ApuntesCurso

November 20, 2020

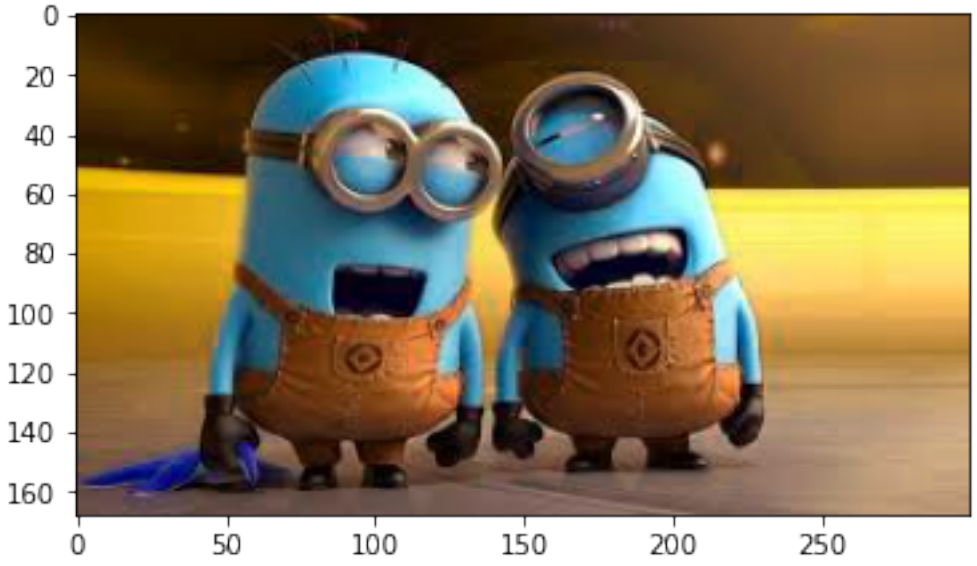
1 OPEN CV

1.0.1 Leer, Escribir y Mostrar una imagen

Leer una imagen Comando: imread()

[1]:
```
#importar la libreria de vision por computadora
import cv2 as cv
import matplotlib.pyplot as plt
#lectura de una imagen
img=cv.imread("img1.jpg")
# mostrar la imagen
plt.imshow(img)
```

[1]:
```
<matplotlib.image.AxesImage at 0x121a7fe7610>
```



Como se observa esta en una esca de azules debido a que esta en BGR entonces se debe cambiar a RGB.

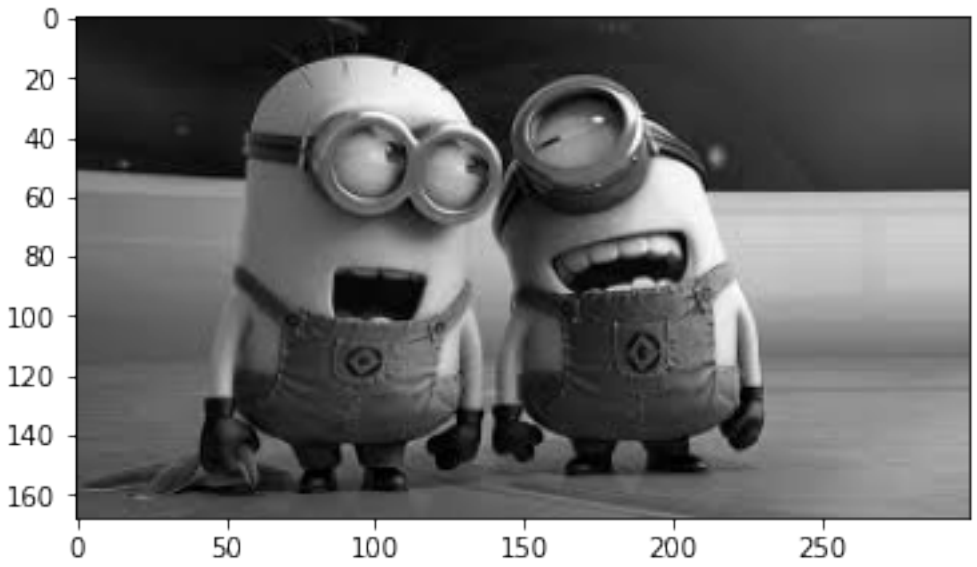
Para mostrar la imagen en una pantalla emergente.

1.0.2 Modos de imread()

Comando para cambiar la imagen a escala de grises: cv.IMREAD_GRAYSCALE

[13]:
```
#importar la libreria de vision por computadora
import cv2 as cv
import matplotlib.pyplot as plt
#lectura de una imagen
img=cv.imread("Imgs\img1.jpg",cv.IMREAD_GRAYSCALE)
# mostrar la imagen
plt.imshow(img, cmap="gray")
```

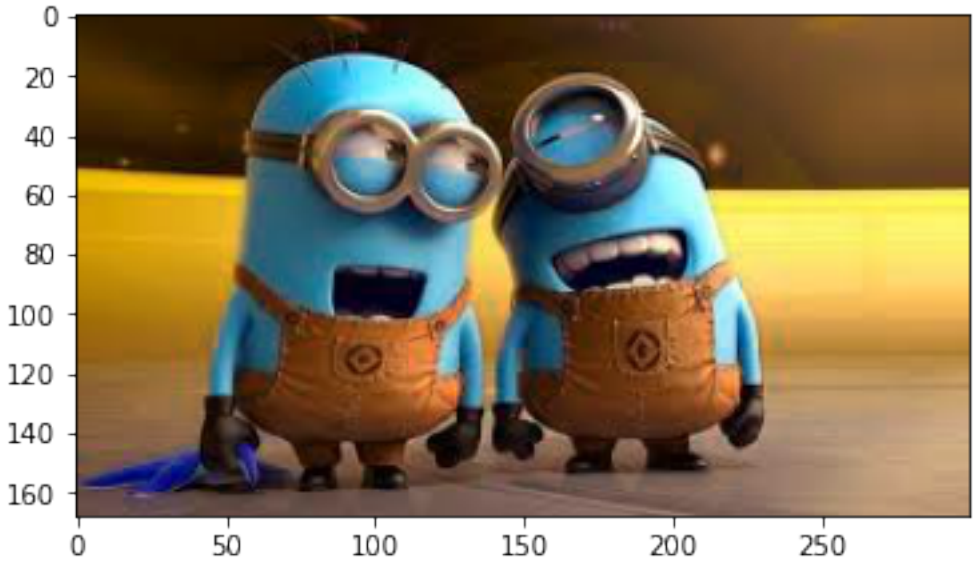
[13]:
```
<matplotlib.image.AxesImage at 0x255aec90af0>
```



Comando para cambiar la imagen a color: cv.IMREAD_COLOR

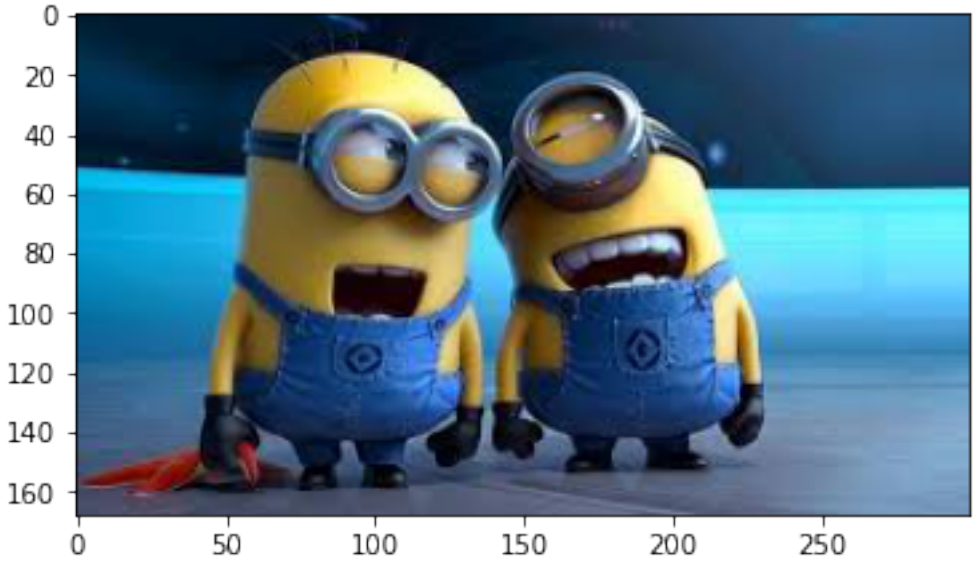
[17]:
```
#importar la libreria de vision por computadora
import cv2 as cv
import matplotlib.pyplot as plt
#lectura de una imagen
img=cv.imread("Imgs\img1.jpg",cv.IMREAD_COLOR)
# mostrar la imagen
plt.imshow(img, cmap="gray")
```

[17]:
```
<matplotlib.image.AxesImage at 0x255aeb7a400>
```



[19]:
```
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

[19]: <matplotlib.image.AxesImage at 0x255aecdc730>



2 Mostrar una imagen en una venta emergente imshow()

[26]:
```
import cv2 as cv
#leer la imagen
img = cv.imread("Imgs/img1.jpg")
cv.imshow("Minions", img)
#cerramos la ventana
cv.waitKey(0)
cv.destroyAllWindows()
```

3 Funcion para guardar la imagen en disco local.

imwrite()

[29]:
```
import cv2 as cv
#leer la imagen
img = cv.imread("Imgs/goku.jpg")
img_gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imwrite("Imgs/gokugris.jpg", img_gray)
```

[29]:
```
True
```

4 Lectura y escritura de archivos de Video recomendado formatos AVI mp4

Reproducir un archivo de video

[30]:
```
import numpy as np
import cv2 as cv


cap = cv.VideoCapture("Videos/iron.mp4")
# creamos un bucle

while(cap.isOpened()):

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

        cv.imshow("Iron Man", frame)

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

    else:
        break
```

```
cap.release()
cv.destroyAllWindows()
```

Acceder a la camara de video.

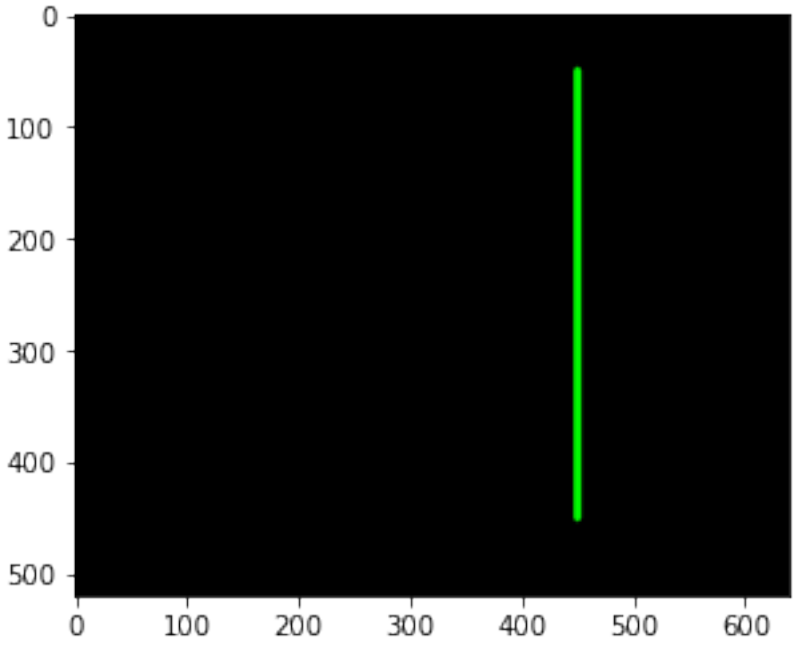
[32]:
```
import numpy as np
import cv2 as cv


cap = cv.VideoCapture(0)
# creamos un bucle

while(cap.isOpened()):

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

        cv.imshow("Camara Local", frame)

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

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

5 FUNCIONES DE DIBUJO

5.0.1 Dibujar linea

[34]:
```
#la funcion cv.line()
#cv.line(img,coordenadas iniciales,coordenadasfinales,color ,grosor)
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

#crear una imagen

img=np.zeros((520,640,3),np.uint8)
# dibujamos la linea en la imagen
cv.line(img,(450,50),(450,450),(0,255,0),6)
#mostrar imagen
plt.imshow(img)
```

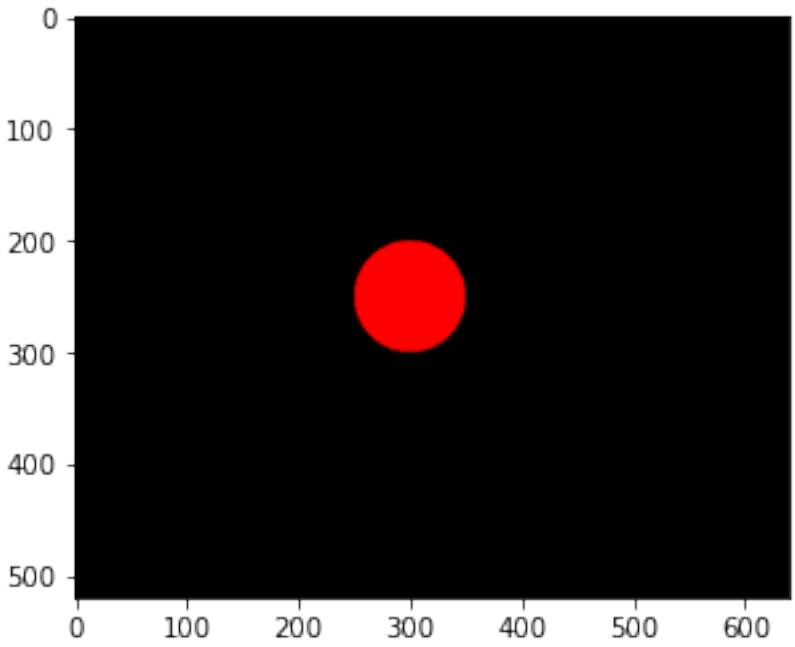
[34]:
```
<matplotlib.image.AxesImage at 0x255ae7cf9d0>
```



Dibujar Circulo

[20]:
```
#la funcion cv.circle()
#paramentros (img,coordenadas centro,radio,color ,grosor)
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

#crear una imagen

img=np.zeros((520,640,3),np.uint8)
# dibujamos la linea en la imagen
cv.circle(img,(300,250),50,(255,0,0),-1)
#mostrar imagen
plt.imshow(img)
```

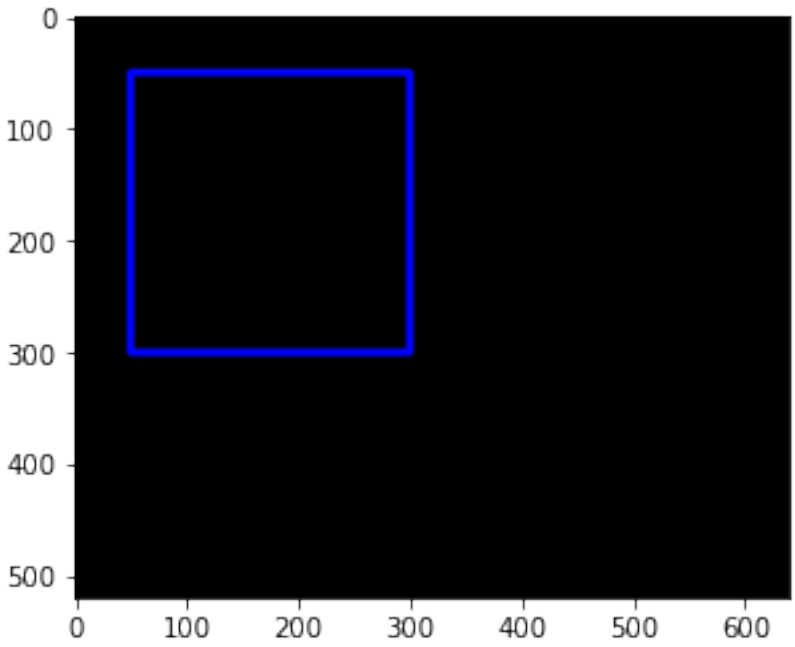
[20]:
```
<matplotlib.image.AxesImage at 0x121b223cc10>
```



6 DibujarRectangulo

[36]:
```
#la funcion cv.rectangle()
#cv.line(img,coordenadas iniciales,coordenadasfinales,color ,grosor)
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

#crear una imagen

img=np.zeros((520,640,3),np.uint8)
# dibujamos la linea en la imagen
cv.rectangle(img,(50,50),(300,300),(0,2,255),5)
#mostrar imagen
plt.imshow(img)
```

[36]:
```
<matplotlib.image.AxesImage at 0x255af108c10>
```



6.0.1 Captura eventos desde el mouse

[1]:
```
import cv2 as cv
clicked = False

#funcion encargad de capturar el click del mouse
def onMouse(evento, x, y,flags, param):
    #variable global
    global clicked

    if(evento == cv.EVENT_LBUTTONUP):
        clicked=True

#creamos objeto cv.VideoCapture()
camCap = cv.VideoCapture(0)
cv.namedWindow("MyWindows")
#configuramos se envie a la funcion los eventos del mouse
cv.setMouseCallback("myWIndow",onMouse)
print("Mostrando estado de la camara, Presione 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()
```

```
Mostrando estado de la camara, Presione en la ventana para detener
```

Programa para dibujar circulos en el estado del mouse.

[3]:
```
import numpy as np
import cv2 as cv
import math
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()
```

7 TAREA

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

[6]:
```
import numpy as np #se importa las librerias
import cv2 as cv
def draw_circleandrectangle(event, x, y, flags, param): #creamos el evento


    if(event == cv.EVENT_LBUTTONDOWN):#condicional que significa si damos click␣
 ↪izquierdo dibuja un circulo

       cv.circle(img, (x, y), 25, (255,200,0), -1 )

    if(event == cv.EVENT_RBUTTONDOWN):#condicional que significa si damos click␣
 ↪derecho dibuja un rectangulo
       cv.rectangle(img, (x+50,y+50), (x,y),(0,255,0),-1)


img = np.zeros((512,512,3), np.uint8)# crea una ventana de 512x512
```

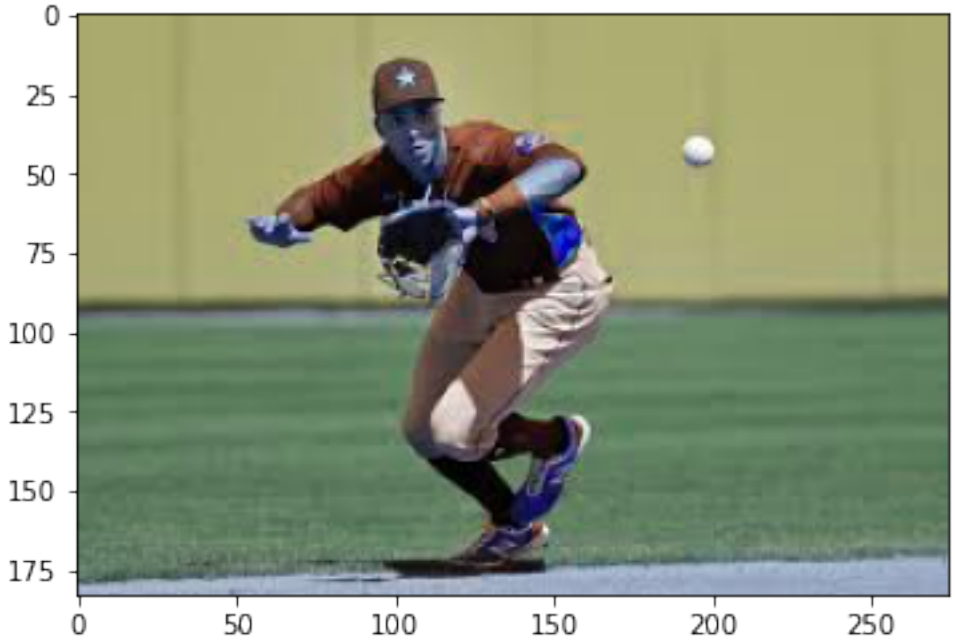
```
cv.namedWindow('Dibujando Circulos y Rectangulos') #identifica el nombre de la␣
 ↪ventana
cv.setMouseCallback('Dibujando Circulos y Rectangulos',␣
 ↪draw_circleandrectangle)#llama al evento en la ventana creada anteriormente

while(True): #condicional de que se muestre la ventana hasta que presionemos la␣
 ↪letra x

    cv.imshow('Dibujando Circulos y Rectangulos',img)
    if (cv.waitKey(1) & 0xFF ==ord('x')):
        break

cv.destroyAllWindows()
```

8 Operaciones con Imagenes

Acceso y Modificacion depixeles

[8]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/img2.jpg")
plt.imshow(img)
```

[8]:
```
<matplotlib.image.AxesImage at 0x1fbd82cb0d0>
```



[9]:
```
#Para acceder al valor del pixel en sus 3 canaes
px=img[100,100]
px
```

[9]:
```
array([106, 131, 105], dtype=uint8)
```

[10]:
```
#accediendo al valor de un pixel en un solo canal B

px=img[100,100,0]
px
```

[10]:
```
106
```

[11]:
```
#accediendo al valor de un pixel en un solo canal G

px=img[100,100,1]
px
```

[11]:
```
131
```

[12]:
```
#accediendo al valor de un pixel en un solo canal R

px=img[100,100,2]
px
```

[12]:
```
105
```

9 Modificacion

[13]:
```
#Modificando el valo de pixeles
img[100,100]=[255,255,255]
img[100,100]
```

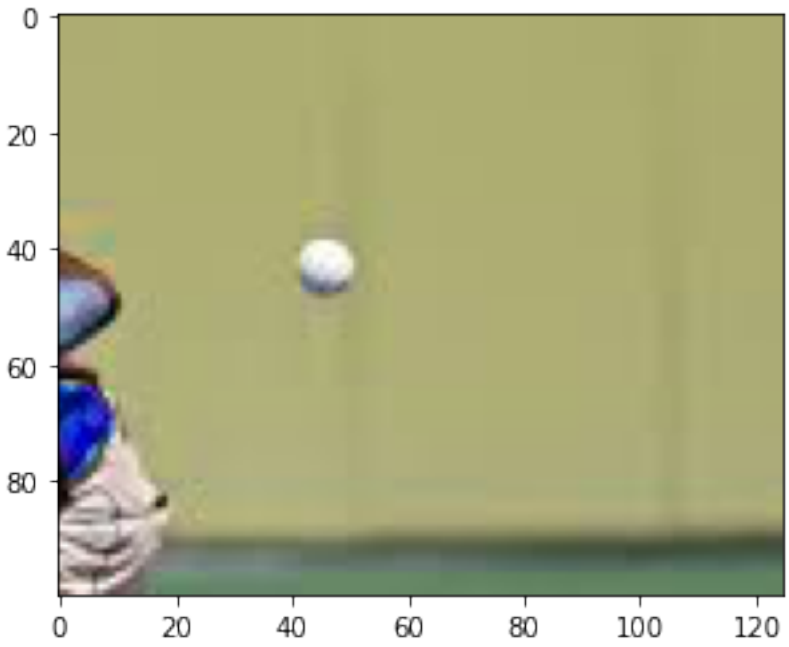
[13]:
```
array([255, 255, 255], dtype=uint8)
```

10 NUMPY

Libreria optimizada para calculos rapidos de matrices.

[15]:
```
#extraer parte de la imagen
img2=img[:100, 150:500]
plt.imshow(img2)
```

[15]:
```
<matplotlib.image.AxesImage at 0x1fbd83bdb80>
```



[16]:
```
#accediendo
img.item(50,50,2)
```

[16]: 117

[19]:
```
#modificando
img.itemset((50,50,2),100)
```

[20]:
```
img.item(50,50,2)
```

[20]: 100

Acediendo a las propiedades de la Imagen Las imagenes poseen propiedades como numero de fulas, numero de columnas, numero de pixeles, typo de dato.

Metodo Shape

[21]:
```
img.shape
```

[21]: (183, 275, 3)

Metodo dtyper

[23]:
```
img.dtype
```

[23]: dtype('uint8')

Size

[24]:
```
#tamaño
img.size
```

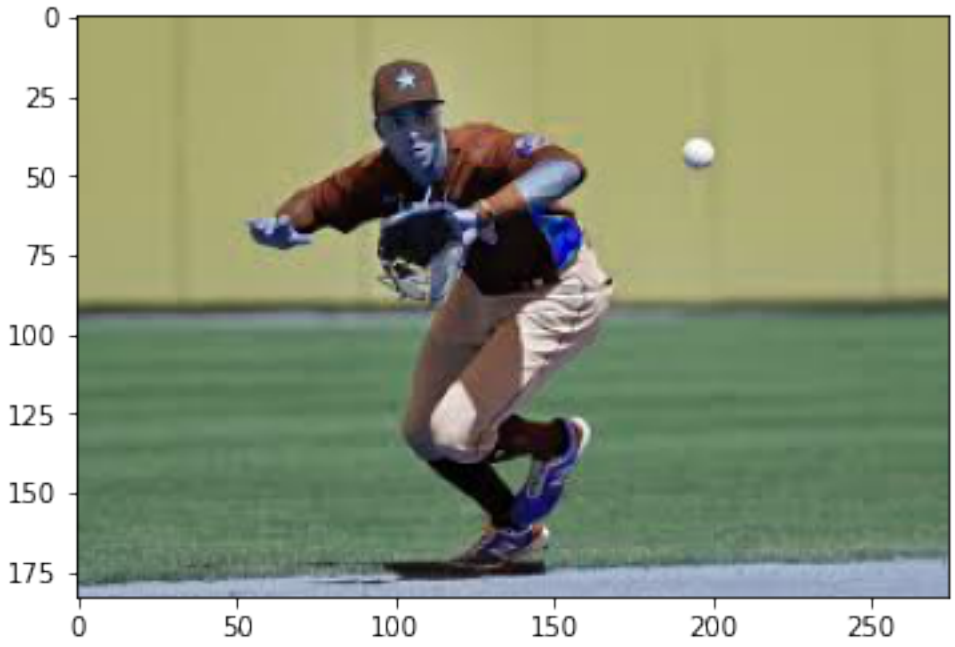
[24]:
```
150975
```

11 ROI DE UNA IMAGEN

SELECCIONAR PARTE DE LA IMAGEN Y COPIARLA

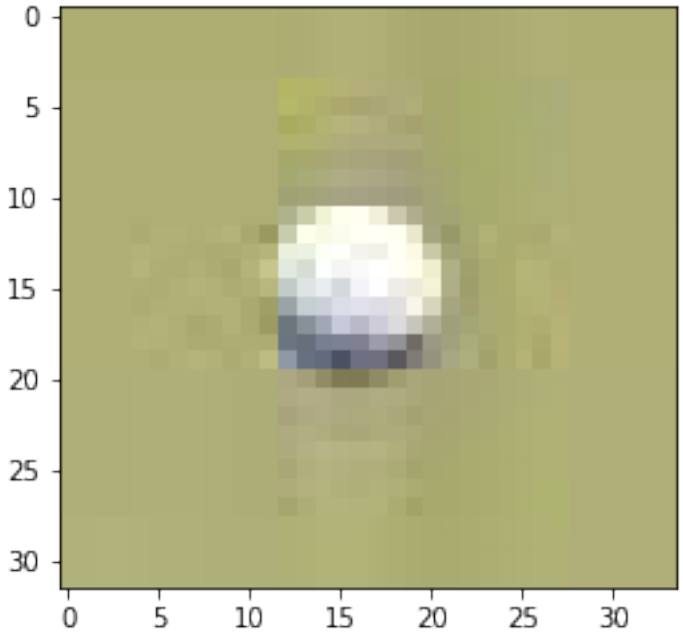
[83]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img=cv.imread("Imgs\img2.jpg")
plt.imshow(img)
```

[83]:
```
<matplotlib.image.AxesImage at 0x1fbd8ae22b0>
```



[84]:
```
#extraer la re1ion de interes
pelota = img [28:60,180:214]
plt.imshow(pelota)
```

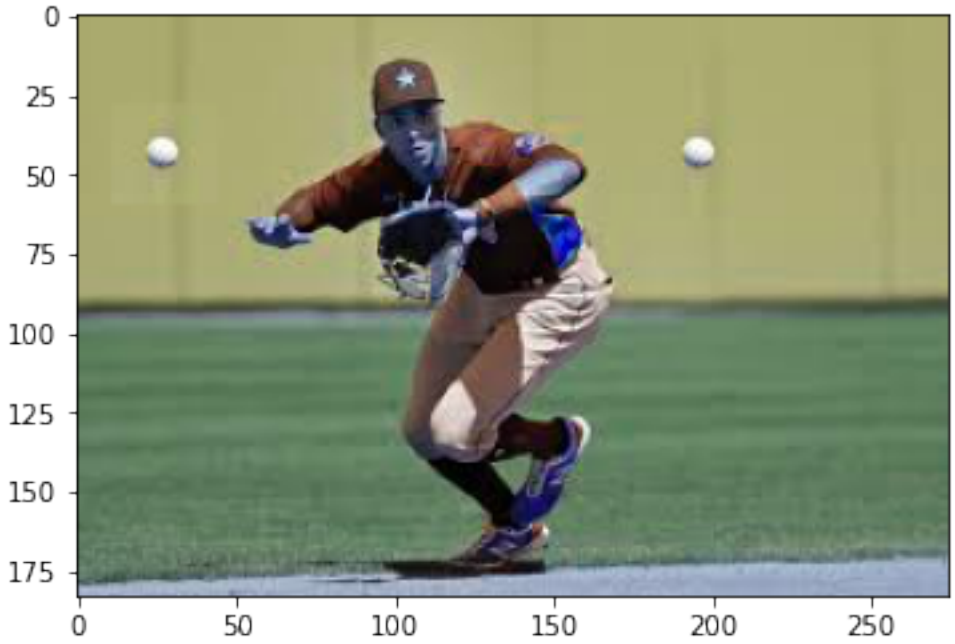
[84]:
```
<matplotlib.image.AxesImage at 0x1fbd8b3a310>
```



Copiamos el roi de la imagen.

[85]:
```
img[28:60,11:45] = pelota

plt.imshow(img)
```

[85]:
```
<matplotlib.image.AxesImage at 0x1fbd9b625b0>
```



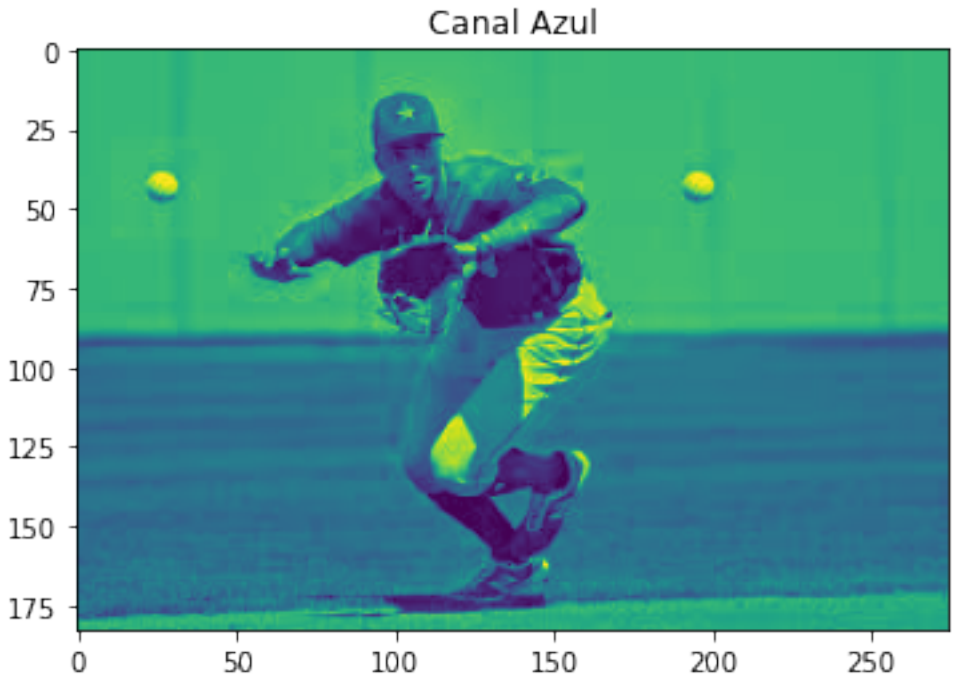
DIVISION Y FUSION DE CANALES

split()

[89]:
```
#nos permite dividir los canales de una imagen
b,g,r = cv.split(img)
```

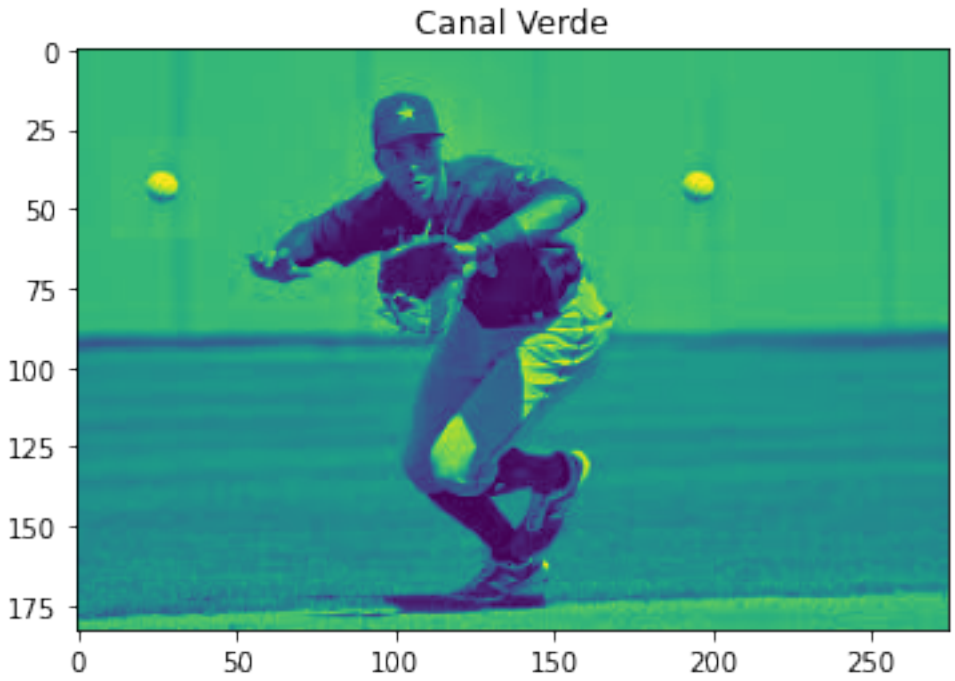
[90]:
```
plt.imshow(b)
plt.title("Canal Azul")
```

[90]: Text(0.5, 1.0, 'Canal Azul')



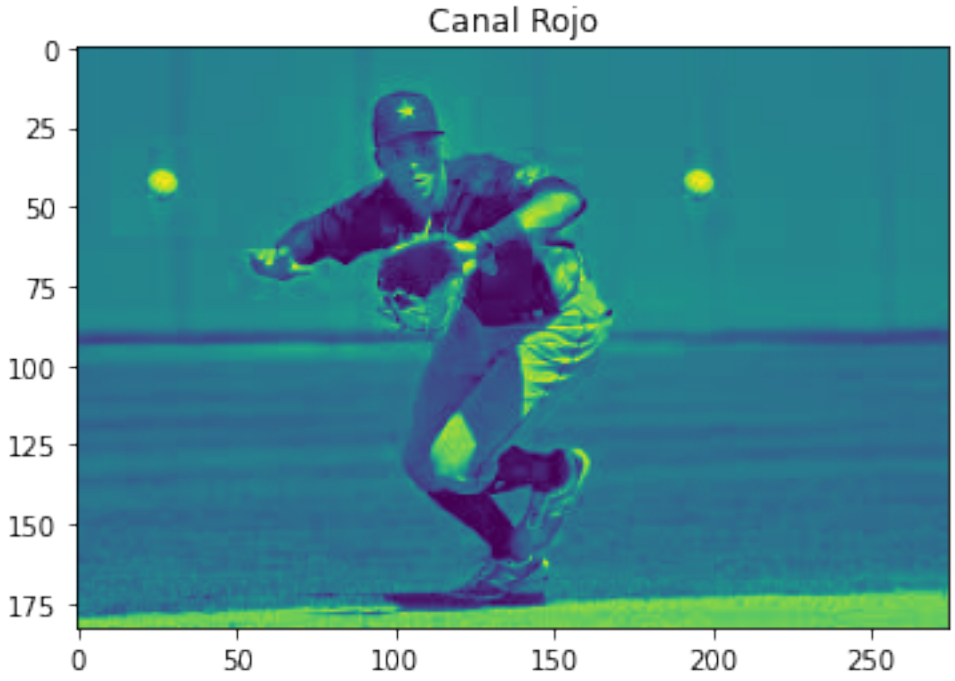
[91]:
```
plt.imshow(g)
plt.title("Canal Verde")
```

[91]: Text(0.5, 1.0, 'Canal Verde')



