Trabajo de Visión por Computador

TRABAJO-1

VALORACIÓN TOTAL: 18 puntos (10 teoría y 8 programación)

Filtrado y Muestreo

Fecha de entrega: 21 de octubre

Informe a presentar

Para este trabajo como para los demás proyectos debe presentar un informe escrito con sus valoraciones y decisiones adoptadas en cada uno de los apartados de la implementación. También deberá incluirse una valoración sobre la calidad de los resultados encontrados. (hacerlo en pdf)

Normas de la entrega de Prácticas: EL INCUMPLMIENTO DE ESTAS NORMAS SIGNIFICA PERDIDA DIRECTA DE 1 PUNTO CADA VEZ QUE SE DETECTE UN INCUMPLIMIENTO.

1. El código se debe estructurar como un único fichero main que irá llamando de forma secuencial a distintas funciones, una por cada apartado de la práctica.
2. El código debe estar obligatoriamente comentado explicando lo que realizan los distintos apartados y/o bloques.
3. Todos los ficheros juntos se podrán dentro de un fichero zip/rar..
4. SOLO ENTREGAR EL CODIGO FUENTE. (NO INCLUIR IMÁGENES)
5. Los path que se usen en la lectura de imágenes o cualquier fichero de entrada debe ser siempre "imagenes/nombre_fichero"
6. Todos los resultados numéricos serán mostrados por pantalla. No escribir nada en el disco.
7. La práctica deberá poder ser ejecutada de principio a fín sin necesidad de ninguna selección de opciones. Para ello se deberá de fijar los parámetros por defecto que se consideren óptimos.
8. Poner puntos de parada para mostrar imágenes o datos por consola y el final de cada apartado
9. NO ESTA PERMITIDO EL USO DE CODIGO C/C++ que no sea absolutamente estándar. Por ejemplo no se permite dimensionar matrices a partir de variables o usar tipos de datos específicos de un compilador.

Forma de entrega:. Subir el zip al Tablón docente de CCIA (decsai.ugr.es).

CUESTIONARIO DE TEORÍA: (estará disponible en la web más adelante)

PROGRAMACIÓN: (8 puntos) Base 55 puntos

Este trabajo de implementación tiene como objetivo aprender a manejar las operaciones de convolución y correlación, la representación de imágenes mediante una pirámide y analizar algunas propiedades de las frecuencias presentes en las imágenes.

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 máscara 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 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 que el valor de la suma de los valores del núcleo es 1) (1.5 punto)**

2) Implementar una función que calcule la convolución de un vector señal 1D con un vector-máscara 1D 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. Pasos:

a) **Implementar una función que calcule la convolución 1D entre dos vectores,dando de salida solo los valores donde ha sido posible el cálculo**
b) **Definir una vector auxiliar de longitud mayor definida a partir de las dimensiones de las dos vectores de entrada, p.e. la señal tiene longitud N y el otro 2k+1 entones el vector auxiliar será N+2k. Copiar en su centro el vector señal y rellenar los extremos usando el criterio de borde elegido**
c) **Ejecutar la función del paso a sobre el vector creado y la máscara. . 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() de opencv (2.5 puntos)**

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. Recordar que la Gaussiana es descomponible en convoluciones 1D por filas y columnas (1 punto)

B) Imágenes Híbridas: (SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns).

Mezclando adecuadamente una parte de las frecuencias altas de una imagen con una parte de las frecuencias bajas de otra imagen, obtenemos una imagen híbrida que admite distintas interpretaciones a distintas distancias (ver hybrid images project page).
Para seleccionar la parte de frecuencias altas y bajas que nos quedamos de cada una de las imágenes usaremos el parámetro sigma del núcleo/máscara de alisamiento gaussiano que usaremos. A mayor valor de sigma mayor eliminación de altas frecuencias en la imagen convolucionada. Para una buena implementación elegir dicho valor de forma separada para cada una de las dos imágenes (ver las recomendaciones dadas en el paper de Oliva et al.). Recordar que las máscaras 1D siempre deben tener de longitud un número impar.

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). (2 puntos)

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

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 , aunque se valorará una implementación personal de la misma con +1 punto) (1 puntos)

OPCION-2:

PROGRAMACIÓN: (5 puntos) Base 33 puntos

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 máscara es extraída del muestreo de una Gaussiana 2D simétrica. Para ello implementaremos haciendo uso de las siguientes funciones auxiliares (2 puntos):

1).- Calculo del vector máscara: Usar la función getGausssianKernel() . Verificar el resultado mostrando los vectores obtenidos

2).- Calcular la convolución de una imagen con una máscara gaussiana, usar filter2d(). Verificar el resultado mostrando resultado sobre distintas imágenes con distintos valores de sigma.

B) Imágenes Híbridas: (SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns). (2 puntos)

Mezclando adecuadamente una parte de las frecuencias altas de una imagen con una parte de las frecuencias bajas de otra imagen, obtenemos una imagen híbrida que admite distintas interpretaciones a distintas distancias (ver hybrid images project page).
Para seleccionar la parte de frecuencias altas y bajas que nos quedamos de cada una de las imágenes usaremos el parámetro sigma del núcleo/máscara de alisamiento gaussiano que usaremos. A mayor valor de sigma mayor eliminación de altas frecuencias en la imagen convolucionada. Para una buena implementación elegir dicho valor de forma separada para cada una de las dos imágenes (ver las recomendaciones dadas en el paper de Oliva et al.). Recordar que las máscaras 1D siempre deben tener de longitud un número impar.

Usar la convolución 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). (1.5 puntos)

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

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. (1 punto)