ELM 463 – DÖNEM PROJESİ RAPORU
GÖRÜNTÜ ÜZERİNDEKİ TRAFİK İŞARETLERİNİN VE LEVHALARININ TESPİT EDİLMESİ

Muhammet Fatih KESKİN
fatih.keskin2017@gtu.edu.tr

ABSTRACT (ÖZET)

Bu projede **"Python"** programlama dili ile görüntü işleme dersi kapsamında görüntü üzerindeki trafik işaretlerinin/levhalarının tespit edilmesi konusu üzerinde çalışılacaktır. Seçilen 8 adet görüntüye eşikleme[1], ayrıt saptama[2], morfolojik operatörler[3], çizgi/elips/daire tespiti[4],filtreleme[5] gibi yöntemler uygulanarak incelenecektir. Son olarak öznitelik çıkartımı[6] yöntemi sayesinde trafik işaretlerinin tespiti yapılacaktır.

1. Giriş

Projede, birbirinden farklı 8 ayrı görüntü üzerinde tek bir algoritma uygulanarak trafik işaretlerinin tespit edilmesi amaçlanmıştır. Görüntüler, birbirlerinden farklı oldukları için farklı koordinatlarda, farklı piksel değerlerinde ve farklı trafik işaretlerine sahip olacaklardır. İlgili parametrelerden ve yöntemlerden faydalanılarak, görüntülerin tek bir algoritma sayesinde analizlerinin yapılması amaçlanmıştır. Bu analizler sayesinde, çeşitli görüntü işleme yöntemleri uygulanacaktır. Analizler sonucu uygun sıralama ile görüntü işleme yöntemleri kullanılarak, hedefe uygun şekilde trafik işaretlerinin tespiti sağlanmış olacaktır.

2. Kullanılan Yöntemler

Bilgisayar ortamında görüntü işleme için, **Python Programlama Dili**'nin bir aracı olan **Jupyter Notebook** kullanılacaktır.

Trafik işaretlerinin çeşitlerinden kaynaklı olarak birçok farklı şekil içeren trafik işareti bulunmaktadır. Sadece Türkiye'de bulunan trafik işaretlerinin örnek alınması durumunda **ters üçgen, üçgen, eşkenar dörtgen, kare, dikdörtgen, sekizgen ve daire** şeklinde bulunabilirler.

Bu durumda tek algoritma sayesinde bütün trafik işaret levhalarının algılanması için bütün olabilecek şekiller kontrol edilmelidir ve bulunan her benzer şekli üzerinde analizler yapılmalıdır.

2.1. Giriş Görüntülerinin Okunması

Analizlerin yapılabilmesi ve yöntemlerin uygulanabilmesi için, giriş görüntüleri **.jpg** formatında **Jupyter Notebook** ortamına eklenmiştir.

Eklenen görüntüler renkli olduğu için gri seviyeye indirgenmiştir.

Son olarak gri seviyeye indirgenmiş görüntünün genişliği 600 piksel olarak ayarlanmıştır. Bunun sebebi daire şekline sahip trafik işareti bulurken kullanılan çap boyutunu bir standarda eşitlemek içindir. Bu işlev sayesinde, seçilen görüntünün genişliği ve uzunluğu giriş görüntüsüyle orantılı olarak ayarlanır. Örneğin; giriş görüntü genişliği (1200,600) ise **resize** fonksiyonu için **(width=600)** seçildiği durumda yeni oluşturulan görüntü boyutları (600,300) olacaktır.

2.2. Top-Hat Dönüşümü

Gri seviyeye indirgenen görüntüler üzerinde ön ve arka planın ayırt edilebilmesi için **Top-Hat Dönüşümü** uygulanmıştır. **Top-Hat Dönüşümü** ile trafik işaretinin koyu renkteki arka planlarının ve içerisindeki açık bölgeleri veya tam tersi durum özelliği kullanılarak, trafik işareti olamayacak bölgeleri bastıran, trafik işareti olabilecek bölgeleri ön plana çıkartılmıştır. Kısaca içerisinde sadece trafik işaretleri veya levhaları bulunan bir görüntü düşünürsek, arka planı koyu renkli iken bulunacak olan şekil daha açık renkli olacaktır (Örn: *Şekil 1*). Veya tam tersi durumlardaki renkler için de geçerli olacaktır.

Şekil 1 – Gri Seviyeli Trafik İşaretleri

Top-Hat Dönüşümü arka plandan farklı aydınlık seviyeli nesneleri araştıran gri seviyeli resimlerin segmentasyonunda kullanılan bir dönüşümdür. Gri seviyeli morfolojik işlemler kullanılarak elde edilir. Tepe veya çukur bölgeleri belirginleştirme özelliğine sahiptir.

A giriş görüntüsü ve B yapı elemanı iken;

- Aydınlık bölgeler için,

$TopHat[A,B] = A - (AoB)$ **(denklem-1)**

- Karanlık bölgeler için,

$TopHat[A,B] = (A \bullet B) - A$ **(denklem-2)**

şeklinde hesaplanmaktadır.

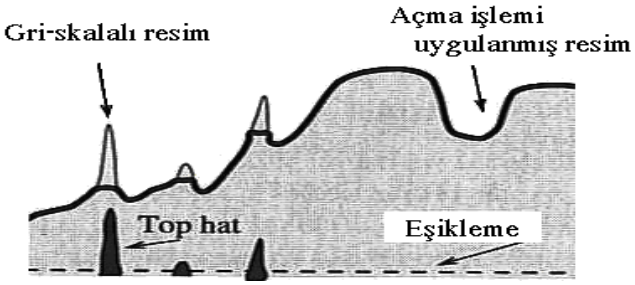
Genişletme İşlemi (dilation): İkili imgedeki nesneyi büyütmeye ya da kalınlaştırmaya yarayan morfolojik işlemdir. Sayısal bir resmi genişletmek resmi yapısal elemanla kesiştiği bölümler kadar büyütmek demektir. Kalınlaştırma işleminin nasıl yapılacağını yapı elemanı belirler. (" $\oplus$ " ile gösterilir)

Aşındırma işlemi (erosion): İkili imgedeki nesneyi küçültmeye ya da inceltmeye yarayan morfolojik işlemdir. Aşındırma işlemi bir bakıma genişletmenin tersi gibidir. Aşındırma işlemi ile sayısal resim aşındırılmış olur. Yani resim içerisindeki nesneler ufalır, delik varsa genişler, bağlı nesneler ayrılma eğilimi gösterir. (" $\ominus$ " ile gösterilir)

Açma işlemi (opening): Genişletme ve aşındırma işlemini ardışıl uygulanmasıyla elde edilir. Bu işlemle birbirine yakın iki nesne görüntüde fazla değişime sebebiyet vermeden ayrılmış olurlar. ("o" ile gösterilir)

Kapama işlemi (closing) : Aşındırma ve genişletme işleminin ardışıl uygulanmasıyla da kapama işlemi elde edilir. Dolayısıyla birbirine yakın iki nesne görüntüde fazla değişiklik yapılmadan birbirine bağlanmış olur. (" $\bullet$ " ile gösterilir)

Top-Hat Dönüşümü Şekil 2'deki gibi, açma işlemi ile orijinal resmin farkı alınarak bulunmaktadır.



Şekil 2 – Top Hat Dönüşümü

2.3. Otsu Eşikleme Metodu

Top-Hat Dönüşümü uygulanmış gri seviyeli görüntü sayesinde ilgilenilen açık renkli objeler belirginleşirken, bir sonraki adımda ise **Otsu Metodu** uygulanmıştır. **Otsu metodu**, gri seviyeli bir görüntü ikili seviyeye indirgenirken kullanılabilecek en uygun eşik değerinin tespit edilmesini sağlar. Normalde gri bir görüntüyü ikili biçime dönüştürmek için bir eşik değeri belirlenir ve bu eşik değeri üstündeki renkler beyaza, altındaki renkler siyaha eşitlenir. Ancak tüm görüntüler aynı niteliğe sahip olmadığı için sabit bir eşik değeri tüm görüntüler için uygun olmayacaktır. Otsu eşik değeri, görüntünün renk dağılımına uygun olarak belirlemektedir. Bu nedenle bir ikili seviyeye indirgenen bir görüntü için en uygun eşik değerini bulacak ve seçilen 8 adet fotoğraf için her birine farklı bir eşik değeri atayacaktır.

Otsu metodu görüntüyü ön ve arka plan olarak iki gruba ayırmaktadır. Bu sayede artık ön ve arka plan olarak ikiye ayrılmış görüntü üzerindeki piksel değerleri ya 0 değerine ya da 255 değerine sahip olacaktır.

Bu durumda oluşan siyah ve beyaz renkler ile görüntü üzerindeki ilgi alanları daha da belirginleştirilmiş olmaktadır.

Otsu Eşikleme Yöntemi:

➔ Giriş görüntüsü okunur ve görüntünün histogramı oluşturulur. **p(i)** normalize histogramdır.

➔ Bulunan histogram grafiğinden veya dağılımından bakılarak grupların ağırlıkları olan $\mathbf{q_1(t)}$ ve $\mathbf{q_2(t)}$ hesaplanır.

➔ Bulunan **p(i)** ve $\mathbf{q_1(t)}$ ve $\mathbf{q_2(t)}$ kullanılarak her grubun ortalama değeri olan $\boldsymbol{\mu_1(t)}$ ve $\boldsymbol{\mu_2(t)}$ hesaplanır.

➔ Bulunan değerler kullanılarak her grubun varyansı hesaplanır. $\sigma^2(t)= \sigma^2_w(t)+ \sigma^2_b(t)$ formülü kullanılarak $\sigma^2_b(t)$ maksimum değeri alması sağlanır.

➔ Bu sayede $\sigma^2_w(t) = q_1(t)* \sigma_1^2(t) + q_2(t)* \sigma_2^2(t)$ değerinin minimum olması sağlanır ve eşikleme değeri $\sigma^2_w(t)$ olarak belirlenir.

Bu işlemler sonucunda gri seviyeli bir görüntü ikili seviyeye indirgenirken kullanılabilecek en uygun eşik değeri tespit edilmiştir. Ayrıca grupların kendi içindeki standart sapmaları küçüktür ve gruplar birbirinden iyi derecede ayrılmış olarak elde edilmiştir.

2.4. Morfolojik Gradient

Otsu Eşikleme Metodu uygulanmış görüntüye bu adımda ise **Morfolojik Gradient** işlemi uygulanmıştır. **Morfolojik Gradient**, bir görüntünün genişlemesi (dilation) ve aşınması (erosion) arasındaki farkı temsil eder (denklem-3).

A giriş görüntüsü ve B yapı elemanı iken;

$$Gradient(f) = (A \oplus B) - (A \ominus B)$$ **(denklem-3)**

Morfolojik Gradient uygulanan görüntüdeki piksel değerleri yakınındaki piksel değerlerinin kontrast yoğunluğunu gösterir. Bu işlem sonucunda görüntüdeki kenarlar daha kalın elde edilmiştir. **Morfolojik Gradient** sayesinde her piksel değerinin (tipik olarak negatif olmayan) o pikselin yakın çevresindeki kontrast yoğunluğunu gösterdiği bir görüntüdür. Kenar algılama ve segmentasyon uygulamaları için kullanışlıdır.

➔ Morfolojik yöntemleri uygulamak için yapı elemanı kullanılmalıdır. Bu yapı elemanı 3x3 boyutlarında ve kare şeklinde seçilmiştir.
➔ Sonraki adımda genişletme işlemi (dilation) sayesinde denklemin ilk görüntüsü elde tutulur. **dilate** fonksiyonu sayesinde oluşturulan 3x3 boyutlu yapı elemanı olan **kernel** ile eşiklenmiş görüntü genişletilir.
➔ Aşındırma işlemi (erosion) sayesinde denklemin ikinci görüntüsü elde tutulur. **erode** fonksiyonu sayesinde oluşturulan 3x3 boyutlu yapı elemanı olan **kernel** ile eşiklenmiş görüntü aşındırılır.
➔ Denklemden de yararlanılarak sonraki adımda ise her iki görüntü birbirinden çıkartılır ve "gradient" sonucu elde edilir.
➔ **Morfolojik gradient** işleminin son adımında ise elde edilen "gradient" görüntüsü giriş görüntüsünden çıkarılır ve **new_grad** görüntüsü elde edilir.

2.5. Canny Kenar Dedektörü

Morfolojik Gradient uygulanmış görüntüye bu adımda ise **Canny Metodu** uygulanmıştır. **Canny Metodu**, görüntü üzerindeki kenar tespiti ile o görüntüdeki nesneler tespit edilebilir, sayısı çıkartılabilir ve özellikleri belirlenebilir. Ayrıca **Canny Metodu** görüntüdeki kenarları algılamak için kullanılan en popüler yöntemdir. Kenarları algılamak için birden fazla işlem uygulanır ve **Canny** çıkış görüntüsü elde edilir. Bu işlemler:

- **Gürültü Azaltma:** Kenar algılama sonuçları görüntü gürültüsüne karşı oldukça hassastır. Görüntüdeki gürültüden kurtulmanın yollarından birisi onu yumuşatmaktır. Gürültüyü azaltmak için orijinal görüntüye 5x5 **Gaussian** filtresi uygulandı. **Gaussian** filtresinin matematiksel formülü 4. denklemde belirtilmiştir.

$$H(i,j) = \frac{1}{2\sigma\pi} \exp\left[-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right]$$ **(denklem-4)**

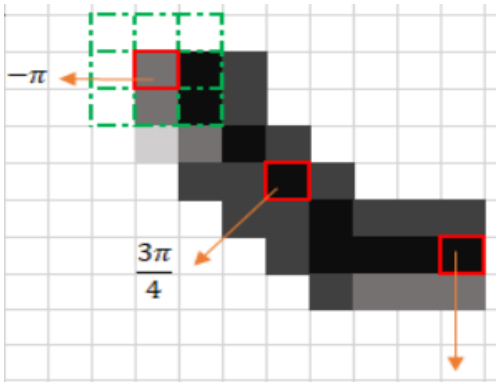
- **Gradyan Hesaplaması:** Kenar algılama operatörlerini kullanarak görüntünün gradyanını hesaplayarak kenar yoğunluğunu ve yönünü tespit eder. Kenarlar, piksel yoğunluğunun değişmesine karşılık gelir. Bunu hesaplamanın en etkili yolu görüntüye "Sobel" filtresi uygulamaktır. Yoğunluk değişimini her iki yönde tespit etmek için x yönünde ve y yönünde farklı filtreler uygulanır. Bu filtreler Şekil 3'teki gibidir. Daha sonra elde edilen Gx ve Gy görüntülerinin mutlak değerleri toplanarak Gradyan hesaplanır.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Şekil 3 – x ve y yönündeki Sobel filtreleri

- **Maksimum Olmayanı Bastırma:** Yukarıdaki adımlardan sonra görüntünün ince kenarlı olmalıdır. Kenarları inceltmek için maksimum olmayanı bastırma işlemi uygulanmalıdır.
Algoritma, gradyan yoğunluk matrisi üzerindeki tüm noktalardan geçer ve kenar yönlerindeki maksimum değere sahip pikselleri bulur. Detay olarak gradyan yoğunluk görüntüsü üzerindeki tüm pikselleri tarayan matris başlatılır. Gradyan'dan yararlanılarak açı değerine göre kenar yönü belirlenir. Açı yönünde bulunan piksellerin, seçilen pikselden daha yüksek yoğunluğa sahip olup olmadığı kıyaslanır ve maksimum olmayan pikseller bastırılır. Şekil 4'teki görüntüdeki gibi kıyaslama yapılır.



Şekil 4 – Maksimum olmayanı bastırma algoritması açıklama görseli

- **Çift Eşik:** Çift eşik 3 tür piksel bulmayı amaçlar. Güçlü, zayıf ve alakasız olarak sınıflandırılabilir. Güçlü pikseller için yüksek eşik değeri kullanılır ve kenara katkıda bulunduğu kabul edilir. Alakasız pikseller için düşük eşik değeri kullanılır ve kenar için alakasız olarak kabul edilir.

- **Histerezis Mekanizması:** Eşikleme sonucunda oluşan görüntüdeki piksel değerleri sıra ile taranır. Her piksel değeri için 8 komşuluk kontrol edilir ve komşularından birisi güçlü piksel ise zayıf pikseller kenar olarak kabul edilir.

Tüm bu işlemler sonucunda görüntüdeki kenarlar tespit edilmiş olur. Bu yöntemin en kullanışlı kenar tespit algoritması olmasının sebebi birden fazla adım uygulanarak sonucun elde edilmeye çalışılmasıdır. Fakat tüm buna kolaylık olarak tek bir fonksiyon sayesinde Canny algoritması görüntüye uygulanabilir.

2.6. Trafik İşaret Levhasının Tespiti

Trafik işaret levhalarının tespiti için konturlar kullanılmıştır. Konturlar, aynı renk veya yoğunluğa sahip tüm sürekli noktaları (sınır boyunca) birleştiren bir eğri olarak basitçe açıklanabilir. Konturlar, şekil analizi ve nesne algılama ve tanıma için kullanışlı bir araçtır.

Kontur Bulma:

- ➔ Kontur bulunurken ilk olarak bütün konturlar bulunur. Algoritma, kontur boyunca yatay, dikey ve köşegen kesimleri sıkıştırır ve yalnızca uç noktalarını bırakır.
- ➔ İkinci adımda ise alan sıralaması ile en iyi 6 adet kontur bulunur. (Buradaki 6 trafik işareti olabilecek şekil sayısını temsilen konulmuştur).
- ➔ Bütün konturler için bir döngü dönerken kontur çevresi hesaplanır ve konturun kapalı bir çevre oluşturduğu belirtilir.
- ➔ Bulunan konturler eğri ise farklı, kapalı bir şekil ise farklı bir yaklaşım değeri verilir.
- ➔ Bu konturun boyutu sayesinde görüntüde bulunan kenar sayıları birbirine eşittir. (bkz: *Tablo -1*)

YAKLAŞIM BOYUTU	ŞEKİL ADI
3	Üçgen, Ters Üçgen
4	Kare, Eşkenar Dörtgen
6	Altıgen
8	Sekizgen

Tablo 1 – Yaklaşım Boyutu ile Şekil Adı İlişkisi

Eğer yakalamaya çalışılan trafik işareti bir yuvarlak ise **Hough dönüşümünden** yararlanılabilinir. **Çember Hough Dönüşümü (CHT)**, kusurlu görüntülerdeki daireleri tespit etmek için dijital görüntü işlemede kullanılan temel bir **öznitelik çıkarma** tekniğidir. Çember adayları, Hough parametre alanında oylama yapılarak ve ardından bir akümülatör matrisinde yerel maksimumlar seçilerek üretilir.

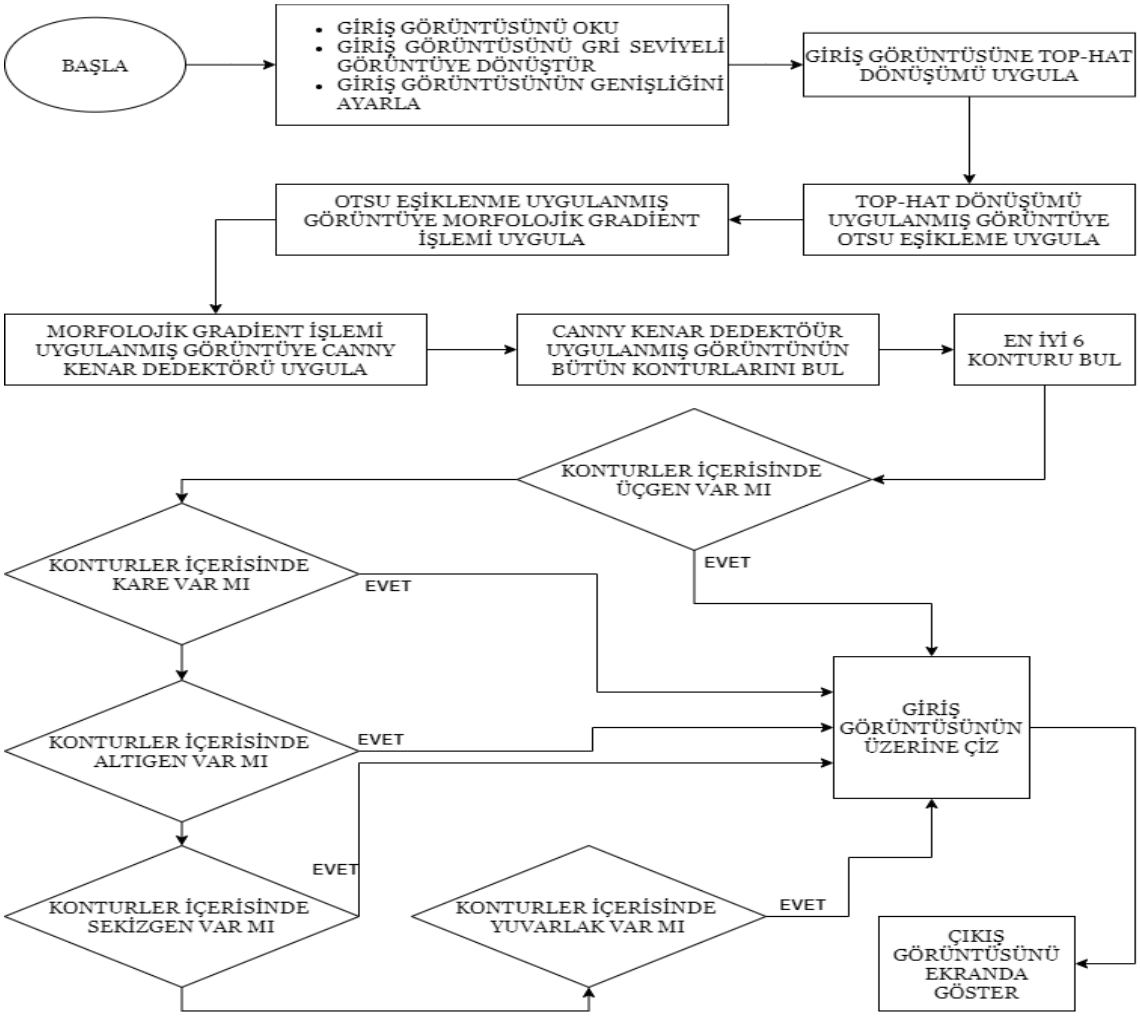
$$(x-a)^2 + (y-b)^2 = r^2 \qquad \textbf{(denklem-5)}$$

- ➔ Canny algoritması uygulanmış görüntü üzerine 3x3 Gauss Filtresi uygulanır.
- ➔ Elde edilen görüntüye Hough Dönüşümü uygulanır ve belirlenen yarıçap değerine göre çember bulunur.
- ➔ Denklem 5'de a ve b dairenin merkez koordinatlarıdır ve (a,b) şeklinde gösterilir. Denklemdeki r yarıçaptır.

➔ Bir 2D noktası (x, y) sabitse, parametreler denklem 5'e göre bulunur. (Parametre alanı üç boyutlu olacaktır (a, b, r))
➔ Bu üç parametre (a, b, r) sayesinde de görüntü üzerinde merkezi belirlenmiş bir çember çizilir.

Tüm bu işlemler yaptıktan sonra karşımıza çıkabilecek bütün trafik levhaları şekillerine bağlı olarak yüksek oranla bulunabilmektedir.

3. Algoritma Akış Şeması



Algoritma Akış Şeması - 1

4. Analiz ve Yorum

4.1. Eğitim Seti

Şekil 5 – input_image1.jpg

Şekil 6 – output_image1.jpg

→ Yukarıdaki Şekil 5'te kullanılan eğitim görüntüsünün sonucu Şekil 6'daki gibidir. Görüntüde yer alan üçgen, yuvarlak ve kare trafik işaret levhaları doğru bir şekilde tespit edilmiştir. Görüntüde yer alan üçgen trafik işaret levhasının iç ve dış yüzeyinde iki tane üçgen olduğu için hem iç üçgen hem de dış üçgen çizilmiştir. Kare ve yuvarlak trafik işaret levhalarının tespitinde herhangi bir sorun ile karşılaşılmamıştır. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

Şekil 7 – input_image2.jpg

Şekil 8 – output_image2.jpg

→ Yukarıdaki Şekil 7'te kullanılan eğitim görüntüsünün sonucu Şekil 8'deki gibidir. Görüntüde yer alan üçgen ve yuvarlak trafik işaret levhaları doğru bir şekilde tespit edilmiştir. Üçgen ve yuvarlak trafik işaret levhalarının tespitinde herhangi bir sorun ile karşılaşılmamıştır. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

Şekil 9 – input_image3.jpg

Şekil 10 – output_image3.jpg

→ Yukarıdaki Şekil 9'da kullanılan eğitim görüntüsünün sonucu Şekil 10'daki gibidir. Görüntüde yer alan üçgen trafik işaret levhasının iç ve dış yüzeyinde iki tane üçgen olduğu için hem iç üçgen hem de dış üçgen tespit edilmiştir. Arkada yer alan trafik işareti görüntü kalitesinden dolayı tespit edilememiştir.

Şekil 11 – input_image4.jpg

Şekil 12 – output_image4.jpg

→ Yukarıdaki Şekil 11'de kullanılan eğitim görüntüsünün sonucu Şekil 12'deki gibidir. Görüntüde yer alan yol trafik işaret levhasının tespiti doğru bir şekilde yapılmıştır. Görüntüde yer alan yuvarlak trafik işaret levhasının tespit edilememesi algoritmada yuvarlak trafik işaretleri için seçilen maksimum yarıçaptan kaynaklıdır. Tabelaların yakınlaştırılmış görüntülerine tasarlanan algoritma uygulandığında başarılı bir şekilde yuvarlak ve üçgen trafik işaretleri tespit edilmiştir.

Şekil 13 – input_image5.jpg

Şekil 14 – output_image5.jpg

→ Yukarıdaki Şekil 13'te kullanılan eğitim görüntüsünün sonucu Şekil 14'teki gibidir. Görüntüde yer alan yol trafik işaret levhasının tespiti doğru bir şekilde yapılmıştır. Görüntünün sağında yer alan tabela iki parçadan oluşuyor bu yüzden algoritma iki farklı kare tespit etmiştir. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

Şekil 15 – input_image6.jpg

Şekil 16 – output_image6.jpg

→ Yukarıdaki Şekil 15'te kullanılan eğitim görüntüsünün sonucu Şekil 16'daki gibidir. Görüntüde yer alan üçgen ve yuvarlak trafik işaret levhaları doğru bir şekilde tespit edilmiştir. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

Şekil 17 – input_image7.jpg

Şekil 18 – output_image7.jpg

→ Yukarıdaki Şekil 17'de kullanılan eğitim görüntüsünün sonucu Şekil 18'teki gibidir. Görüntüde yer alan üçgen ve yuvarlak trafik işaret levhaları doğru bir şekilde tespit edilmiştir. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

Şekil 19 – input_image8.jpg

Şekil 20 – output_image8.jpg

→ Yukarıdaki Şekil 19'da kullanılan eğitim görüntüsünün sonucu Şekil 20'deki gibidir. Görüntüde yer alan sekizgen, yuvarlak ve kare trafik işaret levhaları doğru bir şekilde tespit edilmiştir. Görüntüde yer alan kare trafik işaret levhasının iç ve dış yüzeyinde iki tane kare olduğu için hem iç hem de dış kare çizilmiştir. Aynı durum sekizgen trafik işaret levhası içinde geçerlidir. Tasarlanan algoritma başarılı bir şekilde çalışmıştır.

