**T.C.**

**KOCAELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**



**OPENCV VE QT İLE PLAKA TANIMA PROJESİ**

**CİHAN SAVAŞ İSMET KARADUMAN**

**130202098 140202098**

OPENCV VE QT İLE PLAKA TANIMA PROJESİ

Cihan SAVAŞ1,İsmet Karaduman2

1,2Bilgisayar Mühendisliği(2.Öğretim)

Kocaeli Üniversitesi

1cihansavas.85@gmail.com, 2ikaraduman@yahoo.com

Özet

*Plaka tanıma sistemleri araç tanıma otomasyonunda kullanılan tekniklerden birisidir. Trafik denetleme, gişe otomasyonu ve denetimli saha giriş kontrolü (havaalanı ,hastane, askeri tesis vb.) uygulamalarında verimli olarak kullanılırlar. Bu projede, Türkiye Cumhuriyeti plaka standartlarına uygun sivil plakalı araçlar için bir plaka tanıma sistemi geliştirilmiştir. Projede öncelikle plakası tespit edilecek araca ait resim üzerinden araç plakasının bulunduğu bazı hesaplamalar ve işlemler ile tespit edilmekte ve ayrı bir resim olarak kayıt edilmektedir. Daha sonra elde edilen bu resim üzerinden K-Nearest algoritması kullanılarak resim içerindeki karakterler tespit edilmeye çalışılmaktadır. Ayrıca gerçekleştirilen çalışmada hem plaka tespit aşamasında hem de karakter tespit aşamasında CV\_BGR2GRAY, GaussianBlur, adaptiveThreshold gibi komutlar kullanılarak resim üzerinde dönüştürme işlemleri gerçekleştirilmiştir. FindContours komutu ile karakter sınır bölgeleri tespit edilerek karakter sayısı boyutu ve türü, çeşitli görüntü işleme rutinleri ile algılanmıştır.*

# Giriş

Günümüzde, gerek araç giriş çıkışın kontrollü yapıldığı güvenlik kontrol noktalarında gerekse trafik akışının kontrollü bir şekilde sağlandığı noktalar üzerinde yapılan çalışmaların çoğunda genel olarak araçları özel bir noktadan geçerken tanımlamak, aracın konumunu belirlemek, davranışını gözlemlemek ve buna göre trafik denetimi sağlamak esastır. Genelde araç plaka tanıma işlemleri iki temel adımda gerçekleştirilir; görüntü üzerinde plakanın bulunduğu alanın tespiti ve daha sonrasında bu alan üzerindeki karakterlerin tanınması. Bu doğrultuda bu çalışmada da öncelikle plaka yerinin tespit edilmesi üzerine çalışılmış sonrasında da tespit edilen bölgede bulunan karakterler imagelerinin harfsel ve rakamsal karşılıkları tespit edilmeye çalışılmıştır. Türkiye karayolu trafiğinde plakalar; sivil, resmi, askeri, diplomatik vs. gibi deşişik renk ve formatta olabilmektedir. Bu çalışmada, Türkiye Cumhuriyeti plaka standartlarına uyan *sivil* plakaların tanınması amaçlanmıştır. Bu plakaların genel özelliği; beyaz zemin üzerine siyah karakterlerden oluşması, ilk iki karakterin şehir kodunu belirtmesi ve ondan sonra gelen karakterlerin rastgele harf ve rakam dizisinden oluşmasıdır.

# Temel Bilgiler

Görüntü işleme, bilgisayar ortamında bulunan resimlerin, isteğe uygun bir şekilde düzenlenmesi işlemidir. Görüntü işleme var olan resim üzerinde gerçekleştirilmektedir[1].

**Görüntü İşlemedeki temel amaç**, resmi istenilen efektlerle güzelleştirmek veya istenilen boyuta getirmektedir. Bunlara ek olarak; kullanılan gerekli algoritmalarla yüz tanıma, nesne tanıma gibi güvenlik ve robot endüstrisinde de kullanılmaktadır.

Görüntü işleme işlemlerinden birçok alanda yararlanılmaktadır. Bunlardan bazıları şunlardır;

* Askeri Endüstride yaygın olarak kullanılmaktadır.
* Güvenlik, radar, astronomi, trafik sistemlerinde ve kriminal laboratuarlar da kullanılmaktadır.
* Biyomedikal alalarda kullanılmaktadır.
* Uydu görüntüleme de kullanılmaktadır.

