

INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

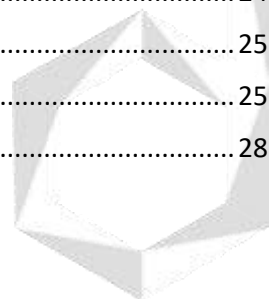
PROGRAMACIÓN: VISIÓN ARTIFICIAL

VISUAL STUDIO CODE & PYTHON 3.9.7: LIBRERÍA OPENCV

Visión Artificial con
Python: Librería OpenCV

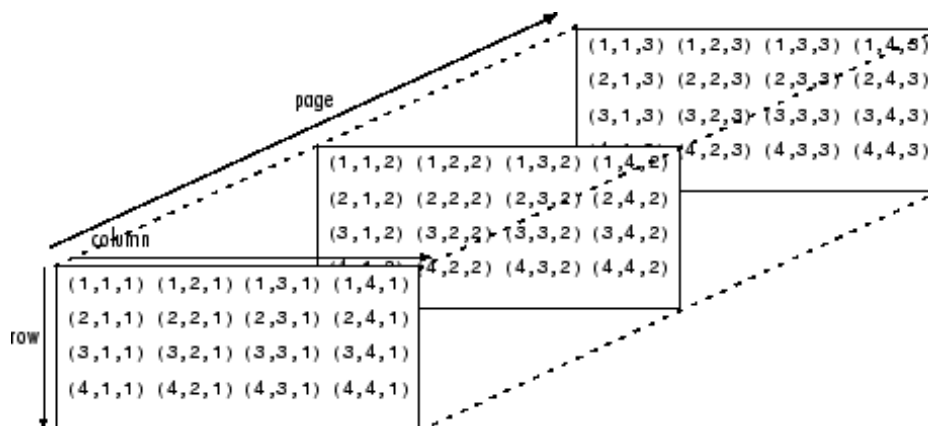
Contenido

| | |
|--|----|
| Introducción a la Visión Artificial | 2 |
| Análisis de Imagen: | 2 |
| 2.- Vecindarios: | 2 |
| 3.- Escala de Grises: | 3 |
| 3.- Histograma:..... | 4 |
| 3.- Brillo:..... | 5 |
| 3.- Contraste: | 5 |
| 3.- Complemento: | 7 |
| 4.- Umbralizar: | 7 |
| 4.- Adaptación Automática al Contraste:..... | 8 |
| 4.- Operaciones Lógicas Aritméticas:..... | 8 |
| Visión Artificial con Python | 9 |
| Filtros | 10 |
| Filtros Espaciales: | 10 |
| Convolución: | 11 |
| Filtros Lineales: | 11 |
| Ruido: | 13 |
| Características de una Imagen (Shape):..... | 14 |
| Rejillas (Numpy): | 14 |
| Filtro de Máximos, Mínimos y Media (Scipy):..... | 15 |
| Filtro de Canny: | 17 |
| Algoritmo de Harris:..... | 18 |
| Detector de Harris:..... | 18 |
| Convolución Kernel: | 19 |
| Operador Sobel: | 21 |
| Detector Canny: | 22 |
| Filtro Suavizado, Algoritmo de Harris, Filtro de Gauss, Detección de Esquinas: | 23 |
| Obtención de Color de una Imagen: | 24 |
| Recorte de una Imagen: | 25 |
| Transformada de Hugh - Detección de Líneas: | 25 |
| Identificación de Rostros – OpenCV – Viola Jones y Patrones Binarios:..... | 28 |



Introducción a la Visión Artificial

Las imágenes digitales son una matriz 3D de tres capas, la R, G y B, estas son analizadas desde la primera posición de su píxel, que siempre se considera como el píxel superior izquierdo.



Análisis de Imagen:

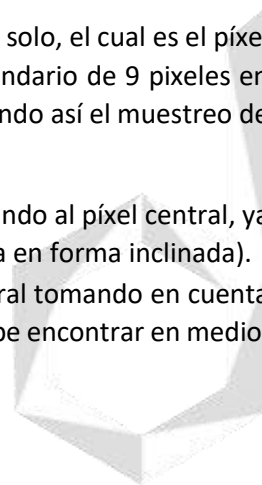
Para analizar una imagen se deben seguir los siguientes pasos, considerando que el píxel es la unidad mínima de una imagen digital:

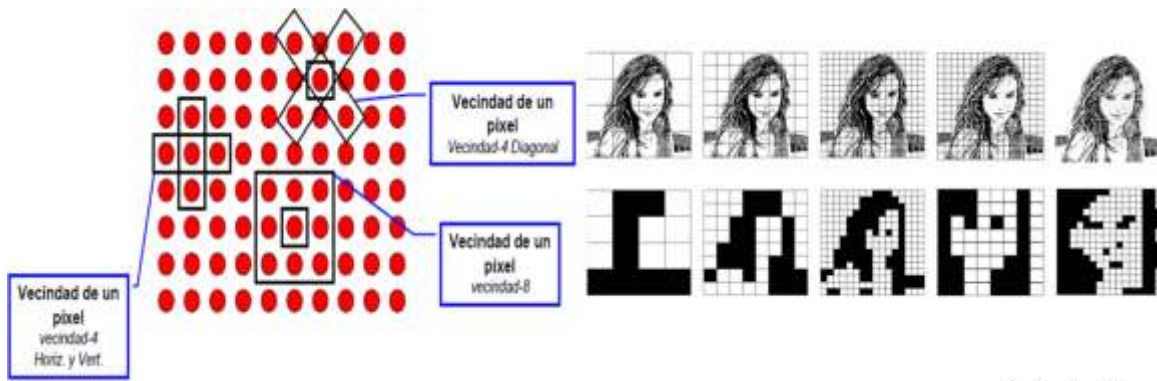
1. **Captura de imagen:** Se refiere a la recopilación de imágenes por medio de fotos o video.
2. **Muestreo de imagen:** Se refiere al análisis de los píxeles de la imagen y sus vecindarios.
3. **Cuantificación:** Se convierte la imagen a una escala de grises y se analiza su histograma para identificar el umbral que se debe usar para identificar una forma específica en la imagen.
4. **Codificación:** Se aplica filtros dependiendo de la condición de brillo y contraste de la imagen original, para finalmente binarizarla y de esta manera separar el fondo del objeto que se quiere identificar.

2.- Vecindarios:

Los píxeles tienen vecindarios, vecindario se refiere a los píxeles que rodean a uno solo, el cual es el píxel de análisis, existen dos tipos, considerando una matriz de 3X3 para crear un vecindario de 9 píxeles en total y teniendo en cuenta que el píxel central es el que se está analizando, realizando así el muestreo de la imagen:

- **Vecindarios de 4 píxeles:** Considera los 4 píxeles que se encuentren rodeando al píxel central, ya sea en forma de cruz (osea en forma horizontal) o en forma de tache (osea en forma inclinada).
- **Vecindarios de 8 píxeles:** Considera los 8 píxeles que rodeen al píxel central tomando en cuenta que la vecindad se compone de 3 filas y 3 columnas donde el central se debe encontrar en medio.





3.- Escala de Grises:

Las imágenes digitales en forma matemática son una matriz 3D que consta de tres capas de varias filas y columnas, dichas capas son las R, G y B. Esto implica que cuando se realiza el análisis de imágenes se deben procesar estas tres matrices sobrepuestas, para mejorar en la velocidad del análisis, se transforma a una escala de grises para que se analice una sola capa que represente las 3 anteriores.

Se cuenta con 3 tipos de imágenes:

- **RGB:** Matriz 3D con una matriz 2D que represente cada color primario RGB en la imagen digital.



- **Escala de grises:** Una sola matriz de dimensión 2D.



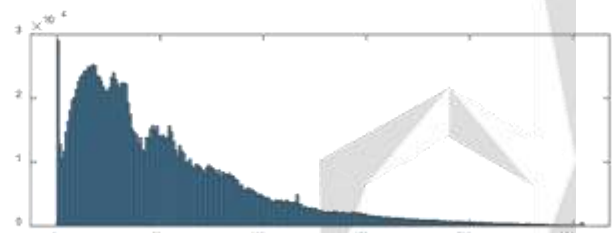
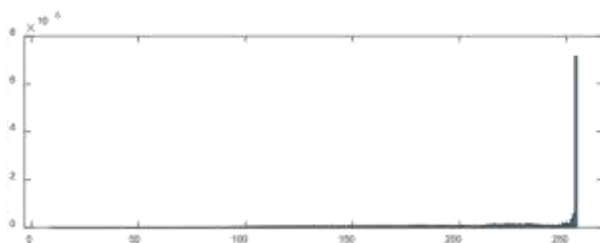
- **Imágenes binarias:** Matriz donde solo se tiene ceros y unos, osea blancos y negros intensos. Este tipo de imagen se usa para diferenciar objetos en una misma imagen.

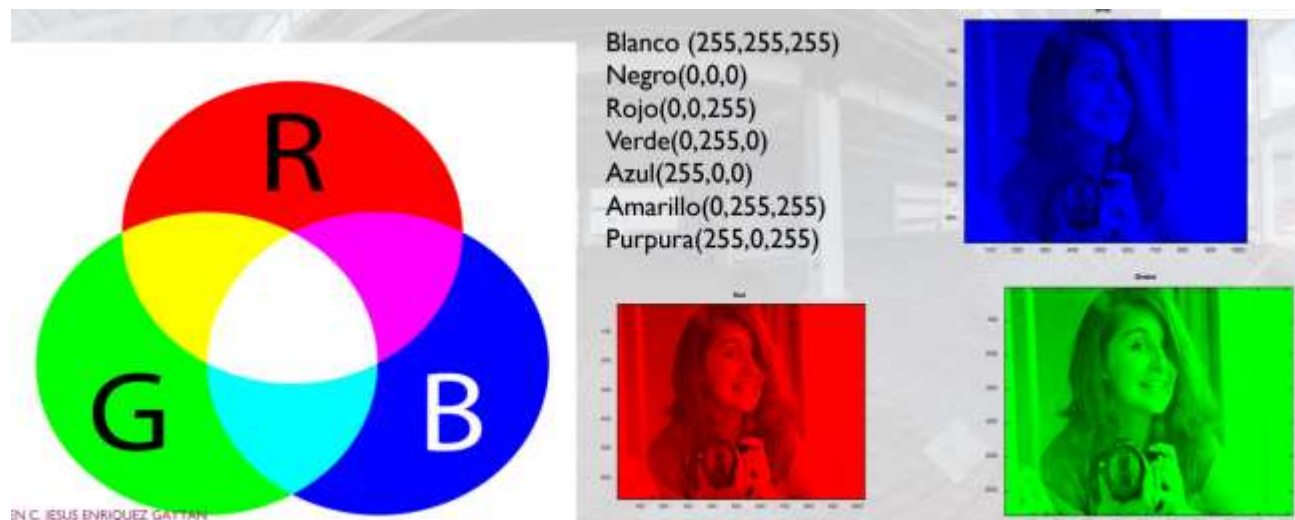


3.- Histograma:

Es una gráfica que representa los distintos tonos de Rojo, Verde o Azul en una imagen, en el eje horizontal se representa cada uno de los tonos del color yendo de 0 a 255 (llamado luminosidad) y en el eje vertical se representa el número de pixeles que corresponden a cada tono de cada capa (llamado porcentaje). El tono de cada color va de 0 a 255 porque cada color se representa con 8 bits, $2^8 = 256$, pero en este número también se considera el cero como valor. La forma de interpretar el resultado del histograma es la siguiente:

Cada capa RGB de la imagen cuando se descompone va de 0 a 255, esto es porque en cada capa mientras más se acerque al valor 255 (osea el blanco) es porque más presente se encuentra ese color en la imagen, por lo que en el histograma la gráfica que más se acerque al 255 en el eje horizontal es la que representa el color que más se encuentra presente en la imagen no importando su amplitud, ya que la amplitud representa cuantos pixeles cuentan con el mismo tono de cada color RGB o también se puede realizar para escala de grises, al usarse el histograma en una escala de grises se reconoce el brillo y contraste de la imagen.





3.- Brillo:

El brillo se refiere a la luminosidad de la imagen, osea que tan oscuros o claros son los pixeles que la conforman.

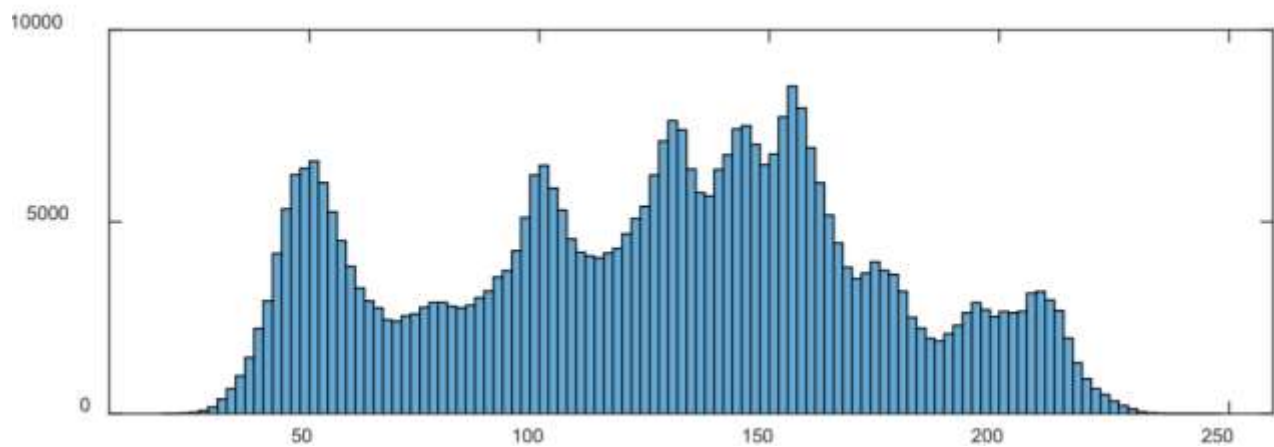
Se logra aumentar el brillo cuando se le suma a una imagen un número entre 1 y 255, si se quiere reducir el brillo se deberá restar en vez de sumar.

3.- Contraste:

El contraste no es otra cosa que el efecto que se produce al destacar un elemento visual en comparación con otro en una misma imagen. El contraste puede darse bien sea por la combinación de diferentes colores, intensidad de luces y sombras, diferencias de tamaño, textura, o cualquier otro elemento visual. Su efectividad depende directamente de los valores del histograma de la imagen.

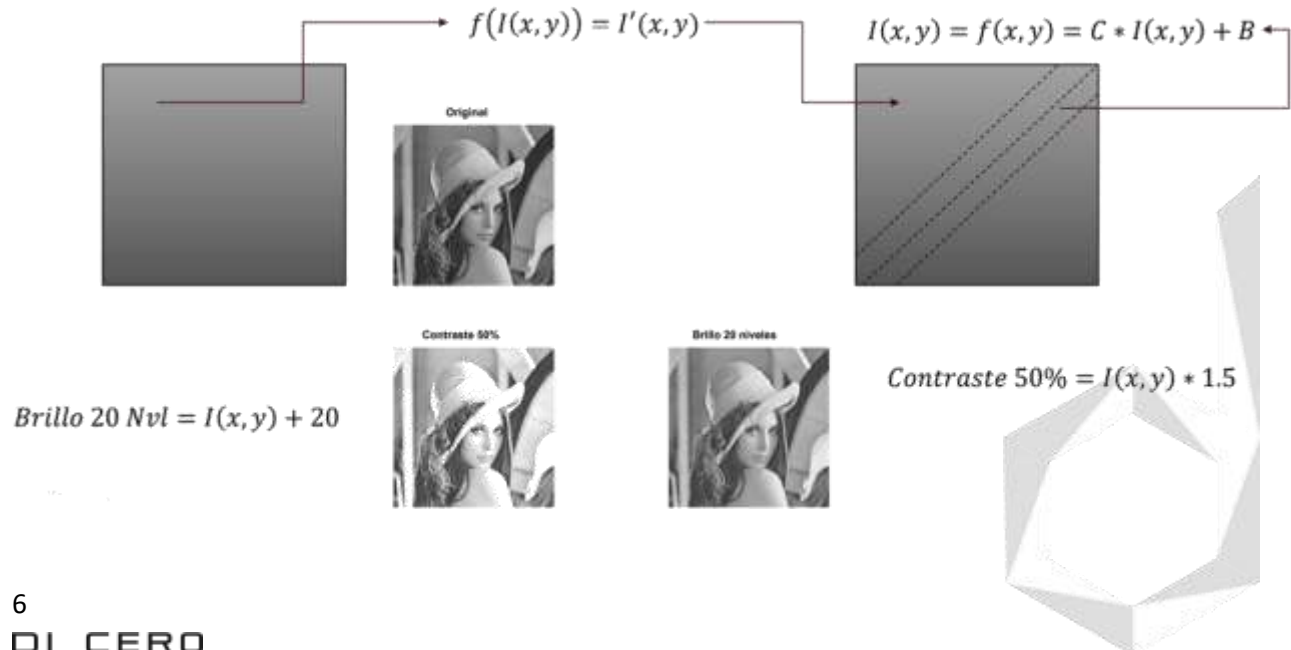
Se logra aumentar el contraste en el código cuando se multiplica la matriz que representa una imagen digital por un número entre 1 y 2, donde la parte decimal indicará el porcentaje de aumento del contraste, por ejemplo, si se multiplica una imagen por 1.5 se estaría aumentando en un 50% el contraste. Si se quiere reducir el contraste se deberá multiplicar por un número entre 0 y 1.

Para poder realizar un buen contraste se debe contar con una imagen cuyo histograma no esté muy cargado de pixeles blancos, osea con la mayoría de sus valores cercanos al valor 255 (del lado derecho de la gráfica) ni muy cargado de pixeles negros, osea muy cargado de valores cercanos al valor 0 (del lado izquierdo de la gráfica), la situación ideal para poder procesar y analizar de buena manera una imagen, es que el histograma de la imagen cuente con una distribución homogénea de pixeles, ni tan cargados del lado derecho (claros) ni del izquierdo (oscuros).



Si este no fuera el caso, se debería pasar la imagen por algún filtro para poder analizar la imagen y separar el fondo del objeto que se busca analizar, que es una operación muy común en visión artificial, para ello se declara una ecuación donde $f(x, y)$ es la matriz de la imagen, C es el contraste y B es el brillo:

$$I(x, y) = f(x, y) = C * I(x, y) + B$$



3.- Complemento:

Lo que hace es restar el valor máximo de la matriz menos la función de la imagen original y de esta manera se obtiene el inverso de la imagen original, en donde si antes había un color blanco, ahora habrá un color negro y viceversa:

$$f_{inv}(x, y) = f_{max} - f(x, y)$$

Original



Complemento de imagen



$$f_{inv}(P) = P_{Max} - P$$

4.- Umbralizar:

Cuando el histograma se obtiene de una imagen con escala de grises se puede indicar su contraste y brillo (iluminación), al hacer esto se puede umbralizar la imagen, osea indicar que, a partir de cierto valor de un tono de gris, se diferencie el fondo y el objeto que se quiere identificar en la imagen. Al umbralizar se crean imágenes binarias, donde al objeto normalmente se le asigna un valor de 1 y al fondo se le asigna un valor de 0, aunque puede ser viceversa, el chiste es diferenciar uno del otro a partir de cierto umbral reconocido en el histograma de la imagen. El umbral es un número entero que puede valer de 0 a 255.

Original



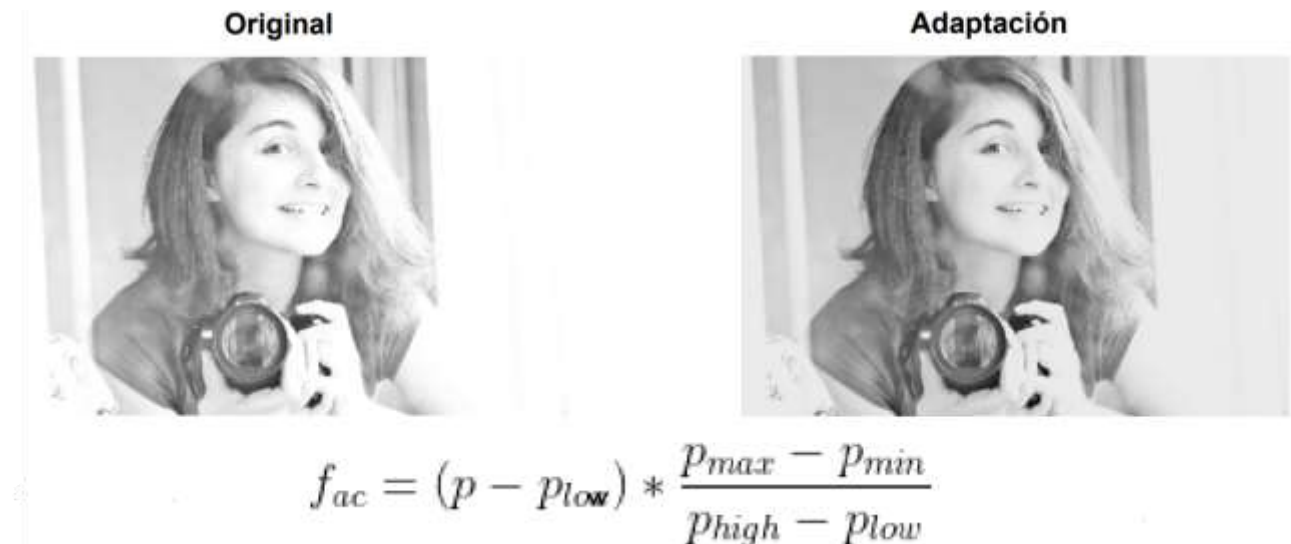
Binaria



4.- Adaptación Automática al Contraste:

