Evaluación 3 – Image Restoration

1) Introducción:

Se busca corromper una imagen aplicando desenfoque o blur y añadiendo ruido, para luego aplicar un método de restauración numérica, basado en máxima verosimilitud.

2) Idea del código:

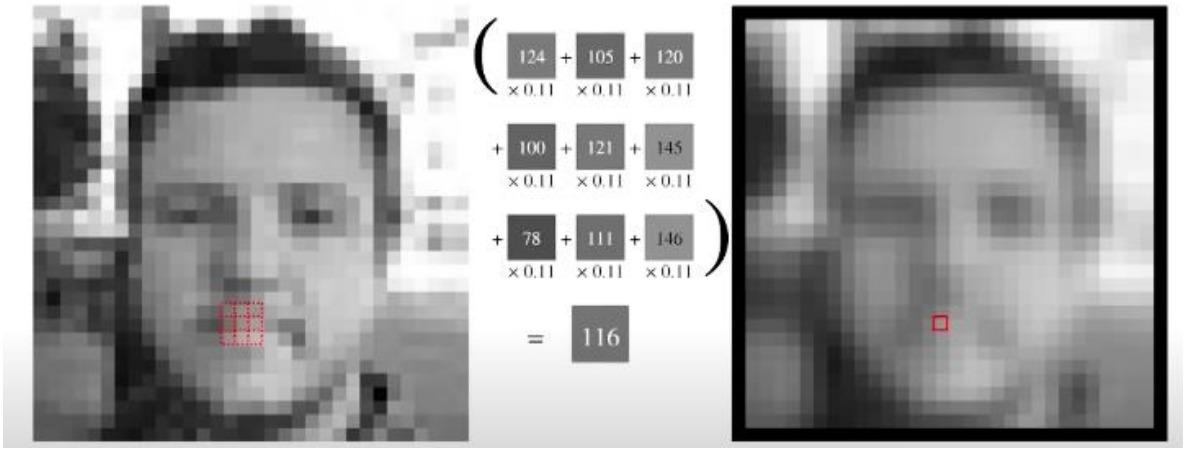
Se genera una imagen mediante "crearTxt.py", la cual será procesada por ImageProccesor, a la cual se le aplica ruido (R) y convoluciona con el kernel (una matríz de tamaño 2M + 1 × 2M + 1), expresada de la siguiente forma:

$$I_{corrupted} = K * I_{real} + R$$

Convolución y Corrupción de Imagen:

Una convolución aplicada sobre una imagen no es más que ir jugando con los valores de los pixeles con un filtro o "Kernel", cada nuevo pixel de la nueva imagen convolucionada con el kernel o matríz de números sobre la imagen original, donde multiplicaremos y sumaremos los valores de cada pixel vecino para obtener así el nuevo valor del pixel, si desplazamos este filtro en toda la imagen obtendremos una imagen nueva, el resultado de esta depende de los valores del kernel, por ejemplo para "distorcionar" o agregar Blur en una imagen se debe hacer el promedio de cada pixel del kernel, por ejemplo, de la siguiente forma:

Obteniendo:



Algoritmo de Restauración:

Se realiza mediante el algoritmo **frprmn** o **Fletcher Reeves Conjugate Gradient Method** utilizado para minimizar el valor de X^2, se calcula el máximo descenso o dirección contraria al gradiente (-G). Recordando que el gradiente obtiene la dirección de máximo crecimiento y el inverso o negativo, obtiene la dirección de máximo decrecimiento. Este método es iterativo calculando la suma de la variación en el siguiente paso.

Métodos por utilizar para la el algoritmo frprmn (): (Para más detalles revisar los comentarios del código).

- **brent:** Utiliza el método de la bisección y secante para hallar un mínimo de una función.
- **mnbrak:** Busca la cuesta abajo o gradiente evaluada en los valores iniciales.
- **linmin:** Retorna el mínimo de una función dado un arreglo de "n" dimensiones.

Imagen Final Restaurada:

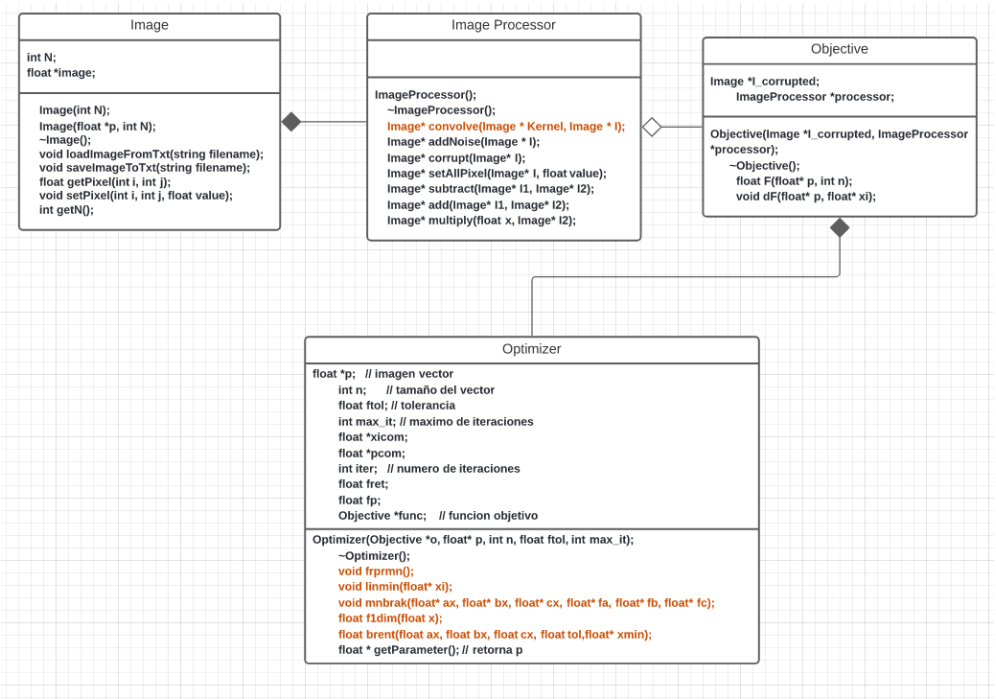
3) **Estructura del Código:**

Clases:

- **Image:** Utilización de la imagen: Obtiene valor de pixel, modifica pixel, guarda imagen en txt y carga imagen en txt.

- **ImageProccesor:** Clase que contiene todo lo utilitario para realizar operaciones con la imagen (convolución, ruido, adición, producto, substracción), por lo tanto, se encuentran los métodos de corrupción de la imagen.
- **Objective:** Contiene las funciones a minimizar y el método correspondiente a calcular el gradiente de esta función, almacenado en un vector.
- **Optimizer:** Clase correspondiente al algoritmo del gradiente conjugado y a la restauración de la imagen a partir de la corrupción provocada en ella por las clases anteriores.

* Se incluye diagrama UML para complementar y visualizar la relación entre las clases.



4) **Eficiencia:**

Solo se evita realizar la convolución y la substracción 2 veces en la clase Objective, se guarda en un atributo de la misma clase.
Todo lo demás en el código fue extraído del libro Numerical Recipes y no se optimizó ningún método, solo se usa el código original.

5) **Fuentes:**

- *Pablo Román (2022). Classroom. Enunciado Evaluación 3:* https://classroom.google.com/u/1/c/NDc5OTUzNzI0NDk3
- *William H. Press ... [and others]. (1992). Numerical recipes in C: the art of scientific computing. Cambridge [Cambridgeshire]; New York: Cambridge University Press.*
- *Wikimedia Foundation. (2022, January 27). Nonlinear conjugate gradient method. Wikipedia. Retrieved June 5, 2022, from https://en.wikipedia.org/wiki/Nonlinear_conjugate_gradient_method*
- *Convolución de imágenes – Dot csv* *https://www.youtube.com/watch?v=V8j1oENVz00&t=355s*