Görüntü işlemede kullanılan bazı terimler;

**Ölçeklendirme;** Resmi istenilen boyutta büyültüp küçültme işlemidir.

**Döndürme;** Resmi istenilen açı ile kendi ekseni etrafında çevrilmesi işlemdir.

**Yansıtma;** Resmin kendi görüntüsü sabit kalıp, belli bir açı veya 180 derece ayrı bir şekilde aynalama (miror) yapılması işlemine denir.

**Renk düzeltmesi;** Resmin üzerindeki renkleri istenilen tona göre ayarlanması işlemi.

**Pixel;** Elektronik ortamda, resmin en küçük birimine denmektedir.

**OpenCv Nedir?**

Bilgisayar ortamında görüyü işlemeyi sağlayan bir kütüphanedir. Windows, Linux ,macOS X gibi bir çok platformda çalıştırılabilir ,ayrıca C, C++, python ve Java gibi dillerde kütüphanesi bulunmaktadır[1].

OpenCV kütüphanesi, BSD lisansı ile lisanslanmıştır. Özgür lisanslar içinde en özgürü olarak bilinen bu lisansta kodu alan kişi, istediği gibi kullanma özgürlüğüne sahiptir [2].

Intel’in görüntü işleme laboratuarlarında geliştirilen ve hız açısından optimize edilen OpenCV kütüphanesi, gerçek zamanlı uygulamalar hedef alınarak geliştirilmiştir. USB 2.0 teknolojisi ile birlikte artık standart bir bilgisayarda bile gerçek zamanlı uygulamalar çalıştırılabilmektedir. Tüm bu gelişmeler oyuncak yapımından endüstriyel üretime kadar pek çok alanda bu kütüphanenin kullanılmasına yol açmıştır [3].

OpenCV kütüphanesi, beş temel bileşenden oluşmaktadır. Computer Vision (Bilgisayarla Görü/Görme) kelimesinin baş harfleri kullanılarak isimlendirilen CV bileşeni, temel resim işleme fonksiyonları ve Bilgisayarla Görü/Görme için kullanılan yüksek seviyeli algoritmaları bünyesinde barındıran beş temel kütüphaneden biridir. Machine Learning Library kelimesinin baş harfleri kullanılarak isimlendirilen MLL bileşeni, adından da anlaşılacağı üzere Makina Öğrenmesi dalı için gerekli istatistiksel verilere ulaşmak, mevcut verileri sınıflandırmak için kullanılan fonksiyonları/araçları içeren diğer bir kütüphanedir. HighGUI bileşeni, slider, form gibi OpenCV kütüphanesi içerisinde tanımlanmış pek çok nesneyi yaratabilmemizi sağlayan bir grafik arabirimi olmakla beraber, resim ve videoları kaydetmek, yüklemek, hafı- zadan silmek için gerekli giriş/çıkış (I/O) fonksiyonlarını da içerir [4].

CXCore bileşeni, OpenCV’ye ait IplImage, cvPoint, cvSize, cvMat, cvHistogram... vs gibi veri yapılarını bünyesinde barındıran, XML desteği de sağlayan bir kütüphanedir. Son olarak CvAux bileşeni, şablon eşleştirme (template-matching), şekil eşleştirme (shape matching), bir objenin ana hatlarını bulma (finding skeletons), yüz tanıma (face-recognition), ağız hareketleri izleme (mouth-tracking), vü- cut hareketlerini tanıma (gesture recognition) ve kamera kalibrasyonu gibi daha pek çok deneysel algoritmaları bünyesinde barındıran kütüphanedir [4].

**Qt Nedir ?**

Qt, belli bir platforma bağımlı kalmadan uygulamalar yapmak amacıyla oluşturulmuş bir geliştirme ortamı, aynı zamanda da bir geliştirme kütüphanesidir[5].1995 yılında Trolltech adlı Norveç’ li bir firma tarafından geliştirilmiştir. Daha sonra ise Haziran 2008 de Nokia tarafından satın alınmıştır[6].