[92]:
```
plt.imshow(r)
plt.title("Canal Rojo")
```

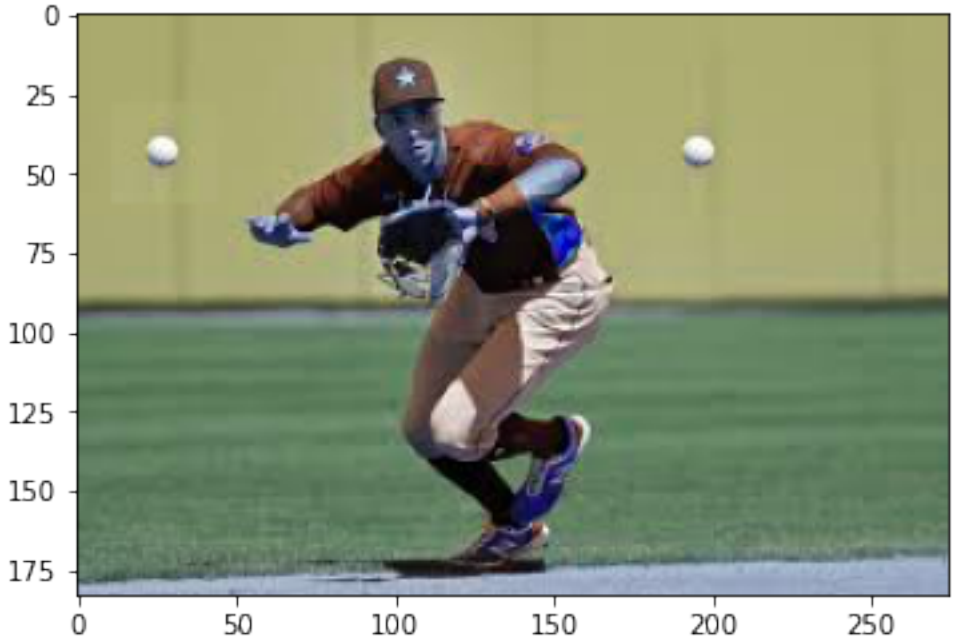
[92]: Text(0.5, 1.0, 'Canal Rojo')



merge() permite unir los canales de una imagen

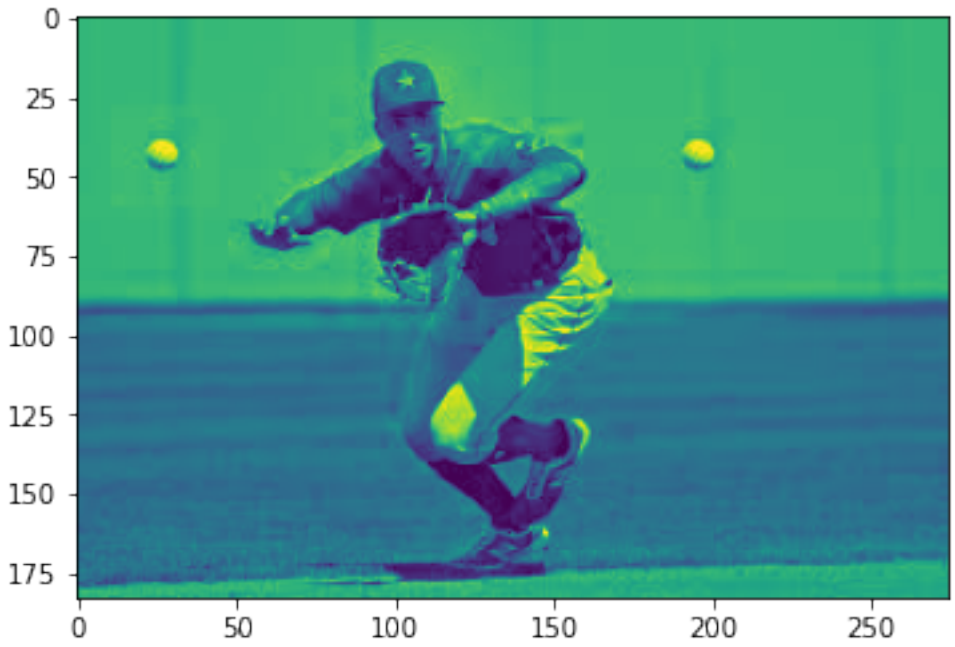
[93]:
```
img_unida = cv.merge((b,g,r))
plt.imshow(img_unida)
```

[93]: <matplotlib.image.AxesImage at 0x1fbd86d7790>



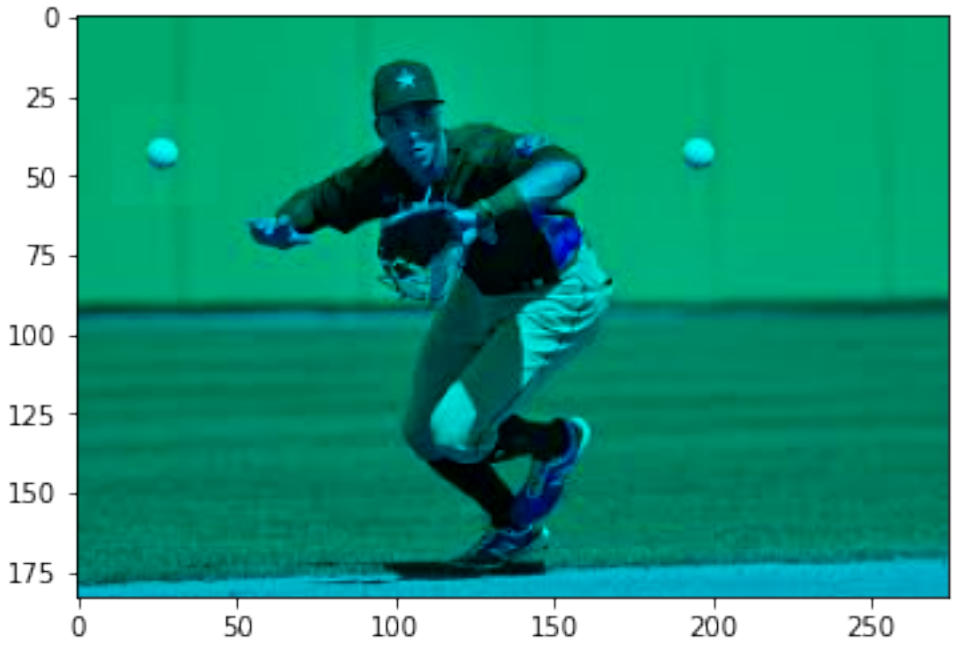
[94]:
```
channel_blue = img[:,:,0]
plt.imshow(channel_blue)
```

[94]: <matplotlib.image.AxesImage at 0x1fbda54a190>



[96]:
```
 img[:,:,0]=0
plt.imshow(img)
```

[96]: <matplotlib.image.AxesImage at 0x1fbda24e610>



12 Operaciones Aritmeticas con Imagenes

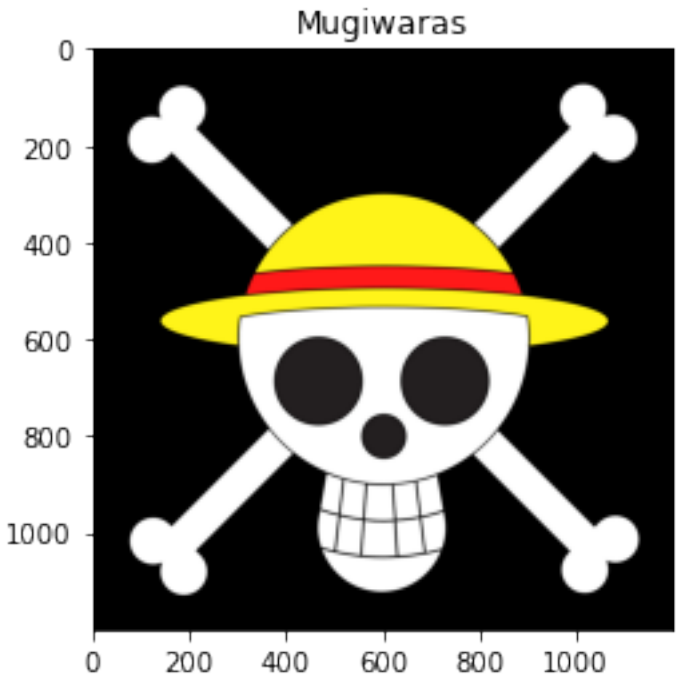
Mescla de Imagenes

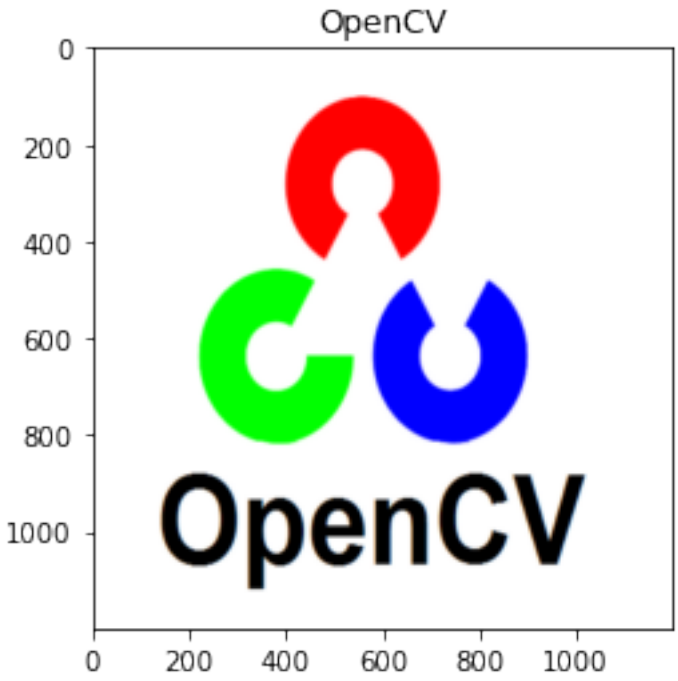
Eso tambien es una adicion de imagen m

[101]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img1 = cv.imread("Imgs/mugi.png")
img2 = cv.imread("Imgs/img3.png")
#cambio de espacio de color
img1=cv.cvtColor(img1, cv.COLOR_BGR2RGB)
img2=cv.cvtColor(img2, cv.COLOR_BGR2RGB)
#IGUALAR EL TAMANO
fil,cols,chan=img1.shape
img2=cv.resize(img2,(cols,fil))
```

[102]:
```
#mostramos las imagenes
plt.figure(1)
plt.imshow(img1)
plt.title("Mugiwaras")

plt.figure(2)
plt.imshow(img2)
plt.title("OpenCV")
```

[102]:
```
Text(0.5, 1.0, 'OpenCV')
```

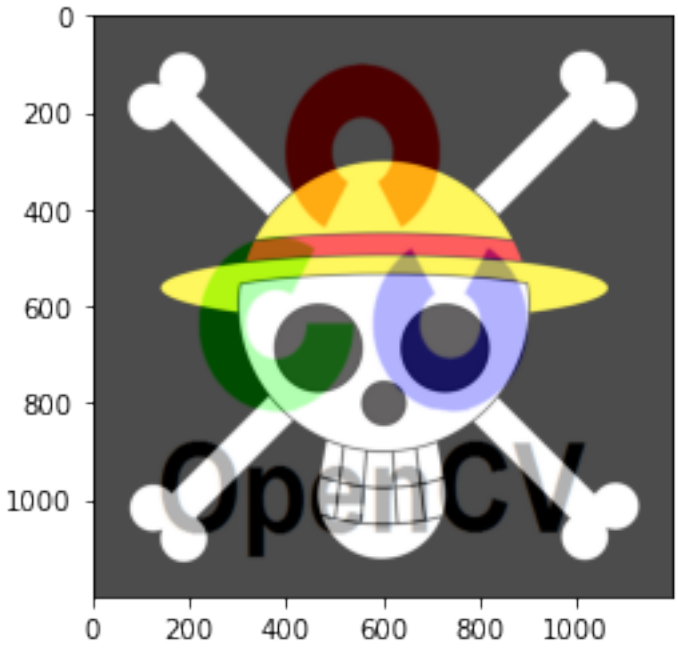




[103]:
```
#combinar imagenes
img_out = cv.addWeighted(img1,0.7,img2, 0.3,0)
```

