

PLAKA TANIMA SISTEMI

SAYISAL GÖRÜNTÜ İŞLEME PROJESİ

211312073 - Hasan Batuhan Akay



PLAKA TANIMA SISTEMİ NEDİR?

Plaka Tanıma Sistemi, kameralardan alınan araç görüntülerinin işlenmesiyle gerçekleştirilir. Bu sistemde, araç görüntüsü üzerinde plaka bölgesinin tespiti yapılır, ardından plaka bölgesi ayırtılırlar. Optik karakter tanıma (OCR) yöntemleri ve görüntü işleme teknikleri kullanılarak plakada yer alan karakterlerin okunması sağlanır. Projenin uygulama alanı geniş bir yelpazeye yayılabilir ve sistem, uygulamanın ihtiyaçlarına göre özel olarak tasarlanmış algoritmalar ve donanım yapıları kullanılarak geliştirilir. Bu sayede, araçların belirlenmesi ve takip işlemlerinde etkili bir çözüm sunar.



UYGULAMA ALANLARI

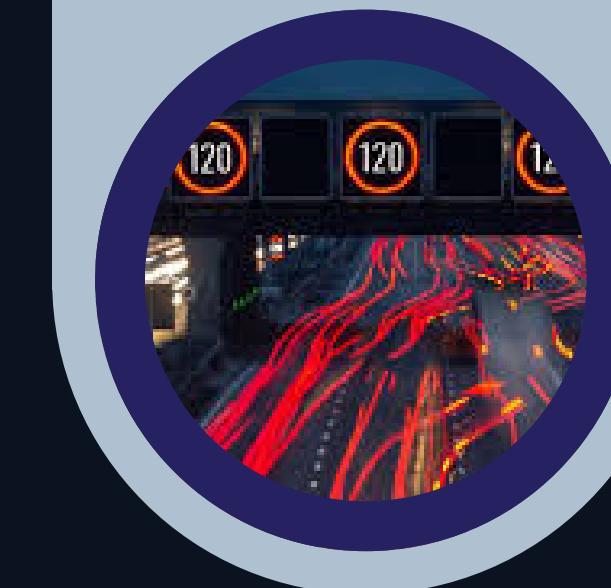


Giriş Kontrolü

Otoparklar, siteler ve işyerleri gibi yerlerde araç giriş çıkışlarını kontrol etmek için kullanılır.

Trafik Denetimi

Hız sınırlarını ihlal eden araçları tespit etmek, trafik ışığı ihlalleri ve diğer trafik kuralı ihlallerini belirlemek için kullanılır.



Otoyol Geçiş Ücreti Toplama

Otomatik geçiş sistemlerinde (OGS ve HGS) araç plakalarını tanıyararak ücret tahsilatı yapılır.

Güvenlik ve Emniyet

Plaka tanıma sistemleri, çalıntı veya aranan araçların tespiti için emniyet birimleri tarafından kullanılır.



LITERATÜR TARAMASI

Plaka tanıma sistemleri üzerine yapılan akademik çalışmalar, bu alandaki teknolojik gelişmelerin temelini oluşturur. İşte bu alanda öne çıkan bazı çalışmalar:

- **Zhang, W., et al. (2017):** Bu çalışmada, konvolüsyonel sinir ağları (CNN) kullanılarak yüksek doğrulukla plaka tanıma sağlanmıştır. Çalışma, özellikle düşük kaliteli görüntülerde bile başarılı sonuçlar elde edilmiştir.
- **Silva, S., et al. (2018):** Araç plaka tanıma sistemlerinde derin öğrenme tekniklerinin kullanımını inceleyen bu çalışma, farklı veri setleri üzerinde yüksek performans göstermiştir.
- **Anagnostopoulos, C. N. E., et al. (2008):** Bu çalışmada, araç plaka tanıma sistemlerinin genel bir incelemesi yapılmış ve farklı yöntemlerin karşılaştırılması sunulmuştur.



PLAKA TANIMA SİSTEMLERİNİN TEMELLERİ

Plaka Tespiti

Sistem, araçların görüntülerini çeker ve bu görüntülerde plaka bölgesini tespit eder. Bu işlemde genellikle kenar bulma, renk filtresi ve segmentasyon gibi görüntü işleme teknikleri kullanılır.

Plaka Bölgesi Ekstraksiyonu

Tespit edilen plaka bölgesi görüntünden çıkarılır ve daha fazla işleme tabi tutulur. Bu aşamada, plakanın eğikliği ve perspektif hataları düzelttilir.

Karakter Bölütme

Plaka üzerindeki karakterler ayırtılırlar. Bu aşama, her bir karakterin plaka bölgesinden ayrı ayrı çıkarılmasını içerir. Karakter bölütme, genellikle morfolojik işlemler ve bağlantı bileşenleri analizi ile gerçekleştirilir.

Karakter Tanıma

Ayrırtılan karakterler, optik karakter tanıma (OCR) teknikleri kullanılarak tanınır. Bu işlemde, yapay sinir ağları, destek vektör makineleri (SVM) veya şablon eşleştirme gibi algoritmalar kullanılabilir.

GÖRÜNTÜ İŞLEME TEKNİKLERİ

Kenar Bulma

Kenar bulma algoritmaları, görüntüdeki keskin değişiklikleri tespit eder ve plaka bölgesinin belirlenmesine yardımcı olur. Canny, Sobel ve Laplace gibi kenar bulma operatörleri yaygın olarak kullanılır.

Renk Filtreleme

Renk滤releme, belirli bir renk aralığındaki pikselleri tespit ederek plaka bölgesinin belirlenmesine yardımcı olur. Plakaların genellikle belirli renklerde olduğu varsayıldığında, bu teknik oldukça etkilidir.

Şablon Eşleştirme

Bu teknik, tanımlanmış bir plaka şablonunu görüntü üzerinde arayarak plaka bölgesini tespit eder. Şablon eşleştirme, özellikle belirli bir plaka formatının kullanıldığı durumlarda etkilidir.

PLAKA TANIMA SİSTEMİNDE KULLANILAN YÖNTEMLER

Canny yöntemi

Kodda Canny Kenar Algılama Yöntemi tercih edilmiştir çünkü:

- Daha kesin ve anlamlı kenarlar sağlar.
- Gürültüye dayanıklıdır.
- Plaka gibi belirgin çerçeveli nesnelerin kenarlarını tespit etmek için uygundur.
- Gerçek zamanlı performans gerektiren uygulamalarda hızlı ve etkili sonuç verir.

Maskeleme

Görüntü üzerinde yalnızca plaka alanını izole ederiz ve arka planı temizleriz. Bu işlem gereksiz detayların işleme sürecine girmesini engeller.

EasyOCR

Plaka bölgesindeki harf ve rakamların hızlı ve doğru bir şekilde tanımlanabilmesi için kullanıldı. Model eğitmeye gerek kalmadan doğrudan bir OCR çözümü sağladı bize.



KULLANILAN KÜTÜPHANELER

datetime: İşlem zaman damgalarının oluşturulması için.

time: Fonksiyonu saniye cinsinden zaman ölçümü yapmak için kullanılır.

numpy: Görüntü verileri üzerinde işlemler için sayısal hesaplama işlemlerini sağlar.

cv2 (OpenCV): Görüntü işleme, kontur algılama ve maskeleme işlemlerini gerçekleştirir.

imutils: Görüntü işleme sırasında konturları işlemek için yardımcı fonksiyonlar sağlar.

save: Tanımlı bir dosyaya veri yazma işlemlerini gerçekleştirir.

easyocr: Görüntüden metin tanıma işlemlerini yapar.

KODLAR

```
● ● ●  
1 import numpy as np  
2 import cv2  
3 import imageProcessing as imgprocess  
4  
5 cap = cv2.VideoCapture(0)  
6  
7 while True:  
8     ret, frame = cap.read()  
9     img = imgprocess.rec(frame)  
10    cv2.imshow('Screen', img)  
11    if cv2.waitKey(20) & 0xFF == ord('q'):  
12        break  
13  
14 cap.release()  
15 cv2.destroyAllWindows()
```

5. **cap = cv2.VideoCapture(0)** : Bu komut, bilgisayarın varsayılan kamerasını başlatır.
7. **while True**: Sonsuz bir döngü başlatılır. Bu döngü, video işleme işlemlerinin sürekli olarak yapılmasını sağlar.
8. **ret, frame = cap.read()** : Kameradan bir kare (frame) alır. ret değeri True ise görüntü başarılı bir şekilde alındığını belirtir.
9. **img = imgprocess.rec(frame)** : imageProcessing modülünde tanımlı rec() fonksiyonu, frame'i alır ve işlenmiş bir görüntü döndürür.
10. **cv2.imshow('Screen', img)** : İşlenmiş görüntüyü ekranда gösterir. img, rec() fonksiyonundan dönen işlenmiş görüntündür.
11. **if cv2.waitKey(20) & 0xFF == ord('q')**: 20 milisaniye boyunca bir tuşa basılıp basılmadığını kontrol eder. Eğer kullanıcı 'q' tuşuna basarsa, döngüden çıkarılır (break komutu).
14. **cap.release()** : Kamerayı serbest bırakır. Bu işlem, video işleme işlemlerinden sonra kameranın doğru bir şekilde kapanmasını sağlar.
15. **cv2.destroyAllWindows()** : Tüm OpenCV pencerelerini kapatır. Bu, ekranın açık kalan pencerelerin düzgün bir şekilde kapatılmasını sağlar.

KODLAR

```
● ● ●  
1 import cv2  
2 import imutils  
3 import numpy as np  
4 import save  
5 from unittest import result  
6 import easyocr  
7 def rec(img):  
8     #Filtreleme işlemi  
9     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
10    gray = cv2.bilateralFilter(gray, 11, 17, 17)  
11    edged = cv2.Canny(gray, 30, 200)  
12    try:  
13        cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  
14        cnts = imutils.grab_contours(cnts)  
15        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]  
16        screenCnt = None  
17        for c in cnts:  
18            peri = cv2.arcLength(c, True)  
19            approx = cv2.approxPolyDP(c, 0.017 * peri, True)  
20            if len(approx) == 4:  
21                screenCnt = approx  
22                break  
23            if screenCnt is None:  
24                detected = 0  
25            else:  
26                detected = 1  
27  
28            if detected == 1:  
29                cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)
```

def rec(img): İşlenmiş görüntü (img), konturlar ve tespit edilen plaka bilgileriyle birlikte geri döndürülür.

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY): Görüntüyü gri tonlamaya çevirir. Bu, işlemlerdeki gereksiz renk bilgilerini atarak sadece yoğunluk değerlerini kullanmayı sağlar.

gray = cv2.bilateralFilter(gray, 11, 17, 17): Gürültüyü azaltırken kenarların korunmasını sağlar. Plakayı daha net hale getirir.

edged = cv2.Canny(gray, 30, 200): Görüntüdeki kenarları tespit eder. Plakanın kenarlarını belirlemek için kullanılır.

cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE) : Kenar algılama sonucunda tespit edilen nesnelerin dış hatlarını bulur.

cnts = imutils.grab_contours(cnts): OpenCV'nin kontur çıktı formatını işler.

cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]: En büyük 10 konturu seçer (alanlarına göre sıralanır). Plaka genelde büyük bir dikdörtgendir.

for c in cnts:

peri = cv2.arcLength(c, True): Konturun çevresini hesaplar.

approx = cv2.approxPolyDP(c, 0.017 * peri, True): Konturu yaklaşık bir çokgene dönüştürür. Plakalar 4 köşeli bir oldukları için len(approx) == 4 koşulu kontrol edilir.

if len(approx) == 4:

screenCnt = approx: İlk dörtgen şekilli kontur screenCnt olarak seçilir.

if detected == 1:

cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3): Tespit edilen plakayı, görüntü üzerinde yeşil bir çizgiyle işaretler.

```
● ● ●  
1 #Maskeleme işlemi  
2 mask = np.zeros(gray.shape, np.uint8)  
3 new_image = cv2.bitwise_and(img, img, mask=mask)  
4 new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1,)  
5  
6 #Kesme işlemi  
7 (x, y) = np.where(mask == 255)  
8 (topx, topy) = (np.min(x), np.min(y))  
9 (bottomx, bottomy) = (np.max(x), np.max(y))  
10 Cropped = gray[topx:bottomx + 1, topy:bottomy + 1]  
11  
12 #EasyOCR işlemi  
13 reader = easyocr.Reader(['en'])  
14 text = reader.readtext(Cropped)  
15 text = text[0][1]  
16 text = text.replace(" ", "")  
17 text = text.upper()  
18 sonuc = 0
```

KODLAR

MASKELEME İŞLEMİ:

mask = np.zeros(gray.shape, np.uint8): Bir "mask" adı verilen tamamen siyah (piksel değerleri 0) bir görüntü oluşturduk.

new_image = cv2.bitwise_and(img, img, mask=mask): Maskeyi orijinal görüntüye uygular. Sadece plaka bölgesini bırakır.

new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1)

Maske üzerindeki yalnızca plaka bölgesi beyaza (255) dönüşür; diğer bölgeler siyah (0) kalır.

EASYOCR İŞLEMİ:

reader = easyocr.Reader(['en']):

EasyOCR kütüphanesi ile bir OCR okuyucu nesnesi oluşturur.

text = reader.readtext(Cropped):

Kesilen plaka bölgesindeki (Cropped) yazıları okur.

text = text.replace(" ", ""):

Plakada okunan metindeki boşlukları (" ") temizler.

KESME İŞLEMİ:

(x, y) = np.where(mask == 255): Maskede beyaz olan (255) bölgeleri bulur ve plaka bölgesini keserek alır.

(topx, topy) = (np.min(x), np.min(y))

(bottomx, bottomy) = (np.max(x), np.max(y)) : Beyaz piksellerin (plaka bölgesinin) en üst, en alt, en sol ve en sağ noktalarını bulur.

Cropped = gray[topx:bottomx + 1, topy:bottomy + 1] Tespit edilen plaka bölgesini orijinal gri tonlamalı görüntüsünden (gray) kesip çıkarır.

KODLAR

```
● ● ●  
1 if(len(text)>=7):  
2     if(int(text[0]) and int(text[1])):  
3         if(str(text[4])):  
4             if(int(text[5:])):  
5                 sonuc=1  
6         else:  
7             if(int(text[4:])):  
8                 sonuc=1  
9 if(sonuc==1):  
10     save.write(text)  
11     print(text)  
12  
13 except Exception:  
14     pass  
15 return img
```

return img :

İşlenmiş görüntüyü (img) çağrıran koda geri döndürür.

except Exception:

Pass :

Hata yakalandığında hiçbir işlem yapılmaması videonun devam etmesini sağlar.

if(len(text) >= 7): Plaka uzunluğunun en az 7 karakter olmasını istedik .

if(int(text[0]) and int(text[1])): Plakanın ilk iki karakterinin rakam olduğunu doğrular.

if(str(text[4])): 4. karakterin bir harf veya uygun bir sembol olduğundan emin olur.

if(int(text[5:])): 5. karakterden itibaren geri kalan kısmın rakam olduğunu kontrol eder.

Sonuç=1: Eğer tüm bu koşullar sağlanıyorsa, sonuc değişkeni 1 olur (geçerli plaka), aksi halde 0 kalır.

else:

if(int(text[4:])):

sonuc = 1: Eğer plaka metninin belirli bir formatta olduğunu kontrol edemiyorsa, metnin geri kalanının sayılarından olduğunu doğrular.

if(sonuc == 1): Plakanın belirli bir formatta olup olmadığını kontrol eder.

save.write(text): Tespit edilen plaka metnini bir dosyaya kaydeder.

print(text) : Geçerli plakayı konsola yazdırır.

KODLAR

```
● ● ●  
1 import datetime  
2  
3 def write(message):  
4     f = open("data.txt", "r")  
5     result = f.read().find(message)  
6     f.close()  
7     if result == -1:  
8         time = datetime.datetime.now()  
9         f = open("data.txt", "a")  
10        f.write(message + " ")  
11        f.write(str(time)+"\n")  
12        f.close()
```

f = open("data.txt", "r"): data.txt adlı dosyayı okuma modunda açar. Bu dosya, daha önce kaydedilmiş mesajları kontrol etmek için kullanılır.

result = f.read().find(message): Dosyanın içerisinde, message değişkeninin bulunup bulunmadığını kontrol eder. Eğer bulunursa, find() fonksiyonu mesajın indeksini döner; eğer bulunmazsa -1 döner.

if result == -1: Eğer mesaj dosyada daha önce yazılmamışsa (result == -1), yeni işlem gerçekleştirilir.

time = datetime.datetime.now(): Şu anki tarih ve zaman bilgisini alır.

f = open("data.txt", "a"): data.txt dosyasını ekleme (append) modunda açıyor. Bu, dosyaya veri eklemek için kullanılır.

f.write(message + " "): Mesajı dosyaya yazar.

f.write(str(time) + "\n"): Zaman damgasını dosyaya yazar.

f.close(): Dosyayı kapatmak için close çağrısı yapar.

KAYNAKÇA

- <https://www.hobilisim.com/pts-nedir/>
- **OpenCV Documentation:** <https://docs.opencv.org/>
- **EasyOCR:** <https://github.com/JaideAI/EasyOCR>
- **Datetime(Python Standart Kütüphanesi):**
<https://docs.python.org/3/library/datetime.html>
- **Imutils:**
<https://github.com/PylImageSearch/imutils>
- **NumPy (numpy):** <https://numpy.org>



TEŞEKKÜRLER

