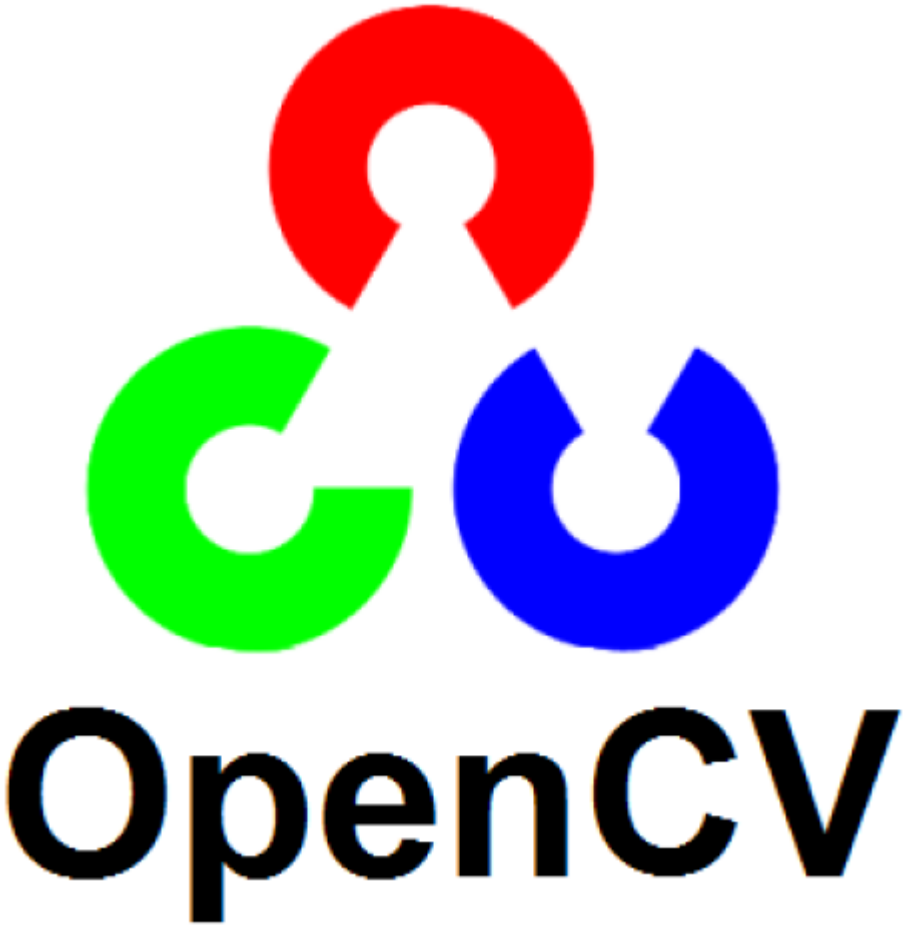
Introducción a la Visión por Computador

Noviembre 2020



1. Imágenes y Matrices

La estructura más importante en una visión artificial es, sin duda, las imágenes. La imagen en visión artificial es una representación del mundo físico capturado con un dispositivo digital. Esta imagen es solo una secuencia de números almacenados en un formato de matriz, como se muestra en la siguiente imagen. Cada número es una medida de la intensidad de la luz para la longitud de onda considerada (por ejemplo, en rojo, verde o azul en imágenes en color) o para un rango de longitud de onda (para dispositivos pancromáticos). Cada punto de una imagen se llama píxel (para un elemento de imagen), y cada píxel puede almacenar uno o más valores dependiendo de si es una imagen gris, negra o blanca (también llamada imagen binaria) que almacena solo uno valor, como 0 o 1, una imagen de nivel de escala de grises que puede almacenar solo un valor, o una imagen en color que puede almacenar tres valores. Estos valores suelen ser números enteros entre 0 y 255, pero puede usar el otro rango. Por ejemplo, 0 a 1 en números de coma flotante como HDRI (imágenes de alto rango dinámico) o imágenes térmicas. La imagen es almacenado en un formato matricial, donde cada pixel tiene una posicion y puede ser 1 referenciado por su fila y columna. En el caso de una imagen en escala de grises una matris simple representa la imagen