```
plt.imshow(img_out)
```

[103]: <matplotlib.image.AxesImage at 0x1fbda08e4c0>

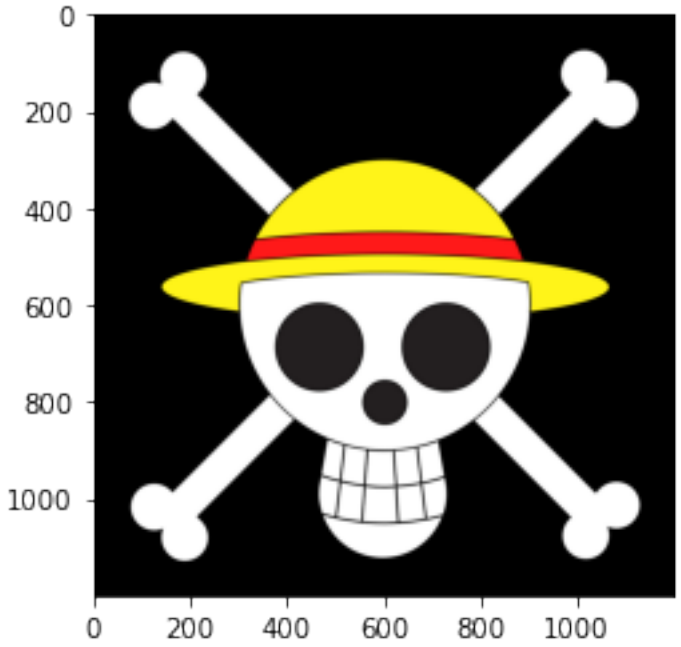


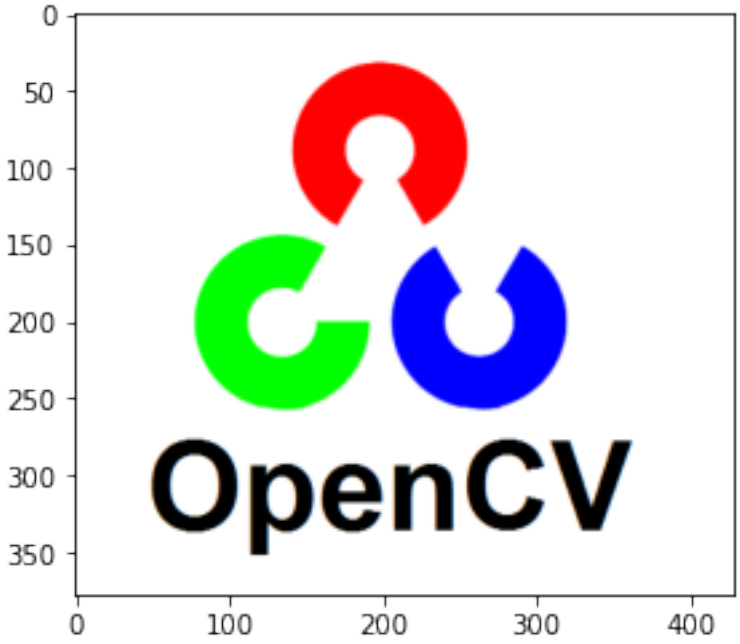
13 Operaciones bit a bit

Operaciones and , or, not y xor

[111]:
```
#immportamos las librerias
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
# cargar nuestras imagenes
img1 = cv.imread('Imgs/mugi.png')
img2 = cv.imread('Imgs/img3.png')
# cambiando espacio de color
mugi = cv.cvtColor(img1, cv.COLOR_BGR2RGB)
cv2 = cv.cvtColor(img2, cv.COLOR_BGR2RGB)
# mostrado las imagenes leidas
plt.figure(1)
plt.imshow(mugi)
plt.figure(2)
plt.imshow(cv2)
```

[111]: <matplotlib.image.AxesImage at 0x1fbda949dc0>

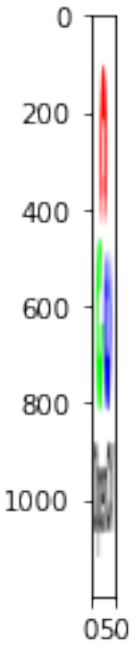




[110]:
```
#cambiamos el tamano de la imagen
fil,col,_=mugi.shape
fil2,col2,_=cv2.shape
cv2=cv.resize(cv2,(col2//2,fil))
```

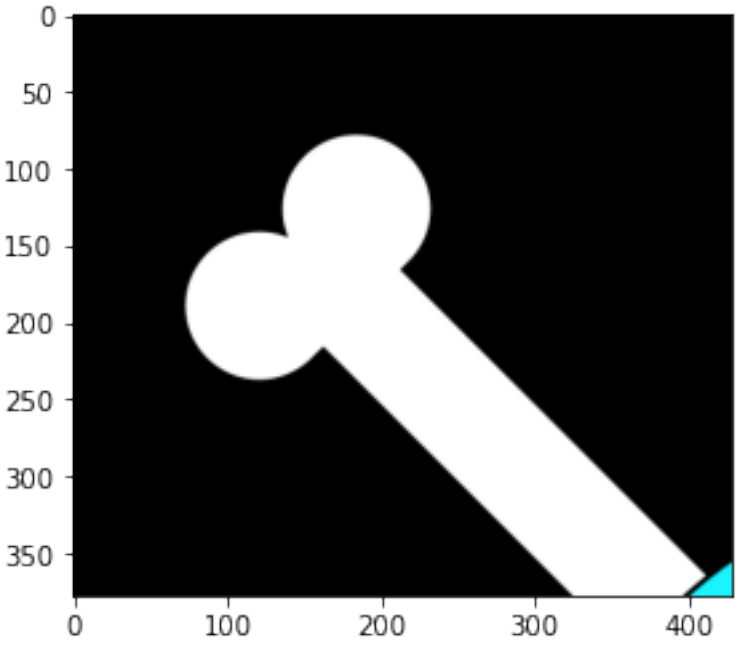
```
plt.imshow(cv2)
```

[110]: <matplotlib.image.AxesImage at 0x1fbda895880>



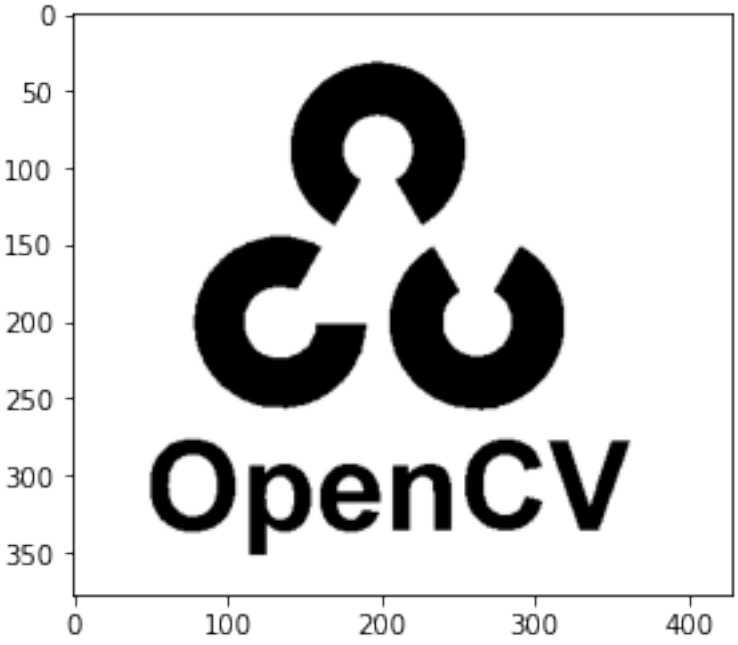
[112]:
```
fil, col,_ = cv2.shape
roi=img1[0:fil,0:col]
plt.imshow(roi)
```

[112]: <matplotlib.image.AxesImage at 0x1fbda8a4f10>



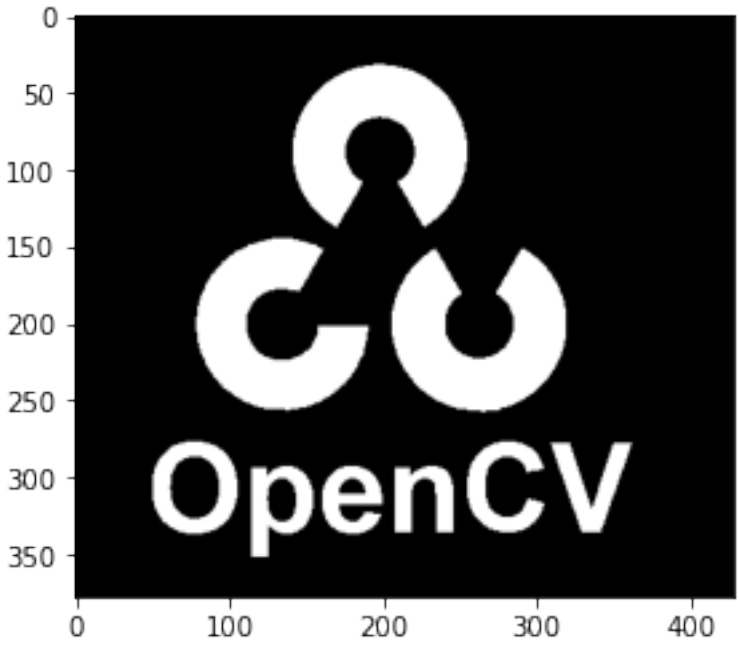
[113]:
```
# creamos una mascara del logotipo
cv2_gray=cv.cvtColor(cv2, cv.COLOR_RGB2GRAY)
ret,mask = cv.threshold(cv2_gray,150,255, cv.THRESH_BINARY)
plt.imshow(mask, cmap="gray")
```

[113]: <matplotlib.image.AxesImage at 0x1fbd9c98d30>



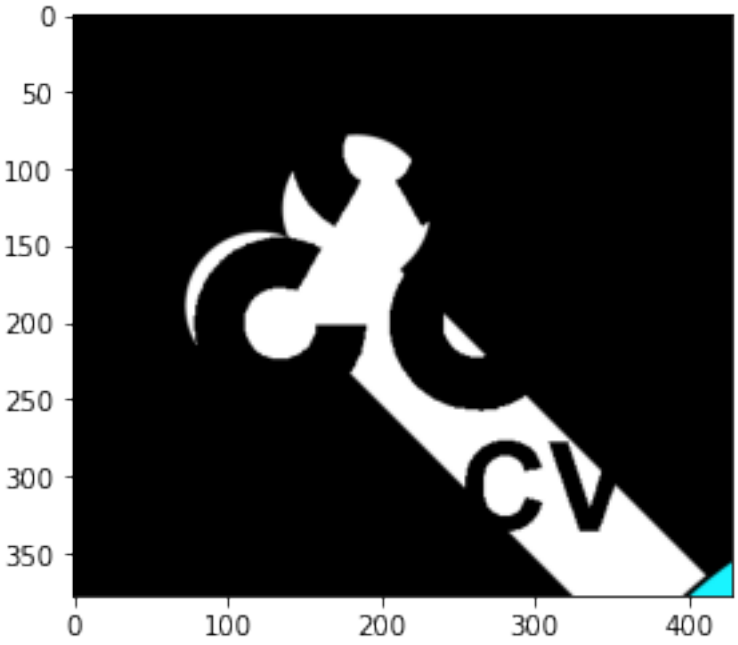
[114]:
```
#Creamos una mascara invertida
mask_inv = cv.bitwise_not(mask)
plt.imshow(mask_inv, cmap="gray")
```

[114]: <matplotlib.image.AxesImage at 0x1fbd9e85100>



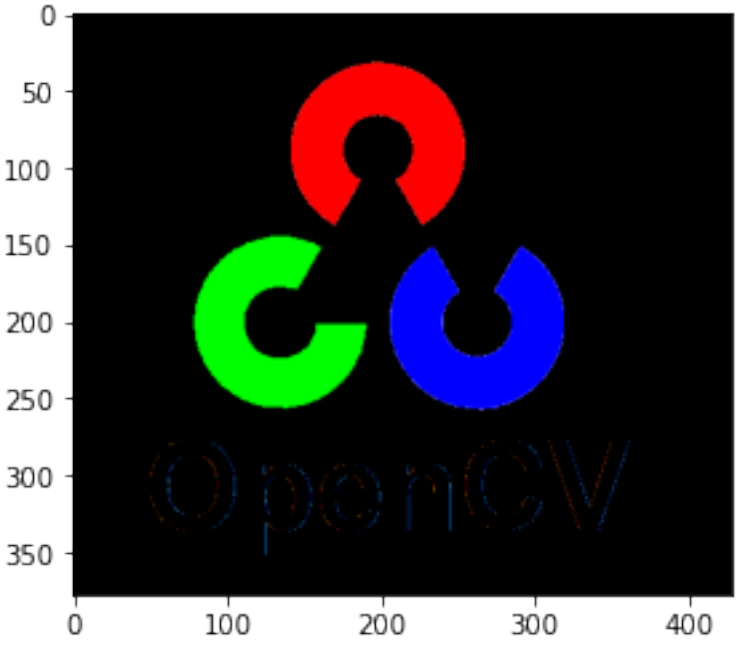
[115]:
```
#tomamos el rol menos la mascara
img1_bg = cv.bitwise_and(roi,roi,mask=mask)
plt.imshow(img1_bg)
```

[115]: <matplotlib.image.AxesImage at 0x1fbda15a250>



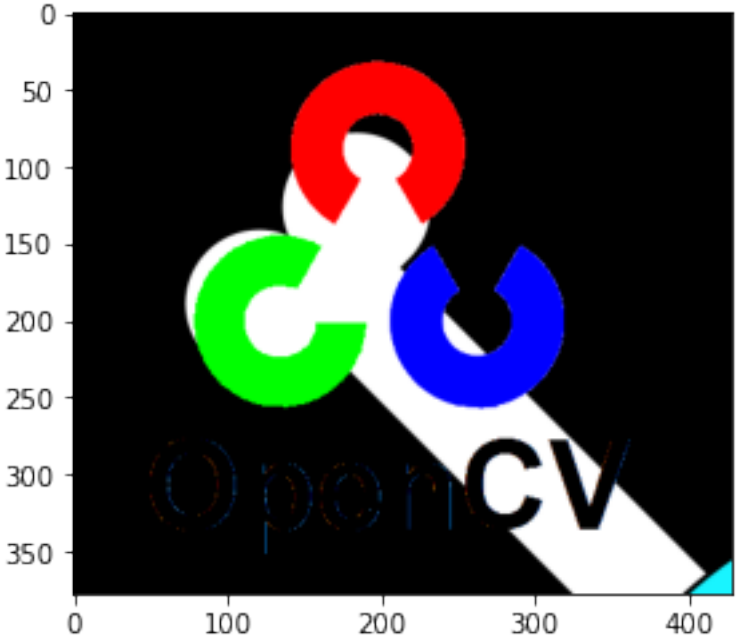
[117]:
```
#tomamos region de interes del logotipo opencv
img2_bg = cv.bitwise_and(cv2,cv2,mask=mask_inv)
plt.imshow(img2_bg)
```

[117]: <matplotlib.image.AxesImage at 0x1fbda718310>



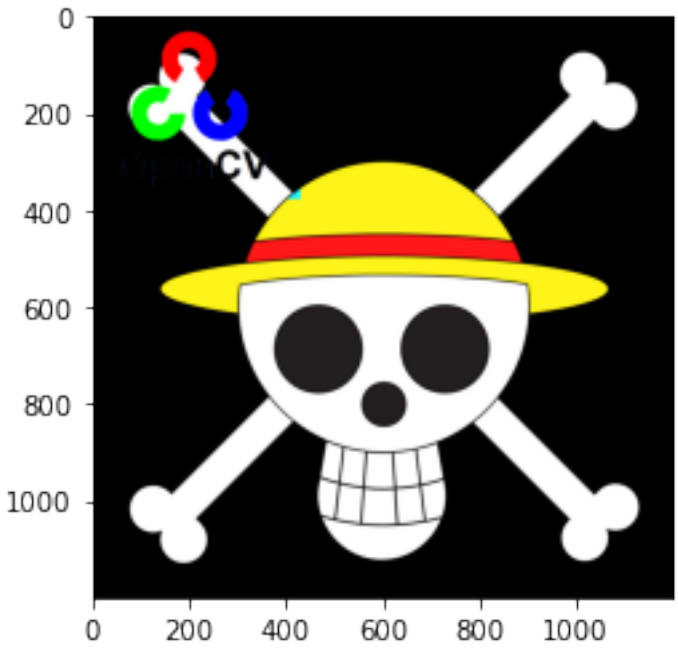
[118]:
```
img = img1_bg + img2_bg
plt.imshow(img)
```

[118]: <matplotlib.image.AxesImage at 0x1fbda9a8280>



[119]:
```
mugi[0:fil,0:col]=img
plt.imshow(mugi)
```

[119]: <matplotlib.image.AxesImage at 0x1fbda3934f0>

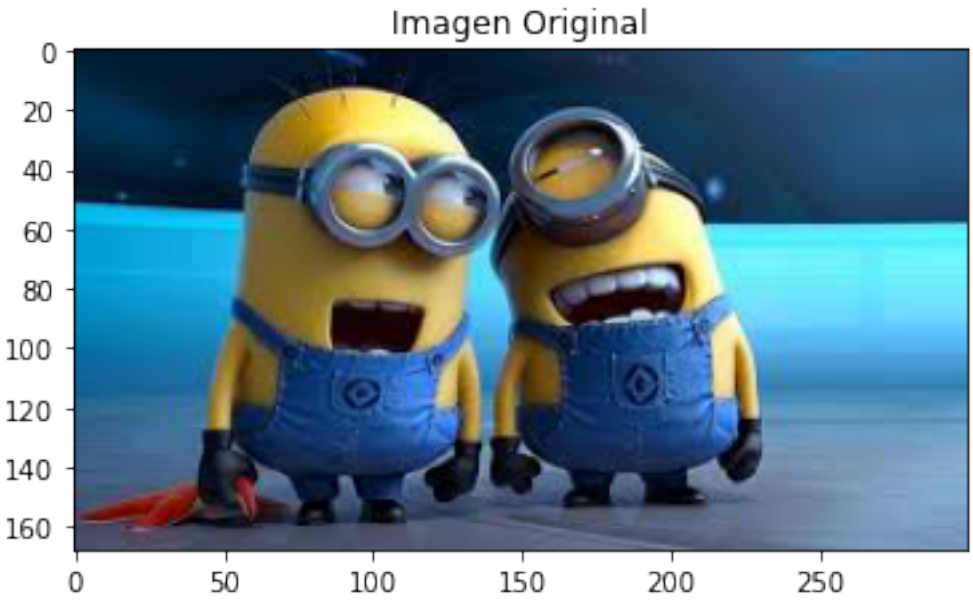


13.0.1 Procesamiento de imagenes

Cambio de Espacio de Color de las Imagenes

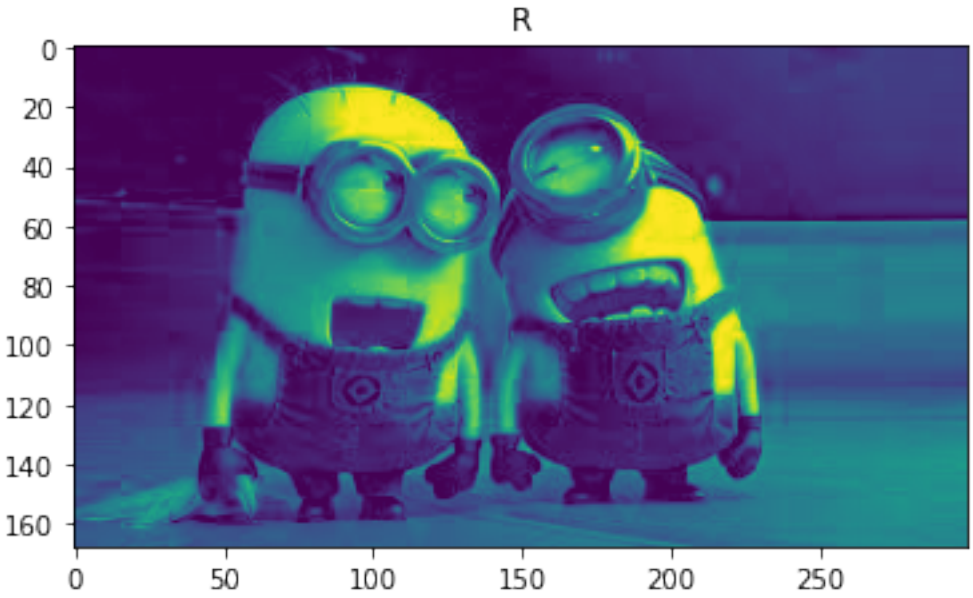
[120]:
```
#Cambio de espacio de color de las imagenes
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img=cv.imread("Imgs\img1.JPG")
img=cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img),plt.title("Imagen Original")
```

[120]:
```
(<matplotlib.image.AxesImage at 0x1fbda3c9610>,
 Text(0.5, 1.0, 'Imagen Original'))
```



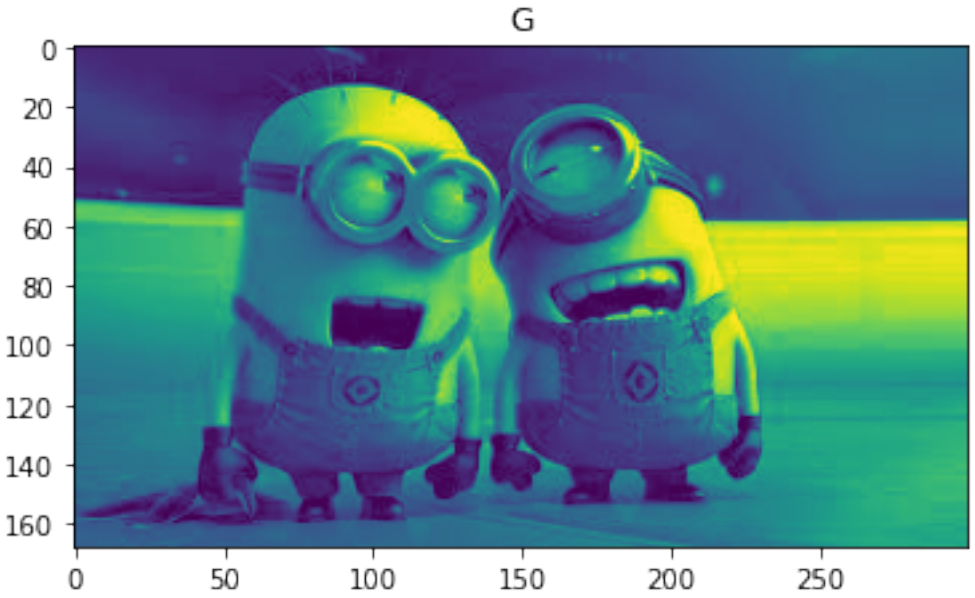
[121]:
```
plt.imshow(img[:,:,0]),plt.title("R")
```

[121]:
```
(<matplotlib.image.AxesImage at 0x1fbda409280>, Text(0.5, 1.0, 'R'))
```



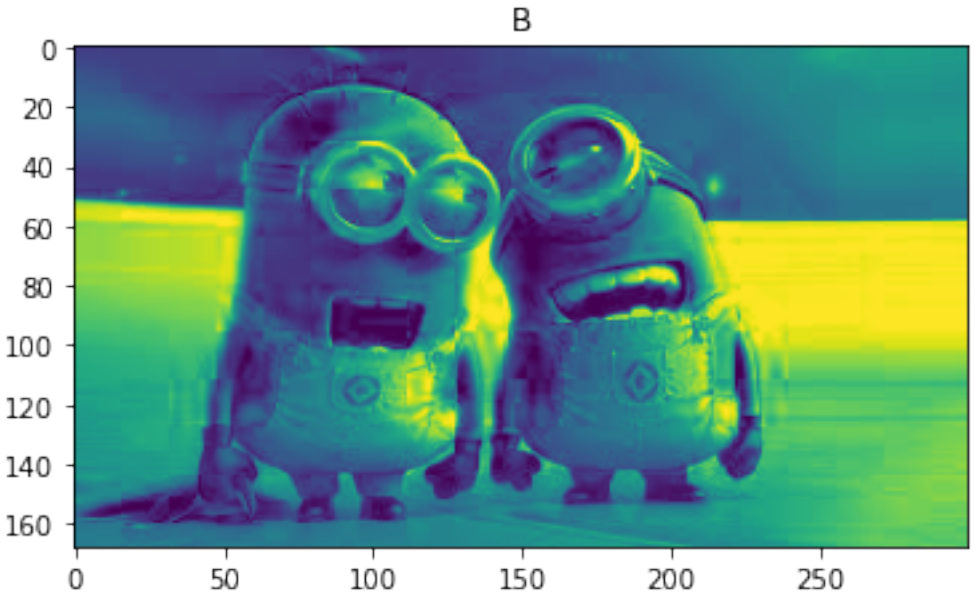
[122]:
```
plt.imshow(img[:,:,1]),plt.title("G")
```

[122]: (<matplotlib.image.AxesImage at 0x1fbda43cbe0>, Text(0.5, 1.0, 'G'))



[124]:
```
plt.imshow(img[:,:,2]),plt.title("B")
```

[124]: (<matplotlib.image.AxesImage at 0x1fbda8c1b20>, Text(0.5, 1.0, 'B'))

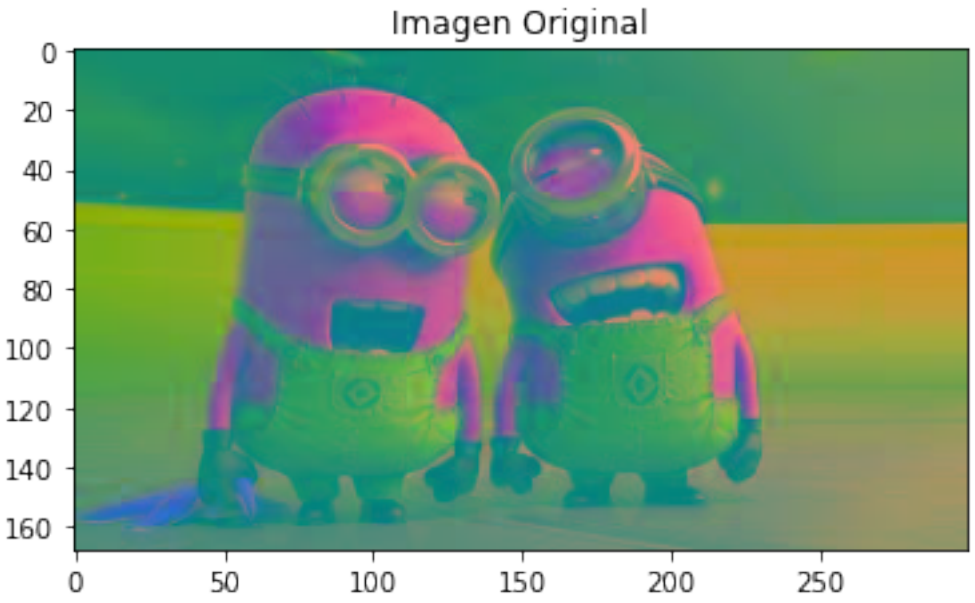


14 YUV

La gente comenzó a pensar en diferentes métodos para separar la información de intensidad, de la información de color. Por lo tanto, se les ocurrió el espacio de color YUV. Y se refiere a la luminancia o intensidad, y los canales U / V representan información de color. Esto funciona bien en muchas aplicaciones porque el sistema visual humano percibe la información de intensidad de manera muy diferente a la información de color.

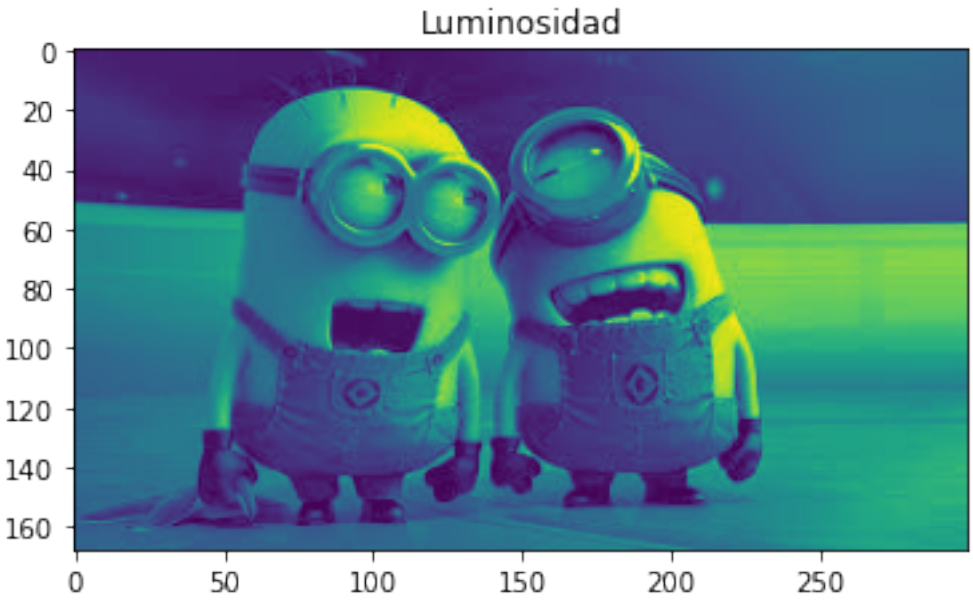
[125]:
```
# importamos librerias
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img = cv.imread("Imgs/img1.jpg")
img = cv.cvtColor(img, cv.COLOR_BGR2YUV)
plt.imshow(img),plt.title("Imagen Original")
```

[125]:
```
(<matplotlib.image.AxesImage at 0x1fbd89f5ca0>,
 Text(0.5, 1.0, 'Imagen Original'))
```



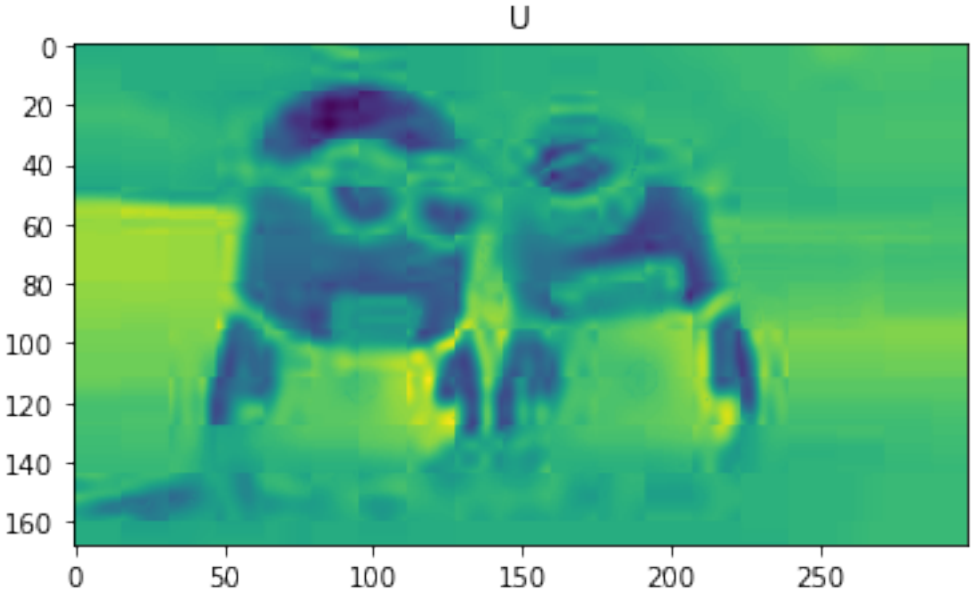
[126]:
```
plt.imshow(img[:,:,0]),plt.title("Luminosidad")
```

[126]:
```
(<matplotlib.image.AxesImage at 0x1fbdaa1c250>, Text(0.5, 1.0, 'Luminosidad'))
```



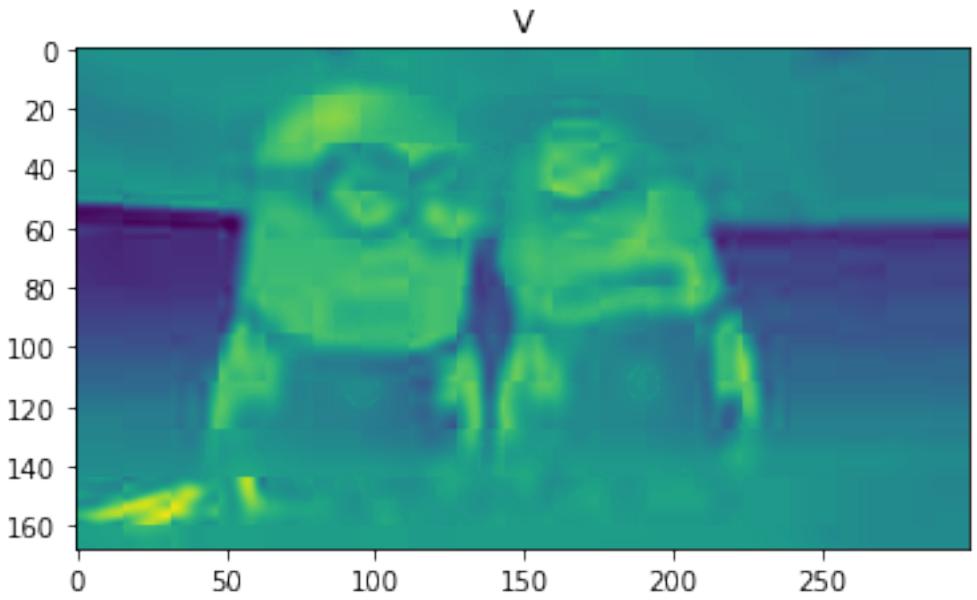
[127]:
```
plt.imshow(img[:,:,1]),plt.title("U")
```

[127]:
```
(<matplotlib.image.AxesImage at 0x1fbdaa5ab20>, Text(0.5, 1.0, 'U'))
```



[128]:
```
plt.imshow(img[:,:,2]),plt.title("V")
```

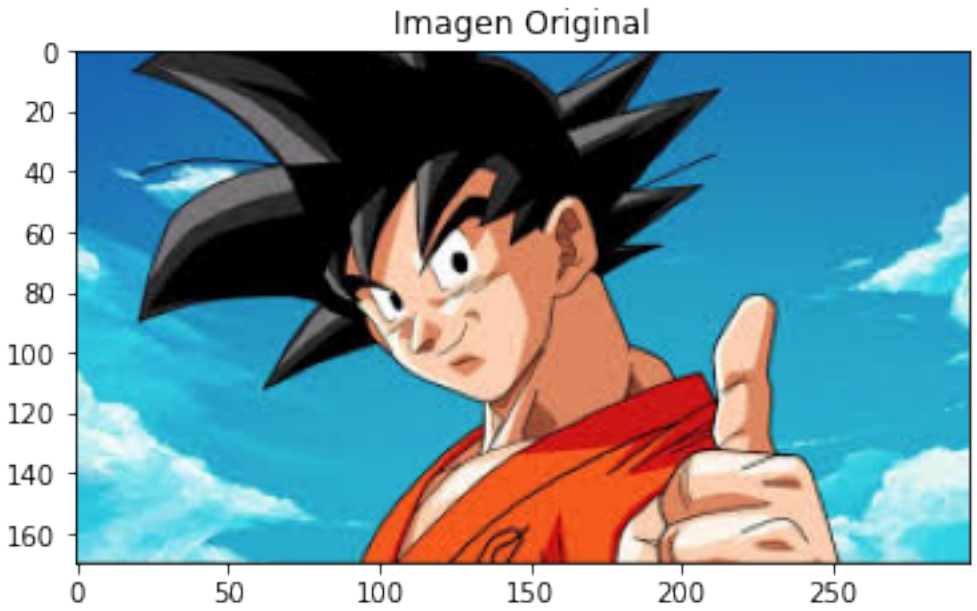
[128]: (<matplotlib.image.AxesImage at 0x1fbdaabf2b0>, Text(0.5, 1.0, 'V'))



HSV: Matiz, saturacion, valor de los colores

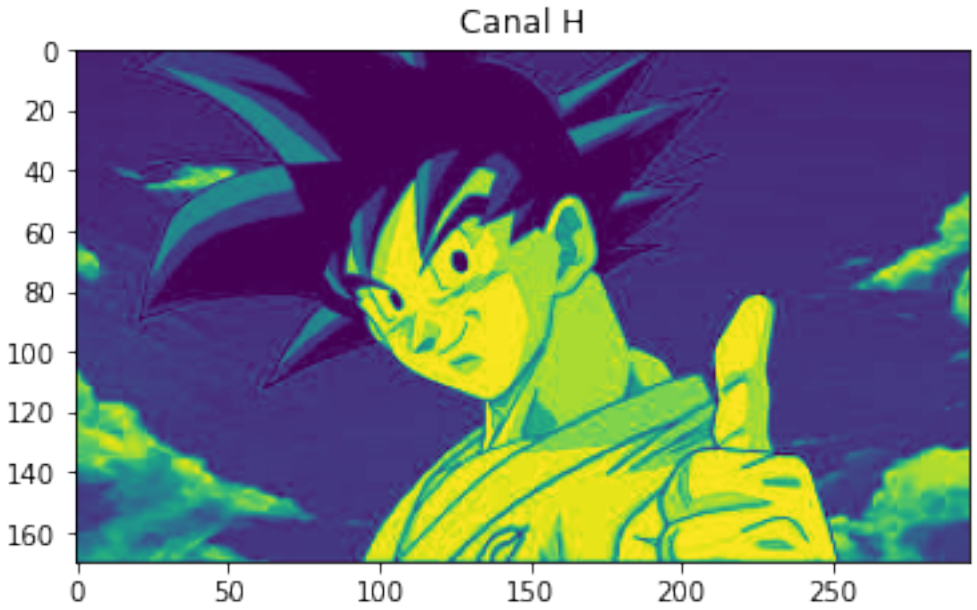
[135]:
```
#Cambio de espacio de color de las imagenes
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img=cv.imread("Imgs\goku.JPG")
img=cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img),plt.title("Imagen Original")
```

[135]: (<matplotlib.image.AxesImage at 0x1fbdac380d0>,
Text(0.5, 1.0, 'Imagen Original'))



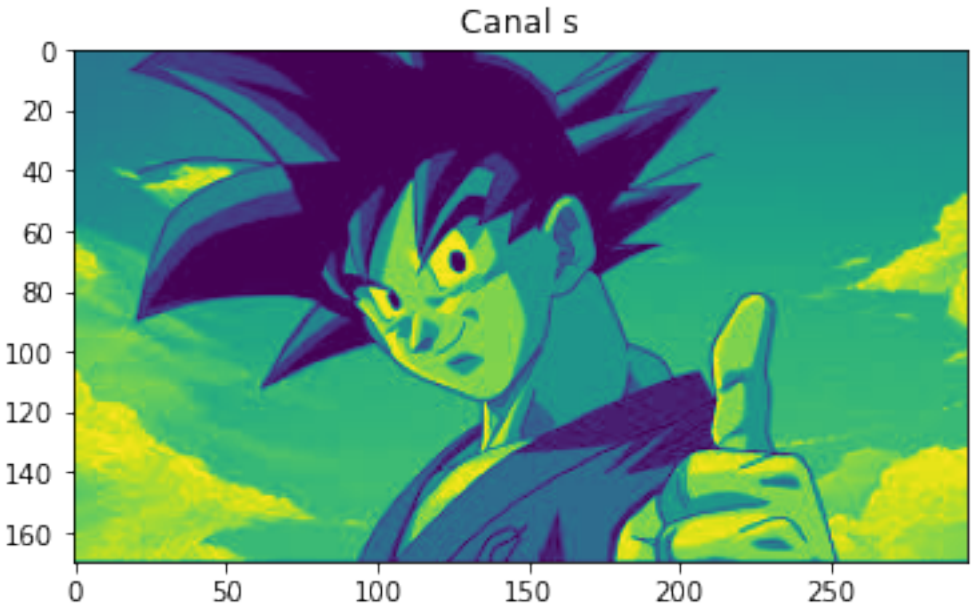
[136]:
```
plt.imshow(img[:,:,0]),plt.title("Canal H")
```

[136]:
```
(<matplotlib.image.AxesImage at 0x1fbdac906a0>, Text(0.5, 1.0, 'Canal H'))
```



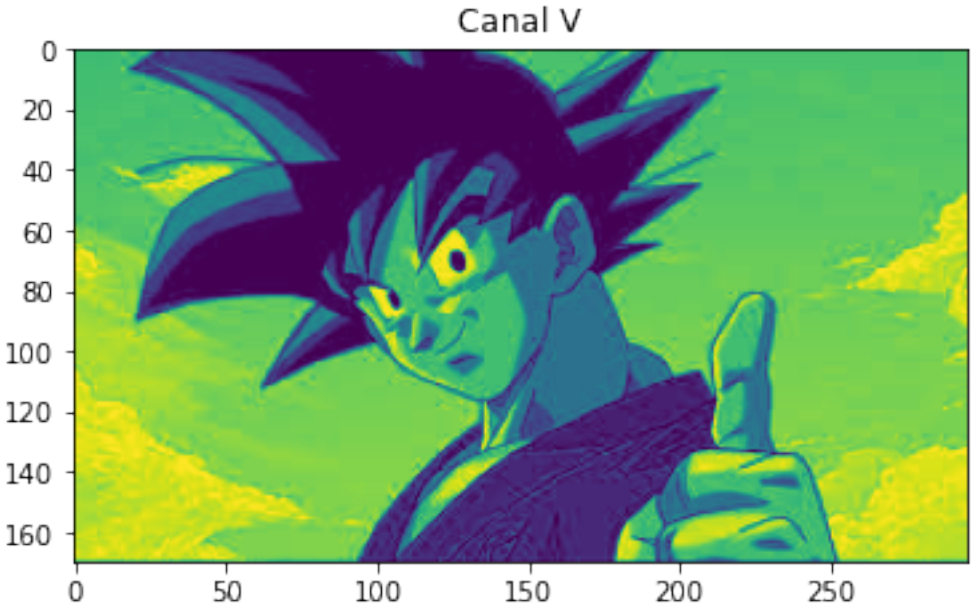
[137]:
```
plt.imshow(img[:,:,1]),plt.title("Canal s")
```

[137]: (<matplotlib.image.AxesImage at 0x1fbdace8b80>, Text(0.5, 1.0, 'Canal s'))



[138]:
```
plt.imshow(img[:,:,2]),plt.title("Canal V")
```

[138]: (<matplotlib.image.AxesImage at 0x1fbdad4b2e0>, Text(0.5, 1.0, 'Canal V'))



15 DETECCION DE COLORES

[1]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

cap= cv.VideoCapture(0)

#HSV azul
azul_bajo = np.array([90,150,55], np.uint8)
azul_alto = np.array([125,255,255], np.uint8)

while cap.isOpened():

    ret, frame = cap.read()

    if ret:

        img_hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

        mask = cv.inRange(img_hsv,azul_bajo,azul_alto)
        res = cv.bitwise_and(frame, frame, mask=mask)
        cv.imshow("IMAGEN ORIGINAL", frame)
        cv.imshow("Mascara", mask)
        cv.imshow("Resultado", res)

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

cap.release()
cv.destroyAllWindows()
```

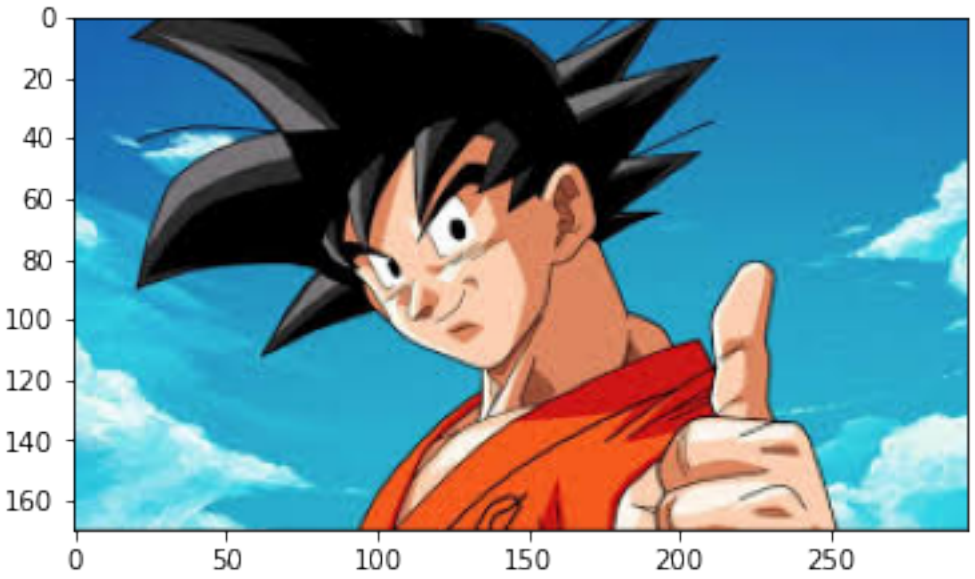
15.0.1 TRANSFORMACIONES GEOMETRICAS

15.0.2 Escalado

Solo trata de cambiar el tamano de una imagen resize se especifica manualemnte o por escala o interpolacion llamados intercubic, interlineal etc

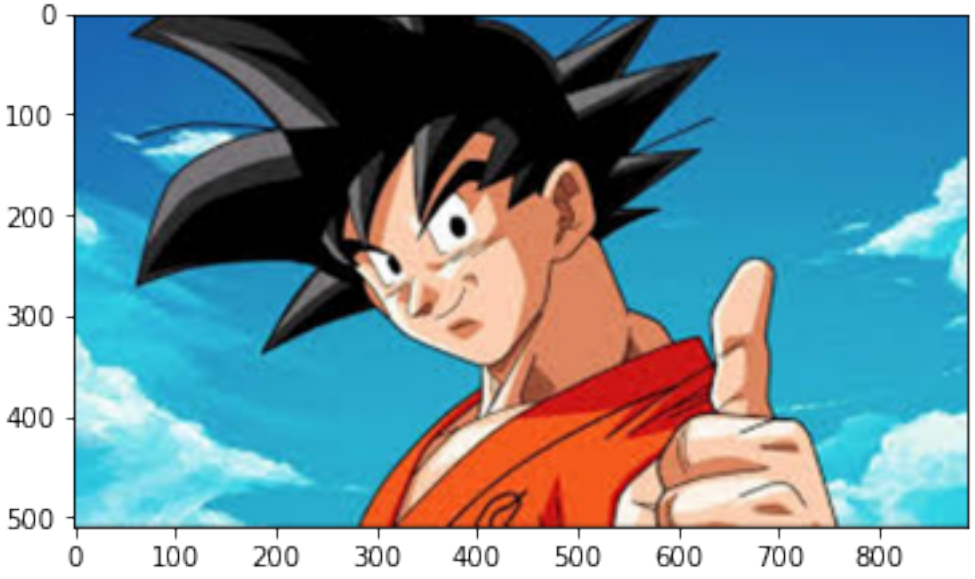
[4]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/goku.jpg")
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

[4]: <matplotlib.image.AxesImage at 0x23341758e80>



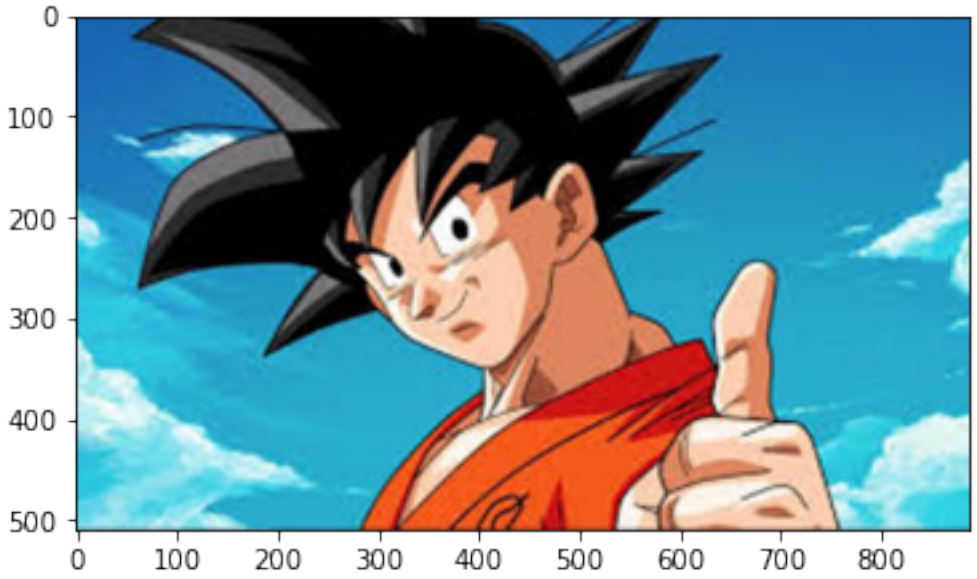
[7]:
```
# escalado
img_escal = cv.resize(img, None, fx=3, fy=3, interpolation= cv.INTER_CUBIC)

plt.imshow(img_escal)
```

