1. Descripción de funciones utilizadas

- cargarImagen(img): Función encargada de cargar la imagen que se desea procesar.
- acomodarFilasImg(lista): Se encarga de acomodar las filas de la matriz para lograr transformarla a una imagen.
- TransformarImagenAArreglo(img): Transforma una imagen en un arreglo para su manipulación matemática a la hora de la aplicación de algún filtro.
- mostrarImagen(original, filtrada, titulo): muestra gráficamente la comparación entre la imagen original y la imagen procesada por el filtro con el fin de que el usuario note los cambios. También le agrega un nombre a la ventana por medio del parámetro “titulo”.
- rankFiltro(img, width, height): Agrega el filtro *Rank* al arreglo de la imagen original con SciPy, y posteriormente la redimensiona a su tamaño original con el ancho y largo respectivo.
- TransformarArregloAImagen(imgFiltrada): Transforma el arreglo que representa la imagen procesada o filtrada a un objeto imagen, con el fin de guardarla en un archivo en disco.
- guardarImagen(nuevaImg, dir, nombre): Guarda la imagen en un archivo con formato png, en el directorio indicado, con el nombre indicado por el usuario.
- evaluaPixel(eval_pixel, otros_pixeles): Evaluación de cada pixel de la imagen original con los pixeles que lo rodean. Mediante un vector de números constantes realiza la función de convolución para calcular el nuevo pixel RGB.
- setPixeles(pixeles, esUno, ancho): Mediante la bandera esUno, toma los pixeles y a cada uno les suma una unidad. Si la bandera está en falso, entonces inicializará los pixeles a su estado inicial.
- desplazarPixeles(pixeles): Le suma 4 unidades a cada pixel, con el fin de ir evaluando según esa cantidad de pasos.
- gaussBlurFiltro(img): Función que realiza todo el proceso de aplicación del filtro de Gauss a la imagen. Calcula los nuevos pixeles mediante la función de convolución, manipula los pixeles que están alrededor del pixel inicial y va saltando según el algoritmo. Nótese que el filtro de Gauss está programado pensando en un kernel de 5x5.

2. Bibliotecas

- Numpy

Esta biblioteca se utilizó para realizar transformaciones de tipos de imágenes a arreglos de números, con el fin de manipular numéricamente la matriz que representa la imagen original.

- Scipy

Biblioteca utilizada dentro del programa para aplicar un filtro de imagen que contiene *ndimage*. Además, se utilizó *misc* para redimensionar la imagen editada por el filtro, ya que automáticamente se cambiaba, por lo que se optó por utilizar esta herramienta.

- Matplotlib

Está biblioteca fue utilizada con el objetivo de mostrar los resultados de las ediciones de las imágenes de manera gráfica, para que el usuario verifique de forma más flexible el resultado.

- Pillow

Biblioteca utilizada para cargar una imagen al programa desde un archivo, así como transformar un arreglo de números que representan la imagen editada a una imagen como tal, para posteriormente guardarla como un archivo en algún directorio, si así se quisiera.

3. Errores encontrados

Durante el proceso de implementación del programa, se encontraron diversos errores los cuales se solucionaron por medio de documentación externa. Los errores más relevantes se describen a continuación.

- **Error #1**: al abrir una imagen mediante un string con Image.open() con formato jpg.
  Solución: Cambiar el tipo de imagen a utilizar, con formato png.

- **Error #2**: Error de codificación caracteres del código.
  Solución: Agregar una línea al principio del código fuente del programa (“#coding=utf-8”). La solución se encontró en el siguiente link:

https://stackoverflow.com/questions/26899235/python-nltk-syntaxerror-non-ascii-character-xc3-in-file-senitment-analysis

- **Error #3**: Las dimensiones de la matriz de la imagen sobrepasan los 32 componentes (ndarray).
  Solución: Utilizar array en vez de ndarray. La solución se encontró en: https://stackoverflow.com/questions/17688094/numpy-array-sequence-too-large

- **Error #4**: Transformar un array de numpy a Image.
  Solución: Utilizar la función del objeto Image llamada Image.fromArray().

- **Error #5**: Guardar una imagen en ciertos directorios del sistema de archivos, ya que faltaban permisos.
  Solución: no guardar archivos en LocalDisk(C:), en los demás directorios sí es posible guardar una imagen.

- **Error #6**: No reconocía el formato de las imágenes por el tipo de las mismas.
  Solución: cambiar el código o tipo de la imagen de uint16 a uint8.

4. Referencias

- The Scipy community. (2008). Multi-dimensional image processing. 4/7/2017, de The Scipy community Sitio web: https://docs.scipy.org/doc/scipy/reference/ndimage.html

- Fredrik Lundh, Alex Clark and Contributors. (1995). Image Module. 4/7/2017, de Pillow Sitio web: http://pillow.readthedocs.io/en/4.1.x/reference/Image.html