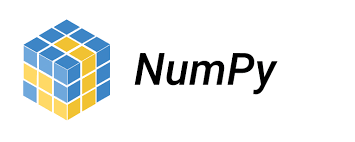
Yazılım tasarımında öncelikle izlediğim yöntem, projenin gerçekleştirilebilmesi için yazılımsal alt yapıyı oluştururken fonksiyonları tek tek ayırıp yazarak basitten komplekse doğru işlemektir.



Şekil

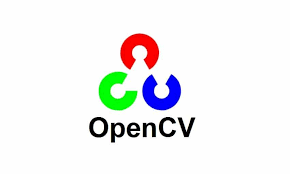
Neden Python programlama dilini seçtik? Python, 2020 IEEE araştırmalarına göre dünya çapında en çok kullanılan programlama dillerindendir. Yüksek seviyeli ve açık kaynak kodludur.

Görüntü işleme, veri bilimi, makine öğrenmesi ve yapay zekâ konusunda da Pyhton çok fazla tercih edilmektedir. Her alanda olduğu gibi bu alanlarda da oldukça zengin kütüphanelere sahiptir. Bu proje için kullandığım kütüphaneler OpenCV, Numpy ve Time şeklindedir.



Şekil

**NumPy** (Numerical Python) bilimsel hesaplamaları hızlı bir şekilde yapmamızı sağlayan bir matematik kütüphanesidir. Python programlama dili için bir kütüphane olup, büyük, çok boyutlu diziler ve matrisler için destek eklerken, bu dizilerde çalışmak için yüksek düzeyli matematiksel fonksiyonların geniş bir koleksiyonudur.



Şekil

**OpenCV** (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. **OpenCV** platform bağımsız bir kütüphanedir, bu sayede **Windows**, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. OpenCV gerçek-zamanlı bilgisayar görüsü uygulamalarında kullanılan açık kaynaklı kütüphane. İlk olarak Intel tarafından geliştirilmiş, daha sonra Willow Garage ve sonra Itseez tarafından sürdürüldü. Bu kütüphane çoklu platform ve BSD lisansı altında açık kaynaklı bir yazılımdır.

Time modülü, zaman ile ilgili işlemler yapabilmemiz için bize çeşitli fonksiyon ve nitelikler sunan bir modüldür.

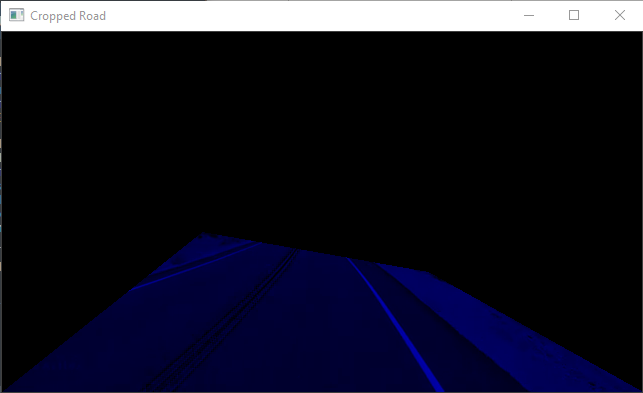
Kodlarımızda kütüphanelerimizi tanımladıktan sonra “calculate\_measures\_4roi” isimli fonksiyon tanımlıyoruz bu fonksiyonumuz ile roi (region of interest) yani ilgilendiğimiz alan anlamına gelen kısmın köşe noktalarını hesaplıyoruz. Bu hesaplamayı yaparken, bize videodan gelen kareyi kullanıyoruz bu kareyi belli bir oranda bölecek şekilde dört noktamızın koordinatlarını belirliyoruz.

Bu koordinatlarını belirlediğimiz dört noktayı diğer bir fonksiyonumuz olan “region\_of\_interest” ile kesme ve maskeleme işlemlerine tabii tutuyoruz. Bu fonksiyonumuzda içine parametre olarak “calculate\_measures\_4roi” den dönen köşe noktalarını ve video karemizi alıyor. Sonuç olarak da maskelenmiş kareyi döndürüoyor.

ROİ belirlememizi sebebi ise çizgi algılama işlemini yaparken bilgisayarımızın video karesinde ilgilenmediğimiz alanlara odaklanmamasını sağlayarak işlem süresini kısaltıyoruz ama asıl sebeplerinden biri de kare içinde kalan değişik bölgelerde gökyüzünde, tünellerde, aracımızın karşısındaki dağlarda yollarda vb. yerlerde anlamsız çizgiler algılamasının önüne geçmektir.



Şekil : Orijinal Video Karesi



Şekil : Maskelenmiş ve Kesilmiş Video Karesi

Şekil 4 ve Şekil 5’te görüldüğü üzere orijinal yolumuz ve belirlediğimiz 4 köşesine göre maskelenmiş yolumuz yukarıda verilmiştir.

Üçüncü fonksiyonumuz olan “drow\_the\_lines” da ise karemizi ve çizgilerimizi fonksiyona vererek işlemlerimizi yapıyoruz. Aldığımız kareyi Numpy kütüphanesi sayesinde kopyalıyoruz. Kopyaladığımız video karesi boyutunda yine Numpy kütüphanesi sayesinde boş bir kare oluşturuyoruz. Genişlik ve yüksekliği video karemizden aldığımız bu boş kareye, üzerinde maskeleme yaptığımız ve kırpma işlemini uyguladığımız karemiz üzerinde tespit ettiğimiz çizgileri **for** döngüsü sayesinde Opencv kütüphanesindeki **line** fonksiyonunu kullanarak çizgilerimizi çizdiriyoruz. Line fonksiyonunun çalışma prensibinde aldığı parametreler sırasıyla çizilmek istenen resim bu bizim projemizde o an videodan gelen karemiz boyutunda oluşturduğumuz boş karemiz, çizginin başlangıç koordinatları, çizginin bitiş koordinatları, çizgi rengimiz ve çizgi kalınlığımız şeklindedir. Fonksiyonumuzun içinde bu işlemden sonra yaptığımız işlem ise ağırlıklarına göre çizgi çizdirdiğimiz boş karemizi ve orijinal video karemizi birleştirmektir. Bu işlemi de yine Opencv kütüphanemizde bulunan **addWeighted** fonksiyonumuz ile yapıyoruz. **addWeighted** fonksiyonumuz da sırasıyla video karemizi, video karemizin ağırlığını (0-1 arasında) çizgilerimizin olduğu kareyi ve bu karenin ağrılığını alarak iki resimde birleştirme işlemini yapıyor. Yazdığımız “drow\_the\_lines” fonksiyonumuzda bu işlemlerin ardından bize çizgileri eklediğimiz karemizi döndürüyor.

Son fonksiyonumuz ise “detect\_lines\_and\_process” fonksiyonumuzdur. Artık bu fonksiyonumuzda tüm işlemlerimizi bitirip karemizi işleyip bu kareyi bize değer olarak döndürmektir. Fonksiyonumuz direkt videodan okuduğumuz kareyi içine alarak önce blurlama işlemi yapmaktadır. Bu blurlama işlemini Opencv kütüphanesinden **GaussianBlur** fonksiyonu ile yapmaktayız. Blurlama yapmamızın amacı resmi yumuşatmak ve resimdeki keskinlikleri azaltıp bilgisayarımızın daha kolay bir algılama işlemi yapmasını sağlamaktır. Yumuşatma işlemimizden sonra yumuşatma yaptığımız kareyinin renigini RGB formatından GRAY formatına çeviriyoruz Opencv aracılığıyla yaptığımız bu işlemde amacımız bundan sonra kullanacağımız ayrıt algılama fonksiyonumuz olan Canny’i kullanabilmek. Canny fonksiyonumuz ile ayrıtlarımızı da algıladıktan sonra karemizdeki kırpma işlemini yapıyoruz yukarıda yazdığımız “calculate\_measures\_4roi” fonksiyonumuz ve Numpy kütüphanemiz sayesinde, çizgilerimizi de bu işlemden sonra algılıyoruz. Opencv içindeki HoughLinesP fonksiyonumuz ile kesilmiş resmimiz üzerinde çizgilerimizi saptıyoruz. Fonksiyonumuzdaki son işlem ise çalışma prensibini açıkladığımız yine kendi yazdığımız “drow\_the\_lines” fonksiyonunu çağırmaktır. İşlemlerimiz tamamlandıktan sonra işlenmiş karemizi döndürerek fonksiyonumuz bitiriyoruz.

metin, sahne, yol, otoban içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil : İşlenmiş Video Karesi

Kodumuzun son kısmı olarak **while** ile son videomuzun okunduğu müddetçe çalışacak bir döngü oluşturuyoruz. Opencv kütüphanemiz yardımıyla videomuzu sırayla okuyor, orijinal resmi ekrana yansıtıyor, “detect\_lines\_and\_process” fonksiyonumuz ile karemizi işliyor time kütüphanemiz ile 0.01 saniye uyutuyor, resmimizi kırparak kırptığımız resmi ekrana yansıtıyor, son olarak da işlenmiş karemizi ekrana yansıtıyoruz. İstediğimiz bir anda işlemi bitirmek için de **if** anahtar kelimemiz ile “q” tuşuna basılıp basılmadığını kontrol ediyoruz.

## Kaynaklar

1 - <https://www.audioremover.com/> - Videodan ses kaydını çıkarmak için kullandığımız site

2 - <https://www.youtube.com/watch?v=ZOZOqbK86t0&t=3975s> – Yol Video verisi için kullandığımız youtube videosu