**Python ile Görüntü İşleme**

**(OpenCV)**

**OpenCV Nedir?**

OpenCV, gerçek zamanlı bilgisayar vizyonunu amaçlayan bir programlama fonksiyon kütüphanesidir. Kütüphane çapraz platformdur ve açık kaynaklı BSD lisansı altında kullanım için ücretsizdir.Kütüphane aynı zamanda makine öğrenmesinide desteklemektedir.

Cplusplus diliyle geliştirilmektedir.Kütüphane içerisinde 500’den fazla algoritma ve bu sayının yaklaşık 10 katı kadar da fonksiyon yer almaktadır.

TensorFlow, Torch/PyTorch , Caffe gibi derin öğrenme frameworklerini destekler.Linux,Windows,macOs,FreeBSD gibi masaüstü işletim sistemlerinde ve İOS , Android gibi mobil işletim sistemlerinde çalışabilmektedir.

**OpenCV ile Neler Yapılabilmektedir?**

* Sabit görselleri , videoları veya webcam görüntülerini okuyup kaydetme
* Yüzleri ve yüz özelliklerini saptama
* Görseller üzerindeki yazıları saptama (plaka,tabela vb.)
* Nesne saptama , sayma ve takibi
* Yüzlerdeki duyguları anlama
* Hareket takibi gibi işlemler yapılabilmektedir.

**OpenCV Hangi Yazılım Dillerini Destekler?**

* C++
* Java
* Python
* Android SDK

**Resim Açma , Resim Okuma , Resim Yazma**

Python dosyası oluşturduktan sonra içerisine openCV kütüphanesini dahil ediyoruz.

import cv2

Resmimizi okutmak için bir tane resim değişkeni oluşturuyoruz.

resim = cv2.imread("Resimler/seinfield.jpg")

Resmimizi bu şekilde okutursak resmimizi olduğu gibi yani renkli olarak açacaktır.Ancak ikinci parametre olarak “0” (sıfır) değerini verirsek resmimiz gri olacaktır.Gri olmasının nedeni tüm pikseller 0-255 arasında bir değer alacak ve tek bir renk kanalından oluşacaktır.Bu konuya ileride daha detaylı değinilecektir.

Resmimizi ekranda göstermek için :

cv2.imshow("Seinfield Resmi",resim)

Burada ilk parametre açılacak pencerenin başlığını ikinci parametre ise resmimizi okuttuğumuz resim değişkenimizi belirtir.

Resim açıldığında hemen kapanmaması için :

cv2.waitKey(0)

waitKey() fonksiyonu resim açıldığında ekranda ne kadar süre kalacağını belirten bir fonksiyondur.Parametreside delay yani bekleme süresidir.Ancak biz parametre olarak “0”(sıfır) değerini verirsek, biz herhangi bir tuşa basmadan resim penceremiz açık kalacaktır.

Açılan tüm pencerelerin kapatılması için:

cv2.destroyAllWindows()

komutu kullanılır.

Program Çıktısı :



**OpenCV Resmimizi Nasıl Yorumlar?**

OpenCV ‘de resimlerimizi bir numpy dizisi olarak tutar bunu görmek için:

print(type(resim))

kodunu çalıştırdığımızda bize

<class 'numpy.ndarray'>

değerini dönmektedir.Buradan resimlerimizin bir numpy array’i olarak saklandığını görebiliriz

**Renk Sistemleri**

Bilgisayarların anlamlandırdığı belirli renk sistemleri ve renk uzayları (color space)bulunmaktadır.Bunlardan bazıları RGB,BGR,HSV ve YCbCr’dır.

**RGB(Red,Green,Blue)**

Çok yaygın kullanılan bir renk sistemidir.İngilizce anlamı olarak red(kırmızı),gree(yeşil),blue(mavi) renklerinin baş harflerinden oluşturulmuştur.

Bu sistem, bilgisayar ve televizyon gibi elektronik cihazların algılama ve sunum mekanizmalarında;ayrıca fotoğrafçılık alanında yaygın şekilde kullanılmaktadır.Birbirine çok yakın biçimde konumlandırılan temel renk kaynakları birbirini etkileyerek insan gözünün algılayabildiği renk dizilimindeki bileşimleri oluşturur.

Renkleri belirtmek için tamsayılar kullanılır.8bitlik tam sayılar kullanıldığında aynı rengin 256 ,16bitlik sayılar kullanıldığında 65536 farklı tonunu elde etmek mümkündür.

OpenCV’de genellikle 8bitlik renkler kullanılır.

**OpenCV ve BGR Mantığı**

RGB renk sistemiyle temelde aynıdır.Sadece Kırmızı ve mavi renkler sıralamada birbirlerinin yerine geçmişlerdir.

OpenCV’nin ana renk sistemi BGR(blue,green,red)’dır.

OpenCV ‘de resmimizi print fonksiyonu ile yazdırdığımızda bize bir matris döner.Bu matris piksellerin BGR renk modelindeki renk skalasını döner.

print(resim)

Blue Green Red

[ 12 15 20]

[ 12 15 20]

[11 14 19]

...

Bize bu şekilde çok uzun bir matris değerleri dönmektedir.Her satır bir pikseli ifade etmektedir ve değerler 3 farklı kanalda gösterilir(R G B).Bu şekilde mavi , yeşil ve kırmızı kullanarak tüm renkler elde edilebilir.Her bir renk skalası için değerler 0 – 255 arasındadır.Yani her bir kanal için 256 farklı ton vardır.

**Peki bir piksel için kaç farklı renk kombinasyonu vardır?**

256 \* 256 \* 256 = 16.777.216 farklı renk kombinasyonu vardır.

**Resimler Neden Gri Tona Çevirilir?**

OpenCV’de görüntü işleme çalışılırken resimler üzerinde gri ton kullanılır.Bunun nedeni resimlerimizin 3 farklı kanalda değil de tek bir kanaldan üretilmesidir.Bu hem resmimizin boyutunu küçültür.Hem de üzerinde çalışmamız gereken pikselleri azaltır.Resim renkli iken üzerinde işlem yapacağımız 3 farklı kanal olacağından iş yükümüz artmaktadır.Bunu önlemek için tek bir kanal üzerinden çalışılır ve 0 – 255 değerleri arasında çalışma yapılır.Burada 0 -> siyah , 255 -> beyazdır.Resim pikselleri sadece bu iki siyah ve beyaz skalası arasında değer alacağı için gri olur.

**Resim Verileri Hangi Türde Saklanıyor?**

Bunu öğrenmek için :

print(resim.dtype)

komutunu çalıştırırız ve bize;

uint8(unsigned integer 8 bit ) değerini döner bu ne demektir:

0 0 0 0 0 0 0 0 = 0(min)

1 1 1 1 1 1 1 1 = 255(max)

**Resmimizin Boyutlarını Öğrenmek ?**

Resmimizin yükseklik , genişlik ve renk kanalı değerlerini öğrenmek için :

print(resim.shape)

komutu çalıştırılır ve bize

(500, 390, 3) şeklinde değer döner.Burada birinci değer yükseklik ikinci değer genişlik üçüncü değer ise kaç tane renk kanalı kullanıldığıdır.Eğer resmimizi gri tonda açsaydık bize (500, 390) değerini dönecekti.

**OpenCv’de Yazı Yazmak**

OpenCV kütüphanesinin doğal olarak desteklediği font sayısı kısıtlıdır.Üstelik bu fontlar sadece ASCII karakterleri desteklemektedir.Yani Türkçe karakter kullanarak yazı yazmamız durumunda desteklenmeyen , Türkçe’ye özgü karakterlerin yerinde soru işareti bulunacaktır.

OpenCV de yazılar putText() fonksiyonu ile yazılır.putText () fonksiyonun özelliklerini inceleyelim:

img=cv.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) parametre tanımları bu şekildedir.Şimdi de bunların anlamlarına bakalım

**Parametreler**

|  |  |
| --- | --- |
| **img** | Arkaplan görüntüsüdür.Hangi resmin üzerine yazı yazılacaksa o resimi ifade eder. |
| **text** | String türünde yazıyı ifade eder Türkçe karakter kullanılamaz.Kullanılırsa Türkçe karakterin yerine soru işareti gelecektir. |
| **org** | Resmin sol alt köşesini piksek cinsinden ifade eder. |
| **fontFace** | Sadece HERSHEYFONT ismindeki fontları tanır.Bu fontlar 8 adettir. |
| **fontScale** | Yazının büyüklüğünü ifade eder.Çarpan olarak ifade edilebilir.Normal ve varsayılan değeri 1’dir. |
| **color** | Yazının rengini belirtir.BGR renk uzayına göre olan renkleri içerir. |
| **thickness** | Harflerin çizgi kalınlığını ifade eder. |
| **lineType** | Yazı tipini ifade eder. |
| **bottomLeftOrigin** | Değer eğer False ise solt alt köşeye göre yazıyı düzgün bir biçimde yazar.True değerini verirsek yazıyı ters(tepetaklak) yazar. |

|  |  |
| --- | --- |
| HERSHEY Fontları Nelerdir ? | |
| FONT\_HERSHEY\_SIMPLEX  Python: cv.FONT\_HERSHEY\_SIMPLEX | normal boyutta sans-serif yazı tipi |
| FONT\_HERSHEY\_PLAIN  Python: cv.FONT\_HERSHEY\_PLAIN | küçük boyutta sans-serif yazı tipi |
| FONT\_HERSHEY\_DUPLEX  Python: cv.FONT\_HERSHEY\_DUPLEX | normal boyutta sans-serif yazı tipi (FONT\_HERSHEY\_SIMPLEX’ten daha karmaşıktır.) |
| FONT\_HERSHEY\_COMPLEX  Python: cv.FONT\_HERSHEY\_COMPLEX | normal boyutta sans-serif yazı tipi |
| FONT\_HERSHEY\_TRIPLEX  Python: cv.FONT\_HERSHEY\_TRIPLEX | normal boyutta sans-serif yazı tipi (FONT\_HERSHEY\_COMPLEX’ten daha karmaşıktır.) |
| FONT\_HERSHEY\_COMPLEX\_SMALL  Python: cv.FONT\_HERSHEY\_COMPLEX\_SMALL | FONT\_HERSHEY\_COMPLEX’in küçük versiyonudur. |
| FONT\_HERSHEY\_SCRIPT\_SIMPLEX  Python: cv.FONT\_HERSHEY\_SCRIPT\_SIMPLEX | El-yazı stili yazı tipi |
| FONT\_HERSHEY\_SCRIPT\_COMPLEX  Python: cv.FONT\_HERSHEY\_SCRIPT\_COMPLEX | FONT\_HERSHEY\_SCRIPT\_SIMPLEX’in daha karmaşık bir şekli |
| FONT\_ITALIC  Python: cv.FONT\_ITALIC | Fontların italik yazılıp yazılmayacağını ifade eder sayısal değeri 16’dır. |

Yazı Tipleri Nelerdir ?(lineType)

|  |  |
| --- | --- |
| Enumerator | |
| FILLED  Python: cv.FILLED |  |
| LINE\_4  Python: cv.LINE\_4 | 4-bağlı hat |
| LINE\_8  Python: cv.LINE\_8 | 8-bağlı hat |
| LINE\_AA  Python: cv.LINE\_AA | Kenar yumuşatıcı hat |

3-YazıYazmak.py

import cv2

import numpy as np

fontlar = [

'FONT\_HERSHEY\_SIMPLEX', 'FONT\_HERSHEY\_PLAIN',

'FONT\_HERSHEY\_DUPLEX','FONT\_HERSHEY\_COMPLEX',

'FONT\_HERSHEY\_TRIPLEX','FONT\_HERSHEY\_COMPLEX\_SMALL',

'FONT\_HERSHEY\_SCRIPT\_SIMPLEX',

'FONT\_HERSHEY\_SCRIPT\_COMPLEX'] *#8 tane fontumuzla yazı yazmak için 8 tane yazı oluşturduk*

imaj = np.ones((720,780,3),np.uint8)\*255*#beyaz imaj (resim)*

for j in range(8):*#font numaraları 8 tane olduğu için sırasıyla 0'dan başlayarak farklı fontlar oluşturacak.Bu değerler 0-7 arasında toplam 8 tanedir.*

cv2.putText(imaj,fontlar[j],(20,40+j\*40),j,1.1,(0,0,0),1)

*'''*

*putText fonksiyonun parametreleri*

*birinicisi Hangi görsele ekleyeceğimizi*

*ikincisi Ne yazacağımızı*

*üçüncüsü Görselin hangi kısmına ekleyeceğimizi*

*dördüncüsü Hangi formatta yazacağımızı*

*beşincisi Skalamızı ekliyoruz (boyut)*

*altıncısı Hangi renk ile yazacağımızı*

*yedincisi de kalınlık belirtir*

*biz burada opencv'nin bize sunduğu tüm fontlar ile yazı yazdık bunu for döngüsü içerisinde yaptık*

*'''*

italik = 16 #yazıyı italik yapmak için fontun değerine 16 ekliyoruz.Bazı font stilleri italik

# yazıyı desteklememektedir.

for j in range(8):

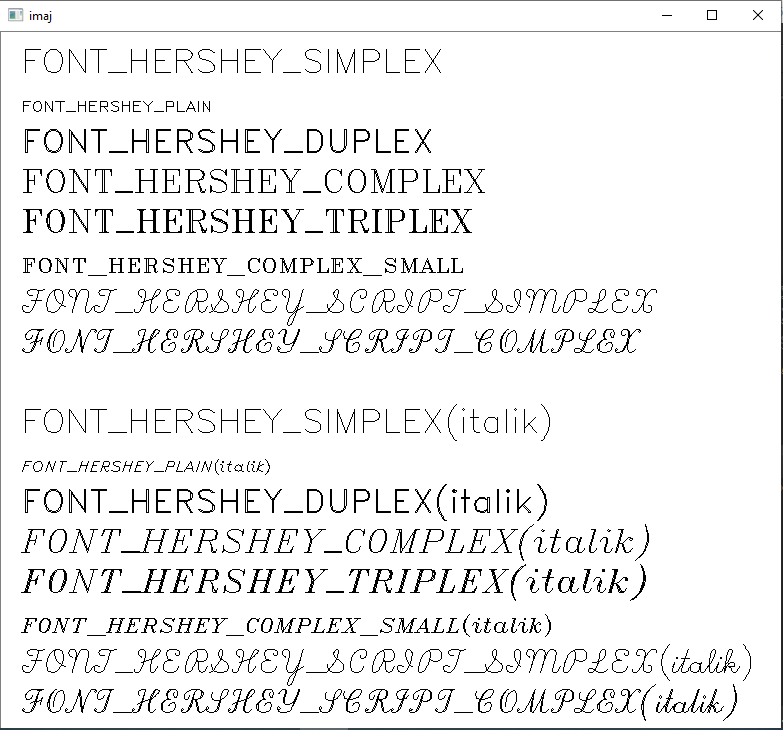
cv2.putText(imaj,fontlar[j]+'(italik)',(20,400+j\*40),j+italik,1.1,(0,0,0),1)

cv2.imshow('imaj',imaj)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı :



Türkçe Karakter Destekli Yazı Yazmak

OpenCV’nin desteklediği fontlar sadece ASCII karakterlerine izin vermekte ama Türkçe karakter yazdırabilmek için pillow kütüphanesi kullanarak bunu çözebiliyoruz.Pillow kütüphanesi açık kaynak kodlu bir grafik işleme kütüphanesidir.Aslında burada yaptığımız pillow kütüphanesiyle yazı şeklinde grafik çizmek bu yüzden Türkçe karakterleri oluşturabiliyoruz.

4-Utf8-DestekliYaziYazmak.py

import cv2

import numpy as np

from PIL import Image, ImageDraw, ImageFont

*# Türkçe karakterleri de yazabilmek icin pillow kutuphanesini kullaniyoruz*

*# pillow kutuphanesi bir goruntu isleme kutuphanesidir.Python da goruntu islemleri yapilabilmesi icin olusturulmustur.*

def print\_utf8\_text(image, text, fontName='DejaVuSerif',

color=(0,0,0), yer=(10,10), boy=48):

font = ImageFont.truetype(fontName, boy) *# font adi ve boyutu*

pilImg = Image.fromarray(image) *# imajı pillow moduna çevir*

*#Image opencv de BGR renk uzayini kullaniyor ancak PIL RGB uzayini kullandigi icin onun cevirimini yapiyoruz.*

draw = ImageDraw.Draw(pilImg) *# imajı hazırla*

*#ImageDraw basit iki boyutlu grafik olusturulmasini saglar*

*#burada yapilan pilimg'i cizmek*

draw.text((yer[0], yer[1]), text,

fill=(color[0], color[1], color[2],0),font = font)

image = np.array(pilImg) *# resmi tekrar opencv'nin kullanabilecegi moda (BGR) ceviriyoruz*

return image

*'''*

*PIL.ImageDraw.Draw.text(xy, text, fill=None, font=None)*

*Parameters:*

*xy –metnin sol ust kosesi*

*text – cizilecek metin*

*fill – metin icin kullanilacak renk*

*font – ImageFont ornegi*

*'''*

if \_\_name\_\_ == "\_\_main\_\_":

imaj = np.ones((400, 700, 3), dtype=np.uint8) \* 255*#beyaz bir zemin olusturak icin*

fontName='verdana.ttf'; renk = (0, 0, 0); yer = (10,10); boy = 64*#sirayla deger atama islemleri.*

*#ayni satirda yazdigimiz icin aralara ; koyduk*

imaj = print\_utf8\_text(imaj, "Ağaç ", fontName,

renk, yer, boy)

*#fonksiyonumuzun icerisine parametrelerimizi gonderiyoruz*

*# ve ayni islemleri farkli yazilar yazmak icin tekrarliyoruz*

fontName = 'tahoma.ttf'

renk = (0, 0, 255); yer = (100,120); boy = 36

imaj = print\_utf8\_text(imaj, "Tuğrul Şahin Kök ", fontName,

renk, yer, boy)

fontName = 'times.ttf'; renk = (128, 0, 0)

yer = (30,200); boy = 48

imaj = print\_utf8\_text(imaj,

"LACİVERT ", fontName, renk, yer, boy)

fontName = 'times.ttf'; renk = (0, 255, 0)

yer = (10,300); boy = 42

imaj = print\_utf8\_text(imaj, "Ömer ", fontName, renk, yer, boy)

cv2.imshow('beyaz fon', imaj)

cv2.moveWindow('beyaz fon', 10, 10)*# pencerenin x,y duzleminde nerede acilacagini belirtir*

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Görseller ve Matris İşlemleri

OpenCV’de resimler birer matris olarak görülür.Matris işlemleri numpy kütüphanesi ile yapılır.Oluşturacağımız görselleri matris ile oluşturur

5-bos\_resim.py

import numpy as np

import cv2

deneme = np.zeros((400,400,3),dtype=np.uint8)

*#deneme[:] = (255,255,255) beyaz piksel degerleri*

*#deneme[:] = (0,0,255) kirmizi piksel degerleri*

*#400x400 piksel boyutlarinda 3 renk kanalina sahip bir siyah pencere olusturur.*

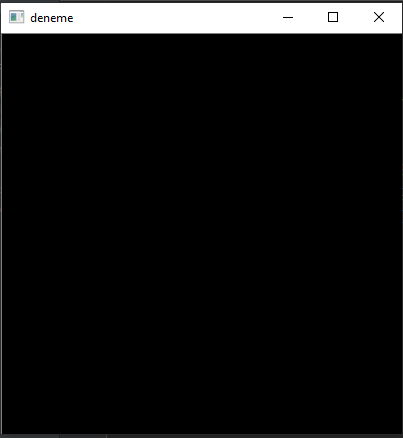
*#bu resmi bir numpy matrisiyle olusturup icerisini sifir ile doldurduk.(np.zeros)*

cv2.imshow('deneme',deneme)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Bu 400x400 piksellik 3 renk kanalına sahip siyah bir penceredir.Bu görüntüyü bir numpy matrisiyle oluşturup içerisini 0(sıfır)’larla doldurduk.(np.zeros)

Bu siyah görüntüyü eğer beyaza çevirmek istersek o zaman görselimizin her bir pikselindeki renk bilgisini beyaza karşılık gelen (255,255,255) demetiyle doldurmamız gerekir.

|  |  |
| --- | --- |
| Beyaz bir kare: | Kırmızı bir kare: |
| C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\beyaz.png |  |

Renk Kanalları

BGR renk uzayına sahip olan OpenCV’de görüntümüzün 3.boyutu renk kanallarını tutar.Herhangi bir resmi oluşturmak için bu renk kanallarının birlikte kullanılması gerekmektedir.Renk kanallarını birer filtre olarak düşünürsek 3 tane renk kanalımız(BGR) üst üste gelince resmimizi renkli biçimde görürüz.Bu renk kanallarını ayrı ayrı görüntülemekte mümkün.Yani resmimizi mavi , yeşil ve kırmızı filtreler ile gösterebilir.Bunu yapmak için bir renk kanalını göstermek için diğer renk kanallarının tüm değerlerini sıfırlamak olacaktır.

6-renk\_kanallari.py

import cv2

deltax=0

deltay=0

img = cv2.imread('../Resimler/monalisa.jpg')

m = img.copy()

m[:,:,1]=0

m[:,:,2]=0

*# mavi renk filtresini elde etmek icin yesil ve kirmizi renk kanallarini sifira esitliyoruz*

y = img.copy()

y[:,:,0]=0

y[:,:,2]=0

*# Yesil renk filtresini elde etmek icin mavi ve kirmizi renk kanallarini sifira esitliyoruz*

k = img.copy()

k[:,:,0]=0

k[:,:,1]=0

*# Kirmizi renk filtresini elde etmek icin yesil ve kirmizi renk kanallarini sifira esitliyoruz*

print(img.shape)*#resmin boyutlarini verir -> yukseklik, genislik, renk kanallari*

*#renk kanallarinin hepsinin ayni anda olmasi resmimizin orijinal halinin ortaya cikmasini sagliyor*

cv2.imshow("Orijinal",img);cv2.moveWindow('Orijinal',10,10)

*'''*

*Yükseklik, genişlik ve kanal sayısına img.shape*

*ile erişebiliriz: Yükseklik indeks 0'da,*

*Genişlik indeks 1'de; ve indeks 2'deki kanal sayısı.*

*'''*

cv2.imshow('MAVi',m)

cv2.moveWindow('Mavi',10,img.shape[0]+deltay)

*#moveWindow(x = 10,ya = yukselik + deltay)*

cv2.imshow('Kirmizi',k)

cv2.moveWindow('Kirmizi',img.shape[1]+deltax,10)

*#moveWindow(x = genislik + deltax , y = 10)*

cv2.imshow('Yesil',y)

cv2.moveWindow('Yesil',img.shape[1]+deltax,img.shape[0]+deltay)

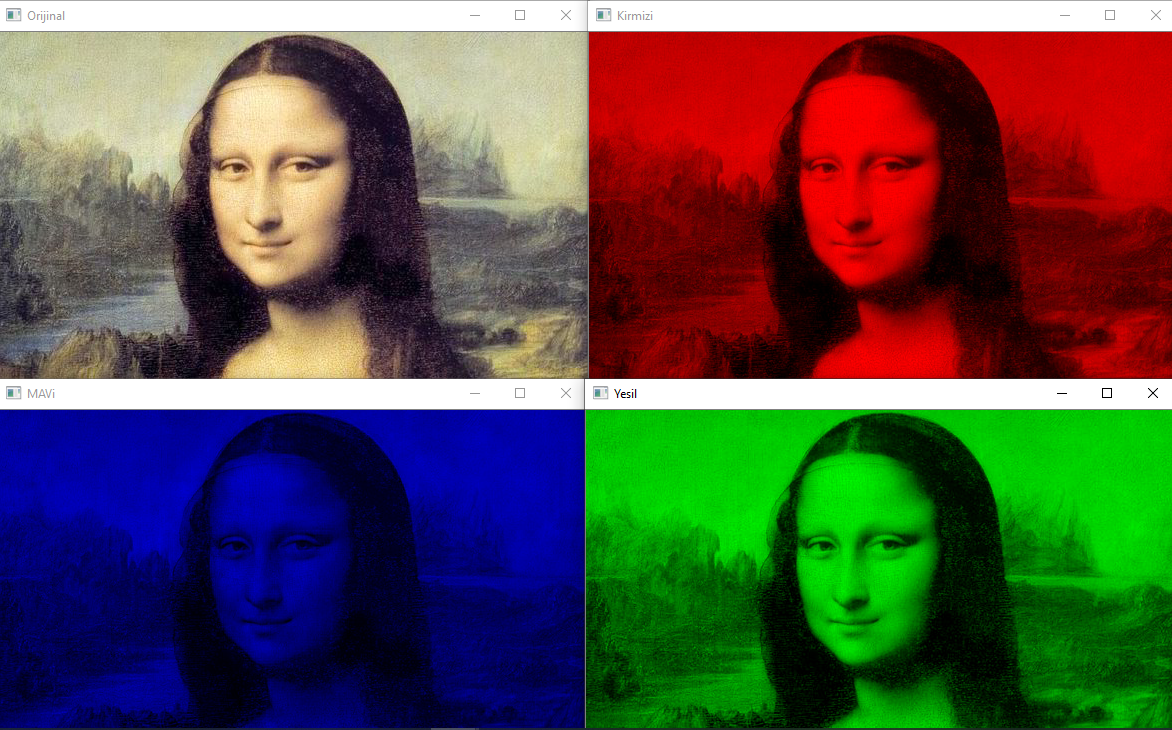
*#moveWindow(x = genislik + deltax , yukseklik + deltay)*

*#moveWindow() ile pencerelerin acilma konumlarini ayarliyoruz*

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Yeniden Boyutlandırmak

OpenCV kullanırken bazen resimleri ihtiyacımız çerçevesinde yeniden boyutlandırmamız gerekebilir.Bunu en-boy oranını koruyarak yapabileceğimiz gibi bu bir zorunluluk değildir.Resimlerimizi istediğimiz boyutlarda tekrar boyutlandırabiliriz.

7-yeniden\_boyutlama.py

import cv2

import random

imaj = cv2.imread('../Resimler/sardunya2.jpg')

cv2.imshow('imaj',imaj)

oran = 0.8

imajlar=[]*#resimleri bir liste icerisinde tutucaz*

for j in range(10):

oran = random.randint(1,25)/25 *# 0 ile 1 arasinda bir sayi olusturur*

rx = int(imaj.shape[1]\*oran) *#Yeni imajin genisligi*

ry = int(imaj.shape[0]\*oran) *#yeni imajin yukseligi*

x = random.randint(100,1600)*#100-1600 arasinda rastgele x koordinati*

y = random.randint(100,800)*#100-800 arasinda rastgele y koordinati*

imajlar.append((str(oran),cv2.resize(imaj,(rx,ry))))

*#listeye ekleme fonksiyonu, cv2.resize(orjinal\_imaj,(yeni\_yukseklik,yeni\_genislik)*

cv2.imshow(imajlar[j][0],imajlar[j][1])

*#pencere ismi olarak kullanilan orani ve yeni olusturan imajin kendisini kullaniyoruz*

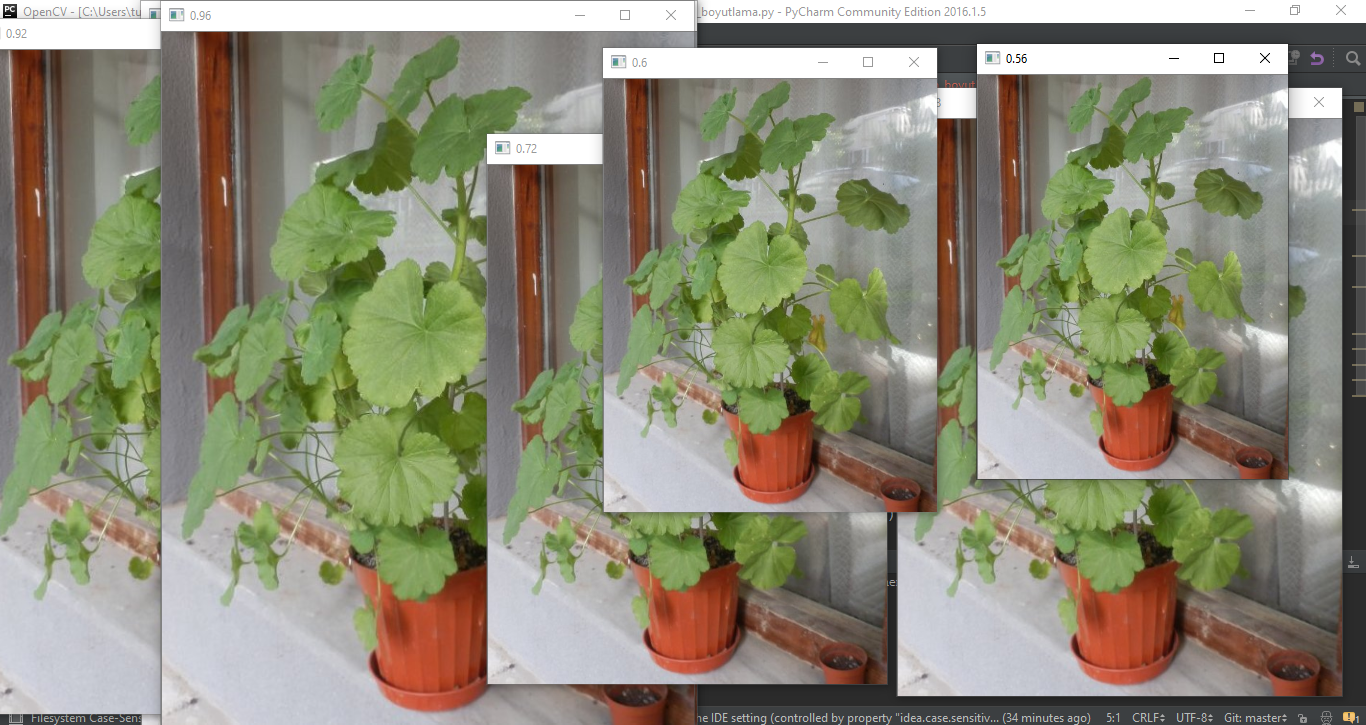
cv2.moveWindow(imajlar[j][0],x,y)

*#bir ust satirda kullanidigimiz isimdeki pencerenin x ve y random komutlarini aliyoruz*

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Maskeleme

Maskeleme resim birleştirme işlemlerinde kullanılan bir araçtır.Maske siyah ve beyazdan oluşan bir görüntüdür.Maske siyah ve beyazdan oluşan bir görüntüdür.Uygulama işlemi gerçekleştirildiğinde beyaz olan pikseller işleme sokulurken siyah olan pikseller işleme alınmazlar.

Örneğimiz de monalisa.jpg dosyamızın maskesini yarısı beyaz diğer yarısı siyah olacak şekilde oluşturduk.Resmimizi ve maskemizi bitwise\_and() komutu ile işleme soktuğumuzda resmimizin yarısı monalisa diğer yarısı ise siyah piksellerden oluşturduk.

8-maskeleme.py

import cv2

import numpy as np

deltax = 0

deltay = 0

img = cv2.imread("../Resimler/monalisa.jpg")

maske = np.ones(img.shape,dtype="uint8")\*255 *#beyaz bir resim olusturduk*

*#img.shape resmin boyutu kadar bir pencere olusturmamizi sagliyor*

maske[:,260:] = [0,0,0]*#resmin ortasina kadar siyah alan olusturmak*

maskeli = cv2.bitwise\_and(img,maske)*#maskenin siyah bolgeleri siyah kalir beyaz kisimlari diger resim ile doldurulcak*

*#cv2.bitwise\_and(ilk girdi dizisi,ikinci girdi dizisi)*

*#ilk parametre okuttugumuz resmimiz ikincisi ise olusturdugumuz siyah maske*

*#bitwise\_and iki resmin bitsel birlesimini saglar*

cv2.imshow('img',img)*#resim gosterme islemi*

cv2.moveWindow('img',10,10)*#resmin nerede cikagi*

cv2.imshow('maske',maske)*#maske gosterimi*

cv2.moveWindow('maske',img.shape[1]+deltax,10)*#resmin nerede cikacagi*

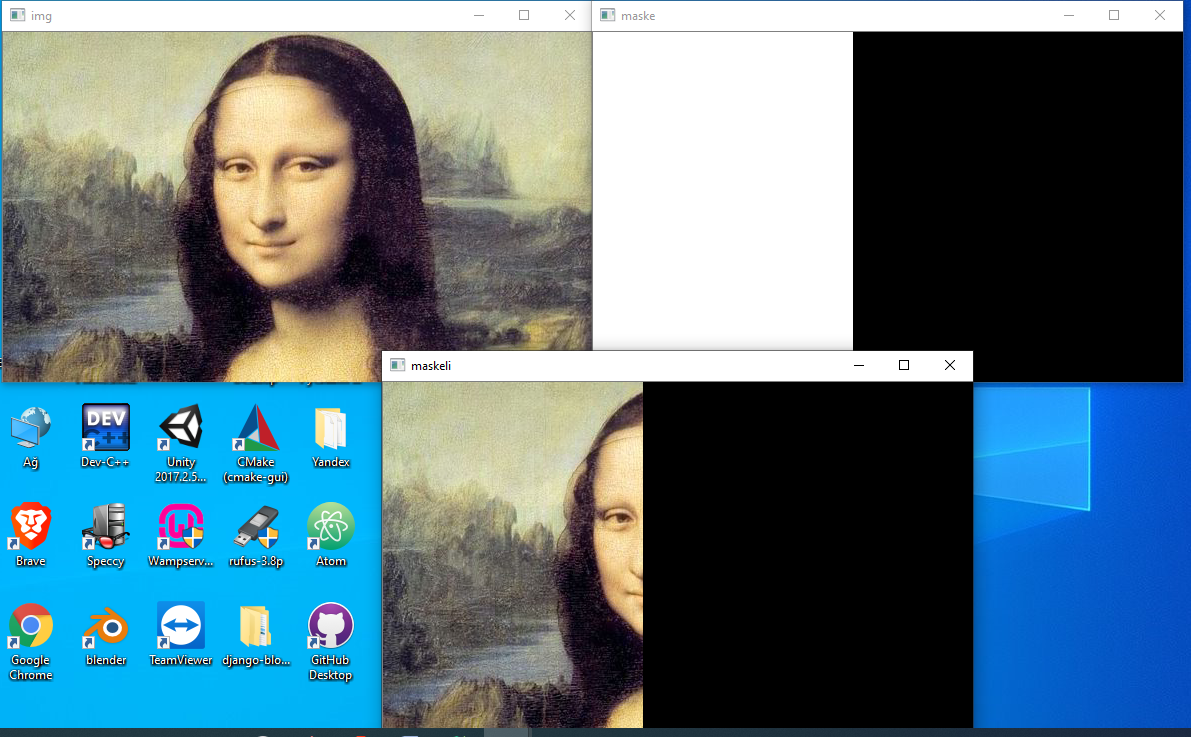
cv2.imshow('maskeli',maskeli)*#maske gosterimi*

cv2.moveWindow('maskeli',380,img.shape[0]+deltay)*#maskeli resmin nerede cikacagi*

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Resim Ekleme

OpenCV’de herhangi bir resim üzerine başka bir resim ekleme işlemi yapabiliyoruz.Uygulama olarak yapmak istediğimiz maske ve ters maske oluşturarak arkplan resmimizin ortasına monalisa.jpg dosyamızı oturtmak.Kodlarımızı inceledikten sonra bunu adım adım gözlemleyelim.

9-resim\_ekleme.py

import cv2

import numpy as np

bekle = True

*#tusa bastigimizda false 'a doner*

arkaplan = np.ones((800,800,3),dtype=np.uint8)\*255*#800x800 boyutlarinda beyaz arkaplan*

if bekle:

cv2.imshow('arkaplan',arkaplan)*#arkaplani goster*

cv2.waitKey(0)

*#Arkaplanin ortasina 360piksel yaricapli siyah bir daire cizimi*

maske = cv2.circle(arkaplan,(400,400),360,(0,0,0),-1)

*#beyaz arkaplanin uzerine 400x400 piksel ve capi 360px olan bir siyah cember cizdiriyoruz*

if bekle:

cv2.imshow('maske',maske)

cv2.waitKey(0)

*#ters maske islemi*

tersmaske = (255-maske)*#bu islem maskedeki 0'lari 255'e , 255'leri 0'a cevirir yani beyazlar siyah,siyahlar beyaz olur*

if bekle:

cv2.imshow('tersmaske',tersmaske)

cv2.waitKey(0)

*#resim yukleme*

img = cv2.imread('../Resimler/monalisa(800x800).jpg')

if bekle:

cv2.imshow('img',img)

cv2.waitKey(0)

*#ters maskeyi img ye uygula*

img = cv2.bitwise\_and(img,tersmaske)*#resim uzerine tersmaskeyi uyguluyoruz*

*#resmin orta kisminda monalisa (maskenin beyaz yerleri) dis kisminda siyah alan olusacak*

if bekle:

cv2.imshow('img',img)

cv2.waitKey(0)

*#diger resmi yukle ve maske uygula*

img2 = cv2.imread('../Resimler/uzay(800x800).jpg')

if bekle:

cv2.imshow('img2',img2)

cv2.waitKey(0)

img2 = cv2.bitwise\_and(img2,maske)*#uzay resmimize maskeyi uyguluyoruz*

*#resmin ortasi siyah kenarlari beyaz oluyor.Beyaz(dis) kisma uzay resmimiz gelmis oluyor.*

if bekle:

cv2.imshow('img2',img2)

cv2.waitKey(0)

*#iki resim birlestirme islemi*

img3 = img + img2*#maske ve tersmaske uygulanmis resimleri birlestiriyoruz*

cv2.imshow('img3',img3);cv2.waitKey(0)

cv2.destroyAllWindows()

*'''*

*İşlem Asamalari:*

*1-arkaplan olusacak*

*2-maske olusacak*

*3-tersmaske olusacak*

*4-monalisa resmini yukledik*

*5-monalisaya ters maske uyguladik*

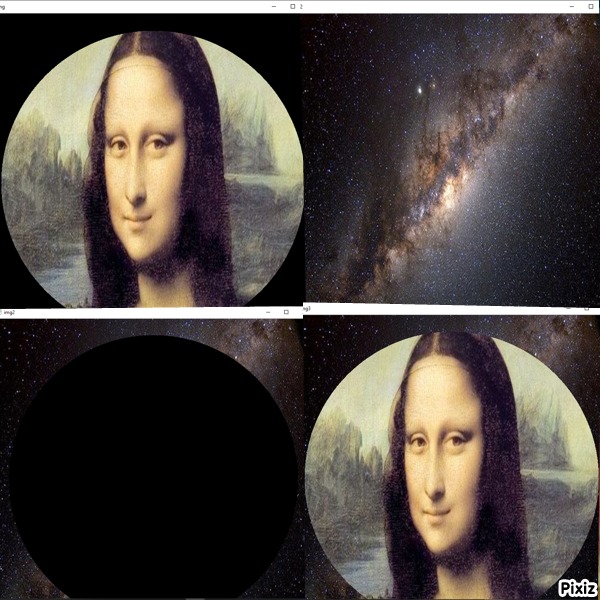
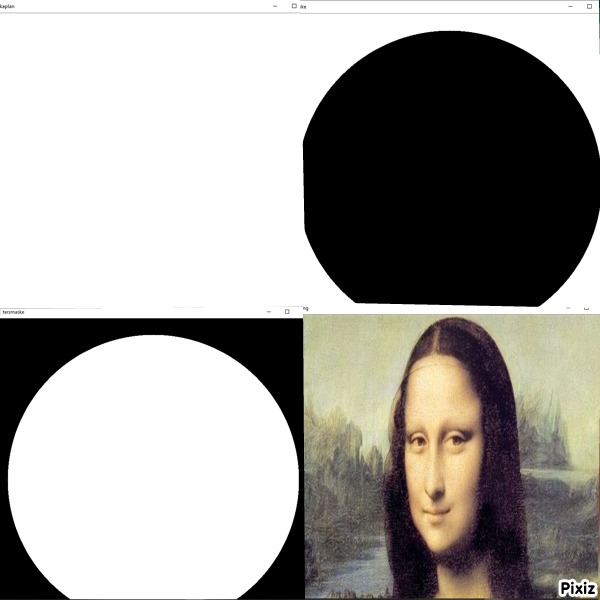
*6-uzay resmini yukledik*

*7-uzay resmine maske uyguladik*

*8-son olarak iki maske uygulanmis goruntuyu birlestirme islemini gerceklestirdik*

*'''*

Program Çıktısı:



Görsellerdeki Keskinlikleri Yumuşatma (Blurring)

“Blurring” terimi bir resmin bulanıklaştırılmasını, yumuşatılmasını ifade eder.”Blurring” işlemi bir resme uygulandığında resmin keskin hatları, kenarları belirginliğini kaybeder.Aynı zamanda detaylar baskın renklerin içerisinde eriyerek kaybolur.Görüntülerin gürültülerini(detaylarını) gidermede etkin bir yoldur.

İhtiyaç durumuna göre birçok bulanıklaştırma(yumuşatma) yöntemi vardır.

* cv2.filter2D(imaj,derinlik,kernel)
* cv2.blur(imaj,kernel)
* cv2.GaussianBlur(imaj,kernel,sigma)
* cv2.medianBlur(imaj,kernel\_boyutu)
* cv2.bilateralFilter(imaj,komsuluk\_capi,sigma\_renk,sigma\_uzay)

**cv2.filter2D(imaj,derinlik,kernel)**

Bu yöntem, resim matrisindeki değerlerin çekirdek (kernel) boyutlarına göre ortalamalarını bulurak çalışır.Örneğin aşağıdaki örnekte 11x11 boyutlu bir kernel(çekirdek) kullanıyoruz. Bu nedenle her pikselin rengi, içinde bulunduğu 11x11’lik çekirdek matrisin içerisinde yer alan ve birbirine komşu olan 121 pikselin renk ortalamasıyla değiştirilir. Dolayısıyla renkler keskinliğini kaybederek ortalama değerlere doğru kayar.Kernel boyutları büyüdükçe bulanıklık da yoğunlaşır.

10-blurring\_filter2D.py

import cv2

import numpy as np

deltax = 0

deltay = 0

imaj = cv2.imread('../Resimler/pou400.png')

n = 11

kernel = np.ones((n,n),np.float32) / (n\*n\*1.0) *#matris isleme girdiginde en fazla 0-1 arasinda deger almasi saglaniyor*

*#(kaynak, derinlik, kernel, capa, delta, sinirtipi)*

blur = cv2.filter2D(imaj,-1,kernel)

*#-1 orjinal resmin boyutlarina gore islem yapilmasini saglar*

cv2.imshow('imaj',imaj)

cv2.imshow('filter2D',blur)

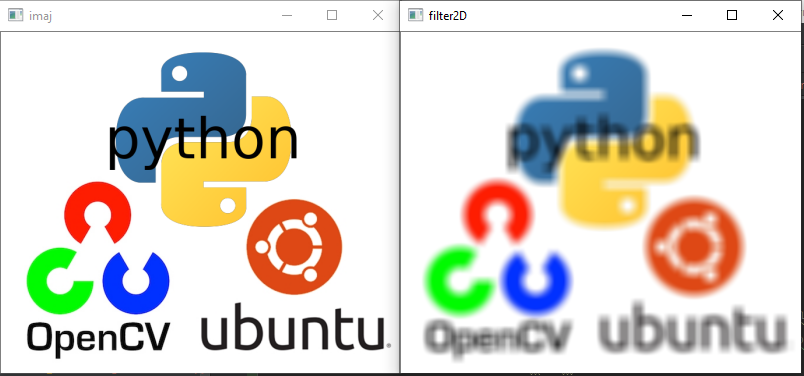
cv2.moveWindow('imaj',10,10)

cv2.moveWindow('filter2D',imaj.shape[1]+deltax,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



**cv2.blur(imaj,kernel)**

Bu yöntemde her piksel değerinin komşu piksel renklerinin ortalama değeriyle değiştirilmesi esasına dayanır.

11-bluring\_blur.py

import cv2

deltax = 0

deltay = 0

imaj = cv2.imread('../Resimler/pou400.png')

*# (imaj, kernel)*

blur = cv2.blur(imaj,(5,5))

*#kernel (cekirdek) boyutu arttilirdikca detaylarin kaybolmasi durumu artar*

cv2.imshow('imaj',imaj)

cv2.imshow('blur',blur)

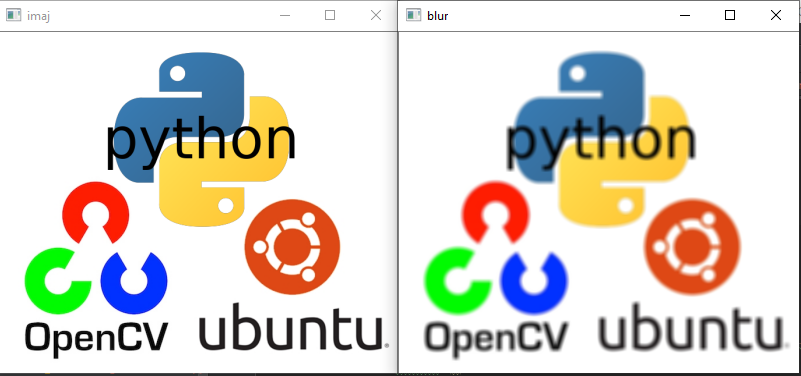
cv2.moveWindow('imaj',10,10)

cv2.moveWindow('blur',imaj.shape[1]+deltax,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



**cv2.GaussianBlur(imaj,kernel,sigma)**

GaussianBlur filtresi görüntü üzerinde düzleştirme işlemi uygular.

12-blurring\_GaussianBlur.py

import cv2

deltax = 0

deltay = 0

imaj = cv2.imread('../Resimler/pou400.png')

*#(imaj,kernel,standart sapma)*

blur = cv2.GaussianBlur(imaj,(11,11),0) *#pozitif tek sayi*

cv2.imshow('imaj',imaj)

cv2.imshow('GaussianBlur',blur)

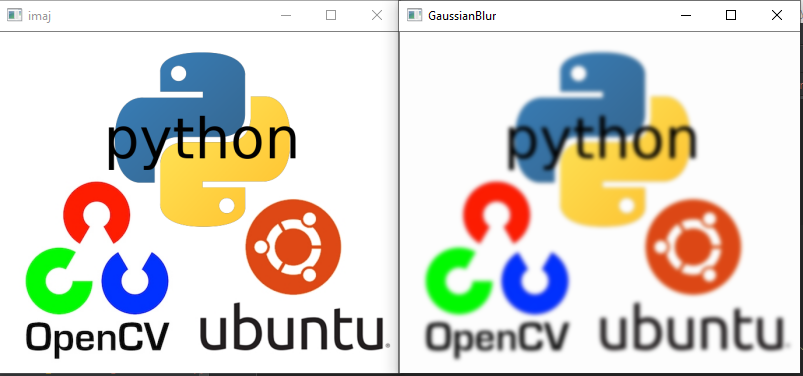
cv2.moveWindow('imaj',10,10)

cv2.moveWindow('GaussianBlur',imaj.shape[1]+deltax,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



**cv2.medianBlur(imaj,kernel\_boyutu)**

Fonksiyon çekirdek alanının altındaki tüm piksellerin medyanını alır ve merkezi eleman bu medyanın değerleriyle değiştirilir.

13-blurring\_medianBlur.py

import cv2

deltax = 0

deltay = 0

imaj = cv2.imread('../Resimler/pou400.png')

*# (imaj,kernel\_boyutu)*

blur = cv2.medianBlur(imaj,11)*# tek sayi olmak zorunda*

cv2.imshow('imaj',imaj)

cv2.imshow('medianBlur',blur)

cv2.moveWindow('imaj',10,10)

cv2.moveWindow('medianBlur',blur.shape[1]+deltax,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



**cv2.bilateralFilter(imaj,komsuluk\_capi,sigma\_renk,sigma\_uzay)**

Uyguladığımız filtreler resimler üzerinde bazı bozulmalara sebep olabilir. Örneğin GaussianBlur işleminde resmin piksellerini ortadaki piksellerin ortalama renklerine göre yumuşatma işlemi yaptığı için resmin kenarlarında bozulmalara yol açabilir. Ancak bilateralFilter işlemi bunu önler resim içerisinde kenarları korur.

14-blurring\_bilateralFilter.py

import cv2

deltax = 0

deltay = 0

imaj = cv2.imread('../Resimler/pou400.png')

*# (imaj,komsuluk\_capi,sigma\_renk,sigma\_uzay)*

blur = cv2.bilateralFilter(imaj,11,175,175)

cv2.imshow('imaj',imaj)

cv2.imshow('bilateralFilter',blur)

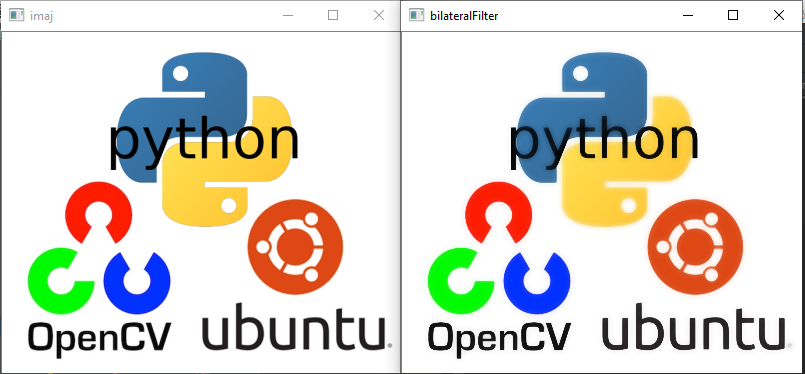
cv2.moveWindow('imaj',10,10)

cv2.moveWindow('bilateralFilter',blur.shape[1]+deltax,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



Görüntüleri Daraltmak ve Genişletmek

Siyah beyaz binary bir resmin ön plan(beyaz) görüntülerini daraltmak veya genişletmek için kullanabileceğimiz 2 tane fonksiyonumuz var: cv2.erode() ve cv2.dilate()

**cv2.erode():**

**cv2.erode(imaj, kernel, iterasyon)** parametrelerini alır.Burada imaj siyah beyaz görüntümüzdür.Kernel çekirdeğimizi ifade eder.İterasyon inceltme işleminin kaç defa tekrar edeceğini ifade eder.Ön plan beyaz görüntümüzün sınırları verilen çekirdek boyutuna aşındırılır.Görüntünün siyah beyaz olması önemlidir.Çekirdek alandaki siyah pikseller beyaza, beyaz pikseller ise siyah piksele dönüştürülme işlemi yapılır.Bu fonksiyon beyaz görüntünün gürültülsünün giderilme işlemini yapar.

**cv2.dilate():**

**cv2.dilate(imaj, kernel, iterasyon).**Parameterini alır.İmaj siyah beyaz resmi ifade eder.Kernel çekirdek boyutumuzdur.İterasyon ise kalınlaştırma işleminin kaç defa gerçekleştirileceğini ifade eder.Fonksiyon çekirdek içerisinde bir tane bile beyaz piksel varsa geriye beyaz, aksi halde siyah döndürür.Dolayısıyla beyaz görüntünün sınırları genişletilmiş olur.

Erode ve dilate işlemleri ard arda uygulanırsa sınır çizgileri daha düzgün bir hale gelir.

15-erode\_dilate.py

import cv2

import numpy as np

imaj = cv2.imread('../Resimler/1.png',0)

kernel = np.ones((5,5),np.uint8)

*#sinirlari inceltme*

erosion = cv2.erode(imaj, kernel, iterations = 1)

*#sinirlari kalinlastirma*

dilation = cv2.dilate(imaj, kernel, iterations = 1)

*'''*

*1-imaj : siyah beyaz resim*

*2-kernel : cekirdek*

*3-iterations : inceltme ve kalinlastirma isleminin tekrarlanma sayisi*

*'''*

*#uc goruntuyu dusey olarak birlestirme*

dusey = np.vstack((imaj, erosion, dilation))

*#vstack 3 resmi alt alta birlestirme yapar*

cv2.imshow('dusey',dusey)

cv2.moveWindow('dusey',600,50)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Program Çıktısı :**



**Sınır Çizgilerinin (Konturların) Belirlenmesi ve İşlenmesi**

Sınır çizgileri sınır çizgilerindeki aynı renk veya yoğunluğa sahip noktaları birleştiren eğrilerdir.Konturlar biçim analizi,nesne saptama ve tanıma işlemlerinde kullanılır.

Kontur belirlemede siyah beyaz görüntüleri kullanmak daha iyi sonuçlar vermektedir.Bu yüzden treshold veya canny uygulamalarından sonra konturların bulunması daha kolay olur.

OpenCV açısından bulunacak nesnenin beyaz arkplanın siyah renkte olması gerekmektedir.

16-esikler.py

import cv2

deltax = 0

deltay = 0

sthrs = ['cv2.THRESH\_BINARY','cv2.THRESH\_BINARY\_INV',

'cv2.THRESH\_TRUNC', 'cv2.THRESH\_TOZERO',

'cv2.THRESH\_TOZERO\_INV', 'cv2.THRESH\_MASK',

'cv2.THRESH\_OTSU','cv2.THRESH\_TRIANGLE']

thrs = [cv2.THRESH\_BINARY, cv2.THRESH\_BINARY\_INV, cv2.THRESH\_TRUNC,

cv2.THRESH\_TOZERO, cv2.THRESH\_TOZERO\_INV, cv2.THRESH\_MASK,

cv2.THRESH\_OTSU, cv2.THRESH\_TRIANGLE]

print(thrs)

gri = cv2.imread('../Resimler/gradient.jpg',cv2.IMREAD\_GRAYSCALE)

gri = cv2.resize(gri,(int(gri.shape[1]/gri.shape[0]\*400),400))

*#*

x = 10; y = 10

cv2.imshow('gri',gri)

cv2.moveWindow('gri',x,y)

x = 45;

for j in range(len(thrs)):

*#burada yapilan islem acilan pencereler ekran boyutuna ulastiginda alt satira gecmesini saglamak*

x += gri.shape[1] + deltax

if x>1600: *#1600 ekran genisligi*

x = 45;y += gri.shape[0] + deltay

ret,esik = cv2.threshold(gri,128, 255,thrs[j])

*#(gri\_imaj, esik\_degeri, max\_degeri, esik\_tipi)*

*#esik\_degeri : 0 - 255*

*#max\_deger : 0 - 255*

*#geriye iki deger dondurur birincisi esik degeri ikinicisi yeni olusan esik goruntusudur.*

*#ret : cv2.THRESH\_OTSU kullanilirsa hesaplanan esik degeri, yoksa verilen esik degeri*

*'''*

*Parameters*

*src : giriş dizisi (çok kanallı, 8 bit veya 32 bit kayma noktasi).*

*thresh : esik degeri.*

*maxval : THRESH\_BINARY ve THRESH\_BINARY\_INV ile kullanilacak esikleme degeri.*

*type : esik turu.*

*'''*

print('ret = ',ret)

cv2.imshow(sthrs[j],esik)

cv2.moveWindow(sthrs[j],x,y)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.png | C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\6.png |
| C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\7.png | C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\8.png |
| C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\9.png | C:\Users\tugko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\10.png |

**Sınır Çizgileri**

**cv2.findContours():**

Konturlar şekil analizi, nesne algılama ve tanıma için kullanışlı bir araçtır.

cv2.findContours(orijinal imaj, mod, metod) parametrelerini alır.

Mod: Kontur bulma yöntemini ifade eder.

Metod: Kontur yaklaşım algoritmasını ifade eder

cv2.drawContours():

Kontür hatları veya dolgulu konturlar çizer.

Cv2.drawContours(image ,contours,contourIdx, color, thickness)

Parametreleri:

image – Hedef goruntu.

contours – Tüm giriş konturları. Her kontur bir nokta vektörü olarak saklanır.

contourIdx – Çizilecek konturu gösteren parametre. Negatifse, tüm konturlar çizilir.

color - kontur rengi

thickness – konturlarin cizgi kalinligi

17-kontur01.py

import cv2

import numpy as np

imaj = cv2.imread('../Resimler/python.png')

*# resmin boyutunu buyutme*

y = imaj.shape[0]\*2

x = imaj.shape[1]\*2

imaj = cv2.resize(imaj,(x,y))

gri = cv2.cvtColor(imaj,cv2.COLOR\_BGR2GRAY)

*'''*

*cv2.cvtColor()*

*Parameters:*

*src: rengi degistirilecek goruntu*

*code: renk alani donusturme kodu*

*'''*

\_,sb = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)

*#\_ = esik degeri , sb = yeni olusan esik goruntusudur*

konturlar = cv2.findContours(sb,cv2.RETR\_EXTERNAL,

cv2.CHAIN\_APPROX\_SIMPLE)[-2]

*# cv2.findContours(orjinal imaj = imaj, mod = cv2.RETR\_EXTERNAL, metod = cv2.CHAIN\_APPROX\_SIMPLE)*

*#mod : kontur bulma yontemi (RETR\_LIST,RETR\_EXTERNAL,RETR\_CCOMP,RETR\_TREE)*

*#metod : kontur yaklasim yontemi (approximation)*

imaj2 = cv2.drawContours(imaj.copy(),konturlar,-1,(0,0,255),2)

*'''*

*cv2.drawContours()*

*Parameters:*

*image – Hedef goruntu.*

*contours – Tüm giriş konturları. Her kontur bir nokta vektörü olarak saklanır.*

*contourIdx – Çizilecek konturu gösteren parametre. Negatifse, tüm konturlar çizilir.*

*color - kontur rengi*

*thickness – konturlarin cizgi kalinligi*

*'''*

*#iki resmi dusey olarak birlestirme*

imaj3 = np.vstack((imaj,imaj2))

cv2.imshow('imaj3',imaj3)

cv2.moveWindow('imaj3',10,10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:



cv2.Canny()

Canny algoritmasını kullanarak görüntüdeki kenarları bulur. Fonksiyon, giriş görüntüsünde kenarları bulur ve Canny algoritmasını kullanarak bunları çıktı haritası kenarlarında işaretler. Eşik1 ve eşik2 arasındaki en küçük değer kenar bağlantısı için kullanılır. En büyük değer, güçlü kenarların başlangıç ​​segmentlerini bulmak için kullanılır

18-kenar01.py

import cv2

*#Nesne kenarlarinin saptanmasi*

deltax = 0

deltay = 0

imaj1 = cv2.imread('../Resimler/bina.jpg')

imaj = cv2.resize(imaj1,(500,500))

kenarlar = cv2.Canny(imaj, 50, 150)

*#geri dondurulen kenar haritasi*

*'''*

*cv2.Canny()*

*Parameters:*

*image – tek kanalli 8 bit giris goruntusu.*

*threshold1 – kesiklik islemi icin ilk esik*

*threshold2– kesiklik islemi icin ikinci esik*

*'''*

maske = cv2.bitwise\_not(kenarlar)

*# bitwise\_not() Bir dizinin her bitini tersine çevirir.*

*# parametresi giris arrayi*

maske = cv2.erode(maske,(5,5),iterations = 2)

*# erode kenarlari inceltme islemi*

*#maske = kenarlar*

imaj2 = cv2.bitwise\_and(imaj,imaj,mask = maske)

*#bitwise\_and iki resmin (maske ve imaj) bitsel birlesimini saglar*

cv2.imshow('imaj',imaj)

cv2.imshow('maske',maske)

cv2.imshow('imaj2',imaj2)

cv2.moveWindow('imaj',10,10)

cv2.moveWindow('maske',imaj.shape[1] + deltax,10)

cv2.moveWindow('imaj2',imaj.shape[1] + maske.shape[1] + deltax, 10)

cv2.waitKey(0)

cv2.destroyAllWindows()

Program Çıktısı:

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

Kamera Görüntülerini Okumak

Opencv’de videodan görüntü yakamak için videoCapture fonksiyonu kullanılır. Bunun için VideoCapture nesnesi oluşturmamız gerekir. Parametresi, aygıt dizini veya bir video dosyasının adı olabilir. Cihaz dizini yalnızca hangi kamerayı belirleyeceğiniz sayıdır. Bu sayılar dahili kamera veya takılı olan tek bir kameramız var ise 0’dır.Dışarıdan ikinci bir kamera takıyorsak bunun değeri 1’dir.

19-kamera\_oku.py

import cv2

def ana():

*# kamera = cv2.VideoCapture(0)*

kamera = cv2.VideoCapture("../Videolar/tomandjerry.mp4")

*#VideoCapture fonksiyonuna istersek video dosyamizi istersekte kamera index imizi verebiliriz.*

*# 0. kamera index i varsayilan kameramizdir.*

while(True):

*#videodan goruntu alma islemi*

ret, kare = kamera.read()

*#goruntu alindimi kontrol et alinmadiysa durdur*

if not ret: break

cv2.imshow('kare',kare) *#goruntuyu gosterme islemi*

cv2.moveWindow('kare',10,10)

cv2.waitKey(0) *#degerini sifir yaparsak kare kare yakalama islemini gerceklestirebiliriz.*

*#cv2.waitKey(25) video oynamaya devam eder*

*#kameranin acilip acilmadigin kontrol eder*

if kamera.isOpened():

*#VideoCapture Sinifini kapatir*

kamera.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == '\_\_main\_\_':

ana()

Program Çıktısı:

|  |  |
| --- | --- |
|  |  |
|  |  |