KIRIKKALE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ

Temel Görüntü İşlemleri

- Görüntü Okuma ve Yazma İşlemleri
- Görüntü Özellikleri
- Görüntü Çerçeveleme
- Görüntüde Aritmetik İşlemler
- Çözünürlüğe Göre Pencereyi Yeniden Boyutlandırma
- Format Değiştirme
- Görüntü Piramitleri
- Görüntüye Çeşitli Şekiller ve Metin Ekleme

Görüntü Okuma ve Yazma İşlemleri

OpenCV kütüphanesi, pek çok farklı programlama dili içerisinde görüntü işleme komutlarının kullanımı için sıklıkla başvurulan en önemli kütüphanelerden birisidir.

https://pypi.org/project/opencv-python/

https://docs.opencv.org/4.5.2/d5/de5/tutorial_py_setup_in_windows.html

Dökümantasyon ve kullanım için yükleme bilgilerine yukarıdaki bağlantılardan ulaşılabilir.

Görüntü Okuma ve Yazma İşlemleri

Kütüphaneyi kullanabilmek için öncelikle dahil edilmesi gerekiyor.

import cv2

Daha sonrasında bir görüntüyü okuma işlemi

imread

komutu ile gerçekleştirilir. Tanımlanan adresten okunan görüntü ise

imshow

komutuyla görüntülenebilir. (imshow komutu ile görüntüleme için «waitkey» komutunun da kullanılması gerekir)

Görüntü Okuma ve Yazma İşlemleri

imread fonksiyonunun kullanılabileceği üç farklı durum vardır:

- cv2.IMREAD_COLOR : Renkli bir görüntü yükler. Resmin herhangi bir şeffaflığı ihmal edilecektir. Varsayılan bayraktır.
- cv2.IMREAD_GRAYSCALE : Görüntüyü gri tonlamalı modda yükler.
- cv2.IMREAD_UNCHANGED : Alfa kanalı da dahil olmak üzere görüntüyü olduğu gibi yükler.

Bu üç bayrak yerine, sırasıyla 1, 0 veya -1 tamsayıları kullanılabilir.

Görüntü Okuma ve Yazma İşlemleri

```
#%%
#Bayraklar

view = cv2.imread("img/kku_kampus.jpg", cv2.IMREAD_COLOR)
cv2.imshow("Manzara", view)
cv2.waitKey(0)

view2 = cv2.imread("img/kku_kampus.jpg", 1)
cv2.imshow("Manzara2", view2)
cv2.waitKey(0)

view3 = cv2.imread("img/kku_kampus.jpg", cv2.IMREAD_GRAYSCALE)
cv2.imshow("Manzara3", view3)
cv2.waitKey(0)

view4 = cv2.imread("img/kku_kampus.jpg", 0)
cv2.imshow("Manzara4", view4)
cv2.waitKey(0)

view5 = cv2.imread("img/kku_kampus.jpg", cv2.IMREAD_UNCHANGED)
cv2.imshow("Manzara5", view5)
cv2.waitKey(0)

view6 = cv2.imread("img/kku_kampus.jpg", -1)
cv2.imshow("Manzara6", view6)
cv2.waitKey(0)
```

Görüntü Okuma ve Yazma İşlemleri

cv2.imshow fonksiyonuyla görüntülenen görüntü penceresi otomatik olarak görüntü boyutuna sığar.

- Fonksiyon kullanımındaki ilk argüman, bir pencere adıdır.
- İkinci argüman ise görüntüdür.

İstenildiği kadar pencere oluşturulabilir, ancak farklı pencere adlarıyla.

cv2.waitKey() bir klavye bağlama işlevidir. Argümanı ise milisaniye cinsinden zamandır. İşlev olarak herhangi bir klavye olayı için belirtilen milisaniyeler boyunca bekler. O sırada herhangi bir tuşa basarsanız program devam eder. 0 iletilirse, bir tuş vuruşu için süresiz olarak bekler. Ayrıca, a tuşuna basılıp basılmadığı gibi belirli tuş vuruşlarını algılamak için de ayarlanabilir.

