



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



VISIÓN ARTIFICIAL



Examen primer parcial

Alumno:

De La Huerta Avalos Gerardo Cristóbal

N. Boleta:

2021630243

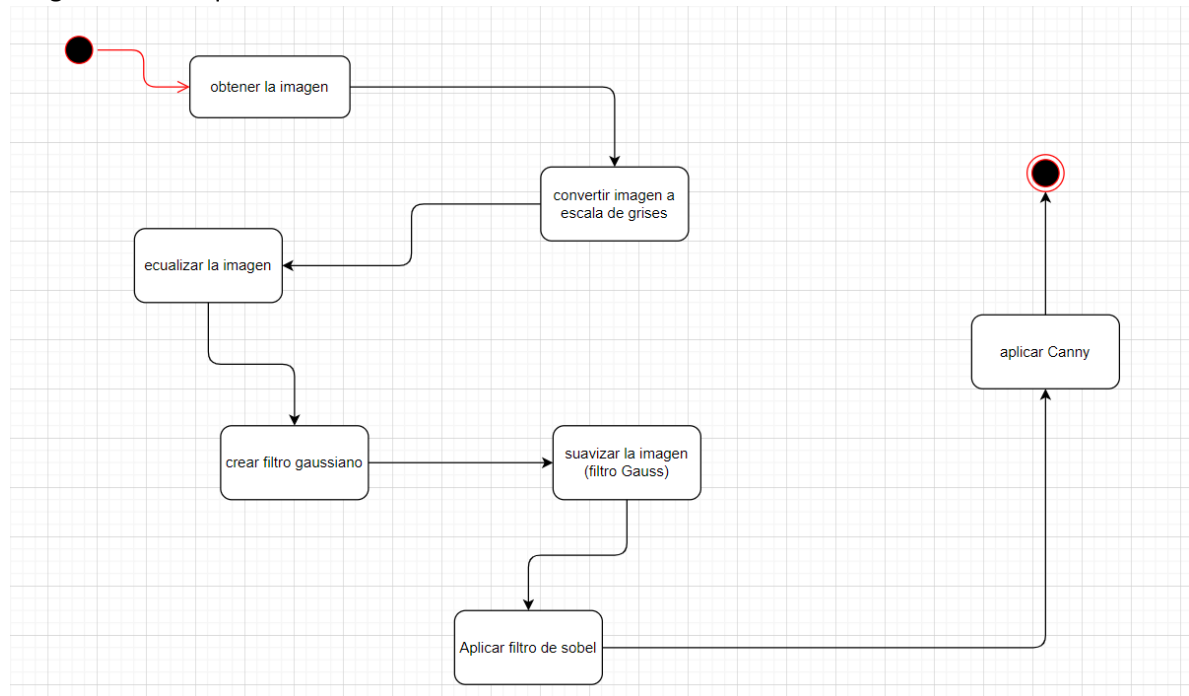
Profesor: **Octavio Sánchez**

Grupo: **5BM1**

Especialidad: **Ingeniería en Inteligencia Artificial**

Fecha: **04/11/2022**

Diagrama de bloques



Proceso:

Para llevar a cabo esta actividad – examen, comencemos por comentar que se hizo uso de programación estructurada y por bloques para poder llegar al resultado presente en el código fuente. Debido a lo mismo, el proceso expresado en este diagrama luce casi idéntico a los pasos en el main de source.cpp.

El procedimiento comienza con recibir la imagen inicial, que en nuestro caso, se encuentra en: src/lenna_main.jpg, para recibirla, simplemente obtenemos los datos de la imagen en una variable de tipo cv::Mat.

Una vez obtenida la imagen, realizamos la conversión a escala de grises, para efectos de esta práctica se optó por llevar a cabo la obtención de escala de grises por promedio. Se guarda la imagen en una nueva variable de tipo cv::Mat.

El siguiente proceso consiste en realizar la ecualización de la imagen, existen diversos métodos para ecualizar la imagen. Una manera de ecualizar manualmente la imagen consiste en obtener los valores del histograma de la imagen, realizar la sumatoria acumulada de las frecuencias de este, y después aplicar una fórmula similar a la siguiente:

$$\text{NuevoPixel} = ((\text{sumaHistograma}[R]-1) / \text{sumaHistograma}[255]) * 255$$

Nótese que, en el desarrollo de la práctica, aplicamos esta fórmula multiplicando primero por 255 y después dividimos entre la sumaHistograma[255], esto debido a una anomalía, en la

que, valores muy pequeños como los obtenidos en la razón, automáticamente los manda a 0, y cambiar el orden resuelve esta anomalía.

Con esto, obtenemos una distribución más uniforme de los niveles de gris de la imagen, y a su vez, obtenemos una imagen con un alto y posible mejor contraste.

El siguiente paso consiste en crear el filtro gaussiano, el filtro gaussiano es un filtro dinámico, el cual puede adoptar distintas dimensiones, dependiendo de las necesidades del usuario. Los valores del filtro, a partir del centro del kernel (filtro, o mascara) toman valores en función de la fórmula de gauss:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Para esta práctica se hace uso de apuntadores de apuntadores, para obtener con ello una matriz que sirva como kernel. El resto, consiste en aplica la fórmula de gauss con base en las coordenadas del filtro a partir del centro de este.

Una vez obtenido el filtro gaussiano, el siguiente paso consiste en aplicar el filtro a la imagen en blanco y negro, para este paso se lleva a cabo la convolución del kernel con la imagen, en la misma, se multiplican valores, se suman todos los resultados, y se concentra el resultado en un punto central. Es por ello por lo que este filtrado puede generar una pérdida de bordes, y para tratar de evitar esto, se optó por expandir la matriz de la imagen, y así evitar que la resultante cuente con dimensiones distintas a la original.

El siguiente paso en la práctica es aplicar el filtro de Sobel sobre la imagen original a blanco y negro, para este caso, el filtro de Sobel se realiza con dos filtros: Gx y Gy, los cuales, presentan características similares, pero no son los mismos filtros. Gracias a que estamos aplicando programación estructurada, podemos reciclar el código de la convolución, únicamente cambiando el kernel por Gx y Gy. Y una vez que obtenemos las dos imágenes resultantes, aplicamos $|G|$, el cual es básicamente $\sqrt{\text{pow}(Gx,2) + \text{pow}(Gy,2)}$.

En el despliegue de la imagen y sus diferentes filtros, se optó por desplegar adicionalmente los filtros Gx y Gy con fines ilustrativos de lo que ocurre en la imagen utilizada tras aplicarle Sobel en X y Sobel en Y

Finalmente, el último paso, es aplica el método de Canny para la obtención de bordes, para este último caso, es necesario aplicar una serie de pasos que requieren de reutilizar el código que ya hemos realizado en los pasos anteriores:

Es necesario ecualizar la imagen original, posteriormente aplicar un suavizado sobre la imagen, al resultado aplicamos los filtros de Sobel en Gx y Gy. Una vez realizado esto, obtenemos $|G|$ que es una distancia euclidiana entre Gx y Gy, la formula se ha expresado en un

párrafo anterior del presente documento. Así mismo es necesario obtener la dirección del gradiente, lo cual es básicamente:

$$\text{Arc tan}(G_y/G_x)$$

Una vez obtenido todo lo anterior, la obtención del borde la obtendremos aplicando el producto de $|G|$ con la dirección del gradiente.

Para obtener un resultado donde se pueda visualizar mejor el borde de la imagen, el autor del presente documento sugiere un valor de sigma igual a 13.

Finalmente, apreciar mejor el filtro gaussiano, es preferible usar una sigma menor a 3.

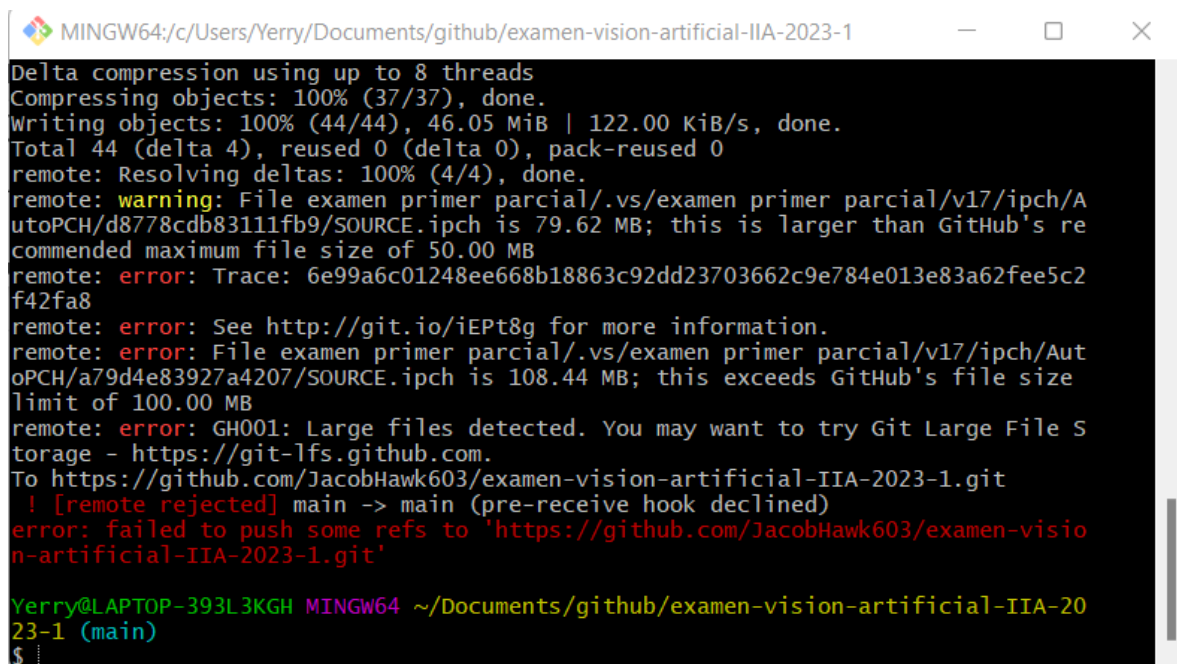
Nota adicional:

Todas las imágenes resultantes a excepción de la original a color y el filtro Canny, son filtros y procesos aplicados sobre la imagen original a color. En el caso del método de Canny, va siguiendo un proceso, es decir, que se ecualiza, se suaviza, se aplica Sobel sobre el resultado de la operación anterior.

Si lo desea el lector, puede modificar el main, para que cada proceso o filtro se aplique sobre una imagen resultante.

Nota importante:

Github no me acepta el proyecto completo, aquí la prueba:



```
MINGW64:/c:/Users/Yerry/Documents/github/examen-vision-artificial-IIA-2023-1
Delta compression using up to 8 threads
Compressing objects: 100% (37/37), done.
Writing objects: 100% (44/44), 46.05 MiB | 122.00 KiB/s, done.
Total 44 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
remote: warning: File examen primer parcial/.vs/examen primer parcial/v17/ipch/AutoPCH/d8778cdb83111fb9/SOURCE.ipch is 79.62 MB; this is larger than GitHub's recommended maximum file size of 50.00 MB
remote: error: Trace: 6e99a6c01248ee668b18863c92dd23703662c9e784e013e83a62fee5c2f42fa8
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File examen primer parcial/.vs/examen primer parcial/v17/ipch/AutoPCH/a79d4e83927a4207/SOURCE.ipch is 108.44 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/JacobHawk603/examen-vision-artificial-IIA-2023-1.git
! [remote rejected] main -> main (pre-receive hook declined)
error: failed to push some refs to 'https://github.com/JacobHawk603/examen-vision-artificial-IIA-2023-1.git'

Yerry@LAPTOP-393L3KGH MINGW64 ~/Documents/github/examen-vision-artificial-IIA-2023-1 (main)
$
```

Debido a lo anterior, es altamente probable que esté haya recibido un enlace de drive, o que haya buscado el enlace de drive que coloqué en el repositorio de git, con únicamente el código fuente del examen.

Debo de aclarar que, SI TENGO EL PAQUETE DE ESTUDIANTE, me aseguré de tenerlo, no obstante, este error ocurre siempre con los proyectos de VS que trato de subir completos a github. Disculpe las molestias que este error le causen, sin embargo, es mi obligación comentar que me está sucediendo esto.