4.2. Test Seti

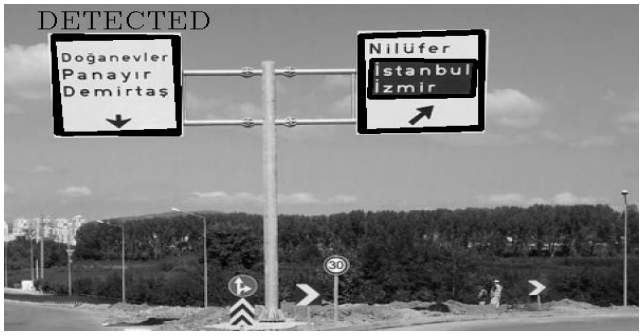
Test seti, tespit edilebilmesi sırasıyla **zordan kolaya** doğru giden görüntüleri içermektedir. Seçilmesi gereken 8 adet görüntüden ayrı olarak aynı algoritma üzerinde denenerek test edilmiştir.

Şekil 21 – test_input_image1.jpg

Şekil 22 – test_output_image1.jpg

→ Yukarıdaki Şekil 21'de kullanılan test görüntüsünün sonucu Şekil 22'deki gibidir. Görüntüde yer alan kare ve yuvarlak trafik işaret levhaları tespit edilmiştir. Arkada bulunan kare işaret levhasının tespit edilememesinin sebebi algoritmada tespit edilmesi istenilen kare işaret levhalarının alanı 20 pikselden büyük seçilmiştir. Görüntüde arkada yer alan kare işaretin tespit edilmemesinin sebebi alanının 20'den küçük olmasıdır.

Şekil 23 – test_ input_image2.jpg



Şekil 24 – test_output_image2.jpg

→ Yukarıdaki Şekil 25'te kullanılan test görüntüsünün sonucu Şekil 26'deki gibidir. Görüntüde yer alan kare trafik işaret levhaları tespit edilmiştir. Görüntünün altında yer alan yuvarlak trafik işaret levhaları tespit edilememiştir. Bunun sebebi algoritmada seçilmesi istenilen yuvarlak trafik işaretlerini maksimum yarıçap değeri 60'tır. Görüntüde yer alan trafik işaretlerinin yarı çapı 60 değerinden küçük olduğu için algoritma tespit edememiştir. Maksimum yarıçap değeri değiştirilirse yuvarlak trafik işaretlerinin tespit edildiği gözlemlenmiştir.

Şekil 25 – test_ input_image3.jpg

Şekil 26 – test_output_image3.jpg

→ Yukarıdaki Şekil 23'te kullanılan test görüntüsünün sonucu Şekil 24'teki gibidir. Görüntüde yer alan kare trafik işaret levhası tespit edilmiştir. Arkada bulunan kare işaret levhasının tespit edilememesinin sebebi algoritmada tespit edilmesi istenilen kare işaret levhalarının alanı 20 pikselden büyük seçilmiştir. Görüntüde arkada yer alan kare işaretin tespit edilmemesinin sebebi alanının 20'den küçük olmasıdır.

Şekil 27 – test_ input_image4.jpg

Şekil 28 – test_output_image4.jpg

→ Yukarıdaki Şekil 27'de kullanılan test görüntüsünün sonucu Şekil 28'deki gibidir. Görüntüde yer alan kare trafik işaret levhası tespit edilmiştir. Görüntüdeki kare işaretin altında yer alan "Konya" yazısının karesi algılanamamıştır. Bunun sebebi algoritmada alanı 20 piksel değerinden büyük olan kare işaretlerin çizilmesi seçildiği içindir. Alan değerinin değeri değiştirilirse "Konya" yazsının tespit edileceği gözlemlenmiştir.

NOT: Dosya boyutu 3MB sınırını geçmediği için anlatımlar görüntüler üzerinden yapılmıştır.

4. Sonuç – Yorum ve İyileştirme Önerisi

- Tasarlanan trafik işaret levhalarının tespit algoritması için 8 eğitim 4 test görüntüsü kullanılmıştır. 12 adet görüntüde toplamda 33 adet trafik işaret levhaları bulunmaktadır ve bunların 23 tanesi başarılı bir şekilde tespit edilmiştir. Elde edilen **başarı oranı %69** olarak hesaplanmıştır.

- Tasarlanan algoritmanın gri seviye görüntüler için başarılı olduğu gözlemlenmiştir. Başarısız olunan görüntülerin sebeplerinden birisi görüntüde yer alan trafik işaret levhalarının büyüklüğünün değişiklik göstermesi, diğeri görüntülerin kalitesidir.

- Trafik işaret şekilleri tespit edilirken yarıçap ve alan değeri göz önüne alınmaktadır. Yuvarlak trafik işaretleri için maksimum yarıçap değeri görüntülerde sabit olursa her yuvarlak trafik işaretinin algılanabileceği gözlemlenmiştir. Bu durumun çözümü için giriş görüntüsü okunduktan sonra giriş görüntülerinde en iyi sonucu veren piksel değerinde tekrardan boyutlandırılarak çözülmeye çalışılmıştır fakat tam sonuç elde edilememiştir.

- Kontur algoritması uygulanırken bazı görüntülerde kenar bulma işlemi sırasında üçgen levhalar için 4-6 ayrıta sahip olduğu, kare levhalar için 8 ayrıta sahip olduğu gözlemlenmiştir. Bu nedenle geliştirilen kontur çizme algoritması sonucu doğru şekilde vermesi için tasarlanmıştır.

- Önerilen yöntemler içerisinden uygulanan yöntemler:
  - ✓ Eşikleme → Otsu Eşikleme Metodu
  - ✓ Ayrıt Saptama → Canny Kenar Dedektörü
  - ✓ Morfolojik Operatörler → Top-Hat Dönüşümü ve Morfolojik Gradient işlemi
  - ✓ Çizgi / elips / line tespiti → Kontur Bulma ve Çizme Algoritması ve Çember Hough Dönüşümü
  - ✓ Filteleme → Sobel Filtresi ve Gauss Filtresi
  - ✓ Öznitelik Çıkartımı → Hough Dönüşümü

- En nihai olarak kaliteli ve uygun uzaklıktan çekilen görüntülerde algoritma başarılı bir şekilde çalışmaktadır. Yukarıda belirtilen durumlar göz önünde bulundurulursa başarı oranında artış gözlemlenebilir.

- **Başarı oranını artırmak için;** bağlantılı bileşen analizi sayesinde eksantrik değerlerine göre şekiller ayırt edilip direkt olarak objeye odaklanarak (yalnızca odaklanılan objenin kalması ve diğer görüntünün silinmesi ile) sonuca gidilebilirdi. (Sadece tahminden ibarettir, denenmemiştir)

5. Kaynaklar

1. Rafael C. GGonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing Using MATLAB".
2. P. Ponce, S. S. Wang, D. L. Wang, "License Plate Recognition-Final Report", Department of Electrical and Computer Engineering, Carnegie Mellon University, 2000.
3. https://www.programcreek.com/python/example/93640/imutils.resize
4. https://docs.opencv.org/3.4/d4/d76/tutorial_js_morphological_ops.html
5. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html
6. https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html
7. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html
8. https://github.com/spmallick/learnopencv/tree/master/otsu-method
9. https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/
10. https://www.pyimagesearch.com/2021/04/28/opencv-morphological-operations/
11. https://github.com/mesutpiskin/computer-vision-guide/blob/master/docs/10-morfolojik-goruntu-isleme.md
12. https://www.projectpro.io/recipes/what-is-morphological-gradient-of-image-opencv
13. https://medium.com/analytics-vidhya/morphological-transformations-of-images-using-opencv-image-processing-part-2-f64b14af2a38
14. Rivest, Jean-François & Soille, Pierre & Beucher, Serge. (1993). Morphological gradients. J. Electronic Imaging. 2. 326-336. 10.1117/12.159642.
15. https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html
16. https://java2blog.com/cv2-canny-python/
17. https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html
18. https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/
19. https://java2blog.com/cv2-findcontours-python/
20. https://learnopencv.com/contour-detection-using-opencv-python-c/
21. https://stackoverflow.com/questions/62274412/cv2-approxpolydp-cv2-arclength-how-these-works
22. https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html
23. https://pretagteam.com/question/cv2approxpolydp-cv2arclength-how-these-works
24. https://www.tutorialkart.com/opencv/python/opencv-python-gaussian-image-smoothing/
25. https://docs.opencv.org/4.x/d3/de5/tutorial_js_houghcircles.html
26. https://en.wikipedia.org/wiki/Circle_Hough_Transform