159	165	185	187	185	190	189	198	193	197	184	152	123
174	167	186	194	185	196	204	191	200	178	149	129	125
168	184	185	188	195	192	191	195	169	141	116	115	129
178	188	190	195	196	199	195	164	128	120	118	126	135
188	194	189	195	201	196	166	114	113	120	128	131	129
187	200	197	198	190	144	107	106	113	120	125	125	125
198	195	202	183	134	98	97	112	114	115	116	116	118
194	206	178	111	87	99	97	101	107	105	101	97	95
206	168	107	82	80	100	102	91	98	102	104	99	72
160	97	80	86	80	92	80	79	71	74	81	81	64
98	66	76	86	76	83	72	71	55	53	61	61	56
60	76	74	70	67	64	63	60	55	49	54	52	54

En el caso de una imagen a color, como mostramos en la figura, sera represntada por una matriz de ancho * alto * numero de colores (RGB)

2. Introducción a OpenCV

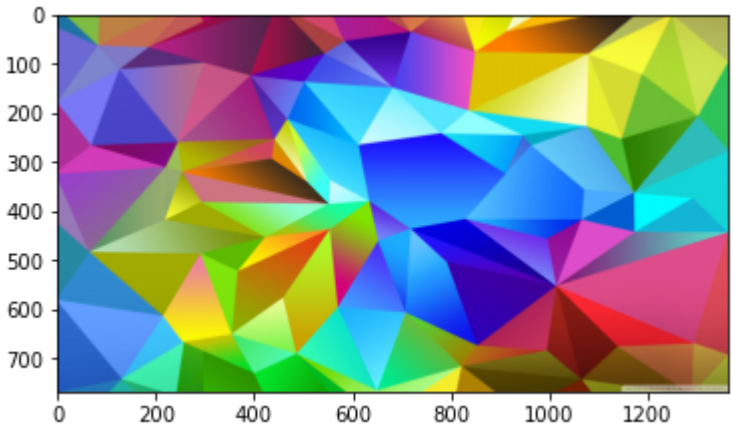
2.1 Leer, escribir y mostrar una imagen

OpenCV provee las funciones imread() e imwrite() que soportan varios formatos de archivos para trabajar con imágenes, la imagen debe estar en el directorio de trabajo actual o se debe pasar la dirección absoluta o relativa de la imagen.

imread()

In [1]:
```
#importar librería vc
import cv2 as cv
import matplotlib.pyplot as plt
#lectura de una imagen
img = cv.imread("imagen1.jpg")
#mostrar imagen
plt.imshow(img)
```

Out[1]:
```
<matplotlib.image.AxesImage at 0x1126844fe20>
```



Por defecto imread() retorna una imagen en formato de color BGR, incluso si usamos una imagen en formato de escala de grises. BGR representa el mismo espacio de color que RGB pero el orden de byte esta invertido.

Modos de imread() (Banderas)

IMREAD_UNCHANGED Python: cv.IMREAD_UNCHANGED Si se establece, devuelve la imagen cargada como está (con canal alfa; de lo contrario, se recorta). Ignore la orientación EXIF.

IMREAD_GRAYSCALE Python: cv.IMREAD_GRAYSCALE Si está configurado, siempre convierta la imagen a la imagen en escala de grises de un solo canal (conversión interna del códec).

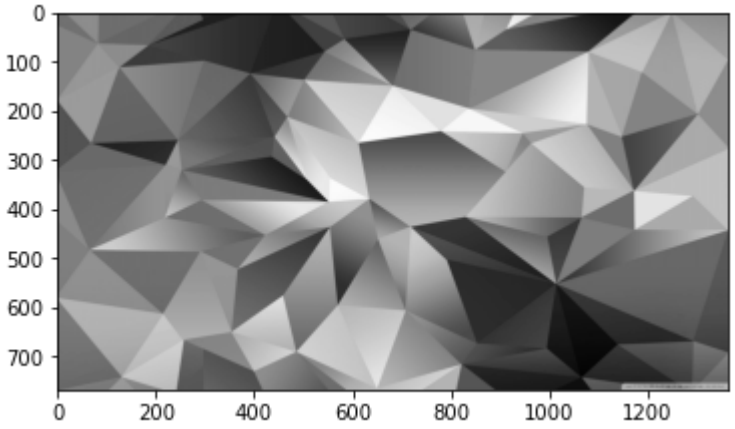
IMREAD_COLOR Python: cv.IMREAD_COLOR Si está configurado, siempre convierta la imagen a la imagen en color BGR de 3 canales.

