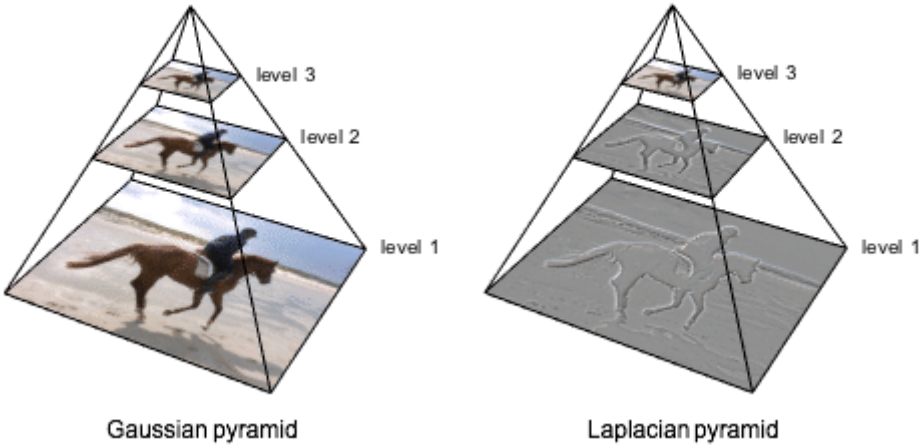
Tarea 1 EL7008 – Primavera 2022
Pirámides de Gauss y Laplace

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: 18 de Agosto de 2022
Fecha entrega: 31 de Agosto de 2022

El objetivo de esta tarea es implementar el cálculo de pirámides de Gauss y Laplace de una imagen, además de realizar filtrado de tipo pasa alto de la imagen.

Las pirámides de Gauss y Laplace son representaciones multi-resolución calculadas a partir de una imagen. Las pirámides consisten en un conjunto de imágenes organizadas en niveles, donde cada imagen tiene la mitad del ancho y alto del nivel anterior (en el caso de pirámides diádicas), como se muestra en la figura.



Gaussian pyramid Laplacian pyramid

Pirámide de Gauss

La pirámide de Gauss contiene varias imágenes que representan distintas resoluciones posibles para una misma imagen. Para poder calcular la imagen de un nuevo nivel de esta pirámide, se debe:

1) Suavizar (filtrar pasa bajos con una gaussiana) la imagen del nivel anterior.
2) Submuestrear (reducir de tamaño) la imagen suavizada. Para esto, se debe elegir sólo las filas y columnas pares de la imagen suavizada, tras lo cual la nueva imagen tiene la mitad del ancho y alto del nivel anterior
3) Almacenar la imagen submuestreada en la pirámide de Gauss

Un pseudocódigo para calcular la pirámide de Gauss G es el siguiente:

```
G_1 = imagen_original
Para i en el rango 2 hasta N:
  imagen_suavizada = blur(G_i-1)
  imagen_submuestreada = subsample(imagen_suavizada)
  G_i = imagen_submuestreada
```

Pirámide de Laplace

La pirámide de Laplace contiene la información que se pierde al ir bajando la resolución de la imagen original en el proceso de formación de la pirámide de Gauss. Suele contener bordes y otras estructuras que se pierden al reducir la resolución. Además, se le suele agregar la última imagen de la pirámide de Gauss. Para poder calcular la imagen de un nuevo nivel de esta pirámide, se debe:

1) Elegir y suavizar la imagen de la pirámide de Gauss correspondiente
2) Restar la imagen de la pirámide de Gauss antes y después de suavizarla
3) Almacenar el resultado de la resta en la pirámide de Laplace
4) Almacenar la última imagen de la pirámide de Gauss en la pirámide de Laplace (sólo para el último nivel)

Un pseudocódigo para calcular la pirámide de Laplace L a partir de la pirámide de Gauss G es el siguiente:

```
Para i en el rango 1 hasta N-1:
  imagen_suavizada = blur(G_i)
  L_i = G_i – imagen_suavizada
L_N = G_N
```

A partir de la pirámide de Laplace es posible reconstruir la imagen original usada para crear las imágenes. Para esto se debe:

1) Elegir el último piso de la pirámide de Laplace
2) Para cada nivel restante de la pirámide de Laplace (desde arriba hacia abajo):
    a. Duplicar el tamaño de la imagen siendo procesada (usando interpolación)
    b. Sumar la imagen correspondiente de la pirámide de Laplace

Un pseudocódigo es el siguiente:

```
imagen = L_N
Para i en el rango N-1 hasta 1:
  imagen = upsample(Imagen)
  imagen = imagen + L_i
```

En la tarea, se deben realizar los siguientes procedimientos:

A. Cálculo de pirámides de Gauss

1. Programar una función (en cython) que reciba una imagen de entrada en escala de grises y una máscara (o kernel), calcule la convolución entre ambas y genere una imagen de salida. No se debe usar funciones de OpenCV que calculen directamente convoluciones.
    - Función: `convolution_cython(input, mask)`
2. Programar una función que genere una máscara gaussiana bidimensional. La función debe recibir la desviación estándar de la gaussiana (parámetro sigma) y el tamaño de la ventana. Se debe notar que el resultado debe normalizarse para que la suma de los valores dentro de la máscara sean iguales a 1.
    - Función: `compute_gauss_mask_2d(sigma, width)`
3. Programar una función que permita suavizar (usando gaussianas) una imagen, usando la función `convolution_cython(input, mask)` ya implementada, y una máscara gaussiana bidimensional calculada mediante `compute_gauss_mask_2d(sigma, width)`. Debe recibir una imagen de entrada, la desviación estándar y el ancho de la ventana
    - Función: `apply_blur(input, sigma, width)`