Görüntü Okuma ve Yazma İşlemleri

Görüntü üzerinde işlemler gerçekleştirdikten sonra kaydetme işlemi ise;

imwrite

komutu ile gerçekleştirilir.

```
cv2.imwrite("img/kku_logo2.jpg", img)
```

Görüntü Özellikleri

Yüklenen görüntü üzerinde belirli bölgelerin renk değerleri öğrenilebilir, genişlik ve yükseklik değerleri ile renk uzayı öğrenilebilir. Yüklenen görüntünün özelliklerini görmek için;

print(img.shape)

Komutu kullanılır. Bu komut sonrasında görüntünün yükseklik-genişlik-renk kanalı bilgileri ekrana yazdırılır.

Örnek Çıktı: (855, 827, 3)

Görüntü Özellikleri

Görüntü üzerinde belirli bir alanın her bir renk kanalı için (RGB – Kırmızı, Yeşil, Mavi) değerlerini yazdırmak da mümkündür.

```
print("50x50 Alanındaki Mavi Renk Değeri:" + str(img.item(50,50,0))) # 0 = Mavi
print("200x200 Alanındaki Yeşil Renk Değeri:" + str(img.item(200,200,1))) # 1 = Yeşil
print("250x250 Alanındaki Kırmızı Renk Değeri:" + str(img.item(250,250,2))) # 2 = Kırmızı
```

```
50x50 Alanındaki Mavi Renk Değeri:255
200x200 Alanındaki Yeşil Renk Değeri:36
250x250 Alanındaki Kırmızı Renk Değeri:6
```

Herhangi bir noktanın piksel değerini yazdırmak da mümkündür.

```
px= img[100,100]
print(px)
```

Görüntü Özellikleri

Görüntünün yükseklik ve genişlik değerlerini öğrenmek içinse;

.shape[0] ve .shape[1]

Komutları kullanılır.

```
height = img.shape[0]
width = img.shape[1]
print('Görüntü Yüksekliği (px)      :', height)
print('Görüntü Genişliği  (px)     :', width)
```

```
Görüntü Yüksekliği (px)      : 855
Görüntü Genişliği  (px)     : 827
```

.size ile toplam piksel değeri görüntülenebilir. Veri türü ise .dtype ile görüntülenebilir.

print(img.size) → 2121255

print(img.dtype) → uint8

Görüntüyü Çerçeveleme

Bir görüntüyü çerçevelemek için copyMakeBorder fonksiyonu kullanılır. Bu fonksiyonun;

src: görüntü yolu,

top-bottom-left-right: çerçevenin piksel kalınlık değeri,

value: renk veya referans noktası,

borderType: çerçeve türü,

gibi parametreleri bulunmaktadır.

cv2.BORDER_CONSTANT: Belirlenen renkte çerçeve ekler.
cv2.BORDER_REFLECT: Görüntünün sınırlarına ayna efekti eklenir.
cv2.BORDER_ REPLICATE: Görüntünün belirlenen piksel alanından çekme yapılmış gibi bir görüntü verir.
cv2.BORDER_WRAP: Çerçevenin karşılıklı kenarlarında kırpma/ekleme görünümü sağlar.

Görüntüyü Çerçeveleme

```
#%%
#Çerçeve Ekleme
border_color = [0, 0, 255 ]
border=cv2.copyMakeBorder(img, 10,10,25,25, cv2.BORDER_CONSTANT, value=border_color)

cv2.imshow("Cerceve",border)
cv2.waitKey(0)


border2= cv2.copyMakeBorder(img,20,20,20,20,cv2.BORDER_REFLECT)
cv2.imshow("Cerceve2",border2)
cv2.waitKey(0)

campus = cv2.imread("campus.jpg")
border3= cv2.copyMakeBorder(campus,50,50,50,50,cv2.BORDER_REPLICATE)
cv2.imshow("Cerceve3",border3)
cv2.waitKey(0)

border4= cv2.copyMakeBorder(campus,50,50,50,50,cv2.BORDER_WRAP)
cv2.imshow("Cerceve4",border4)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Görüntüyü Çerçeveleme

openCV kütüphanesiyle yapılabildiği gibi, matplotlib kütüphanesinden pyplot fonksiyonu kullanılarak da yine görüntüleme işlemi gerçekleştirilebilir.

Ancak burada OpenCV ile pyplot arasındaki renk dizilimi farkları ve color-grayscale görüntü okuma farkları göz önünde bulundurulmalıdır.

Görüntüyü Çerçeveleme

