Taller de introducción a la programación: Trabajo Practico 1

Ph. D. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Computacion.

28 de abril de 2022

El presente trabajo práctico pretende introducir al estudiante en el proceso de análisis, diseño e implementación de un sistema de software orientado objetos, usando recursividad de listas y conceptos elementales de manipulación de matrices.

- **Modo de trabajo**: en grupos de 3
- **Fecha de entrega:** Domingo 15 de mayo del 2022
- **Tipo de entrega:** digital, por medio de la plataforma TEC-digital.

Parte I
Motivación

Los sistemas modernos electrónicos y de computación reciben alimentación de datos a partir de diversos tipos de sensores desde el mundo real. Es común la implementacion de sistemas con sensores de diversos tipos, por ejemplo sensores de radiofrecuencia, infrarrojos utilizados para las tele-comunicaciones, hasta sensores más sofisticados, como las cámaras digitales.

Un problema muy frecuente con el cual los diseñadores de modernos sistemas digitales se enfrentan es el problema del **ruido.** Los sensores al estar expuestos directamente al ambiente para medir las diferentes magnitudes físicas (luz, ondas electromagnéticas, fuerza, etc.), muchas veces terminan «percibiendo» distorsiones en la señal sensada por diversas razones. Entre ellas podemos citar el ruido térmico que afecta los componentes electrónicos más sensibles, difíciles condiciones de la señal a sensar (baja amplitud de la magnitud física a medir, interferencia de otras señales, etc.), entre muchas otras. Existen diversos tipos de ruido, entre uno de los más estudiados en la literatura es el ruido impulsivo o de «sal y pimienta». Este tipo de ruido se presenta cuando existen pérdidas de información en la transmisión imágenes digitales, o fallos de los dispositivos electrónicos que convierten la señal analógica de cada uno de los sensores de luz que componen la matriz de sensores en la cámara a un valor digital. La Figura 1 muestra una imagen afectada por este tipo de ruido.

Existen otros tipos de ruido, como el ruido Gaussiano o el «speckle», el cual es muy frecuente en imágenes tomadas por los los radares de apertura sintética (SAR, por sus siglas en inglés), usados para sensar diferentes tipos de objetos terrestres (cuerpos acuosos, vegetación, construcciones, etc.). Una imagen tomada por un SAR de ejemplo se observa en la Figura 2.

Figura 1: Ejemplo de una imagen afectada por ruido impulsivo o de tipo «sal y pimienta».

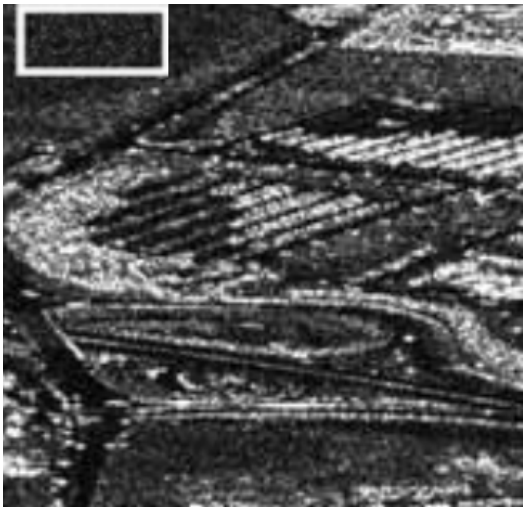

Figura 2: Ejemplo de una imagen generada por un SAR (Synthetic aperture radar).

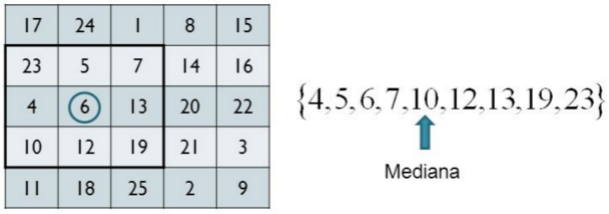

Figura 3: Cálculo de la mediana, en una ventana de 3×3.

Figura 4: Ejemplo de disminución del ruido impulsivo con el filtro de medianas.

Parte II

Atenuación del ruido con un filtro digital

Para el presente trabajo práctico, se le propone diseñar un programa que elimine o al menos atenúe el efecto del ruido impulsivo, comentado en la sección anterior. Para ello, el programa implementará el concepto de un filtro digital, el cual consiste en modificar la señal original «filtrando» o eliminando atributos de la imagen que posiblemente correspondan a elementos asociados con ruido, y por lo tanto, indeseados. El programa recibirá imágenes en escala de grises, es decir, con cada valor de pixel en el intervalo$[0 - 255]$. El enfoque sugerido para implementar el filtro digital, se basa en el concepto que inspira toda una familia de filtros, llamado filtros de orden de rango. Tales filtros recorren cada pixel en la imagen, y por cada uno, se analiza un vecindario o ventana.

En la imagen de la Figura 3, se muestra la definición de una ventana de 3×3 pixeles, para en este caso el pixel de valor 6. Los filtros de orden de rango básicamente proponen entonces por cada ventana, tomar todos los pixeles y ordenarlos de manera ascendente, para entonces tomar como nuevo valor de la imagen de salida, el valor de en «medio» o lo que estadísticamente se le refiere como la **mediana**. De esta manera, la imagen de salida quedará reconstruída a partir de las medianas de todas las ventanas definidas a partir de cada uno de los pixeles de la imagen de entrada. El algoritmo de filtrado recibirá entonces la imagen a filtrar en escala de grises, y el tamaño del vecindario o ventana a examinar por cada pixel en la imagen. Para ordenar los pixeles en la imagen, se propone investigar e implementar al menos **dos algoritmos de ordenamiento basado en recursividad de pila**. Para la modificacion de la matriz, usar **recursividad de cola**.

No se permite tomar código directamente de la red o llamar a librerías externas para realizar el ordenamiento de los pixeles o el filtrado de la imagen.

Un ejemplo de la salida del filtrado de medianas se muestra en la Figura 4.

Parte III

Requerimientos del programa

El programa debe ser desarrollado en Python, usando el **paradigma orientado a objetos (POO)**, usando **solamente recursividad**, y además debe cumplir los siguientes requerimientos funcionales (**usando numpy**):

1. Preguntar al usuario la ruta y nombre de la imagen usada de entrada, además de la ruta donde se guardará la imagen procesada (salida).
2. Preguntar al usuario el ancho de la ventana a utilizar.

Opcionalmente el programa puede:

1. Implementar al menos los siguientes algoritmos de ordenamiento, basados en recursividad (de cola, pila o simple), según una investigación previa y debidamente documentada.
   a) Algoritmo de seleccion
   b) Algoritmo quick-sort
2. De forma opcional y por **10 puntos adicionales**, investigar e implementar un algoritmo de ordenamiento adicional a los vistos en clase.
3. Presentar al usuario el tiempo de ejecución, en milisegundos (*ms)* de la operación de filtrado.

Finalmente, además del código fuente del programa, es requisito entregar la documentación pertinente del programa. El formato del mismo será previamente especificado, posiblemente usando la plataforma LaTeX (mediante el programa Lyx).

No se permite tomar código de la red para implementar las funcionalidades principales o llamar a librerías externas para realizar el ordenamiento de los pixeles o el filtrado de la imagen.

1. Implementación

Para construír este y los demas proyectos, es necesario utilizar el siguiente conjunto de herramientas:

1. Star UML, para la diagramación de los casos de uso, y el diagrama de clases.
2. El paquete *Anaconda* (Python 3.5) el cual incluye la librería *scipy* necesaria para la manipulación de imágenes (se usará el IDE Spyder).
3. Lyx junto con LaTeX para la elaborar la documentación externa.

2. Evaluación

El siguiente corresponde al esquema de evaluación:

1. Implementación (70 %)
   a) Correcto funcionamiento en las pruebas funcionales (45 %)
   b) Uso correcto de las prácticas recomendadas de programación en el curso: documentación interna, variables y métodos signiticativos, etc. (20 %)
2. Documentación externa (30 %)

a) Seguimiento de la estructura propuesta y adecuados comentarios y redaccion (siguiendo las etapas del ciclo de resolucion de problemas) (15 %).

b) Detalle una seccion de pruebas donde se compara el tiempo de ejecucion total para filtrar una imagen de los dos enfoques implementados (5 %), **usando una tamanio de ventana pequenio (**$K = 3$**) y otro grande** $K = 21$.

c) Correcta redacción, uso de figuras y citas bibliográficas (10 %)