

# Cristopher Coronado - Tarea 1

November 21, 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 referenciado por su fila y columna. En el caso de una imagen en escala de grises una matriz simple representa la imagen

En el caso de una imagen a color, como mostramos en la figura, sera representada 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()` y `imwrite()` que soportan varios formatos de archivos para trabajar con imagenes, la imagen debe estar en el directorio de trabaja actual o debes pasar la direccion absoluta o relativa de la ubicacion de la imagen.

#### 2.1.1 `imread()`

```
[5]: # Importar las librerías
import cv2 as cv
import matplotlib.pyplot as plt

# lectura de una imagen
img = cv.imread("./imgs/t1/superman.jpg")

# mostrar la imagen
plt.imshow(img)
```

[5]: <matplotlib.image.AxesImage at 0x23c75312e08>



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.

#### 2.1.2 Modos `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).

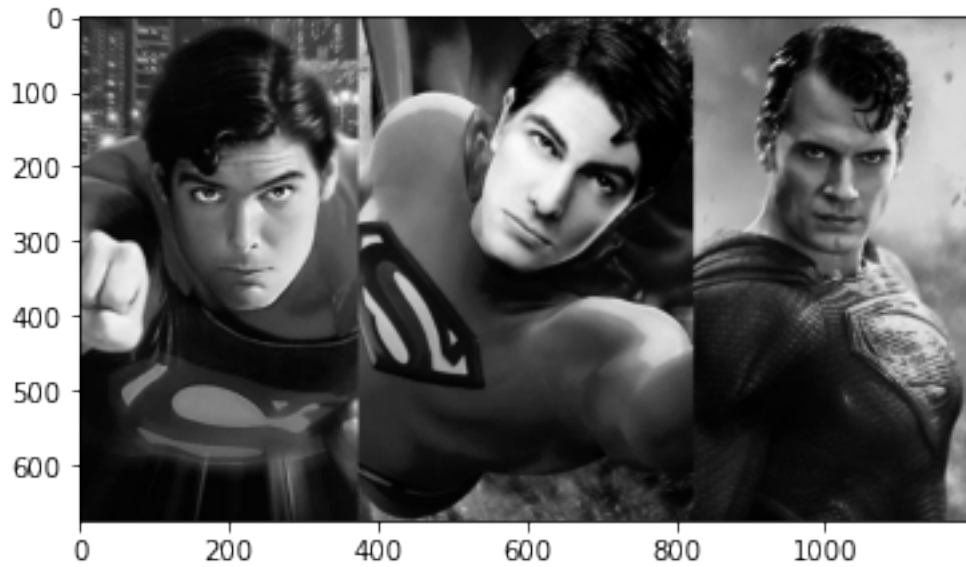
`IMREAD_COLOR` Python: `cv.IMREAD_COLOR` Si está configurado, siempre convierta la imagen a la imagen en color BGR de 3 canales.

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

#### 2.1.3 `cv.IMREAD_GRAYSCALE`

```
[6]: import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread("./imgs/t1/superman.jpg", cv.IMREAD_GRAYSCALE)
plt.imshow(img, cmap="gray")
```

[6]: <matplotlib.image.AxesImage at 0x23c755df988>



#### 2.1.4 cv.IMREAD\_COLOR

```
[7]: import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread("./imgs/t1/superman.jpg", cv.IMREAD_COLOR)
plt.imshow(img, cmap="gray")
```

[7]: <matplotlib.image.AxesImage at 0x23c7564e308>



```
[8]: img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

```
[8]: <matplotlib.image.AxesImage at 0x23c756b5948>
```



### 2.1.5 imshow()

```
[ ]: import cv2 as cv
# Leemos la imagen
img = cv.imread("./imgs/t1/superman.jpg")
cv.imshow("Horizon", img)
# Cerramos la imagen
cv.waitKey(0)
cv.destroyAllWindows()
```

### 2.1.6 imwrite()

La funcion imwrite nos permite escribir en disco o guardar una imagen, le pasamos como argumento la ruta 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

```
[ ]: import cv2 as cv
img = cv.imread("./imgs/t1/superman.jpg")
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imwrite("./imgs/t1/superman_gray.jpg", img_gray)
```

## 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

```
[ ]: import numpy as np
import cv2 as cv
# crear un objeto VideoCapture su parametro sera la direccion del video
cap = cv.VideoCapture("./videos/MujerMaravilla2.mp4")
# creamos un bucle
while(cap.isOpened()):

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

        cv.imshow("Trailer Mujer Maravilla 2", frame)

        if(cv.waitKey(10) & 0xFF ==ord("q")):
            break
    else:
        break
cap.release()
cv.destroyAllWindows()
```

Acceder a la cámara

```
[ ]: import numpy as np
import cv2 as cv
# crear un objeto VideoCapture su parametro sera la direccion del video (0 para
↪webcam)
cap = cv.VideoCapture(0)
# creamos un bucle
while(cap.isOpened()):

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

        cv.imshow("Cámara Activa", frame)

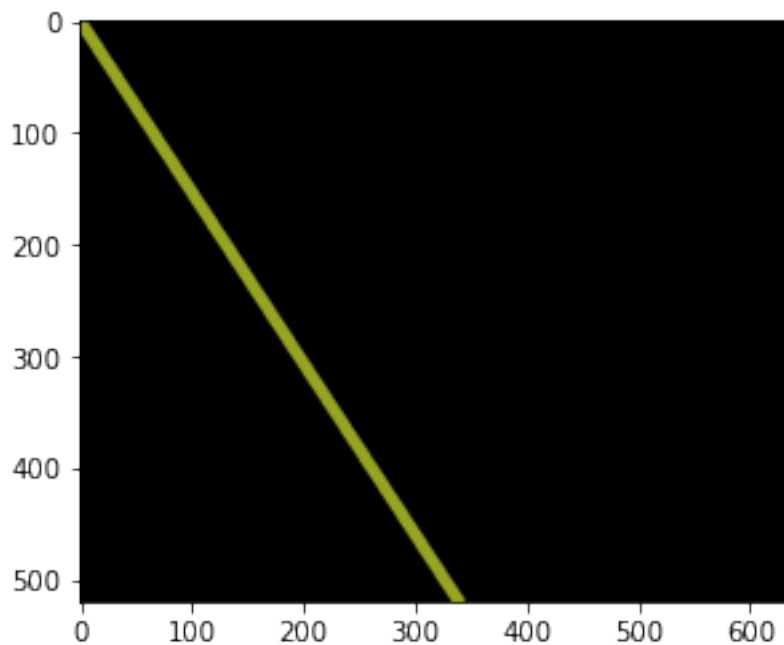
        if(cv.waitKey(10) & 0xFF ==ord("q")):
            break
    else:
        break
cap.release()
cv.destroyAllWindows()
```

Funciones para dibujar figuras geométricas

Línea

```
[9]: #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),(340,520),(153,164,38),10)
#mostrar imagen
plt.imshow(img)
```

[9]: <matplotlib.image.AxesImage at 0x25ff1dd99c8>



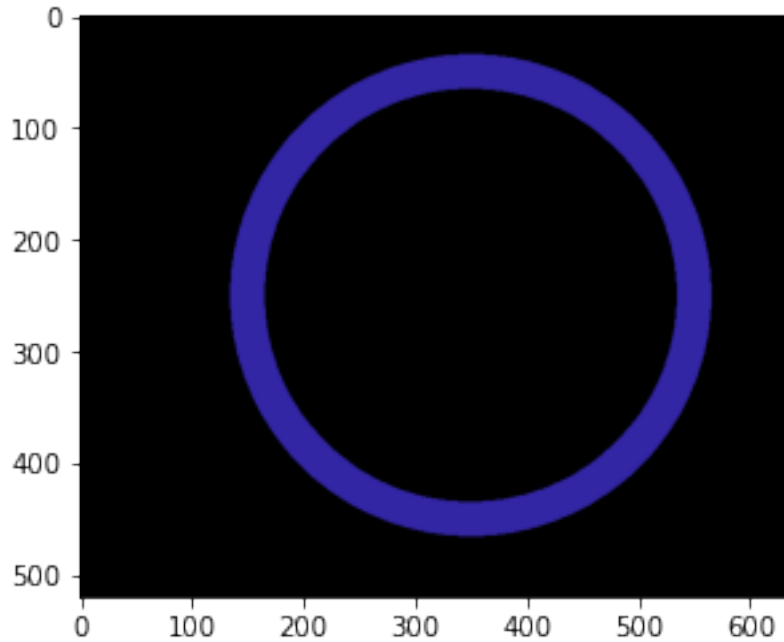
Dibujar un círculo