4. Programar una función que reciba una imagen y genere una imagen submuestreada. Para realizar esto, se debe crear una imagen con la mitad del tamaño de la imagen original, y copiar sólo los pixeles pares a la nueva imagen.
    - Función: `do_subsample(img)`
5. Programar una función que, a partir de una imagen, calcule una pirámide de Gauss, que tenga 5 niveles en total (considerando la imagen original como el primer nivel)
    - Función: `calc_gauss_pyramid(input, levels)`
6. Programar la función que permita visualizar (mostrar) la pirámide de Gauss
    - Función: `show_gauss_pyramid(pyramid)`
7. Probar el sistema de cálculo de pirámides sobre 4 imágenes entregadas, mostrando las imágenes de las pirámides.

B. Cálculo de pirámides de Laplace

1. Programar una función que permita restar dos imágenes, píxel a píxel.
    - Función: `subtract(input1, input2)`
2. Programar una función que, a partir de una imagen, calcule una pirámide de Laplace, que tenga 5 niveles en total.
    - Función: `compute_laplace_pyramid(input, levels)`
3. Programar una función que dada una pirámide de Laplace de una imagen, reconstruya la imagen original.
    - Función: `do_reconstruct(laplacepyr)`
4. Programar la función que permita visualizar (mostrar) la pirámide de Laplace. Tenga en cuenta que esto puede requerir tener que escalar y tomar el valor absoluto de los valores de los pixeles.
    - Función: `show_laplace_pyramid(pyramid)`
5. Probar el sistema de cálculo de pirámides sobre 4 imágenes entregadas, mostrando las imágenes de las pirámides y la imagen reconstruida a partir de la pirámide de Laplace y el último piso de la pirámide de Gauss

C. Filtrado Pasaaltos

1. Programar funciones que permitan realizar el filtrado pasa alto de una imagen utilizando (a) un filtro de tipo *Laplacian of Gaussian* y (b) un filtro de tipo *Derivative of Gaussian*. Pruebe los filtros con las 4 imágenes de prueba, considerando distintos tamaños de ventana de los filtros, los cuales se relacionan con el valor de desviación estándar de las funciones Gaussianas.
2. Aplicar los filtros a cada piso (imagen) de la pirámide de Gauss de las imágenes de prueba. Para cada piso debe elegir el tamaño de la ventana a utilizar y por ende la desviación estándar de las funciones Gaussianas. Para cada piso compare y analice las imágenes obtenidas con cada filtro y la imagen del nivel correspondiente de la pirámide de Laplace.

El código a implementar debe basarse en el código base entregado y debe poder funcionar en Colaboratory. Se debe programar en python usando OpenCV. Las operaciones de convolución (filtrado) se deben programar en cython. En esta tarea, las imágenes a color deben ser transformadas a escalas de grises para simplificar la implementación.

Se entrega un código base (un notebook de Colaboratory) que se puede ejecutar, pero no realiza el cálculo de las pirámides ni la reconstrucción de la imagen original. Sin embargo, contiene instrucciones que permiten programar las partes del código faltantes.

En esta tarea no se puede usar funciones de OpenCV que calculen convoluciones, calculen máscaras u operen sobre la imagen completa.

El informe debe contener como mínimo: introducción, marco teórico (descripción de los algoritmos), descripción muy breve de lo realizado en cada sección del informe junto con el código implementado en dicha sección, resultados (mostrando las imágenes resultantes), análisis de los resultados y conclusiones generales. Si no se reporta alguna sección en el informe, se considerará no realizada.

Los informes y los códigos deben ser entregados en el día miércoles 31 de Agosto a las 23:59, mediante u-cursos. Cada día de retraso (incluyendo fines de semana) será castigado con un punto de descuento en la nota.

Importante:

1. La evaluación de esta tarea considerará el correcto funcionamiento del sistema, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión del código en las secciones del informe correspondientes, así como la prolijidad y calidad del mismo.
2. Se entrega una pauta que indica la estructura esperada del informe.

Importante:

Los informes y códigos (los notebooks .ipynb) se deben subir a u-cursos.

Además, los informes se deben subir a la plataforma turnitin, la cual permite detectar copias y plagios. El link para subir los informes a turnitin es:

https://uchile.turnitin.com/originality/hand-in-link/new?jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyZWR3b29kSWQiOiIzYzgxZGQ2Yi01OWE2LTQ2MGEtODE2OC01MGI0OWM1Y2YyYTEiLCJleHAiOjE2NjM4MTU2MDAsImlhdCI6MTY2MDgzMTgzNCwianRpIjoiMDVmYWI0N2QtMzE5My00MWM1LTlhMmMtNzM3Yjg3YTRiMTg0IiwidGVuYW50IjoidWNoaWxlIiwiZm9sZGVySWQiOiIxYmU2Y2E2NC1kNDAwLTRjNTItYjZhNy01ZDlhZTIyZmY1OWIifQ.FEKe7JwcGDKO8JJk-KVBtDm72kKZHaI5DXTfQgE0YWQ

El código completo se debe adjuntar como un anexo dentro del mismo informe, y debe estar en formato texto (no pantallazo). Se permite que el código esté como texto plano en el Anexo. Para esto, el notebook se puede bajar primero como un archivo .py, y luego el texto se puede copiar al informe.

El informe se debe escribir en Latex o MS Word, **no se permite entregar el notebook como informe**.

Nota: ejemplos de imágenes de salida esperadas:

Gauss (5 pisos):

Laplace (5 pisos):