IMREAD_ANYCOLOR Python: cv.IMREAD_ANYCOLOR Si se establece, la imagen se lee en cualquier formato de color posible.

cv.IMREAD_GRAYSCALE

In [4]:
```
import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread("imagen1.jpg", cv.IMREAD_GRAYSCALE)
plt.imshow(img, cmap="gray")
```

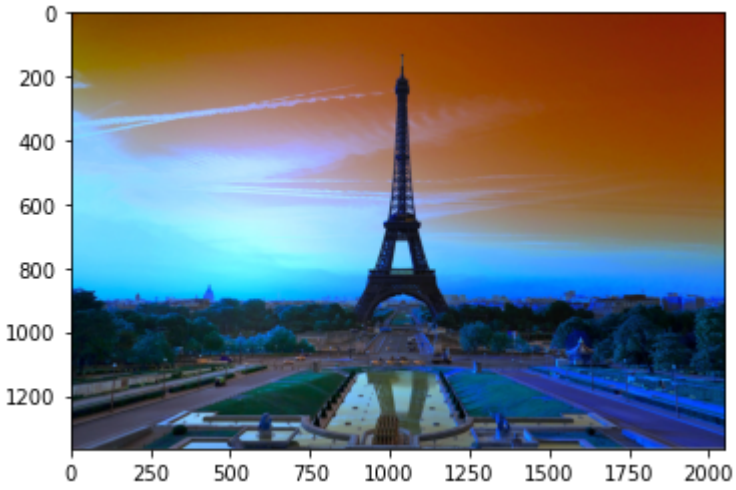
Out[4]: <matplotlib.image.AxesImage at 0x112686f58b0>



cv. IMREAD_COLOR

In [9]:
```
import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread("imagen2.jpg", cv.IMREAD_COLOR)
plt.imshow(img, cmap="gray")
```

Out[9]: <matplotlib.image.AxesImage at 0x1126b22eeb0>



In [10]:
```
#Transformar de BGR a RGB
img = cv.imread("imagen2.jpg", cv.IMREAD_COLOR)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[10]: <matplotlib.image.AxesImage at 0x1126b578910>

imshow()

In [13]:
```
import cv2 as cv
img = cv.imread("imagen1.jpg")
#muestra la imagen en una ventana
cv.imshow("Imagen",img)
#cierra la imagen
cv.waitKey(0)
cv.destroyAllWindows()
```

imwrite()

La funcion imwrite nos permite escribir en disco o guardar una imagen, le pasamos como argumento la ruta donde donde guardar con el nombre a guardar y la extencion, y como segundo argumento la imagen a guardar si todo sale sin errores devuelve True

In [2]:
```
import cv2 as cv
img = cv.imread("imagen1.jpg")
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imwrite("result/imagen1gray.jpg",img_gray)
```

Out[2]: True

2.2 Lectura y Escritura de Archivos de Video

OpenCV dispone de las clases **VideoCapture()** y **VideoWriter()** que soporta varios formatos de archivos de video Los formatos admitidos varían según el sistema, pero siempre deben incluir AVI. A través de su método read (), una clase de VideoCapture puede sondearse para nuevos cuadros hasta llegar al final de su archivo de video. Cada cuadro es una imagen en formato BGR.

Reproducir video desde un archivo

In [15]:
```
import numpy as np
import cv2 as cv

# crear un objeto VideoCapture su parametro sera la direccion del video
cap = cv.VideoCapture("videoTBT.mp4")
# creamos un bucle

while(cap.isOpened()):

    ret,frame = cap.read()
    if ret:

        cv.imshow("Video Motor a escala", frame)

        if(cv.waitKey(10) & 0xFF ==ord("q")):
            break

    else:
        break
cap.release()
cv.destroyAllWindows()
```

Acceder a la cámara

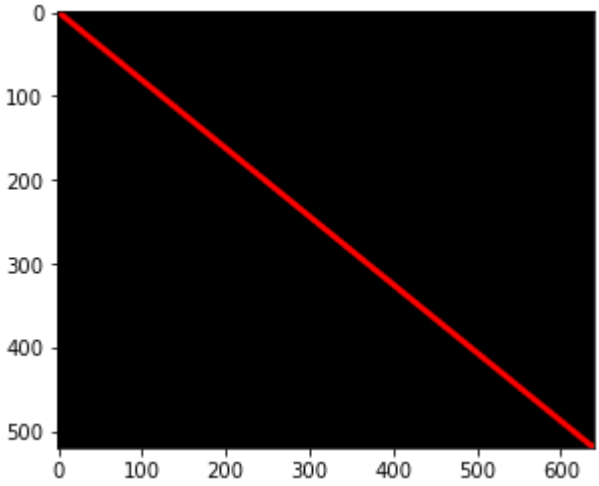
In [16]:
```
import numpy as np
import cv2 as cv

# crear un objeto VideoCapture su parametro sera la direccion del video (0 par
a webcam)
cap = cv.VideoCapture(0)
# creamos un bucle

while(cap.isOpened()):

    ret,frame = cap.read()
    if ret:

        cv.imshow("Video Motor a escala", frame)