```
from matplotlib import pyplot as plt
RED = [255,0,0]
img1 = cv2.imread('kku_logo.jpeg')

replicate = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REPLICATE)
reflect = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REFLECT)
wrap = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_WRAP)
constant= cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_CONSTANT,value=RED)

plt.subplot(231),plt.axis("off"),plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)),plt.title('ORIGINAL')
plt.subplot(232),plt.axis("off"),plt.imshow(replicate,'gray'),plt.title('REPLICATE')
plt.subplot(233),plt.axis("off"),plt.imshow(reflect,'gray'),plt.title('REFLECT')
plt.subplot(234),plt.axis("off"),plt.imshow(img1,'gray'),plt.title('ORIGINAL_GRAY')
plt.subplot(235),plt.axis("off"),plt.imshow(wrap,'gray'),plt.title('WRAP')
plt.subplot(236),plt.axis("off"),plt.imshow(constant,'gray'),plt.title('CONSTANT')
plt.show()
```

Aritmetik İşlemler

Görüntü Ekleme

OpenCV kütüphanesinden cv2.add() kullanarak veya NumPy işlemi olarak;

res = img1 + img2

ile iki görüntü eklenebilir. Bu işlem için her iki görüntü de aynı derinlikte ve türde olmalıdır veya ikinci görüntü yalnızca skaler bir değer olabilir.

Bu iki işlem arasında (OpenCV ile Numpy eklemeleri) arasında bir fark vardır. OpenCV eklemesi doygun bir işlemdir, Numpy eklemesi ise bir modül işlemidir.

İki resim eklendiğinde bu durum daha belirgin olacaktır. OpenCV işlevi daha iyi bir sonuç sağlayacaktır. Bu nedenle OpenCV işlevlerine bağlı kalmak daha iyidir.

```
import numpy as np

x = np.uint8([230])
y = np.uint8([60])

print(cv2.add(x,y))
print(x+y)
```

```
[[255]]
[34]
```

Aritmetik İşlemler

Görüntü Karıştırma

Bu işlem de aynı zamanda görüntü eklemedir, ancak görüntülere karıştırma veya şeffaflık hissi vermesi için farklı ağırlıklar verilir. Resimler aşağıdaki denkleme göre eklenir:

$$g(x) = (1 - \alpha) f_0(x) + \alpha f_1(x)$$

Alfa değeri 0 ile 1 arasında değiştirilerek, bir görüntüden diğerine geçiş yapılabilir.

Aritmetik İşlemler

Görüntü Karıştırma

```
dim = (300, 300)
img1 = cv2.imread('kku_logo2.png')
img1 = cv2.resize(img1,dim)
img2 = cv2.imread('opencv_logo.png')
img2 = cv2.resize(img2,dim)
```

