T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇ PLAKA TANIMA SİSTEMİNİN TASARIMI

YÜKSEK LİSANS TEZİ

GÜNAY MUSAYEVA

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

KASIM 2015

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇ PLAKA TANIMA SİSTEMİNİN TASARIMI

YÜKSEK LİSANS TEZİ

GÜNAY MUSAYEVA
Y1313.010015

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ahmad BABANLI

KASIM 2015

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1313.010015** numaralı öğrencisi **Günay MUSAYEVA** 'nın **"ARAÇ PLAKA TANIMA SİSTEMİNİN TASARIMI"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 30.06.2015 tarih ve 2015/13 sayılı kararıyla oluşturulan jüri tarafından oy.birliği ile Tezli Yüksek Lisans tezi olarak .kabul..edilmiştir.

Öğretim Üyesi Adı Soyadı	İmzası
Tez Savunma Tarihi :18/11/2015	
1)Tez Danışmanı: Prof. Dr. Ahmad BABANLI	.....
2) Jüri Üyesi : Yrd. Doç. Dr. Duygu ÇELİK ERTUĞRUL	.....
3) Jüri Üyesi : Yrd. Doç. Dr. M. Ahmed SHAH	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "ARAÇ PLAKA TANIMA SİSTEMİNİN TASARIMI" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (18/11/2015)

Günay MUSAYEVA /

ÖNSÖZ

Bu tez çalışmasında öncelikle örüntü tanıma alanları açıklanmıştır. Örüntü tanıma yöntemlerinin günümüzde en çok uygulandığı alanlardan biri olan plaka tanıma ayrıntılı olarak incelenmiştir. Tezin uygulama kısmında Azerbaycan Devlet plakalarına sahip görüntüler işlenmiştir. Bu görüntülere çeşitli algoritmalar uygulanarak farklı sonuçlar elde edilmiştir. Çalışmam boyunca değerli fikir ve önerileriyle beni yönlendiren, her konuda destek veren tez danışmanım Prof. Dr. Ahmad BABANLI' ya, eğitimim süresince emeği geçen tüm hocalarıma ve abime teşekkürü bir borç bilirim.

Haziran 2015 YL Öğrencisi Günay MUSAYEVA

İÇİNDEKİLER

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ARAÇ PLAKA TANIMA SİSTEMİNİN TASARIMI

ÖZET

Yapay zekâ mantığı üzerine kurulan otomatik sistemler artık her alanda kendini göstermektedir. Zamanla bu sistemler artarak insan emeğinin yerini dolduruyor. Sistemlere insan gibi düşünmeyi ve doğru işlemi yapmayı öğretilmesiyle beraber kullanıcı şirketler daha düşük maliyete, daha doğru sonuçlar elde etmiş oluyorlar.

Her alanda bulunduğu gibi trafikte de otomatik akıllı sistemler 1950 yılından beri mevcuttur ve hala gelişmektedir. Trafikte kullanılan sistem Araç Plaka Tanıma sistemi çekilmiş görüntüden plaka kısmını bularak ve bulunan alanda karakterleri seçip farklı yöntemlerle tanıma işlemini yürütüyor. Sonuç ise metinsel plaka çıktısı oluyor. Bulunan plaka ile şirket kendine yönelik işler uygular ve ya sisteme başka sistemler entegre ederek diğer işlemler de yapıla bilir.

Her ülkenin kendine özgü, eşsiz plaka düzeni vardır. Bu nedenle plaka tanıma sistemleri diğer otomotiv sistemlerden farklı olarak ülkeye göre değişmektedir. Plakaların değişik düzenleri nedeniyle farklı ülkelerin sistemlerinde farklı algoritmalar kullanılmaktadır. Çünkü plaka düzenine göre farklı algoritmalar farklı başarı yüzdesi göstermektedir.

Bu tezde plaka tanıma sistemleri çok fazla gelişmemiş Azerbaycan Devleti için plaka tanıma sistemi geliştirilmiştir. Bu sistemde hem hazır kütüphaneler, hem de daha önce Türkiye plakaları için geliştirilmiş olan algoritmalar Azerbaycan plakasına uygun olarak değiştirilerek, ilaveler edilerek kullanılmıştır. Birkaç farklı yöntemlerin kullanılması sistemin plaka tanımadaki başarısını artırmıştır.