        if(cv.waitKey(10) & 0xFF ==ord("q")):
            break

    else:
        break
cap.release()
cv.destroyAllWindows()
```

Funciones para dibujar figuras geométricas

Línea

In [5]:
```
#función cv.line(imagen, coordenadas iniciales,
#coordenadas finales, color, grosor)
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
#crear una imagen
img = np.zeros((520,640,3), np.uint8)
#dibujar la linea
img = cv.line(img,(0,0),(640,520),(255,0,0),6)
#mostrar imagen
plt.imshow(img)
```

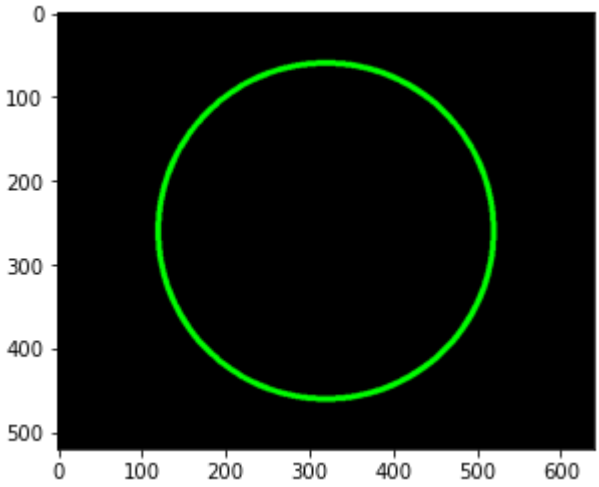
Out[5]:
```
<matplotlib.image.AxesImage at 0x21461bc1940>
```



Dibujar un círculo

In [9]:
```
#función cv.circle()
#parámetros: img, coordenadas centro, radio, color, grosor
#función cv.line(imagen, coordenadas iniciales,
#coordenadas finales, color, grosor)
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
#crear una imagen
img = np.zeros((520,640,3), np.uint8)
#dibujar círculo
img = cv.circle(img,(320,260),200,(0,255,0),5)
#mostrar imagen
plt.imshow(img)
```

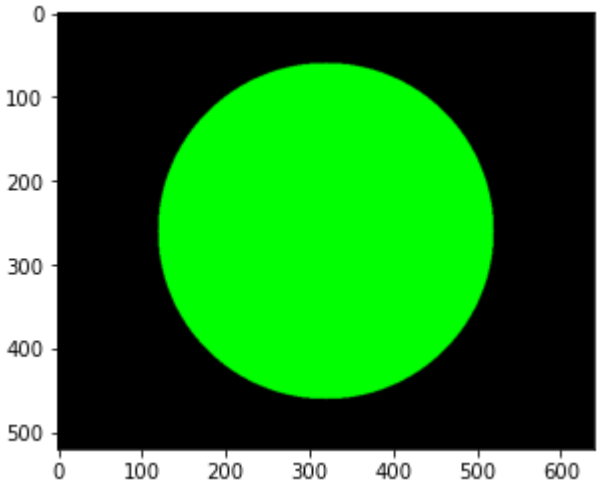
Out[9]: <matplotlib.image.AxesImage at 0x21461d7c400>



Círculo relleno

In [17]:
```
#función cv.circle()
#parámetros: img, coordenadas centro, radio, color, grosor
#función cv.line(imagen, coordenadas iniciales,
#coordenadas finales, color, grosor)
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
#crear una imagen
img = np.zeros((520,640,3), np.uint8)
#dibujar círculo (-1 para rellenar el círculo)
img = cv.circle(img,(320,260),200,(0,255,0),-1)
#mostrar imagen
plt.imshow(img)
```

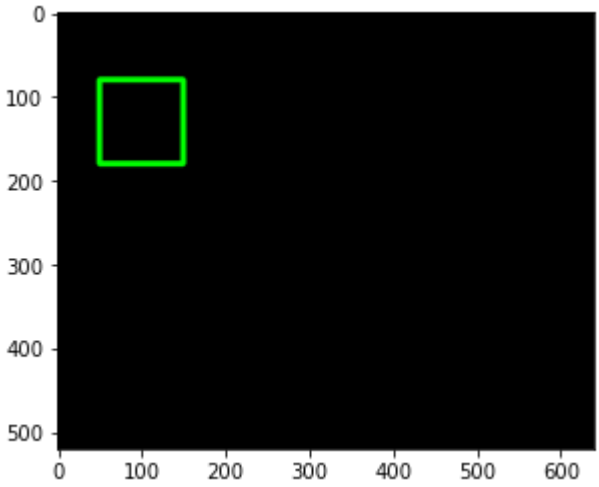
Out[17]:
```
<matplotlib.image.AxesImage at 0x1126bbcc910>
```



Dibujar rectángulo

In [10]:
```
#función cv.rectangle()
#parámetros: img, coordenadas esquina superior izq, coordenada esquina inf de
r, color, grosor

import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
#crear una imagen
img = np.zeros((520,640,3), np.uint8)
#dibujar rectangulo
img = cv.rectangle(img,(50,80),(150,180),(0,255,0),5)
#mostrar imagen
plt.imshow(img)
```

Out[10]: <matplotlib.image.AxesImage at 0x21461dcafa0>



Captura de eventos del Mouse

OpenCV permite que las ventanas con nombre se creen, redibujen y destruyan usando las funciones **namedWindow(), imshow() y destroyWindow()**. Además, cualquier ventana puede capturar la entrada del teclado a través de la función **waitKey ()** y la entrada del mouse a través de la función **setMouseCallback()**. Veamos un ejemplo donde mostramos cuadros de entrada de cámara en vivo

Función setMouseCallback()

In [13]:
```
import cv2 as cv
clicked = False
#funcion encargada de capturar el mouse
def onMouse(evento,x,y,flags,param):
    #variable glogal
    global clicked

    if(evento == cv.EVENT_LBUTTONUP):
        clicked = True
#crear objeto video cv.VideoCapture()
camCap = cv.VideoCapture(0)
cv.namedWindow("MyWindow")
#configurar envío de eventos a función onMouse
cv.setMouseCallback("MyWindow",onMouse)

print("Mostrar estado cámara, click en la ventana para detener")
ret,frame = camCap.read()

while(ret and cv.waitKey(1)==-1 and not clicked):
    cv.imshow("MyWindow", frame)
    ret, frame = camCap.read()
camCap.release()
cv.destroyAllWindows()
```

