

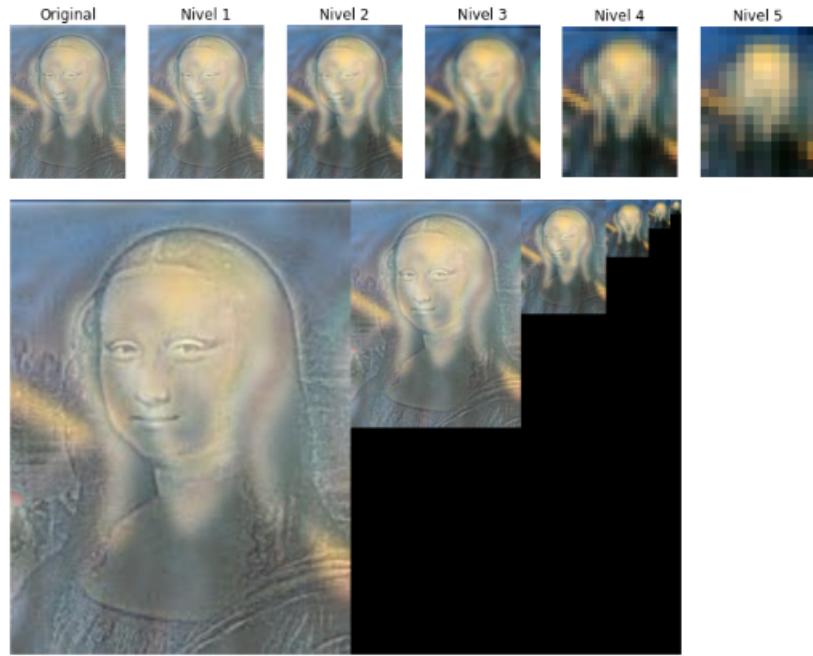


Figura 66: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

Por último, queríamos hacer la hibridación de una imagen en el que el papel del color no fuese tan trascendental como en las dos últimas imágenes. Por lo que se escogió un pie y una zapatilla. La imagen del pie va a ser la de las frecuencias bajas y la zapatilla la de las frecuencias altas por razones obvias. El pie es una imagen más lisa, mientras que las zapatillas presentan un montón de bordes y contornos.

En los experimentos hemos tenido que aumentar las frecuencias altas de las zapatillas para que destaque sobre el pie. En el último quizás se ha emborronado demasiado la imagen del pie. Un buen equilibrio lo podemos observar en el primer y segundo experimento.

