Bilgisayar Görmesi
OpenCV ile Görüntü İşleme

Dr. Öğr. Üyesi Fatma AKALIN

DERS 1: Python ve OpenCV Kütüphanesi Kullanımı

DERS KONUSU: Python ve OpenCV kütüphanesi kullanımı hakkında bilgi verilecektir

DETAYLI DERS İÇERİĞİ:

-Temel Python Kullanımı

-OpenCV Kütüphanesi Tanıtımı

-OpenCV ile Görüntü İşleme Uygulamaları

DERS 2: Görüntü İşlemede Filtreleme

DERS KONUSU: Görüntü İşlemede Filtreleme

DETAYLI DERS İÇERİĞİ:

-Konvolüsyon İşlemi ve Filtreler

-Görüntü Bulanıklığı (Blur) ve Kenar Belirleme

-Open CV ile Filtreleme İşlemleri

DERS 3: İleri Filtreleme Teknikleri

DERS KONUSU: İleri filtreleme teknikleri hakkında bilgi verilecektir.

DETAYLI DERS İÇERİĞİ:

-Farklı Filtreleme Teknikleri ve Uygulamaları

-OpenCV ile İleri Filtreleme Uygulamaları

DERS 4: Yapay Sinir Ağlarına Giriş ve Bilgisayar Görmesi Uygulamaları

DERS KONUSU: Yapay sinir ağlarına giriş hakkında bilgi verilecektir.

DETAYLI DERS İÇERİĞİ:

-Yapay Sinir Ağlarının Temelleri

-Derin Öğrenme ve Evrişimli Sinir Ağları (CNN)

-Evrişimli Sinir Ağı Mimarisinin Anatomisi

-Nesne Tespit Algoritmaları

Bugün DERS 1 kapsamında ilerleyeceğiz

DERS 1: Python ve OpenCV Kütüphanesi Kullanımı

DERS KONUSU: Python ve OpenCV kütüphanesi kullanımı hakkında bilgi verilecektir

DETAYLI DERS İÇERİĞİ:

-Temel Python Kullanımı

-OpenCV Kütüphanesi Tanıtımı

-OpenCV ile Görüntü İşleme Uygulamaları

Python Dili Hakkında

Python, 1990 yılından bu yana büyük bir topluluk tarafından geliştirilen kıvrak ve dinamik bir dildir. Söz dizimi sadedir ve kolay öğrenilir. Ayrıca, Windows, Linux ve MacOSX gibi pek çok platform üzerinde çalışabilme özelliğine sahiptir. Google, NASA ve YouTube gibi pek çok uluslararası kurum ve şirket Python programlama dilinden yararlanmaktadır.

Bu ders kapsamında Pthon dili, görüntü işleme uygulamalarında gerçekleyeceğimiz senaryoları anlayacak seviyede işlenecektir. Bu bağlamda Python programlama dilinin özünü oluşturan,

-karakter dizileri,

-sayılar,

-listeler,

-demetler,

-sözlükler,

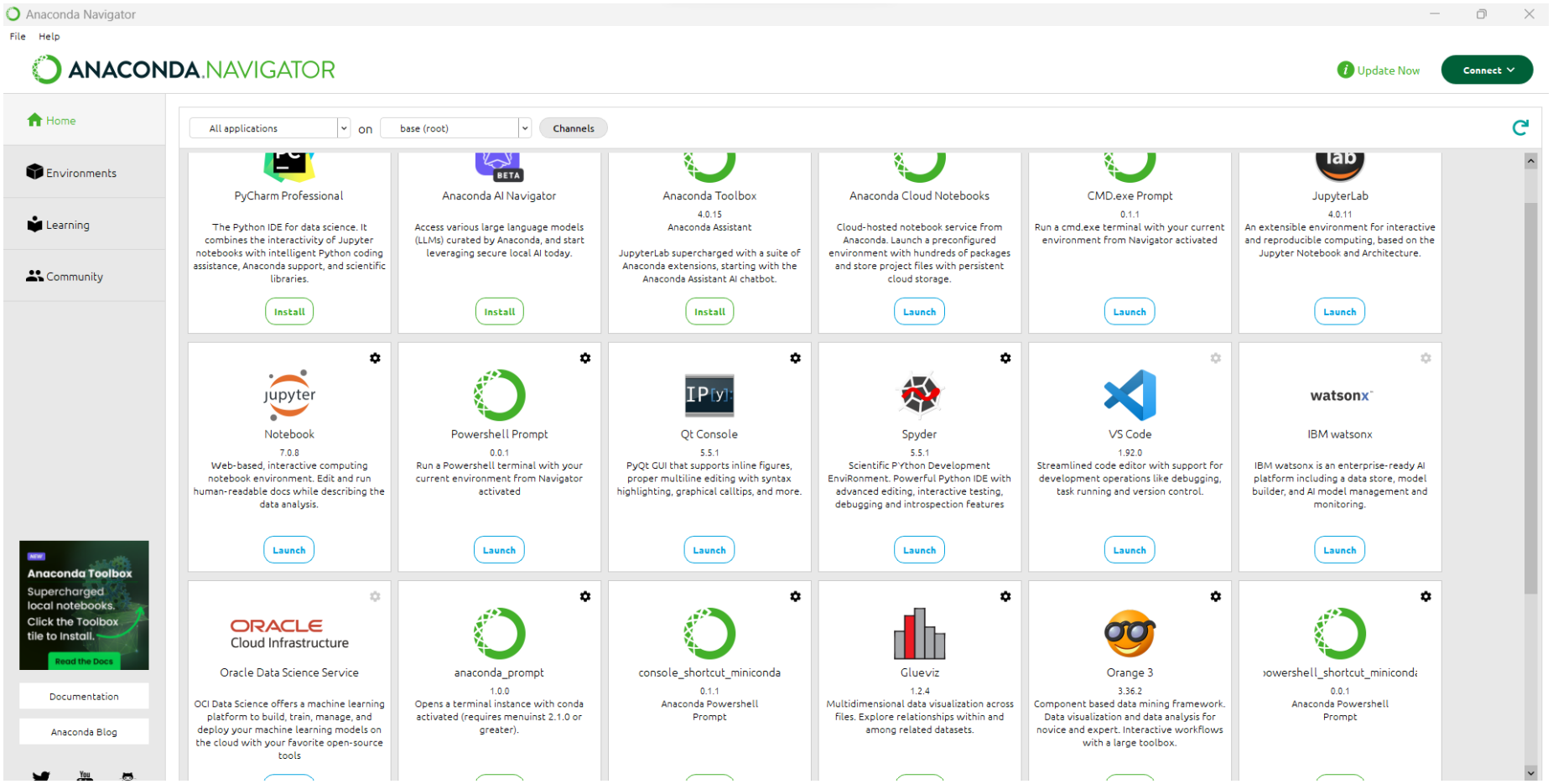
-kümeler,

-fonksiyonlar

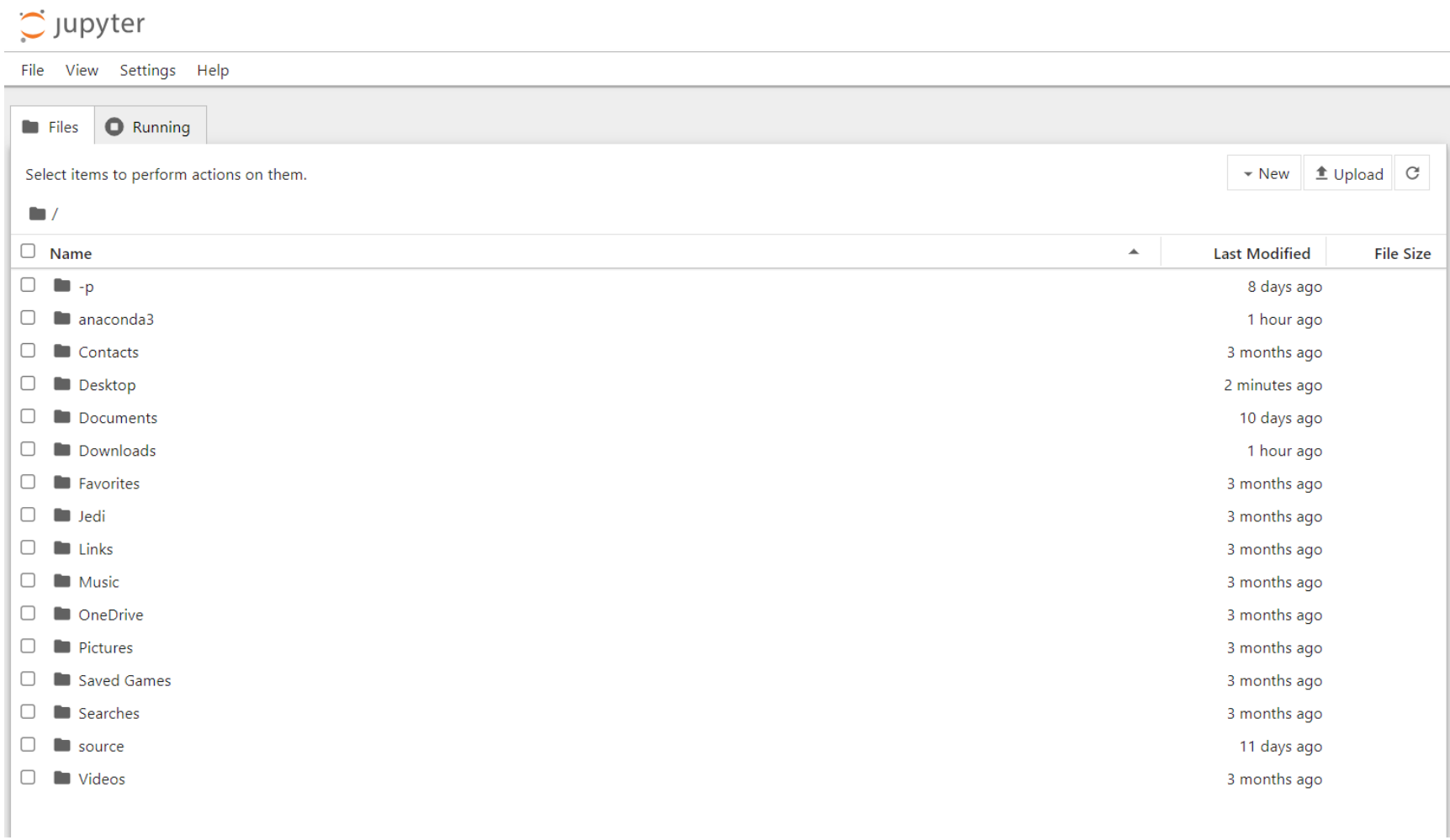
gibi temel düzey konular işlenecektir.

Ders sürecinde bir Python dağıtımı olan Anaconda kullanılacaktır.

Anaconda programı yüklendiğinde farklı kod geliştirme ortamları ile birlikte gelmektedir. Bu ders kapsamında interaktif kod geliştirme ortamı olan Jupyter Notebook kullanılacaktır.



İnteraktif bir kod geliştirme ortamı sunan Jupyter Notebook, kullanıcıların kodu yazıp çalıştırdıkları anda sonuçları anında incelemelerini sağlar. Veri analizi, veri görselleştirme ve makine öğrenmesi projelerinde aktif olarak kullanılır.



TEMEL PYTHON KULLANIMI

Bu kısım python_giriş_seviye.ipynb isimli dosyadan anlatılacaktır.

Fakat slayta da anlatacağım kısımların ekran görüntüsünü ekleyeceğim.

Fatma Akalın

```
Fatma Akalın
```

https://www.w3schools.com/python/python_datatypes.asp

```
Değişken tipleri önemli bir konudur burada yapacağımız örneklerin yanı sıra yukarıdaki linki incelemeliyiz
```

[1]:
```
x = 5
print(type(x))
```

```
<class 'int'>
```

[2]:
```
y=0.5
print(type(y))
```

```
<class 'float'>
```

[3]:
```
z="fatma"
print(type(z))
```

```
<class 'str'>
```

https://www.w3schools.com/python/python_operators.asp

[5]:
```
a = 10
b = 12
c = 0
if a and b and c:
    print("All the numbers have boolean value as True")
else:
    print("Atleast one number has boolean value as False")
```

```
Atleast one number has boolean value as False
```

https://www.w3schools.com/python/python_while_loops.asp

[7]:
```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

https://www.w3schools.com/python/python_for_loops.asp

[8]:
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

https://www.geeksforgeeks.org/python-logical-operators/

[6]:
```
a = 10
if not (a%3 == 0 or a%5 == 0):
    print("10 is not divisible by either 3 or 5")
else:
    print("10 is divisible by either 3 or 5")
```

```
10 is divisible by either 3 or 5
```

https://www.geeksforgeeks.org/python-list-clear-method/

[9]:
```
lis_1 = [1, 3, 5]
lis_2 = [7, 9, 11]
```

[10]:
```
lis_1
```

[10]:
```
[1, 3, 5]
```

[11]:
```
lis_2
```

[11]:
```
[7, 9, 11]
```

[12]:
```
lis_1.clear()
```

[14]:
```
lis_1
```

[14]:
```
[]
```

[13]:
```
del lis_2[:]
```

[15]:
```
lis_2
```

[15]:
```
[]
```

https://www.w3schools.com/python/ref_keyword_del.asp

[16]:
```
x = ["apple", "banana", "cherry"]
del x[0]
print(x)
```

```
['banana', 'cherry']
```

[17]:
```
x=list(range(10))
x
```

[17]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[18]:
```
del x[0]
```

[19]:
```
x
```

[19]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

https://www.w3schools.com/python/python_for_loops.asp

[20]:
```
for x in range(6):
  print(x)
```

```
0
1
2
3
4
5
```

[21]:
```
for x in range(2, 6):
  print(x)
```

```
2
3
4
5
```

[22]:
```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

https://www.w3schools.com/python/python_lists.asp

[23]:
```
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry', 'apple', 'cherry']
```

[24]:
```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

```
Python Collections (Arrays)
There are four collection data types in the Python programming language:

List is a collection which is ordered and changeable. Allows duplicate members.
Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
Dictionary is a collection which is ordered** and changeable. No duplicate members.

*Set items are unchangeable, but you can remove and/or add items whenever you like.

**As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered
```

https://www.w3schools.com/python/python_tuples.asp

```
Ordered
When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
Unchangeable
Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.
Allow Duplicates
Since tuples are indexed, they can have items with the same value:
```

[26]:
```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry', 'apple', 'cherry')
```

[27]:
```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

```
3
```

https://www.w3schools.com/python/python_sets.asp

[28]:
```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'banana', 'apple', 'cherry'}
```

[29]:
```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

```
3
```

https://www.w3schools.com/python/python_dictionaries.asp

[30]:
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

[31]:
```
print(len(thisdict))
```

```
3
```

[32]:
```
thisdict.keys()
```

[32]:
```
dict_keys(['brand', 'model', 'year'])
```

[33]:
```
thisdict.values()
```

[33]:
```
dict_values(['Ford', 'Mustang', 1964])
```

[34]:
```
thisdict.items()
```

[34]:
```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

ÖNEMLİ

[35]:
```
newliste=list(thisdict.items())
```

[36]:
```
newliste
```

[36]:
```
[('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)]
```

https://www.w3schools.com/python/ref_string_join.asp

[39]:
```
maxmin=[10,80,70,90,1]
print(max(maxmin))
```

```
90
```

[40]:
```
print(min(maxmin))
```

```
1
```

[42]:
```
for i in range(len(maxmin)):
    if maxmin[i] == 1:
        maxmin[i] = 7
```

[43]:
```
print(min(maxmin))
```

```
7
```

https://www.w3schools.com/python/python_functions.asp

```
A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.
```

Creating a Function

[44]:
```
def my_function():
  print("Hello from a function")
```

Calling a Function

[45]:
```
my_function()
```

```
Hello from a function
```

Arguments

[46]:
```
def my_function(fname):
  print(fname + " EKLENEN YENİ İSİM FATMA")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil EKLENEN YENİ İSİM FATMA
Tobias EKLENEN YENİ İSİM FATMA
Linus EKLENEN YENİ İSİM FATMA
```

Number of Arguments

[47]:
```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

```
Emil Refsnes
```

Default Parameter Value

[48]:
```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

Passing a List as an Argument

[49]:
```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

```
apple
banana
cherry
```

Return Values

[50]:
```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

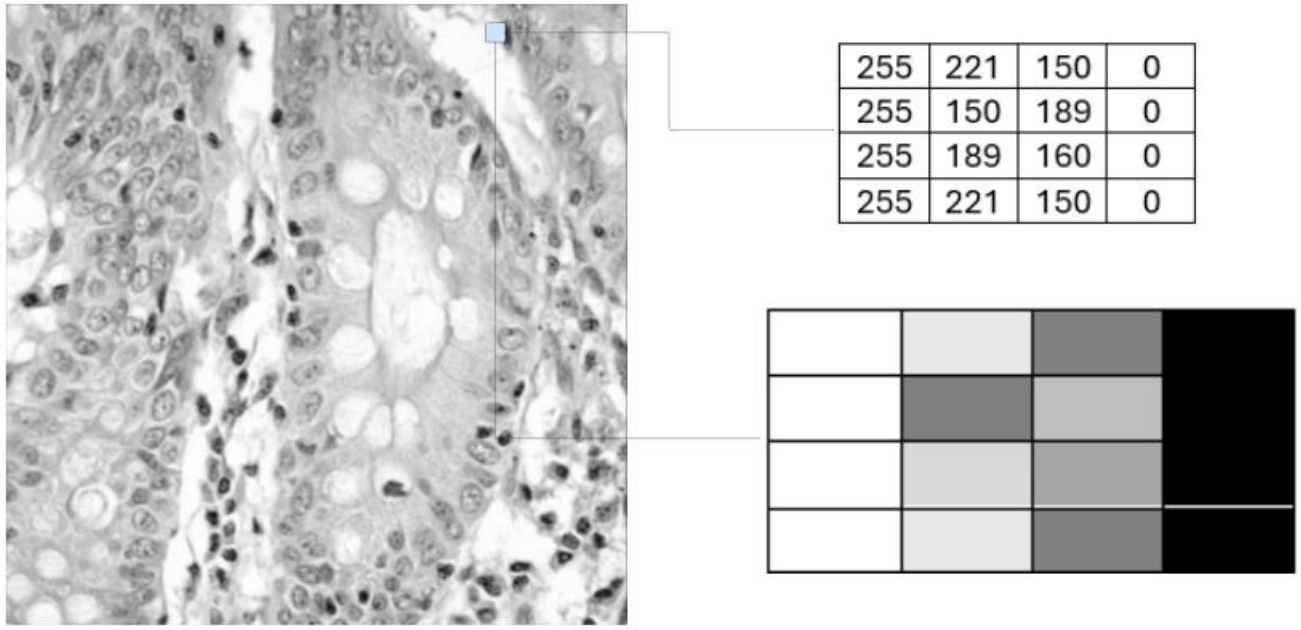
```
15
25
45
```

OPENCV KÜTÜPHANESİ TANITIMI

GÖRÜNTÜ bilgi içeren veri kaynağıdır. Görüntü işleme ve analiz teknikleri kullanılarak hassas ve kritik bilgilerin çıkarılmasını sağlar. Bu ders kapsamında dijital bağlamda piksellerden meydana gelen ve bilgisayarlarda depolanan matrislerden bahsediyor olacağız.

Peki görüntüler bilgisayarlar için ne ifade ediyor?

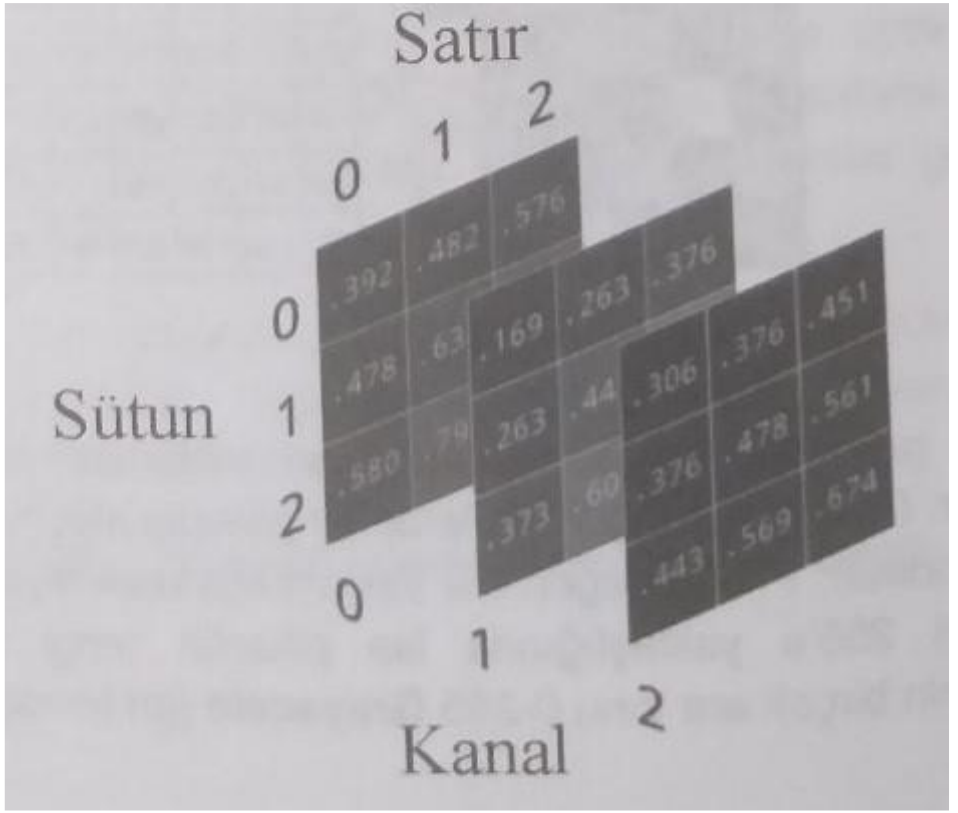
Bilgisayarda gördüğümüz tüm görseller birer matristir. Eğer ki görüntü grinin tonlarından oluşuyor ise bir pikselin alabileceği değer 0-255 arasında değişmektedir



Bir pikselin 0 değerine sahip olması, o pikselin net siyah renk değerine sahip olduğu anlamına gelmektedir. Aynı şekilde 255 renk değerin sahip olan pikseller ise net beyaz renk değerine sahiptir.

Görüntü «https://www.kaggle.com/datasets/biplobdey/lung-and-colon-cancer» linkinden tedarik edilmiştir.

Gri tonlamalı (grayscale); siyah beyaz ve ara gri tonlarını ifade etmeyi sağlayan bir renk uzayıdır. Öte yandan Bilgisayarda renkli görüntüler üretmek için kullanılan birçok renk uzayı mevcuttur. Renkli görüntüler ise bilgisayarda 3 kanallı matrisler ile ifade edilmektedir. Yandaki şekilde renkli bir fotoğrafın bilgisayar ortamındaki karşılığı görülmektedir.



M. Ümit AKSOYLU, Projelerle Yapay Zeka ve Bilgisayarlı Görü, Kodlab Yayınları, 1. Baskı, 2021.

Bu renkli görüntüye ilişkin en arka sırada kırmızı renk değerlerini ifade eden bir matris görülmektedir. Orta sırada yeşil, en ön sıradaysa mavi renklerin değerini ifade eden matris vardır.

Bu görüntüde görülebileceği üzere ekrandaki her bir eleman bir pikseldir ve içinde kırmızı-yeşil-mavi renklerin 0-255 tanım aralığındaki değerlerini barındırır. 0. bant(kanal) kırmızı kanal (R), 1. bant yeşil (G), 2. bant ise mavi (B) renk tonlarının 0-255 tanım aralığında değerlerinin tutulduğu matrislerdir.

Eğer R, G ve B kanalları için sırasıyla 10, 50 ve 80 değerlerini kullanırsak, bu değerler şu şekilde birleştirilir:(R, G, B) = (10, 50, 80) veya #0A3250

Özetle, RGB renk uzayında görüntülerin kırmızı, yeşil, mavi renklerinin farklı oranlarda aynı pikselin içinde birleşmesi sayesinde renkli görsellerin oluşturulmasıdır.

Örneğin 1920X1080 çözünürlüğündeki bir bilgisayar ekranı aslında 1920 piksel uzunluğunda 1080 piksel genişliğinde 3 boyutlu bir matristir. Bu matrisin tüm elemanlarının piksel değeri 0 olursa simsiyah bir ekran tüm elemanlarının piksel değeri 255 olursa da bembeyaz bir ekran elde ederiz. Siyah-beyaz ekranlarda farklı tonlamalar; matrisin her bir hücresindeki değerin 0-255 arasındaki renkleri alması ile ara tonların üretilmesi vasıtasıyla gerçekleşir.

RGB (Red, Green, Blue) renk uzayında kırmızı yeşil ve mavi renklerin belirli piksel değerlerinde karışımı ile (255^3) 16.581.375 farklı renk üretilmektedir.

Bilgisayarda görsel dosya formatları PNG, JPG, JPEG gibi çeşitli uzantılara sahiptir. Bunun sebebi bu görüntülerin farklı algoritmaları ile kodlanması ve sıkıştırılmasıdır. Bu sıkıştırma işini kodlayıcı ve çözücü olarak da bilinir.

Görüntü işleme, kodlanmış görsellerin çözümlenerek görüntü matrisleri haline getirilmesi, bu matrisler üzerinde optimizasyon, dönüşüm, filtreleme ve benzeri işlemlerin uygulanması işlemlerini konu almaktadır. Bu uygulamalar bilgisayar biliminin de alt dallarından biri olan sayısal görüntü işleme alanının konularıdır. Ayrıca görüntü işleme; matris işlemleri, renk uzayları, görüntü formatları gibi birçok matematiksel işlem tekniği içerir. Bu derste görüntü işleme üzerine olan çalışmalarımız OpenCV kütüphanesi vasıtasıyla gerçekleştirilecektir. Fakat farklı kütüphaneler ve işlevleri hakkında bilgi sahibi olmanız için en sık kullanılan diğer görüntü işleme kütüphanelerine de değinilecektir.

GraphicsMagick

Tıpkı OpenCVgibi derlenmiş bir kütüphanedir. Dolayısıyla bir API vasıtasıyla Python üzerinden erişilebilir.

GraphicsMagick, görüntü işleme ve grafiklerle ilgili çeşitli görevler için kullanılan bir yazılım kütüphanesidir. İlk olarak **ImageMagick**'ten türemiştir ve birçok görüntü biçimiyle çalışabilme yeteneğine sahiptir. GraphicsMagick, genellikle daha hızlı, daha hafif ve daha az bellek kullanımı ile ön plana çıkmaktadır.

Mahotas

Tıbbi alanlarda kullanılan görüntüler (biyo-görüntüler) üzerinde bilgisayarlı görü ve yapay zeka tekniklerini uygulayabilmek için geliştirilmiş bir görüntü işleme kütüphanesidir. Mahotas kütüphanesi bu biyo-görüntüleri(Örneğin bir röntgen görseli, hücresel ve moleküler görüntüler veya MR görüntüleri biyo-görüntüleri örnektir) büyük ölçekli ve yüksek verimlilikle analizinde kullanılmak üzere geliştirilmiş bir kütüphanedir. Dolayısıyla morfolojik işlemler gibi birçok fonksiyona sahiptir.

Numpy

Özellikle matris ve vektörlerle ilgili matematiksel işlemleri gerçekleştirebilmek için sıklıkla kullanılan Numpy, aynı zamanda bir görüntü işleme kütüphanesidir. Daha doğrusu, görüntü işleme amacıyla da kullanılabilen bir kütüphanedir. Asıl amacı vektörler, matrisler ve büyük diziler üzerinde işlem yapmak olsa da bildiğiniz üzere görüntü işleme esasen bu işlemlere dayanmaktadır.

Pillow/ PIL

Python programlama dilinde temel görüntü işleme görevleri için geliştirilmiş açık Kaynak kodlu bir kütüphanedir. PIL, bir görüntü üzerinde okuma, yeniden ölçekleme, farklı görüntü formatlarında kaydetme gibi işlemleri gerçekleştirebilir.

Pillow, Tidelifttarafından desteklenen gelişmiş bir PIL dağıtımıdır. Filtreleme, görüntü ayrıştırma, derinlik işleme ve benzeri fonksiyonları içerir. Pillow ayrıca çok çeşitli görüntü formatlarını da desteklemektedir. PIL kütüphanesini kullanarak bir görseli dosya olarak okuyabilir ve Python dili ile üzerinde işlemler yapabiliriz.

Scikit-Image

Scikit-Image tıpkı Scikit-Learn gibi SciKit sbünyesinde geliştirilen görüntü işleme ve bilgisayarlı görü kütüphanesidir. Scikit-image ücretsizdir ve açık kaynak kodludur. Diğer çoğu görüntü işleme kütüphanesinin aksine Python ile geliştirilmiştir. Ancak Python kodlarını C kodlarına dönüştürerek C derleyicisi üzerinden derlenmesini sağlamaktadır.

OpenCV

OpenCV(Open Source ComputerVision), gerçek zamanlı ve çevrim dışı görüntü işleme ve bilgisayarlı görü uygulamalarında kullanılan açık kaynak kodlu görüntü işleme kütüphanesidir.

OpenCV; C ve C++ programlama dilleriyle geliştirilmiştir. Bu nedenle doğrudan C++ ile kullanılabilir. Bunun dışında OpenCV'nin birçok programlama dili için de desteği bulunmaktadır. OpenCV başlıca; Python, C#, Perl, Javascript, Matlab ve Ruby gibi birçok programlama dilinde kullanılabilir.

Bu derste görüntü işleme uygulamaları için kullanacağımız OpenCV kütüphanesi oldukça performanslı ve güçlüdür. Görüntü işlemenin yanında yüz tanıma, nesne takibi, derinlik hesaplama ve benzeri ileri seviyeli bilgisayarlı görü görevlerini kolaylıkla gerçekleştirebilir. OpenCV yalnızca görüntüler üzerinde lineer cebir işlemleri yapmakla kalmaz, bunun ötesinde sezgisel yaklaşımlarla bilgisayarın görme ve gördüğünü anlama tabanlı görevleri yerine getirmesini sağlar. OpenCV kütüphanelerinin bazı kullanım alanlarına örnek vermek gerekirse;

OpenCVKütüphanesinin Kullanım Alanları

Yüz tespiti: Canlı görüntü veya bir fotoğraf veya video üzerinde insan yüzlerinin yüz bileşenlerinin tespiti.

Yüz tanıma: Yüzün özelliklerini çıkararak yüze bir kimlik atamak. Bu uygulama akıllı telefonlarda da sıklıkla kullanılmaktadır.

Hareket yakalama: Kamera veya görüntüleme aygıtı karşısında yapılan hareketlerin yakalanarak sanal ortamda 3D veya 2D bir nesnenin hareketlendirmesinde kullanılması.

İnsan bilgisayar etkileşimi: Örneğin felçli bireylerin gözleriyle kontrol edebileceği bir bilgisayar yazılımının geliştirilmesi.

Gezgin robotlar: İnsansız hava araçları veya otonom araçlar gibi kendi kendine görme, karar verme mekanizmalarına sahip olması beklenen akıllı ve hareketli donanımlar.

Nesne tanıma: Bir nesnenin tespit edilmesi, görüntü içerisindeki koordinatların belirlenmesi ve nesne türünün, renginin, vb. özelliklerinin saplanması.

Görüntü sınıflandırma: Bir görüntü içerisinde yer alan farklı parametreleri işaretleme ve ayrıştırılması.

Arttırılmış gerçeklik: Canlı görüntü üzerinde sanal ortamda bir nesne yerleştirmek veya sanal ortamıyla gerçek dünya arasında görüntü işleme algoritmalarını kullanarak bir etkileşim geliştirilmesi.

Tıbbi analizler: Tomografi cihazlarında toplanan sayısal görüntünün birleştirilerek kesit görseller haline getirilmesi.

OPENCV İLE GÖRÜNTÜ İŞLEME UYGULAMALARI

OpenCV Kütüphanesini Öğrenelim

Bu ders kapsamında yapılan uygulamalarda kullanılacak görüntüler, https://www.kaggle.com/datasets/biplobdey/lung-and-colon-cancer linkinden temin edilmiştir.

hafta2_uygulama1.ipynb

OpenCV Uygulamalarına Giriş

OpenCV(Open Source ComputerVision), **gerçek zamanlı** ve **çevrim dışı** görüntü işleme ve bilgisayarlı görü uygulamalarında kullanılan açık kaynak kodlu görüntü işleme kütüphanesidir. Open CV kütüphanesini pip paket yöneticisi ile kurabilirsiniz. Bunun için terminalde;

```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\fatma\anaconda3\lib\site-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.17.3 in c:\users\fatma\anaconda3\lib\site-packages (from opencv-python) (1.21.6)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ygments (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ltk (c:\users\fatma\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -heel (c:\users\fatma\anaconda3\lib\site-packages)
```

```
import cv2
cv2.__version__
```

```
'4.7.0'
```

TEMEL OPENCV FONKSİYONLARI

Görsel Okuma ve Kaydetme

1-)Görsel Okuma

OpenCV kütüphanesini kullanarak bir **görsel dosyasını okumak** için **imread** fonksiyonu kullanılmaktadır. İmread fonksiyonu, farklı formatlarda istiflenmiş **görsel dosyalarını okur** ve bir **değişkenin** içerisinde **matris biçiminde bellekte** tutar. OpenCV kütüphanesinin **okuyabildiği ve yazabildiği** görsel formatı yaklaşık **20 çeşittir**. Bu dosya formatlarına Windows bipmaps(*.bmp), JPEG files (*.jpeg), Portable Network Graphics (*.png) örnek olarak verilebilir.

Görsel formatlarının birbirine göre farkları ve **kullanım alanları** mevcuttur. Örneğin **GIF** formatındaki görseller **hareketli animasyonlar** içerebilirken **PNG** formatındaki görsel ise **Alfakanalı** (saydamlık) içerebilmektedir.

OpenCV Kütüphanesi, **farklı çeşitlerde** görsel dosyalarını **okuma ve yazma** konusunda oldukça **geniş kapsamlıdır**. Kütüphane, kendisini kullanan yazılımcıların tüm bu dosya formatlarında kolaylıkla okuma ve yazma işlemlerini gerçekleştirebilmesini sağlamak üzere geliştirilmiştir. **Böylelikle tek bir fonksiyon ile 20'ye yakın dosya formatında okuma ve yazmaya** olanak sağlamaktadır.

İmread fonksiyonu **ilk parametre** olarak okunacak **görsel dosyanın dosya düzenini**, **ikinci parametre** olarak ise **resmin okuma modunu** (flag) almaktadır. İmread fonksiyonunun desteklediği **üç çeşit okuma modu** mevcuttur.

1-Okuma Modu:IMREAD_COLOR

2-Okuma Modu:IMREAD_GRAYSCALE

3-Okuma Modu:IMREAD_UNCHANGED

1-Okuma Modu : IMREAD_COLOR

Bu okuma modunda **görsel dosya renkli** olarak yüklenir. Görselin alfa kanalı yani **saydamlığı ihmal** edilir. Eğer bir dosya okuma modu (flag) verilmezse yani fonksiyon yalnızca dosya dizini ile çağrılırsa **varsayılan bayrak IMREAD_COLOR** olacaktır. OpenCV kütüphanesini kullanarak değişken içerisindeki bir görseli **ekrana yansıtmak** için **imshow** fonksiyonu kullanılmaktadır. Bu fonksiyon parametre olarak aldığı değişkeni görsel olarak bir pencere içerisinde göstermektedir. Yani imshow fonksiyonu **ilk parametre** olarak oluşturulacak **pencerenin başlığını ikinci parametre** olarak ise **görseli barındıran değişkeni** parametre olarak almaktadır.

Fakat bu parametreleri alan kod satırı çalıştığında **pencere çok hızlı bir biçimde kaybolacaktır.** Bunun sebebi kodlar yukarıdan aşağı doğru çalıştığı için görevini yerine getiren **imshow** fonksiyonunun otomatik olarak **sonlanmasıdır**. Bunu engellemek için OpenCV'nin imshowfonksiyonundan sonra genellikle **kullanıcının bir tuşa basması** istenir. Bunu sağlayabilmek için OpenCV'nin **waitkey** fonksiyonu kullanılmaktadır. Herhangi bir tuşa **basılmasının ardından açık olan tüm OpenCV pencerelerinin sonlandırılması** için **destroyAllWindows()** fonksiyonu kullanılabilir. Böylelikle bir pencere içerisinde görseli ekranı çizen ve bir tuşa basıldığında OpenCVpencerelerini sonlandıran **kod** yandaki gibi olacaktır.

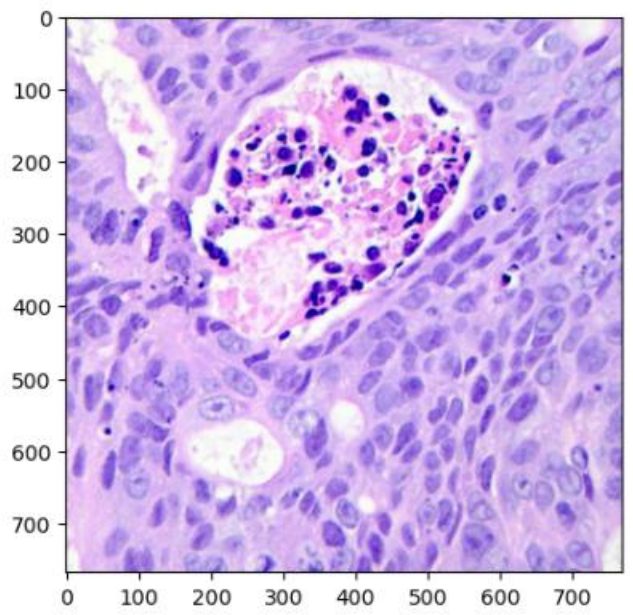
1-Okuma Modu:IMREAD_COLOR ve Görseli Ekrana Çizdirme

[1]:
```
import cv2
gorsel1=cv2.imread("colonca736.jpeg",cv2.IMREAD_COLOR)
cv2.imshow("Orijinal adenocarcinoma görseli",gorsel1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[3]:
```
import cv2
gorsel2=cv2.imread("colonn1265.jpeg",cv2.IMREAD_COLOR)
cv2.imshow("Orijinal benign tissue görseli",gorsel2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[11]:
```
#import matplotlib.pyplot as plt
#image1=cv2.cvtColor(gorsel1,cv2.COLOR_BGR2RGB)
#plt.imshow(image1)
```

[11]: `<matplotlib.image.AxesImage at 0x2d53595c9a0>`



2-Okuma Modu : IMREAD_GRAYSCALE

Görsel dosya **gri tonlama** modunda eklenir.

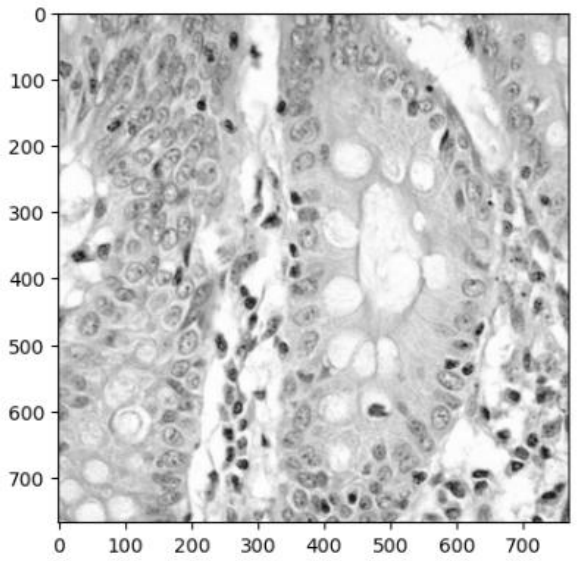
2-Okuma Modu:IMREAD_GRAYSCALE ve Görseli Ekrana Çizdirme

[16]:
```
import cv2
gorsel1=cv2.imread("colonca736.jpeg",cv2.IMREAD_GRAYSCALE)
cv2.imshow("Orijinal adenocarcinoma görseli",gorsel1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[17]:
```
import cv2
gorsel2=cv2.imread("colonn1265.jpeg",cv2.IMREAD_GRAYSCALE)
cv2.imshow("Orijinal benign tissue görseli",gorsel2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[25]:
```
#import matplotlib.pyplot as plt
#image2=cv2.cvtColor(gorsel2,cv2.COLOR_BGR2RGB)
#plt.imshow(image2)
```

[25]:
```
<matplotlib.image.AxesImage at 0x2d53579eb20>
```



3-Okuma Modu : IMREAD_UNCHANGED

Görsel dosya **alfa kanalı** dahil olmak üzere yüklenir. Dolayısı ile görsel dosya içerisindeki **saydamlık**(transparency) **korunur**.

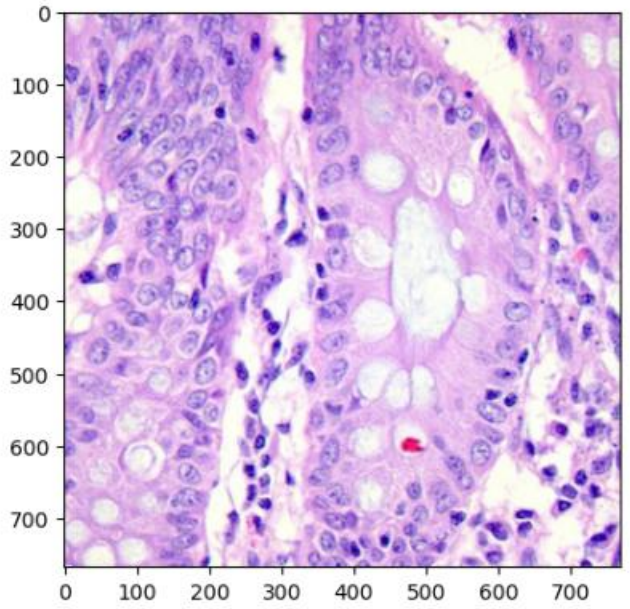
3-Okuma Modu:IMREAD_UNCHANGED ve Görseli Ekrana Çizdirme

[30]:
```
import cv2
gorsel3=cv2.imread("colonca736.jpeg",cv2.IMREAD_UNCHANGED)
cv2.imshow("Orijinal adenocarcinoma görseli",gorsel3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[31]:
```
import cv2
gorsel3=cv2.imread("colonn1265.jpeg",cv2.IMREAD_UNCHANGED)
cv2.imshow("Orijinal benign tissue görseli",gorsel3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[32]:
```
#import matplotlib.pyplot as plt
#image3=cv2.cvtColor(gorsel3,cv2.COLOR_BGR2RGB)
#plt.imshow(image3)
```

[32]:
```
<matplotlib.image.AxesImage at 0x2d538f71310>
```



Görseli Kaydetme

Open CV kütüphanesini kullanarak değişken içerisindeki **bir görseli dosya olarak kaydetmek** için **imwrite** fonksiyonu kullanılmaktadır. Bu fonksiyon parametre olarak aldığı değişkeni görsel dosyası olarak diske kaydetmektedir.

İmwrite fonksiyonu **ilk parametre** olarak kaydedilecek **görsel dosyasının adını** veya dosya dizindeki konumunu, **ikinci parametre** olarak ise görseli barındıran **değişkeni** parametre olarak almaktadır.

Bu doğrultuda **resmi okuyan** ve **siyah beyaz** olarak **png** formatında sıkıştırarak kayıt eden bir pythonprogramı yazalım.

Yanda sunulan Python programında verdiğimiz **sıkıştırma oranı 0 ile 9** arasında değer almaktadır. Burada maksimum olarak **9** değerini vererek **daha yüksek sıkıştırma** yapabilir ve **daha düşük boyut ve .png dosyası** elde edebiliriz

Görseli Kaydetme

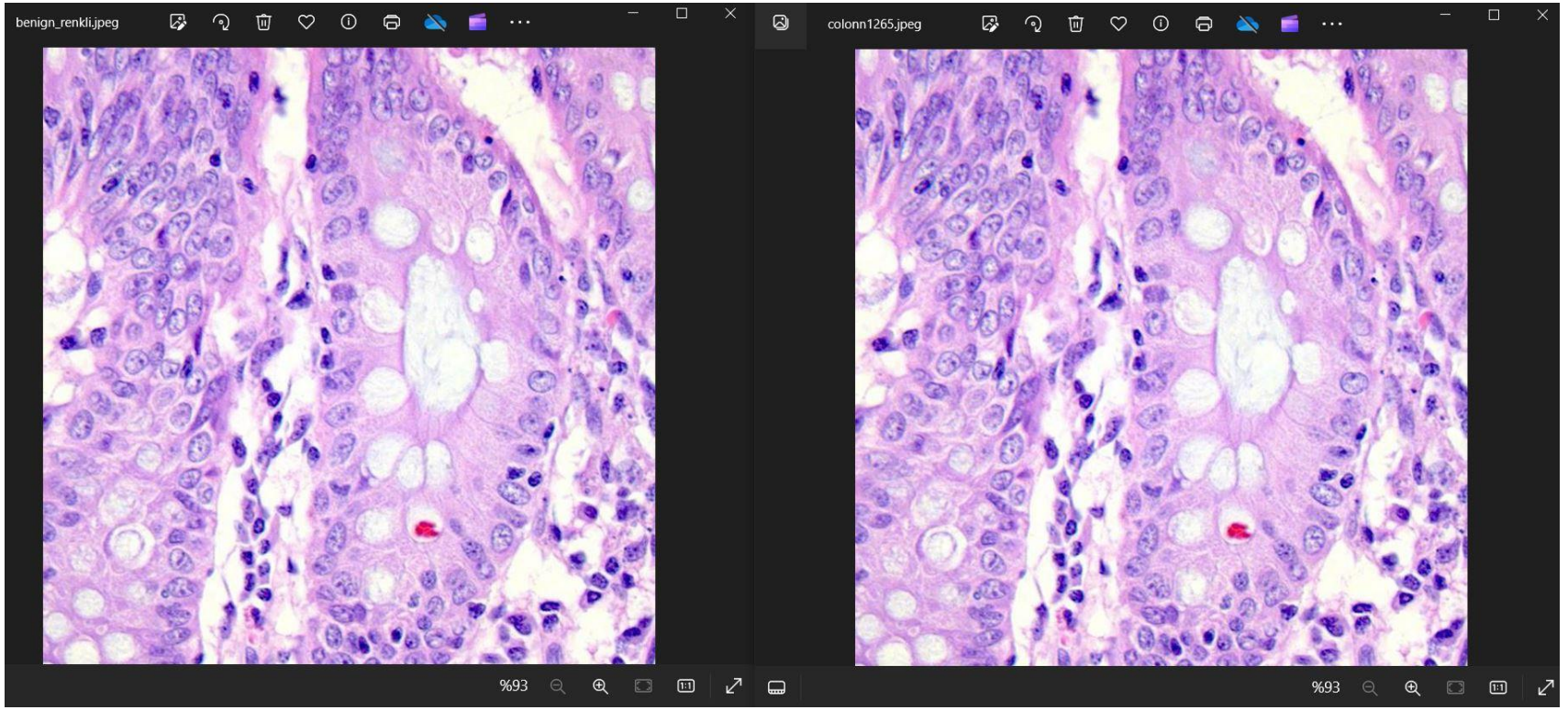
[39]:
```
import cv2
import numpy as np
gorsel4=cv2.imread("colonn1265.jpeg",cv2.IMREAD_GRAYSCALE)
cv2.imwrite("benign.jpeg",gorsel4,[int(cv2.IMWRITE_PNG_COMPRESSION),5])
```

[39]:
```
True
```

KARŞILAŞTIRMA

[41]:
```
import cv2
import numpy as np
gorsel4=cv2.imread("colonn1265.jpeg")
cv2.imwrite("benign_renkli.jpeg",gorsel4,[int(cv2.IMWRITE_PNG_COMPRESSION),5])
```

[41]: True



RENK UZAYLARI İŞLEMLERİ

İnsan gözü esasen **kırmızı yeşil ve mavi renkleri ayırt edebilir**. Gördüğümüz diğer renkler özünde bu üç rengin farklı oranlarda **kombinasyonlarından** oluşmaktadır. Bu bağlamda insan gözü **yaklaşık 1 milyon farklı** renk tonunu ayırt edebilmektedir. **Fiziksel olarak renkler, ışığın dalga boyundaki** farklılıklar ile oluşmaktadır. Bu nedenle **bilindik evrende yaklaşık 100 milyar** farklı renk mevcuttur. Elbette ki **tüm** renkler (hatta renklerin çok büyük bir çoğunluğu) **insanların algılayamayacağı** aralıktadır.

Bilgisayarlar insanların algılayabildiği renkleri (görünür renk kümesini) kapsayan kümelenmiş renk grupları ile **görüntüler** üzerinde çalışırlar. Bu kümelere **renk uzayı** denir.

Günümüzde bilgisayarların üzerinde işlem yapabileceği rengin bir sınırı yoktur. Ancak **renk uzayları**, genellikle **görünür renk üzerinde işlem yapabilmek üzere kümelenmiş renkleri** kapsar.

Görüntü işleme için **4 yaygın renk uzayı** kullanılır.

1-Grayscale Renk Uzayı

Grayscale yani gri tonlamalı renk uzayı, tüm renklerin siyah ve beyaz arasında gri rengin tonlarına sahip olduğu bir renk uzayıdır. Bu renk uzayında **net siyah (0) ve net beyaz (255)** renk değerleri arasında **grinin farklı tonları** mevcuttur.

Gri tonlamalı renk uzayı eskiden büyüklerimizin evinde bulunan siyah beyaz televizyonların ekranlarına benzetilebilir. Bilgisayarda okunan herhangi bir görüntü grayscale renk uzayına dönüştürülebilir. Bu durumda görüntü siyah-beyaz yani gri tonlamalı olacaktır.

NOT : Gri tonlamalı renk uzayı; en sıkıştırılmış daha doğrusu **en az renk sayısına sahip renk uzayıdır.** Bu nedenle makine öğrenmesi, nesne tanıma, optik karakter tanımlama gibi **görüntü anlamlandırma üzerine** geliştirilen sezgisel yaklaşımlarda **gri tonlamaya dönüştürme** sıklıkla kullanılmaktadır

RGB Renk Uzayı

Red, green ve blue yani **kırmızı, yeşil ve mavi** renklerinin baş harfleri ile adlandırılmıştır. Bu parametrelerden her biri kırmızı, yeşil veya mavi renk değerlerini tutar. Bu parametrelerdeki **şiddet bilgisi ile kırmızı, yeşil ve mavi renkleri karıştırılır ve ara renkler elde edilir**.

R, G, B renk uzayında her bir değer 0.255 arası değer alır. Yani

R=(0,255),

G=(0,255),

B=(0,255)

Dolayısıyla **RGB renk uzayında 255^3 farklı renk** mevcuttur.

HSV Renk Uzayı

Hsv (Hue, Saturation, Value) renk uzayı adını **renk özü, doygunluk ve parlaklık** kavramlarından almaktadır. **RGB renk uzayında** bir rengin tanımlaması için **üç ayrı rengin bileşimi** kullanılıyordu. Burada ise bir renk; **renk özütü, parlaklık ve doygunluk** değerlerinin **birleşimi** ile elde edilmektedir.

YUV Renk Uzayı

YUV, genellikle **video görüntülerini işlemek** için kullanılan bir renk uzayıdır.

YUV,

Y:Luminance

U:Chrominance

V:Chrominance2

Sözcüklerinin baş harflerinden oluşmaktadır.

YUV renk uzayının geliştirilmesinde renkli bir görüntü veya video için **insan algısı dikkate alınarak renklerin temsilinde hataları azaltılmış bir bant** oluşturulması amaçlanmıştır. Böylelikle görseldeki **renk hataları veya sıkıştırma kusurları** doğrudan RGB gösterimine kıyasla daha etkili bir şekilde **insan algısından gizlenir**. Bu nedenle YUV renk uzayı, **film ve sinema sektöründe video kodlayıcı/kod çözücülerinde** tercih edilmektedir. Video kodlama ve depolanmasında **özellikle tercih** edilmesinin sebebi RGB renk uzayına göre **daha az bir byte kullanarak** görüntülerin **istiflenmesine**olanak sağlamasıdır. Bu durum **disk kullanımı ve veri transferi, bant genişliği açısından YUV renk uzayını** büyük medya dosyalarında **popüler** kılmaktadır

Renk Uzayı Dönüşümleri

GrayScale renk uzayına dönüştürmek

OpenCV kütüphanesinde **bir renk uzayında kodlanmış görüntüyü, başka bir renk uzayındaki karşılığına dönüştürmek** için ***cvtColor*** fonksiyonu kullanılır.

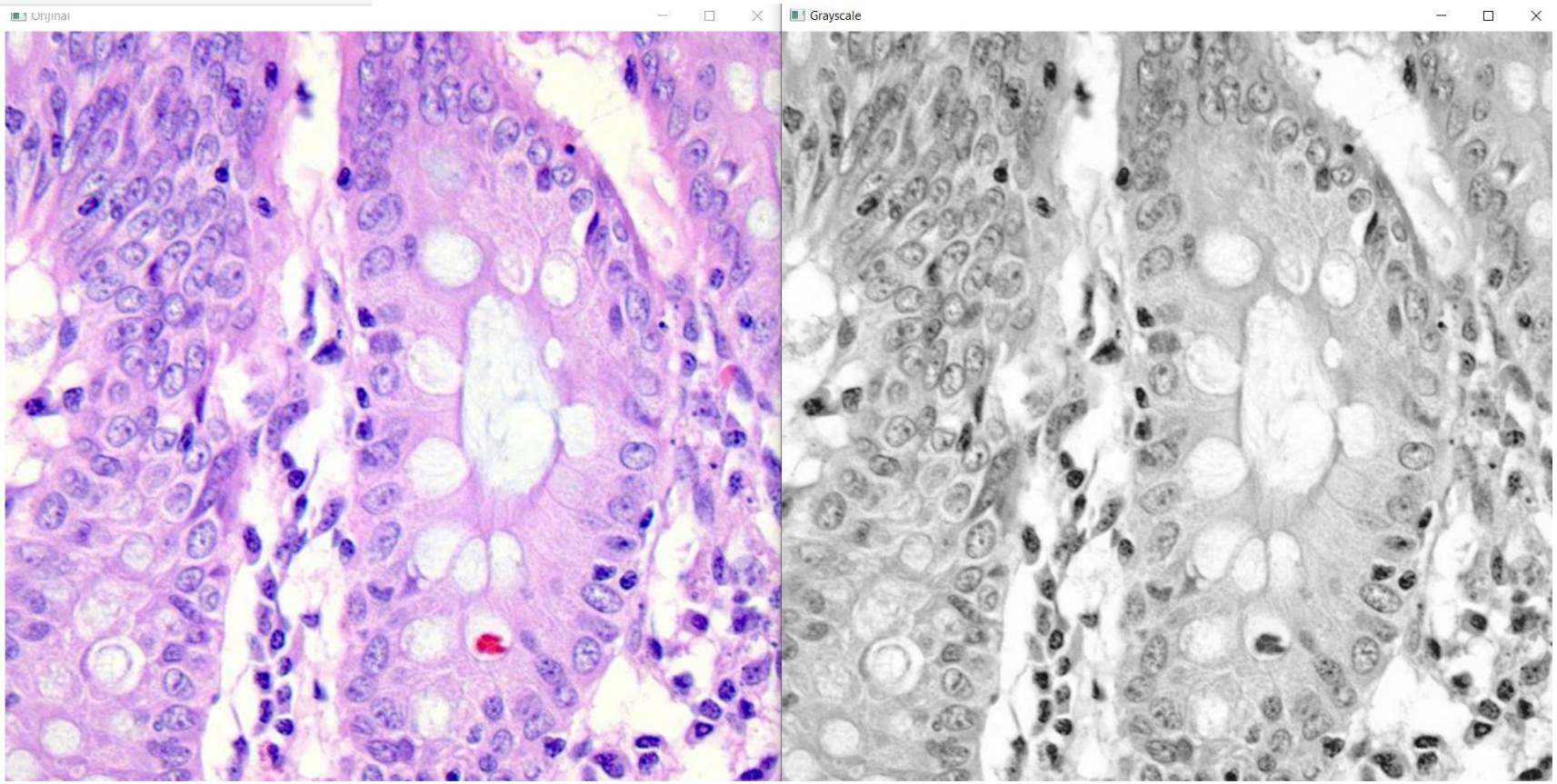
Örnek kullanım:

cikti=cv2.cvt.Color(kaynak,hedef_renk_uzayi)

Grayscale renk uzayına dönüştürmek

```
[4]: gorsel5=cv2.imread("colonn1265.jpeg")
     gorsel6=cv2.cvtColor(gorsel5,cv2.COLOR_BGR2GRAY)
```

```
[6]: cv2.imshow("Orijinal",gorsel5)
     cv2.imshow("Grayscale",gorsel6)
     cv2.waitKey(0)
     cv2.destroyAllWindows()
```



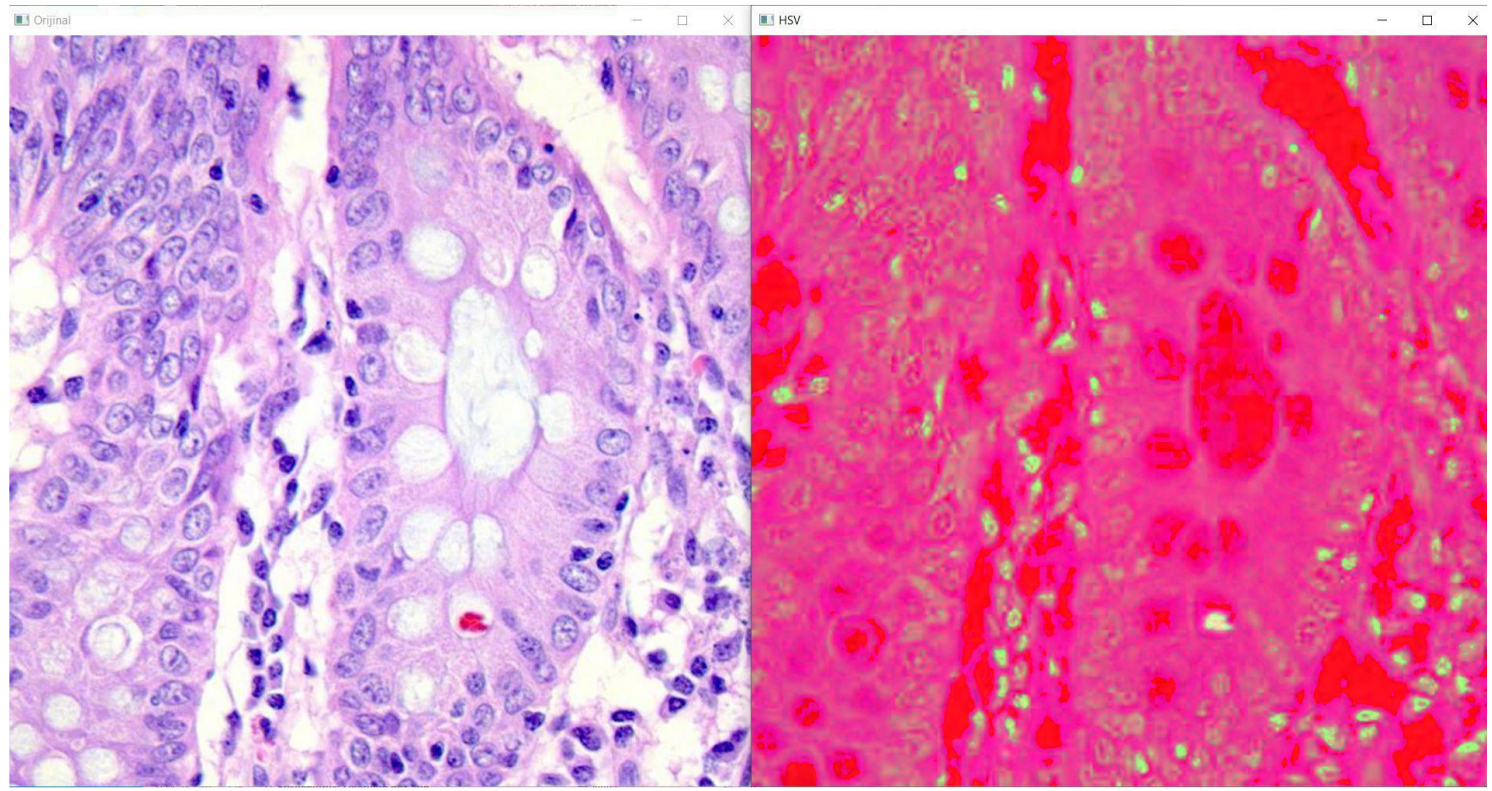
HSV renk uzayına dönüştürmek

RGB uzaydaki bir görüntüyü HSV renk uzayına taşıyalım. Burada taşıma işlemini değerleri dönüştürmeden yaptığımız için her bir pikselin R-G-B parametreleri H-S-V değeri olarak değişecektir

HSV renk uzayına dönüştürmek

[9]:
```
gorsel7=cv2.imread("colonn1265.jpeg")
gorsel8=cv2.cvtColor(gorsel7,cv2.COLOR_BGR2HSV)
```

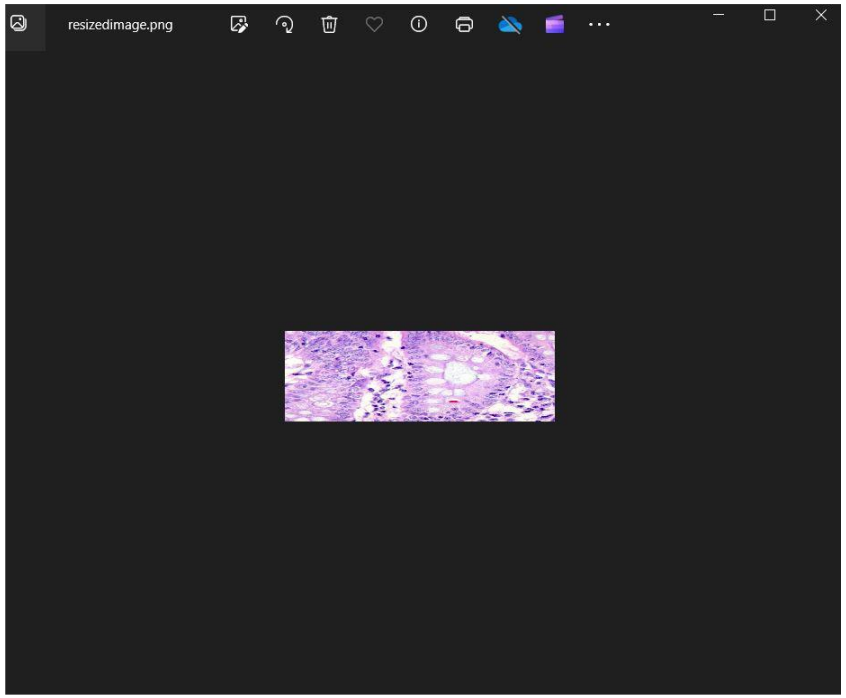
[10]:
```
cv2.imshow("Orijinal",gorsel7)
cv2.imshow("HSV",gorsel8)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Yeniden Boyutlandırma

OpenCV kütüphanesini kullanarak bir görseli **yeniden boyutlandırmak** için **resize** fonksiyonu kullanılmaktadır. Resize fonksiyonu**, matris olarak işlenilmiş görsel** üzerinde **piksel tabanlı geometrik dönüşüm** teknikleri uygulayarak **yeniden boyutlandırır**. Bu teknikler; **küçültme** için «INTER-AREA» **yakınlaştırma** için ise «INTER-CUBIC» ve «INTER-LINEAR» isimleri ile bilinmektedir. **Yeniden boyutlandırma** işleminde bir geometrik dönüşüm tekniği belirtilmediği sürece **varsayılan** olarak «INTER-LINEAR» kullanılmaktadır.

Yandaki şekilde görüldüğü üzere ilgili **görselin boyutları 768x768** pikseldir. **Üçüncü boyut** ise resmin **üç renk kanalını (R,G,B)** ifade etmektedir. Bu görseli **300x100** boyutlarına dönüştürelim ve dönüşüm sonrası yeni matrisimizi kayıt edelim.



YENİDEN BOYUTLANDIRMA

[12]:
```
import cv2
gorsel9=cv2.imread("colonn1265.jpeg")
```

Görseli dosyadan okuduk. Görselin boyutlarını öğrenelim.

[13]:
```
print(gorsel9.shape)
```

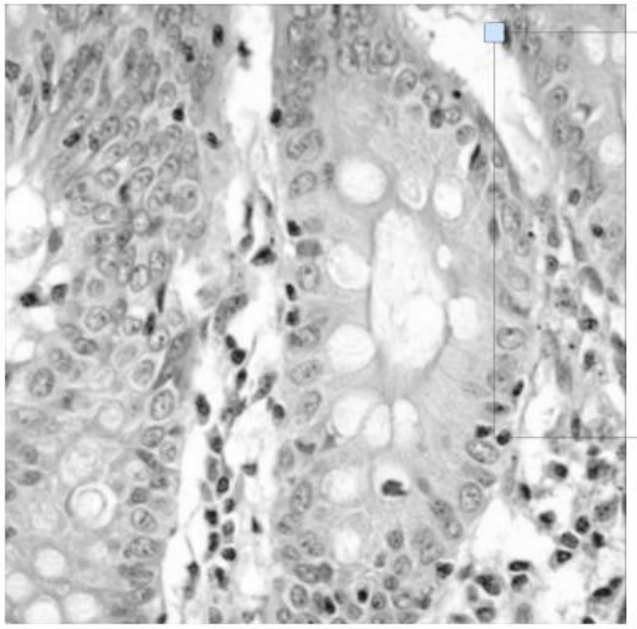
```
(768, 768, 3)
```

[15]:
```
boyutlar=(300,100)
gorsel10=cv2.resize(gorsel9,boyutlar,interpolation=cv2.INTER_CUBIC)
cv2.imwrite("resizedimage.png",gorsel10)
```

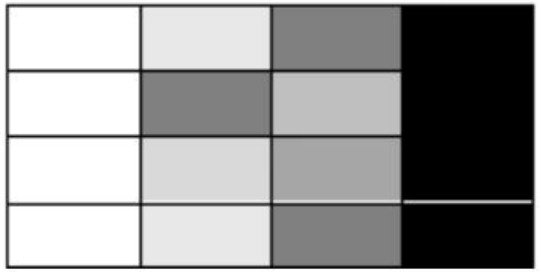
[15]:
```
True
```

PİKSEL DÜZEYİNDE İŞLEMLER

Görüntü işleme kapsamında işlediğimiz diğer başlıklarda **OpenCV'nin hazır fonksiyonlarını** kullanarak **görseller** üzerinde işlem yapmıştık. Bildiğiniz üzere **görseller özünde matristir.** Dolayısıyla biz bu matris üzerinde **piksel düzeyinde işlemler de**yapabiliriz. Yandaki şekilde siyah beyaz tonlamalı bir görselin matris olarak gösterimi görülmektedir.



255	221	150	0
255	150	189	0
255	189	160	0
255	221	150	0



Burada her bir piksel **255-0** arasında değerler almaktadır**. 255 tam beyaz, 0 ise tam siyah** anlamına gelmektedir.

Matris üzerinde **piksellerle** yapacağımız işlemlere basit bir örnek olarak kırpma işlemini verebiliriz.

1-Örneğin telefonların kamerasında **negatif** adı verilen bir filtre vardır. Bu filtre, kamera **görüntüsündeki her rengi anlık olarak tersine çevirmektedir**. Bu doğrultuda siyah-beyaz veya gri tonlamalı bir **görselin negatifini almak, tüm pikselleri tek tek maksimum değerden(255) çıkarıp mutlak değerine almak kadar kolaydır**.

Şimdi, çok renkli**(çok kanallı)** matrislere bu **işlevselliği** kazandırmak için bir uygulama yapalım.

PİKSEL DÜZEYİNDE İŞLEMLER

Negatif Örneği

[27]:
```
import cv2
```

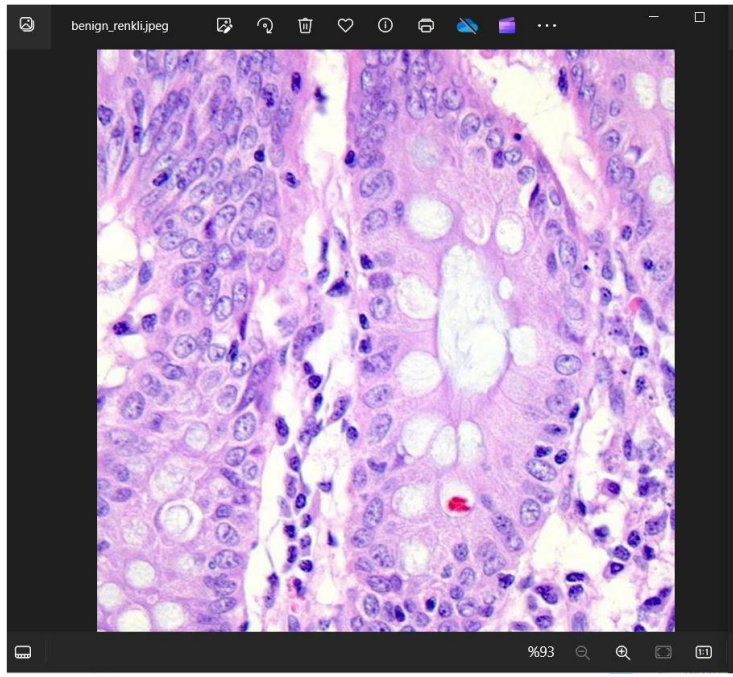
[28]:
```
gorsel14=cv2.imread("colonn1265.jpeg")
```

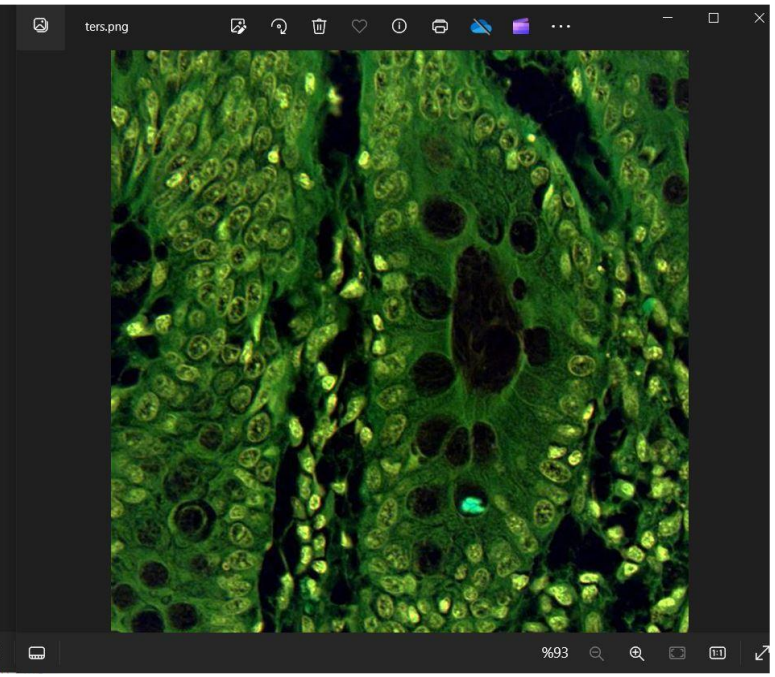
[29]:
```
en,boy=gorsel14.shape[:2]
```

[30]:
```
for x in range(en):
    for y in range(boy):
        for i in range(3):
            gorsel14[x][y][i]=255-gorsel14[x][y][i]
```

[31]:
```
cv2.imwrite("ters.png",gorsel14)
```

[31]:
```
True
```





Yüksek Karşıtlık Teması Örneği

[35]:
```
import cv2
gorsel15=cv2.imread("colonn1265.jpeg")
en,boy=gorsel15.shape[:2]
```

[36]:
```
basamakSayi=4
basamakDeger=int(255/basamakSayi)
```

[38]:
```
for x in range(en):
    for y in range(boy):
        max=0
        max_indis=0
        for i in range(3):
            pikselDeger=gorsel15[x][y][i]
            if(pikselDeger>max):
                max=pikselDeger
                max_indis=i
    bolum=int(max/basamakDeger)
    gorsel15[x][y]=[0,0,0] #tüm kanalların piksel değerlerini 0 yaptık
    gorsel15[x][y][max_indis]=basamakDeger*bolum #en büyük piksel değerine sahip oaln kanalın rengini en yakın basamağa yuvarladık
cv2.imwrite("karsit.png",gorsel15)
```

[38]: True

UYARI

Esasen Open CV’nin bünyesinde barındırdığı **Yapay Zeka tabanlı olmayan Tüm fonksiyonlar** bizim **bu başlık kapsamında** yaptıklarımız gibi geliştirilmiş **piksel düzeyinde işlemlerdir**. Dolayısıyla kodlanmış bir **pngdosyasını okuyup matris haline getirebildikten sonra** üzerinde belli başlı işlemlerin yapılabileceği **basit bir görüntü işleme kütüphanesini** siz de **yazabilirsiniz**

İşaretleme Fonksiyonları

OpenCV ile görüntü matrisleri üzerinde **piksel bazında** yapılabilecek işlemleri neredeyse **sınırı yoktur**. **Görüntü işlemi** ile ilgili senaryolara ilişkin ihtiyaç duyulabilecek **birçok işaretleme fonksiyonu** mevcuttur. Bu fonksiyonlar kullanılarak **video** veya **görsel** üzerinde **bir çizgi çizilebilir, geometrik şekiller oluşturulabilir veya yazı yazılabilir**. Özünde tüm bunlar basit **birer matris işlemidir**. Bu derste 4 temel işaretleme fonksiyonunu işleyeceğiz.

Çizgi Çizme

OpenCV ile bir **görsel üzerinde çizgi** oluşturmak için **line** fonksiyonu kullanılmaktadır.

-line(görsel, baslangic, bitis, renk, kalinlik)

Şimdi bir örnek yapalım.

İŞARETLEME FONKSİYONLARI

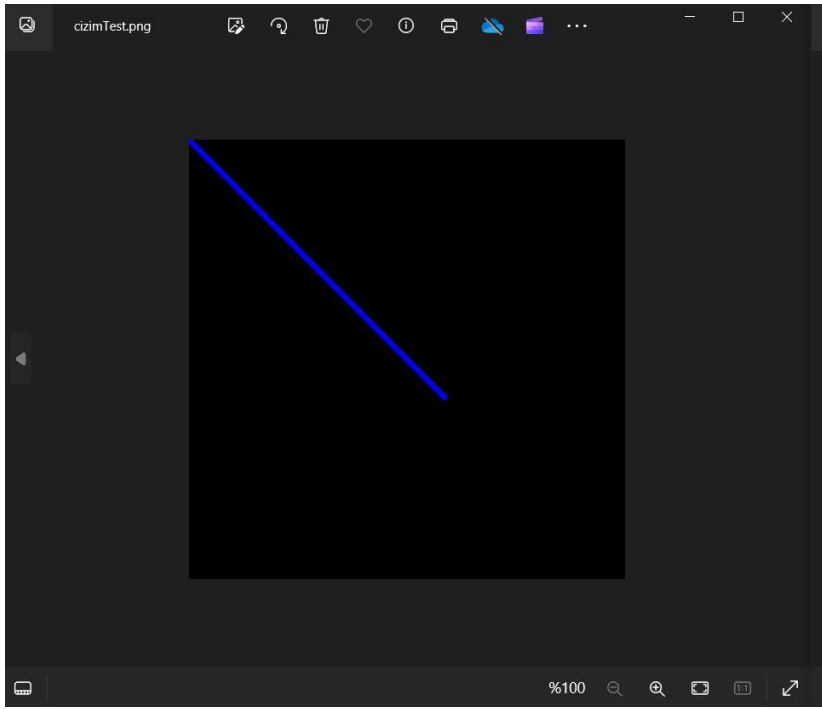
[40]:
```
import numpy as np
import cv2
```

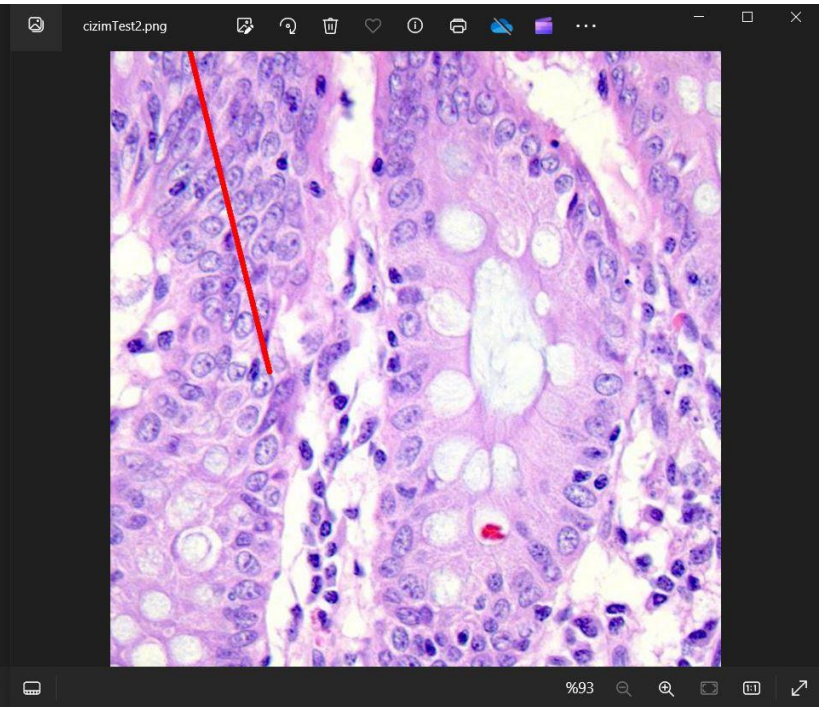
[44]:
```
#siyah boş bir görsel oluşturalı
gorsel16=np.zeros((512,512,3),np.uint8)
Baslangic=(0,0)
Bitis=(300,300)
Renk=(255,0,0)
Kalinlik=5
cv2.line(gorsel16,Baslangic,Bitis,Renk,Kalinlik)
cv2.imwrite("cizimTest.png",gorsel16)
```

[44]: True

[45]:
```
import numpy as np
import cv2
gorsel17=cv2.imread("colonn1265.jpeg")
Baslangic=(100,0)
Bitis=(200,400)
Renk=(0,0,255)
Kalinlik=5
cv2.line(gorsel17,Baslangic,Bitis,Renk,Kalinlik)
cv2.imwrite("cizimTest2.png",gorsel17)
```

[45]: True





Dörtgen Çizme

OpenCVile bir **dörtgen çizmek** için **rectangle**fonksiyonu kullanılmaktadır.

rectangle(görsel,baslangic,bitiş,renk,kalinlik)

Burada **başlangıç**noktası, **dörtgenin sol noktasını** işaret etmektedir. **Bitiş nokta sağ alt noktasını** işaret etmektedir

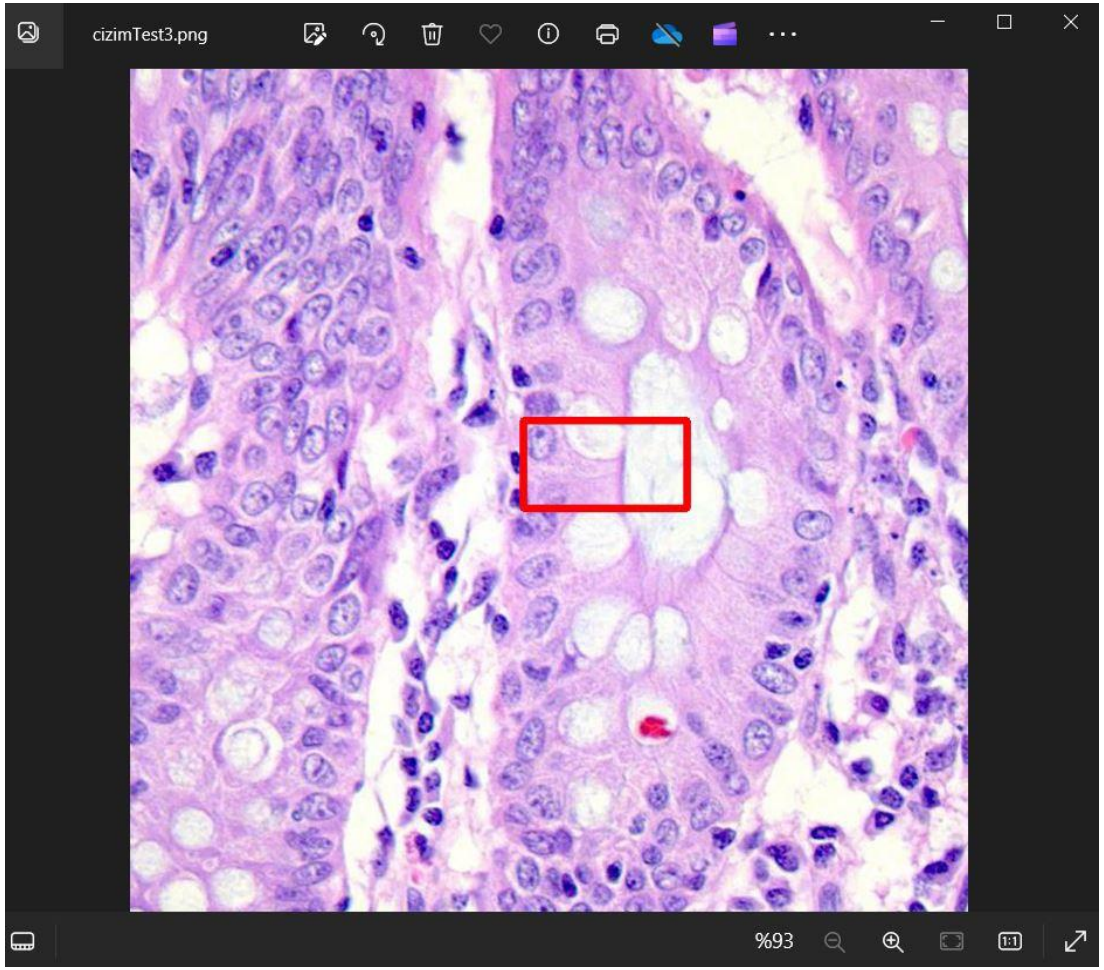
Dolayısıyla **sol üst ve sağ alt noktalar bir dörtgenin köşegeni üzerinde olduğundan diğer noktaları da basit bir matematiksel hesap yardımı ile oluşturulur** ve dörtgen **çizilir**.

Dörtgen çizme en sık kullanılan işaretleme fonksiyonlarından biridir. **Yüz tanıma veya nesne tanıma**gibi işlemlerde tanınan objeyi dörtgen içine almak oldukça **yaygındır**.

DÖRTGEN ÇİZME

[46]:
```
import numpy as np
import cv2
gorsel18=cv2.imread("colonn1265.jpeg")
Baslangic=(360,320)
Bitis=(510,400)
Renk=(0,0,255)
Kalinlik=5
cv2.rectangle(gorsel18,Baslangic,Bitis,Renk,Kalinlik)
cv2.imwrite("cizimTest3.png",gorsel18)
```

[46]:
```
True
```



Poligon Çizme

Poligonlar **düzgün biçimi olmayan çokgenlerdir**. Dörtgen, beşgen .. n'den cisimler bir poligon olarak adlandırılmaktadır. OpenCVkütüphanesi ile görüntü üzerinde poligon çizdirebiliriz.

Poligon çizmek için **polylines** fonksiyonu kullanılır. Çizeceğimiz poligonun **köşe noktalarının koordinatları bir parametre** olarak bu fonksiyonla sağlanacak biçimde çokgen çizdirebiliriz.

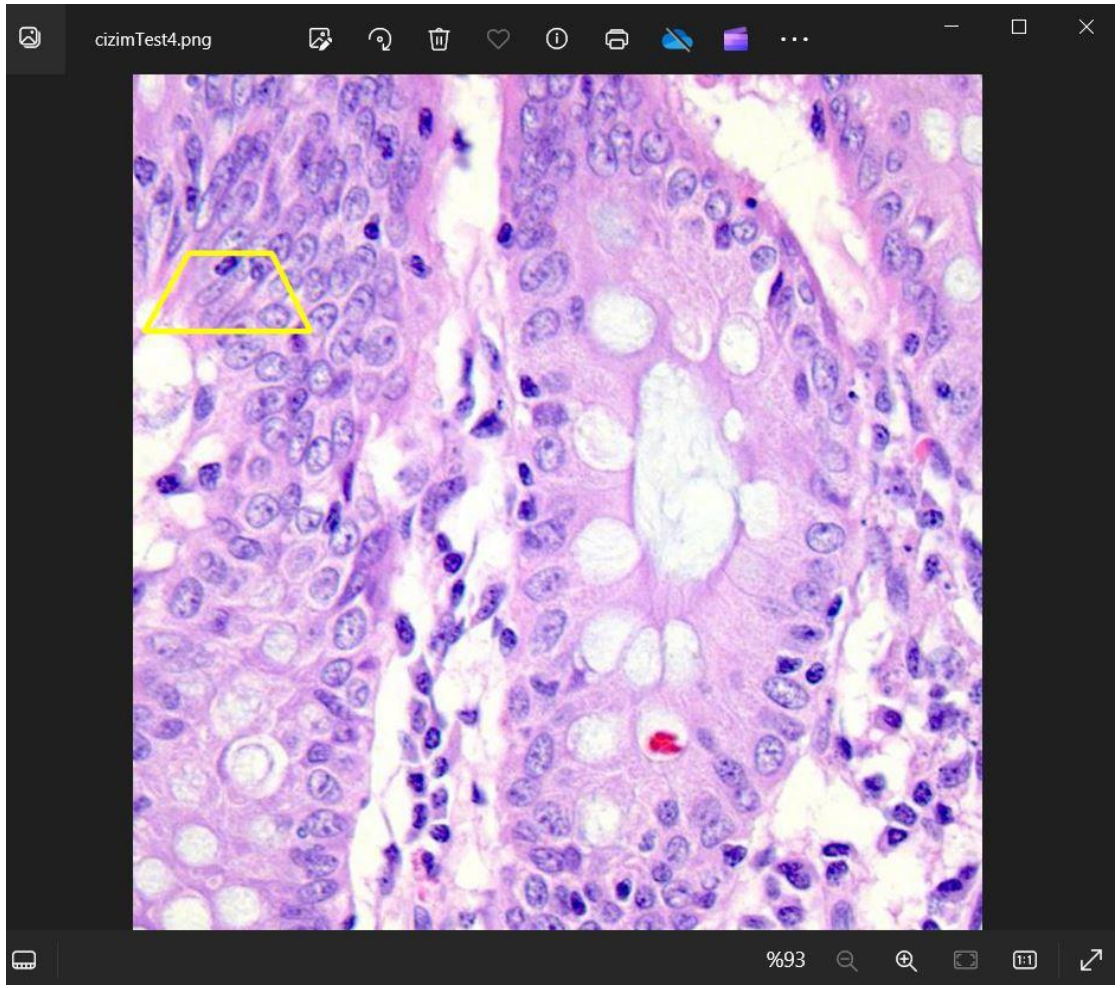
polylines(görsel,koseNoktalari,kapalilik,renk,kalinlik)

POLİGON ÇİZME

```
[47]: import numpy as np
      import cv2
      gorsel19=cv2.imread("colonn1265.jpeg")
      koseNoktalari=np.array([[50,160],[125,160],[160,230],[10,230]],np.int32)
      koseNoktalari=koseNoktalari.reshape((-1,1,2))
      kapalilik=True
      renk=(0,255,255)
      kalinlik=3
      cv2.polylines(gorsel19,[koseNoktalari],kapalilik,renk,kalinlik)
      cv2.imwrite("cizimTest4.png",gorsel19)

[47]: True
```

Kapalılık parametresi, dizi içerisinde verilen son köşe noktası ile ilk köşe noktası arasında bir çizgi olup olmamasını belirtmektedir. Eğer **false** olur ise cisim **açık** olacaktır.

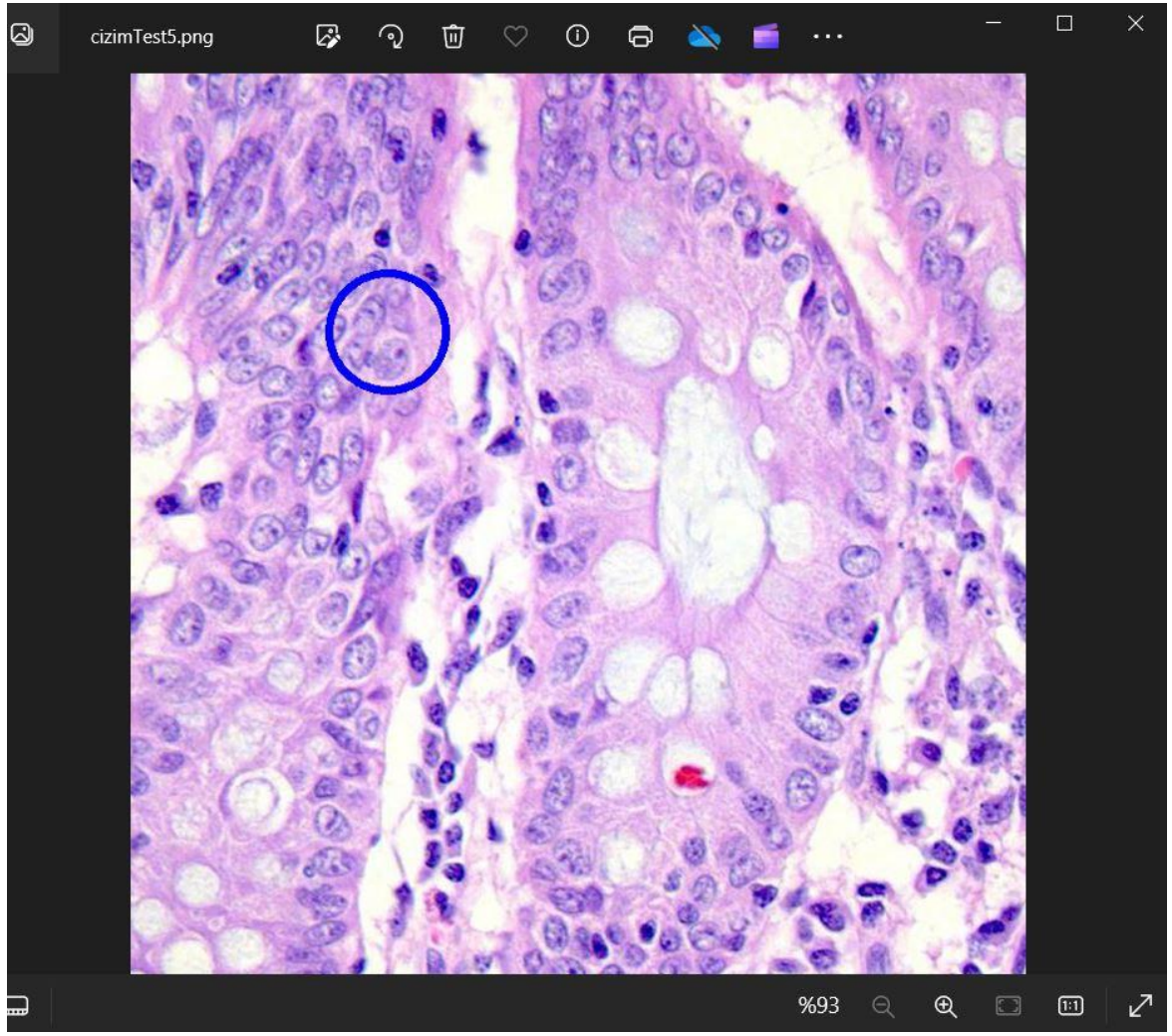


Çember ve Daire Çizme

OpenCVile bir görsel üzerinde çember çizmek için **circle**fonksiyonu kullanılmaktadır.

circle(gorsel,merkez,yaricap,renk,kalinlik)

Çizilecek **çemberin merkezi, yarıçapı, rengi ve kalınlık değeri** belirtilmelidir.
Bir görsel üzerinde çember çizelim.



Çember ve Daire Çizme

[52]:
```
import numpy as np
import cv2
gorsel20=cv2.imread("colonn1265.jpeg")
Merkez=(220,220)
yariCap=50
Renk=(255,0,0)
Kalinlik=5
cv2.circle(gorsel20,Merkez,yariCap,Renk,Kalinlik)
cv2.imwrite("cizimTest5.png",gorsel20)
```

[52]: True

Kalınlık değerini **-1 olarak verirsek çemberin içi doldurulur**. Yani daire çizdirmiş oluruz.

Yazı Yazma

OpenCVile bir **görsel üzerine yazı yazmak** için **putText** fonksiyonu kullanılmaktadır.

putText(görsel,yazilacakMetin,koordinatlar,font,fontBoyutu,renk,kalinlik,satir Tipi)

Yazı Yazma

[3]:
```
import numpy as np
import cv2
gorsel21=cv2.imread("colonn1265.jpeg")
yazilacakMetin="Serdivan/Sakarya"
koordinatlar=(100,400)
font=cv2.FONT_HERSHEY_SIMPLEX
fontBoyutu=2
renk=(0,0,255)
kalinlik=3
satirTipi=cv2.LINE_AA
cv2.putText(gorsel21,yazilacakMetin,koordinatlar,font,fontBoyutu,renk,kalinlik,satirTipi)
cv2.imwrite("yaziTest.png",gorsel21)
```

[3]:
```
True
```

putText fonksiyonu **9 farklı font tipini ve 4 farklı satır** tipini desteklemektedir. Desteklenen satır tipleri yandaki şekilde verilmiştir.

OpenCV Satır Tipleri Tablosu:

Satır Tipi	Python	Açıklama
FILLED	`cv.FILLED`	
LINE_4	`cv.LINE_4`	4-bağlantılı satır
LINE_8	`cv.LINE_8`	8-bağlantılı satır
LINE_AA	`cv.LINE_AA`	Kenarları yumuşatılmış satır

OpenCV Font Tablosu:

Font	Python	Açıklama
FONT_HERSHEY_SIMPLEX	`cv.FONT_HERSHEY_SIMPLEX`	Standart sans-serif fontu
FONT_HERSHEY_PLAIN	`cv.FONT_HERSHEY_PLAIN`	Küçük ölçekli sans-serif fontu
FONT_HERSHEY_DUPLEX	`cv.FONT_HERSHEY_DUPLEX`	Standart sans-serif fontu (FONT_HERSHEY_SIMPLEX'den daha karmaşıktır)
FONT_HERSHEY_COMPLEX	`cv.FONT_HERSHEY_COMPLEX`	Standart serif fontu
FONT_HERSHEY_TRIPLEX	`cv.FONT_HERSHEY_TRIPLEX`	Standart serif fontu (FONT_HERSHEY_COMPLEX'den daha karmaşıktır)
FONT_HERSHEY_COMPLEX_SMALL	`cv.FONT_HERSHEY_COMPLEX_SMALL`	Küçük ölçekli FONT_HERSHEY_COMPLEX fontu
FONT_HERSHEY_SCRIPT_SIMPLEX	`cv.FONT_HERSHEY_SCRIPT_SIMPLEX`	El yazısı benzeri font
FONT_HERSHEY_SCRIPT_COMPLEX	`cv.FONT_HERSHEY_SCRIPT_COMPLEX`	FONT_HERSHEY_SCRIPT_SIMPLEX fontunun bir varyantıdır
FONT_ITALIC	`cv.FONT_ITALIC`	Doğrudan bir font tipi değildir. Diğer font tipleri ile birlikte bir yazıyı italik olarak yazdırmak için kullanılır.

Bu **fontların tam sayı (integer)** tipinden sayı olarak **ID numaraları** mevcuttur. Her defasında uzun uzun **font isimlerini** yazmak yerine **ID numaralarını** kullanmak **tercih** edilebilir.

```
FONT_HERSHEY_SIMPLEX = 0
FONT_HERSHEY_PLAIN = 1
FONT_HERSHEY_DUPLEX = 2
FONT_HERSHEY_COMPLEX = 3
FONT_HERSHEY_TRIPLEX = 4
FONT_HERSHEY_COMPLEX_SMALL = 5
FONT_HERSHEY_SCRIPT_SIMPLEX = 6
FONT_HERSHEY_SCRIPT_COMPLEX = 7
FONT_ITALIC = 16
```

AYRIŞTIRMA TEKNİKLERİ

Bilgisayarlı görü teknolojisinin yeni geliştiği yani biraz daha ilkel olduğu dönemlerde **nesne tanıma** ve **nesne takibi** üzerine geliştirilen tüm uygulamalar **matris işlemlerine dayanan filtreleme ve ayıklama algoritmaları** kullanılarak geliştirilmekteydi. Günümüzde ihtiyaca göre halen bu algoritmalar kullanılmaktadır.

Bu ders kapsamında görüntüler üzerinde yapacağımız işlemler esasen **ihtiyacımızolmayan noktaların, şekillerin, renklerin görüntüden ayıklanması** ve **önemli kenar bölgelerinin işaretlenmesi** gibi teknikler içermektedir. Dolayısıyla **ayrıştırma teknikleri**, görüntü üzerinde çalışan yapay zeka modellerinin ***veri setleri ve giriş verileri üzerinde*** bir nevi **optimizasyon**işlemi görevi görmektedir. **Preprocessing(ön işleme)** olarak ise ***düzleştirme, konvülüsyon, histogramlıtoplama gibi ağırlık bazlı işlemler*** uygulanmaktadır

Renk Filtreleme

Renk filtreleme **istenmeyen renklerin görüntüden temizlenmesi, görüntüdeki renklerin belirli bir aralığa sıkıştırılması** veya **spesifik bir renge sahip nesnelerin takibi** için sıklıkla kullanılmaktadır.

Bu doğrultuda renk filtrelemenin aktif olarak kullanıldığı bir bilgisayarlı görü teknolojisi olan **Motion Capture** yani hareket yakalama teknolojisinden bahsedebiliriz. Bu teknoloji**, bilgisayar oyunları ve animasyon filmlerinin yapımında sıklıkla kullanılan** *animasyon oluşturma metodudur*. Burada oyuncular üzerinde kırmızı noktalar olan özel bir kostüm giyerler. Tamamen yeşil kaplı oda içerisinde oyuncuların eklem noktalarının kırmızı olarak işaretlendiği kıyafetlerle yaptıkları hareketler anlık olarak kaydedilir ve belli başlı algoritmalarla işlenerek sanal ortamdaki 3D karakterin animasyonuna dönüştürülmesi sağlanır. **Motion Capture günümüzde renk filtreleme ile geliştirilebilen en gelişmiş teknolojiler** arasındadır.

Kenar Tespiti

Kenar tespiti, bilgisayarlı görü uygulamaları için oldukça önemli bir adımdır. **El yazısı tanımlama, optik karakter tanıma uygulamaları (plaka okuma vb.), nesne tanımı gibi birçok bilgisayarlı görü uygulamasının zemininde** yer almaktadır.

Görüntüde sınıflandırılabilecek şekilde **farklı olarak kümelenmiş renkleri** çevreleyen **hatlar** kenar olarak adlandırılır. Esasen bu uygulama **yüksek geçişleri ayrıştıran bir filtreleme** işlemidir. Dolayısıyla **yüksek** geçiş değerine (frekansına) sahip olan noktaların **ayrıştırılmasını**, **düşük** frekansı noktaların ise **bloklanmasını** sağlar.

Kenarları tespit etmek için yaygın olarak dört teknik kullanılır.

1)Sobel

Sobel filtresi ile kenarları ayrıştırmak için bir **tarama yönü** seçilmelidir. Bu bağlamda **dikey ve yatayda yüksek frekanslı kenarların** tespitinin sağlanması amaçlanır.

sobel_yatay=cv2.Sobel(görsel,cv2.CV_64F,**1,0**,ksize=5)

sobel_dikey=cv2.Sobel(görsel,cv2.CV_64F,**0,1**,ksize=5)

Kenar Tespiti

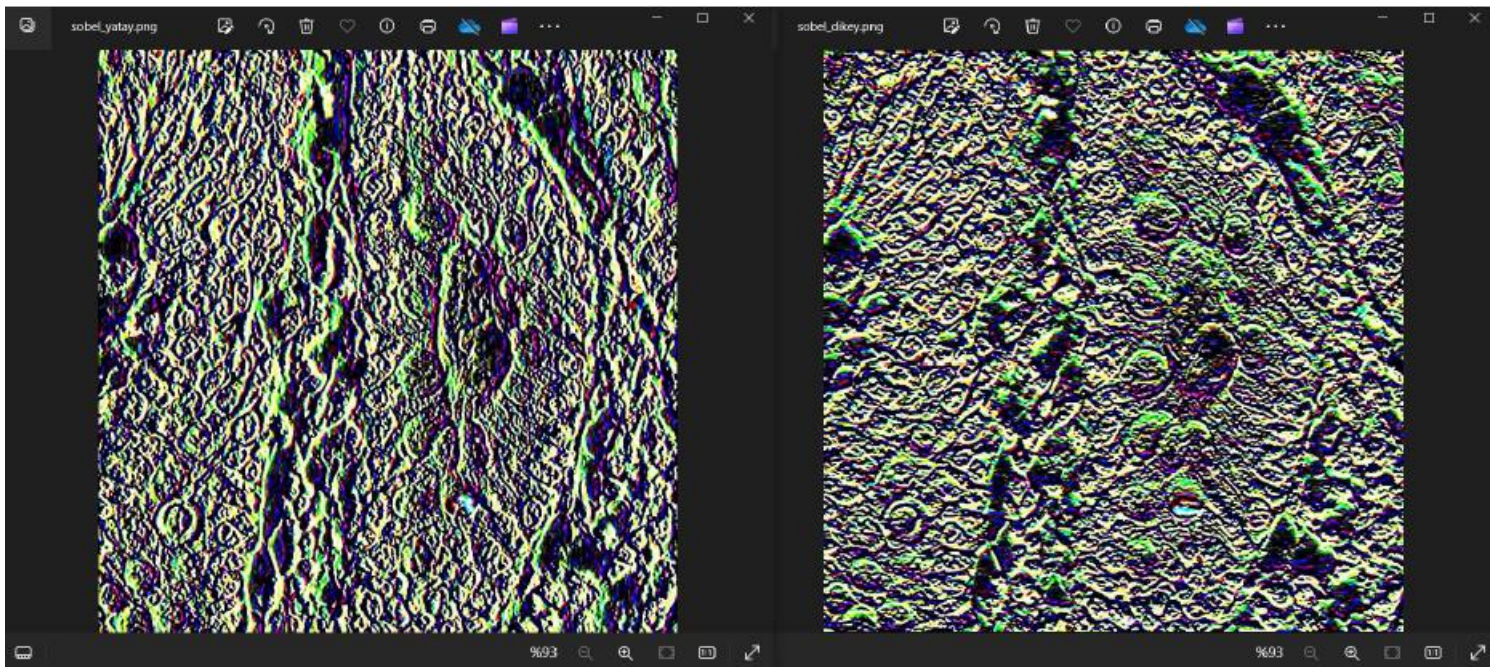
1)SOBEL

[16]:
```
import numpy as np
import cv2
gorsel22=cv2.imread("colonn1265.jpeg")
sobel_yatay=cv2.Sobel(gorsel22,cv2.CV_64F,1,0,ksize=5)
cv2.imwrite("sobel_yatay.png",sobel_yatay)
```

[16]:
```
True
```

[18]:
```
import numpy as np
import cv2
gorsel23=cv2.imread("colonn1265.jpeg")
sobel_dikey=cv2.Sobel(gorsel23,cv2.CV_64F,0,1,ksize=5)
cv2.imwrite("sobel_dikey.png",sobel_dikey)
```

[18]:
```
True
```



NOT

Sobel fonksiyonuna verilen 3. parametre 1, 4. parametre 0 olursa filtreleme **yatay**; 3. parametresi 0, 4. parametre 1 olur ise filtreleme **dikey** yönde olmaktadır. ksize değişkeni ise filtreleme için kullanılan çekirdek boyutunu belirtmektedir. Çekirdek boyutu **küçüldükçe ayıklanan içerik daha detaylı** olmaktadır.

2)Laplacian

Laplacian **oldukça hassas** bir kenar ayrıştırma filtresidir. Görseli **doku bazında** ayrıştırabilmektedir.

Laplacian filtresi ile kenarları ayrıştıralım.

Laplacian=cv2.Laplacian(gorsel,cv2.CV_64F)

2)LAPLACİAN

```
[23]: import numpy as np
      import cv2
      gorsel24=cv2.imread("colonn1265.jpeg")
      laplacian=cv2.Laplacian(gorsel24,cv2.CV_64F)
      cv2.imwrite("laplacian.png",laplacian)

[23]: True
```

3)CANNY

```
[28]: import numpy as np
      import cv2
      gorsel25=cv2.imread("colonn1265.jpeg")
      canny=cv2.Laplacian(gorsel25,200,300)
      cv2.imwrite("canny.png",canny)

[28]: True
```

3)Canny

Canny, **nesne tespiti** için geliştirilen sezgisel sistemlerde çoğunlukla görsel ön işleme aşamasında kullanılır. **İki eşik değeri kullanılarak geçiş frekansının yüksek olduğu hatlar** işaretlenir.

Canny=cv2.Canny(görsel,200,500)

Burada fonksiyonun aldığı ikinci parametre **düşük eşik** değeridir. Üçüncü parametre ise **yüksek eşik** değeridir.

2)LAPLACİAN

```
[23]: import numpy as np
      import cv2
      gorsel24=cv2.imread("colonn1265.jpeg")
      laplacian=cv2.Laplacian(gorsel24,cv2.CV_64F)
      cv2.imwrite("laplacian.png",laplacian)

[23]: True
```

3)CANNY

```
[28]: import numpy as np
      import cv2
      gorsel25=cv2.imread("colonn1265.jpeg")
      canny=cv2.Laplacian(gorsel25,200,300)
      cv2.imwrite("canny.png",canny)

[28]: True
```

Alt eşik (low threshold - 200): Bu değerin altındaki gradyan değerleri **kenar olarak kabul edilmez**.

Üst eşik (high threshold - 500): Bu değerin üzerindeki gradyan değerleri **kesin kenar olarak kabul edilir**.

İki eşik arasında kalan pikseller, **üst eşik değerinin bağlantılı olduğu sürece kenar olarak kabul edilir**, aksi takdirde kenar olarak alınmaz.

Özetle:

Alt eşik (200) → Daha küçük kenarları filtreler, fazla gürültü oluşmasını engeller.

Üst eşik (500) → Güçlü kenarları belirler.

Daha düşük eşik değerleri, daha fazla kenar tespit ederken **gürültü artabilir**. Daha yüksek eşik değerleri, **yalnızca güçlü kenarları seçer** ancak bazı önemli kenarları kaçırabilir.

SINIR ve ŞEKİL TESPİTİ

Sınır ve şekillerin tespiti, nesne tanıma için oldukça önemli bir adımdır. Örneğin OpenCVile bir optik okuyucu geliştirmek için içi siyah renk ile dolu dairelerin tespit edilmesine ihtiyaç duyarız. Aynı şekilde görüntüdeki **nesnelerin ayrıştırılması, sınıflandırılması, gibi birçok sezgisel mekanizma için de sınır ve şekil tespiti** oldukça önemli bir aşamadır.

KAYNAKÇA

- Her Yönüyle Python, Kodlab Yayınları, Fırat Özgül, 9. Baskı
- Projelerle Yapay Zeka ve Bilgisayarlı Görü, KodlabYayınları, M. Ümit Aksoylu, 1. Baskı,2021
- Python ile Uçtan Uca Veri Bilimi, dikeyeksen yayınları, Engin Bozaba,1. Baskı
- https://chatgpt.com/