[7]: <matplotlib.image.AxesImage at 0x233424a4310>



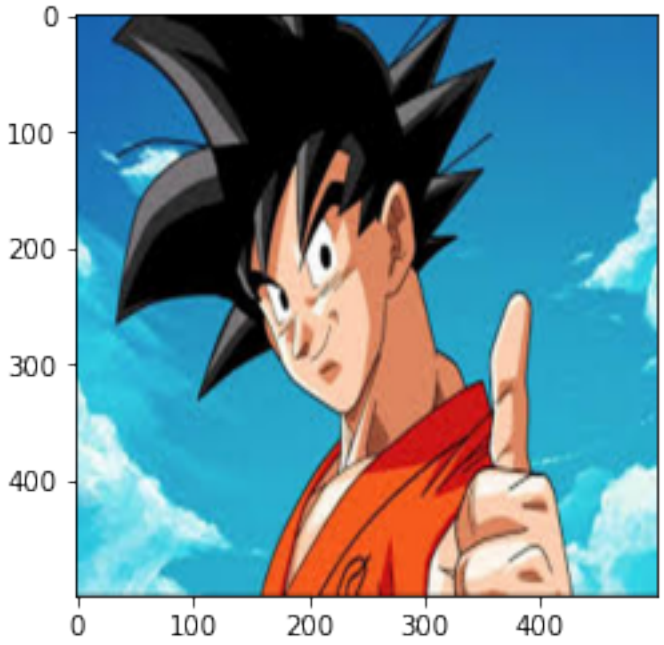
[11]:
```
h,w,_ = img.shape
result = cv.resize(img,(w*3,h*3), interpolation = cv.INTER_CUBIC)
plt.imshow(result)
```

[11]: <matplotlib.image.AxesImage at 0x233425157f0>



[12]:
```
h,w,_ = img.shape
result = cv.resize(img,(500,500), interpolation = cv.INTER_CUBIC)
plt.imshow(result)
```

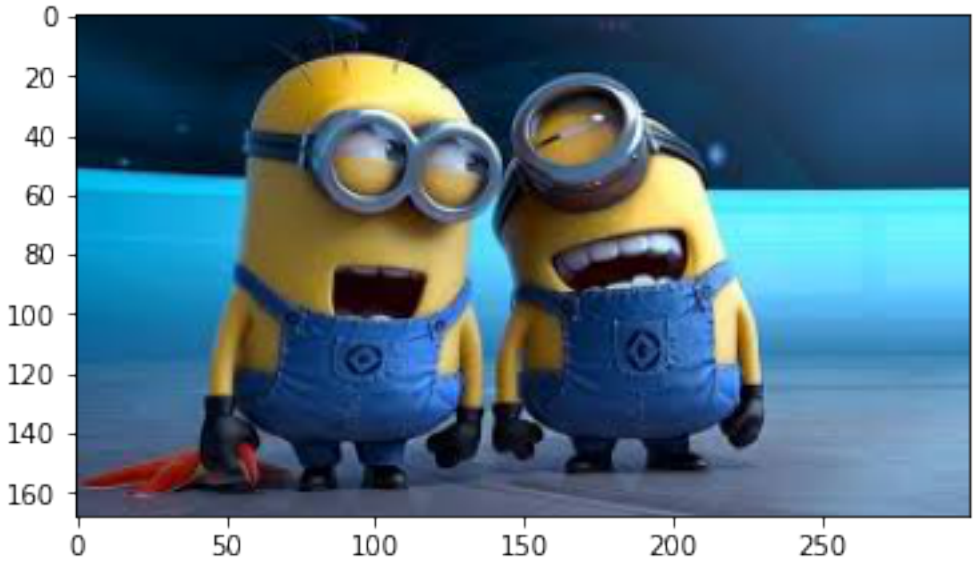
[12]: <matplotlib.image.AxesImage at 0x23341d79df0>



16 Rotacion

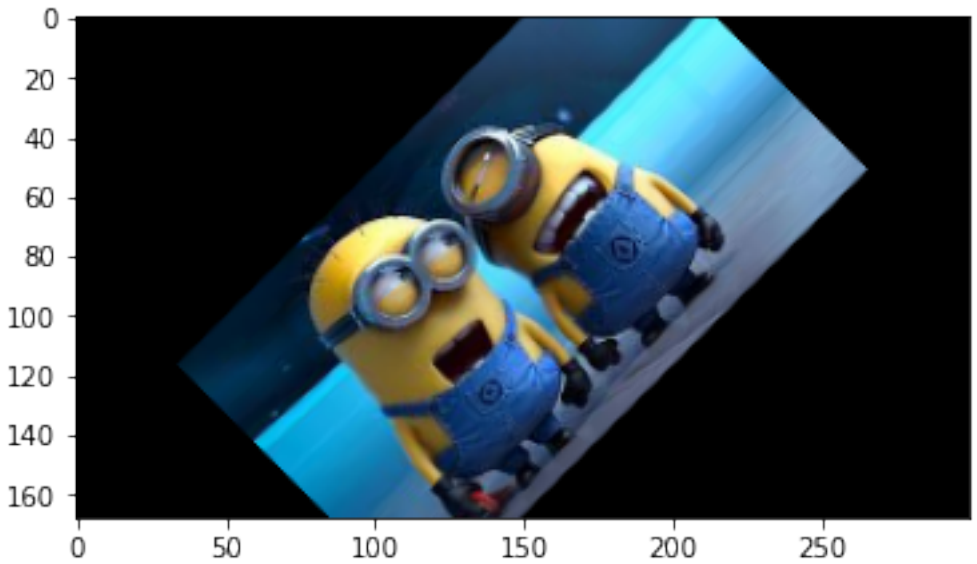
[13]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/img1.jpg")
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

[13]:
```
<matplotlib.image.AxesImage at 0x23341cd8b50>
```



[19]:
```
#girar la imagen 90`
fil,col,_ = img.shape
m = cv.getRotationMatrix2D(((col-1)/2.0,(fil-1)/2.0),45,0.7)
#transformacion de la imagen
result = cv.warpAffine(img, m, (col,fil) )
plt.imshow(result)
```

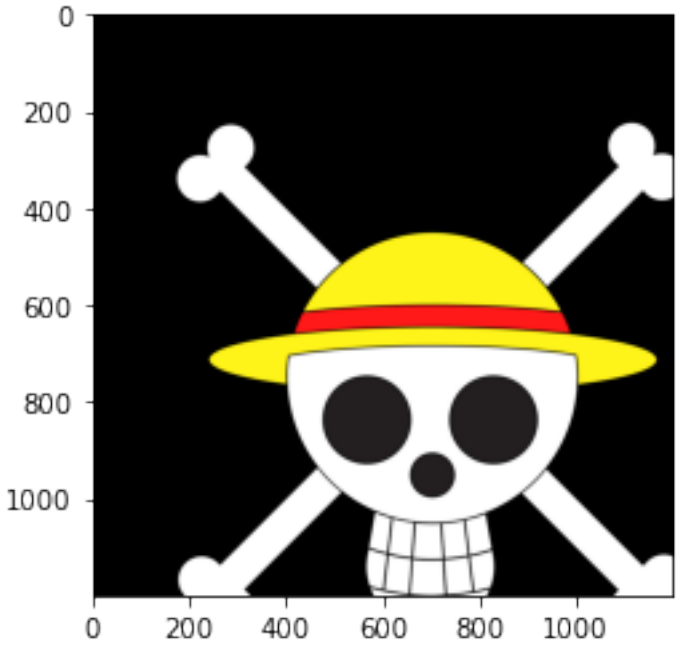
[19]: <matplotlib.image.AxesImage at 0x23341696910>



TRASLACION DE UNA IMAGEN

[22]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/mugi.png")
img=cv.cvtColor(img, cv.COLOR_BGR2RGB)
fil, col,_ = img.shape
M = np.float32([[1,0,100],[0,1,150]])
result=cv.warpAffine(img,M,(col,fil))
plt.imshow(result)
```

[22]:
```
<matplotlib.image.AxesImage at 0x2334595d6d0>
```

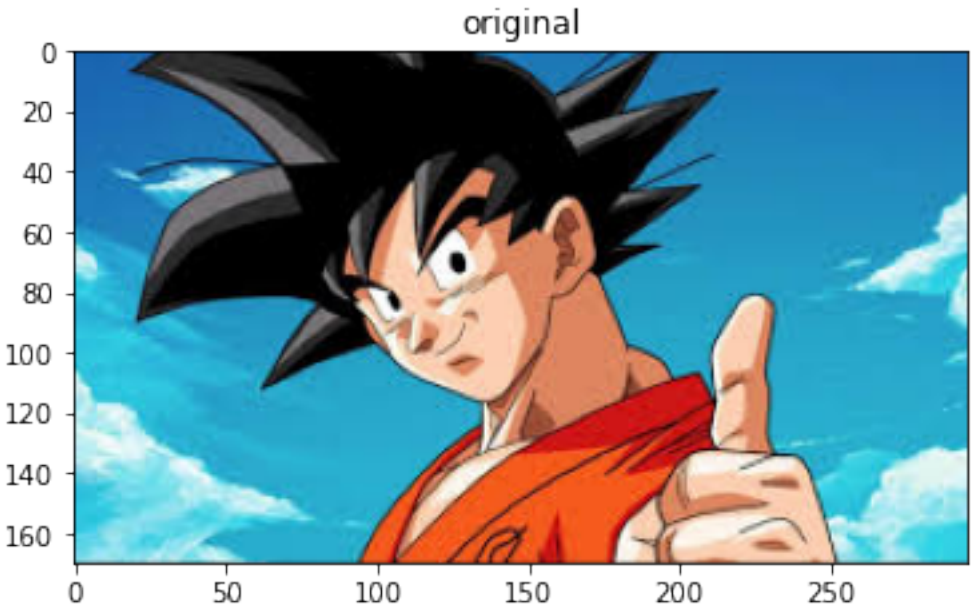


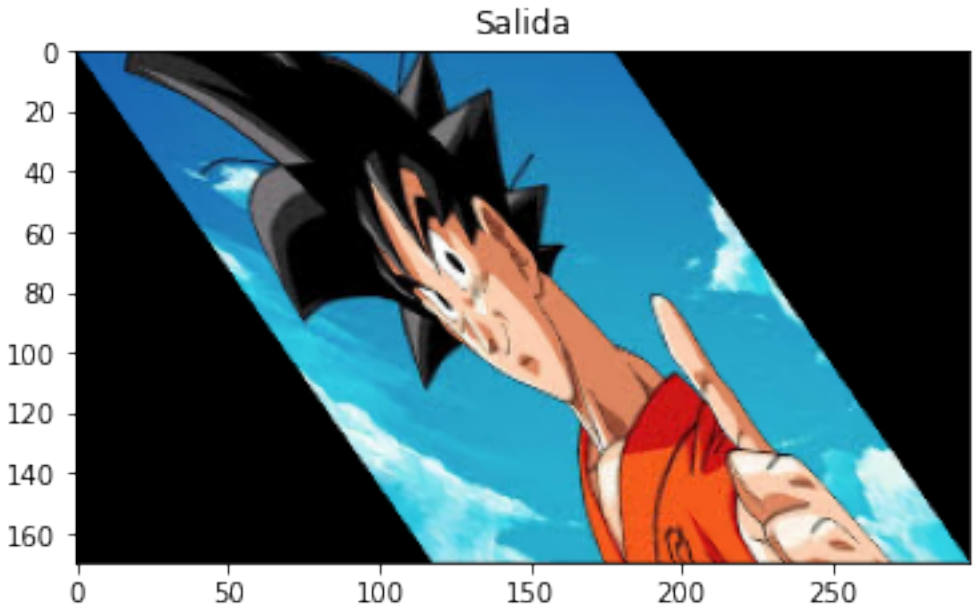
17 TRANSFORMACION AFINE

[4]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/goku.jpg")
img=cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.figure(1)
plt.imshow(img), plt.title("original")
```

```
fil, col,_ = img.shape

#puntos de entrada

scr_points = np.float32([[0,0],[col-1,0],[0,fil-1]])

#puntos de salida
dts_points = np.float32([[0,0] , [int(0.6*(col-1)),0] , [int(0.
 ↪4*(col-1)),fil-1]])

matris_Afin = cv.getAffineTransform(scr_points,dts_points)
#Transformacion
output = cv.warpAffine(img,matris_Afin,(col,fil))
#salida

plt.figure(2), plt.imshow(output), plt.title("Salida")
```

[4]:
```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1d2d9417cd0>,
 Text(0.5, 1.0, 'Salida'))
```

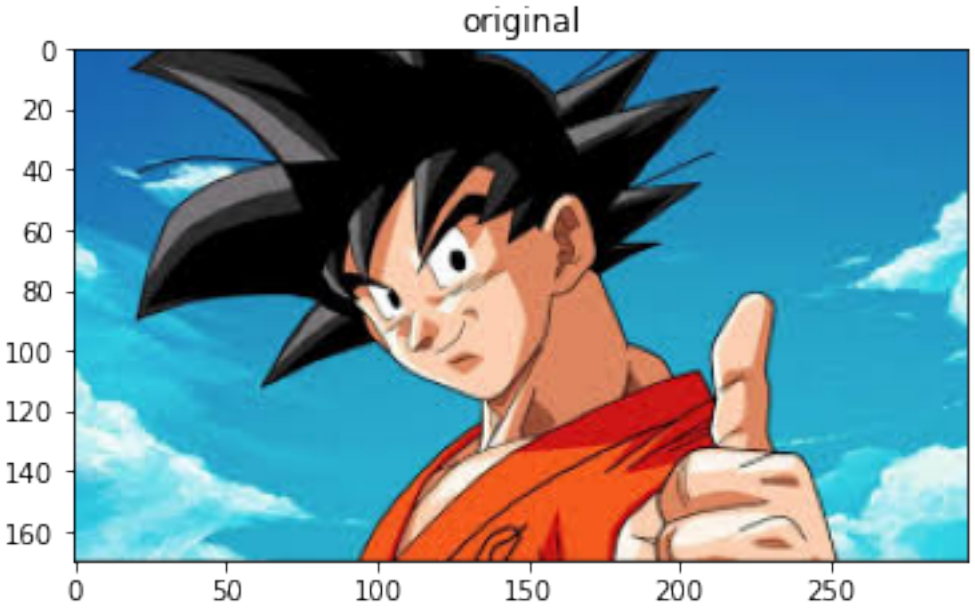


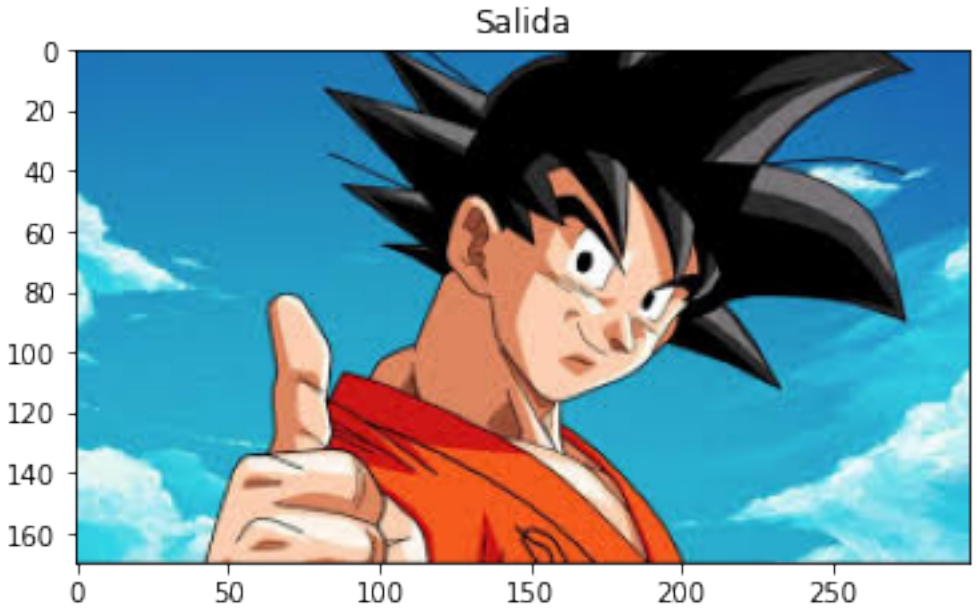


18 ESPEJO

[8]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("Imgs/goku.jpg")
img=cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.figure(1)
plt.imshow(img), plt.title("original")

fil, col,_ = img.shape

#puntos de entrada

scr_points = np.float32([[0,0],[col-1,0],[0,fil-1]])

#puntos de salida
dts_points = np.float32([[col-1,0], [0,0],[col-1, fil-1]])

matris_Afin = cv.getAffineTransform(scr_points,dts_points)
#Transformacion
output = cv.warpAffine(img,matris_Afin,(col,fil))
#salida

plt.figure(2), plt.imshow(output), plt.title("Salida")
```

[8]:
```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1d2d959c130>,
 Text(0.5, 1.0, 'Salida'))
```





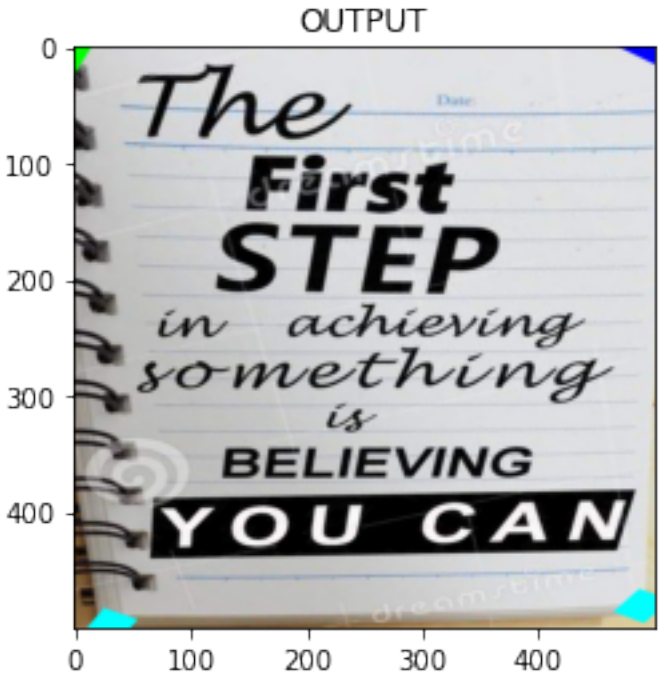
19 TRANSFORMACIONES DE PERSPECTICA

[36]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img=cv.imread("Imgs/imgper.jpg")
img= cv.cvtColor(img,cv.COLOR_BGR2RGB)
#arribaizquiera
img[170:200,520:550]=[0,255,0]
#arriba derecha
img[10:40,820:850]=[0,0,255]
#abajoizquiera
img[610:640,800:830]=[0,255,255]
#abajo derecha
img[410:440,1150:1180]=[0,255,255]
plt.imshow(img)
```

[36]:
```
<matplotlib.image.AxesImage at 0x1d2df733250>
```

[69]:
```
pts1=np.float32([[540,180],[790,640],[840,30],[1190,440]])
pts2=np.float32([[0,0],[0,500-1],[500-1,0],[500-1,500-1]])
matris = cv.getPerspectiveTransform(pts1,pts2)

output=cv.warpPerspective(img, matris,(500,500))
plt.imshow(output),plt.title("OUTPUT")
```

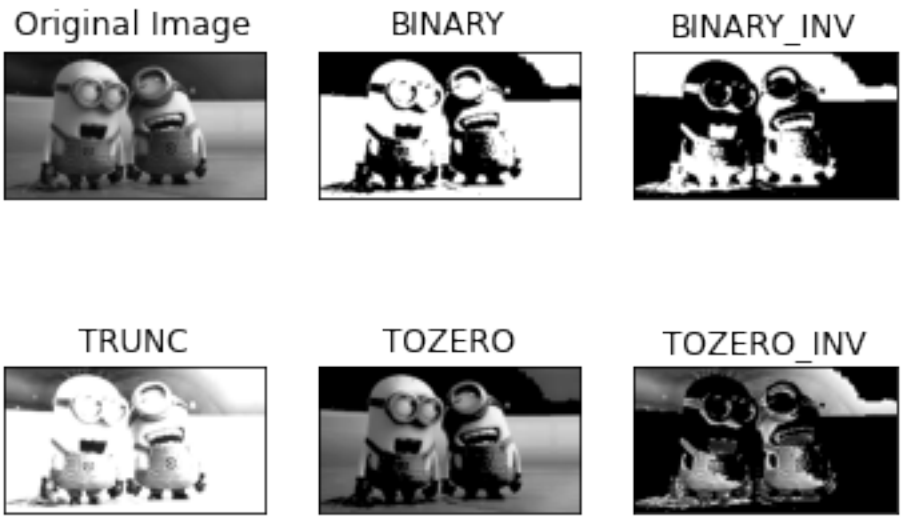
[69]: (<matplotlib.image.AxesImage at 0x1d2e348f070>, Text(0.5, 1.0, 'OUTPUT'))



20 UMBRALIZACION DE IMAGENES

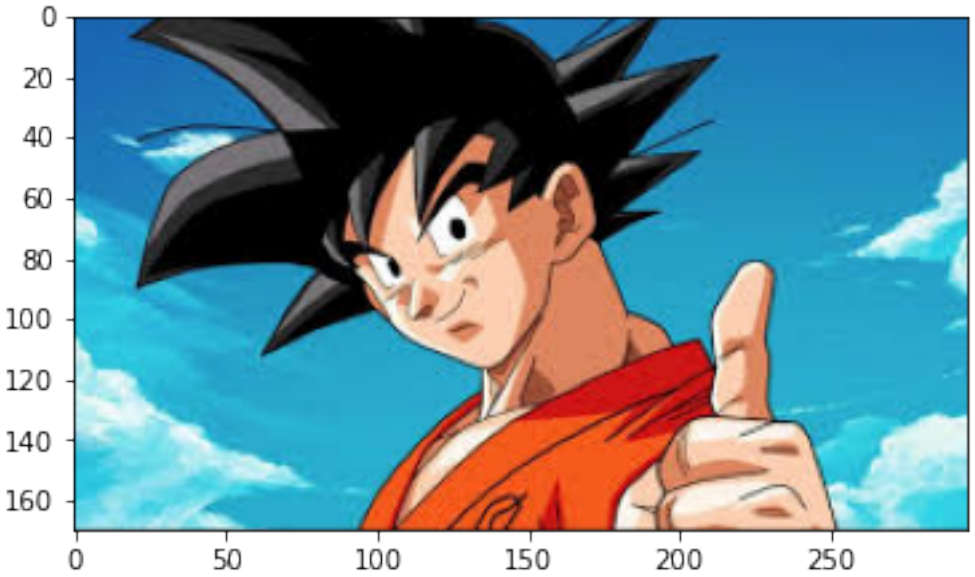
UMBRALIZACION SIMPLE

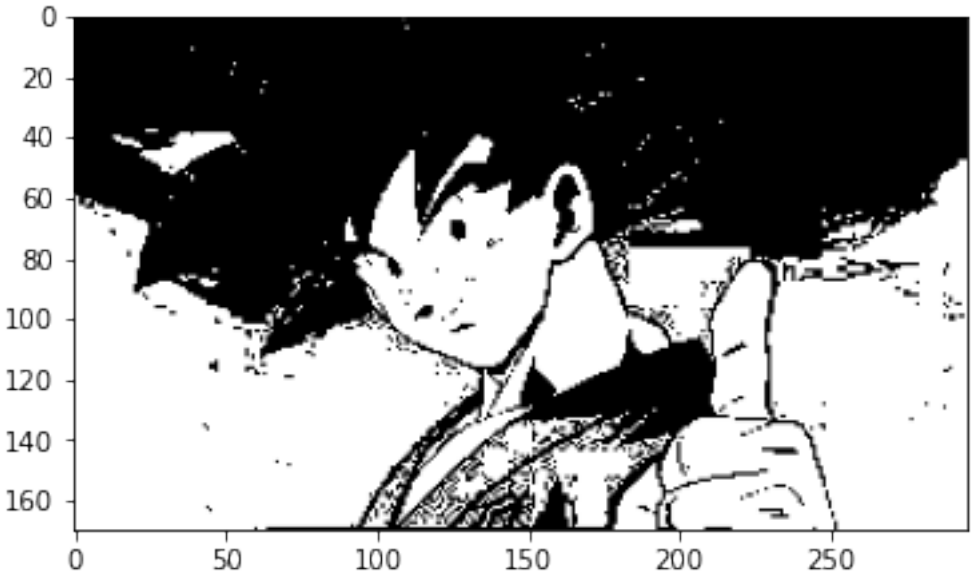
[13]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('imgs/img1.jpg')
img = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
ret,thresh1 = cv.threshold(img,70,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,70,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,70,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,70,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,70,255,cv.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```



[14]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('imgs/goku.jpg')
img = cv.cvtColor(img,cv.COLOR_BGR2RGB)
img2 = cv.cvtColor(img,cv.COLOR_RGB2GRAY)
ret,th1 = cv.threshold(img2,127,255,cv.THRESH_BINARY)
plt.figure(1),plt.imshow(img)
plt.figure(2),plt.imshow(th1, cmap="gray")
```

[14]:
```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1d2da655b50>)
```





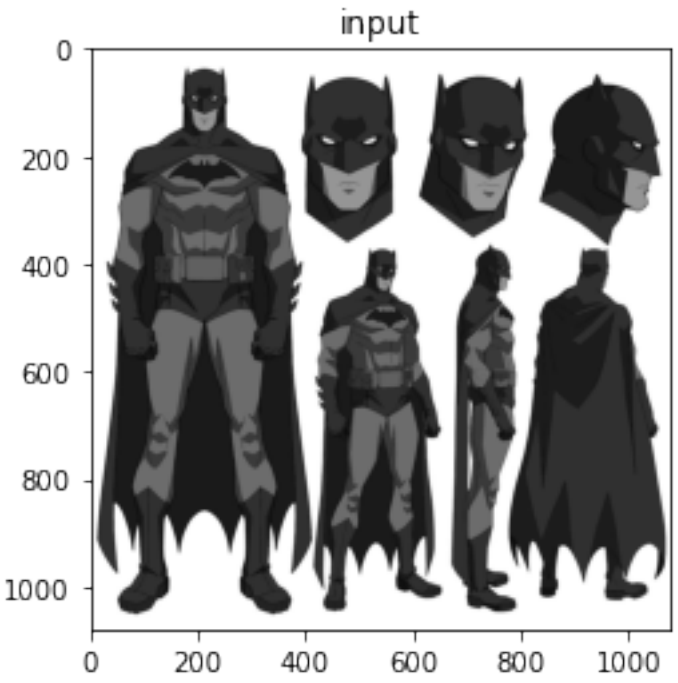
21 Umbral Adaptativo

Es un algoritmo que umbraliza dependiendo de la iluminazion.

[7]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

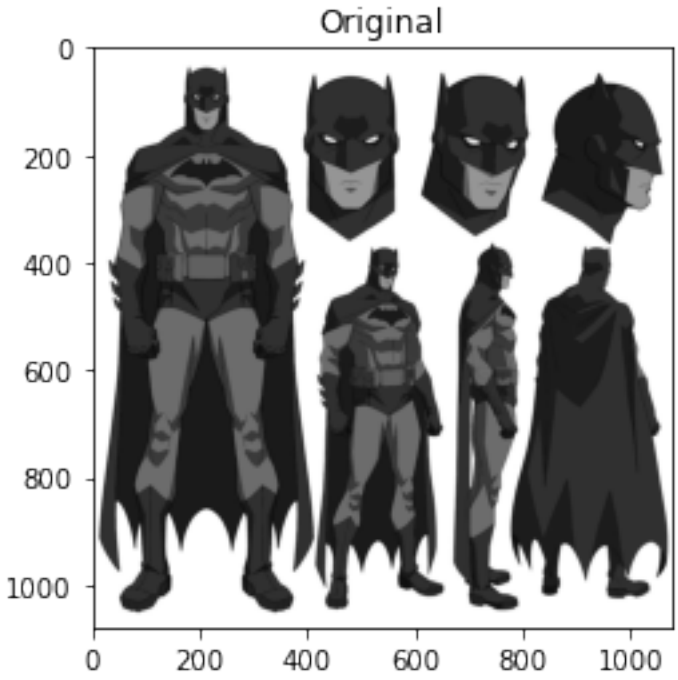
```
#lectura de imagen
img= cv.imread("imgs/bat.jpg")
img_gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#filtrado
img_gray=cv.medianBlur(img_gray, 3)
plt.imshow(img_gray, cmap="gray"), plt.title("input")
```

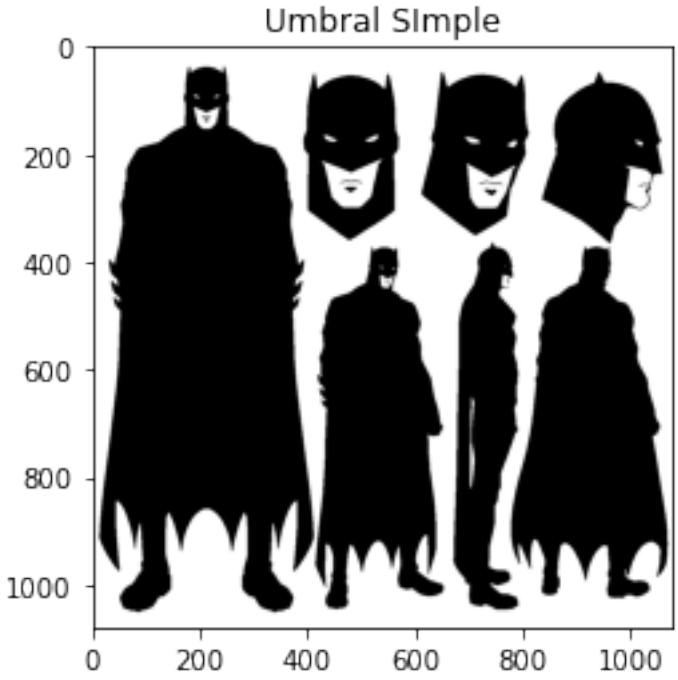
[7]: (<matplotlib.image.AxesImage at 0x1b430feb610>, Text(0.5, 1.0, 'input'))

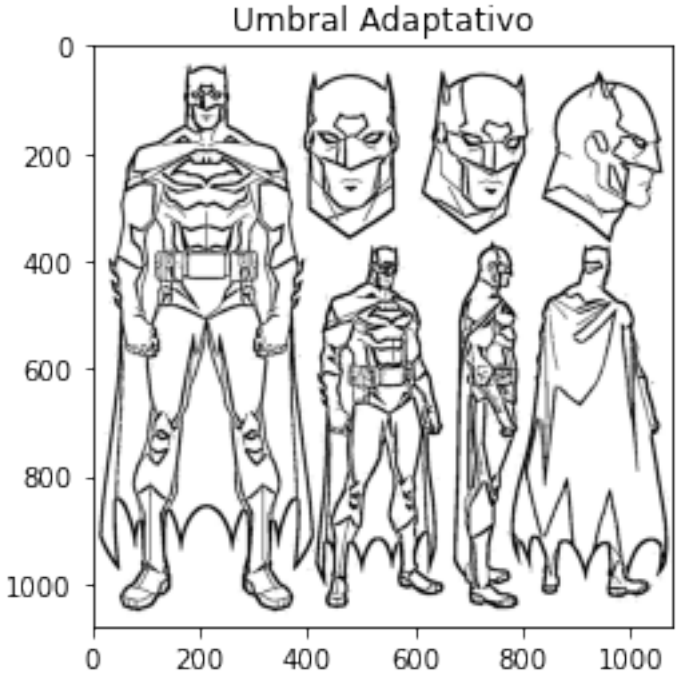


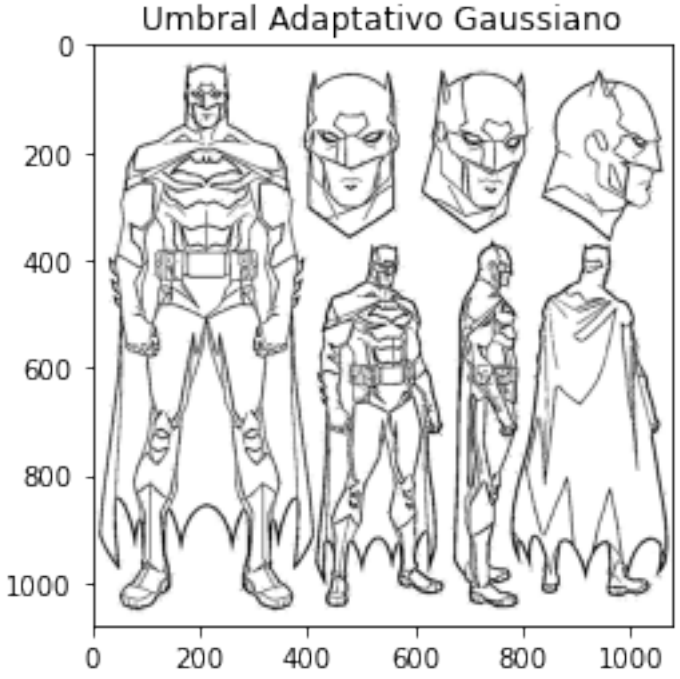
[13]:
```
#Umbral simple
ret, th1=cv.threshold(img_gray, 127,255, cv.THRESH_BINARY)
#Umbral Adaptativo
th2=cv.adaptiveThreshold(img_gray,255, cv.ADAPTIVE_THRESH_MEAN_C, cv.
 ↪THRESH_BINARY, 11,2)
#Umbral adaptativo gausiano
th3=cv.adaptiveThreshold(img_gray,255, cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.
 ↪THRESH_BINARY, 11,2)

plt.figure(1),plt.imshow(img_gray, cmap="gray"),plt.title("Original")
plt.figure(2),plt.imshow(th1, cmap="gray"),plt.title("Umbral SImple")
plt.figure(3),plt.imshow(th2, cmap="gray"),plt.title("Umbral Adaptativo")
plt.figure(4),plt.imshow(th3, cmap="gray"),plt.title("Umbral Adaptativo␣
 ↪Gaussiano")
```

[13]: (<Figure size 432x288 with 1 Axes>,
<matplotlib.image.AxesImage at 0x1b4310ada90>,
Text(0.5, 1.0, 'Umbral Adaptativo Gaussiano'))





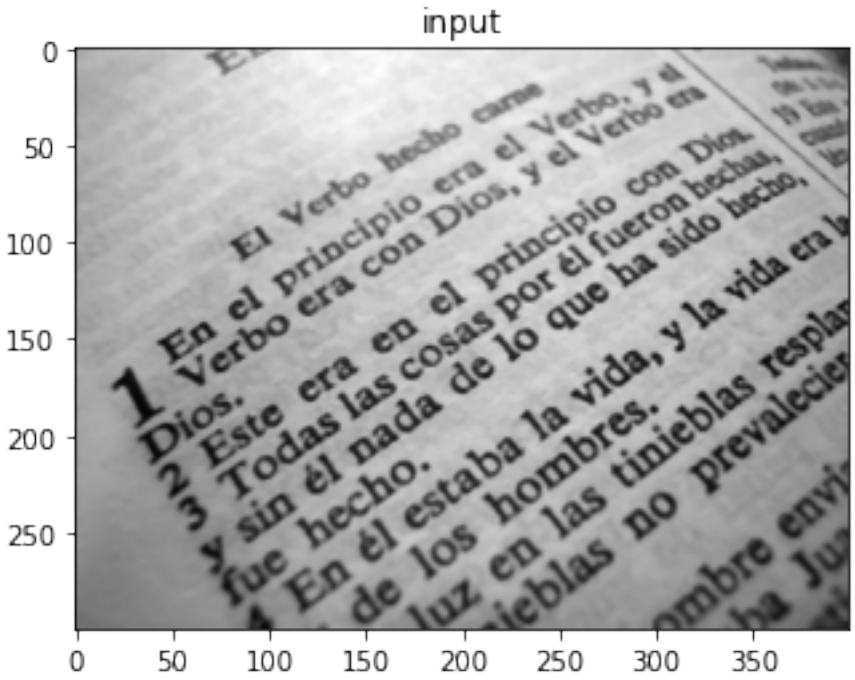




Si fuera texto lo que hace es mejorar la calidad de texto para luego pasar a texto digital.

[16]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

#lectura de imagen
img= cv.imread("imgs/txt.jpg")
img_gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#filtrado
img_gray=cv.medianBlur(img_gray, 3)
plt.imshow(img_gray, cmap="gray"), plt.title("input")
```

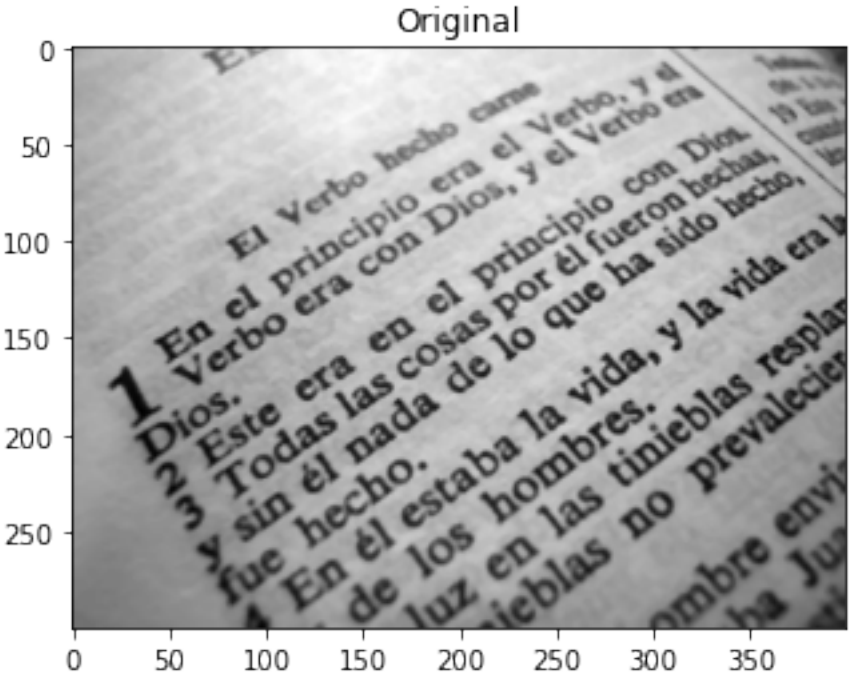
[16]:
```
(<matplotlib.image.AxesImage at 0x1b4300b33d0>, Text(0.5, 1.0, 'input'))
```