```
join = cv2.addWeighted(img1,0.2, img2,0.8, 0)
cv2_imshow(join)
```

Aritmetik İşlemler

Bitsel İşlemler

Bit düzeyinde AND, OR, NOT ve XOR işlemlerini kapsar.

Görüntünün herhangi bir bölümünü çıkarırken, dikdörtgen olmayan RoI'yi (Region of Interest) tanımlarken ve bunlarla çalışırken oldukça faydalı olacaklardır.

- AND: Bit düzeyinde bir AND, yalnızca ve ancak her iki pikselin de sıfırdan büyük olması durumunda doğrudur.
- OR: İki pikselden biri sıfırdan büyükse, bit düzeyinde VEYA doğrudur.
- XOR: Bit düzeyinde bir XOR, yalnızca iki pikselden birinin sıfırdan büyük olması, ancak her ikisinin birden olmaması durumunda doğrudur.
- NOT: Bit düzeyinde NOT, bir görüntüdeki "açık" ve "kapalı" pikselleri tersine çevirir.

Konuyla ilgili uygulamalar için:
https://www.pyimagesearch.com/2021/01/19/opencv-bitwise-and-or-xor-and-not/

Aritmetik İşlemler

Bitsel İşlemler

cv2.cvtColor() yöntemi, bir görüntüyü bir renk uzayından diğerine dönüştürmek için kullanılır.

OpenCV'de 150'den fazla renk alanı dönüştürme yöntemi mevcuttur.

cv2.COLOR_BGR2GRAY renkli görüntüleri gri tonlamalı görüntülere dönüştürür.

Aritmetik İşlemler

Bitsel İşlemler

cv.threshold işlevi, eşikleme işlemi uygulamak için kullanılır.

İlk parametre, gri tonlamalı bir görüntü olması gereken kaynak görüntüdür.

İkinci parametre, piksel değerlerini sınıflandırmak için kullanılan eşik değeridir.

Üçüncü parametre, eşiği aşan piksel değerlerine atanan maksimum değerdir.

OpenCV, fonksiyonun dördüncü parametresi tarafından verilen farklı türlerde eşikleme sağlar.

Aritmetik işlemler

Bitsel İşlemler

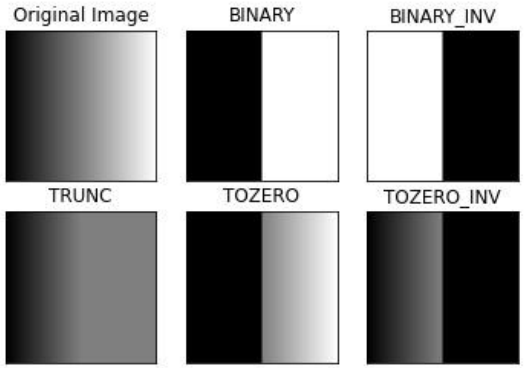
Eşikleme çeşitleri şunlardır:

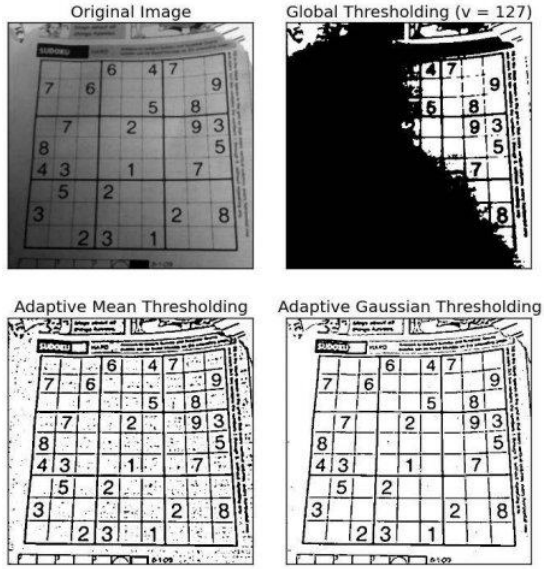
- **cv.THRESH_BINARY**
- **cv.THRESH_BINARY_INV**
- **cv.THRESH_TRUNC**
- **cv.THRESH_TOZERO**
- **cv.THRESH_TOZERO_INV**

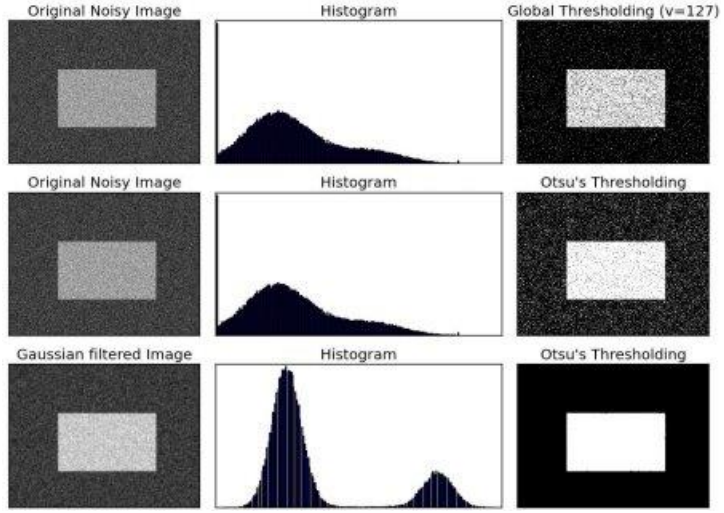
Bu eşikleme işlemlerine ait detaylı bilgilere:
https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html
sayfasından erişilebilir.

Aritmetik İşlemler

Eşikleme Örnekleri







otsu

Aritmetik İşlemler

