Métodos Computacionales

Taller 4 — 2018-10

La solución a este taller debe subirse por SICUA antes de las 5:00PM del lunes 16 de abril del 2018. Si se entrega la tarea antes del lunes 2 de abril del 2018 a las 11:59PM los ejercicios se van a calificar con el bono indicado.

(10 puntos) Los archivos del código deben estar en un único repositorio `NombreApellido_taller4`, por ejemplo si su nombre es Emma Goldman el repositorio debe llamarse `EmmaGoldman_taller4`. Al clonarlo debe crearse la carpeta `EmmaGoldman_taller4` con tres carpetas: `punto_1`, `punto_2` y `punto_3`

En la implementación principal de los algoritmos solicitados la copia y reutilización de código de cualquier fuente de internet (incluído el repositorio del curso) deja la nota en cero.

Todas las respuestas deben ser escritas en C++.

1. **Suave**

   (30 (35) puntos) Escriba un código para aplicar un suavizado gaussiano sobre una imagen de formato `png` de entrada.

   El ejecutable debe poder llamarse como `./suave imagen.png n_pixel_kernel`, donde `imagen.png` es el nombre del archivo de entrada y `n_pixel_kernel` es el ancho de la gaussiana del suavizado medida en pixeles. El resultado se debe guardar en el archivo `suave.png` con la imagen ya suavizada.

2. **Filtro**

   (30 (35) puntos) En clase trabajamos el filtrado de una señal unidimensional quitándole las frecuencias altas y dejando las frecuencias bajas. Ahora usted va a intentar algo similar con una imagen. Escriba un programa que haga el filtrado de una imagen de dos maneras. La primera que deje pasar las frecuencias bajas; la segunda que deje pasar las frecuencias altas. En ambos casos implemente un filtro suave.

   Ver la siguiente referencia: `http://paulbourke.net/miscellaneous/imagefilter/`.

   El ejecutable debe poder llamarse como `./filtro imagen.png alto` o `./filtro imagen.png bajo`. donde `imagen.png` es el nombre del archivo de entrada y `altas/bajas` marca el tipo de frecuencias que deja pasar. El resultado se debe guardar en el archivo `altas.png` o `bajas.png` con la imagen resultante.

3. **Fourier no uniforme**

   (30 (35) puntos) El algoritmo que vimos en clase para la transformada de Fourier discreta es aplicable cuando los datos están distribuídos uniformemente en el tiempo.

   Ahora usted va a implementar un algoritmo que funciona cuando los puntos no están distribuídos de manera uniforme. Para esto va a construir primero el polinomio de Lagrange correspondiente a los N pares de puntos $t_i, x(t_i)$. Con esto va a construir un nuevo conjunto de N pares de puntos que si se encuentran uniformemente distribuidos en t para calcular la transformada de Fourier.

   El ejecutable debe poder llamarse como `./fourier datos.txt` donde `datos.txt` es el nombre del archivo de entrada con dos columnas: la primera corresponde al tiempo, la segund a la

función del tiempo. El resultado se debe guardar en el archivo `tranformada.txt` donde se guarda el conjunto de N valores complejos en tres columnas: la primera corresponde a la frecuencias, la segunda a la parte real de la transformada y la tercera a la parte imaginaria.

Nota: Utilice esta libreria `https://github.com/glennrp/libpng` para leer y escribir las imágenes. Para la transformada de Fourier debe escribir su propia implementación. En todos los casos el ejecutable debe poder crearse con el comando `make`.