



GÖRÜNTÜ İŞLEME YARDIMCI NOTLARI -2023-



GENEL FONKSİYON ve KOMUTLAR

Resim üzerine çerçeve

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread("anahtar.jpg")
x[0:2,0:]=[255,0,0] #en üst satıra 3 piksel kalınlıkta mavi çizgi [BGR, RGB değil.]
x[-3:,0:]=[0,255,0] #en alt satıra 3 piksel kalınlıkta yeşil çizgi
x[0:,0:5]=[0,0,255] #en soldaki sütuna 5 piksel kalınlıkta kırmızı çizgi
x[0:,-10:,0:3]=0 #Dikkat! Bu gösterim farklı. En sağdaki sütuna 10piksel kalınlıkta siyah çizgi

cv2.imshow('cerceve',x)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Gri Seviye Dönüşüm ve Tip Dönüşüm İşlemleri

Gri seviye dönüşüm işlemleri görüntü işlemede çok sık olarak kullanılmaktadır. Resimlerin filtreleme, eşikleme ve öznetelik çıkartma gibi bir takım işlemlerinden önce çoğu zaman gri seviyeye dönüştürülmeleri gerekir. Gri dönüşümlerde bazı farklı algoritmalar uygulanır. En çok kullanılanlardan bir tanesi BGR bantlarının ortalamasının alınmasıdır. Aşağıdaki programda bununla ilgili örnekler bulunmaktadır. En doğru yöntem için sonuçlar değerlendirilmelidir.

Ayrıca aşağıdaki örnekte python üzerinde dizilerin ve matrislerin tip dönüşüm örnekleri de bulunmaktadır.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread("headquarters.jpg")
gri1=(x[:,:,0]+x[:,:,1]+x[:,:,2])/3 #modlu işlem. toplamları 255'i geçen değerler hep 255 olarak kalır.
gri2=x[:,:,0]/3+x[:,:,1]/3+x[:,:,2]/3
print(x.dtype)
print(gri1.dtype) # gri1 ve gri2 artık float tipine dönüştüler.
print(gri2.dtype)
```

```
cv2.imshow('gri1',gri1)
cv2.imshow('gri2',gri2)
```

```

cv2.waitKey(0)
cv2.destroyAllWindows()
gri1=np.uint8(gri1) #tip dönüşümü yapıldı.
print(gri1.dtype)
gri2=np.uint8(gri2)#tip dönüşümü yapıldı.
print(gri2.dtype)

```

```

cv2.imshow('gri1',gri1)
cv2.imshow('gri2',gri2)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

Aşağıdaki algoritmayı kullanarak kalemler.jpg resminde gri seviye dönüşüm işlemini gerçekleştiriniz.

$$\text{Gri}=0.11*B + 0.44*G + 0.28*R$$

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread("balonlar2.jpg")
y=0.11*x[:, :,0]+0.44*x[:, :,1]+0.28*x[:, :,2]
y=np.uint8(y) #tip dönüşümü yapıldı.
cv2.imshow('orjinal',x)
cv2.imshow('gri seviye',y)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

Aynı programı for döngüsü kullanarak gerçekleştiriniz.

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread("kalemler.jpg")
for i in range(x.shape[0]):
    for j in range(x.shape[1]):
        x[i,j]=0.11*x[i,j,0]+0.44*x[i,j,1]+0.28*x[i,j,2]
cv2.imshow('orjinal',x)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

SubPlot() kullanımı:

Bazı zamanlarda birden fazla resmi aynı anda görmemiz gerekebilir. Bu durumda plt.subplot() fonksiyonunu kullanabiliriz. Kullanımı için **matplotlib** kütüphanesinin eklenmesi şarttır.

Kullanımı:

```
plt.subplot(1, 2, 1), plt.imshow( '1. görüntü', goruntu1)
```

```
plt.subplot(1, 2, 2), plt.imshow( '2. görüntü', goruntu2)
```

(1,2,1)'in anlamı: 1x2'lik bir matris aç ve bu matrisin 1. Elemanının bulunduğu yere ilk gelen imshow() görüntüsünü yerleştir demektir.

Örnek:

```
import cv2 as cv  
from matplotlib import pyplot as plt  
image1 = cv.imread('balonlar2.jpg')  
image2 = cv.imread('anahtar.jpg')  
plt.subplot(1, 2, 1), plt.imshow(image1, 'gray')  
plt.subplot(1, 2, 2), plt.imshow(image2, 'gray')  
plt.show()
```