Anahtar Kelimeler: *Plaka Tanıma Sistemi, Trafikte Gelişen Teknoloji, Karakter Tanıma*

CAR LICENCE PLATE RECOGNITION SYSTEM DESIGN

ABSTRACT

Automatic systems that are built on artificial intellegence logics presents themselves in all areas. Over the times, these systems are repalcin / fullfilling the manpower. Companies, by teaching the systems think like humans tries to obtain better results with low cost.

Like in all areas, automatic intellegence exists in trafic since 1950 and still in development. License Plate recognition systems by detecting the plate part of the pictured, runs the function to find the characters with differnt methods. The result is going to textual output plate. Companies can use these found numbers as referral to their missions or can integrate it to other systems.

As each county has its unique license plate design, designes are variable depending on the country. Thus, different countries use different algorithms. The reason is that, different algorithms show different success percentage on the basis of the designs.

İn this thesis, licence plate recognition system is developed for the developing county of Azerbaijan. Both existing libraries and revised algoritms that are previously developed for Turkish lincese plates are used in the system. Using several methods led the systems to the success in regards of license plate success.

Keywords: *License plate recognition system, developing technology on traffic, character recognition.*

1 GİRİŞ

Günümüzde hızla gelişen teknoloji tüm alanlarda kendisini göstermektedir. Durmadan düşünen insan beyinleri yeni fikirler ortaya atarak, yeni icatlar geliştirerek, herkesi hatta bu işin içerisinde olanları bile şaşırtmaya devam ediyor. Durmadan gelişen teknoloji sayesinde günlük yaşamımızda karşılaştığımız problemlerin aradan kaldırılması ve ya tekrarlanan işlemelerin olması durumunda işleri hızlandırmak ve daha fazla yanlış yapma olasılığı olan insan faktörünü en aza indirmek için otomatik sistemlere geçiş yapılıyor.

1.1 Çalışma Konusu

Günlük yaşamda duyulan gereksinimlerden bir tanesi de araçlardır. Araca olan ihtiyaç bununla beraber insanların yaşam kalitelerinin yükselmesi, otomobil reklamlarının artması doğal olarak araba sayısını da arttırmaktadır. Bununla beraber otomobil fabrikalarının rekabeti sayesinde de bir-birinden kaliteli arabalar satışa sunuluyor. Bu artış otomobil sektörlerinde araç akışının kontrolü gibi bazı sorunlara yol açıyor. Bu yüzden de akışın kontrolü ve yönetilmesi için akıllı ulaşım sistemlerine de ihtiyaç artmaktadır. Bu ihtiyaçlardan kaynaklanarak çalışma konusu olarak örüntü tanımaya dayalı olan Plaka Tanıma Sistemleri araştırılmıştır. Günümüzde trafik denetimi amaçlı, detektörleri ve radyo frekanslarını kullanan radarlar, mikrodalga detektörleri, yolun altına yerleştirilen tüpler de bulunmaktadır [1]. Bunların bazı dezavantajlarından dolayı trafik kontrolünü daha iyi ve güvenli bir şekilde sağlamak için bilgisayar tabanlı otomatik Plaka Tanıma Sistemlerine geçiş yapılıyor. Araç tespit işlemlerinde, Plaka Tanıma Sistemleri(PTS) insandan kaynaklanan hataları en aza indiren daha kullanışlı bilgisayar destekli otomatik sistemlerdir. Tez boyunca bu sistemlerden PTS olarak bahs edilecekdir. PTS kullanımı zamanı kameradan başka fazladan herhangi bir donanıma gerek kalmıyor. Her bir otomobil benzersiz plakaya sahip olduğu için, kamera aracılığıyla alınan otomobil görüntüsünden bazı algoritmalar kullanılarak plaka kısmı bulunuyor ve plaka üzerinde örüntü tanıma yöntemleri uygulanarak araç tespit ediliyor.

