UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INGENIERÍA EN COMPUTACIÓN

CÁLCULO VECTORIAL

APLICACIONES DE LAS DERIVADAS PARCIALES EN VISIÓN ARTIFICIAL:

DETECCIÓN AUTOMÁTICA DE BORDES.

(Ya le mostramos el proyecto)

Chávez Navarro Salma
García González Karina
García Muratalla Luis Ángel
Mercado Romero Yahir Ricardo
Valentin Baena Karla

OBJETIVO

Diseñar un detector automático de bordes de imágenes aplicando el concepto de derivadas parciales.

INTRODUCCIÓN

La visión artificial es una disciplina científica, es decir, trata de conseguir que las máquinas puedan percibir y comprender una o varias imágenes y actuar de una manera determinada. La visión artificial engloba todas las aplicaciones industriales y no industriales en las que una combinación de hardware y software proporciona orientación operativa a los dispositivos en la ejecución de sus funciones basándose en la captura y el procesamiento de imágenes.

Este tipo de visión no necesita analizar todos los detalles de una imagen, sino solo aquellos que impulsen a realizar una acción determinada. Un ejemplo claro de esto es la visión artificial de un coche autónomo sólo se preocupa de detectar obstáculos para evitar colisiones.

La principal importancia por la que un robot pueda ver y entender perfectamente el entorno en el que se desarrollas es para llevar a cabo una tarea de la mejor manera, esto debido a que a través de la visión artificial cualquier máquina puede realizar diversas tareas, un ejemplo claro de esto son los carros de la empresa Tesla, los cuales son capaces de manejar por ti y para poder llegar a realizar eso necesita una visión amplia del entorno en el que están para que de esta manera la máquina sea capaz de analizar dicho entorno y a través de ese análisis ejecute sus tareas correspondiente de la mejor manera posible, además que este nos es el único ejemplo que puede haber del porqué es importante que cualquier tipo de máquina pueda ver y entender su entorno, es por eso que la principal finalidad de la visión artificial es hacer que el robot o máquina en cuestión logre analizar el mundo real tal como lo ven los humanos y de esta manera poder tomar la acciones necesarias para lograr un resultado determinado y esto se lograr hoy en día gracias a los sensores, cámaras, sistemas de visión que se implementan para que una máquina sea capaz “ver”. Ahora si nos vamos a temas de opiniones más generales, si en algún punto del futuro los humanos tengan la opción de comprar robots para realizar una tarea del hogar, obviamente los robots en cuestión deben ser capaces de entender el entorno en donde están para que dependiendo de esto realice la tarea que se le asigne.

La detección de bordes es una herramienta fundamental en el procesamiento de imágenes y en visión por computadora, tiene como objetivo la identificación de puntos en una imagen digital en la que el brillo de la imagen cambia drásticamente o, más formalmente, esta detección de bordes se preocupa de la localización e identificación de significativas variaciones de niveles grises en una imagen digital. Esta información es muy útil en aplicaciones para imágenes en reconstrucción, movimiento, reconocimiento, mejoramiento, restauración, compresión, etc.

Por otro lado el l modelo de color RGB es un modelo de color aditivo en el que los colores rojo, verde y azul se mezclan en varias proporciones para formar una matriz diferente de colores. Este nombre se le dio con las primeras letras de tres colores primarios rojo, verde y azul. El objetivo principal del modelo de color RGB es la detección y representación de imágenes en sistemas electrónicos, como televisores y computadoras, aunque también se ha utilizado en la fotografía.

Ahora en cálculo, una derivada parcial de una función de diversas variables, es la derivada respecto a cada una de esas variables manteniendo las otras como constantes. Las derivadas parciales son útiles en cálculo vectorial, geometría diferencial, funciones analíticas, física, matemática, etc. Las derivadas parciales de una función multivariable las definiremos también mediante un límite, si este límite existiera, haciendo extensiva la definición de una derivada ordinaria. Por consiguiente:

$$f_x(a,b) = g'(a) \ \ donde \ g(x) = f(x,b)$$

De acuerdo con la definición, tenemos lo siguiente:

$$g'(a) = \lim_{h \to 0} \frac{g(a+h)-g(a)}{h}$$

De tal forma que la ecuación queda de la siguiente manera:

$$f_x(a,b) = \lim_{h \to 0} \frac{f(a+h,b)-f(a,b)}{h}$$

De igual manera, la derivada parcial de f con respecto a y en (a,b), denota por $f_y(a,b)$, se obtiene al mantener fija la variable x(x=a) y determina la derivada ordinaria de b de la función $g(y) = f(a,y)$:

$$f_y(a,b) = \lim_{h \to 0} \frac{f(a+h,b)-f(a,b)}{h}$$

DEFINICIÓN DEL PROBLEMA

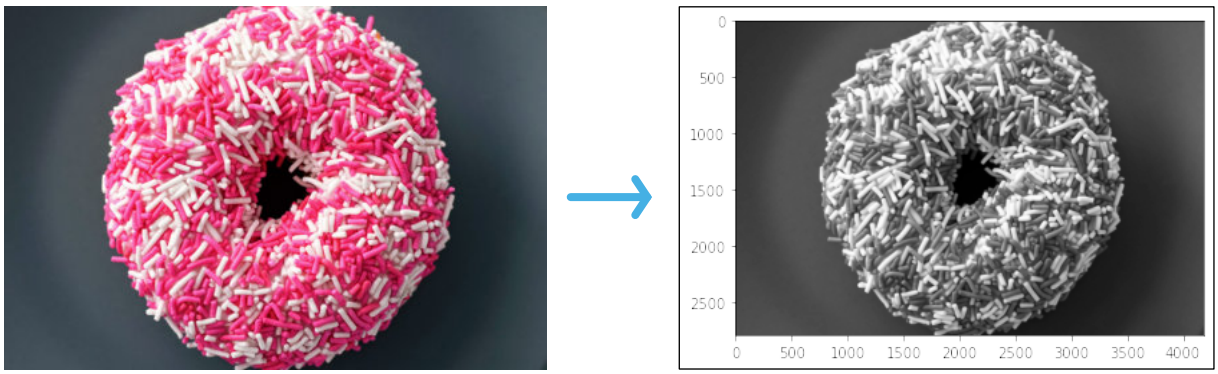
En los últimos años han existido grandes avances, y actualmente existen muchos algoritmos que permiten a las computadoras entender el contenido de imágenes y videos. Por ejemplo, existen algoritmos para identificar los objetos que aparecen en una imagen, o detectar la presencia de rostros en una fotografía, e incluso reconocer su identidad.

En este proyecto estudiaremos uno de los problemas más básicos de visión artificial, que es la detección de bordes de una imagen. Esto consiste en identificar la orilla o contorno de un objeto. Imagina que estás programando un robot para que agarre una manzana. Sería de gran utilidad para el robot conocer el contorno de dicha manzana. Para implementar este proyecto se usará lenguaje Python.

ACTIVIDADES

1.- A partir de una imagen *.jpg* muestra las imágenes resultantes de calcular la derivada parcial con respecto a x, la derivada parcial con respecto a y, el gradiente, y el gradiente con los colores invertidos (puntos 6, 7, 8 y 9 del procedimiento).

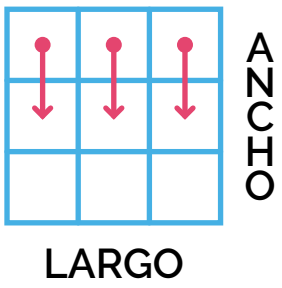
Para detectar y resaltar el borde de un objeto en una imagen es más fácil si la trabajamos en escala de grises.



Para poder mostrar los bordes de un objeto utilizaremos las derivadas parciales. Como ya sabemos una imagen es una matriz donde cada celda tiene un valor correspondiente al color. Lo primero que necesitamos hacer es crear la matriz donde vamos a almacenar los valores comparados de la imagen que ya está en escala de grises, para rellenarla necesitamos dos ciclos "for" que recorrerán las filas y las columnas:

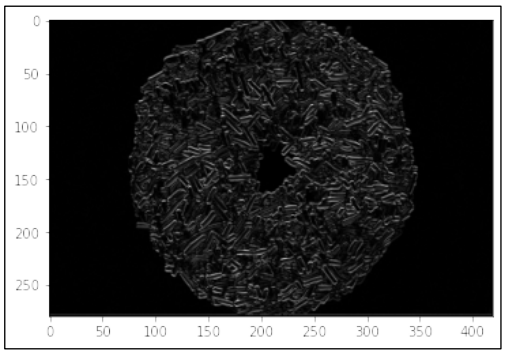
```
for i in range(0,ancho-1):
    for j in range(0,largo-1):
        dx[i][j]= abs(imagen[i+1][j] - imagen[i][j])
```

El código compara cada una de las celdas que hay a lo largo de la imagen con la celda que tiene debajo y repite el proceso con cada fila excepto la última, pero ¿cómo hace esta comparación?: simplemente obtiene la diferencia que hay entre los dos valores y guarda esos resultados en la nueva matriz que creamos. El color que tiene por valor "0" es el negro por lo cual entre menor sea la diferencia, más oscuro será el valor que guardamos y entre mayor sea, más claro será debido a que el blanco tiene por valor "1".

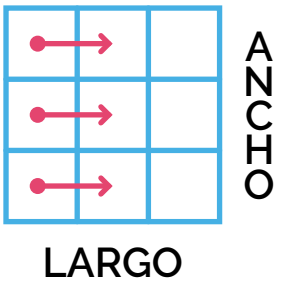



```
dx = np.zeros(shape=(ancho,largo))

for i in range(0,ancho-1):
    for j in range(0,largo-1):
        dx[i][j]= abs(imagen[i+1][j] - imagen[i][j])


plt.imshow(dx,cmap = 'gray')
ne = plt.show()
```

Derivada parcial respecto a x

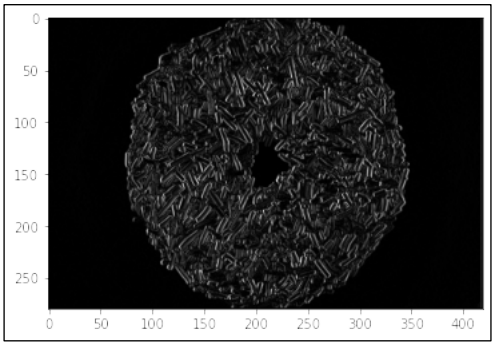


Posteriormente, el programa vuelve a realizar la comparación, sólo que en este caso cambia la dirección. Ahora hace el movimiento de contrastar cada celda con la que tiene a un lado, repitiendo este proceso con todas las columnas exceptuando la última.



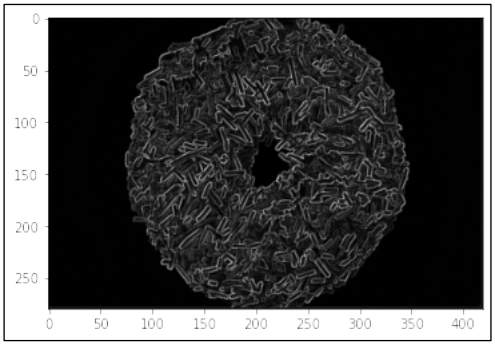

```
dy = np.zeros(shape=(ancho,largo))
for i in range(0,ancho-1):
    for j in range(0,largo-1):
        dy[i][j]= abs(imagen[i][j+1] - imagen[i][j])

plt.imshow(dy,cmap = 'gray')
ne = plt.show()
```

Derivada parcial respecto a y



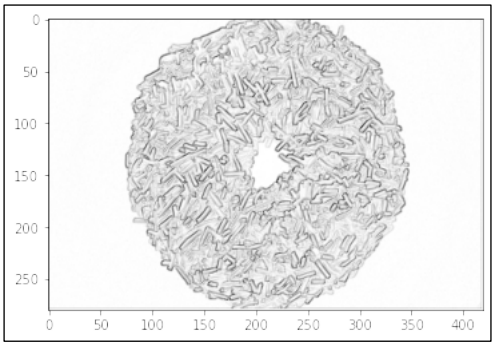
El gradiente suma la derivada parcial de x más la derivada parcial de y:

```
gradiente = np.zeros(shape=(ancho,largo))

for i in range(0,ancho):
    for j in range(0,largo):
        gradiente[i][j]= dx[i][j] + dy[i][j]

plt.imshow(gradiente,cmap = 'gray')
ne = plt.show()

```



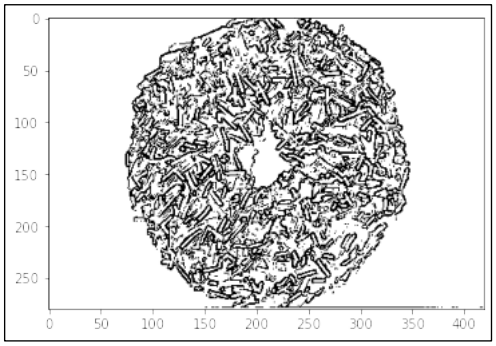
Aún podemos hacer que los bordes contrasten más si invertimos los colores de cada pixel, para ello simplemente utilizamos la diferencia que tienen con 1 y listo, los colores oscuros tendrán una diferencia mayor y por lo tanto un tono más claro caso contrario con los claros.

```
for i in range(0,ancho):
    for j in range(0,largo):
        gradiente[i][j]= 1 - gradiente[i][j]

plt.imshow(gradiente,cmap = 'gray')
ne = plt.show()
```



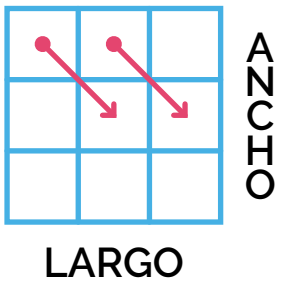
2.- Continua con el desarrollo del código para lograr que en la imagen final resalten más los bordes, haciendo que tu imagen resultante esté formada únicamente por pixeles blancos y negros.

```
umbral = 0.85

for i in range(0,ancho-1):
    for j in range(0,largo-1):
        if gradiente[i][j] <= umbral:
            gradiente[i][j] = 0
        else:
            gradiente[i][j] = 1


plt.imshow(gradiente,cmap = 'gray')
ne = plt.show()
```

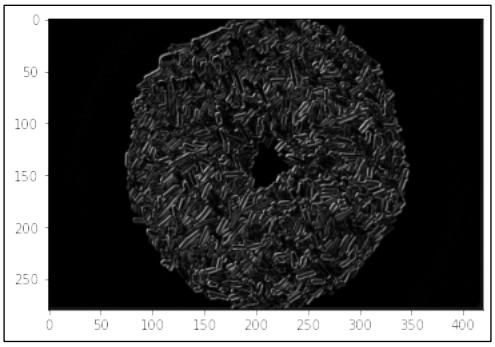


3.- Diseña un programa que implemente la detección de bordes siguiendo la dirección de la diagonal ↘

Para lograr la diagonal, el programa debe comparar cada celda con la que tiene a un paso al lado y un paso hacia abajo, todas excepto la última fila y la última columna.

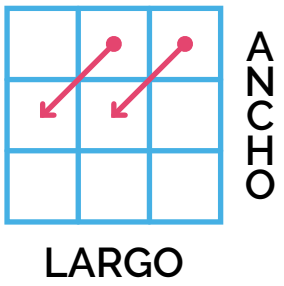



```
diagonal1 = np.zeros(shape=(ancho,largo))

for i in range(0,ancho-1):
    for j in range(0,largo-1):
        diagonal1[i][j] = abs(imagen[i+1][j+1] - imagen[i][j])

plt.imshow(diagonal1,cmap = 'gray')
ne = plt.show()
```



4.- Diseña un programa que implemente la detección de bordes siguiendo la dirección de la diagonal ↙

Para esta diagonal, el programa debe comparar cada celda con la que tiene a un paso hacia abajo y un paso atrás, todas excepto la última fila y la última columna.




```
diagonal2 = np.zeros(shape=(ancho,largo))

for i in range(0,ancho-1):
    for j in range(0,largo-1):
        diagonal2[i][j] = abs(imagen[i+1][j-1] - imagen[i][j])
        if j != 0:
            j -= 1

plt.imshow(diagonal2,cmap = 'gray')
ne = plt.show()
```

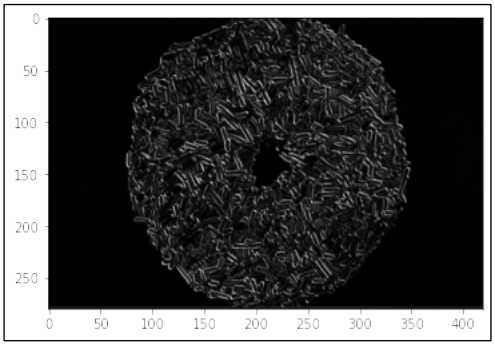
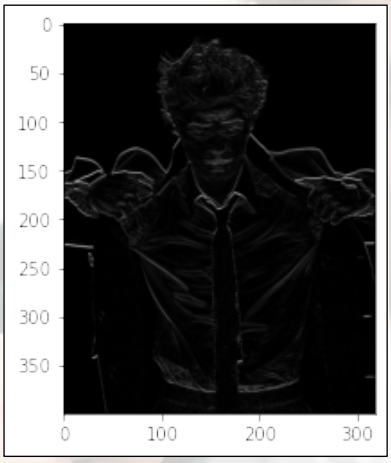
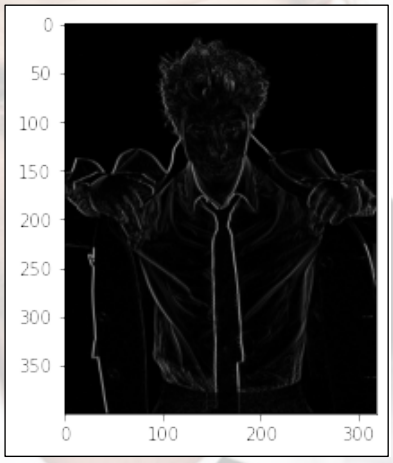


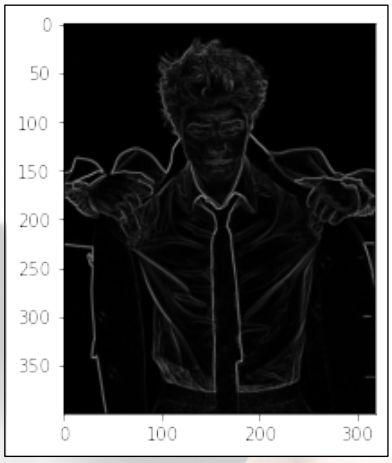
IMAGEN 1



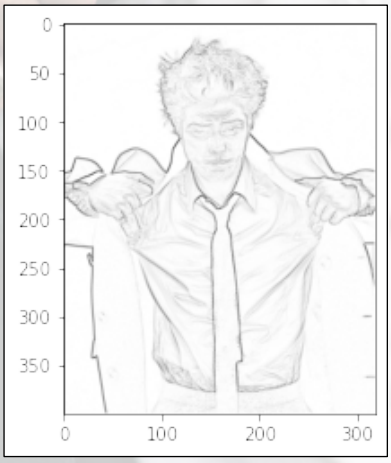
Derivada de x



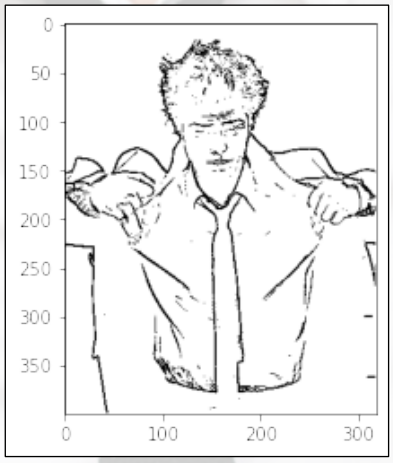
Derivada de y



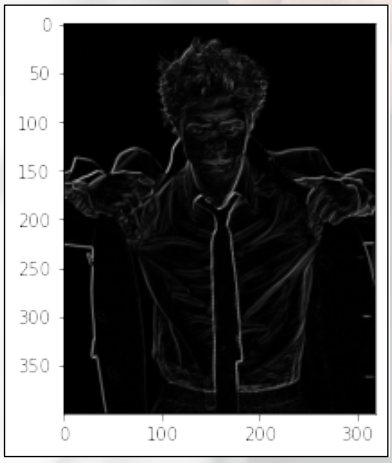
Gradiente



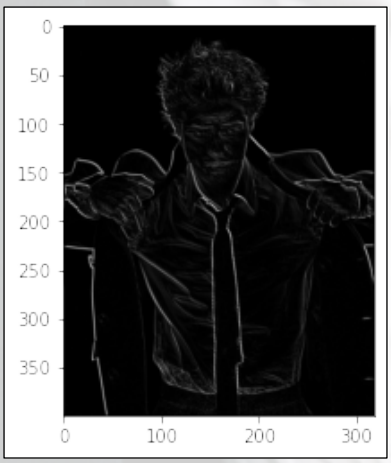
Gradiente Invertida



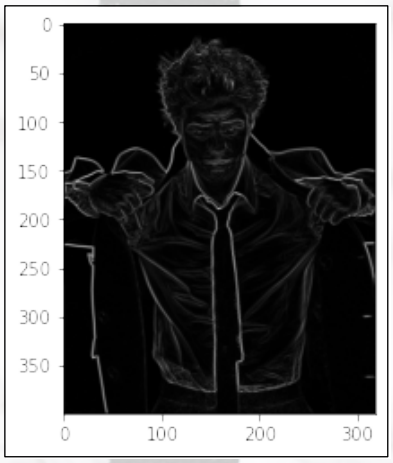
Umbral



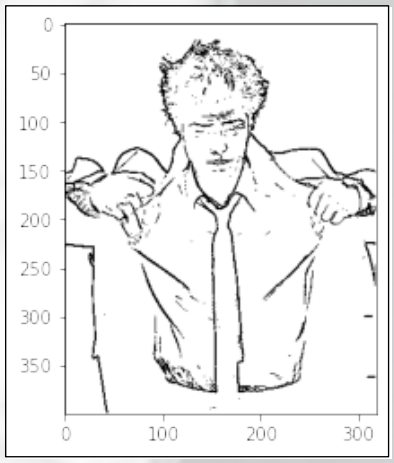
Diagonal ↘



Diagonal ↙

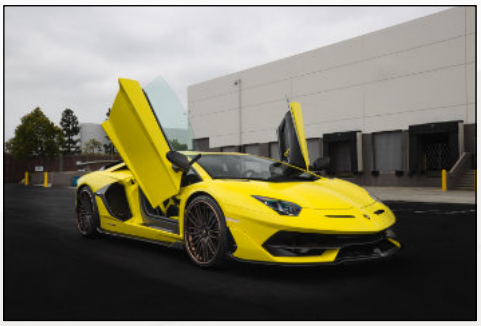


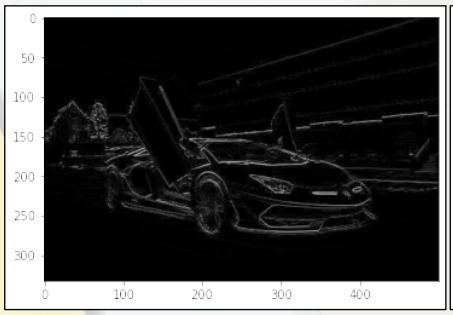
Suma Diagonales



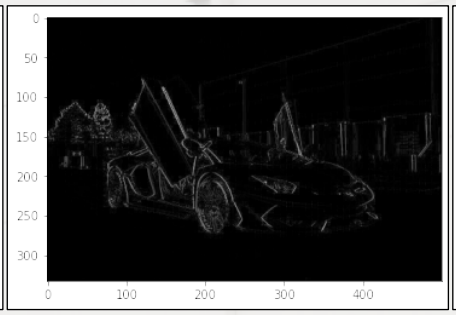
Diagonales Invertidas

IMAGEN 2

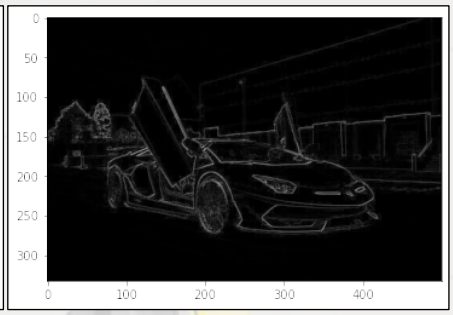




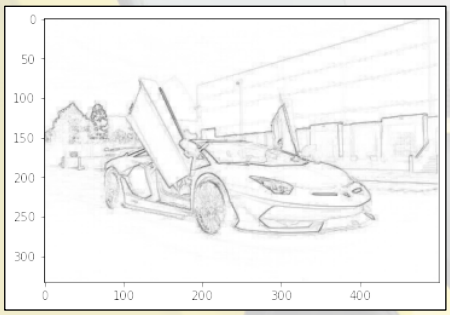
Derivada de x



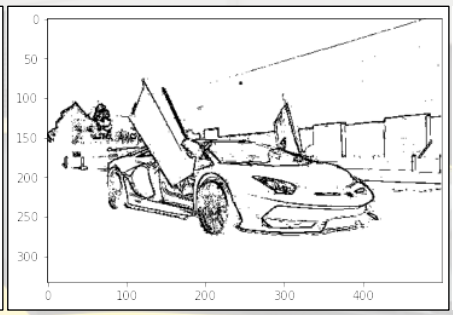
Derivada de y



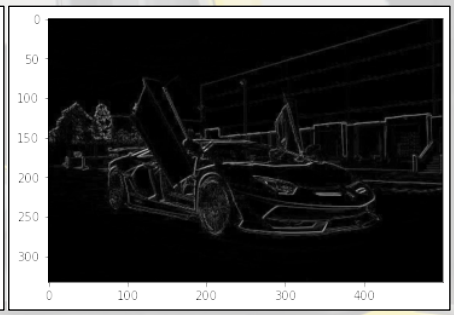
Gradiente



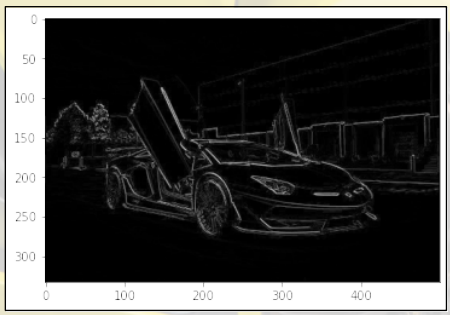
Gradiente Invertida



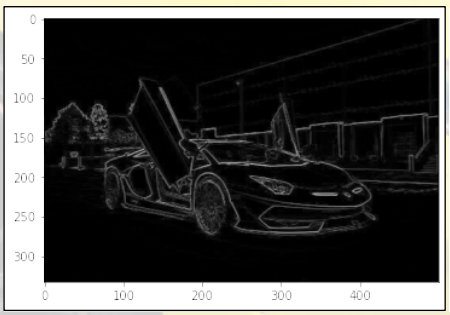
Umbral



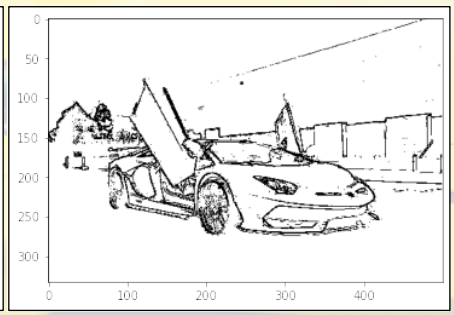
Diagonal ↘



Diagonal ↙

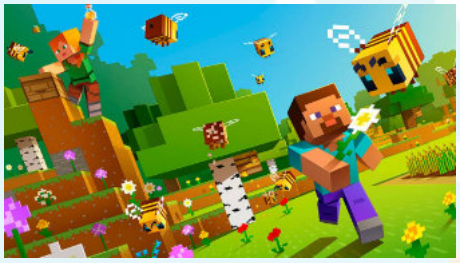


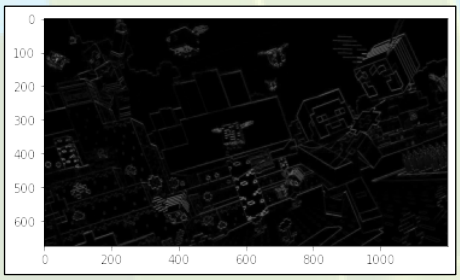
Suma Diagonales



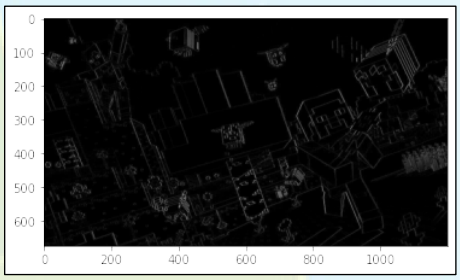
Diagonales Invertidas

IMAGEN 3

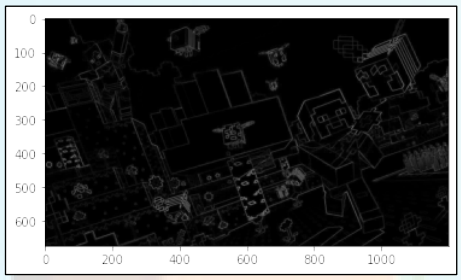




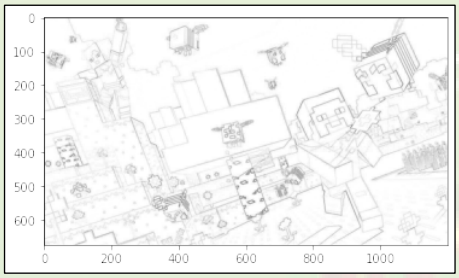
Derivada de x



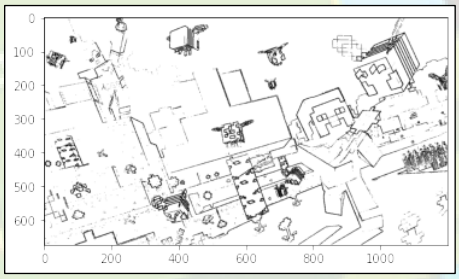
Derivada de y



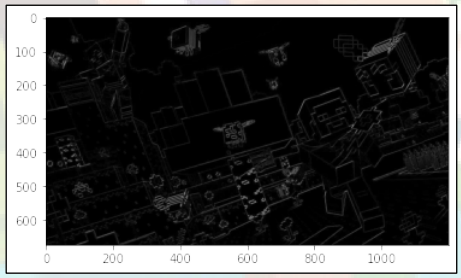
Gradiente



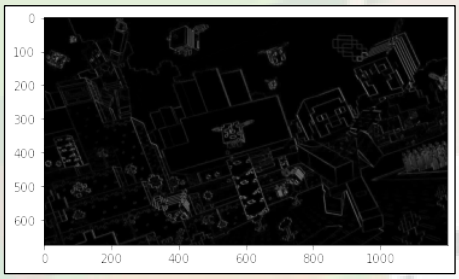
Gradiente Invertida



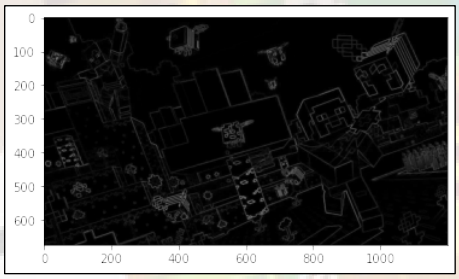
Umbral



Diagonal ↘



Diagonal ↙



Suma Diagonales

Diagonales Invertidas

REFLEXIONES Y CONCLUSIONES

1. ¿Qué aplicaciones le podrías dar a la detección automática de bordes?
   Se puede utilizar para recortar imágenes, reconocer objetos, los autos inteligentes lo pueden utilizar para identificar peatones, se puede usar para convertir una foto a texto en tiempo real, también para el reconocimiento facial, encriptación de imágenes, etc.

2. ¿Qué sucedería si hubiéramos conservado el signo de las derivadas parciales?
   Obtenemos una imagen con bordes poco uniformes, lo que hace que sea más difícil resaltar los bordes.

3. ¿Se te ocurre alguna aplicación de detección de bordes en la que sea de utilidad el signo de la derivada?
   En realidad no mucho, pero quizás pueda ser útil para detectar si hay mas claros o más oscuros en alguna parte de la imagen.