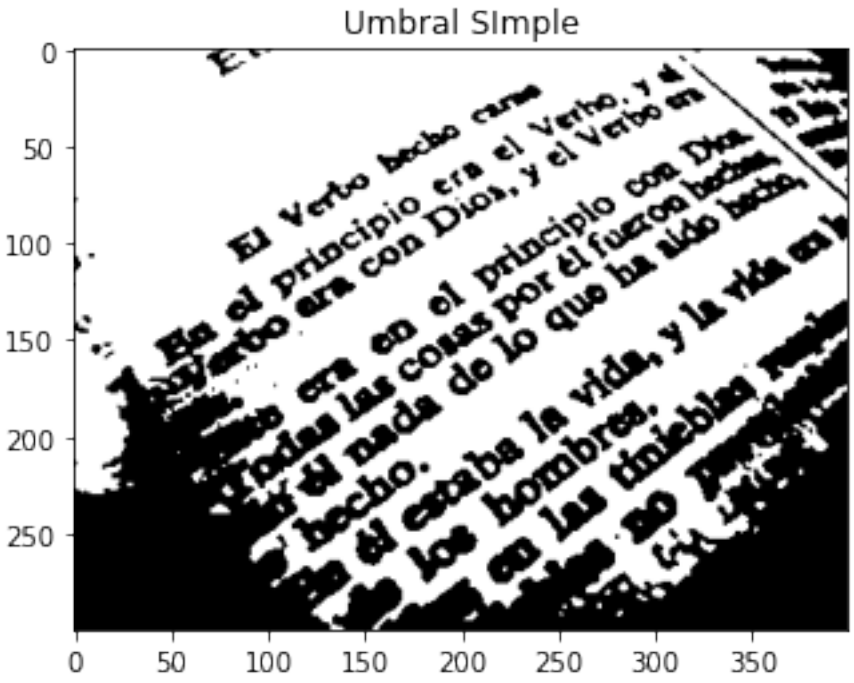


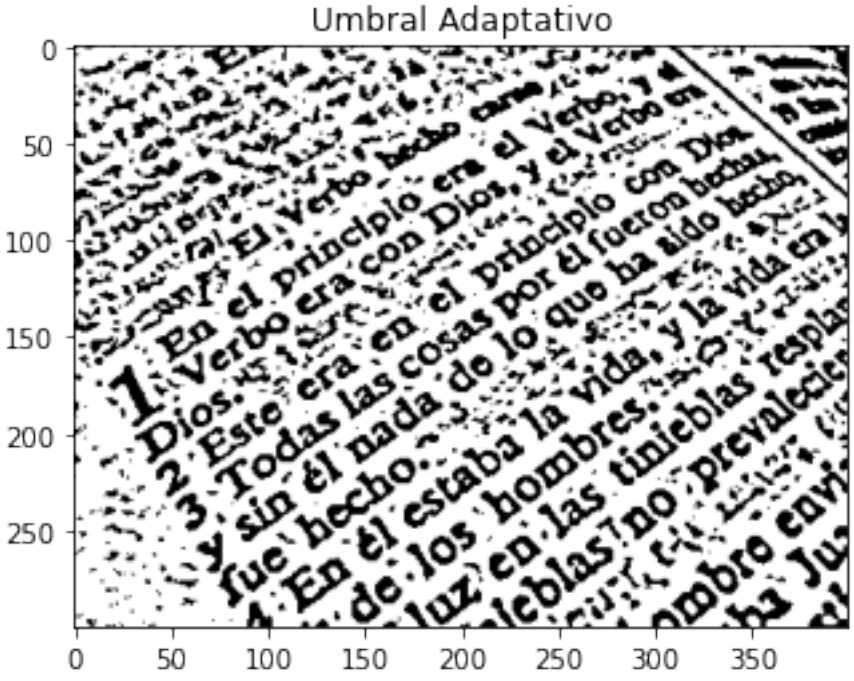
[17]:
```
#Umbral simple
ret, th1=cv.threshold(img_gray, 127,255, cv.THRESH_BINARY)
#Umbral Adaptativo
th2=cv.adaptiveThreshold(img_gray,255, cv.ADAPTIVE_THRESH_MEAN_C, cv.
 ↪THRESH_BINARY, 11,2)
#Umbral adaptativo gausiano
th3=cv.adaptiveThreshold(img_gray,255, cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.
 ↪THRESH_BINARY, 11,2)

plt.figure(1),plt.imshow(img_gray, cmap="gray"),plt.title("Original")
plt.figure(2),plt.imshow(th1, cmap="gray"),plt.title("Umbral SImple")
```

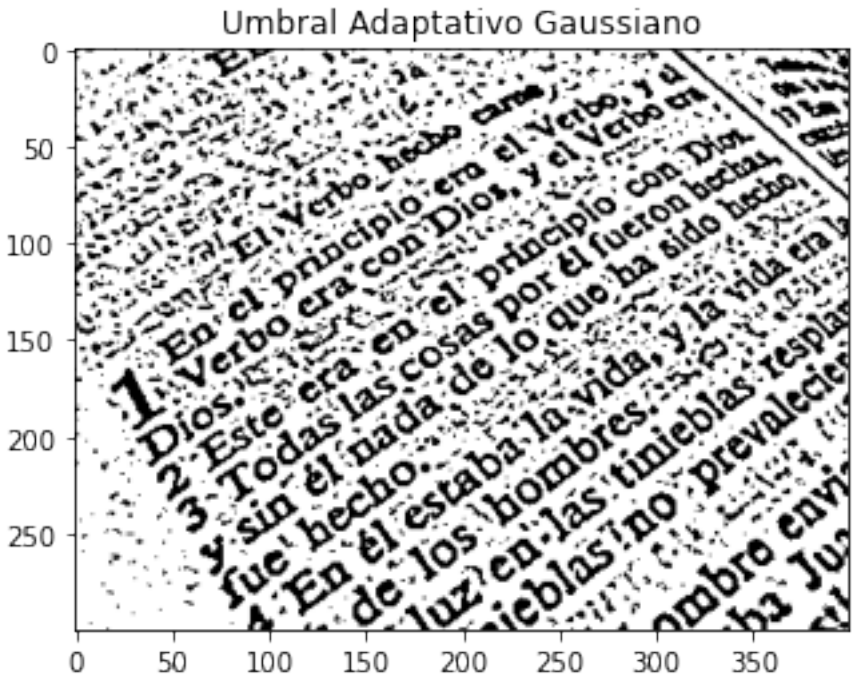
```
plt.figure(3),plt.imshow(th2, cmap="gray"),plt.title("Umbral Adaptativo")
plt.figure(4),plt.imshow(th3, cmap="gray"),plt.title("Umbral Adaptativo␣
↪Gaussiano")
```

[17]:
```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1b42f36ea30>,
 Text(0.5, 1.0, 'Umbral Adaptativo Gaussiano'))
```




Umbral SImple
0
50
100
150
200
250
0
50
100
150
200
250
300
350


Umbral Adaptativo
0
50
100
150
200
250
0
50
100
150
200
250
300
350

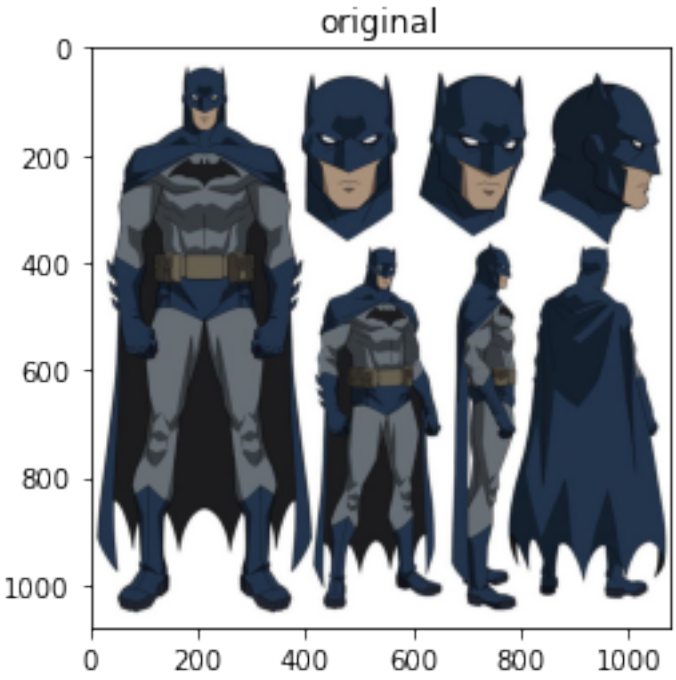


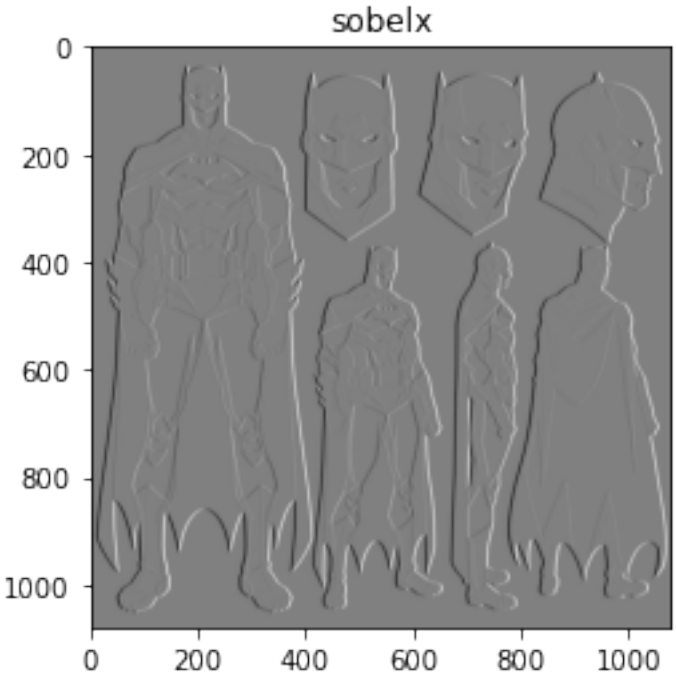
22 Deteccion de bordes es un filtrado en paso alto

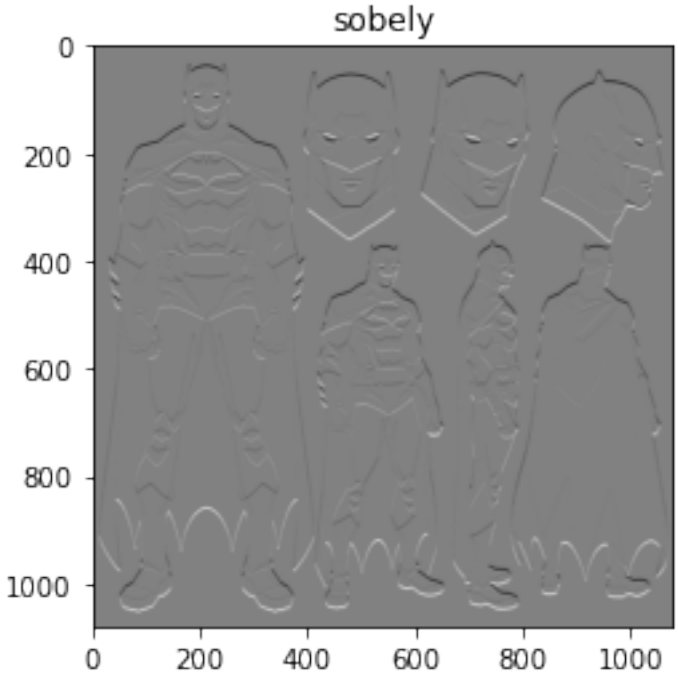
23 Sobel

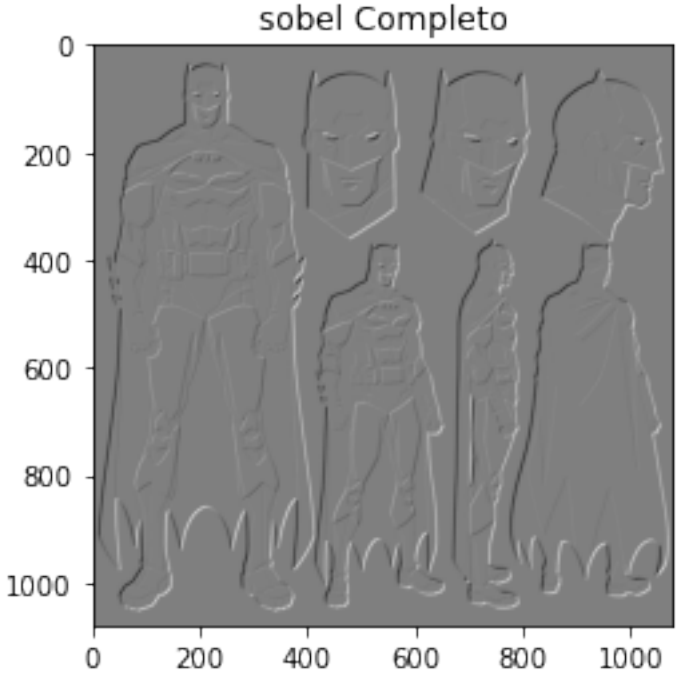
[36]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("imgs/bat.jpg")
img= cv.cvtColor(img, cv.COLOR_BGR2RGB)
img_gray=cv.cvtColor(img, cv.COLOR_RGB2GRAY)
#filtro de sobel en direccion x con profundidad de cv.64F
sobelx=cv.Sobel(img_gray, cv.CV_64F, 1,0 , ksize=7)
sobely=cv.Sobel(img_gray, cv.CV_64F, 0,1 , ksize=7)

sobel = sobelx+sobely

plt.figure(1), plt.imshow(img), plt.title("original")
plt.figure(2), plt.imshow(sobelx, cmap="gray"), plt.title("sobelx")
plt.figure(3), plt.imshow(sobely, cmap="gray"), plt.title("sobely")
plt.figure(4), plt.imshow(sobel, cmap="gray"), plt.title("sobel Completo")
```

[36]: (<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1b42f49bcd0>,
 Text(0.5, 1.0, 'sobel Completo'))





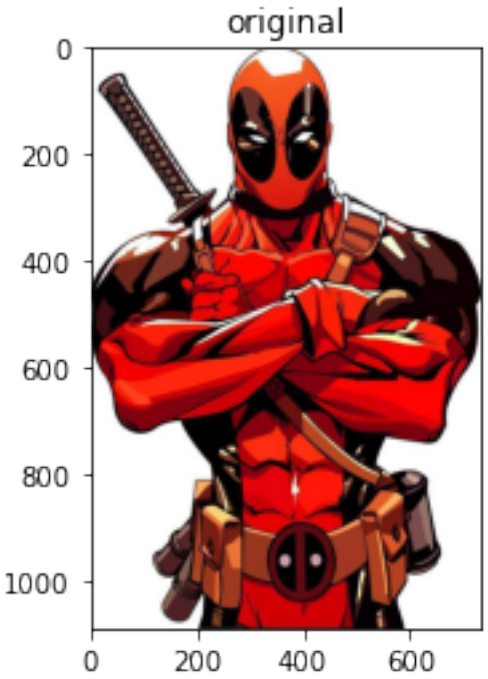

sobely
0
200
400
600
800
1000
0
200
400
600
800
1000

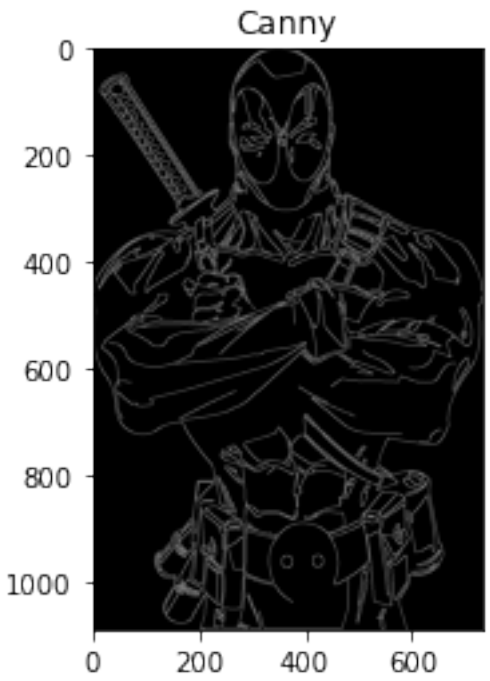

sobel Completo
0
200
400
600
800
1000
0
200
400
600
800
1000

24 Canny detector de bordes

[38]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("imgs/pool.jpg")
img= cv.cvtColor(img, cv.COLOR_BGR2RGB)
img_gray=cv.cvtColor(img, cv.COLOR_RGB2GRAY)

canny=cv.Canny(img_gray, 100, 255)
plt.figure(1), plt.imshow(img), plt.title("original")
plt.figure(2), plt.imshow(canny, cmap="gray"), plt.title("Canny")
```

[38]:
```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.image.AxesImage at 0x1b431f57130>,
 Text(0.5, 1.0, 'Canny'))
```





Es la base para Machine Learning y Deep Learning

25 Deteccion de Objetos con Haar Cascade

Como entranar un clasificador haar se necesita imagenes positivas es decir imagenes con objtos de estudio y negativos escenarios sin dicho objeto.

26 Detector de rostros

[61]:
```
import numpy as np
import cv2 as cv
face_cascade = cv.CascadeClassifier("haarcascade_frontalface_default.xml")
img=cv.imread("imgs/rostro.jpg")
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)

faces= face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=1)#el␣
 ↪1 elimina flasos ngativos
for x,y,w,h in faces:
    cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv.imshow("faces", img)
cv.waitKey(0)
cv.destroyAllWindows()
```

27 DETECTOR DE OJOS

[56]:
```
import numpy as np
import cv2 as cv
eyes_cascade = cv.CascadeClassifier("haarcascade_eye.xml")
img=cv.imread("imgs/rostro.jpg")
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)

faces= eyes_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=8)#el␣
↪3 elimina flasos ngativos
for x,y,w,h in faces:
    cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv.imshow("faces", img)
cv.waitKey(0)
cv.destroyAllWindows()
```

28 DETECCION DE OBJETOS DESDE LA CAMARA WEB

29 Deteccion de rostro

[68]:
```
import numpy as np
import cv2 as cv
face_cascade = cv.CascadeClassifier("haarcascade_frontalface_default.xml")
cap=cv.VideoCapture(0)

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


        gray=cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        faces= face_cascade.detectMultiScale(gray, scaleFactor=1.3,␣
↪minNeighbors=2)#el 3 elimina flasos ngativos
        for x,y,w,h in faces:
            cv.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

        cv.imshow("faces", frame)

        if cv.waitKey(10) & 0xFF == ord('q'):
            break
    else:
        break
```

```
cap.release()
cv.destroyAllWindows()
```

30 Deteccion de ojos

[73]:
```
import numpy as np
import cv2 as cv
face_cascade = cv.CascadeClassifier("haarcascade_eye.xml")
cap=cv.VideoCapture(0)

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



        gray=cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        faces= face_cascade.detectMultiScale(gray, scaleFactor=1.3,␣
↪minNeighbors=2)#el 3 elimina flasos ngativos
        for x,y,w,h in faces:
            cv.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

        cv.imshow("faces", frame)

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


cap.release()
cv.destroyAllWindows()
```