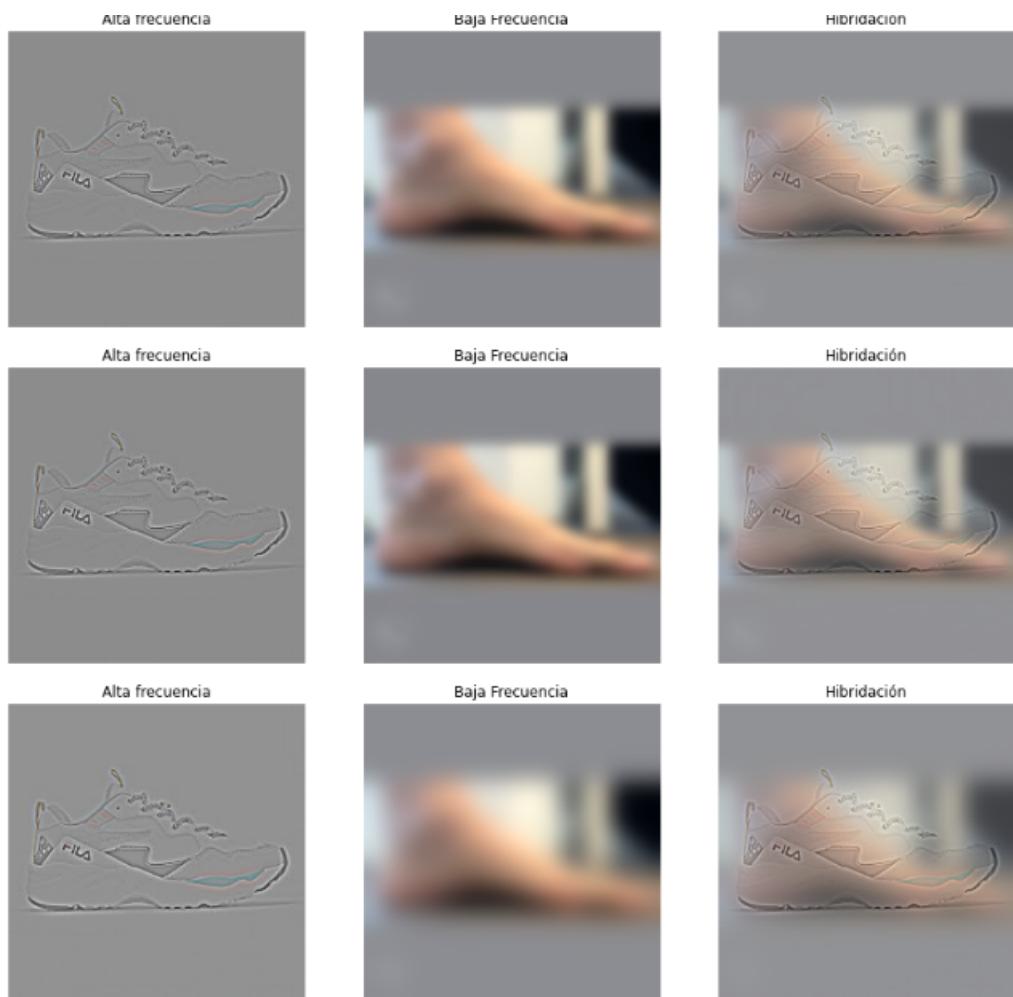


Figura 67: Experimentos realizados

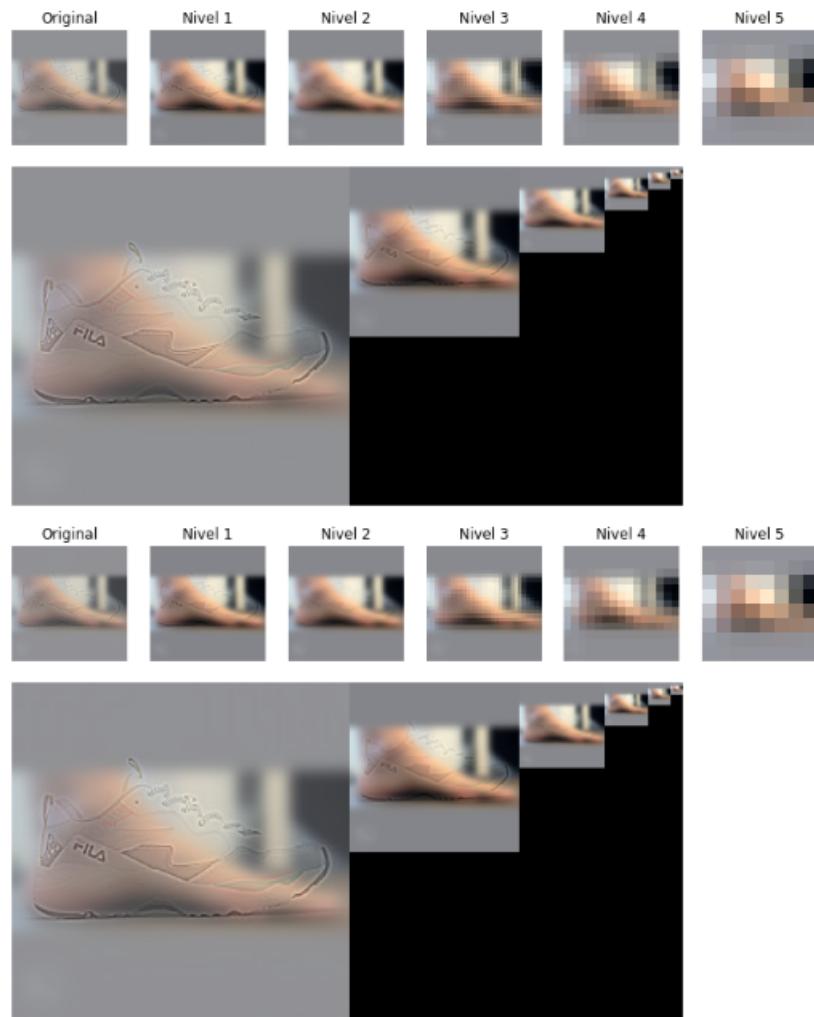


Figura 68: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

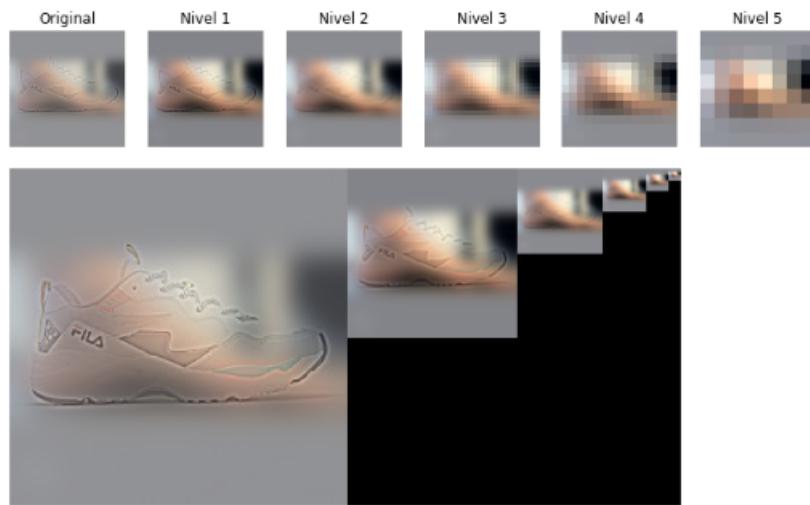


Figura 69: Resultados de hibridación visualizándolos a través de una pirámide Gaussiana

## **4. Bibliografía**

Manual Numpy  
Documentación Opencv  
Imagen de Pilar Aranda  
Imagen de Nicolás Perez  
El grito  
Monalisa  
Imagen del Pie  
Imagen de las zapatillas

## 5. Anexo

*cv.getDerivKernels( dx, dy, ksize[, kx[, ky[, normalize[, ktype]]]] ) -> kx, ky*

- kx: coeficientes fila
- ky: coeficientes columna
- dx: orden de derivdad con respecto x
- dy: orden de derivada con respecto y
- ksize: puede ser FILTER\_SCHARR,1,3,5, o 7
- normalize: bandera que indica si queremos normalizar o no
- ktype: tipo de coeficientes, puede ser CV\_32f o CV\_64F

*cv.GaussianBlur( src, ksize, sigmaX[, dst[, sigmaY[, borderType]]] ) -> dst*

Esta función suaviza una imagen usando un filtro gaussiano convolucionandola con el filtro indicado

- src: imagen de entrada
- dst: imagen de salida del mismo tamaño y tipo que src
- ksize: Tamaño del kernel gaussiano. ksize.width y ksize.height pueden ser distintos pero ambos deben ser positivos e impares. También pueden ser zeros y se calculan a partir del sigma dado
- sigmaX: Desviación típica del kernel Gaussiano en la dirección del eje X
- sigmaY: Desviación típica Gaussiana en la dirección del eje Y. Si sigmaY = 0 entonces tomará el valor dado en sigmaX. Si ambos sigmas son cero, se calculan usando el parametro ksize

*cv.pyrUp( src[, dst[, dstsize[, borderType]]] ) -> dst* Función que expande una imaagen y luego la emborrona. Por defecto el tamaño de la imagen de salida es del doble que la de entrada. El emborronamiento se realiza con el mismo kernel que usa pyrDown multiplicado por 4.

- src: imagen de entrada
- dst: imagen de salida
- dstsize: tamaño de la imagen de salida
- borderType: Método de interpolación, solo soporta BORDER\_DEFAULT

*cv.pyrDown( src[, dst[, dstsize[, borderType]]] ) -> dst* Emborrona y reduce una imagen. Por defecto la reducción se hace con la mitad del tamaño original. Cuando emborrona lo hace con una matriz normalizada obtenida a partir del outer product del vector [1, 4, 6, 4, 1] consigo mismo.

- src: imagen de entrada
- dst: imagen de salida
- dstsize: tamaño de la imagen de salida

- borderType: Método de interpolación, no soporta BORDER\_CONSTANT

***cv.Laplacian( src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]] ] ) -> dst*** Calcula la laplaciana de una imagen. Calcula la laplaciana de una imagen sumando la segunda derivada con respecto de x e y utilizando el operador de Sobel.

- src: imagen de entrada
- dst: imagen de destino
- ddepth: Profundidad de la imagen de salida. Se refiere al número de bits con que se almacenan los números. Cuando mayor sea el numero de bits mayor es el rango que puede almacenar
- ksize: Tamaño usado para computar el filtro de la segunda derivada. Debe de ser impar
- scale: Factor de escala opcional para calcular los valores de la laplaciana. Por defecto no se aplica ninguna escala
- delta: Valor delta opcional que se suma al resultado para guardarlo en la imagen de salida
- borderType: metodo e interpolación. No soporta BORDER\_WRAP