Para que se mejore de forma automática un poco el contraste y la definición de la imagen, se debe aplicar la siguiente ecuación considerando un rango de tono de grises en el contraste que solamente se quiere tomar en cuenta llamados f_H y f_L :

$$f_{AC}(x,y) = (f(x,y) - f_{min}) * \frac{f_{max} - f_{min}}{f_H - f_L}$$

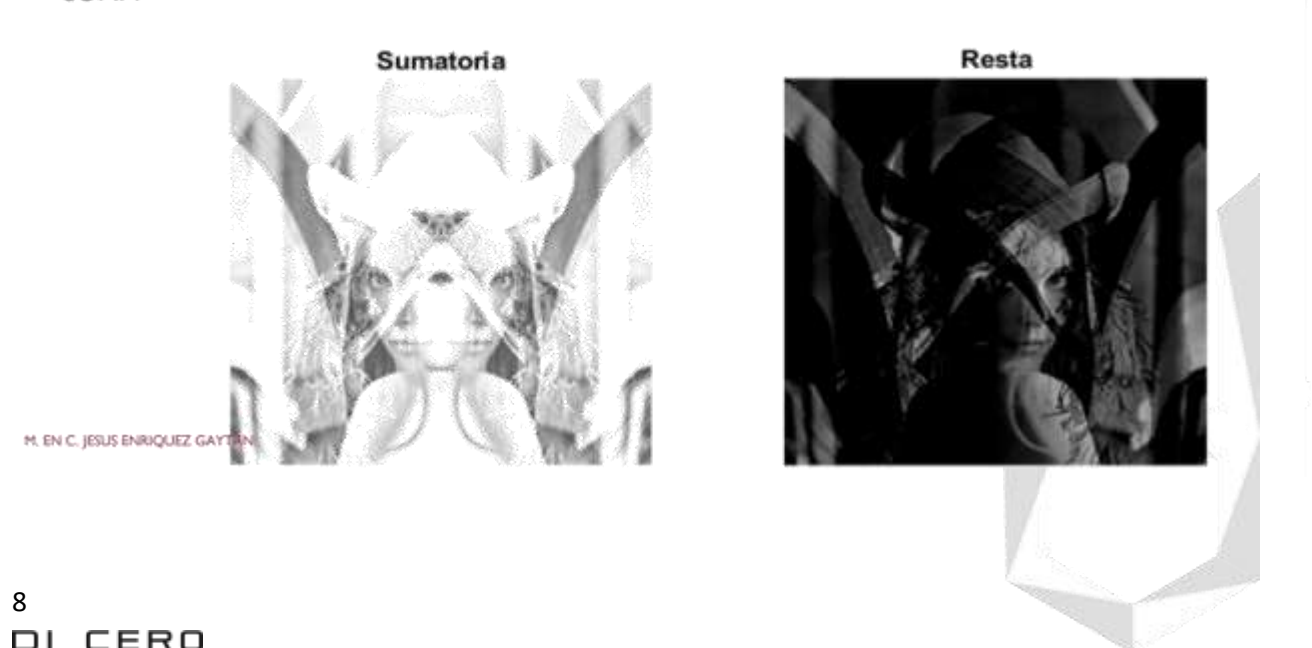


4.- Operaciones Lógicas Aritméticas:

La suma o resta de dos imágenes sirve para segmentación de ciertos objetos en la imagen, esto es para reconocimiento de patrones, con el fin de reconocer objetos específicos en una imagen, detección de movimiento o posiciones y también para meter o quitar ruido

■ SUMA $I' = (x,y) = I_A(x,y) + I_B(x,y)$

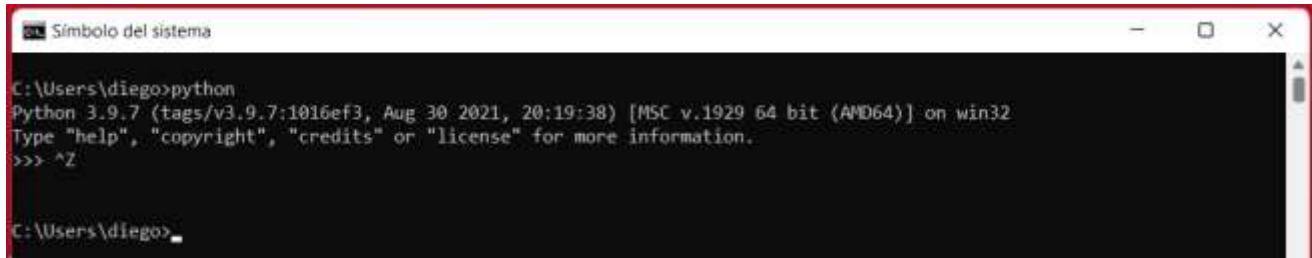
■ RESTA $I' = (x,y) = I_A(x,y) - I_B(x,y)$



Visión Artificial con Python

Para realizar visión artificial con Python se debe hacer uso de la librería OpenCV, para ello se debe contar con lo siguiente, que normalmente se instala desde la consola del sistema (CMD) y utilizar los editores de Código Visual Studio o Visual Studio Code:

- **Python:** La versión que se utiliza en estos ejercicios es la 3.9.7
 - <https://www.python.org/downloads/>



```
Símbolo del sistema
C:\Users\diego>python
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ^Z


C:\Users\diego>
```

- **OpenCV:** Es la librería que se utiliza para la visión artificial, en su página hay varios cursos gratuitos para aprender a hacer detección de distintas cosas.
 - <https://opencv.org/releases/>



INSTALAR OPEN CV

- Visual Studio 2022
- CMD --
- C:\Users\ingmk>pip install virtualenv
- C:\Users\ingmk>pip install virtualenvwrapper-win
- Acceso al entorno
- mkvirtualenv opencv
- pip install opencv-contrib-python
- python
- import cv2
- print(cv2.__version__)
- <https://github.com/Asadullah-Dal17/Basic-Augmented-reality-course-opencv/blob/master/README.md>
- pip install opencv-contrib-python



Pasos para la instalación y configuración de OpenCV:

- Instalar Python
- pip install
- pip install virtualenv
- pip install virtualenvwrapper-win
- mkvirtualenv opencv
- pip install opencv-contrib-python



Filtros

Los filtros lo que hacen son lo siguiente:

- **Suavizar la imagen:** Esto se realiza para quitar el ruido que puede tener una imagen, haciendo que se vea mejor, aunque a veces esto puede significar perder definición en la imagen resultante.
- **Identificación de bordes:** Se basa en la derivación de la imagen y sirve para encontrar los bordes en las imágenes, ayudando así a la identificación de objetos.

Filtros Espaciales:

Los filtros espaciales son aquellos donde el píxel nuevo obtenido después de haber aplicado el filtro no es nada más resultado de la operación matemática sobre el mismo píxel de análisis sino de la aportación de los píxeles de su vecindario, en filtros lineales es importante considerar el tamaño del filtro y vecindario que puede tener el siguiente número de filas y columnas en la vecindad de análisis:

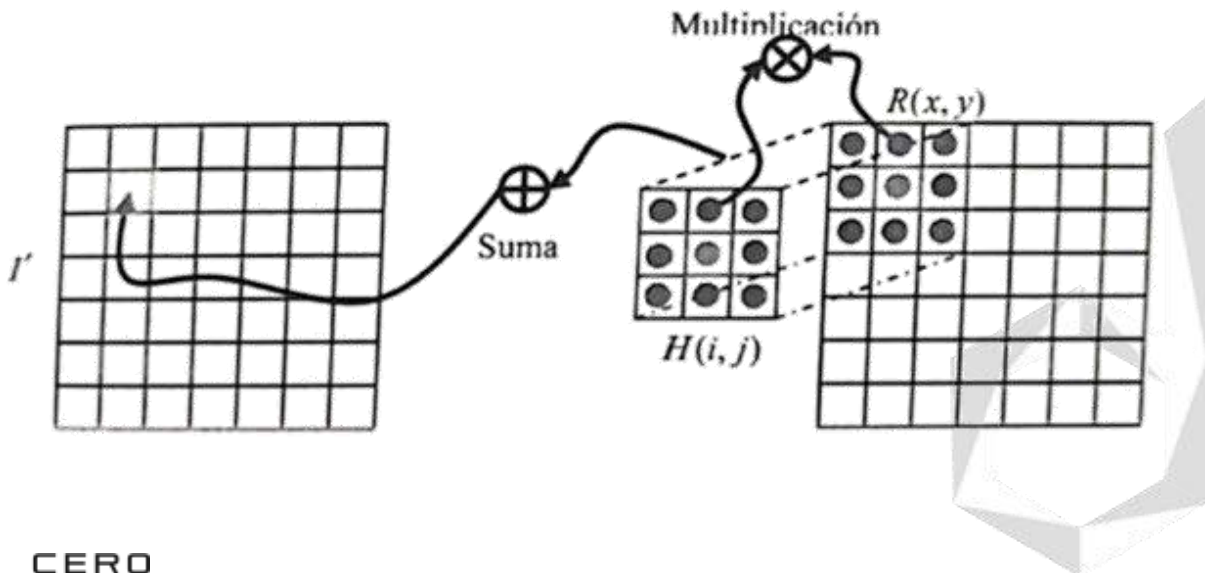
- 3X3
- 5X5
- 7X7
- 31X31

Esto se utiliza para dar un suavizado a la imagen y quitar así el ruido.

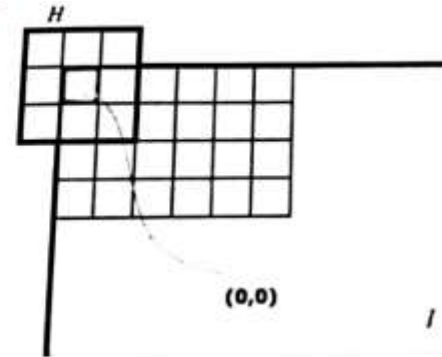
Los filtros espaciales se llevan a cabo por la aplicación de una convolución entre el píxel de interés, sus vecindarios y los coeficientes de la matriz de coeficientes del filtro.

$$I_{(x,y)} = \frac{1}{9} * \sum_{j=-1}^1 \sum_{i=-1}^1 I(x+i, y+j)$$
$$I'_{(x,y)} = \sum_{(i,j) \in R(x,y)} I(x+i, y+j) * H(i,j)$$

La Matriz H es llamada máscara, esta está predefinida con ciertos valores y con esta máscara y la imagen original $I_{(x,y)}$ se aplica la convolución.



- Los filtros espaciales a las imágenes se llevan a cabo por la aplicación de una convolución entre el pixel interés, sus vecindarios y los coeficientes de la matriz de coeficientes del filtro
- Se quita el borde de la imagen siendo N-1 y M-1



$$I'(x, y) = I(x-1, y-1) \cdot H(-1, -1) + I(x-1, y) \cdot H(-1, 0) + I(x-1, y+1) \cdot H(-1, 1) \\ + I(x, y-1) \cdot H(0, -1) + I(x, y) \cdot H(0, 0) + I(x, y+1) \cdot H(0, 1) + \\ I(x+1, y-1) \cdot H(1, -1) + I(x+1, y) \cdot H(1, 0) + I(x+1, y+1) \cdot H(1, 1)$$

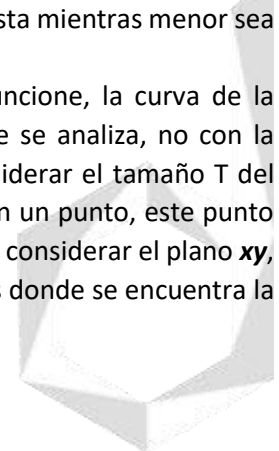
Convolución:

Una convolución es un operador matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g . Poniéndolo como un ejemplo mundano, es como calcar una imagen nueva sobre una hoja vacía contra luz y al final ver que tanto se parece la nueva a la original, esto es más o menos la convolución.

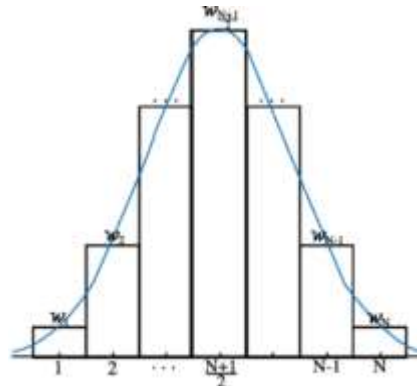
Filtros Lineales:

Se utiliza para realizar el suavizado de imágenes, esto lo que hace es difuminar la imagen quitando así el ruido, si se aplica a una imagen que tiene aplicado por ejemplo un reconocimiento de bordes, puede elegirse un tamaño de filtro en específico para que solamente se reconozca los bordes que sirvan, en vez de reconocerlos todos.

- Filtro Box:** Hace que todo sea cero y lo que queremos que salga del filtro que sea 1, creando así imágenes binarias.
- Filtro Gaussiano:** El filtro gaussiano lo que hace es aplicar una función gaussiana discreta, donde se busca que la curva se encuentre justo en donde está el píxel central de la imagen que se quiere analizar, recordemos que el tamaño T del filtro debe ser impar y es la parte discreta, mientras que las variables xy son las coordenadas de los píxeles pertenecientes a toda la imagen. La variable sigma afecta la falda de la función gaussiana, haciendo su curva más angosta mientras menor sea el valor de σ y más ancha mientras sea mayor el valor de sigma.
 - Es importante mencionar que para que el filtro gaussiano funcione, la curva de la función gaussiana debe intersectarse con el píxel principal que se analiza, no con la vecindad sino con el central, para que esto pase se debe considerar el tamaño T del filtro, con eso se tendrá la función gaussiana en 3D centrada en un punto, este punto será el píxel principal, de esta manera se puede ver que se debe considerar el plano xy , como la matriz donde se encuentra la imagen y la dirección z es donde se encuentra la curva de la gaussiana 3D.



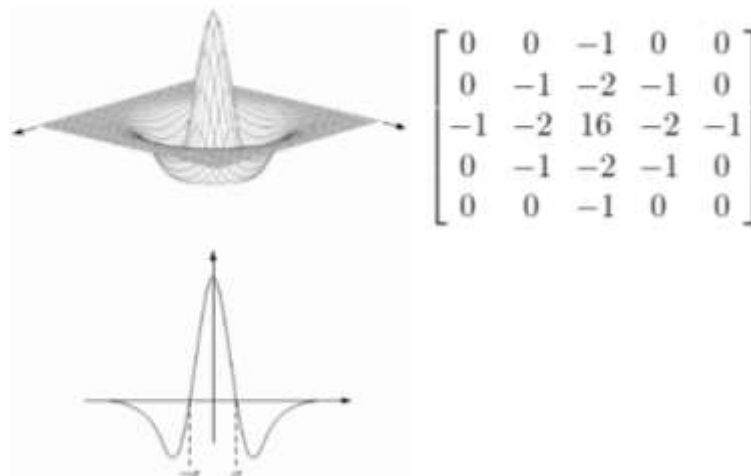
$$G\sigma_{(x,y)} = e^{-\frac{\left(x-\frac{T-1}{2}\right)^2 + \left(y-\frac{T-1}{2}\right)^2}{2\sigma^2}}$$



$$Hg = \frac{1}{57} \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

- **Filtro de Laplace:** Realiza también un suavizado de la imagen, pero en donde se obtienen no solo los máximos después de aplicar los filtros, sino los mínimos, esto se podrá observar cuando se realice una derivación discreta de la imagen, esto para la detección de bordes.

$$M\sigma_{(x,y)} = \frac{1}{\sqrt{2\pi\sigma^3}} \left(1 - \frac{x^2 + y^2}{\sigma^2}\right) e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



La derivación de imágenes se lleva a cabo respecto a los ejes **xy** de la imagen, los valores positivos del eje x si van hacia la derecha como normalmente se acostumbra, pero los valores positivos del eje y van hacia abajo, teniendo su punto inicial en la esquina superior izquierda. Es posible realizar la derivación porque la imagen se considera como una función de dos variables $f(x, y)$, para ello se realiza el gradiente de la función hacia las direcciones horizontal y vertical, las imágenes mostradas con G_x y G_y son totalmente distintas ya que G_x enfatiza en los bordes horizontales de la imagen y G_y enfatiza en los bordes verticales, para poder ver ambos bordes unidos se unifican G_x y G_y de una forma vectorial por medio de los dos

gradientes, esta parte no es tan importante, sino a la matriz que se llega después de haber realizado los cálculos matemáticos que son las siguientes para cada máscara.

- **Laplace:** La matriz a la que se llega es la siguiente, después de haber realizado los cálculos matemáticos y considerando una matriz con vecindad de 3X3 en una escala de grises ya que es de una sola capa (dimensión).

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f(x)}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Dos variables

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

La derivada de segundo orden con respecto a al variable x:

$$\frac{\partial^2 f(x, y)}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

De forma similar, con respecto a y:

$$\frac{\partial^2 f(x, y)}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

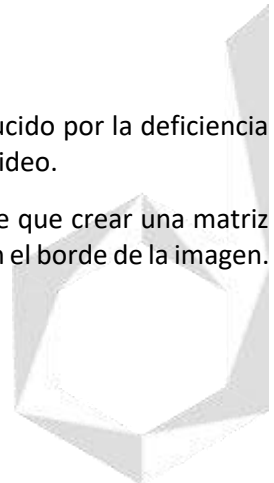
$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

$$W = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Ruido:

El ruido digital es la variación aleatoria del brillo o el color en las imágenes, producido por la deficiencia del dispositivo de entrada, osea la cámara digital con la que se tomó la imagen o video.

Este ruido se puede introducir en una imagen de forma manual, para ello se tiene que crear una matriz de 2X3, más pequeña o grande que la matriz del vecindario porque no debe estar en el borde de la imagen.





Uso de estructura para agregar ruido

| | |
|------------|--------------|
| (x, y) | $(x+1, y+1)$ |
| $(x+1, y)$ | $(x+1, y+1)$ |
| $(x+2, y)$ | $(x+2, y+1)$ |

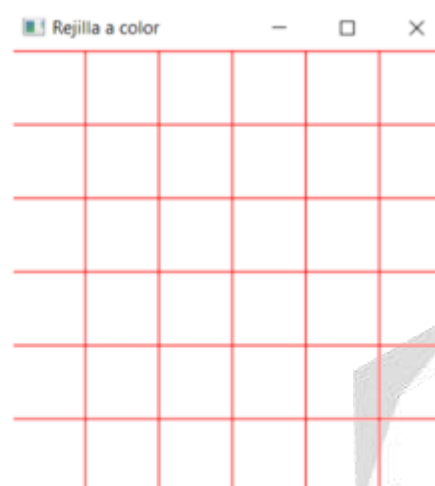
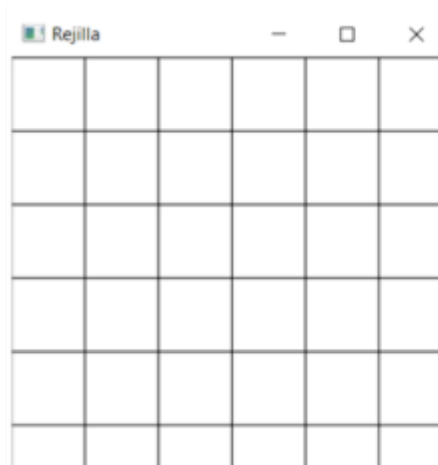
Características de una Imagen (Shape):

Las características de una imagen con python se pueden obtener usando el método **shape()**, esto para poder escalarlas y de esta manera tratar dos imágenes en conjunto para por ejemplo sumarlas o restarlas entre sí:

- **Tamaño:** Es el número de pixeles que pesa la imagen, sirve para saber el espacio de memoria que ocupa y su unidad de medida es el byte, que representa 8 bits.
- **Alto:** Altura de la imagen dada en pixeles.
- **Ancho:** Amplitud de la imagen dada en pixeles.
- **Canales:** Se refiere al número de capas de la matriz que conforma la imagen, si es una imagen RGB se tendrán tres canales y si es una imagen en escala de grises se tendrá uno solo.
- **Tipo:** El tipo de dato primitivo que representa la imagen, esto puede ser de los siguientes tipos:
 - **uint8:** Número entero de 8 bits, osea que va de 0 a $2^8 = 256$

Rejillas (Numpy):

Las rejillas en Python se pueden crear a manita a través de **bucles for** o usando una función predefinida de la librería **OpenCV**. Para dar color a la rejilla se utilizan los métodos **zeros()** o **ones()**, de la misma forma en la que se realiza para dar color a las imágenes sintéticas, osea las imágenes creadas con matrices a través del código Python, utilizando la librería **numpy**.



Filtro de Máximos, Mínimos y Media (Scipy):

El filtro de máximos y mínimos se utiliza para el manejo de ruido en una imagen, para ello se instala la librería **Scipy**, los filtros de máximos y mínimos son más utilizados en imágenes de escala de grises con tonos muy oscuros o binarias ya que se utilizan para obtener mayor definición en los tonos negros o blancos de una imagen, por ejemplo, es utilizado en radiografías para determinar de mejor manera las formas de los huesos en la imagen de blanco y negro, mientras que los filtros de media son utilizados en imágenes con escala de grises normales ya que estos ayudan a recuperar una imagen con escala de grises, quitando su ruido y manteniendo en mayor medida su color y forma original.



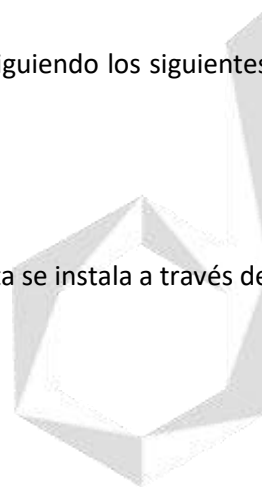
Para aplicar filtros de máximos y mínimos se instala la librería Scipy de Python, siguiendo los siguientes principios matemáticos:

$$I'_{(x,y)} = \min\{I(x+i)|(i,j) \in R\} = \min(R_{(x,y)})$$

$$I'_{(x,y)} = \max\{I(x+i)|(i,j) \in R\} = \max(R_{(x,y)})$$

Scipy es una librería de funciones matemáticas para la programación científica, esta se instala a través de consola con el siguiente comando:

- `pip install scipy`

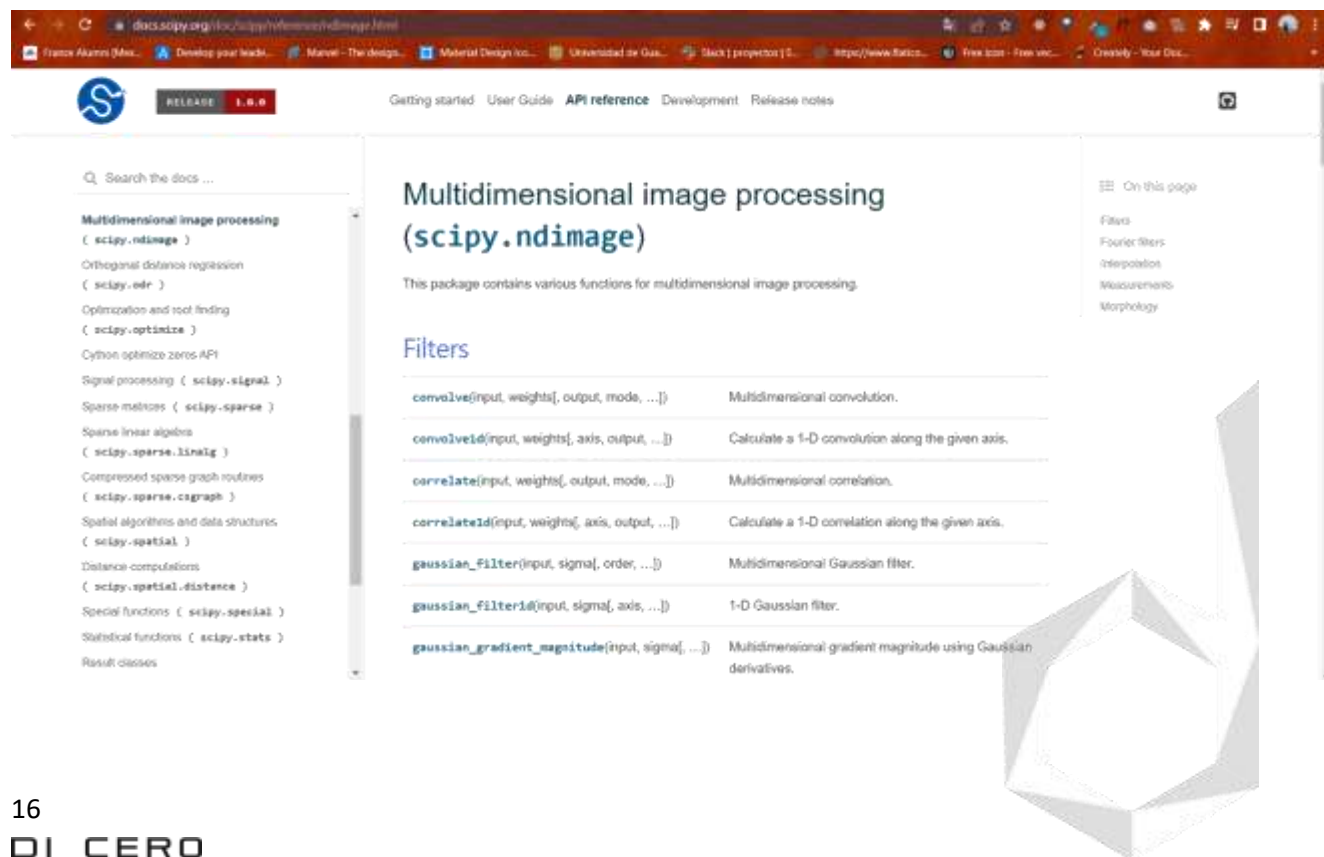


Para observar todos los filtros de imagen que se pueden utilizar con la API de la librería, se puede consultar la documentación del siguiente link:

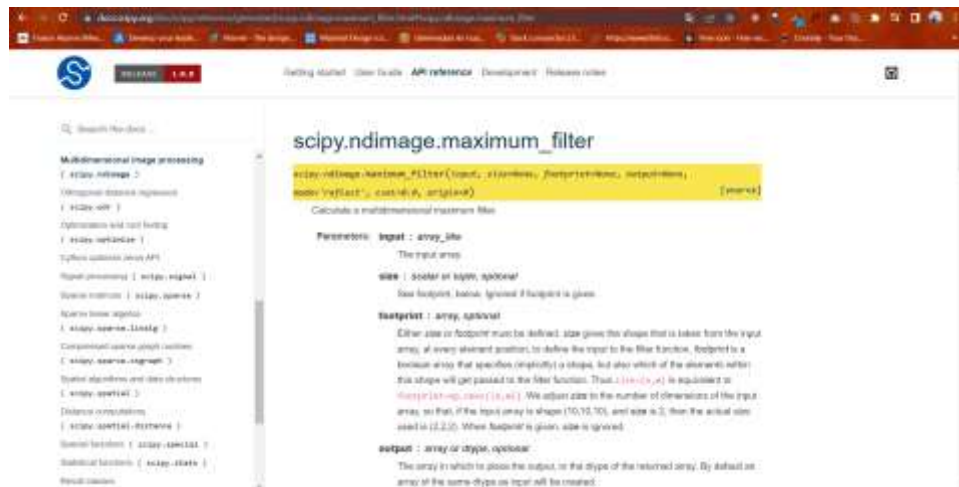
<https://docs.scipy.org/doc/scipy/reference/index.html>



En la API se describen todas las cosas que se pueden hacer con la librería, fuera de la aplicación de filtros, pero en específico se debe saber que para utilizar filtros se utiliza el paquete `ndimage`, en esta parte de la documentación es donde se pueden encontrar los distintos métodos de filtros que se pueden aplicar.



En específico la documentación para aplicar el filtro de máximos es el siguiente:

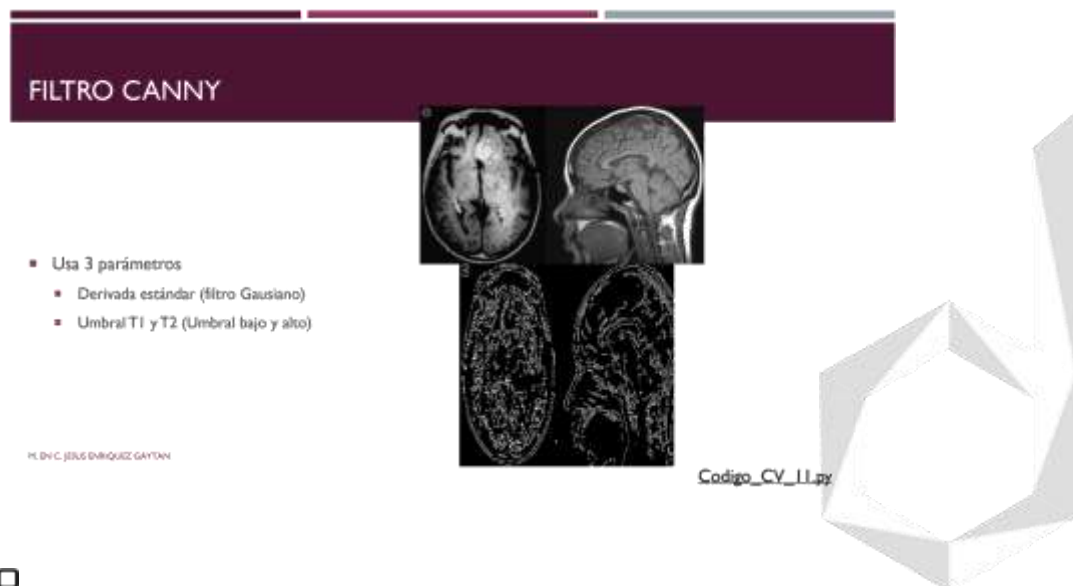


Filtro de Canny:

El filtro de Canny sirve para la identificación de bordes, esto para identificar y separar cada parte de una imagen, por lo general esto se aplica a una imagen en escala de grises para que en vez de que se tenga que analizar 3 capas de matrices, solo se analice una y de esta manera pueda realizar en análisis de bordes de una mejor y más veloz manera, los pasos para aplicar esto son los siguientes:

1. Identificación de la figura en la imagen que se quiere analizar.
2. Conversión a escala de grises de la imagen.
3. Detectar esquinas.
4. Quitar ruido.
5. Repetir estos procesos con la mayor rapidez posible ya que normalmente el análisis de imagen se realiza en tiempo real.

Normalmente estos filtros se basan en la aplicación de primeras y segundas derivadas de la imagen, ya que por ejemplo cuando se tiene el valor de un píxel negro y alado un píxel blanco en una imagen con escala de grises, se realiza un cambio abrupto en el color y esto se puede identificar como un borde, principalmente esto se aplica en los filtros de Gauss y Laplace. Para ello se parte del algoritmo de Harris.



Algoritmo de Harris:

El algoritmo de Harris explica la manera en la que se puede aplicar la primera derivada en imágenes, para ello se considera que estas derivadas se realizan en función de la dirección de la imagen, realizando una derivada parcial respecto a la dirección horizontal y vertical, contando con las siguientes ecuaciones, sabiendo que la función $I(x, y)$ representa la matriz de la imagen en escala de grises:

$$I_x(x, y) = \frac{d(I(x, y))}{dx}$$

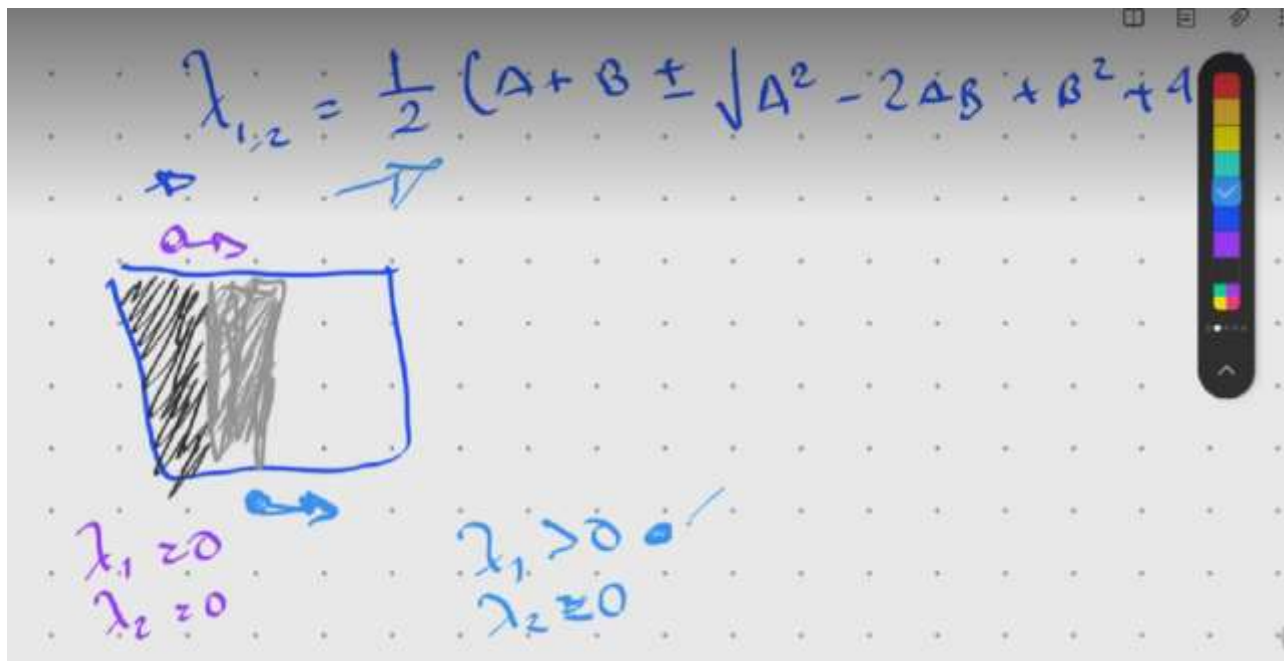
$$I_y(x, y) = \frac{d(I(x, y))}{dy}$$

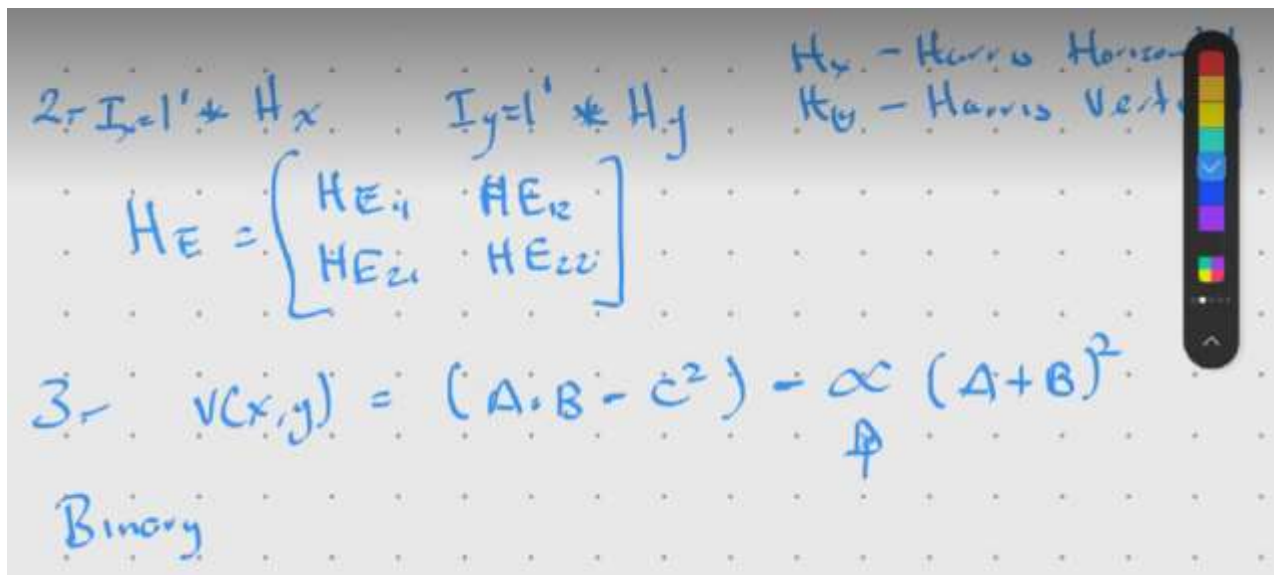
La derivación de imágenes se lleva a cabo respecto a los ejes **xy** de la imagen, los valores positivos del eje x si van hacia la derecha como normalmente se acostumbra, pero los valores positivos del eje y van hacia abajo, teniendo su punto inicial en la esquina superior izquierda. Es posible realizar la derivación porque la imagen se considera como una función de dos variables $I(x, y)$.

Después de hacer el cálculo de las derivadas, se llega a una matriz cuadrada que representa la derivada de la dirección horizontal (x) y vertical (y).

Detector de Harris:

Para poder identificar los elementos de la imagen y aplicar la derivada correctamente a la imagen, primero se debe aplicar un suavizado a la imagen por medio del filtro de media o promedio, luego se debe multiplicar el resultado por las matrices de Harris que representan la derivada horizontal y vertical para de esta manera obtener la matriz H que represente las dos, en este punto ya se podrá identificar bordes, para finalmente binarizar la imagen y así identificar un elemento específico de la imagen.





La matriz de Harris horizontal y vertical están definidas por ciertos valores específicos, que son los siguientes, dependiendo del tamaño de la matriz de la imagen o de la vecindad:

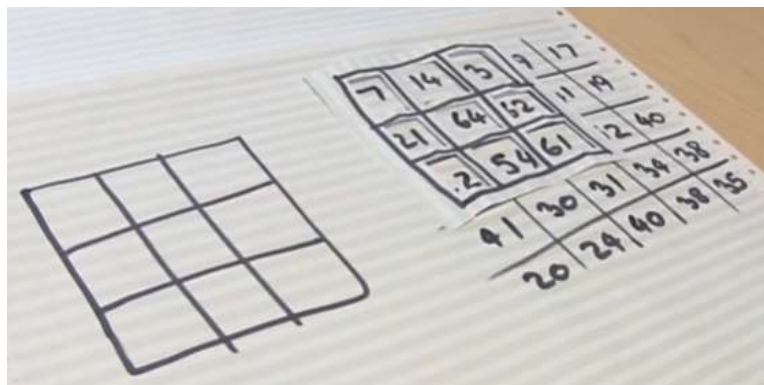
$$H_x = [-0.5 \quad 0 \quad 0.5], \quad H_y = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix}$$

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

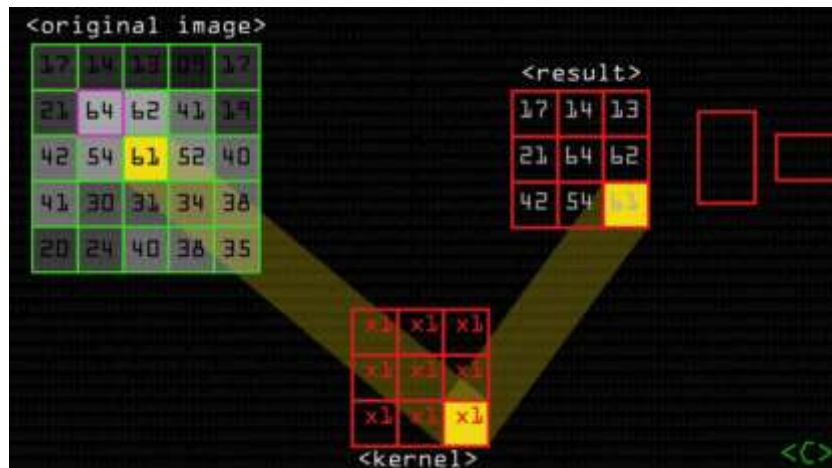
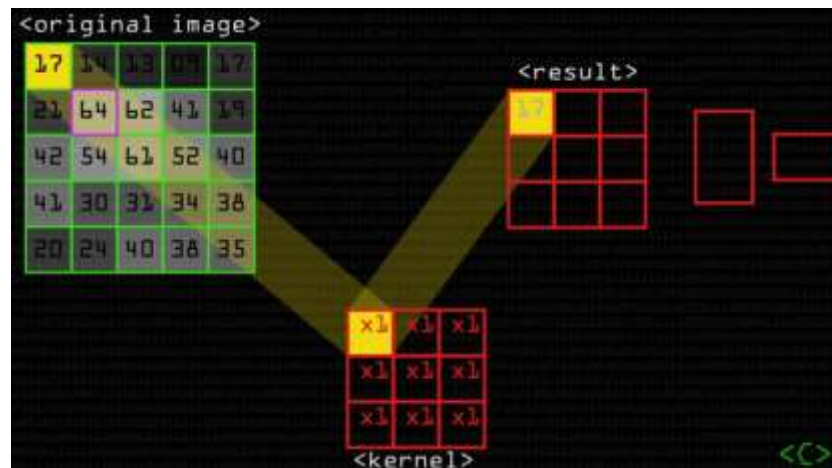
Convolución Kernel:

La convolución Kernel es un proceso en donde se toma una matriz pequeña de números y se aplica sobre una imagen completa por partes, recordemos que una convolución es un operador matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g .

Al usar distintos números en el Kernel se puede suavizar, detectar bordes, esquinas, etc. En la imagen, recordemos igual que el Kernel se va a aplicar sobre el vecindario de pixeles de la imagen, teniendo un solo pixel central de análisis en la imagen.



Lo que se hace con el Kernel y los valores de color del vecindario de la imagen donde se está aplicando el Kernel es que se multiplica cada valor correspondiente del Kernel con el valor de color de la imagen sobre la coordenada en específico donde se encuentra en el momento, para posteriormente sumarlo todo y normalizarlo, dividiendo entre el número de valores del Kernel. Básicamente sacando un promedio y lo que se hace con ese valor es que se asigna al pixel central de análisis actual, para después moverse un lugar hacia la derecha y hacer lo mismo, recorriendo así la imagen de izquierda a derecha y de arriba hacia abajo cuando haya recorrido una fila completa, aplicando de esta forma el Kernel a toda la imagen, usando vecindarios del mismo tamaño que la matriz del Kernel.



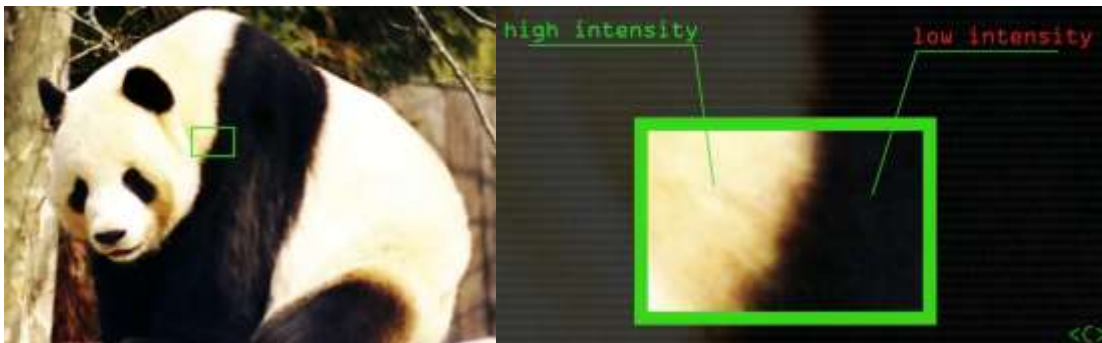
Este proceso crea una nueva imagen, no sobre escribe los valores de color de la imagen original. En los pixeles de las esquinas se hace lo mismo, analizando el pixel central de la esquina, pero no tomando en

cuenta los pixeles que no existen, haciendo un promedio más pequeño, con menos valores del Kernel. Para obtener el suavizado de una imagen, se tienen dos matrices principales 3X3, la de promedio y la de Gauss, que trata de representar una campana de Gauss de la forma más simplificada posible alrededor del pixel de análisis.

$$Avg = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad Gauss = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

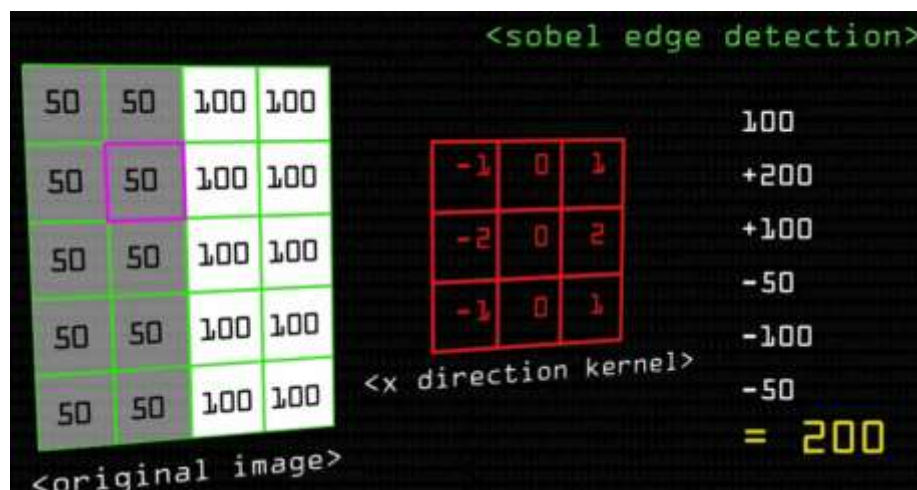
Operador Sobel:

El Kernel del operador Sobel sirve para la identificación de bordes, el concepto base para la identificación de bordes es hacer un análisis de la imagen donde se identifique un cambio brusco de color, un valor alto de la imagen resultante indica un borde empinado y un valor bajo indica un borde no tan profundo. Este se aplica en la dirección horizontal (x) y la dirección vertical (y).



$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad Sy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Esto se ve ilustrado en un ejemplo a continuación, donde el cambio brusco de valor de color en una imagen con escala de grises, claramente indica un borde:



Este análisis indica la orientación del borde, pero para juntar las dos direcciones para obtener un valor llamado G, con el fin de obtener este valor se aplica la ley de Pitágoras con las direcciones xy, el valor G

indica la magnitud de la profundidad del borde (a este valor se le llama gradiente G), pero con este valor se puede obtener también el ángulo de inclinación del borde aplicando el arco tangente o tangente inversa con las direcciones xy.

$$G = \sqrt{Sx^2 + Sy^2} \quad \theta = \tan^{-1} \left(\frac{Sy}{Sx} \right)$$



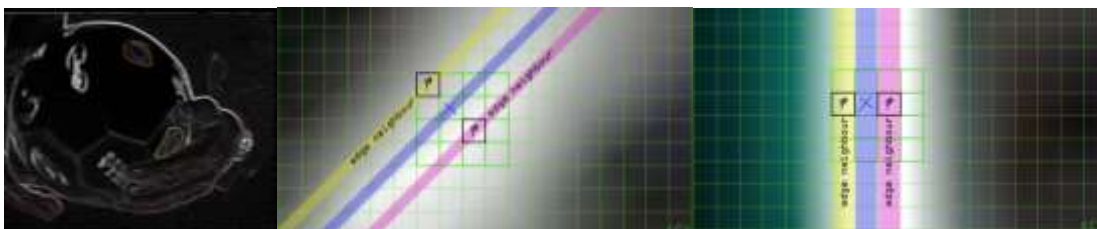
Detector Canny:

Lo que hace el detector Canny es mejorar la detección de bordes, ya que quita los bordes que no nos interesan y deja solamente los que, si nos interesan, haciendo que el borde sea una sola línea en vez de varias como lo muestra el operador Sobel, el operador Canny recibe como entrada la salida del operador Sobel. El proceso de procesamiento de una imagen es:

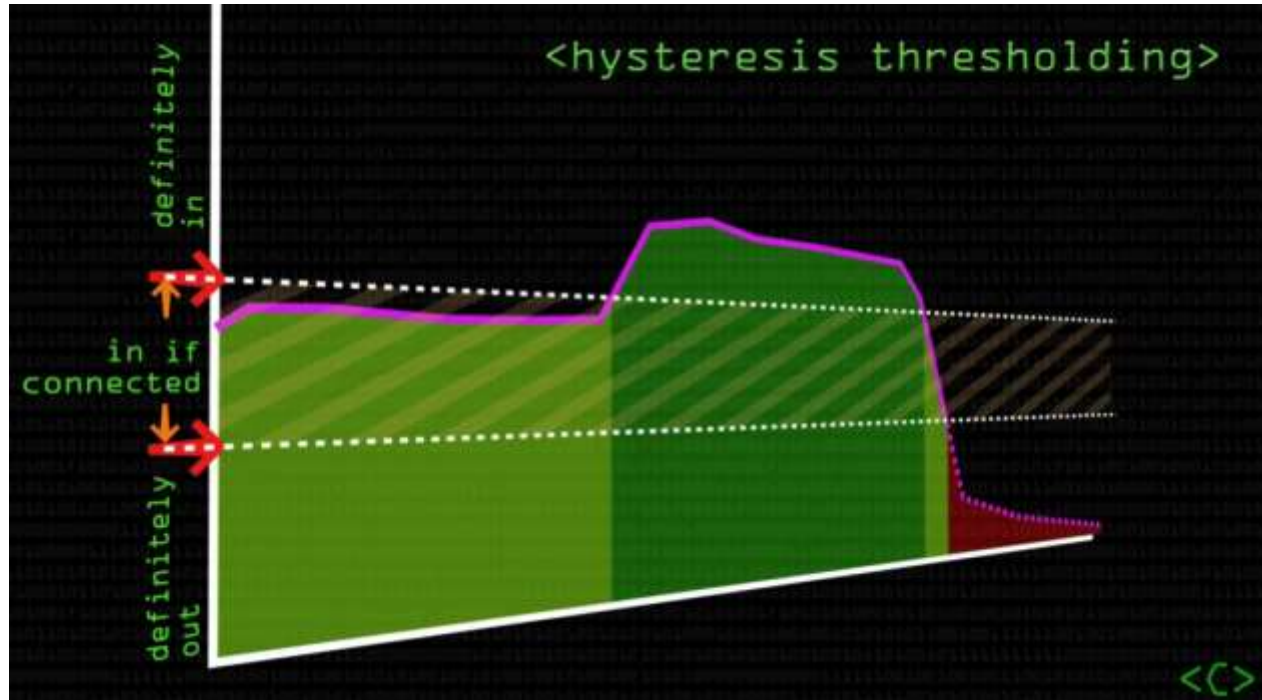
1. Capturar la imagen.
2. Convertirla a escala de grises.
3. Suavizarla (difuminarla).
4. Operador sobel (detección de bordes): Gradiente y orientación.
5. Detector Canny.



Para identificar mejor los bordes de una imagen, primero se analiza si el pixel central de análisis es un máximo local en el vecindario en la escala de grises que tiene valores de 0 a 255, esto lo hace para mejorar el análisis de bordes, para ello considera la inclinación del borde también.



El operador Canny ya que realizó la identificación de bordes, ejecutará un proceso llamado umbral de histéresis que hace uso de dos umbrales, para preservar solamente los bordes más pronunciados y quitar los bordes que no nos interesan. Para saber cuales son los valores que se quiere obtener de los bordes, se puede obtener el histograma de la imagen obtenida con los bordes y ver que tonos de gris son los que representan la mayoría de los bordes que queremos preservar.



Filtro Suavizado, Algoritmo de Harris, Filtro de Gauss, Detección de Esquinas:

Al suavizar la imagen se quita brillo y se hacen más uniformes algunas figuras de forma sutil, para ello se usa el filtrado promedio. El filtrado promedio es un método para suavizar las imágenes al reducir la cantidad de variación de intensidad entre los píxeles vecinos. El filtro promedio funciona moviéndose a través de la imagen píxel por píxel, reemplazando cada valor con el valor promedio de los píxeles vecinos, incluido él mismo. Los pasos con los que se analizan los bordes de la imagen entonces son los siguientes:

1. Cargar la imagen.
2. Se suaviza la imagen por medio del filtro promediador:
 - a. $H_p = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
 - b. $I' = I_{original} * H_p$
3. Se calcula la imagen con respecto a las coordenadas xy:
 - a. $H_x = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix}$
 - b. $I_x = I' * H_x$
 - c. $H_y = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix}$
 - d. $I_y = I' * H_y$
4. Ya teniendo las matrices de Harris xy se calcula la matriz de Harris H_E :

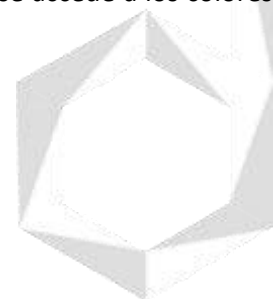


- a. $H_E = \begin{bmatrix} H_{E11} & H_{E12} \\ H_{E21} & H_{E22} \end{bmatrix}$
 - b. $H_{E11} = I_x^2$
 - c. $H_{E22} = I_y^2$
 - i. En estas dos ecuaciones se realiza la función cuadrada porque se tiene un comportamiento parabólico, que identifica cambios abruptos en la matriz, porque un pixel vale 255 y de repente 0 en la escala de grises, esto para aplicar una derivada en la imagen e identificar bordes.
 - d. $H_{E12} = H_{E21} = I_x * I_y$
5. Después de haber aplicado el filtro de Harris, se debe aplicar el filtro de Gauss, para mejorar así la identificación de bordes. La fórmula para aplicar esto en conjunto con la matriz de Harris es la siguiente:
- a. $E = \begin{bmatrix} H_{E11} * H\alpha & H_{E12} * H\alpha \\ H_{E21} * H\alpha & H_{E22} * H\alpha \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$
6. Después de haber aplicado el filtro de Gauss, se deben obtener los valores propios (eigenvalores), estos se refieren a los que representan las esquinas en la imagen. Los valores del umbral como son eigenvalores serán valores altos, alrededor de los 1000, 2000, etc. por lo que se da una condición de umbral, en donde se dice que el umbral está dado siempre que el eigenvalor sea mayor al umbral th propuesto.
- a. $Eig(x, y) = (A * B - C^2) - \alpha(A + B)^2$
 - b. $th = \text{propuesto, con valor de 1000 o mayor}$
 - c. $U(x, y) = Eig(x, y) > th$

Obtención de Color de una Imagen:

Para mostrar y almacenar el color de una imagen se debe crear una función a manita para que cada que se dé clic sobre la imagen, se obtenga el color de la parte en donde se dio clic, para ello se debe recibir como parámetros de la función:

- **El evento del mouse:** Este evento forma parte de los eventos descritos por la librería OpenCV.
- **Las coordenadas en x,y de la parte de la imagen donde se dio clic:** Obtenidas de la imagen ya que esta es una matriz.
 - Cuando se obtiene las coordenadas x,y de una imagen, estas se obtienen en forma de array, recordemos que la imagen RGB es tal cual una matriz de 3 dimensiones y esta se puede manejar como tal, de esta manera cuando se realice el evento de clic en la imagen, podemos obtener las coordenadas **xy** de la parte donde se dio clic de la imagen.
 - **toList():** Este método se aplica a una variable de tipo array para convertirla en una lista
 - Específicamente en visión artificial esto se usa para poder acceder a las coordenadas x,y de las 3 capas o dimensiones RGB, como se muestra en el siguiente ejemplo, recordando que en python se accede a los colores RGB en el orden BGR:
 - **color** = img[x, y].toList()
 - **azul** = img[x, y][0]
 - **rojo** = img[x, y][1]
 - **verde** = img[x, y][2]



| | 1 | 2 | 3 | 4 | 5 | X |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| Y | | | | | | |

- `Color= img[3,2]` el valor es guardado como array
- `color = img[3, 2].tolist()`
- `Azul = img[3, 2][0]`
- `verde = img[3, 2][1]`
- `Rojo = img[3, 2][2]`

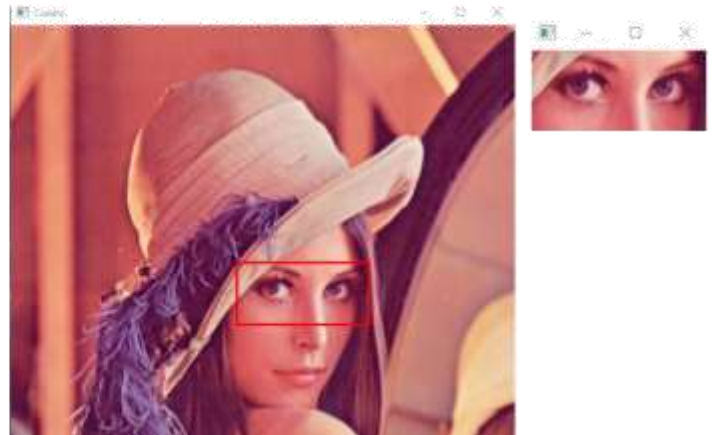
- **Bandera:** Indica cuando ya se realizó el evento del mouse
- **Parámetros:**

Recorte de una Imagen:

Se realiza una operación muy similar a la anterior para hacer un recorte de la imagen:

RECORTE DE REGIONES

| | 1 | 2 | 3 | 4 | 5 | X |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| Y | | | | | | |



Transformada de Hugh- Detección de Líneas:

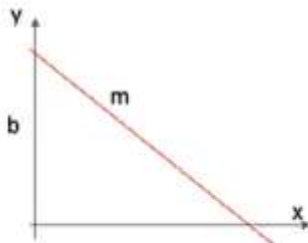
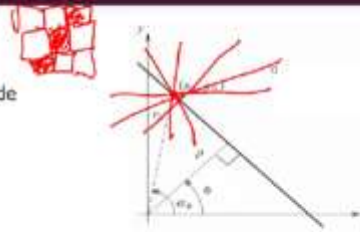
La operación matemática de la transformada de Hough lo que hace es mapear puntos de los bordes de la imagen, para luego medir si la pendiente es la misma, si es así, identifica que ha encontrado una línea:

LA TRANSFORMADA DE HOUGH. DETECCIÓN DE LÍNEAS

Usar un detector de bordes para obtener los puntos de la imagen que pertenecen a la frontera de la figura deseada.

$$y = mx + b$$

La ecuación de la recta en coordenadas polares
 $x \cos \theta + y \sin \theta = \rho$

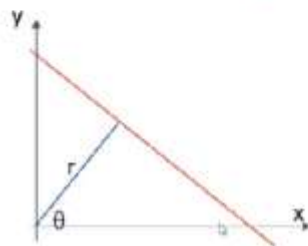


The straight line is normally parameterized as:

$$y = mx + b$$

Where m is the slope and b is the intercept.

NOTE: m goes to infinity for vertical lines.

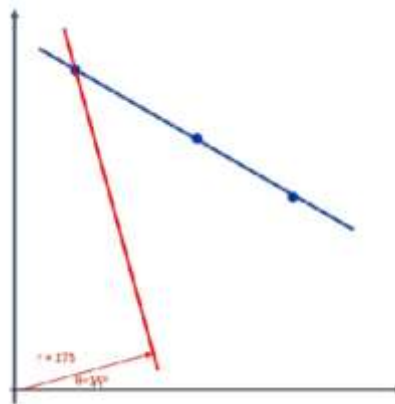


The line can also be represented as:

$$r = x \cos \theta + y \sin \theta$$

where r is the distance from the origin to the closest point on the straight line.

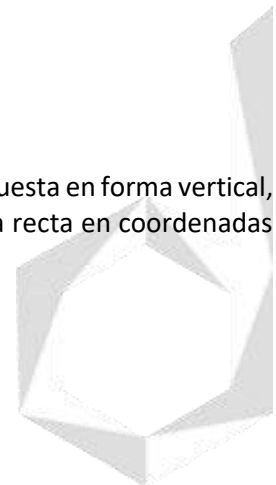
(r, θ) corresponds to the Hough space representation of a line.



Pero el problema con utilizar la ecuación de la recta normal es que, si la línea está puesta en forma vertical, la pendiente se va a indeterminar, por lo cual se prefiere utilizar la ecuación de la recta en coordenadas polares, que es exactamente igual pero descrita de una forma diferente.

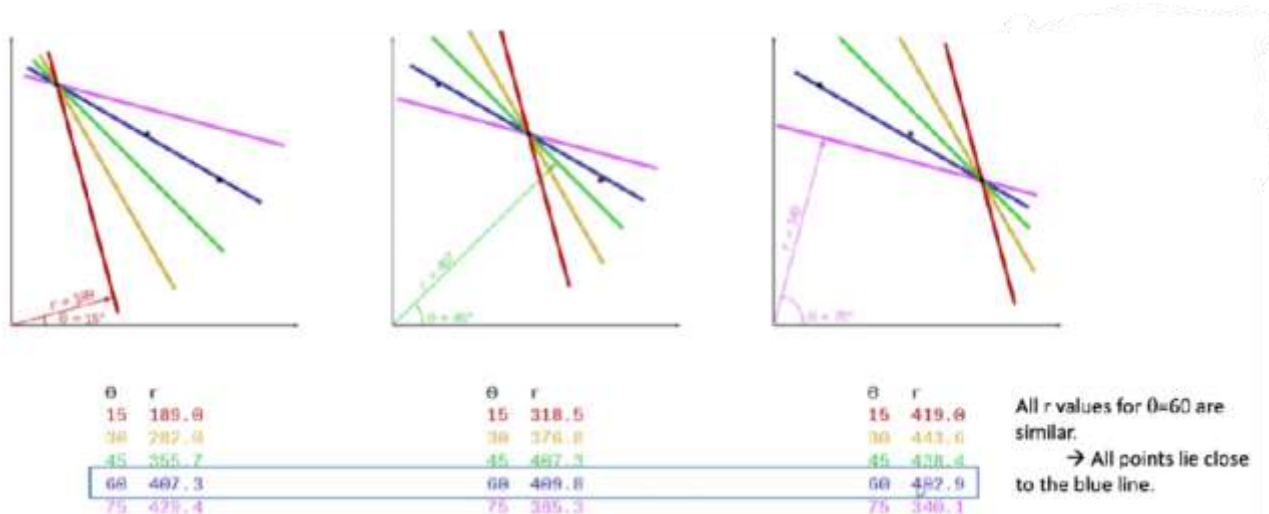
$$y = mx + b$$

$$x \cos \theta + y \sin \theta = \rho$$



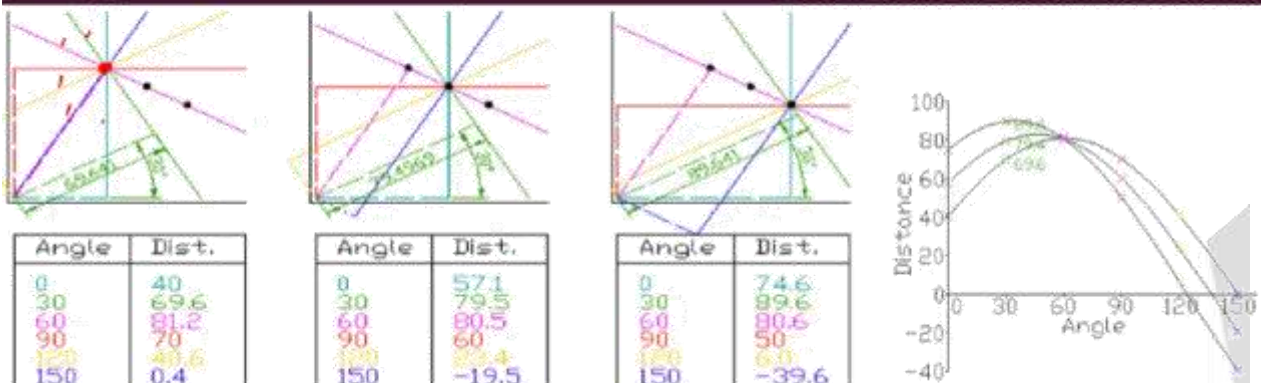
Los pasos con los que se analizan los bordes de la imagen entonces son los siguientes:

1. Cargar la imagen.
2. Detectar los bordes de la imagen.
3. Detectar las líneas en los bordes de la imagen:
 - a. Por cada punto en la imagen si es que las coordenadas (x, y) están en un borde:
 - i. Para todos los posibles ángulos Θ :
 1. Calcular p en los puntos (x, y) con un ángulo Θ .
 2. Incrementar la posición (p, Θ) en el acumulador.
 3. Buscar las posiciones con los mayores valores de p en el acumulador.
 4. Devolver las rectas cuyos valores son los mayores en el acumulador.



Recordemos que la línea p debe ser perpendicular a la recta que está midiendo, desde el origen del plano cartesiano (x, y) tomando en cuenta el ángulo.

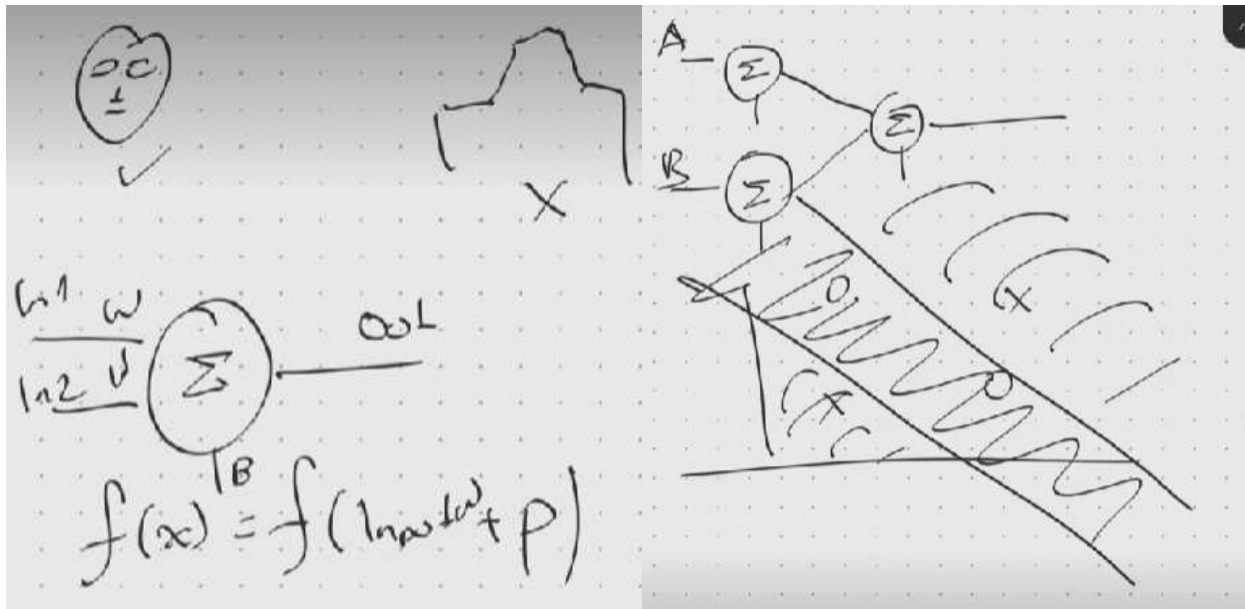
LA TRANSFORMADA DE HOUGH. DETECCIÓN DE LÍNEAS



Identificación de Rostros – OpenCV – Viola Jones y Patrones Binarios:

La red neuronal de OpenCV ya está entrenada, está la ventaja que tiene es que ya está entrenada, esta lo que hace es identificar a qué conjunto pertenece cada entrada de datos, esto lo que implica es que la red vaya aprendiendo, ya que una sola neurona no puede realizar un cálculo complejo, sino que se necesita una red de neuronas para que esto pueda ser posible.

Esto a grandes rasgos representa un análisis de patrones.



Las redes neuronales eran entrenadas para identificar rostros, pero por ejemplo en la cara de los hombres, usualmente la frente es más salida, lo que significa más sombra en la parte de los ojos, mientras que en las de las mujeres, esto no es así, por eso es que ciertos aspectos del análisis de rostros toman mucho tiempo en perfeccionarse.

