Python ve OpenCV
Görüntü İşleme

Yılmaz ALACA | Emine YALÇIN

Yılmaz ALACA

2001 Adana doğumlu yazar. İlk ve orta öğrenimini Gazi İlk ve Ortaöğretim Okulu'nda tamamlamıştır. Orta okul döneminde C++ yazılım dili ile tanışmış ve oyun geliştirme üzerine çalışmalar gerçekleştirmiştir. Lise öğrenimini Şehit Temel Cingöz Anadolu Lisesi'nde tamamlayan yazar, Lise döneminde TÜBİTAK dahil olmak üzere çeşitli Robotik ve Kodlama şenliklerine katılmış, işitme engellilerin hayatını kolaylaştırmak, fast food zincirleri için garson robot ve yüksek ateş hastalığından çocukları korumak amacıyla projeler geliştirmiştir.

Üniversite öğrenimine Adana Alparslan Türkeş Bilim ve Teknoloji Üniversitesi'nde Elektrik ve Elektronik Mühendisliği bölümüne başlamış, kurduğu İNOVASYON Topluluğu'nda C++, Python ve Gömülü Sistemler üzerine eğitimler vermiştir.

Şu an üniversite son sınıf öğrencisi olan yazar, okuduğu dönem boyunca Udemy'de eğitmenliğe başlamış ve 2,5 yılda 80 farklı ülkeden 35 Bin'den fazla öğrenciye ulaşmıştır. Udemy'de C++, Python, Gömülü Sistemler, Makine Öğrenmesi, Derin Öğrenme, Oyun Geliştirme ve Görüntü İşleme konularında eğitimleri bulunmaktadır. 2022-2023 yılları arasında Dene Yap Türkiye'de Robotik ve Kodlama eğitmenliği yapmıştır.

İlk kitabı, Şubat 2023 yılında, Üşengeçler için Arduino ismiyle yayınlanmıştır. İkinci kitabı, Mayıs 2023 yılında Python ile Gömülü Sistemlerde Arduino Arayüz Geliştirme ismiyle yayınlanmıştır. SkillShare ve Tutorialspoint online eğitim sitelerinde Raspberry Pi ve Bilgisayarlı Görü (Computer Vision) üzerine eğitimleri bulunmaktadır.

Şu anda Bilişim School bünyesinde içerik koordinatörü olarak çalışmaktadır.

15 Şubat 2022 tarihinde hayata gözlerini yuman kıymetli dedem Yılmaz ALACA'ya ithafen.

Yılmaz ALACA

instagram.com/yilmazalacaa

twitter.com/alacayilmaz01

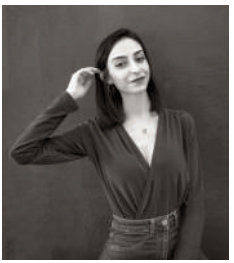
linkedin.com/in/yilmazalaca

udemy.com/user/yilmaz-alaca

youtube.com/@yilmazalaca

yilmazalaca01@gmail.com

Emine YALÇIN



2001 Gaziantep doğumlu yazar, ilk ve orta öğrenimini; Nuriye Zekeriya Kına İlk ve Ortaöğretim Okulu'nda, Liseyi; İstanbul Gazianteplileri Anadolu Lisesi'nde dereceyle tamamlamıştır.

Lisans eğitimine Adana Alparslan Türkeş Bilim ve Teknoloji Üniversitesi'nde Havacılık ve Uzay Mühendisliği son sınıf öğrencisi olarak devam etmektedir.

Lisans eğitim döneminde; Çukurova Teknokent Itouch Yazılım Firması'nda stajını tamamlamış ve Temsa Global Sanayi ve Tic. A.Ş.'de AR-GE bölümünde staja kabul almıştır. Okurken aynı zamanda çeşitli kurumlarda İleri Matematik, İngilizce, Robotik ve Kodlama alanlarında eğitmenlik yapmaktadır. Ayrıca, Deneyap Türkiye Atölyelerinde Elektronik Programlama ve Nesnelerin İnterneti, Yazılım Teknolojileri alanlarında eğitmenlik yapmaktadır.

İlk eseri olan, Üşengeçler için Arduino Şubat 2023'te yayınlanmıştır. Yazar; 2020 yılından bu yana çeşitli programlama dilleri (Python, C++, MATLAB) ve gömülü sistemler (Arduino, Raspberry Pi, PLC) üzerinde çalışmalar yapmakta ve projeler geliştirmektedir.

Bununla birlikte tasarım alanında da çalışmalar yapmaktadır. Autocad ve Solidworks Programlarını kullanarak 3 boyutlu çizimler ve tasarımlar yapmaktadır. Aynı zamanda, çeşitli analiz programlarını (Ansys) kullanarak Aerodinamik analizler üzerine çalışmalar yapmaktadır. Roketler ve hava araçları üzerine çalışmakta ve projelerde mentorluk yapmaktadır.

Teşekkür

Beni her zaman motive eden, yol gösteren, kendi potansiyelimi keşfetmemde ve kendimi göstermemde bana fırsat sağlayan; hayatımdaki tüm değerli insanlara ve bu kitabı alan tüm okurlarımıza sonsuz teşekkürlerimi sunarım.

Emine YALÇIN

instagram.com/emiine.yalcn

linkedin.com/in/emineyalcn

emineyalcin015@gmail.com

ÖNSÖZ

2021 senesinde başlayan ve hala devam eden yazılım serüvenime ben de Python öğrenerek başlamıştım. Yazılım ile tanışma ve projelerimi uygulamaya alma serüvenim Yazılım Eğitmenim Yılmaz ALACA ile başladı. İlk başlarda kurslar ve videolar izleyerek öğrenmeye başladım fakat, hep konuları her detayına kadar öğrenebileceğim bir kaynak kitabın eksikliğini yaşadım. Kitap yazmaya başladığımızda yaşamış olduğum bu eksikliği kapatmak, yeni öğrenmeye başlayanların her detayı kolaylıkla öğrenebilmesini sağlamak için hem **Üşengeçler için Arduino** kitabımızda hem de bu kitabımızda olabildiğince anlaşılır ve detaylı bir anlatım sağlamaya çalıştık.

Kitap içerisinde ilk olarak Python programlama dilinin ne olduğunu ve Python temel bilgilerini öğreneceksiniz. Temel bilgileri anlatırken özellikle konu anlatımlarından hemen sonra örnek kodlar ve uygulamalar yaparak sizler için anlaşılır ve akılda kalıcı bir öğretim sağladık. Devamında ise çeşitli görüntü işleme projelerini geliştirebileceğiniz Python ile görüntü işleme konusunda çokça örnek ve uygulamalar sayesinde detaylıca bilgi sahibi olacaksınız. Son olarak nesne tespiti işlemlerinin nasıl yapıldığı konusunu öğrenerek Python ile nesne tespiti işlemlerini kolaylıkla yapabileceksiniz.

Özellikle Python ile görüntü işleme konusunda kitap yazmamızın en büyük sebebi, bu konuda çok fazla projelerin yapılıyor olması ve hayatımızın artık her alanında görüntü işleme ile karşılaşıyor olmamızdı. Telefonu açarken yüz tanıma ile giriş sağlayabilmemiz, hastanelerde tıbbi cihazlardaki görüntüleme teknolojilerinde ya da otomobillerde bulunan şerit takibi sistemlerinde görüntü işleme konusundan yararlanılmaktadır. Bu durumda yazılımcıların gözdesi haline gelen görüntü işleme konusunda öğrenme isteği de artmıştır. Kitabımız ile görüntü işleme konusunda bilgi sahibi olup kendinizi geliştirerek örnek uygulamalar gerçekleştirebilirsiniz.

6 Şubat 2023 Kahramanmaraş / Gaziantep depremi olarak bildiğimiz ama maalesef 11 ilimizi (Kahramanmaraş, Kilis, Diyarbakır, Adana, Osmaniye, Gaziantep, Şanlıurfa, Adıyaman, Malatya, Hatay, Elazığ) etkileyen bu büyük afedi yaşadıktan sonra bozulan psikolojimizi ancak ve ancak bir şeyler üreterek ve kendimizi geliştirerek aşacağımıza inandığımız için ilk kitabımızın hemen ardından, **Python ve OpenCV ile Görüntü İşleme** kitabımızı oluşturmaya karar verdik.

Büyük bir sabırla ve emekle, yeri geldiğinde motivasyon kaybı yaşadığımızda boş boş ekrana bakarak, daha sonra motivasyon sağlayıp saatlerce durmadan yazarak, kitabımızı tamamladık. Unutmamalıyız ki karşılaştığımız her türlü engel ve problemlerden kendimizi motive ederek bir başkasına ihtiyaç duymadan ve vazgeçmeden kurtulabiliriz. Bu yüzden bir işe kalkışırken çıkabilecek engellerin de farkında olarak tam motivasyon ve özgüvenle, her türlü engeli aşacağınıza inanarak başlamalısınız. Her zaman söylediğimiz gibi Hayallerinizden VAZGEÇMEYİN! Önsözümü Ulu Önder Mustafa Kemal Atatürk'ün önemli bir sözüyle tamamlamak isterim.

Yerinde duran, geriye gidiyor demektir... İleri, daima ileri!

1938 ∞ Mustafa Kemal ATATÜRK

2023, Emine YALÇIN

Gaziantep - Türkiye

Kitaba dair örnek uygulama ve kaynak kodları aşağıdaki linkten indirebilirsiniz.

https://l24.im/KVrXyP

Değerli okurlarım kitabımıza başlamadan önce bir ricam olacak.

Kitapta hata gördüğünüzde hemen kızmayın, sizlerin de desteğiyle hatalarımızı ve eksikliklerimizi tarafıma bildirmeniz durumunda bir sonraki baskıda düzeltilecektir. :)

İçindekiler

1

Temel Python Bilgisi

Bu Bölümde

Bu bölümde, sizlerle birlikte Python yazılım dili serüvenine başlayacağız. Böylelikle, görüntü işleme kavramına giriş yapmadan önce Python yazılım dilinin temellerini detaylı bir şekilde öğrenecek ve örneklerle kendinizi geliştirebileceksiniz.

Kitabımızın bu bölümü tüm temel bilgileri kapsamaktadır. Kitap içerisinde OOP (Object Oriented Programming - Nesne Yönelimli Programlama) konusuna yer vermedik. Çünkü bu, kitabımızın kapsam alanının dışına çıkmasına sebep olurdu.

Haydi serüvenimize başlayalım. :)

PYTHON NEDİR?

1991 yılında Guido Van Rossum tarafından geliştirilmiş bir programlama dilidir Python. Kullanımı diğer dillere oranla oldukça basittir. Gerek sağladığı özgürlük gerekse syntax kurallarının kullanım kolaylığı açısından da popüler bir yazılım dilidir. Python, nesne yönelimli ve yüksek seviyeli bir yazılım dilidir. Bilindiği üzere C ve C++ gibi diller derlenebilir dillerken, Python yorumlanabilir bir programlama dilidir. Yani kodumuzda bir hata var ise hata okunana kadarki kısım çalışırken hatanın olduğu satıra gelince hata alırız ve böylelikle hatayı kolayca bulup düzeltiriz. Python, her türlü bilgisayar işletim sistemlerinde (Linux, Mac, Windows gibi) çalışabilir. Bizler Python programlama dili ile birçok karmaşık projenin yazılımını gerçekleştirebiliriz. Python yazılım dilinin kullanıldığı alanlar arasında; görüntü işleme, web programlama, güvenlik uygulamaları, arayüz geliştirme, veri bilimi, oyun geliştirme ve makine öğrenmesi yer almaktadır.

NEDEN PYTHON ÖĞRENMELİYİZ?

Python günümüz programlama dilleri arasında en popüler diller arasında yer almaktadır. Python söz dizimi oldukça kolay bir dildir ve bu kolaylığı açısından diğer dillere kıyasla bizlere zamandan tasarruf sağlamaktadır. Öğrenmesi kolay ve iş imkânı oldukça geniş olduğundan birçok yazılımcı tarafından tercih edilmektedir. Günümüzde Python kullanan kuruluşlar arasında; Google, NASA, Amazon, Facebook, Instagram ve Wikipedia gibi kurumlar yer almaktadır. Python çok fazla kütüphaneye sahiptir ve yapılmak istenen işlemler için bu kütüphaneler bizlere faydalı olmaktadır. Kısacası Python öğrenmesi oldukça kolay ve başlangıç için uygun bir dildir.

PYTHON KURULUMU

Python'da kodlarımızı çalıştırabilmemiz için öncelikle bilgisayara kurulumunu gerçekleştirmemiz gerekmektedir. Haydi hep beraber adım adım Python' un kurulumunu gerçekleştirelim.

1. Adım

Arama motoruna "Python" yazıp aratalım.

2. Adım

Arama motorundaki ilk seçeneğe yani Python' un kendi sitesine girelim.

www.python.org

3. Adım

Sayfa üzerinden karşımıza çıkan **Download** seçeneğine tıklayalım.

4. Adım

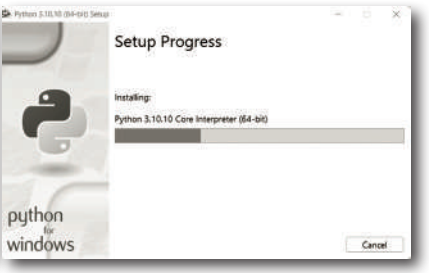
İndirmek istediğimiz sürüme tıklayıp indirme işlemini gerçekleştirelim.

5. Adım

İndirilen dosyayı açıp, seçilmiş seçeneği ve **Add python.exe to PATH** seçimlerini işaretleyerek, **Install now** seçeneğine tıklayıp kurulumu gerçekleştirelim.

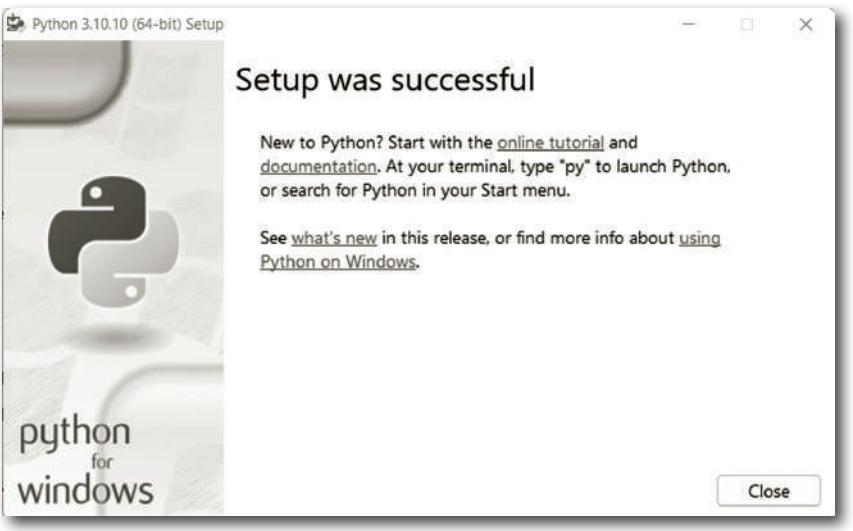
6. Adım

Install now seçeneğine basıp kurulumunu gerçekleştiririz.



7. Adım

Tebrikler, kurulum başarıyla gerçekleşti.



Kodlarımızı incelediğimizde görüldüğü gibi 2 farklı değişken oluşturduk ve bu değişkenlere `input` fonksiyonunu atadık. Metinsel veri oluşturduk ve hangi bilginin girilmesi gerektiğine `dair` kullanıcıyı bilgilendirdik. Son olarak çıktı alma fonksiyonumuzla kullanıcıyı selamladık. Bu şekilde kodlarımızı çalıştırdığımızda görüntüdeki gibi bir sonuçla karşılaşacağız.

```
Lutfen ismini gir:Ozgur
Lutfen soyadini gir:YALCIN
Merhaba: Ozgur YALCIN
PS D:\tt>
```

Böylesi daha güzel oldu değil mi :)?

Standart girdi fonksiyonu ile girdiğimiz verilerin string olarak tanımlandığını belirtmiştik. Yani örneklerimizde bulunan `isim` ve `soyad` değişkenleri otomatik bir şekilde string olarak algılanıyor.

Peki, `50` değerini girersek ne olur?

Bu değerde string olarak tanımlanmaktadır. Örneğin: hesap makinesi uygulaması geliştirmek istiyoruz. Doğrudan `sayı` değişkenlerine girdi fonksiyonuyla değer girilirse hata ile karşılaşırız. Bu durumu çözmek için de tip dönüşüm yapılarını kullanmamız gerekiyor. Bu sayede, string veriyi sayısal verilere dönüştürebilir ve üzerinde işlemler yapabiliriz.

TİP DÖNÜŞÜMLERİ

Python'da tip dönüşümleri, bir veri türünden başka bir veri türüne dönüşüm yapmaktır. Bu dönüşümler, verileri doğru formatta kullanabilmek veya işleyebilmek için kullanılmaktadır. Tip dönüşümlerini kullanıcıdan girdi almadan önce kullanmak çok önemlidir. Örneğin: Kullanıcıdan sayısal bir değer almak istiyorsak `input()` fonksiyonundan önce uygun **tip dönüşüm** fonksiyonunu kullanmamız gerekir. Aksi takdirde girilen değeri, Python string bir ifade olarak algılayıp sayısal işlemleri doğru bir şekilde gerçekleştirmeyecektir. İşte Python'da yaygın olarak kullandığımız sayısal tip dönüşümü yapıları:

» **`int()`** Integer bildiğimiz üzere tam sayıyı ifade etmektedir. `int()` yapısı ise, sayısal bir ifadeyi tam sayı değerine dönüştürür. Yani bizler kullanıcıdan bir tam sayı değeri almak istiyorsak `input()` fonksiyonunun başına `int()` fonksiyonumuzu yazmamız gerekir.

```
filtrelenmis_liste = list(filter(ciftSayiMi, sayilar))
# Sonucu çıktı aldık
print(filtrelenmis_liste)
```

Kodlarımızı incelediğimizde öncelikle filtreleme işleminde uygulanacak fonksiyonumuzu oluşturduk. `ciftSayiMi` fonksiyonu. Fonksiyon içerisinde bir sayının kalansız bölünmesini kontrol ettik ve değeri döndürdük. Görüldüğü üzere `sayilar` isminde bir liste oluşturduk ve değerleri tanımladık. Devamında ise `filtrelenmis_liste` isminde bir değişken oluşturduk ve önce liste dönüşüm işlemi gerçekleştirip, devamında ise `filter` fonksiyonumuzu çağırdık. İlk parametrede `ciftSayiMi` fonksiyonumuzun ismini tanımladık. Diğer parametrede ise filtreleyeceğimiz listemizin adını tanımladık. Son olarak sonucu çıktı aldık. Bu şekilde kodlarımızı çalıştırdığımızda görüntüdeki gibi bir çıktı ile karşılaşacağız.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\tt> & C:/Users/alaca/AppData/Local/Microsoft/WindowsApps/python3.9.exe d:/tt/duzeltme29.py
[2, 4, 6, 8, 10]
PS D:\tt>
```

MODÜLLER

Python'da **modül** kavramı, kodları modüler ve düzenli bir şekilde yönetebilmek için kullanılan bir yapıdır. Modüller; Python programlamada fonksiyonların, sınıfların ve değişkenlerin toplandığı dosyalardır. Bu dosyalar, kodları belirli işlevlere veya yapısal düzenlemelere göre gruplandırmayı sağlar. Bu sayede programları daha organize bir şekilde çalıştırabiliriz. Aynı zamanda kod tekrarını önler ve kodları daha kolay yönetilebilir hale getirir. Python modülleri, ayrıca üçüncü taraf geliştiriciler tarafından da oluşturulabilir ve Python paket yöneticileri aracılığıyla diğer kullanıcılar tarafından erişilebilir hale getirilebilir.

Bir modül üzerinden işlem yapabilmek için önce o modülü import etmemiz yani, modülü program içine dahil etmemiz gereklidir. Bu işlemi **import** anahtar kelimesini kullanarak gerçekleştirebiliriz. Kod satırı üzerinden inceleyelim.

```
import modul_ismi # modül ismini dahil ettik
```

Kodumuzda da görüldüğü üzere yapmamız gereken, öncelikle `import` yazıp sonrasında `modül` ismini yazmamızdır. Böylelikle kullanmak istediğimiz modülü ve modül içerisinde bulunan fonksiyonları kolaylıkla kullanabiliriz. Şimdi Python'da kullanılan bazı önemli modülleri öğrenelim ve modüller konusunda daha detaylı bilgi sahibi olalım.

MATH MODÜLÜ

Python'da math modülünü matematiksel işlemleri yapmak için kullanırız. Bu modül temel matematiksel işlemleri gerçekleştirmek için gerekli fonksiyonları içerir. Örneğin; üstel işlem (üs alma), logaritma, trigonometrik fonksiyonlar (sinüs, kosinüs, tanjant), karekök alma, faktöriyel hesaplama ve daha birçok matematiksel işlemi yapmak için "math" modülünü kullanabiliriz. Bahsettiğimiz her bir işlem için özel fonksiyonları bünyesinde barındırır.Bizler math modülünü dahil ettikten sonra bu fonksiyonları rahatlıkla kullanabiliriz. math modülü sayesinde, matematiksel hesaplamaları daha hızlı ve doğru bir şekilde gerçekleştirebilmekteyiz.

Şimdi sizlerle math modülünü nasıl dahil edeceğimizi öğrenelim.

```
import math  # math modülünü dahil ettik
```

Gördüğünüz gibi önce `import` yazıp sonrasında modülümüzün ismini yani, `math` kelimesini yazdık. Böylelikle `math` modülü Python'da kodlarımıza dahil etmiş olduk.

Peki, bizler `math` modülünü dahil ettikten sonra matematik işlemleri yaptırabileceğimiz fonksiyonlarımızı nasıl çalıştırabiliriz, bunun hakkında bilgi sahibi olalım. Öncelikle, modül ismimizi dahil ettikten sonra bir alt satıra geçerek modül içerisinden kullanmak istediğimiz işleme ait fonksiyonu bulmamız gerekir. Örneğin bizler karekök alma işlemi yapmak istiyorsak `sqrt()` fonksiyonunu kullanmalıyız. Bu işlem için kod düzenimiz şu şekilde olmalıdır.

```
import math  # math modülünü dahil ettik
math.sqrt(parametre) # karekök alma işlemi
```

Kodumuzu incelediğimizde ilk başta modülümüzü dahil ettiğimizi görüyoruz. Daha sonra bir alt satıra geçerek modül ismimizi yazarız ve **nokta** (.) sembolümüzü koyarız. Devamında herhangi bir boşluk kullanmadan fonksiyonumuzun ismini yazarız ve parantezimizi açarız. Parantez içerisine işlem yapmak istediğimiz parametreyi girmemiz gereklidir. Örneğin: 16 değerinin karekökünü hesaplamak istersek kodumuzu şu şekilde yazmamız gerekmektedir.

```
import math  # math modülünü dahil ettik
print(math.sqrt(16)) # 16 değerinin karekökünü görüntüle
```

2

Görüntü İşleme

Bu Bölümde

Python yazılım dilini keşfettiğimize göre artık görüntü işleme serüvenine giriş yapabiliriz.

Bu bölümde, sizlerle birlikte bir görüntüyü okuyup görüntü üzerinde çeşitli işlemler yapmayı öğreneceğiz. Video veya webcam açmayı öğrenip morfolojik dönüşümler, konturlama, fare ve çeşitli tespit işlemlerini nasıl gerçekleştirebileceğimiz hakkında da bilgi sahibi olacağız.

Python yazılım dilini öğrendiğimize göre, görüntü işleme konusuna giriş yapabiliriz. Sizlerle birlikte öncelikle bilmemiz gereken kavramları açıklamalar ve örneklerle öğrenip, devamında ise kodlama ve projelerle kendimizi bu alanda geliştireceğiz.

Haydi görüntü işleme serüvenine başlayalım.

GÖRÜNTÜ İŞLEME NEDİR?

Görüntü işleme; dijital görüntülerin analizi, manipülasyonu ve anlama sürecidir. Bu süreçte, bir görüntüyü dijital formatta alırız (.png, .jpg, .bmp vb.) ve onu bilgisayar tarafından anlaşılabilir ve işlenebilir hale getiririz. Görüntü işleme genellikle bir veya daha fazla adımda gerçekleştirilir: Ön İşleme, Geliştirme, Segmentasyon ve Tanıma gibi.

ÖN İŞLEME

Görüntü işlemenin ilk adımı olan ön işleme, görüntünün kalitesini iyileştirme, gürültüyü azaltma, kontrastı ayarlama ve düzeltmeler gibi işlemleri içerir. Bu adımlar, görüntünün daha sonraki işleme aşamaları için daha uygun hale gelmesini sağlar.

GELİŞTİRME

Geliştirme adımında, görüntüdeki özellikleri vurgulamak veya gizlemek için filtreler, dönüşümler veya diğer teknikler kullanılır. Örneğin; kenar tespiti veya parlaklık düzenlemesi gibi işlemler, görüntünün belli özelliklerini belirginleştirmek için kullanılabilir.

SEGMENTASYON

Segmentasyon, görüntüdeki farklı nesne veya bölgeleri ayırt etme işlemidir. Bu adımda, pikseller belirli özelliklere göre gruplanır veya etiketlenir. Örneğin, bir görüntüdeki nesneleri belirlemek veya arka planı çıkarmak için renk tabanlı veya kenar tabanlı segmentasyon yöntemleri kullanılabilir.

TANIMA

Görüntüdeki nesneleri veya desenleri tanımlama adımıdır. Bu adımda, önceden eğitilmiş bir makine öğrenme modeli veya desen tanıma algoritması kullanılarak görüntüdeki nesneleri sınıflandırabilir veya tanıyabiliriz. Örneğin; bir araba, insan veya köpek gibi nesneleri tanımlamak için derin öğrenme modelleri kullanılabilir.

Görüntü işlemenin birçok uygulaması vardır. Tıp alanında röntgen veya MR görüntülerinin analizi için kullanılabilir. Otomotiv endüstrisinde, sürüş yardımı sistemlerinde veya otonom araçlarda kullanılabilir. Güvenlik sistemlerinde, yüz tanıma veya nesne takibi gibi görevlerde kullanılabilir. Ayrıca sanat, eğlence, tarım, robotik ve birçok başka alanda da kullanılır. Görüntü işleme, hızlı bir şekilde gelişen bir alan olup sürekli olarak yeni teknikler, algoritmalar ve araçlarla ilerlemektedir. Özellikle derin öğrenme yöntemlerinin gelişmesi, görüntü işleme alanında önemli bir dönüm noktası olmuştur. İlerleyen bölümlerde hem derin öğrenme hem de makine öğrenmesi hakkında detaylı bilgi sahibi olacağız. Gö-

rüntü işleme hakkında bilgi sahibi olduğumuza göre şüphesiz bakmamız gereken bir de bilgisayarlı görü kavramıdır. Haydi bilgisayarlı görü hakkında bilgi sahibi olalım.

Bilgisayarlı Görü Nedir?

Bilgisayarlı görü, bir bilgisayarın görüntüleri analiz edebilmesi ve anlayabilmesi için kullanılan bir alanı ifade eder. Bu, dijital görüntülerin işlenmesi, anlaşılması, yorumlanması ve hatta kararlar alınması için kullanılan yöntemlerin ve algoritmaların geliştirildiği bir araştırma ve uygulama alanıdır.

Bilgisayarlı görü, temel olarak iki ana bileşenden oluşur: Görüntü İşleme ve Makine Öğrenmesi.

Görüntü işleme, dijital görüntülerin işlenmesi için kullanılan teknikleri ve yöntemleri içerirken; **Makine öğrenmesi,** görüntüleri analiz etmek ve desenleri tanımak için kullanılan algoritmaların ve modellerin eğitilmesiyle ilgilenir.

Bilgisayarlı görü, birçok uygulama alanında kullanılmaktadır. Bunlardan bazıları şunlardır:

- **Nesne Tanıma ve Sınıflandırma:** Bilgisayarlı görü, nesneleri tanıma ve sınıflandırmada kullanılır. Örneğin; bir görüntüdeki arabaları, insanları veya hayvanları tanımak için derin öğrenme modelleri kullanılabilir.
- **Yüz Tanıma:** Bilgisayarlı görü, yüz tanıma teknolojilerinin temelini oluşturur. Bir görüntüdeki yüzleri algılayabilir, tanıyabilir ve hatta duygusal ifadeleri tespit edebilir.
- **Görüntü İşaretleme ve Etiketleme:** Bilgisayarlı görü, görüntülerin otomatik olarak etiketlenmesi veya içeriklerinin anlaşılması için kullanılabilir. Örneğin, sosyal medya platformlarında görüntülerin içeriklerini otomatik olarak tanımlamak için kullanılabilir.
- **Medikal Görüntüleme:** Bilgisayarlı görü, tıbbi görüntüleme tekniklerinde kullanılır. Röntgen, MRI, CT taramaları gibi medikal görüntülerin analizi ve teşhisi için kullanılan yöntemler geliştirilebilir.
- **Otomotiv ve Ulaşım:** Sürücüsüz araçlar ve ileri sürüş yardım sistemleri için bilgisayarlı görü önemlidir. Görüntülerin analiziyle trafik işaretleri, yaya algılama ve şerit takibi gibi işlemler gerçekleştirilebilir.
- **Güvenlik ve Gözetim:** Bilgisayarlı görü, güvenlik kameraları ve gözetim sistemlerinde kullanılır. Şüpheli davranışları tespit etmek, hırsızlık önleme, yüz tanıma tabanlı erişim kontrolü gibi amaçlarla kullanılabilir.

GÖRÜNTÜ İŞLEME VE BILGISAYARLI GÖRÜ ARASINDAKI FARK NEDIR?

Bilgisayarlı görü ve görüntü işleme, bilgisayarların dijital görüntüleri almasını, analiz etmesini ve değiştirmesini içeren iki ilişkili alanı ifade eder. İki kavram birbirine yakın olsa da bazı farklılıkları vardır. İşte bilgisayarlı görü ve görüntü işleme arasındaki ana farklar:

1. Tanım

- **Görüntü İşleme:** Görüntü işleme, dijital görüntüler üzerinde çeşitli işlemler gerçekleştirerek görüntülerin iyileştirilmesi, analizi veya anlamlandırılmasıyla ilgilenen bir disiplindir. Görüntü işleme algoritmaları, görüntüdeki özellikleri algılamak, tanımak, sınıflandırmak veya çıkarmak için kullanılabilir.
- **Bilgisayarlı Görü:** Bilgisayarlı görü, bilgisayarların optik sensörler veya kameralar aracılığıyla görüntüleri algıladığı, işlediği ve anlamlandırdığı bir alandır. Bu, bilgisayarların nesneleri, insanları veya ortamları görüntüden anlamalarına ve bu bilgiyi kullanarak kararlar vermelerine olanak tanır.

2. Odak Noktası

- **Görüntü İşleme:** Görüntü işleme, görüntülerin analizi ve manipülasyonu üzerine odaklanır. Bu; görüntülerdeki piksel değerlerinin değiştirilmesini, filtrelenmesini, sıkıştırılmasını veya gürültüden arındırılmasını içerebilir. Görüntü işleme algoritmaları; örneğin kenar tespiti, nesne tespiti, görüntü sınıflandırması gibi spesifik görevleri gerçekleştirmek için kullanılabilir.
- **Bilgisayarlı Görü:** Bilgisayarlı görü, görüntü işlemeyle birlikte bilgisayarların görüntüleri anlama ve yorumlama becerisini kapsar. Bilgisayarlı görü; nesneleri tanımak, yüzleri algılamak, hareketi izlemek, derinlik bilgisi elde etmek, 3D modeller oluşturmak ve benzeri daha karmaşık görevleri gerçekleştirmek için kullanılır.

GÖRÜNTÜ BOYUTUNA VE KANAL SAYISINA ULAŞMAK

Şimdi ise okuduğumuz bir görüntünün boyutuna nasıl ulaşabileceğimizi ve kanal sayısını nasıl görüntüleyebileceğimizi öğrenelim. Dijital ortamdaki görüntüler 2 boyutludur. Bunu, genişlik ve yükseklik olarak düşünebiliriz. Bir görüntüdeki kanal sayısı ise görüntüye ait piksellerdeki rengi temsil eder. Örneğin, renkli görüntüler **RGB** formatındadır. Bu, her pikselde 3 farklı kanalın olduğunu belirtir. Gri formatlı görüntülerde ise bu sayı 1'dir. OpenCV kütüphanesini kullanarak görüntü boyutlarına ve kanal sayısına erişebilmemiz için yeni bir yapı öğrenmemiz gereklidir. Bu yapı, shape dizisidir. Kullanımı, `goruntu_matris_degiskeni.shape` şeklindedir. Bize, bir dizi içerisinde 3 farklı değer verir. Sırasıyla Yükseklik, Genişlik ve Kanal Sayısı'dır. Şimdi ise kodumuzu inceleyelim ve yapımızı nasıl kullanabiliriz bunu öğrenelim.

Örnek Kod

```
# Modülümüzü ekledik
import cv2
# Görüntümüzü okuduk
resim=cv2.imread("dog.jpg")
# Görüntü boyutu ve kanal sayısına ulaştık
yukseklik,genislik,kanal_sayisi=resim.shape
# Çıktı aldık
print("Genislik:      {}\nYukseklik:      {}\nKanal      sayisi:      {}\n".
format(genislik,yukseklik,kanal_sayisi))
```

Kodlarımızı incelediğimizde öncelikle **OpenCV** modülümüzü dahil ettik. Devamında ise görüntümüzü okuduk ve `resim` ismindeki matris değişkenine atadık. Görüldüğü üzere yan yana 3 farklı değişken oluşturduk. Böylelikle shape yapımızı kullandığımızda sırasıyla; `yükseklik`, `genişlik` ve `kanal` sayısı değerlerimiz belirttiğimiz değişkenlere tanımlanacak. Son olarak elde ettiğimiz verileri çıktı aldık. Kodumuzu bu şekilde çalıştırırsak görüntüdeki gibi bir çıktı ile karşılaşacağız.

```
Drive/Masaüstü/ucuncu_kitap/tepkime.py
PS C:\Users\alaca\OneDrive\Masaüstü\ucuncu_kitap> & C:/Users/alaca/AppData/Local/Microsoft/WindowsApp
Drive/Masaüstü/ucuncu_kitap/gyk.py
Genislik: 600
Yukseklik: 600
Kanal sayisi: 3

PS C:\Users\alaca\OneDrive\Masaüstü\ucuncu_kitap>
```

Görüntü Boyutunu Değiştirmek

Görüntümüzün bilgilerine nasıl ulaşacağımızı öğrendik. Şimdi ise görüntülerin genişlik ve yükseklik değerlerini nasıl değiştirebileceğimize odaklanalım. Bunun için kullanabileceğimiz bir fonksiyon var. Adı, `resize()` fonksiyonudur.

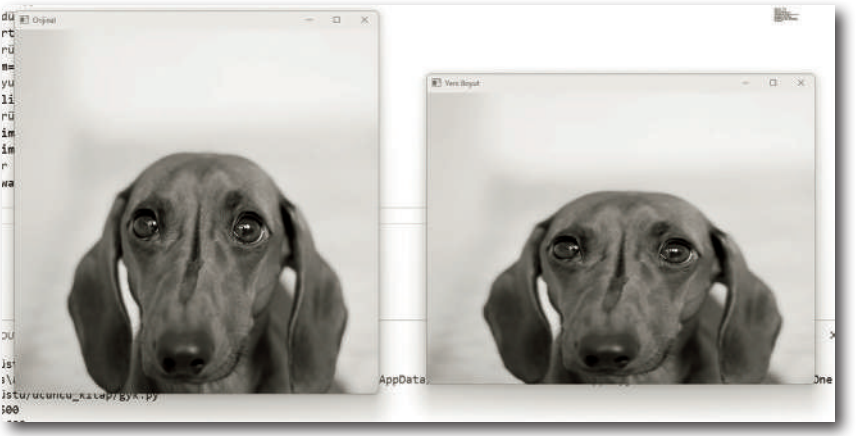
Kullanımı, `cv2.resize(matris_degiskeni,yeni_boyut)` şeklindedir. 2 farklı parametre alır. İlk parametre, boyutu değişecek görüntünün matris değişken adıdır. İkinci parametre ise demet içerisinde genişlik ve yükseklik değerlerinin verildiği boyut değerleridir.

Örneğin: yeni boyutumuzun genişliği 640, yüksekliği ise 480 olsun. İkinci parametreye (640,480) şeklinde tanımlama yapmalıyız. Fonksiyonumuz hakkında bilgi sahibi olduğumuza göre şimdi kodumuzu inceleyelim.

Örnek Kod

```
# Modülümüzü ekledik
import cv2
# Görüntümüzü okuduk
resim=cv2.imread("dog.jpg")
# Boyutumuzu değiştirdik
farkli_boyut=cv2.resize(resim,(640,480))
# Görüntümüzü çıktı aldık
cv2.imshow("Orijinal",resim)
cv2.imshow("Yeni Boyut",farkli_boyut)
# Bir tuşa basana kadar ekranda kalacak
cv2.waitKey(0)
```

Kodlarımızı incelediğimizde öncelikle modülümüzü tanımladık ve görüntümüzü okuduk. Devamında ise `farkli_boyut` isminde yeni bir matris değişkeni oluşturup, `resize()` fonksiyonumuzu değişkene atadık. Fonksiyon içerisinde öncelikle hangi görüntüyü boyutlandıracaksak onun matris değişkenini tanımladık. Bildiğiniz gibi bu resim matrisidir. İkinci parametrede ise demet oluşturup, görüntümüzün yeni boyutlarını tanımladık. Buna göre; genişlik değeri 640 piksel, yükseklik değeri ise 480 piksel olacaktır. Devamında, her 2 görüntümüzü de çıktı alıp, ekranda tuttuk. Bu şekilde kodumuzu çalıştırdığımızda görüntüdeki gibi bir sonuçla karşılaşacağız. Bir sonraki görseli inceleyebilirsiniz.



NOT Görüntü boyutlarını tanımlarken, sadece sabit bir veri şeklinde tanımlama yapmak zorunda değiliz. Örneğin `GENISLIK` gibi bir değişken oluşturup, değerini 1280 olarak atayabilir ve bu değişkeni demet içerisinde tanımlayabiliriz.

Şimdi ise koda bakalım.

Örnek Kod

```
# Genişlik ve Yükseklik değerlerimizi tanımladık
GENISLIK=1280
YUKSEKLIK=720
# Boyutumuzu değiştirdik
farkli_boyut=cv2.resize(resim,(GENISLIK,YUKSEKLIK))
```

Bu şekilde yaptığımızda da herhangi bir sorunla karşılaşmayacağız.

Piksel Yoğunluğuna Ulaşmak

Gelelim piksellerdeki renk yoğunluğuna. Normal şartlarda dijital görüntüler **RGB** renk formatındadır. Ancak, OpenCV kütüphanesi **RGB** yerine **BGR** renk formatını kullanmaktadır. Bunun sebebi OpenCV'nin eskiden kullandığı API'dir.

Bu sebeple renk yoğunluğu dizilimi kırmızı, yeşil ve mavi yerine; mavi, yeşil ve kırmızıdır. Biz de eğer piksel yoğunluklarını öğrenmek istiyorsak, öncelikle mavi, sonra yeşil ve son olarak kırmızı renk yoğunlukları için değişken tanımlamalıyız. Bildiğiniz gibi dijital görüntülerin 2 boyutlu olduğunu belirtmiştim. Bu sebeple, bunlar birer matristir. Matris ise satır ve sütunlardan oluşur. Satırlar y eksenini, sütunlar ise x eksenini temsil eder.

Tespit İşlemleri

Bu başlığımız altında OpenCV modülünde bulunan çizgi ve daire tespiti işlemleri hakkında bilgi sahibi olacağız. Özellikle çizgi ve daire tespiti işlemleri yapmak önemlidir. Çünkü kullanıldığı pek çok alan bulunmaktadır. Bazı kullanım alanları hakkında bilgi sahibi olalım.

» **Nesne Tanıma ve Takip:** Çizgi ve daire tespiti, nesne tanıma ve takip sistemlerinde yaygın bir şekilde kullanılır. Nesnelerin belirli bir düzen veya şekle sahip olduğu durumlarda, bu şekillerin tespiti, nesnelerin konumunu ve hareketini takip etmek için kullanılabilir.

» **Robotik ve Otomasyon:** Çizgi ve daire tespiti, robotik ve otomasyon sistemlerinde önemli bir rol oynar. Örneğin, bir endüstriyel robotun bir çizgiyi veya dairesel bir objeyi takip ederek bir görevi gerçekleştirmesi gerekebilir. Bu durumda, çizgi ve daire tespiti, robotun konumunu ve yönlendirmesini belirlemesine yardımcı olur.

» **Yüz Tanıma:** Çizgi ve daire tespiti, yüz tanıma algoritmalarında da kullanılabilir. Örneğin, gözlerin ve ağızın daire şekline yakın olma eğiliminde olduğu düşünülürse, bu şekillerin tespiti, yüz tanıma sistemlerinde önemli bir adımdır.

» **Trafik İşaretleri ve Yol İşaretlemeleri:** Çizgi tespiti, trafik işaretlerini veya yol işaretlemelerini tespit etmek için kullanılır. Örneğin, otonom sürüş sistemleri, yol işaretlerini tespit ederek aracın seyir halini belirlemekte ve uygun şekilde tepki vermektedir.

Tespit işlemlerinin kullanım alanı hakkında bilgi sahibi olduğumuza göre, OpenCV modülünü kullanarak, bu işlemlerin nasıl gerçekleştirileceğini öğrenelim.

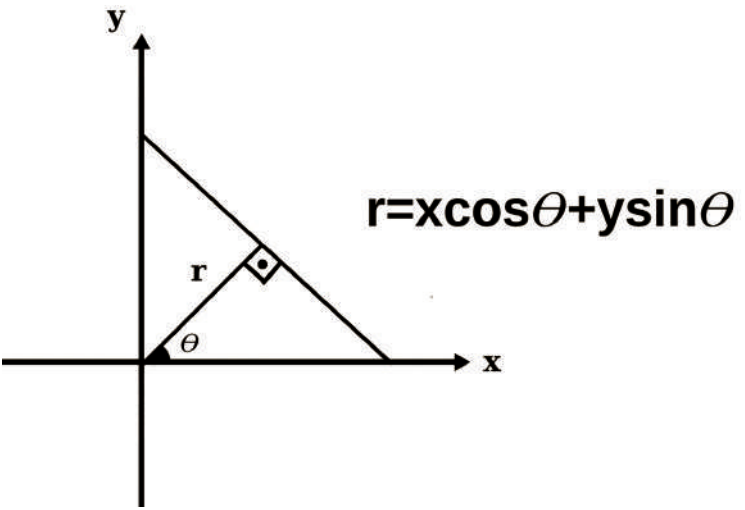
Çizgi Tespiti

OpenCV modülü kullanılarak çizgi çizdirebilmek için yeni bir fonksiyon öğrenmeliyiz. Ancak, bunun öncesinde işin teorisine bakalım. Koordinat sisteminde bir çizgi, **y=mx+c** formülü ile temsil edilir. Bir de bir doğruyu parametrik olarak ifade edebildiğimiz **r= xcosθ + ysinθ** olarak formülümüz mevcuttur.

Bu denklem, bir doğrunun her bir noktasını (**x, y**) koordinat düzleminde belirlemek için iki parametre olan **θ** (**theta**) ve **r'i** (**rho**) kullanır. Formülümüzde bulunan **x** ve **y**, doğru üzerindeki herhangi bir noktayı temsil eder. **θ** (**theta**) ise doğrunun eğitim açısını belirleyen parametredir. **θ** (**theta**) değeri **0** ila **180 derece** olabilir. **0 derece** olması durumunda, doğrunun **yatay yönde** ilerlediğini, **90 derece** olması durumunda ise doğrunun **dikey yönde** ilerlediğini gösterir.

r (**rho**) ise doğrunun orijinden uzaklığını temsil eder. **r** pozitif, negatif veya sıfır olabilir. Formüldeki "**xcosθ**" terimi, doğrunun eğim açısına bağlı olarak **x** eksenindeki bir kaymayı temsil eder. Eğer **θ = 0 derece** ise, yani doğru yatay ise, bu terim yok olur ve doğru yalnızca **x** ekseninde yer alır. Ancak, **θ** değeri değiştikçe, doğru **x** ekseninde kayar.

Benzer şekilde, "**ysinθ**" terimi, doğrunun eğim açısına bağlı olarak **y** eksenindeki bir kaymayı temsil eder. Eğer **θ = 90 derece** ise, yani doğru dikey ise, bu terim yok olur ve doğru yalnızca **y** ekseninde yer alır. Fakat, **θ** değeri değiştikçe, doğru **y** ekseninde kayar. Görsel bir şekilde zihninizde canlanması için grafiği inceleyebilirsiniz.



İşin teorisi hakkında bilgi sahibi olduğumuza göre şimdi gerekli olan fonksiyonumuzu öğrenelim. Fonksiyonun adı `HoughLinesP` fonksiyonudur. 6 farklı parametre alır.

Kullanımı, `HoughLinesP(kenar_matrisi,rho,theta_acisi,esik_deger,min_cizgi_uzunlugu,bosluk)` şeklindedir.

İlk olarak kenar tespiti yaptığımız görüntüyü tanımlamamız gereklidir.

Devamında ise `rho` parametresini tanımlıyoruz. Bu, varsayılan bir değerdir ve hesaplamalarda kullanılacak piksel olarak çözünürlüğü temsil eder. 1 olarak tanımlanmaktadır. Daha yüksek bir `rho` değeri, `Hough` dönüşümünde kullanılan piksel birimini artırır ve bu da tespit edilen çizgilerin daha az sayıda olmasına neden olabilir. Yani, daha genel bir çizgi tespiti gerçekleşir.

Üçüncü parametre olarak radyan açısını tanımlamamız gerekli. Radyan cinsinden olduğu için ve düzgün hesaplama yapabilmek için pi sayısını 180'e bölmemiz gerekiyor. Böylelikle, küçük açılar kullanılarak çizgi tespit işlemi gerçekleştirilebilir. Büyük değer girilmesi durumunda tespit edilecek çizgiler azalır.

Dördüncü parametre ise asgari kesişim sayısını temsil eder. Bu, geçerli çizgilerin doğru tespit edilebilmesi için girilen parametredir. Görsele göre değeri değiştirilebilmektedir.

Beşinci parametre ise çizgi uzunluğu için önemlidir. Örneğin: 10 değerini girersek, 10 piksel uzunluğu ve altındaki çizgiler, çizgi sayılmamaktadır.

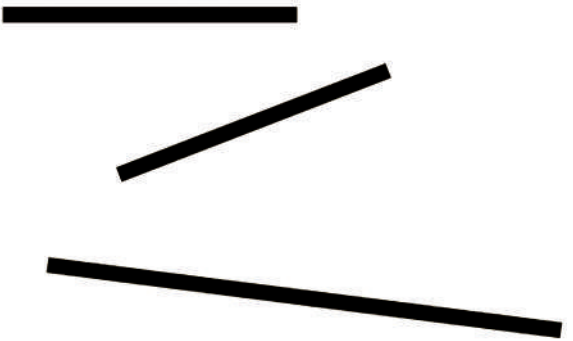
Altıncı ve son parametremiz, çizgi hatlarını birleştirmek için girilmesi gereken boşluk değeridir. Düşük bir değer girilmesi durumunda çizgiler arasındaki boşluk az kabul edilir ve çizgi, düzgün bir şekilde çizdirilmeyebilir.

Fonksiyonumuz hakkında bilgi sahibi olduğumuza göre örnek kodumuzu oluşturalım ve inceleyelim.

Örnek Kod

```
# Modülümüzü tanımladık
import cv2
import numpy as np
# Görüntüyü yükleme
goruntu = cv2.imread("cizgi.png")
# Görüntüyü gri tonlamaya dönüştürme
gri = cv2.cvtColor(goruntu, cv2.COLOR_BGR2GRAY)
# Gürültüyü azaltma
bulanik = cv2.GaussianBlur(gri, (5, 5), 0)
# Kenar tespiti
kenar = cv2.Canny(bulanik, 50, 150)
# Çizgi tespiti
cizgiler = cv2.HoughLinesP(kenar, 1,np.pi/180, threshold=100,
minLineLength=50, maxLineGap=100)
# Tespit edilen çizgileri çizme
if cizgiler is not None:
    for cizgi in cizgiler:
        x1, y1, x2, y2 = cizgi[0]
        cv2.line(goruntu, (x1, y1), (x2, y2), (0, 255, 0), 3)
# Sonuçları gözlemleyelim
cv2.imshow("Cizgi tespiti", goruntu)
cv2.waitKey(0)
```

Kodlarımızı incelediğimizde ilk olarak gerekli olan modüllerimizi tanımladık ve görüntümüzü okuduk. Çizgi tespiti için ben bir görsel oluşturdum. Görüntüdeki gibi. Siz de Drive üzerinden bu görsele ulaşabilirsiniz.



Devamında ise görüntümüzü **gri** renk formatına dönüştürdük ve **Gauss** filtresi uyguladık. Böylelikle, daha gürültüsüz bir görüntü üzerinde çizgi tespiti yapabileceğiz. Görüldüğü üzere **Canny** kenar algoritması işlemini de kullandık. Çünkü çizgi çizdirebilmemiz için görselimizde bulunan çizgilerin kenarlarını tespit etmemiz gerekiyor. Bununla birlikte çizgi tespiti için gerekli olan fonksiyonumuzu çağırdık ve cizgiler ismindeki değişkene atadık. Böylelikle, görüntü üzerindeki çizgiler tespit edildikten sonra bizlere numpy dizisi olacak şekilde çizgilerin koordinatları döndürülecek. Biz de bu koordinatlara göre çizgi çizdirebileceğiz.

Fonksiyonumuzun ilk parametresine kenar matrisimizi atadık. Bu, kenar tespiti işlemi gerçekleştirdiğimiz matrisimizdir. Elde edilen kenara göre çizgiler bulunacak.

İkinci parametre varsayılan `rho` değeri olduğu için `1` olarak ayarladık. Üçüncü parametre ise theta açısıdır. Düşük açılarla işlem yaptığımızda düzgün çizgiler çizdirebildiğimizi öğrenmiştik. Bu sebeple, Pi sayısını 180'e böldük ve üçüncü parametre olarak ayarladık. Dördüncü parametre olarak eşik değerimizi tanımladık. Bu, `threshold` isminde varsayılan bir parametredir. Bu sebeple, doğrudan parametreyi çağırdık ve değerini `100` olarak ayarladık. Çizgi uzunluğunu kontrol eden parametremizi ise `minLineLength` isminde tanımlanmıştır. Biz de bunu `50` olarak ayarladık.

Son olarak iki noktanın aynı çizgide sayılabilmesi için gerekli olan `maxLineGap` parametresini çağırdık ve `100` olarak ayarladık. Unutulmamalı ki bu değerleri ben kendi görselime göre ayarladım. Eğer farklı bir görsel üzerinde çalışacaksanız, parametrelerde değişiklik yapmanız gerekebilir.

3

Nesne Tespiti

Bu Bölümde

Bu bölümde, Python yazılım dilini ve görüntü işleme kavramını öğrendiğimize göre şimdi nesne tespiti hakkında bilgi sahibi olalım.

Bu bölüm ile birlikte, OpenCV kullanılarak yüz ve göz tespit işlemlerinin nasıl yapıldığı hakkında bilgi sahibi olacağız. Ayrıca, algoritmanın çalışma mantığı ve XML kavramı hakkında da bilgi sahibi olacağız.

Bu başlığımız altında OpenCV modülü kullanılarak nesne tespiti işlemini nasıl gerçekleştirebileceğimizi öğreneceğiz. Şu ana kadar Python programlama dilini ve görüntü işleme konusunu öğrendik.

Artık son aşamaya giriş yapıyoruz. Bu da nesne tespiti işlemi. OpenCV'de nesne tespiti işlemi gerçekleştirilebilmesi için **Haar Cascade** algoritmasını bilmemiz ve çalışma mantığını anlamamız gerekiyor.

En basit haliye, bulunmak istenen nesneler eğitilir ve ona benzer görüntülerin bulunduğu resim veya videolardan nesne bulunmaya çalışılır.

Haar Cascade algoritması, nesnelerin belirli özelliklerini tanımlayan öznitelik tabanlı bir yaklaşım kullanır. Genellikle yüz tespiti için kullanılır, ancak diğer nesnelerin tespiti için de uyarlanabilir. Temel olarak, algoritma bir görüntüyü taramak için bir pencere süreci kullanır ve her bir pencere bölgesindeki öznitelikleri hesaplar (görüntüdeki gibi).

Öznitelikler, görüntüdeki yoğunluk değişimlerini temsil eder ve bu değişimlerin varlığına dayanarak nesneleri tanımlamak için kullanılır. Tabii tanımlama işlemi de eğitim süreci sayesinde gerçekleştirilir. Eğitim süreci, pozitif ve negatif örneklerden oluşan bir veri seti gerektirir. Pozitif örnekler, aranan nesnenin görüntüleridir, negatif örnekler ise nesnenin olmadığı alanlardan seçilen görüntülerdir. Eğitim sürecinde, özniteliklerin pozitif ve negatif örnekler arasındaki ayrım gücü hesaplanır ve en iyi ayrım yapabilen öznitelikler seçilir. Bu süreç, zayıf sınıflandırıcıları bir araya getirerek güçlü bir sınıflandırıcı oluşturur. Haar

Cascade algoritması, gerçek zamanlı uygulamalarda kullanılabilen hızlı bir nesne algılama yöntemidir. Ancak, tam olarak doğru sonuçlar elde etmek için iyi bir eğitim veri seti ve doğru parametre ayarları gerektirir. Ayrıca, bazı zorluklara da sahiptir, örneğin, yüz tespiti gibi döndürülmüş nesnelerin veya farklı ölçeklerdeki nesnelerin tespitini zorlaştırabilir. OpenCV modülü kullanarak nesne tespiti yapabilmek ve Haar Cascade algoritmasını kullanabilmemiz için yeni yapılar öğrenmemiz gerekiyor. Haydi bu yapıları inceleyelim.

XML Nedir?

XML (Extensible Markup Language), veri depolamak ve taşımak için kullanılan bir işaretleme dilidir. Nesne tespiti bağlamında, XML dosyaları sıklıkla nesne etiketlerinin, konumlarının ve diğer ilgili bilgilerin saklandığı etiketlenmiş verileri temsil etmek için kullanılır. XML dosyaları, nesne tespiti algoritmalarında kullanılan eğitim veri setlerini oluşturmak için sıklıkla kullanılır. Eğitim veri setleri, tespit yapılacak nesnelerin örnek görüntülerini içerir.

Yüz Tespiti

XML, hakkında bilgi sahibi olduğumuza göre diğer yapılarımızı öğrenelim. OpenCV modülünde, eğer yüz tespiti veya göz tespiti gibi işlemler yapmak istiyorsak, bunlar için hazır XML dosyaları bulunmaktadır. Bu dosyalara ulaşmak için **opencv->sources->data->haarcascades** yolunda bulunan klasörleri izleyebilirsiniz. **haarcascades** klasörünü açtığınızda hazır bulunan XML dosyalarıyla karşılaşacaksınız.

Örneğin, **haarcascade_frontalface_default.xml** dosyası yüz tespiti için kullandığımız XML dosyasıdır. Örneğin, **haarcascade_eye.xml** dosyadası da yüz tespiti için kullandığımız bir XML dosyasıdır. Bu dosyalar, daha önceden eğitilmiş sınıflandırıcı dosyaları olarak da açıklanabilir. Tabii ki bu dosyalar üzerinde işlemler gerçekleştirebilmemiz için bir sınıf öğrenmemiz gerekiyor.

Sınıfın adı, **`CascadeClassifier`** sınıfıdır.

CascadeClassifier, nesne algılama (object detection) alanında kullanılan bir algoritmadır. Daha spesifik olarak, yüz algılama gibi belirli nesneleri görüntü veya videolarda tespit etmek için kullanılır. 1 adet parametre alır.

Kullanımı, `CascadeClassifier("siniflandirici_dosya.xml")` şeklindedir.

Tabii ki sadece tek bir sınıf kullanmamız yeterli değildir. Eğer nesne tespiti işlemi gerçekleştirilmek isteniyorsa, görüntüler aynı zamanda gri formatlı olmalıdır. Çünkü gri formatlı görüntüler üzerinden nesne tespiti işlemi daha basit ve kolay bir şekilde gerçekleştirilir. Görüntü üzerinde nesne tespiti yapılabilmesi için yeni bir metot öğrenmemiz gerekiyor. Bu metot, görüntüdeki nesneleri tespit ettikten sonra, görüntüyü çevreleyebileceğimiz bir **dikdörtgen koordinatı** döndürür. Biz de bu bilgilere göre dikdörtgen oluştururuz ve nesneyi tespit edebiliriz.

Metodun ismi `detectMultiScale` metodudur. 3 farklı parametre alır.

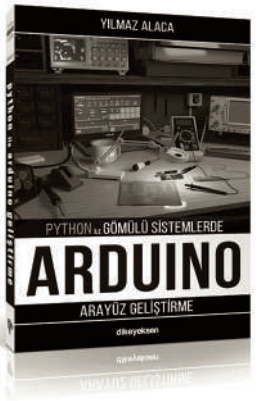
Kullanımı, `siniflandirici_nesnesi.detectMultiScale(gri_goruntu, scaleFactor, minNeighbors)` şeklindedir.

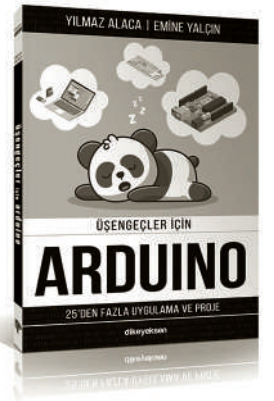
İlk parametrede gri formatlı görüntümüzü tanımlamamız gerekiyor.

İkinci parametre ise ölçek faktörüdür. Bu parametre, nesne tespiti sırasında görüntüde yapılan kaydırma miktarını kontrol eder. `Haar Cascade` algoritması, nesneyi tespit etmek için farklı boyutlarda pencere bölümleri üzerinde kayar. `scaleFactor`, her adımda kaydırmanın oranını belirler. Değer genellikle 1.1 veya 1.2 gibi bir küçültme faktörüdür. Örneğin, 1.1 değeri, her adımda pencerenin %10 oranında küçültülmesi anlamına gelir. Daha küçük bir değer, daha yüksek hassasiyet elde ederken daha yavaş tespit süresi sağlar. Ancak çok küçük değerler, tespitin hatalı olabileceği nesneleri kaçırabilir. Bu sebeple, parametrenin değerini düzgün bir şekilde ayarlamamız önemlidir.

Üçün ve son parametremiz ise `minNeighbors` parametresidir. Bu parametre, tespit edilen nesne adaylarının birleştirilmesi için kullanılır. `Haar Cascade` algoritması, bazı durumlarda aynı nesneyi birden fazla kez algılayabilir. `minNeighbors` değeri, aynı nesneye ait olması gereken aday dikdörtgenlerin birleştirilmesi için belirtilen minimum komşu sayısını belirler. Yani, bu parametre ne kadar yüksekse, tespit edilen dikdörtgenlerin birbirine ne kadar yakın olması gerektiğini kontrol eder. Daha yüksek bir `minNeighbors` değeri, daha güvenilir ancak daha az tespit yapar. Daha düşük bir değer, daha fazla tespit yapar, ancak yanlış pozitif sonuçlara neden olabilir. Bu sebeple `[3-6]` aralığında değer girilmesi tavsiye edilir.

Bu işlemden sonra bir dizi içerisinde 4 farklı değer döndürülür. Bu değerler, tespit edilen nesneyi çevreleyen dikdörtgeni oluşturabilmemiz için gerekli olan değerlerdir. Sırasıyla; `x` değeri (dikdörtgenin sol üst noktası için), `y` değeri (dikdörtgenin sol üst noktası için), dikdörtgenin genişliği ve dikdörtgenin yüksekliğidir.





Yılmaz ALACA | Emine YALÇIN

Tüm Kitap ve Setler