```
img1 = cv2.imread('kku_kampus.jpg')
img2 = cv2.imread('kku_logo.jpeg')
logo=cv2.resize(img2, [300,300])

# sol üst köşeye logo boyutlarında bir RoI oluşturuyoruz
rows, cols, channels = logo.shape
roi = img1[0:rows, 0:cols ]

# Logo maskesi ve ters maske işlemleri
img2gray = cv2.cvtColor(logo,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)

# Logo alanını maskeleme
img1_bg = cv2.bitwise_and(roi, roi, mask = mask_inv)

# Logo resmi içerisinden sadece logoyu alma
img2_bw = cv2.bitwise_and(logo, logo, mask = mask)

# Logoyu ROI belirlenen alana ekleme
dst = cv2.add(img1_bg, img2_bw)
img1[0:rows, 0:cols ] = dst

cv2_imshow(img1)
```

Çözünürlüğe Göre Pencereyi Yeniden Boyutlandırma

Görüntülerin her zaman orijinal boyutları değil de, farklı çözünürlük oranlarıyla da işlemler yapılması gerekebilir. Yeniden boyutlandırılabilir görüntüler oluşturmak için matematiksel işlemlerle pencere boyutları ayarlanabilir.

```
#Görüntü Boyutlandırma
img=cv2.imread("img/kku_kampus.jpg")
resolution = 600, 400
scala_width=resolution[0]/img.shape[1]
scala_height=resolution[1]/img.shape[0]

scala=min(scala_width,scala_height)

window_width=int(img.shape[1]*scala)
window_height=int(img.shape[0]*scala)

cv2.namedWindow('Boyutlanabilir Pencere',cv2.WINDOW_NORMAL)
cv2.resizeWindow('Boyutlanabilir Pencere',window_width, window_height)
cv2.imshow('Boyutlanabilir Pencere',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Format Değiştirme

Bir görüntüyü kaydederken .JPG veya .PNG türünde kayıt gerçekleştirilebilir.

cv2.**imwrite**(filename, img[, params])

Parametreler:

Filename: Dosya adı

img: Kayıt edilecek görüntü

Params:

paramId_1, paramValue_1, paramId_2, paramValue_2, … çiftleri olarak kodlanmış biçime özgü kaydetme parametreleridir.

- JPEG için, 0 ile 100 arasında bir kalite (CV_IMWRITE_JPEG_QUALITY) olabilir (ne kadar yüksekse o kadar iyidir). Varsayılan değer 95'tir.
- PNG için, 0'dan 9'a kadar sıkıştırma düzeyi (CV_IMWRITE_PNG_COMPRESSION) olabilir. Daha yüksek bir değer, daha küçük boyut ve daha uzun sıkıştırma süresi anlamına gelir. Varsayılan değer 3'tür.
- PPM, PGM veya PBM için ikili format bayrağı (CV_IMWRITE_PXM_BINARY), 0 veya 1 olabilir. Varsayılan değer 1'dir.

Örnek için: https://www.life2coding.com/save-opencv-images-jpeg-quality-png-compression/

Görüntü Piramitleri

Görüntü işlemede resmin orijinal çözünürlüğü her zaman kullanılmayabilir. Özellikle nesne bulma ve/veya görüntü üzerinde farklı bölgelerde çalışma gibi işlemlerde, çözünürlük değiştirme işlemleri gerçekleştirilebilir. Bu işlem için;

cv2.pyrUp(),

cv2.pyrDown()

Fonksiyonları kullanılmaktadır. Gauss Piramidi ve Laplace Piramidi olmak üzere iki tür görüntü piramidi oluşturma yöntemi vardır.

Gauss Piramidi

Sürekli yenilenen bir bilgisayar tarafından işlenen bir dizi görüntüdür. Bir görüntü bulanıklaştığında ve bir alt kümesi üretildiğinde, genellikle kendisinden daha küçük bir resim kümesi üretilir.

Bir görüntünün pikselleri veya en küçük parçaları, genellikle önceki görüntüdekilerden daha fazla ortalama içerik alır.

Farklı piramit seviyeleri, orijinal resmi temsil eden üst kısım ve art arda gelen görüntüleri temsil eden alt seviyelere karşılık gelir.

Gauss Piramidi

Piramidin birinci seviyesine olan k düzeyi denir. Genellikle ilk görüntüden bir alt örnek oluşturulmadan önce bir Gauss bulanıklığı üretilerek işlenir; sonraki aşamaya genellikle seviye k + 1 denir.

Konvolüsyon adı verilen işlem, genellikle bu görüntünün belirli bir kısmındaki yoğunluğun ortalamasını içeren bu aşamada gerçekleştirilir.

Gauss piramidinin her basamağının hesaplanması, genellikle bir dizi matematiksel formül gerektirir.

Bunlar trigonometri, türev ve fonksiyon prensiplerini bütünleştirir; görüntü işleme ile ilgili sorunları çözmek için uzun denklemler kullanılabilir.

Laplace Piramidi

Laplace piramitleri Gauss piramitlerinden oluşur. Laplace piramitleri için özel bir işlem yoktur ve OpenCV'de özel bir fonksiyon bulunmamaktadır.

Laplace piramidi görüntüleri kenar görüntüleri gibidir.

Laplace piramidindeki bir seviye; Gauss piramidindeki bir seviye ile Gauss piramidinin en üst seviyedeki görüntüsünün arasındaki farktan oluşur.

Pyrup() ve pyrdown()

```
#Görüntü Piramitleri
resim=cv2.imread('img/kku_kampus.jpg')
pyr_ust=cv2.pyrUp(resim);
cv2.imshow('Kampus Orjinal',resim)
cv2.imshow('Piramit Bir Ust Seviye',pyr_ust)
cv2.waitKey(0)
cv2.destroyAllWindows()

pyr_alt=cv2.pyrDown(resim)
pyr_alt2=cv2.pyrDown(pyr_alt)
cv2.imshow('Orjinal Resim', resim)
cv2.imshow('Piramit Bir Alt Seviye', pyr_alt)
cv2.imshow('Piramit iki Alt Seviye', pyr_alt2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Görüntü Üzerine Çeşitli Şekiller Ekleme

Bu işlemleri gerçekleştirmek için kullanılan fonksiyonlar:

- **cv2.line()**,
- **cv2.circle()** ,
- **cv2.rectangle()**,
- **cv2.ellipse()**,
- **cv2.putText()**

Bu fonksiyonların görevleri İngilizce isimlerinden de anlaşılabilir.

Görüntü Üzerine Çeşitli Şekiller Ekleme

Bu fonksiyonlarla kullanılan çeşitli parametreler vardır:

img : Şekilleri çizmek istediğiniz resim,

color : Şeklin rengi. BGR için dizi olarak sunulur. örneğin: mavi için (255,0,0). Gri tonlama için sadece skaler değer iletilir.

thickness : Çizginin veya dairenin kalınlığı vs. Daire gibi kapalı şekiller için -1 değeri girilirse şekil doldurulacaktır. varsayılan kalınlık = 1.

lineType : Hattın türü, ister 8 bağlantılı, ister kenar yumuşatılmış hat vs. Varsayılan olarak 8 bağlantılıdır. cv2.LINE_AA, eğriler için kenar yumuşatma çizgisi verir.

Görüntü Üzerine Çeşitli Şekiller Ekleme

Çizgi

Bir çizgi çizmek için çizginin başlangıç ve bitiş koordinatlarını belirtmek gerekir.

Dikdörtgen

Bir dikdörtgen çizmek için dikdörtgenin sol üst köşesine ve sağ alt köşesine ihtiyaç vardır.

Görüntü Üzerine Çeşitli Şekiller Ekleme

Daire

Bir daire çizmek için merkez koordinatları ve yarıçapı belirtilir.

Elips

Elips çizmek için birkaç parametre iletilir. Bir parametre merkez konumudur (x,y). Sonraki parametre eksen uzunluklarıdır (ana eksen uzunluğu, ikincil eksen uzunluğu). Açı, elipsin saat yönünün tersine dönme açısıdır. startAngle ve endAngle, ana eksenden saat yönünde ölçülen elips yayının başlangıcını ve bitişini belirtir. 0 ve 360 değerleri vermek tam elipsi verir.

Metin ekleme

Resimlere metin eklemek için şu bilgiler girilir:

- Yazılacak metin,
- Metin koordinatları,
- Yazı tipi türü (Desteklenen yazı tipleri için cv2.putText()),
- Yazı Tipi boyutu,
- renk, kalınlık, lineType vb. Daha iyi görünüm için lineType = cv2.LINE_AA

Görüntü Üzerine Çeşitli Şekiller Ekleme

```
#%%
#Çizgi
img = cv2.imread("img/kku_kampus.jpg")
x,y,ch =img.shape # yüklenen resmin çözünürlüğü değişkenlere alınıyor
cv2.line(img,(0,0),(y-1,x-1),(0,0,255),3) # Köşeden köşeye 3 piksel kırmızı renkli çizgi
cv2.imshow("Kampüs", img)
cv2.waitKey(0)
```

```
#%%
#Dikdörtgen
cv2.rectangle(img,(0,int(x/2)),(int(x/4),int(y/4)),(0,255,0),3)
cv2.imshow("Kampüs", img)
cv2.waitKey(0)
```

```
#%%
#Daire
cv2.circle(img,(int(y/2),int(x/2)), 63, (255,255,0), 1)
cv2.imshow("Kampüs", img)
cv2.waitKey(0)
```

```
#%%
#Elips
cv2.ellipse(img,(int(y/4),int(x/4)),(100,50),0,0,180,255,1)
cv2.imshow("Kampüs", img)
cv2.waitKey(0)
```

```
#%%
#Metin
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Kirikkale University',(0,x-50), font, 2,(255,255,255),5,cv2.LINE_AA)
cv2.imshow("Kampüs", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Uygulamalar ve parametrelerin açıklamaları için: https://docs.opencv.org/4.5.2/d6/d6e/group__imgproc__draw.html

KAYNAKLAR

- OpenCV, Online: https://docs.opencv.org/4.5.2/d5/de5/tutorial_py_setup_in_windows.html, Erişim T.: 30.08.2021.
- Pypi, Online: https://pypi.org/project/opencv-python/, Erişim T.: 30.08.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html, Erişim T.: 30.08.2021.
- Aydın, Y. E., MobilHanem, Online: https://www.mobilhanem.com/opencvde-resim-cevceveleme-islemi-ve-cozunurluge-gore-islemler/, Erişim T.: 30.08.2021.
- OpenCV, Online: https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html, Erişim T.: 31.08.2021.
- Pyimagesearch, Online: https://www.pyimagesearch.com/2021/01/19/opencv-bitwise-and-or-xor-and-not/, Erişim T.: 31.08.2021
- Life2Coding, Online: https://www.life2coding.com/save-opencv-images-jpeg-quality-png-compression/, Erişim T.: 01.09.2021.
- Netinbag, Online: https://www.netinbag.com/tr/internet/what-is-a-gaussian-pyramid.html, Erişim T.: 01.09.2021.
- Şimşek, T. E., Online: https://www.turanerdemsimsek.com/2017/11/goruntu-piramidi.html, Erişim T.: 01.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_drawing_functions/py_drawing_functions.html#drawing-functions, Erişim T.: 01.09.2021.
- OpenCV, Online, https://docs.opencv.org/4.5.2/d6/d6e/group__imgproc__draw.html, Erişim T.: 01.09.2021.