Anlatılan avantajlar nedeniyle de bitirme tezinin konusu olarak otomatik PTS seçilmiştir. Tezde Azerbaycan Devlet Plakaları için örnek bilgisayar tabanlı PTS kurulacak ve bu sisteme insanın görme ve algılama özellikleri kazandırılarak, plaka tespit edilerek karakterler tanınacaktır.

1.2 Tezin Amacı

Günümüzde otomobiller insanların en vazgeçilmez araçlarından biri olduğu için her yerde araçlarla iç-içeyiz. Birçok alanda olduğu gibi trafikte de aynı zamanda araçların bulunduğu her yerde güvenliğe ihtiyaç duyulmaktadır. Bu nedenle trafik akışının güvenli şekilde kontrolü, yönetimi ve farklı alanlarda araçların ve ya insanların güvenliği için çeşitli sistemlere gereksinim vardır. Yazılan bitirme tezinde görülen işlerin amacı da araçları algılayarak konumu, plakası, sahibi gibi farklı bilgileri belirlemek ve bu bilgileri kullanarak trafik akışının kontrolünü ve ya araç giriş çıkış izni olan kapılarda geçişleri kolaylaştırmaktır

2 ÖRÜNTÜ TANIMA

Örüntü düzenli bir şekilde bir-birilerini takip ederek yenileyen, tekrarlayan elemanlar kümesidir. Örnek gösterirsek bir ay içerisinde olan günler, gün içerisinde olan saatler ve b. Gelişen teknolojinin temeli olan yapay zekaya giden yol da örüntüden geçmektedir. Yapay zekada kullanılan örüntü kümelerine örnek gösterirsek: parmak izi, insan yüzü, araba plakası, elle yazılan kelime, retina ve b.

Örüntü tanıma ise bir-biriyle ilişkisi olan ve ya ortak benzerlikleri olan nesnelerin önceden belirlenen benzer özellikleri olan sınıflara atanmasıdır. Örüntü tanımanın en önemli amaçları: bilinmeyen örüntü kategorilerine belli bir şekil vermek ve bilinen bir kategoriye mahsus olan her hangi örüntüyü teşhis etmektir [2]. Örüntü tanıma farklı kaynaklarda farklı şekillerde tanımlanmaktadır:

- Fiziksel objelerin veya olayların önceden belirlenmiş bir veya daha fazla kategoriye ayrılmasıdır (Duda and Hart).
- Çok boyutlu uzayda yoğunluk fonksiyonunun tahmini veya bu uzayı kategori veya sınıflara ayırma problemidir (Fukunaga).
- Ölçülen verilerin tanımlanması veya sınıflandırılması(tanıma) ile uğraşan bilim (Schalkoff).
- X Gözlenen değerine ω ismini vermektir (Schürmann).
- Örüntü tanıma "Bu nedir" sorusuna verilen cevapla ilgilidir. (Morse).

Örüntü tanıma günümüz teknolojisinin temelini oluşturduğu için tanıma ile ilgili birçok alana uygulanmaktadır; Biyometrik tanıma, optik karakter tanıma, konuşma tanıma, el yazısı tanıma, insan ve makine tanılama, finansal tahmin, tıbbi tanı ve b.

2.1 Optik Karakter Tanıma(OKT)

Optik karakter tanıma(OKT) örüntü tanımanın uygulandığı alanlardan birisidir. İngilizcede Optical Character Recognition (OCR) olarak kaynaklarda söz ediliyor. OKT taranmış kâğıt üzerindeki evraklar, dijital kamerayla çekilmiş görüntüler gibi belgeleri arana bilen, değiştirile bilen pdf, word gibi bilgisayar destekli formata çeviren sistemdir. Örnek verirsek kitabın her hangi bir sayfasını tarayıcı ile tarattıktan

sonra OKT sistemiyle onu word dosyasına çevirme işlemidir. Bu teknoloji optik mekanizma yolu ile karakterlerin tanınmasını sağlar. İnsani varlık olduğumuz için bizim gözlerimiz de birer optik mekanizmadır. Gözlerin gördüğü görüntüler beyin için birer girdi sayılıyor. Bu girdileri anlama yeteneği ise birçok faktöre bağlı olarak her kişiye göre değişir. OKT insan okuma yeteneği gibi işleyen fonksiyonellere sahip birer teknolojidir. Ama her şeye rağmen OKT insanın okuma becerisiyle rekabet edemez.

