ELM463 – DÖNEM PROJESİ
HIZ İŞARETİNİN TESPİT EDİLMESİ

Yağmur DERYA
171024011
yagmur.derya2017@gtu.edu.tr

ABSTRACT (ÖZET)

Bu projedeki amaç, görüntüdeki hız işaretinin tespit edilmesidir. Tespit için önce giriş görüntüsü okunup gürültülerin önüne geçmek için filtrelenmiş, ardından eşikleme işleminden geçirilmiştir. Gereksiz detaylarından kurtulmak için eşiklenen görüntü uygun filtre boyu ile tekrar yumuşatılıp ardından görüntüdeki kenarlar tespit edilmiştir.

Kenar tespitinden sonra, hemen sonra daire tespiti yapmak yerine görüntünün tekrar uygun bir filtre ile yumuşatıldıktan sonra daire tespiti yapılmasıyla daha iyi sonuçlar elde edilmiştir.

Daire tespiti ile bulunan hız işareti kırpılıp, görüntüyü iyileştirmek için eşikleme işlemi ve morfolojik operatörler kullanılmıştır. Morfolojik operatörlerin yapısal elemanının boyunun tüm giriş görüntülerine uygun olması için görüntünün boyu 150x150 olacak şekilde sabitlenmiştir. Bu şekilde tespit edilen levhanın siyah-beyaz, temiz bir görüntüsü elde edilmiştir.

ANAHTAR KELİMELER

Hız İşareti, Filtreleme, Eşikleme, Daire Tespiti, Canny Kenar Tespiti, Morfolojik Operatörler

1. Giriş

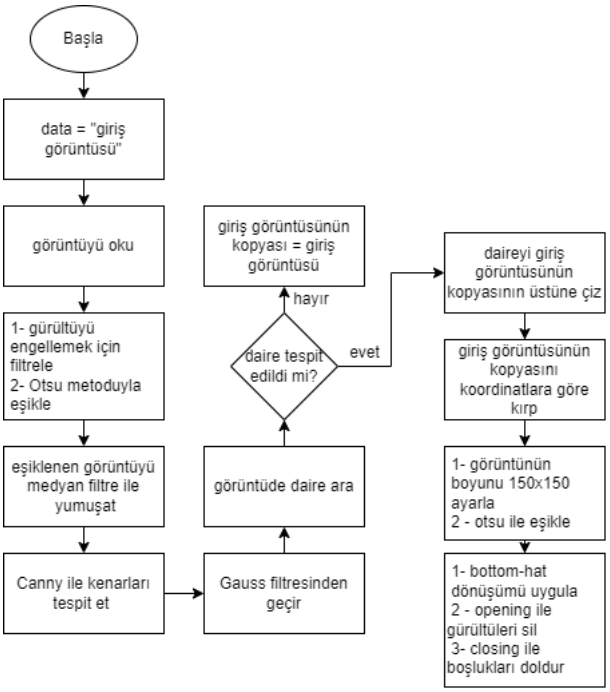
Trafik işaretleri, araç içindeki insanların ve yayaların güvenliği için oldukça önemlidir. Trafik işaretlerine uymamanın yanı sıra, sürücünün trafik işaretini takip etmeye çalışırken dikkatini yolundan ayırması da kazaya sebep olabilmektedir. Bu problemin çözümü için trafik işaretlerinin otomatik tanımlandığı sistemlere ihtiyaç duyulmaya başlanmıştır.

Benzer şekilde günümüzde sıklıkla duyduğumuz otonom araçlar, sürücüye ihtiyaç duymadan, trafiği ve çevrenin durumunu anlayarak gidebilen araçlardır. Otomatik kontrol sistemindeki yazılımlar sayesinde, yapay zeka ile trafik işaretleri vb. birçok nesne algılanarak sürüş sırasında güvenlik sağlanmaktadır.

Bu projede amaç, bahsedilen trafik probleminin çözümünün ilk adımı olarak belli trafik işaretlerinin tespiti için bir algoritma gerçeklemektir. Tespit edilmek üzere daire şeklindeki hız işaretleri seçilmiştir.

Algoritma, gri seviyeli görüntüler için geliştirilmiştir. Eğitim setindeki 7 görüntü kullanılarak algoritma geliştirilmiş, daha sonra test setindeki 9 görüntü ile algoritmanın performansı test edilmiştir.

2. Trafik İşaretinin Algılanması



Şekil 1. Akış diyagramı.

2.1. Verilerin Elde Edilmesi

Dijital ortamdan kolay, orta ve zor olarak sınıflandırılabilecek 16 adet veri elde edildi. Elde edilen veriler eğitim ve test seti olarak iki gruba ayrıldı. Eğitim seti için 7, test seti için 9 veri seçildi.

Algoritma, 8 bitlik gri seviyeli görüntüler kullanılarak geliştirileceği için, elde edilen veriler "IrfanView" ile açılıp uzantısı ".pgm" (portable gray map) olacak şekilde kaydedildi.

2.2. Algoritmanın Gerçeklenmesi

Uygun formata getirilen verilerin tümü "Jupyter Notebook"ta bir hücrede tanımlandı. "Numpy" kütüphanesinin yanında, görüntü işleme fonksiyonlarını kullanabilmek için "OpenCV" [1] kütüphanesi tanımlanıp, kütüphanenin fonksiyonları kullanıldı.

Kodun hızlı çalışması ve kalabalık olmaması için tüm veriler yorum satırına alındı. İncelenmesi istenen veri yorum satırından çıkartılarak kod derlenerek, tek seferde tek bir verinin detaylı sonuçlarının görünmesi sağlandı.

İlk adımda "Seçici Gauss Filtresi" olarak da adlandırılan "Bilateral Filtreleme" ile giriş görüntüsündeki gürültü filtrelendi. Bilateral filtre, Gauss filtresinden farklı olarak, kenarlardaki keskinliği korurken, görüntüdeki gürültüyü azaltmak ve görüntüyü yumuşatmak için kullanılmaktadır. [2] Filtreleme işlemi yapılırken, her piksel komşularının ağırlıklı ortalamasını almaktadır.

Gürültü filtrelendikten sonra, görüntü Otsu metodu ile eşiklendi. Herhangi bir eşikleme yöntemi yerine Otsu metodunun tercih edilmesinin sebebi, işaretlerin üstüne düşen gölgelerin etkisini azaltmak ve daha sonraki aşamada tespit edilecek dairenin formunu en iyi şekilde koruyabilmektir.

Otsu metodu ile eşikleme yapılırken, sınıf içi varyans hesaplanıp, bu değeri en küçük olmasını sağlayan değer eşik değeri olarak seçilir.

Şekil 2.a. Giriş görüntüsü.

Şekil 2.b. Otsu metodu ile eşiklenmiş görüntü.

Şekil 2.a'da işaretin üstündeki gölge görülmektedir. Otsu metodu ile eşiklenendikten sonra (Şekil 2.b) gölge probleminin çözüldüğü görülmektedir.

Kolay zorluktaki verilerde arka plan düz veya düze yakındır fakat zor ve orta zorluktaki veriler için aynı durum geçerli değildir. Bu problemin çözümü için medyan filtesi kullanılarak, hız işaretinde büyük kayıplar olmadan arka plandaki detayların bulanıklaştırılabileceği düşünüldü.

Medyan filtresi, komşu piksel değerlerini sıralayıp ortadaki değeri merkeze koyar. Keskin kenarlar ortalama değer filtresine göre daha iyi korunur.

Eğitim setindeki verilerle yapılan incelemeler sonucunda, medyan filtrenin boyu 11x11 seçildiğinde, işaretlerde istenmeyen kayıpların oluşmadığı ve arka plandaki detayların da fark edilir seviyede azaldığı görüldü (bkz. Şekil 3).

Şekil 3. Medyan filtre ile Şekil 2.b'nin filtrelenmesi.

Daha sonra, bulanıklaştırılan görüntüdeki kenarlar Canny kenar tespiti ile tespit edildi [3]. Canny ile kenar tespiti, John F. Canny tarafından geliştirilmiş bir kenar tespit algoritmasıdır. Sobel gibi kenar tespit yöntemlerinden farklı olarak, iki eşik değerinin olması sayesinde, istenmeyen kenarların önüne geçilir ve kırıklar varsa doldurulur.

Kenarları tespit edilen görüntü (bkz. Jupyter Notebook fig.4) direkt daire tespit işlemine sokulduğunda, küçük detayların tespitte sorun çıkarabileceği fark edildi. bu sebeple daire tespiti yapılmadan önce Gauss filtresi ile tekrar bulanıklaştırılarak detayları azaltıldı (bkz. Jupyter Notebook fig.5) ve bir sonraki aşamada daire tespiti yapılırken daha iyi sonuçlar elde edildi.

Yüksek frekanslar kenarları ifade etmektedir. Gauss filtresi, görüntünün yüksek frekanslarının bastırılmasıyla yumuşatılma işlemidir. Dolayısıyla alçak geçiren filtredir.

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \textbf{eşitlik-1}$$

Standart sapma büyüdükçe, yumuşatma oranı artar fakat frekans uzayında çalışılıyorsa tam tersi durum geçerlidir.

Kenarları tespit edilen görüntü bulanıklaştırıldıktan sonra, OpenCV kütüphanesindeki HoughCircles() fonksiyonu [4] ile görüntüdeki daire tespit edildi. Denemeler sonucunda, seçilen verilere uygun yarıçap aralığının [30, 300] olduğuna karar verildi.

HoughCircles() ile bulunan dairelerin 3 parametresi vardır. İlk iki parametre dairenin merkez koordinatları, 3. parametre ise yarıçap bilgisidir.

Şekil 4. Tespit edilen daire.

Fonksiyon ile elde edilen bu 3 parametre kullanılarak orijinal görüntü kırpılarak orijinal görüntüden sadece hız işaretinin görüntüsü elde edilmiş oldu (Şekil 5.b).

Şekil 5.a. Görüntüde tespit edilen hız işareti.

Şekil 5.b. İşaret kırpıldıktan sonra elde edilen görüntü.

Kırpılan görüntü önce yeniden boyutlandırıldı. Bu şekilde morfolojik operatörler için yapılandırma elemanının büyüklüğünün her görüntüye uygun olması amaçlandı. Görüntü tekrar Otsu yöntemiyle eşiklenerek, üzerinde gölge olabilecek görüntülerdeki gölgenin etkisi ortadan kaldırıldı. Eğitim setindeki bazı görüntülerin eşiklendikten sonra ufak tefek gürültü veya boşluklarının olduğu görüldü. Görüntüleri iyileştirmek için dilation ve erosion işlemleri uygulandı. Dilation işlemi ile beyaz gürültüler temizlenirken, erosion işlemi ile de küçük siyah boşluklar kapatıldı [5].

Şekil 6. Morfolojik operatörlerden sonra elde edilen görüntü.

Şekil 6 incelendiğinde, Şekil 5.b'deki bulanıklığın göz yorması sebebiyle, daha iyi bir görüntü elde edildiği söylenebilir fakat görüntünün daha iyi olup olmadığı nerede kullanılacağına göre değiştiği için kesin bir yargıda bu aşamada bulunulamaz.

En son aşamada EasyOCR [7] kütüphanesi kurularak, en son elde edilen görüntü (şekil 6) ile şekil 5.b'de elde edilen görüntülerden tabeladaki yazı okunarak hız limitinin tespitinde hangi sonucun daha iyi olduğunu tespit etmek amaçlandı.

```
Hız limiti (fig.7): 50
Hız limiti (fig.9 b): 50
```

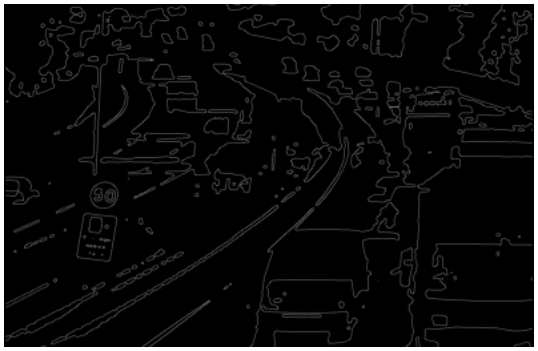
Şekil 7. Okunan işaretlerden elde edilen sonuç.

"egitim_50_2.pgm" verisi için her ikisinden de doğru sonuç elde edildi. Daha sonra hem eğitim hem de test setindeki veriler kontrol edildiğinde çoğunda her ikisi de doğru sonuç verse de Şekil 5.b'deki görüntünün "test_50.pgm" görüntüsünde hatalı sonuç verdiği görüldü. Dolayısıyla Şekil 6'nın bu durumda daha iyi bir görüntü olduğu doğrulanmıştır.

```
Hız limiti (fig.7): 60
Hız limiti (fig.9 b): 50
```

Şekil 8. "test_50.pgm" görüntüsü için okunan değerler.

Şekil 7.a. Giriş görüntüsü (test_50.pgm)



Şekil 7.b. Canny ile kenarların tespiti.

Şekil 7.c. Tespit edilen işaretin kırpılmış görüntüsü.

Şekil 7.d. Eşikleme ve morfolojik operatörler sonrası elde edilen görüntü.

Şekil 7 (a-b-c-d)'de test grubundaki görüntüden elde edilen sonuç görülmektedir. Hız işareti başarılı bir şekilde görüntüde tespit edilmiştir. Eğitim ve test gruplarında toplam 16 adet görüntü olduğu için her birinin sonucu rapora eklenememiştir.

3. Sonuç ve Yorum

Eğitim setindeki veriler kullanılarak gerçeklenen algoritma daha sonra test setindeki görüntülerle test edildi ve bütün görüntülerdeki hız işareti doğru şekilde tespit edildi.

Tespit yapılırken zorlayıcı nokta, minimum ve maksimum yarıçap değerlerini doğru seçebilmekti. Farklı uzaklıktaki işaretlerin yarıçapları birbirlerinden oldukça farklı çıkmaktadır. Projedeki amaç farklı uzaklıktaki görüntülerin boyutlarını ayarlamak olmadığı için, veriler algoritmaya sokulmadan önce trafik işaretlerinin boyutları belli bir aralıkta olacak şekilde ayarlanarak bu problem çözüldü.

Daire tespitinde kullanılan fonksiyonun merkezi bulmakta iyi çalışırken, yarıçap hesabında ufak tefek hatalar barındırdığı görüldü. Fakat bu problem için bu hatalar göz ardı edilebilecek boyutta olup, trafik işareti tespiti yapılmasına engel olmamıştır.

Görüntü birçok kez farklı filtrelerden geçirilerek yumuşatıldı. Tek seferde bu yumuşatılma işlemi yapılamadı çünkü bulunması istenen işaretin detayları kaybedilebilir. Bu sebeple aşamalı olarak amaca göre farklı filtreler kullanılarak yumuşatılma işlemi yapıldı.

Bir görüntüde birden fazla aynı şekildeki işaretin tespiti yapılsaydı, bileşen etiketleme yöntemi ile tek tek tespit edilen o şekildeki objeler alınıp incelenebilirdi.

İşaret üzerindeki sayıların okunması yapılırken, kütüphanenin kurulumu ve okuma fonksiyonu kodu yavaşlatmaktadır. Problemdeki amaç sayının tespiti değil işaretin tespiti olduğu için o kısımlar yorum satırı olarak bırakılmıştır.

Kaynaklar

[1]https://docs.opencv.org/4.x/d7/da8/tutorial_table_of_content_imgproc.html
[2]https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
[3]https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
[4]https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html
[5]https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html
[6]https://developpaper.com/traffic-sign-recognition-based-on-opencv/
[7]https://www.analyticsvidhya.com/blog/2021/06/text-detection-from-images-using-easyocr-hands-on-guide/