```
[7]: #función cv.circle()
#parámetros: img, coordenadas centro, radio, 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,(350,250),200,(50,38,164),30)
```

```
#mostrar imagen  
plt.imshow(img)
```

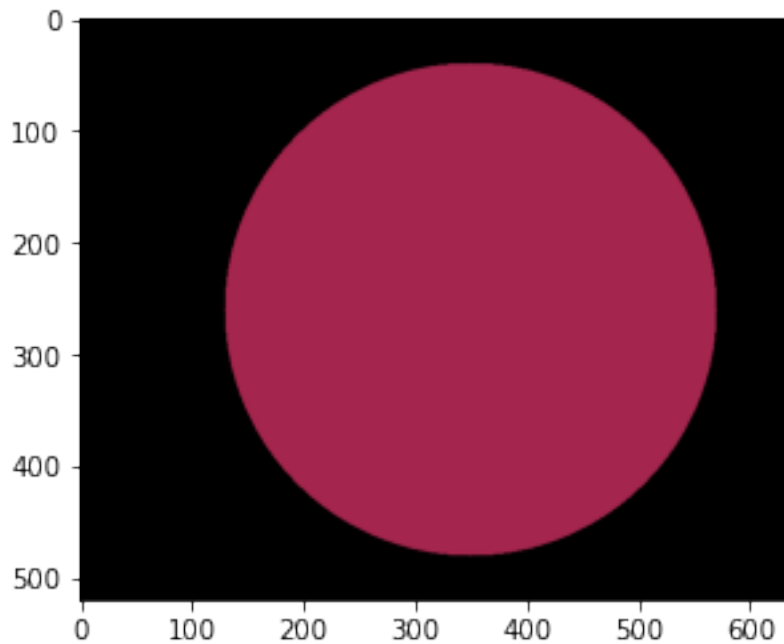
[7]: <matplotlib.image.AxesImage at 0x25ff1e6e908>



Círculo relleno

```
[11]: #función cv.circle()  
      #parámetros: img, coordenadas centro, radio, 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,(350,260),220,(164,38,78),-1)  
      #mostrar imagen  
      plt.imshow(img)
```

[11]: <matplotlib.image.AxesImage at 0x25ff200d308>

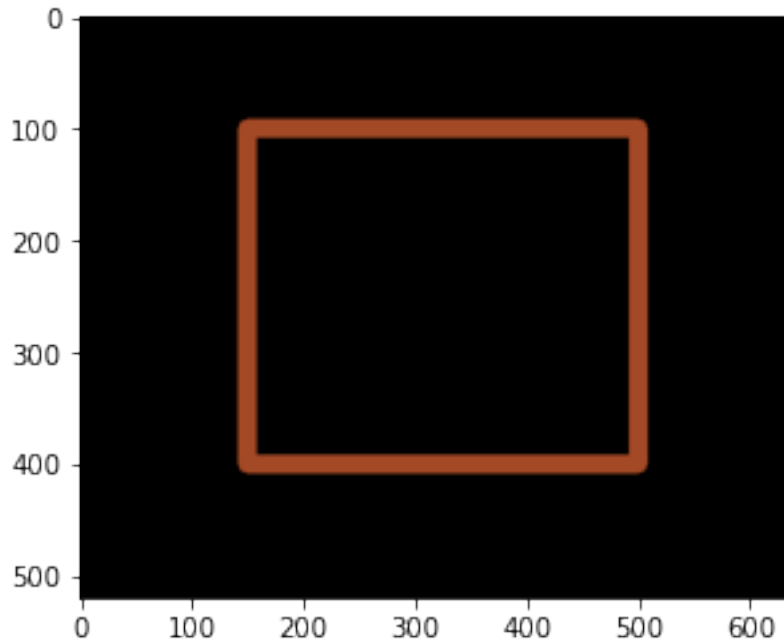


Dibujar rectángulo