```
Mostrar estado cámara, click en la ventana para detener
```

El argumento para waitKey () es un número de milisegundos para esperar la entrada del teclado. El valor de retorno es -1 (lo que significa que no se ha presionado ninguna tecla) o un código de tecla ASCII, como 27 para Esc. Podemos enumerar todos los eventos del mouse disponibles con el siguiente codigo:

In [18]:
```
import cv2 as cv
events = [i for i in dir(cv) if 'EVENT' in i ]
events
```

Out[18]:
```
['EVENT_FLAG_ALTKEY',
 'EVENT_FLAG_CTRLKEY',
 'EVENT_FLAG_LBUTTON',
 'EVENT_FLAG_MBUTTON',
 'EVENT_FLAG_RBUTTON',
 'EVENT_FLAG_SHIFTKEY',
 'EVENT_LBUTTONDBLCLK',
 'EVENT_LBUTTONDOWN',
 'EVENT_LBUTTONUP',
 'EVENT_MBUTTONDBLCLK',
 'EVENT_MBUTTONDOWN',
 'EVENT_MBUTTONUP',
 'EVENT_MOUSEHWHEEL',
 'EVENT_MOUSEMOVE',
 'EVENT_MOUSEWHEEL',
 'EVENT_RBUTTONDBLCLK',
 'EVENT_RBUTTONDOWN',
 'EVENT_RBUTTONUP']
```

Dibujar circulos en donde se haga click

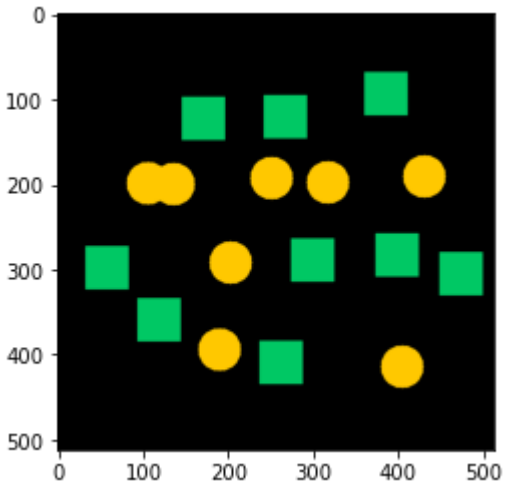
In [ ]:
```
import numpy as np
import cv2 as cv
def draw_circle(event, x, y, flags, param):

    if(event == cv.EVENT_LBUTTONDBLCLK):

        cv.circle(img, (x, y), 25, (255,0,0), -1 )

img = np.zeros((512,512,3), np.uint8)

cv.namedWindow('Dibujando Circulos')
cv.setMouseCallback('Dibujando Circulos', draw_circle)

while(True):
    cv.imshow('Dibujando Circulos',img)
    if (cv.waitKey(1) & 0xFF ==ord('q')):
        break

cv.destroyAllWindows()
```

TAREA 1

Realizar un ejercicio capturando eventos con el mouse donde se pulse el Botón izquierdo dibuje un circulo, pulsamos el botón derecho dibuja un rectángulo

In [30]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
#Función que dibuja círculos o rectángulos
def draw_form(event, x, y, flags, param):

    #cuando se presione el botón izquierdo (círculo)
    if(event == cv.EVENT_LBUTTONUP):
        cv.circle(img, (x, y), 25, (0,200,255), -1 )

    #cuando se presione el botón derecho (rectángulo)
    if(event == cv.EVENT_RBUTTONUP):
        cv.rectangle(img, (x-25, y-25), (x+25,y+25),(100,200,0),-1)

#crea una imagen
img = np.zeros((512,512,3), np.uint8)

cv.namedWindow('TAREA 1')
cv.setMouseCallback('TAREA 1', draw_form)

while(True):
    cv.imshow('TAREA 1',img)
    if (cv.waitKey(1) & 0xFF ==ord('q')):
        break

cv.destroyAllWindows()
img = cv.cvtColor(img,cv.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[30]: <matplotlib.image.AxesImage at 0x21464257df0>



In [ ]:

In [ ]: