

Visión por computador

Trabajo 1



Cuestionario

1.- ¿Cuáles son los objetivos principales de las técnicas de visión por computador? Poner algún ejemplo si lo necesita

Los objetivos de las técnicas de visión por computador serían a partir de los píxeles de una imagen, sacar información de esa imagen. Por lo tanto se pueden percibir e interpretar objetos, personas... Además, se puede sacar formas geométricas mediante el cálculo de propiedades del mundo.

Por lo tanto, podríamos saber si una foto es de un paisaje, reconociendo árboles, montañas... o incluso ver mediante la geometría si se ha tomado con una perspectiva.

Por lo tanto, el objetivo de la visión por computador sería interpretar imágenes.

2.- ¿Una máscara de convolución para imágenes debe ser siempre una matriz 2D?

No tiene porque ser siempre 2D, ya que la operación de convolución tiene la propiedad de separabilidad. Por lo tanto, se podría aplicar convolución primero por filas o columnas, y luego la restante, como se ha realizado en esta práctica. Esto sería una máscara de convolución 1D.

¿Tiene sentido considerar máscaras definidas a partir de matrices de varios canales como p.e. el tipo de OpenCV CV_8UC3?

No tendría sentido, ya que en estas máscaras trabajamos con reales para los puntos. Por lo tanto, al tener los colores RGB intensidades diferentes, están diferenciados y no podrían tratarse. Como cada color representa un canal, tienen información distinta.

3.- Expresar y justificar las diferencias y semejanzas entre correlación y convolución.

Semejanzas:

- Son técnicas de procesado de imágenes.
- No tienen orden temporal y causal.
- Funcionan casi de la misma manera.
- Si los filtros son simétricos, no hay diferencia en las operaciones.

Diferencias:

- La convolución suele describir transformaciones de sistemas lineales (filtrado de suavizado).
- Con la convolución se oscurecen imágenes, se elimina ruido...
- La correlación suele medir la similitud de patrones.
- Con la correlación se detectan objetos.

4.- ¿Los filtros de convolución definen funciones lineales sobre las imágenes?

Si, ya que correlación y convolución son operaciones de desplazamiento lineal. Como vimos en la función de convolución, esta se comporta de igual manera para cada pixel de la imagen, transformando la imagen ligeramente, como un alisado.

¿y los de mediana?

El filtro de mediana no es lineal, lo que implica que no se obtienen mediante operaciones de convolución. Este filtro es un suavizado, que consiste en reemplazar el valor de los píxeles originales por la mediana de los valores de la máscara. Obtiene resultados buenos en suavizados de ruido de sal y pimienta. Además no elimina las altas frecuencias.

5.- ¿La aplicación de un filtro de alisamiento debe ser una operación local o global sobre la imagen?

La aplicación de un filtro de alisamiento es lineal, por lo que será una operación local. Se pueden obtener con convolución, aplicando una ventana deslizante.

6.- Para implementar una función que calcule la imagen gradiente de una imagen dada pueden plantearse dos alternativas:

a) Primero alisar la imagen y después calcular las derivadas sobre la imagen alisada

b) Primero calcular las imágenes derivadas y después alisar dichas imágenes.

Discutir y decir que estrategia es la más adecuada, si alguna lo es.

Para que la primera derivaba refleje los saltos que contienen información en la imagen, habría que alisarla primero para igualar los píxeles cercanos, por si tenemos algunos saltos que nos perjudiquen. Por lo tanto la opción correcta sería la A.

7.- Verificar matemáticamente que las primeras derivadas (respecto de x e y) de la Gaussiana 2D se puede expresar como núcleos de convolución separables por filas y columnas. Interpretar el papel de dichos núcleos en el proceso de convolución.

Sabemos que la función Gaussiana es separable en núcleos de convolución, una para filas (x) y otra para columnas(y).

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

Aquí podemos ver como quedarían las dos funciones separadas, siendo en este caso las dos (identicas) 1D Gaussian.

*El proceso de verificación podría ser algo como:

For independent random variables X and Y , the distribution f_Z of $Z = X+Y$ equals the convolution of f_X and f_Y :

$$f_Z(z) = \int_{-\infty}^{\infty} f_Y(z-x) f_X(x) dx$$

Given that f_X and f_Y are normal densities,

$$\begin{aligned} f_X(x) &= \frac{1}{\sqrt{2\pi}\sigma_X} e^{-\frac{(x-\mu_X)^2}{2\sigma_X^2}} \\ f_Y(y) &= \frac{1}{\sqrt{2\pi}\sigma_Y} e^{-\frac{(y-\mu_Y)^2}{2\sigma_Y^2}} \end{aligned}$$

Substituting into the convolution:

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_Y} e^{-\frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}} \frac{1}{\sqrt{2\pi}\sigma_X} e^{-\frac{(x-\mu_X)^2}{2\sigma_X^2}} dx \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sqrt{\sigma_X^2 + \sigma_Y^2}} \exp\left[-\frac{(z - (\mu_X + \mu_Y))^2}{2(\sigma_X^2 + \sigma_Y^2)}\right] \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2 + \sigma_Y^2}}} \exp\left[-\frac{\left(x - \frac{\sigma_X^2(z - \mu_Y) + \sigma_Y^2\mu_X}{\sigma_X^2 + \sigma_Y^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2 + \sigma_Y^2}}\right)^2}\right] dx \\ &= \frac{1}{\sqrt{2\pi(\sigma_X^2 + \sigma_Y^2)}} \exp\left[-\frac{(z - (\mu_X + \mu_Y))^2}{2(\sigma_X^2 + \sigma_Y^2)}\right] \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2 + \sigma_Y^2}}} \exp\left[-\frac{\left(x - \frac{\sigma_X^2(z - \mu_Y) + \sigma_Y^2\mu_X}{\sigma_X^2 + \sigma_Y^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2 + \sigma_Y^2}}\right)^2}\right] dx \end{aligned}$$

The expression in the integral is a normal density distribution on x , and so the integral evaluates to 1. The desired result follows:

$$f_Z(z) = \frac{1}{\sqrt{2\pi(\sigma_X^2 + \sigma_Y^2)}} \exp\left[-\frac{(z - (\mu_X + \mu_Y))^2}{2(\sigma_X^2 + \sigma_Y^2)}\right]$$

Referencia:

https://en.wikipedia.org/wiki/Sum_of_normally_distributed_random_variables#Proof_using_convolution

8.- Verificar matemáticamente que la Laplaciana de la Gaussiana se puede implementar a partir de núcleos de convolución separables por filas y columnas. Interpretar el papel de dichos núcleos en el proceso de convolución.

La verificación podría ser como esto:

$$G_\sigma(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$\Delta[G_\sigma(x, y) * f(x, y)] = [\Delta G_\sigma(x, y)] * f(x, y) = LoG * f(x, y)$$

$$\frac{d}{dt}[h(t)*f(t)] = \frac{d}{dt} \int f(\tau)h(t-\tau)d\tau = \int f(\tau)\frac{d}{dt}h(t-\tau)d\tau = f(t)*\frac{d}{dt}h(t)$$

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

$$\frac{\partial^2}{\partial x^2}G_\sigma(x, y) = \frac{x^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2} - \frac{1}{\sigma^2}e^{-(x^2+y^2)/2\sigma^2} = \frac{x^2 - \sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

$$\frac{\partial^2}{\partial y^2}G_\sigma(x, y) = \frac{y^2 - \sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

$$LoG \triangleq \Delta G_\sigma(x, y) = \frac{\partial^2}{\partial x^2}G_\sigma(x, y) + \frac{\partial^2}{\partial y^2}G_\sigma(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

Referencia:

<http://fourier.eng.hmc.edu/e161/lectures/gradient/node8.html>

9.- ¿Cuáles son las operaciones básicas en la reducción del tamaño de una imagen? Justificar el papel de cada una de ellas.

Para reducir el tamaño de una imagen primero se convoluciona la imagen con un núcleo de Gauss. Y después se eliminan algunas columnas y filas.

10.- ¿Qué información de la imagen original se conserva cuando vamos subiendo niveles en una pirámide Gaussiana? Justificar la respuesta.

Lo que vamos a conservar conforme vamos subiendo niveles en una pirámide Gaussiana serán las frecuencias bajas, ya que las altas se han ido perdiendo a cada nivel que ha pasado con la convolución.

11.- ¿Cuál es la diferencia entre una pirámide Gaussiana y una Piramide Laplaciana? ¿Qué nos aporta cada una de ellas?

Con la pirámide Gaussiana lo que vamos haciendo es generar un alisamiento en las imágenes y con la pirámide Laplaciana vamos sacando los bordes. La Gaussiana sería un filtro pasa bajas y la Laplaciana un filtro pasa banda.

12.- Cual es la aportación del filtro de Canny al cálculo de fronteras frente a filtros como Sobel o Robert.

Con el filtro de Canny conservamos los bordes de mayor calidad. En Sobel por ejemplo se observa más ruido y bordes con menos definición.

Canny lo consigue ya que aplica histéresis para guardar algunos bordes, eliminando ruido.

13.- Buscar e identificar una aplicación real en la que el filtro de Canny garantice unas fronteras que sean interpretables y por tanto sirvan para solucionar un problema de visión por computador.

Las aplicaciones reales que podrían estar presente, podrían ser la detección de rostros en las fotos, la detección de objetos por parte del ordenador...

Ejercicios

A) Implementar una función de convolución (ejemplo, void my_imGaussConvol(Mat& im, Mat& maskCovol, Mat& out)) debe ser capaz de calcular la convolución 2D de una imagen con una máscara. Ahora supondremos que la máscra es extraída del muestreo de una Gaussiana 2D simétrica. Para ello implementaremos las siguientes funciones auxiliares:

1).- Calculo del vector máscara: Sea $f(x) = \exp(-0.5 \frac{x^2}{\sigma^2})$ una σ^2 función donde σ (sigma) representa un parámetro en unidades píxel. Implementar una función que tomando sigma como parámetro de entrada devuelva una máscara de convolución representativa de dicha función. Justificar los pasos dados (Ayuda: comenzar calculando la longitud de la máscara a partir de sigma y recordar le valor de la suma de los valores del núcleo)

```
Mat out;
//Leo las imágenes pasadas por parámetros:
Mat img = leeImagen(nomb1, color);

/*********************PARTE 1********************/
//Calculo la máscara:
if (parte == 2) {
    int tam_masc = TamanioMascara(sigma);
    Mat mask;
    CalculaMascara(mask, sigma, tam_masc);
```

Primero leo una imagen, y a través del parámetro “sigma” calcularé el tamaño de la máscara, para después crear esa máscara que se aplicará en el siguiente apartado.

2) Implementar una función que calcule la convolución de un vector señal 1D con un vector-máscara de longitud inferior al de la señal usando dos posibles tipos de condiciones de contorno (uniforme a ceros y reflejada). La salida será un vector de igual longitud que el vector señal de entrada. (Ayuda: Definir una matriz auxiliar con longitud definida a partir de las dimensiones de las dos matrices de entrada. Trabajar sobre él y copiar el resultado obtenido en la matriz de salida. En el caso de que el vector de entrada sea de color, habrán de extraerse cada uno de los tres vectores correspondientes a cada una de las bandas, calcular la convolución sobre cada uno de ellos y volver montar el vector de salida. Usar las funciones split() y merge())

```
*****PARTE 2*****  
Mat sal1;  
sal1 = AniadirBordes(img, tam_masc, modo_borde);  
Mat sal2;  
sal2 = AplicaConvolucion(sal1, mask, tam_masc);  
QuitarBordes(sal2, tam_masc);  
out = sal2;  
}
```

Una vez calculada la máscara, vamos a añadir bordes a la imagen original, obteniendo este resultado como la salida de la función. En esta función tendremos la posibilidad de indicar el modo que queremos para el borde, pudiendo ser uniforme a ceros o un borde reflejado.

Con los bordes ya aplicados, aplicamos la convolución a esta imagen y con la máscara anteriormente calculada. La convolución se tratará distinta en función del tipo de la imagen.

Una vez aplicada convolución, le quitaremos los bordes añadidos, y ya tendremos lista la imagen final.

3.- Implementar una función que tomando como entrada una imagen y el valor de sigma calcule la convolución de dicha imagen con una máscara Gaussiana 2D. Usar las funciones implementadas en el punto anterior.

```
*****PARTE 3*****
if (parte == 3) {
    my_imGaussConvol(img, out, sigma);
}

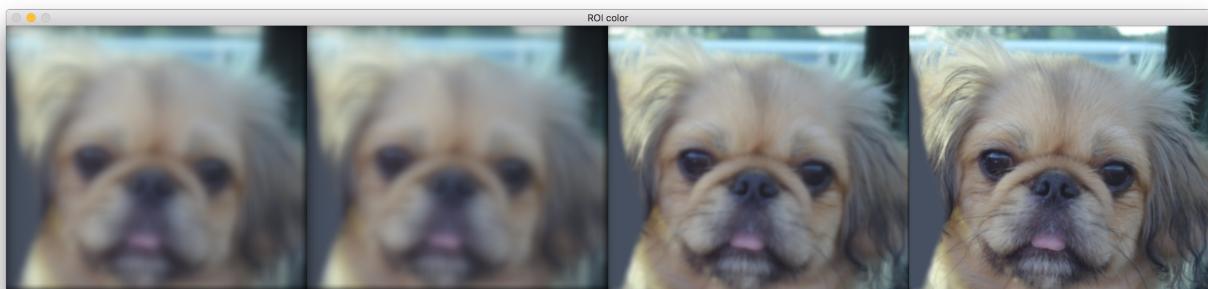
return out;
```

En este apartado, la única diferencia sería juntar todo el trabajo descrito en los dos anteriores apartados en una sola función, quedando algo como:

```
/*Función que calcula la convolución directamente en 2D */
void my_imGaussConvol(Mat img, Mat& out, float sigma) {
    int tam_masc = TamanioMascara(sigma);
    Mat mask;
    CalculaMascara(mask, sigma, tam_masc);
    Mat sal1;
    sal1 = AnadirBordes(img, tam_masc);
    out = AplicaConvolucion(sal1, mask, tam_masc);
    QuitarBordes(out, tam_masc);
}
```

De esta forma, se pueden ver los dos apartados anteriores unidos, por lo que los resultados son los mismos.

Ahora voy a añadir algunos resultados con diferentes valores de sigma, modo_borde, color e imagen:



En este caso he usado la imagen “dog.bmp”. En la primera y segunda la única diferencia es el borde aplicado, siendo en el primero el uniforme a ceros y reflejado, respectivamente. La imagen es en color y sigma tiene un valor de 10. No se aprecian diferencias significativas con los bordes.

En la tercera, usando un borde uniforme a ceros, vemos la diferencia con respecto a la primera, ya que en ella hemos usado un sigma igual a 3.

Por último, en la última hemos usado un sigma igual a 1.



En este caso, tenemos la imagen “lena.jpg” y en escala de grises. Los valores de las dos primeras capturas son sigma igual a 10 y borde uniforme a ceros y reflejado, respectivamente.

En la tercera imagen el borde es uniforme a ceros y un sigma igual a 5.

En la última, el borde es uniforme a ceros y el sigma pasa a valer 3.

Por lo tanto, hemos podido ver las diferencias a la hora de usar distintos sigmas. Con respecto al tipo de borde, no se aprecia una diferencia demasiado llamativa.

B) Imágenes Híbridas

Mezclando adecuadamente la parte de frecuencias altas de una imagen con las parte de frecuencias bajas de la 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. 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.

Usar la función que hemos implementado en el apartado A para elegir los sigmas más adecuados para la selección de frecuencias en parejas de imágenes (ver fichero de datos).

1. Implementar una función que genere las imágenes de baja y alta frecuencia.

```
/* Función para crear imágenes híbridas */
Mat CrearImagenHibrida(Mat img1, Mat img2, vector<Mat>& imagenes, int sigma1, int sigma2) {
    Mat alta_frec, baja_frec, res;

    //Aplicamos convolución para obtener las altas frecuencias
    my_imGaussConvol(img1, alta_frec, sigma1);
    img1.convertTo(img1, CV_8UC3);
    alta_frec.convertTo(alta_frec, CV_8UC3);
    //Sacamos solo las altas frecuencias
    alta_frec = img1 - alta_frec;

    //Aplicamos convolución para obtener las bajas frecuencias
    my_imGaussConvol(img2, baja_frec, sigma2);
    //La imagen híbrida será la suma de las altas frecuencias de una con
    //las bajas frecuencias de la otra
    res = alta_frec + baja_frec;

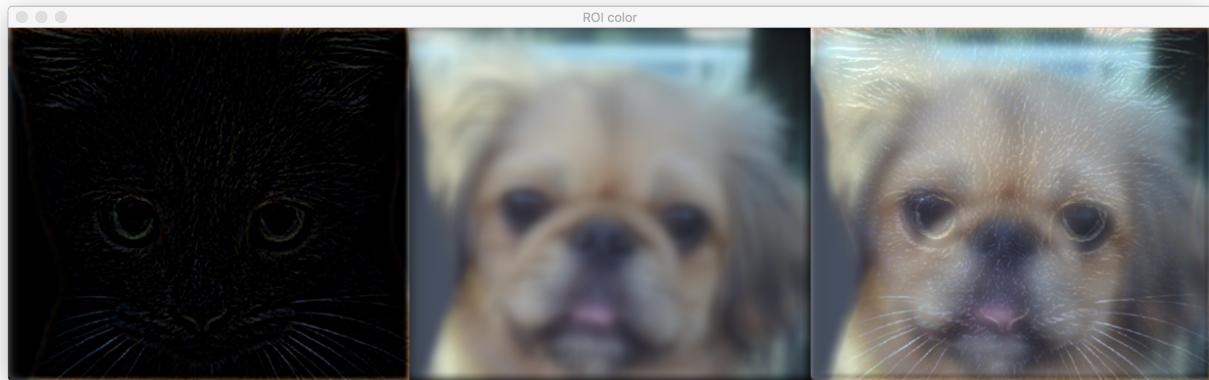
    imagenes.push_back(alta_frec);
    imagenes.push_back(baja_frec);
    return res;
}
```

En esta función, a la que le pasamos dos imágenes y los sigmas deseados para cada una, vamos a sacar dos nuevas imágenes. Una con las altas frecuencias de una de ellas, y la otra con las bajas frecuencias de la otra. De esta forma vamos a juntar ambas imágenes y obtendremos una imagen híbrida.

2. Escribir una función para mostrar 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)

Esta función la tenía implementada en el Trabajo0 para crear un ROI, por lo que solo ha sido modificarla un poco para el Ejercicio A y la he usado para este ejercicio.

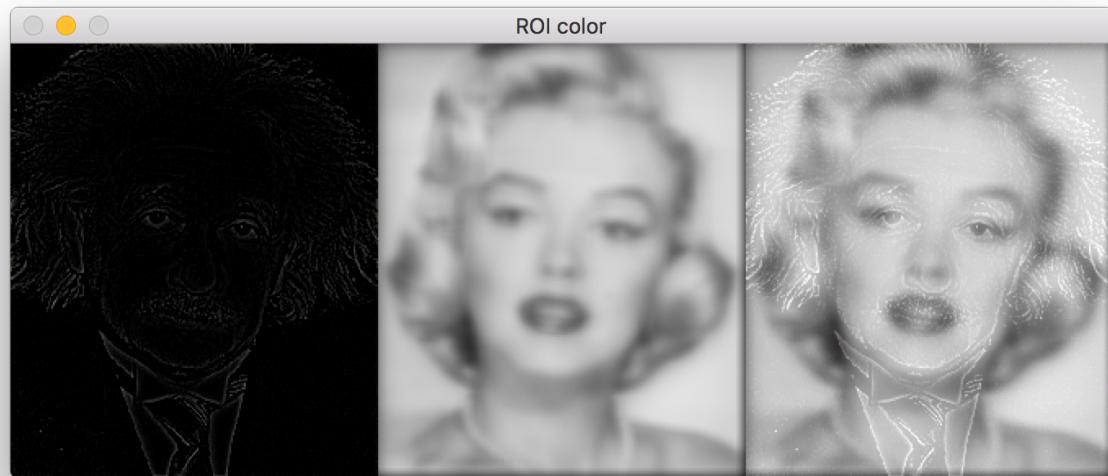
Podemos ver algunos resultados de imágenes híbridas como estos:



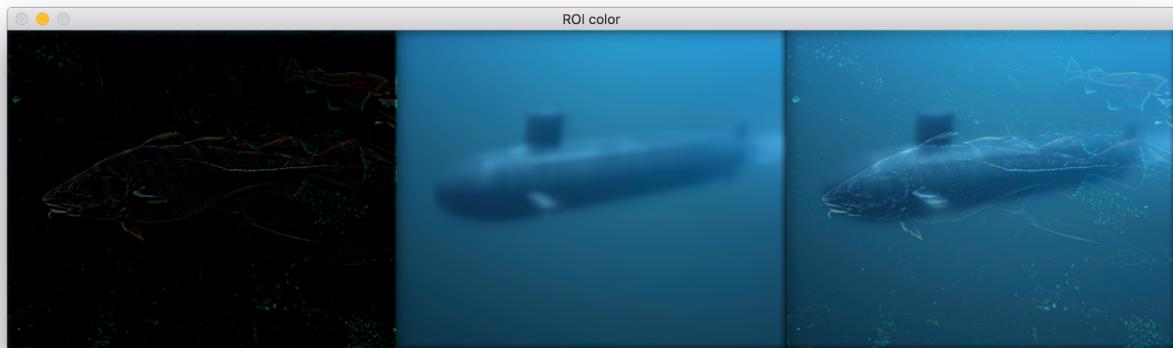
En esta he realizado la híbrida entre cat y dog, con sigma 5 y 9 respectivamente.



En esta he realizado la híbrida entre motorcycle y bicycle, con sigma 2 y 7 respectivamente.



En esta he realizado la híbrida entre einstein y marilyn, con sigma 2 y 6 respectivamente.



En esta he realizado la híbrida entre fish y submarine, con sigma 3 y 7 respectivamente.



En esta he realizado la híbrida entre bird y plane, con sigma 4 y 7 respectivamente.

*El valor de sigma en cada caso ha dependido del que he considerado más oportuno para cada par de imágenes.

C) Construir una pirámide Gaussiana de al menos 5 niveles con las imágenes híbridas calculadas en el apartado anterior. Mostrar los distintos niveles de la pirámide en un único canvas e interpretar el resultado. (En este punto se puede usar la función pyrDown() de OpenCV si se desea)

```
***** APARTADO C *****
string nombr13 = "imagenes/cat.bmp", nombr14 = "imagenes/dog.bmp";
Mat img13 = leeImagen(nombr13, 1); Mat img14 = leeImagen(nombr14, 1);
int sigmaa, sigmab;
sigmaa = 5; sigmab = 9;
vector<Mat> imagenes_hib6, imagenes_pir;
Mat hibrida = CrearImagenHibrida(img13, img14, imagenes_hib6, sigmaa, sigmab);
imagenes_pir.push_back(hibrida);
PiramideGaussiana(imagenes_pir);
pintaMROI(imagenes_pir, "Piramide Gaussiana");
```

Se puede ver como a partir de una imagen híbrida se le aplica la función de PiramideGaussiana:

```
/* Función para crear una pirámide Gaussiana */
void PiramideGaussiana(vector<Mat>& imagenes) {
    Mat aux1;
    Mat aux2;
    imagenes[0].copyTo(aux1);
    imagenes[0].copyTo(aux2);
    int niveles = 5;
    for (int i = 0; i < niveles; i++) {
        pyrDown(aux1, aux2, Size(aux2.cols / 2, aux2.rows / 2));
        aux2.copyTo(aux1);
        imagenes.push_back(aux2);
    }
}
```

Con esta función se obtiene, con ayuda de pyrDown, una pirámide gaussiana de 5 niveles (yo he añadido la imagen original).

El resultado es el siguiente: (lo que aparece en la segunda imagen es debido a que las pruebas las he tenido que hacer en mi portátil)