```
[14]: #función cv.rectangle()
      #parámetros: img, coordenadas esquina superior izq, coordenada esquina inf der,
      ↪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 rectángulo
      img = cv.rectangle(img,(150,100),(500,400),(164,73,38),15)
      #mostrar imagen
      plt.imshow(img)
```

```
[14]: <matplotlib.image.AxesImage at 0x25ff214d108>
```





### 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()`

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

```

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

```

```

[16]: ['EVENT_FLAG_ALTKEY',
      'EVENT_FLAG_CTRLKEY',
      'EVENT_FLAG_LBUTTON',
      'EVENT_FLAG_MBUTTON',
      'EVENT_FLAG_RBUTTON',
      'EVENT_FLAG_SHIFTKEY',
      'EVENT_LBUTTONDOWNCLK',
      'EVENT_LBUTTONDOWN',
      'EVENT_LBUTTONUP',
      'EVENT_MBUTTONDOWNCLK',
      'EVENT_MBUTTONDOWN',
      'EVENT_MBUTTONUP',
      'EVENT_MOUSEHWHEEL',
      'EVENT_MOUSEMOVE',
      'EVENT_MOUSEWHEEL',
      'EVENT_RBUTTONDOWNCLK',
      'EVENT_RBUTTONDOWN',
      'EVENT_RBUTTONUP']

```

Dibujar círculos en donde se haga click

```

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

    if(event == cv.EVENT_LBUTTONDOWNCLK):

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

img = np.zeros((512,512,3), np.uint8)
cv.namedWindow('Dibujando círculos')
cv.setMouseCallback('Dibujando círculos', draw_circle)
while(True):
    cv.imshow('Dibujando círculos',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.

```

[1]: # Importar librerías
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

# Definir función para dibujar las figuras
def draw_figures(event, x, y, flags, param):

    # Dibujar círculos al dar click izquierdo
    if(event == cv.EVENT_LBUTTONDOWN):
        cv.circle(img, (x, y), 25, (255,0,0), 5 )

    # Dibujar rectangulos al dar click derecho
    if(event == cv.EVENT_RBUTTONDOWN):
        cv.rectangle(img, (x-25, y-25), (x+25, y+25), (100, 200, 0), 5)

# Crear la imagen
img = np.zeros((512, 512, 3), np.uint8)

cv.namedWindow('Tarea 1')
cv.setMouseCallback('Tarea 1', draw_figures)

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

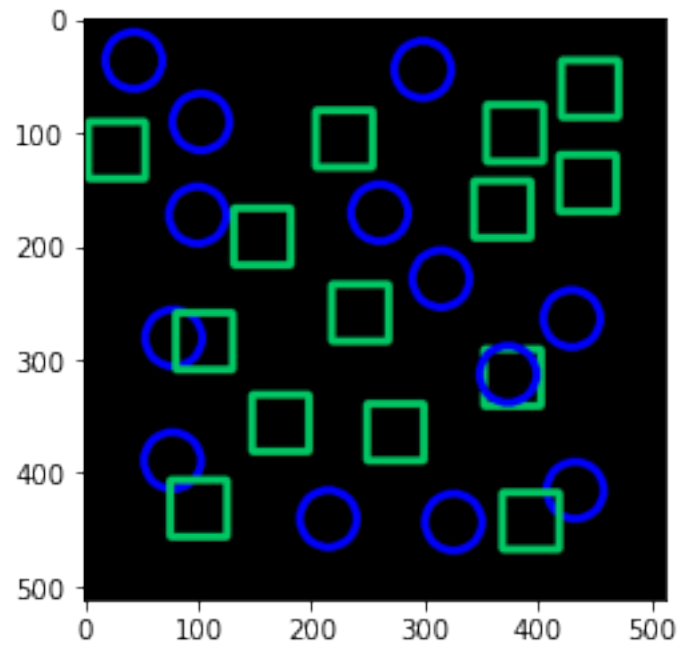
cv.destroyAllWindows()
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)

```

```

[1]: <matplotlib.image.AxesImage at 0x211bc951e48>

```



[ ]: