KIRIKKALE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ

Temel Görüntü İşlemleri

- OpenCV ile Performansı Ölçme
- OpenCV'de Varsayılan Optimizasyon
- Performans Optimizasyon Teknikleri
- Kameradan Video Yakalama
- Dosyadan Video Oynatma
- Video Kaydetme
- Mouse ile Çizim
- Renk Paleti İşlemleri
- Renk Alanlarını Değiştirme
- Nesne İzleme
- Görüntülerin Geometrik Dönüşümleri

OpenCV ile Performansı Ölçme

Görüntü işlemede saniyede çok sayıda işlemle uğraşıldığı için uygulanan kodun sadece doğru çözümü değil aynı zamanda en hızlı şekilde olması gerekmektedir. Bu konuda OpenCV fonksiyonları:

- **cv2.getTickCount**,
- **cv2.getTickFrequency**

OpenCV'nin yanı sıra Python'da, yürütme zamanını ölçmeye yardımcı olan "zaman" modülü de bulunmaktadır.

Başka bir modül olan "profil" de, koddaki her bir işlevin ne kadar zaman aldığı, işlevin kaç kez çağrıldığı gibi kod hakkında ayrıntılı bir rapor almak için kullanılabilir. Ipython'da ise bu özellikler entegre edilmiştir.

OpenCV ile Performansı Ölçme

cv2.getTickCount, bir referans olaydan sonra (makinenin açıldığı an gibi) bu işlevin çağrıldığı ana kadar saat döngülerinin sayısını döndürür.

Bu nedenle, bir işlevin yürütülmesinden önce ve sonra çağrılırsa, işlevi yürütmek için kullanılan bir dizi saat döngüsü elde edilebilir.

cv2.getTickFrequency, yürütme zamanını saniye cinsinden bulmak için saat döngülerinin sıklığını veya saniyedeki saat döngülerinin sayısını döndürür.

```
import cv2

e1 = cv2.getTickCount()
print("Çalışma zamanı")
e2 = cv2.getTickCount()
time = (e2 - e1) / cv2.getTickFrequency()
print(time)
```

```
Çalışma zamanı
0.0001453
```

Nan	Type	Size	
e1	int	1	1162591231198
e2	int	1	1162591232651
time	float	1	0.0001453

OpenCV ile Performansı Ölçme

```
#%%
#Sample Blur

img = cv2.imread('campus.jpg')

e1 = cv2.getTickCount()
for i in range(5,49,2):
    img = cv2.medianBlur(img,i)
e2 = cv2.getTickCount()
t = (e2 - e1)/cv2.getTickFrequency()
print (t)

import time
t1 = time.time()
for i in range(5,49,2):
    img = cv2.medianBlur(img,i)
t2 = time.time()
t3 = t2 - t1
print (t3)
```

```
In [19]: img = cv2.imread('campus.jpg')
    ...:
    ...: e1 = cv2.getTickCount()
    ...: for i in range(5,49,2):
    ...:     img = cv2.medianBlur(img,i)
    ...: e2 = cv2.getTickCount()
    ...: t = (e2 - e1)/cv2.getTickFrequency()
    ...: print (t)
0.7363433

In [20]: import time
    ...: t1 = time.time()
    ...: for i in range(5,49,2):
    ...:     img = cv2.medianBlur(img,i)
    ...: t2 = time.time()
    ...: t3 = t2 - t1
    ...: print (t3)
0.6707360744476318
```

OpenCV'de Varsayılan Optimizasyon

OpenCV işlevlerinin çoğu, SSE2, AVX, vb. kullanılarak optimize edilmiştir.

Not: SSE2 (Streaming SIMD Extensions 2), Intel tarafından ilk kez 2000 yılında Pentium 4'ün ilk sürümüyle tanıtılan Intel SIMD (Tek Yönerge, Çoklu Veri) işlemci ek yönerge setlerinden biridir.

Not: Gelişmiş Vektör Uzantıları (AVX, Sandy Bridge Yeni Uzantılar olarak da bilinir), Intel tarafından Mart 2008'de önerilen ve ilk olarak Intel tarafından Sandy Bridge ile desteklenen Intel ve AMD mikroişlemciler için x86 komut seti mimarisinin uzantılarıdır.

Optimize edilmemiş kod da içerir. Dolayısıyla, sistemimiz bu özellikleri destekliyorsa, bunlardan yararlanmalıyız.

Derleme sırasında varsayılan olarak etkindir. Bu nedenle OpenCV, etkinleştirilmişse optimize edilmiş kodu çalıştırır, aksi takdirde optimize edilmemiş kodu çalıştırır.

Etkin/devre dışı olup olmadığını kontrol etmek için cv2.useOptimized() fonksiyonu ve etkinleştirmek/devre dışı bırakmak için cv2.setUseOptimized() fonksiyonu kullanılabilir.

OpenCV'de Varsayılan Optimizasyon

Aradaki farkı inceleyelim:

```
In [21]: cv2.useOptimized()
Out[21]: True

In [22]: %timeit res = cv2.medianBlur(img,49)
30.1 ms ± 457 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

In [23]: cv2.setUseOptimized(False)

In [24]: cv2.useOptimized()
Out[24]: False

In [25]: %timeit res = cv2.medianBlur(img,49)
34.2 ms ± 998 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

IPython'da Performansı Ölçme

Bazen iki benzer işlemin performansını karşılaştırmak gerektiğinde IPython'da, "%timeit" komutu kullanılabilir. Daha doğru sonuçlar elde etmek için kodu birkaç kez çalıştırır. Tek satır kodlarını ölçmek için uygundur.

Örneğin, aşağıdaki işlemlerinden hangisi daha hızlıdır?

```
x=5
%timeit y=x**2
%timeit y=x*x
```

```
z=np.uint8([5])
%timeit y=z*z
%timeit y=np.square(z)
```

IPython'da Performansı Ölçme

```
IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: x=5

In [2]: %timeit y=x**2
222 ns ± 34.8 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)

In [3]: %timeit y=x*x
47.3 ns ± 0.547 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)

In [4]: import numpy as np

In [5]: z=np.uint8([5])

In [6]: %timeit y=z*z
358 ns ± 14.7 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

In [7]: %timeit y=np.square(z)
370 ns ± 7.13 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

Python skaler işlemleri, Numpy skaler işlemlerinden daha hızlıdır.

Bu nedenle, bir veya iki eleman içeren işlemler için Python skaleri, Numpy dizilerinden daha iyidir.

Numpy, dizinin boyutu biraz daha büyük olduğunda avantaj sağlar.

IPython'da Performansı Ölçme

Python ve Numpy'nin maksimum performansından yararlanmak için birkaç teknik ve kodlama yöntemi vardır.

Burada dikkat edilmesi gereken en önemli şey, önce algoritmayı basit bir şekilde uygulamaya çalışmaktır.

Çalıştıktan sonra çalışma profilini çıkararak kod içerisindeki darboğazlar bulunup optimize edilebilir.

IPython'da Performansı Ölçme

- Python'da döngüleri mümkün olduğunca kullanmaktan kaçınılmalı, özellikle iç içe ikili/üçlü döngüler vb. doğal olarak yavaştırlar.
- Numpy ve OpenCV vektör işlemleri için optimize edildiğinden, algoritmayı/kodu mümkün olan maksimum ölçüde vektörleştirilmeli.
- Gerekmedikçe dizinin kopyaları yapılmamalı. Dizi kopyalama maliyetli bir işlemdir.
- Tüm bu işlemleri yaptıktan sonra bile, kod hala yavaşsa veya büyük döngülerin kullanılması kaçınılmazsa, daha hızlı hale getirmek için Cython gibi ek kütüphaneler kullanılabilir.

Kameradan Video Yakalama

Bazı durumlarda, kameradan canlı bir akış yakalamak gerekebilir.

OpenCV'de bunun için bir arayüz vardır.

Video çekmek için bir VideoCapture nesnesi oluşturmak gerekir.

Argümanı ise cihaz dizini veya bir video dosyasının adı olarak verilebilir.

Cihaz dizini, hangi kamerayı seçeceğinizi belirten sayıdır. Çünkü birden fazla kamera bulunuyor olabilir. Ancak normalde bir kamera bağlanacaktır. Bu yüzden sadece 0 (veya -1) verilebilir. 1 vb. diyerek (varsa) ikinci kamerayı seçebilirsiniz. Bundan sonrasında, kare kare yakalanır. Son olarak kameradan yakalama işlemi bitirilir.

Kameradan Video Yakalama

```python
import cv2

cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read() #frame frame okuma

    # Görüntüleme
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): #q tuşuna basılana kadar göstermeye devam eder
        break

cap.release()
cv2.destroyAllWindows()
```

cap.read() bir bool (Doğru/Yanlış – True/False) döndürür. Çerçeve doğru okunursa True olacaktır. Bu dönüş değerini kontrol ederek videonun alınma durumu öğrenilebilir. Bazen video yakalama başlamamış olabilir. Bu durumda, bu kod bir hata gösterir.

cap.isOpened() yöntemiyle başlatılıp başlatılmadığını kontrol edilebilir. Eğer False değeri dönerse, cap.open() kullanarak açılabilir.

Videonun bazı özelliklerine cap.get(propId) yöntemi kullanılarak erişilebilir. Burada propId, 0 ile 18 arasında bir sayıdır. Her sayı, videonun bir özelliğini (eğer o video için geçerliyse) belirtir. Bu değerlerden bazıları cap.set(propId, value) kullanılarak değiştirilebilir.

Örneğin, cap.get(3) ve cap.get(4) ile çerçeve genişliği ve yüksekliği kontrol edilebilir.
Gelen değeri 320x240 olarak değiştirmek için ret = cap.set(3.320) ve ret = cap.set(4,240) kullanılabilir.

Dosyadan Video Oynatma

Kameradan çekim yapmakla aynıdır, sadece kamera dizinini video dosyası adıyla değiştirmek yeterlidir. Ayrıca çerçeveyi görüntülerken cv2.waitKey() için uygun zamanı kullanılması gerekir. Çok az ise video çok hızlı, çok yüksek ise video yavaş olacaktır (Yani, videoları bu şekilde ağır çekimde görüntüleyebilirsiniz). Normal durumlarda **25 milisaniye** yeterlidir.

```
#%%
#Video Oynatma
cap = cv2.VideoCapture('university.mp4')

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

    cv2.imshow('frame',frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Video Kaydetme

Bunun için bir VideoWriter nesnesi oluşturulur. Çıktı dosyası adı belirtilir.

Ardından **FourCC** kodu belirtilir. **FourCC**; video codec bileşenini belirtmek için kullanılan 4 baytlık bir koddur. Mevcut kodların listesi fourcc.org'da bulunuyor. Platforma bağlıdır.

Daha sonra saniyedeki kare sayısı (fps) ve kare boyutu verilmelidir.

Sonuncusu ise isColor bayrağıdır. **True ise**, kodlayıcı renkli çerçeve bekler, **aksi takdirde** gri tonlamalı çerçeve ile çalışır.

```
#%%
#Video Kaydetme

cap = cv2.VideoCapture(0)

# Codec belirleyerek Video Objesi oluşturma
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,1)
        out.write(frame) # Frame yazma

        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

Mouse ile Çizim

İlk olarak, bir fare olayı (**Mouse event**) gerçekleştiğinde yürütülen bir fare geri çağırma işlevi (**Mouse callback**) oluşturuyoruz.

Fare olayları; sol düğme aşağı, sol düğme yukarı, sol düğme çift tıklama vb. ile ilgili herhangi bir şey olabilir.

Her fare olayı için koordinatları (x, y) verir. Bu etkinlik ve konumla ne istenirse yapılabilir. Mevcut tüm olayları listelemek için Python terminalinde şu komutlar yazılabilir:

```
In [4]: import cv2

In [5]: events = [i for i in dir(cv2) if 'EVENT' in i]

In [6]: print(events)
['EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'EVENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON',
'EVENT_FLAG_SHIFTKEY', 'EVENT_LBUTTONDBLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP', 'EVENT_MBUTTONDBLCLK',
'EVENT_MBUTTONDOWN', 'EVENT_MBUTTONUP', 'EVENT_MOUSEHWHEEL', 'EVENT_MOUSEMOVE', 'EVENT_MOUSEWHEEL', 'EVENT_RBUTTONDBLCLK',
'EVENT_RBUTTONDOWN', 'EVENT_RBUTTONUP']
```

Mouse ile Çizim

Bir fare geri çağırma işlevi oluşturma, her yerde aynı olan belirli bir biçime sahiptir.

Yalnızca işlevin yaptığı şeyde farklılık gösterir. Yani fare geri çağırma işlevi bir şey yapar.

Örneğin yandaki kodlar çift tıkladığımız yerde bir daire çizer.

```
import cv2
import numpy as np

# mouse callback
def draw_circle(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        cv2.circle(img,(x,y),100,(255,0,0),-1)

img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
```

Mouse ile Çizim

Çizim yapar gibi belirli alanları seçmek içinse:

```python
import cv2
import numpy as np

drawing = False # Mouse durumu için bir bool değeri
mode = True # dikdörtgen ve daire için mod. Mod için 'm' tuşu kullanılacak
ix,iy = -1,-1


# mouse callback fonksiyonu
def draw_circle(event,x,y,flags,param):
    global ix,iy,drawing,mode

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix,iy = x,y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == True:
            if mode == True:
                cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
            else:
                cv2.circle(img,(x,y),5,(0,0,255),-1)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        if mode == True:
            cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
        else:
            cv2.circle(img,(x,y),5,(0,0,255),-1)

img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)

while(1):
    cv2.imshow('image',img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    elif k == 27: #ESC
        break

cv2.destroyAllWindows()
```

Renk Paleti Olarak İzleme Çubuğu

Her zaman sabit renk istenmeyebilir. İstenirse belirtilen rengi gösteren basit bir uygulama oluşturulabilir.

Blue, Green, Red renklerinin her birini belirtmek için rengi ve üç hareket çubuğunu gösteren bir pencere ile izleme çubuğu kaydırınca buna bağlı olarak pencere rengi değişir.

Varsayılan olarak, başlangıç rengi Siyah olarak ayarlanabilir.

Renk Paleti Olarak İzleme Çubuğu

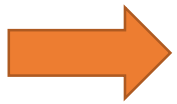
cv2.getTrackbarPos() fonksiyonu için;

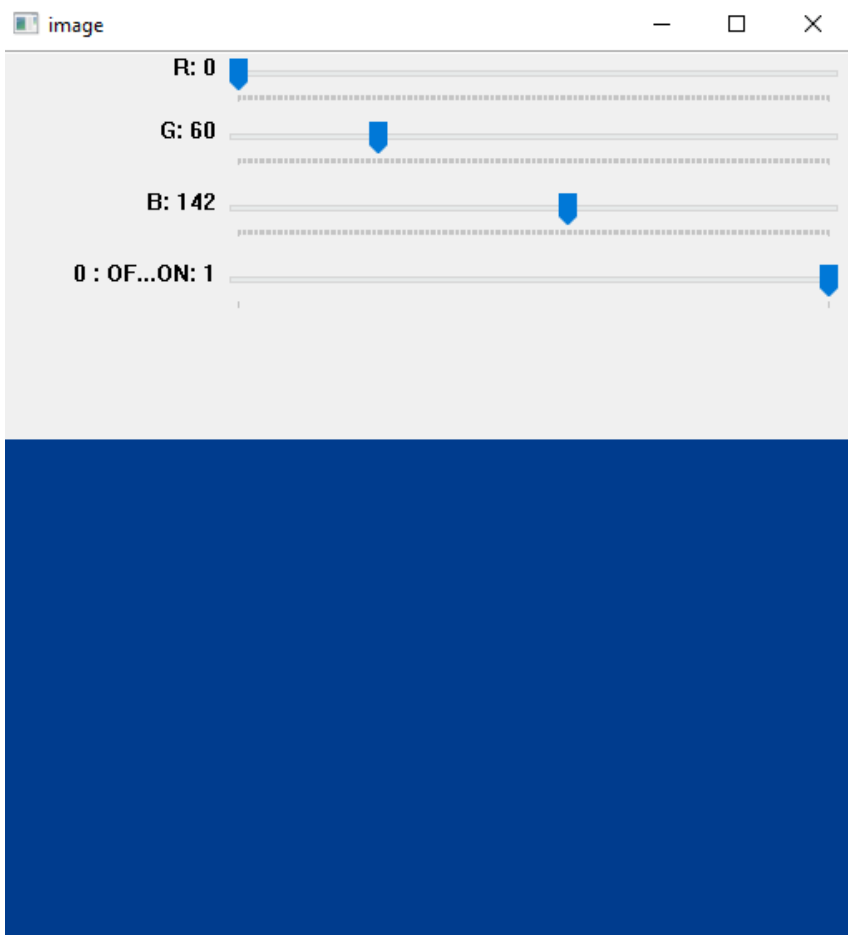
- ilk parametre izleme çubuğu adı,
- ikincisi eklendiği pencere adı,
- üçüncü parametre varsayılan değer,
- dördüncüsü maksimum değerdir.
- beşinci parametre izleme çubuğu değeri her değiştiğinde yürütülen işlev.

Bu işlevin her zaman izleme çubuğu konumu olan varsayılan bir argümanı vardır.

Renk Paleti Olarak İzleme Çubuğu

```python
import cv2
import numpy as np

def nothing(x):
    pass


img = np.zeros((300,512,3), np.uint8) # 300x512 çözünürlükte bir siyah görüntü
cv2.namedWindow('image')

# Trackbar oluşturma
cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

# kontrol için switch oluşturma
switch = '0 : OFF \n1 : ON'
cv2.createTrackbar(switch, 'image',0,1,nothing)

while(1):
    cv2.imshow('image',img)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

    # Trackbar pozisyonlarının okunması
    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')
    s = cv2.getTrackbarPos(switch,'image')

    if s == 0:
        img[:] = 0
    else:
        img[:] = [b,g,r]

cv2.destroyAllWindows()
```





Renk Uzayını Değiştirme

OpenCV'de 150'den fazla renk alanı dönüştürme yöntemi mevcuttur. En sık kullanılan iki tanesi:

MEVCUT	DÖNÜŞÜM	FONKSİYON
BGR	GRAY	cv2.COLOR_BGR2GRAY
BGR	HSV	cv2.COLOR_BGR2HSV

Renk dönüştürme için, bayrağın dönüştürme türünü belirlediği

cv2.cvtColor(input_image, flag)

fonksiyonu kullanılır.

Renk Alanlarını Değiştirme

```
In [3]: flags = [i for i in dir(cv2) if i.startswith('COLOR_')]

In [4]: print(flags)
```

['COLOR_BAYER_BG2BGR', 'COLOR_BAYER_BG2BGRA', 'COLOR_BAYER_BG2BGR_EA',
'COLOR_BAYER_BG2BGR_VNG', 'COLOR_BAYER_BG2GRAY', 'COLOR_BAYER_BG2RGB',
'COLOR_BAYER_BG2RGBA', 'COLOR_BAYER_BG2RGB_EA', 'COLOR_BAYER_BG2RGB_VNG',
'COLOR_BAYER_GB2BGR', 'COLOR_BAYER_GB2BGRA', 'COLOR_BAYER_GB2BGR_EA',
'COLOR_BAYER_GB2BGR_VNG', 'COLOR_BAYER_GB2GRAY', 'COLOR_BAYER_GB2RGB',
'COLOR_BAYER_GB2RGBA', 'COLOR_BAYER_GB2RGB_EA', 'COLOR_BAYER_GB2RGB_VNG',
'COLOR_BAYER_GR2BGR', 'COLOR_BAYER_GR2BGRA', 'COLOR_BAYER_GR2BGR_EA',
'COLOR_BAYER_GR2BGR_VNG', 'COLOR_BAYER_GR2GRAY', 'COLOR_BAYER_GR2RGB',
'COLOR_BAYER_GR2RGBA', 'COLOR_BAYER_GR2RGB_EA', 'COLOR_BAYER_GR2RGB_VNG',
'COLOR_BAYER_RG2BGR', 'COLOR_BAYER_RG2BGRA', 'COLOR_BAYER_RG2BGR_EA',
'COLOR_BAYER_RG2BGR_VNG', 'COLOR_BAYER_RG2GRAY', 'COLOR_BAYER_RG2RGB',
'COLOR_BAYER_RG2RGBA', 'COLOR_BAYER_RG2RGB_EA', 'COLOR_BAYER_RG2RGB_VNG',
'COLOR_BGR2BGR555', 'COLOR_BGR2BGR565', 'COLOR_BGR2BGRA', 'COLOR_BGR2GRAY',
'COLOR_BGR2HLS', 'COLOR_BGR2HLS_FULL', 'COLOR_BGR2HSV', 'COLOR_BGR2HSV_FULL',
'COLOR_BGR2LAB', 'COLOR_BGR2LUV', 'COLOR_BGR2Lab', 'COLOR_BGR2Luv', 'COLOR_BGR2RGB',
'COLOR_BGR2RGBA', 'COLOR_BGR2XYZ', 'COLOR_BGR2YCR_CB', 'COLOR_BGR2YCrCb',
'COLOR_BGR2YUV', 'COLOR_BGR2YUV_I420', 'COLOR_BGR2YUV_IYUV', 'COLOR_BGR2YUV_YV12',
'COLOR_BGR5552BGR', 'COLOR_BGR5552BGRA', 'COLOR_BGR5552GRAY', 'COLOR_BGR5552RGB',
'COLOR_BGR5552RGBA', 'COLOR_BGR5652BGR', 'COLOR_BGR5652BGRA', 'COLOR_BGR5652GRAY',
'COLOR_BGR5652RGB', 'COLOR_BGR5652RGBA', 'COLOR_BGRA2BGR', 'COLOR_BGRA2BGR555',
'COLOR_BGRA2BGR565', 'COLOR_BGRA2GRAY', 'COLOR_BGRA2RGB', 'COLOR_BGRA2RGBA',
'COLOR_BGRA2YUV_I420', 'COLOR_BGRA2YUV_IYUV', 'COLOR_BGRA2YUV_YV12',
'COLOR_BayerBG2BGR', 'COLOR_BayerBG2BGRA', 'COLOR_BayerBG2BGR_EA',
'COLOR_BayerBG2BGR_VNG', 'COLOR_BayerBG2GRAY', 'COLOR_BayerBG2RGB',
'COLOR_BayerBG2RGBA', 'COLOR_BayerBG2RGB_EA', 'COLOR_BayerBG2RGB_VNG',
'COLOR_BayerGB2BGR', 'COLOR_BayerGB2BGRA', 'COLOR_BayerGB2BGR_EA',
'COLOR_BayerGB2BGR_VNG', 'COLOR_BayerGB2GRAY', 'COLOR_BayerGB2RGB',
'COLOR_BayerGB2RGBA', 'COLOR_BayerGB2RGB_EA', 'COLOR_BayerGB2RGB_VNG',
'COLOR_BayerGR2BGR', 'COLOR_BayerGR2BGRA', 'COLOR_BayerGR2BGR_EA',
'COLOR_BayerGR2BGR_VNG', 'COLOR_BayerGR2GRAY', 'COLOR_BayerGR2RGB',
'COLOR_BayerGR2RGBA', 'COLOR_BayerGR2RGB_EA', 'COLOR_BayerGR2RGB_VNG',
'COLOR_BayerRG2BGR', 'COLOR_BayerRG2BGRA', 'COLOR_BayerRG2BGR_EA',
'COLOR_BayerRG2BGR_VNG', 'COLOR_BayerRG2GRAY', 'COLOR_BayerRG2RGB',
'COLOR_BayerRG2RGBA', 'COLOR_BayerRG2RGB_EA', 'COLOR_BayerRG2RGB_VNG',
'COLOR_COLORCVT_MAX', 'COLOR_GRAY2BGR', 'COLOR_GRAY2BGR555', 'COLOR_GRAY2BGR565',
'COLOR_GRAY2BGRA', 'COLOR_GRAY2RGB', 'COLOR_GRAY2RGBA', 'COLOR_HLS2BGR',
'COLOR_HLS2BGR_FULL', 'COLOR_HLS2RGB', 'COLOR_HLS2RGB_FULL', 'COLOR_HSV2BGR',
'COLOR_HSV2BGR_FULL', 'COLOR_HSV2RGB', 'COLOR_HSV2RGB_FULL', 'COLOR_LAB2BGR',
'COLOR_LAB2LBGR', 'COLOR_LAB2LRGB', 'COLOR_LAB2RGB', 'COLOR_LBGR2LAB',
'COLOR_LBGR2LUV', 'COLOR_LBGR2Lab', 'COLOR_LBGR2Luv', 'COLOR_LRGB2LAB',
'COLOR_LRGB2LUV', 'COLOR_LRGB2Lab', 'COLOR_LRGB2Luv', 'COLOR_LUV2BGR',
'COLOR_LUV2LBGR', 'COLOR_LUV2LRGB', 'COLOR_LUV2RGB', 'COLOR_Lab2BGR', 'COLOR_Lab2LBGR',
'COLOR_Lab2LRGB', 'COLOR_Lab2RGB', 'COLOR_Luv2BGR', 'COLOR_Luv2LBGR', 'COLOR_Luv2LRGB',
'COLOR_Luv2RGB', 'COLOR_M_RGBA2RGBA', 'COLOR_RGB2BGR', 'COLOR_RGB2BGR555',
'COLOR_RGB2BGR565', 'COLOR_RGB2BGRA', 'COLOR_RGB2GRAY', 'COLOR_RGB2HLS',
'COLOR_RGB2HLS_FULL', 'COLOR_RGB2HSV', 'COLOR_RGB2HSV_FULL', 'COLOR_RGB2LAB',
'COLOR_RGB2LUV', 'COLOR_RGB2Lab', 'COLOR_RGB2Luv', 'COLOR_RGB2RGBA', 'COLOR_RGB2XYZ',
'COLOR_RGB2YCR_CB', 'COLOR_RGB2YCrCb', 'COLOR_RGB2YUV', 'COLOR_RGB2YUV_I420',
'COLOR_RGB2YUV_IYUV', 'COLOR_RGB2YUV_YV12', 'COLOR_RGBA2BGR', 'COLOR_RGBA2BGR555',
'COLOR_RGBA2BGR565', 'COLOR_RGBA2BGRA', 'COLOR_RGBA2GRAY', 'COLOR_RGBA2M_RGBA',
'COLOR_RGBA2RGB', 'COLOR_RGBA2YUV_I420', 'COLOR_RGBA2YUV_IYUV',
'COLOR_RGBA2YUV_YV12', 'COLOR_RGBA2mRGBA', 'COLOR_XYZ2BGR', 'COLOR_XYZ2RGB',
'COLOR_YCR_CB2BGR', 'COLOR_YCR_CB2RGB', 'COLOR_YCrCb2BGR', 'COLOR_YCrCb2RGB',
'COLOR_YUV2BGR', 'COLOR_YUV2BGRA_I420', 'COLOR_YUV2BGRA_IYUV',
'COLOR_YUV2BGRA_NV12', 'COLOR_YUV2BGRA_NV21', 'COLOR_YUV2BGRA_UYNV',
'COLOR_YUV2BGRA_UYVY', 'COLOR_YUV2BGRA_Y422', 'COLOR_YUV2BGRA_YUNV',
'COLOR_YUV2BGRA_YUY2', 'COLOR_YUV2BGRA_YUYV', 'COLOR_YUV2BGRA_YV12',
'COLOR_YUV2BGRA_YVYU', 'COLOR_YUV2BGR_I420', 'COLOR_YUV2BGR_IYUV',
'COLOR_YUV2BGR_NV12', 'COLOR_YUV2BGR_NV21', 'COLOR_YUV2BGR_UYNV',
'COLOR_YUV2BGR_UYVY', 'COLOR_YUV2BGR_Y422', 'COLOR_YUV2BGR_YUNV',
'COLOR_YUV2BGR_YUY2', 'COLOR_YUV2BGR_YUYV', 'COLOR_YUV2BGR_YV12',
'COLOR_YUV2BGR_YVYU', 'COLOR_YUV2GRAY_420', 'COLOR_YUV2GRAY_I420',
'COLOR_YUV2GRAY_IYUV', 'COLOR_YUV2GRAY_NV12', 'COLOR_YUV2GRAY_NV21',
'COLOR_YUV2GRAY_UYNV', 'COLOR_YUV2GRAY_UYVY', 'COLOR_YUV2GRAY_Y422',
'COLOR_YUV2GRAY_YUNV', 'COLOR_YUV2GRAY_YUY2', 'COLOR_YUV2GRAY_YUYV',
'COLOR_YUV2GRAY_YV12', 'COLOR_YUV2GRAY_YVYU', 'COLOR_YUV2RGB',
'COLOR_YUV2RGBA_I420', 'COLOR_YUV2RGBA_IYUV', 'COLOR_YUV2RGBA_NV12',
'COLOR_YUV2RGBA_NV21', 'COLOR_YUV2RGBA_UYNV', 'COLOR_YUV2RGBA_UYVY',
'COLOR_YUV2RGBA_Y422', 'COLOR_YUV2RGBA_YUNV', 'COLOR_YUV2RGBA_YUY2',
'COLOR_YUV2RGBA_YUYV', 'COLOR_YUV2RGBA_YV12', 'COLOR_YUV2RGBA_YVYU',
'COLOR_YUV2RGB_I420', 'COLOR_YUV2RGB_IYUV', 'COLOR_YUV2RGB_NV12',
'COLOR_YUV2RGB_NV21', 'COLOR_YUV2RGB_UYNV', 'COLOR_YUV2RGB_UYVY',
'COLOR_YUV2RGB_Y422', 'COLOR_YUV2RGB_YUNV', 'COLOR_YUV2RGB_YUY2',
'COLOR_YUV2RGB_YUYV', 'COLOR_YUV2RGB_YV12', 'COLOR_YUV2RGB_YVYU',
'COLOR_YUV420P2BGR', 'COLOR_YUV420P2BGRA', 'COLOR_YUV420P2GRAY',
'COLOR_YUV420P2RGB', 'COLOR_YUV420P2RGBA', 'COLOR_YUV420SP2BGR',
'COLOR_YUV420SP2BGRA', 'COLOR_YUV420SP2GRAY', 'COLOR_YUV420SP2RGB',
'COLOR_YUV420SP2RGBA', 'COLOR_YUV420p2BGR', 'COLOR_YUV420p2BGRA',
'COLOR_YUV420p2GRAY', 'COLOR_YUV420p2RGB', 'COLOR_YUV420p2RGBA',
'COLOR_YUV420sp2BGR', 'COLOR_YUV420sp2BGRA', 'COLOR_YUV420sp2GRAY',
'COLOR_YUV420sp2RGB', 'COLOR_YUV420sp2RGBA', 'COLOR_mRGBA2RGBA']

Nesne İzleme

BGR görüntüsünü HSV'ye dönüştürme işlemi öğrenildiğine göre, bu işlem renkli bir nesneyi çıkarmak için kullanılabilir.

HSV'de bir rengi temsil etmek RGB renk uzayından daha kolaydır. Mavi renkli bir nesneyi çıkartmak için bir uygulama yapacak olursak yöntem:

1. Videonun her karesi alınır,
2. BGR'den HSV renk alanına dönüştürme işlemi yapılır,
3. HSV görüntüsü belirlenen mavi renk aralıkları için eşiklenir,
4. Nesne içindeki belirlenen aralıktaki mavi renk görüntü alanları görüntülenir (ya da istenilen işlem yapılır)

Nesne İzleme

```python
import cv2
import numpy as np

cap = cv2.VideoCapture('university.mp4')

while(1):

    #Video'dan frame frame görüntü alma
    _, frame = cap.read()

    # BGR -> HSV dönüştürme
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # HSV içinde Mavi Renk için eşik değerleri belirleme
    lower_blue = np.array([110,50,50])
    upper_blue = np.array([130,255,255])

    # HSV için tanımlanan mavi renk değerlerini maske oluşturma
    mask = cv2.inRange(hsv, lower_blue, upper_blue)

    # Bitwise-AND ile görüntüyü maskeleme
    res = cv2.bitwise_and(frame, frame, mask=mask)

    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()
```

İlk etapta görüntüde bazı gürültüler olması normal. Bunlar daha sonra daha ayrıntılı işlemlerle temizlenebilir.

Görüntülerin Geometrik Dönüşümleri

cv2.getPerspectiveTransform

OpenCV'de, dönüştürme işlemi için kullanılabilecek olan cv2.warpAffine ve cv2.warpPerspective olmak üzere iki dönüşüm fonksiyonu vardır. cv2.warpAffine 2x3 dönüşüm matrisi alırken cv2.warpPerspective girdi olarak 3x3 dönüşüm matrisi alır.

Ölçekleme

Ölçekleme işlemi görüntünün yeniden boyutlandırılmasıdır. OpenCV'de, bu amaç için cv2.resize() fonksiyonu tanımlanmıştır. Resmin boyutu manuel olarak belirlenebilir veya ölçekleme faktörüyle yapılabilir. Farklı interpolasyon yöntemleri kullanılır.

Görüntülerin Geometrik Dönüşümleri

Interpolasyon, varolan sayısal değerleri kullanarak, boş noktalardaki değerlerin tahmin edilmesidir. Tercih edilen interpolasyon yöntemleri, küçültme için cv2.INTER_AREA ve yakınlaştırma için cv2.INTER_CUBIC (yavaş) ve cv2.INTER_LINEAR'dır. Varsayılan olarak, tüm yeniden boyutlandırma amaçları için kullanılan interpolasyon yöntemi cv2.INTER_LINEAR'dır.

```
import cv2
img = cv2.imread('campus.jpg')

res = cv2.resize(img, None, fx=2, fy=2, interpolation = cv2.INTER_CUBIC)

cv2.imshow('Orjinal',img)
cv2.imshow('Resized',res)
cv2.waitKey(0)
cv2.destroyAllWindows()


#Veya

height, width = img.shape[:2]
res = cv2.resize(img,(2*width, 2*height), interpolation = cv2.INTER_CUBIC)
cv2.imshow('Orjinal',img)
cv2.imshow('Resized',res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Görüntülerin Geometrik Dönüşümleri

Çevirme

Bu işlemde görüntüler (x, y) koordinatlarına göre belirli ölçülerde kaydırılır. Ölçülerle matris işlemi için: (t_x , t_y)

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

Bir Numpy dizisi ve **cv2.warpAffine()** fonksiyonu yardımıyla, bu işlem gerçekleştirilebilir.

Görüntülerin Geometrik Dönüşümleri

Döndürme

Bir görüntünün bir *teta* açısında döndürülmesi, formun dönüşüm matrisi ile elde edilir.

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Ancak OpenCV, tercih edilen herhangi bir yerde döndürme gerçekleştirebilmek için ayarlanabilir bir dönüş merkezi ile ölçeklendirilmiş dönüş sağlar. Değiştirilmiş dönüşüm matrisi şu şekilde verilir:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot \text{center.x} - \beta \cdot \text{center.y} \\ -\beta & \alpha & \beta \cdot \text{center.x} + (1-\alpha)\cdot \text{center.y} \end{bmatrix}$$

burada

$$\alpha = \text{scale} \cdot \cos\theta,$$
$$\beta = \text{scale} \cdot \sin\theta$$

Görüntülerin Geometrik Dönüşümleri

Döndürme

Bu dönüşüm matrisini bulmak için OpenCV, cv2.getRotationMatrix2D adlı bir fonksiyon sağlar.

```
#Döndürme

img3 = cv2.imread('img/kku_logo.jpeg',0) #grayscale
rows, cols = img3.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
res3 = cv2.warpAffine(img3,M,(cols,rows))

cv2.imshow('img3',res3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Görüntülerin Geometrik Dönüşümleri

Afin Dönüşümü

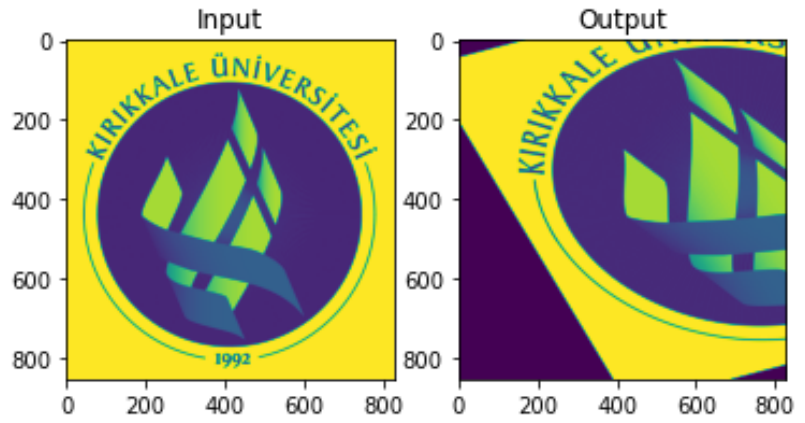
Afin dönüşümde, orijinal görüntüdeki tüm paralel çizgiler çıkış görüntüsünde hala paralel kalır.

Dönüşüm matrisini bulmak için, giriş görüntüsünden üç noktaya ve çıkış görüntüsündeki karşılık gelen konumlarına ihtiyaç vardır. Ardından cv2.getAffineTransform, cv2.warpAffine'e dönüştürülerek 2x3'lük bir matris oluşturulur.

Görüntülerin Geometrik Dönüşümleri

Afin Dönüşümü

```python
#Afin Dönüşümü

import matplotlib.pyplot as plt

img4 = cv2.imread('img/kku_logo.jpeg',0)
rows,cols = img4.shape

pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])

M = cv2.getAffineTransform(pts1, pts2)

afn = cv2.warpAffine(img4, M, (cols,rows))

plt.subplot(121),plt.imshow(img4),plt.title('Input')
plt.subplot(122),plt.imshow(afn),plt.title('Output')
plt.show()

#Veya

cv2.imshow('Input',img4)
cv2.imshow('Output',afn)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Görüntülerin Geometrik Dönüşümleri

Perspektif Dönüşümü

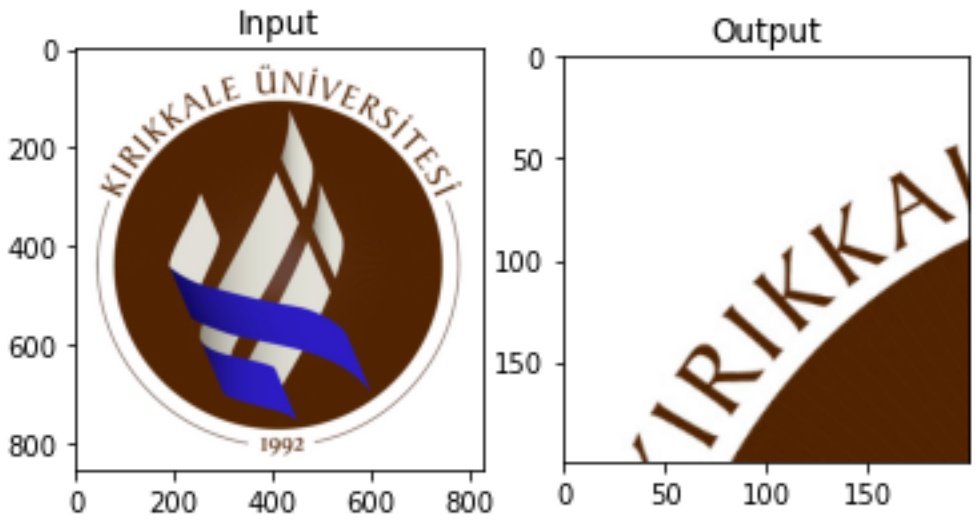
Perspektif dönüşümü için *3x3* dönüşüm matrisine ihtiyaç vardır. Düz çizgiler, dönüşümden sonra da düz kalacaktır. Bu dönüşüm matrisini bulmak için giriş görüntüsünde 4 noktaya ve çıkış görüntüsünde bunlara karşılık gelen noktalara ihtiyaç vardır.

Bu 4 noktadan 3 tanesi doğrusal olmamalıdır. Daha sonra dönüşüm matrisi cv2.getPerspectiveTransform işleviyle bulunabilir. Ardından bu 3x3 dönüşüm matrisiyle cv2.warpPerspective fonksiyonu uygulanır.

Görüntülerin Geometrik Dönüşümleri

Perspektif Dönüşümü

```
#Perspektif Dönüşümü

img5 = cv2.imread('img/kku_logo.jpeg')
rows, cols, ch = img5.shape

pts1 = np.float32([[10,10],[275,15],[20,270],[275,280]])
pts2 = np.float32([[0,0],[200,0],[0,200],[200,200]])

M = cv2.getPerspectiveTransform(pts1,pts2)

prs = cv2.warpPerspective(img5,M,(200,200))

plt.subplot(121),plt.imshow(img5),plt.title('Input')
plt.subplot(122),plt.imshow(prs),plt.title('Output')
plt.show()
```



Kaynaklar

- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_optimization/py_optimization.html#optimization-techniques, Erişim T.: 03.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html#display-video, Erişim T.: 04.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_mouse_handling/py_mouse_handling.html#mouse-handling, Erişim T.: 04.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_trackbar/py_trackbar.html#trackbar, Erişim T.: 06.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html#converting-colorspaces, Erişim T.: 06.09.2021.
- OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html#geometric-transformations, Erişim T.: 06.09.2021.