Qt kullanarak geliştirdiğiniz uygulamaları üzerinde herhangi bir değişiklik yapmadan pek çok masaüstü bilgisayar ve gömülü işletim sistemlerinde kullanabilirsiniz. Qt, MacOS 10.2.8+, X11 ile UNIX (Linux, FreeBSD, Solaris) ve Windows 98/NT/2000/XP ve üzerini desteklemektedir. Qt ile yazmış olduğunuz kodları bu platformlar arasında rahatça taşıyabilirsiniz.[6]

Qt genellikle görsel uygulamalar yazacak C++ geliştiricileri tarafından tercih edilmektedir. Ancak Qt yi sadece bir görsel uygulama kütüphanesiymiş gibi düşünmek de yanlış olur. Qt bünyesinde her türlü araç ve kütüphaneyi barındıran çok geniş bir ortamdır. Qt ‘nin içerisinde veritabanı ve network uygulamaları geliştirmek için gerekli birçok fonksiyon bulunmaktadır. Özellikle veritabanı konusunda Qt çok büyük kolaylıklar sağlamaktadır. Qt,  Oracle,Ms Sql Server, Sybase Adeptive Server, IBM DB2, PostgreSQL, MySQL, Borland Interbase, SQLite, ve ODBC-uyumlu tüm veritabanlarını desteklemektedir[6].

**Qt Yararları Nelerdir?**

Qt de rogram yazmanın programcıya sağladığı yararları şu şekilde sıralayabiliriz[7].

* Qt'nin en büyük yararı aslında, C++'ın  yüksek performansı ve görselliğin bu performansla birlikte kullanılmasını sağlamasıdır.
* Qt'de yazdığımız kodlar platform bağımsız çalışabilir.
* **C++** ile görsel bileşenleri kullanabilmek için bir çok satır kod yazma zorunluluğu Qt ile ortadan kalkmakta bu konuda Qt yazılımcılara esneklikler sağlamaktadır
* Qt ile veritabanı erişimi uygulamaları,ağ erişim uygulamaları, dosyalar, GUI  araç takımıyla birlikte programlama imkanı ve bir çok faydası var.
* Qt'nin bir çok dökümantasyonu mevcuttur. Örnek uygulamaları inceleyerek, kendinizi geliştirip, daha da üzerine koyarak, iyi yazılımlar gerçekleştirebilirsiniz.
* Qt bizlere **C++**'ın hızını kullanabilmemizi sağlar. Böylece yazılımda performans artışı sağlanır

# Geliştirilen Mimari

## Projede Karşılaşılan Zorluklar

Türk sivil plaka standardına göre motorlu taşıt plakaları 110mmX526mm,77mm karakter yüksekliğine ve 10mm karakterler arası açıklığa sahip tek sıra dizilmiş karakterlerden oluşmalıdır. Plakaların resmi bir standardı olmasına rağmen, trafikteki araçların önemli bir kısmı standartlara uygun olmayan plakalara sahiptir. Ayrıca plakalar üzerinde çeşitli çıkartmalar, fosforlar, pullar,vidalar ve çamur gibi yabancı nesneler bulunması plaka tanıma otomasyonunu güçleştiren faktörlerdir.

## Projede Gerçekleştirilen İşlem Adımları

Projenin ilk aşamasında üzerinde işlem gerçekleştirilecek araca ait görüntü dosyası projeye dahil edilmektedir. Projede test için kullanılan bazı araçlara ait görüntüler Şekil 1, 2 ve 3 te görülmektedir.



Şekil-1 : Örnek Araç 1



Şekil-2 : : Örnek Araç 2



Şekil-3 : Örnek Araç 3

Araca ait görüntü sisteme dahil edildikten sonra öncelikle araç plakasının yerinin belirlenmesi gerekmektedir. Bu adım ile ilgili gerçekleştirilen işlem adımları şu şekildedir;

Öncelikle araca ait resim üzerinde CV\_BGR2GRAY, ADAPTIVE\_THRESH\_GAUSSIAN\_C, ADAPTIVE\_ THRESH\_ MEAN\_C renk uzayları uygulanır (Şekil-4)



Şekil-4 : Renk Uzayı İşlemleri

**İlgili Kod Bloğu:**

cv::Mat kaynakresim;

cv::Mat Kaynak\_Gri;

cv::Mat Kaynak\_Thresh;

kaynakresim = cv::imread("araclar/1.jpg");

cv::cvtColor(kaynakresim,Kaynak\_Gri,CV\_BGR2GRAY);

cv::adaptiveThreshold(Kaynak\_Gri,Kaynak\_Thresh,255,cv::ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv::ADAPTIVE\_THRESH\_MEAN\_C, 11, 2);

Bu işlemler ile kaynakresim üzerindeki pixel değeri sadece 0 (siyah) ve 255 (beyaz) değeri olarak değiştirilmiş olur. Renk uzayı işlemleri sonrasında görüntüdeki renk değeri “at” komutuyla okunur. Okunan değerlerde siyah pixellerin sayısı kontrol işlemi yapılır eğer siyah pixel sayısı istenilen değer aralığında ise (genişlik ve boy olarak) bu okunan pixeller kenar olarak değerlendirilerek kenar dizisine kaydedilir. Resim tarama işlemi bittikten sonra kenar dizisi kontrol işlemleri yapılır. Plaka kenarları olarak kabul edilen kenarların hassasiyet değerleri ve en (2 kenarın x kordinatları arasındaki fark), boy (bir kenarın başlangıç ve bitiş arasındaki y kordinat farkı), ve oran (boy/en ) değerleri gözönünde bulundurularak kontrolleri yapılır. Eğer yapılan kontroller sonucunda bu kenarlara ait değerler istenilen aralıkta ise bu kontrollerden başarılı olarak geçen kenarlar, plakanın kenarları olarak değerlendirilir ve kare dizisine atanır.

**İlgili Kod Bloğu:**

int pixelrengi;

int kenar[kaynakresim.cols][3], kare[kaynakresim.cols][6], kenarsayisi=0;

int x,y,siyah\_say=0;

//Resim tarama işlem başlangıçı

for(x=0;x<kaynakresim.cols;x++)++)

{

for(y=0;y<kaynakresim.rows;y++)

{

//Taranan resmin pixel renk değeri alınıyor

pixelrengi=kaynakresim.at<unsigned char>(y,x);

//Resimdeki pixel değeri siyah ise;

if(pixelrengi==0 )

{

siyah\_say++;

}

else if(pixelrengi==255 )

// Taranan pixel’in değeri beyaz ise;

{

if(siyah\_say>30 ) )

//(Siyah sayısı 30 dan fazla ise kenar olarak kabul et ve //kenar kordinatlarını matrise gönder)

{

kenar[kenarsayisi][0]=x;

kenar[kenarsayisi][1]=y-siyah\_say;

kenar[kenarsayisi][2]=y;

kenarsayisi++;

siyah\_say=0;

}

siyah\_say=0;

}

}

}

for (int var = 0; var < kenarsayisi; ++var)

{

cv::rectangle(kaynakresim,cv::Point(kenar[var][0]-20, kenar[var][1]), cv::Point(kenar[var][0] +20,kenar[var][2]),cv::Scalar(100,100,50),1,cv::LINE\_4,0);

}

İnt sonuc, en, oran, boy, kontrol=0 , karesayisi=0, hassasiyet;

hassasiyet=5;

for (int x = 0; x < kenarsayisi; x++)

//Kenar matrisi kontrol ediliyor

{

for (int y = 0; y < kenarsayisi; y++)

//Karşılaştırma yapılacak olan 2.Kenar seçiliyor

{

if (x!=y)

{

sonuc= kenar[y][1]-kenar[x][1];

//2 kenarın X kordinatları arasındaki

//pixel farkı kontrol ediliyor

if (sonuc<=hassasiyet && sonuc>=0)

{

kontrol++;

}

sonuc= kenar[y][2]-kenar[x][2];

//2 kenarın Y kordinatıları arasındaki

//pixel farkı kontrol ediliyor

if (sonuc<=hassasiyet && sonuc>=0)

{

kontrol++;

}

//Eğer kenarlar yukarıdaki X ve Y kordinat //Hassasiyetinden geçebildiyseler kare bir alan olabilir

if (kontrol==2)

{

boy=kenar[y][0]-kenar[x][0];

en=kenar[x][2]-kenar[x][1];

oran=boy/en;

if(boy>en )

{

//PLAKA boyutu sorgulama

if(oran>=4 && oran<=5 && boy>=120 &&boy<=300 && en>=40)

{

//qDebug()<<oran<<boy<<en;

kare[karesayisi][0]=kenar[x][0];

kare[karesayisi][1]=kenar[x][1];

kare[karesayisi][2]=kenar[x][2];

kare[karesayisi][3]=kenar[y][0];

kare[karesayisi][4]=kenar[y][1];

kare[karesayisi][5]=kenar[y][2];

karesayisi++;

}

}

}

kontrol=0;

}

}

}

//Kare dizisindeki ilk plaka resmini göster

resimdenkesital(anaresim,kare[0][0],kare[0][1],kare[0][3],kare[0][5]);

cv::rectangle(anaresim,cv::Point(kare[0][0],kare[0][1]-5),cv::Point(kare[0][3],kare[0][5]+5),cv::Scalar(0,0,255),1,cv::LINE\_AA,0);

cv::imshow("Ana Resim",anaresim);

Yukarıdaki işlemler neticesinde kare olarak atanan değerler gözönünde bulundurularak tespit edilen kordinatlar ile anaresimdeki plakanın çerçevesinin kırmızı renkte bir kare içerisinde gösterilmesi sağlanmış olur. Kırmızı çerçeve içerisine alınan kısım kesit alma fonksiyonu ile ayrı bir image dosyası olarak kayıt edilir. (Şekil-5).



Şekil-5 : Plaka Bölgesinin Ayrılması.

Anaresimden plakanın bulunduğu yer tespit edilerek ayrıldıktan sonra plaka olarak değerlendirilen alan içerisinde yer alan karakterin tanımlaması işlemine geçilir. Bu işlem için öncelikle plaka image içerisinde yer alan karakterlere ait değerlerin programa tanıtılması gerekmektedir. Projenin bu aşamasında ayrı bir program ile bu işlem gerçekleştirilir. Bu program içerisinde Türkiye Cumhuriyeti plaka sisteminde yer alan harf ve rakamlara ait fontların hem resimsel değerleri hemde ASCII kod karşılıkları xml formatında dosyalara aktarılmıştır. Bu işlem ile ilgili işlem adımları şunlardır;

Öncelikle içerisinde fontların bulunduğu image dosyası sisteme yüklenmiştir.

**İlgili Kod Bloğu:**

KarakterlerinResmi = cv::imread("Karakterset.png");

if (KarakterlerinResmi.empty()) {

std::cout << "Hata : Resim Ana Dosya

Yuklenmedi\n\n";

return(0);

}

Daha sonra program içerisinde belirtilen özelliklere ait countour lar tespit edilmiştir. Her countourun çevresi tespit edilerek kırmızı countourlara ait değerler bir çerçeve içerisine alınmıştır. Kullanıcıdan çerçeve içerisindeki karakteri klavye üzerinden tanımlaması istenmiştir. Eğer tespit edilen alan bir karakter içermiyorsa space tuşu ile bu tanımlamadan vazgeçilebilir.

**İlgili Kod Bloğu:**

cv::cvtColor(KarakterlerinResmi, KarakterlerinGriResmi, CV\_BGR2GRAY);

//Karakter Ana Resim Griye Cevrildi

cv::GaussianBlur(KarakterlerinGriResmi, KarakterlerinBulanikResmi, cv::Size(5, 5), 0);

//Karakter Ana Resim Blura Cevrildi

cv::adaptiveThreshold(KarakterlerinBulanikResmi, KarakterlerinSBResmi, 255, cv::ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv::THRESH\_BINARY\_INV, 11, 2); cv::imshow("KarakterlerinSBResmi", KarakterlerinSBResmi);

// SB (Tresh) resim göster

KarakterlerinSBResmiKopyasi = KarakterlerinSBResmi.clone();

//Islemler oncesi son resmin kopyasi alindi

cv::findContours(KarakterlerinSBResmiKopyasi, ptContours, v4iHierarchy, cv::RETR\_EXTERNAL, cv::CHAIN\_APPROX\_SIMPLE);

for (unsigned int i = 0; i < ptContours.size(); i++) {

if ((cv::contourArea(ptContours[i]) > Min\_Contour\_Alani) && (cv::contourArea (ptContours[i]) < Max\_Contour\_Alani))

{

//Bulunan counter minimum alandan buyukse isleme al cv::Rect Countercevresi = cv:: boundingRect (ptContours[i]);

//Counterların Cevresini Belirledik

cv::rectangle(KarakterlerinResmi, Countercevresi, cv::Scalar(0, 0, 255), 2);

//Cerceveyi kirmizi cizdik

cv::Mat matKarakter = KarakterlerinSBResmi(Countercevresi);

//Cerve icini aldik

cv::Mat matYeniBoyKarakter; cv::resize(matKarakter, matYeniBoyKarakter, cv::Size(Karakter\_Genisligi, Karakter\_Yuksekligi));

//Yeni bulunan karakterin gösterilmesi

cv::imshow("Karakter", matKarakter);

cv::imshow("YeniBoy Karakter", matYeniBoyKarakter);

cv::imshow("KarakterlerinResmi", KarakterlerinResmi);

int intChar = cv::waitKey(0)&255;

if (intChar == 27)

{

return(0);

}

else if (std::find(Karakterler.begin(), Karakterler. end(), intChar) != Karakterler.end())

{

//Yeni karakter resminin xml dosya icin hazirlanmasi

matKarakterNo.push\_back(intChar);

cv::Mat matKarakterResimFloat;

matYeniBoyKarakter.convertTo(matKarakterResimFloat, CV\_32FC1);

cv::Mat matKeskinKarakterResimFloat = matKarakterResimFloat.reshape(1, 1); matKarakterNoKarsilik.push\_back(matKeskinKarakterResimFloat);

}

Programın son aşamasında kullanıcıdan alınan geri dönüşümler ve ASCII kodu olarak bir xml dosyasına bu karaktere ait contour değerleride ayrı bir image dosyasına kayıt edilmiştir. Şekil-6’da işlemlere ait değerler yer almaktadır.

cv::FileStorage KarakterDegerleriDosyasi("classifications.xml", cv::FileStorage::WRITE);

if (KarakterDegerleriDosyasi.isOpened() == false)

{

std::cout << "Hata, Karakter Degerleri Dosyasi Acilamadi\n\n";

return(0);

}

KarakterDegerleriDosyasi << "classifications" << matKarakterNo; KarakterDegerleriDosyasi.release();

// Karakter Resim Degerleri Dosya İslemleri

cv::FileStorage KarakterResimDegerleriDosyasi("images.xml", cv::FileStorage::WRITE);

if (KarakterResimDegerleriDosyasi.isOpened() == false)

{

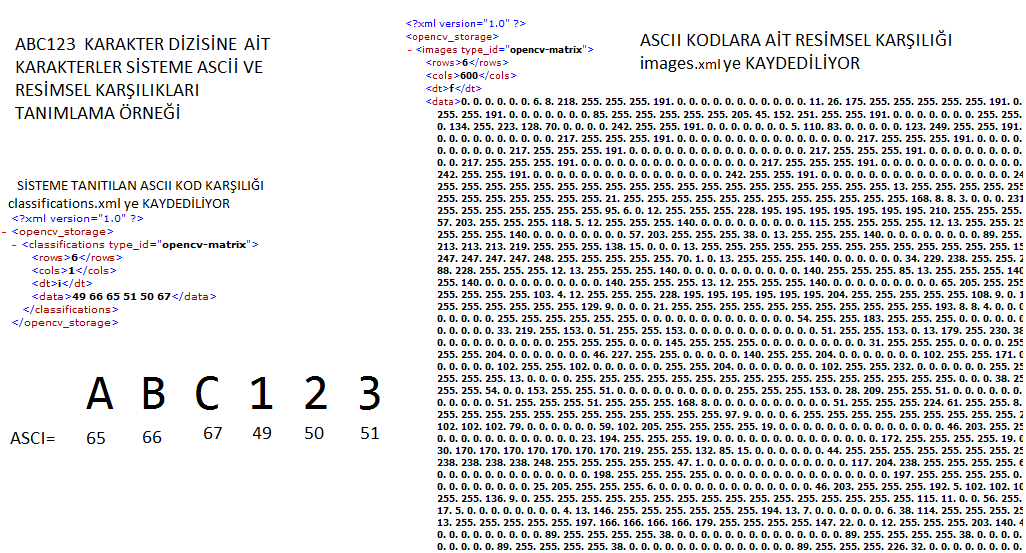
std::cout << "Hata, Karakter Degerleri Resim Dosyasi Acilamadi\n\n";

return(0);

}

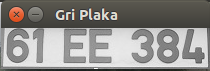
KarakterResimDegerleriDosyasi << "images" << matKarakterNoKarsilik;

KarakterResimDegerleriDosyasi.release();



(Şekil-6: Karakterlerin tanımlanması)

Sisteme verilen görüntü bilgisi, plaka çerçevesinin sınırladığı alan içerisindeki karakterleri içermektedir.



Şekil – 6 : Karakter tespiti için gönderilen görüntü

cv::Mat matTestPlakaResim = cv::imread("Plaka.png");

Bazı durumlarda karakter boyu ve genişliği plakadan plakaya farklılık arz edebilmektedir. Bunun için projenin belli kısımlarında alınan imageler üzerinde tekrar boyutlandırma işlemleri gerçekleştirilmiştir.

Yukarıda açıklanan işlem adımları ile anaresim üzerindeki plaka olarak tespit edilen alan ayrı bir imaj dosyası olarak kayıt edilmiş, bu imaj dosyasındaki karakterlerin tespiti ile ilgili olarakta karakter tanımlamaları tamamlanmıştır.

Projenin bundan sonraki adımında gerçekleştirilen işlemleri ise şu şekilde sıralayabiliriz;

İlk olarak karakter eğitme programında xml dosyalarına kayıt edilen değerler tekrar dosyalardan değişkenlere aktarılmıştır. Değişkenlerde yer alan karakter tanımlama bilgileri K-Nearest algoritmasında kullanılmak üzere [KNearest::train](http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html) komutu ile sisteme dahil edilmiştir.

**İlgili Kod Bloğu:**

cv::Mat matKarakterResimDegerleri;

cv::FileStorage KarakterResimDegerleriDosyasi("images.xml", cv::FileStorage::READ);

if (KarakterResimDegerleriDosyasi.isOpened() == false)

{

std::cout << "Hata, Karakter Resim Dosyasi acilmadi\n\n";

return(0);

}

KarakterResimDegerleriDosyasi["images"] >> matKarakterResimDegerleri;

KarakterResimDegerleriDosyasi.release();

// KNeasrest islemleri

cv::Ptr<cv::ml::KNearest> kNearest(cv::ml::KNearest::create());

kNearest->train(matKarakterResimDegerleri, cv::ml::ROW\_SAMPLE,matKarakterNoDegerleri);

Daha sonrasında karakter tanımlama işleminde olduğu gibi [findContours](http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours)[9] komutuyla plaka bölgesinde yer alan karakter veya karakter benzeri contourslar ayrılarak belirtilmiştir**.** Bulunan contoursların Max ve Min değerleri kontrol edilerek uygun görülenler ayrı bir değişkene aktarılmıştır. Geçerli contourslar için plakada yer alan karakterler çerçeve içine alınır ve bir takım dönüşümlere tabi tutulurlar. Sonrasında elde edilen countour değerleri [Knearst::findNearest](http://docs.opencv.org/3.0-beta/modules/ml/doc/k_nearest_neighbors.html)[8] methoduyla daha önceki tanımlanan karakterler değerleri ile karşılaştırılarak uygun karakter değerine ulaşılır.

**İlgili Kod Bloğu:**

std::vector<std::vector<cv::Point> > ptContours;

std::vector<cv::Vec4i> v4iHierarchy;

cv::findContours(matTestPlakaSBResimKopya, ptContours, v4iHierarchy, cv::RETR\_EXTERNAL, cv::CHAIN\_APPROX\_SIMPLE);

// Countor Tümünün Bulunmasi

for (unsigned int i = 0; i < ptContours.size(); i++)

{

Counterislemleri Counterislemleri1;

Counterislemleri1.ptContour = ptContours[i];

Counterislemleri1.Countercevresi = cv::boundingRect(Counterislemleri1.ptContour);

Counterislemleri1.Counteralani = cv::contourArea(Counterislemleri1.ptContour); ResimdekiTumCounter.push\_back(Counterislemleri1);

}

// Gecerli Counterlarin Ayrilmasi

for (unsigned int i = 0; i <

ResimdekiTumCounter.size(); i++)

{

if (ResimdekiTumCounter[i].CounterAlaniKontrol())

{

ResimdekiGecerliCounter.push\_back(ResimdekiTumCounter[i]);

}

}

std::sort(ResimdekiGecerliCounter.begin(), ResimdekiGecerliCounter.end(), Counterislemleri ::CounterSiralama);

std::string strBulunanPlaka;

for (unsigned int i = 0; i <ResimdekiGecerli Counter. size(); i++)

{

cv::rectangle(matTestPlakaResim, ResimdekiGecerli Counter[i].Countercevresi, cv::Scalar(0, 255, 0), 2);

//Counter Cerceve Belirle

cv::Mat matKarakter = matTestPlakaSBResim(ResimdekiGecerliCounter[i].Countercevresi);

//Matrise Ata

cv::Mat matYeniBoyKarakter;

cv::resize(matKarakter, matYeniBoyKarakter, cv::Size(Karakter\_Genisligi, Karakter\_Yuksekligi));

//Boyutunu Ayarla Gerekli Donusumleri Yap

cv::Mat matYeniBoyKarakterFloat; matYeniBoyKarakter.convertTo(matYeniBoyKarakterFloat, CV\_32FC1);

cv::Mat matYeniBoyKeskinKarakterFloat = matYeniBoyKarakterFloat.reshape(1, 1);

cv::Mat matBulunanKarakter(0, 0, CV\_32F);

//Resimdeki karakteri KNearest ile bul

kNearest>findNearest(matYeniBoyKeskinKarakterFloat, 1, matBulunanKarakter);

//Bulunan Degeri Stringe Cevir

float FloatBulunanKarakter = (float)matBulunanKarakter.at<float>(0, 0);

strBulunanPlaka = strBulunanPlaka + char(int(FloatBulunanKarakter));

}

std::cout << "\n\n" << "Bulunan Plaka Degeri = " << strBulunanPlaka << "\n\n";

cv::imshow("Bulunan Plaka Resim", matTestPlakaResim);

cv::waitKey(0);

# Bulgular

Geliştirilen sistemde plakadaki harf ve rakamları tanımlamak için test amacı ile kullanılan araç plakalarına ait karakter fontları tanımlanmıştır. Tanımlanmış karakterler dışındaki farklı font değerlerine sahip araç plakalarının da okunabilmesi için bu fontlarında sisteme tanıtılması gerekmektedir.

# Sonuçlar

Bu çalışmada Türkiye Cumhuriyeti plaka standartlarına uygun plakaya sahip bazı araç görüntüleri kullanılmıştır. Bu görüntüler çeşitli işlemlere tabi tutularak plakanın bölgesi tespit edilmiştir. Daha sonrasında elde edilen plaka görüntüsü üzerinde, bir takım görüntü işleme teknikleri ve K-Nearst algoritması kullanılarak, plakalarda yer alan karakterler otomatik olarak tanımlanmıştır. Bu işlemler sırasında sistem performansını olumsuz etkileyen bazı faktörler tespit edilmiştir. Bunlar; plakaların karayollarının belirlediği standartlara uygun olmaması, plaka alanında yabancı nesnelerin var olması vb. dir.

**Kaynakça**

[1]<http://www.bilgiustam.com/goruntu-isleme-ve-open-cv-nedir/>

[2] <http://www.bilisim-kulubu.com/sozluk/>

[3] INTEL CORPORATION: Intel researchers teach computers to ‘read lips’ to improve accuracy of speech recognition software. M2 Presswire, Coventry, Apr 28,2003, pg1.

[4] Bradski, G. and Kaehler, A., “Learning OpenCV: Computer Vision with the OpenCV Library”, O’Reilly Media, Amerika Birleşik Devletleri, 16-17 (2008).

[5] <https://www.qt.io/company/>

[6] <http://gamzecukurluoz.blogspot.com.tr/2010/03/qt-nedir.html>

[7]<http://www.batuhanduzgun.com/post/2010/08/24/QT-Nedir.aspx>

[8][http://docs.opencv.org/3.0beta/modules/ml/doc/ k\_nearest\_neighbors.html](http://docs.opencv.org/3.0beta/modules/ml/doc/%20k_nearest_neighbors.html)

[9][http://docs.opencv.org/2.4/modules/imgproc/doc/structural\_analysis\_and\_shape\_descriptors.html?highlight=findcontours#findcontourshttp://docs.opencv.org/2.4/modules/imgproc/doc/structural\_analysis\_and\_shape\_descriptors.html?highlight=findcontours#findcontours](http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours)