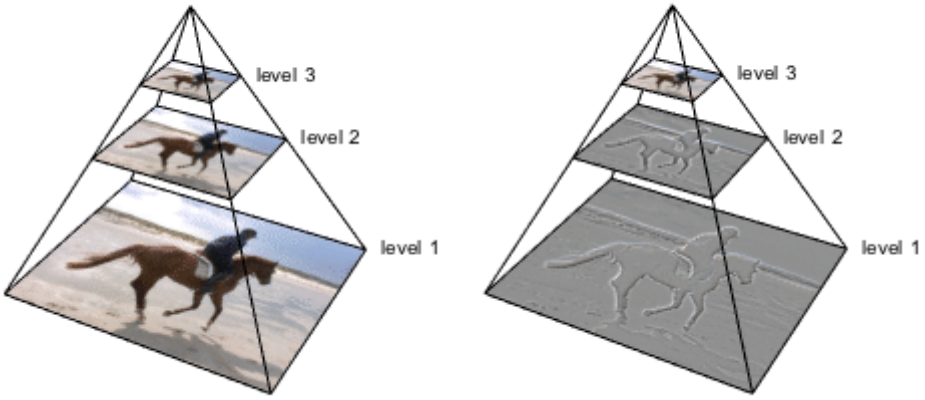
Tarea 1 EL7008 – Primavera 2019
Pirámides de Gauss y Laplace

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: Martes 06/08/2019
Fecha entrega: Miércoles 21/08/2019

El objetivo de esta tarea es implementar el cálculo de pirámides de Gauss y Laplace de una imagen, y luego reconstruir dicha imagen a partir de las pirámides. A continuación se hará una descripción breve de las pirámides.



Gaussian pyramid Laplacian pyramid

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.

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:

```
G1 = imagen_original
Para i en el rango 2 hasta N:
  imagen_suavizada = blur(Gi-1)
  imagen_submuestreada = subsample(imagen_suavizada)
  Gi = 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
```

Reconstrucción

Finalmente, 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
```

El código se debe programar en C++ usando OpenCV. En esta tarea, las imágenes a color deben ser transformadas a escalas de grises para simplificar la implementación.

Se entrega un código base que compila y 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 la tarea, se deben realizar los siguientes procedimientos:

A. Cálculo de pirámides de gauss:

1. Programar una función que reciba una imagen de entrada en escala de grises y una máscara, calcule la convolución entre ambas y genere una imagen de salida. No se debe usar funciones de OpenCV que calculen directamente convoluciones.
    a. Función: `Mat convolution(Mat input, Mat mask)`

2. Programar dos funciones: una que genere una máscara gaussiana horizontal, y otra que genere una máscara gaussiana vertical. Ambas funciones deben recibir la desviación estándar de la gaussiana (parámetro sigma) y el tamaño de la ventana (unidimensional). Se debe notar que el resultado debe normalizarse para que la suma de los valores dentro de la máscara sean iguales a 1.
    a. Función 1: `Mat compute_gauss_horiz(double sigma, int width)`
    b. Función 2: `Mat compute_gauss_vert(double sigma, int height)`
3. Programar una función que permita suavizar (usando gaussianas) una imagen. Debe recibir una imagen de entrada, la desviación estándar y el ancho de la ventana
    a. Función: `Mat do_blur(Mat input, double sigma, int height)`
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.
    a. Función: `Mat subsample(Mat input)`
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)
    a. Función: `vector<Mat> compute_gauss_pyramid(Mat input, int nlevels)`
6. Probar el sistema de cálculo de pirámides sobre 4 imágenes entregadas, guardando las imágenes de las pirámides (ya implementado).

B. Cálculo de pirámides de Laplace:

1. Programar una función que permita restar dos imágenes.
    a. Función: `Mat subtract(Mat input1, Mat input2)`
2. Programar una función que, a partir de una imagen, calcule una pirámide de Laplace, que tenga 5 niveles en total.
    a. Función: `vector<Mat> compute_laplace_pyramid(Mat input, int nlevels)`
3. Programar una función que permita guardar las imágenes de la pirámide de Laplace.
    a. Función: `void save_laplace_pyramid(vector<Mat> pyramid)`
4. Probar el sistema de cálculo de pirámides sobre 4 imágenes entregadas, guardando las imágenes de las pirámides. Se debe notar que las pirámides de Laplace pueden contener valores pequeños y también negativos. Esto se puede manejar aplicando algún tipo de normalización a las imágenes, pero sólo al momento de guardarlas (es decir, este proceso no debe afectar la pirámide que se usará para la reconstrucción).

C. Reconstrucción de imágenes:

1. Programar una función que permita sumar dos imágenes.
    a. Función: `Mat add(Mat input1, Mat input2)`
2. Programar una función que permita duplicar el tamaño de una imagen, interpolando los puntos intermedios.
    a. Función: `Mat upsample(Mat input)`
3. Programar una función que reciba una pirámide de Laplace y el último piso de una pirámide de Gauss. La función debe entregar una reconstrucción de la imagen original a partir de las pirámides.
    a. Función: `Mat reconstruct(vector<Mat> laplacepyramid)`
4. Probar el sistema de reconstrucción usando las 4 imágenes entregadas, guardando la imagen resultante.

Se entrega un proyecto que provee una funcionalidad básica. De este modo, el alumno debe enfocarse en programar los algoritmos pedidos y hacer las pruebas solicitadas.

El código a implementar debe basarse en el código base entregado.

El informe debe contener como mínimo: introducción, marco teórico (descripción de los algoritmos), partes relevantes del código dentro de cada sección del informe, resultados (mostrando las imágenes resultantes), análisis de los resultados y conclusiones generales.

Los informes y los códigos deben ser entregados en el día miércoles 21 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: 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 de las partes importantes del código en las secciones del informe correspondientes, así como la prolijidad y calidad del mismo.

Se agrega una pauta que indica la estructura esperada del informe.

Nota 1: Algunas cosas deben ser modificadas dependiendo de la versión de OpenCV

OpenCV 2	OpenCV 3 y 4
CV_MINMAX	NORM_MINMAX
CV_BGR2GRAY	COLOR_BGR2GRAY

Nota 2: ejemplos de imágenes de salida esperadas:

Gauss (5 pisos):

Laplace (5 pisos):