OKT teknolojisi hem el yazısı hem de basılı metinleri tanıya bilir. Ama performansı girdi belgelerinin kalitesine doğrudan bağlıdır. Girdi olarak el yazısı görüntüsü verildiği zaman daha kısıtlanmış yani düzenli yazı olması sistemin performansını artıracaktır. Eğer girdi görüntüsü kısıtlanmış el yazısı olmazsa, teknolojinin insan okuma becerisine yetişmesi için daha uzun bir yoldan geçmesi gerektiğini görmek mümkündür. Ancak bilgisayarların hızlı okuma gücü ve teknik gelişmeler, bu teknolojini ideale yakın hale getiriyor.

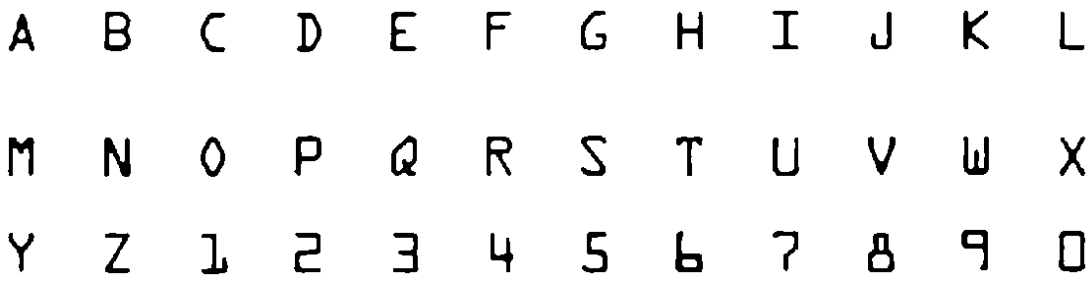
2.1.1 OKT tarihte ve günümüzde

Okuma gibi yapılan insan işlevlerinin makinalar tarafından hayata geçirileceği eskiden sadece birer hayaldi. Karakter tanımanın kökenleri aslında 1870lerden öncesinde bile buluna bilinir. Bu yılda Bostonlu C.R.Carey mozaik fotoselleri kullanan görüntü ileti sistemi olan retina tarayıcısını icat etti. İki yıl sonra Polonyalı P. Nipkow modern televizyon ve okuma makinelerinin her ikisi için büyük bir gelişme ola bilecek ardışık tarayıcını icat etmiştir.

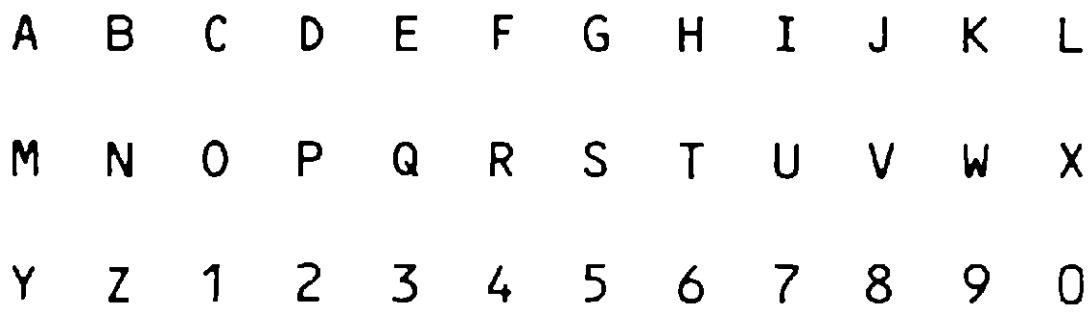
19.yüzyılın ilk 10 yılında farklı girişimler, kör deneylerin yardımı sayesinde OKT ile cihazların geliştirilmesini hayata geçirdi. Ancak, OKT'ın modern sürümü dijital bilgisayarın gelişmesi ile 1940'ın ortalarına kadar görünmedi. 1950 itibariyle teknolojik devrim yüksek bir hızda ilerlerken, elektronik bilgi işlem önemli bir alan haline geldi. Veri girişi delikli kartlarla gerçekleştiriliyordu ve kullanım miktarının artırmanın en uygun maliyetli yolunu bulmaya ihtiyaç vardı. Aynı zamanda okuma makineleri teknolojisi yeterince olgunlaşmıştı ve 1950.yılın ortalarına kadar OKT makinaları piyasada mevcut hale gelmişti. İlk gerçek OKT okuma makinesi 1954 yılında Reader's Digest (Amerikan dergisi) için kurulmuştur. Ekip bu makinanı delikli kartlara daktilo edilmiş satış raporlarını bilgisayara giriş için kullanmıştı.

1960-1965 yıllarında ortaya çıkan ticari amaçlı OKT sistemleri birinci nesil OKT sistemleri olarak adlandırılıyordu. Bu nesil makinaları özellikle sıkıntılı harf şekillerinin okunması için karakterize edilmişti. Kullanılan semboller özellikle makinanın okuması için tasarlanmıştı ve bu yüzden ilk makinalar son derece doğallıktan uzaktılar. Zamanla on değişik yazı tipi okuma becerisine sahip makinalar görünmeye başladı. Yazı tiplerinin sayısı, uygulanan örüntü tanıma yöntemi ile sınırlıydı. Kullanılan şablon eşleştirme yöntemi her karakter görüntüsü ile prototip(ilk) görüntüler kütüphanesinin her fontunun her karakteri ile karşılaştırıyordu.

İkinci nesil okuma makinaları 1960.yılının ortalarında ve 1970.yılının başlangıçlarında ortaya çıkmıştır. Bu sistemler makine baskılı yazıları tanımakla beraber el yazılarını da tanıma becerisine sahipti. El yazısı karakterleri düşünüldüğünde, karakterler dizisi rakamlarla, bazı harfler ve sembollerle sınırlandırıldı. Bu türün ilk ünlü sistemi olan IBM 1287, 1965.yılında New York'taki Dünya Fuarı'nda sergilenmiştir. Ayrıca bu dönemde Toshiba posta kodu numaraları için ilk otomatik harf sınıflandırma makinesini geliştirdi ve Hitachi de düşük maliyete yüksek performanslı ilk OKT makinesini yaptı. Bu gelişim döneminde en önemli iş OKT alanında standardizasyonun yapılması oldu. 1966.yılında OKT gereksinimleri çalışması eksiksiz olarak tamamlandı ve Amerika standart karakterler dizisini tanımladı; OCR-A. Bu yazı tipi son derece stilize edilmişti ve karakter tanımanı kolaylaştırmak için tasarlanmasına rağmen, insanlar tarafından da okuna bilir şekildeydi. Aynı zamanda Avrupa da kendi yazı standartlarını tasarladı (OCR-B) ve bu yazı Amerika yazı standartlarına göre daha doğaldı. Daha sonra bazı girişimciler iki yazı tipini bir standartta birleştirdi ve iki standarttı da okuma becerisine sahip daha güçlü makinalar ortaya çıktı.



Şekil 2.1 Amerika Karakter Standarttı OCR-A



Şekil 2.2 Avrupa Yazı Standarttı OCR-B

1970.yılının ortalarında ortaya çıkan, üçüncü nesil OKT sistemleri, düşük kaliteli belgelerin, büyük baskılı yazıların ve el yazmalarının tanınmasında eski nesil sistemlere meydan okudu. Teknolojinin düşük maliyetli ve yüksek performanslı olması donanım teknolojisinde etkileyici gelişmeler sağladı. Bu gelişmelere rağmen makine piyasasına sunulan çok basit OKT cihazları bile insanların işlerine yaramaya devam ediyordu. Kişisel bilgisayarların ve lazer yazıcılarının metin üretimi alanına hâkim olmaya başlaması öncesindeki zamanda yazı yazmanın OKT için özel bir yeri vardı. Aynı baskı aralıkları ve yazı tipinin küçük rakamları basit tasarlanmış OKT cihazını daha kullanışlı hale getiriyordu. Karalamalar sıradan daktilolarda oluşturuluyordu ve nihai düzenleme için bilgisayar üzerinden OKT aygıtlarına aktarılıyordu.

OKT makinaları 1950.yılında piyasada satışa sunulmaya başlasa da, dünya çapında 1986.yılına kadar sadece birkaç bin sistem satılmıştı. Bunun başlıca nedenlerinden birisi ürünün fiyatlarının çok yüksek olmasıydı. 1986 sonrasında donanımın giderek ucuzlamaya başlamasıyla ve OKT sistemlerinin yazılım paketleri şeklinde sunulmasıyla beraber satışlarda önemli ölçüde artım görünmeye başladı. 1993.yılında fiyatların düşmesi nedeniyle, artık birkaç bin sistem sadece bir hafta içerisinde satılıyordu.

2000'li yıllarda, OKT bulut bilişim ortamında ve smartfonda gerçek zamanlı yabancı dil çevirici uygulaması hizmetlerini sundu. Bu yıllarda çeşitli ticari amaçlı ve açık kaynak kodlu OKT sistemleri birçok yaygın yazım sistemleri olan Latin, Kiril, Arap, İbrani, Hint, Bengal, Devanagari, Tamil, Çin, Japon ve Kore karakterleri için de kullanılabilir olmaya başladı.

Günümüzde tanıma teknolojisi baskılı yazıların tanınmasında son derece yüksek seviyeye ulaşmıştır. OKT sistemleri baskılı yazıları tanımasında gösterdiği başarıyı,

el yazılarının tanımasında daha elde etmemiştir. El yazıları tanıma baskılı yazı tanımaya göre daha karışık işlemdir. Çünkü el yazısı kişiden kişiye değişiyor. Aynı zamanda yazma hızına, yazma şekline göre de farklılık göstermektedir. Buna rağmen günümüzde geliştirilmeye devam eden OKT sistemleri birçok alanda kullanılarak insan emeğinin yerini tutmaktadır. Aynı zamanda karakter tanıma teknolojilerinin sağladığı kolaylıklar nedeniyle birçok alanda bu teknolojilerin hızla uygulanmasına başlanmıştır: Mektupların üzerindeki posta kodlarının tanınmasında, trafikte plaka tanıma sistemlerinde, bankalara gönderilen çeklerin otomatik olarak okunmasında, çeşitli alanlarda kimlik numaralarının okunmasında ve b. Farklı alanlarda ve ya kişisel olarak kullanılmakta olan bazı OKT sistemleri mevcuttur:

- MODI (Microsoft Office Document Imaging)
- ABBYY FineReader
- Tesseract

2.1.2 OKT ile ilgili yapılmış çalışmalar

Fukushima tarafından neocognitron ağ yapısı kullanılarak tanıma işlemi gerçekleştirilmiştir. Piksel tabanlı bu tanıma sistemi hiyerarşik ve çok katmanlı yapıya sahipti. Özellikle harflerin ve rakamların tanınmasında kullanılmıştır [3]. Türkiye'de bu ağ yapısının kişisel bilgisayarlara uyarlaması konusunu Boğaziçi üniversitesi mezunu İrfan Oyman 1992.yılında bitirme projesi olarak işlemiştir [4].

Le Cun, rakam karakterlerin normalize edilmiş formlarını kullanarak, piksel tabanlı bir tanıma uygulaması gerçekleştirmiştir. Uygulamada dört katmanlı geri yayılım ağı tasarlanmıştır [5].

Melek Sarıcaoğlu el yazısı tanınması için yeni bir dilimleme algoritması geliştirmiştir. Bu çalışmada o insan beyninin yapısal ve fonksiyonel özelliklerinden yararlanarak geliştirilen, bir-birilerine ağırlık bağlantılarıyla bağlanmış sinir adı verilen basit hesap elemanlarından oluşan ve programlamak yerine eğitilme yönteminin esas alındığı yapay sinir ağları yardımıyla el yazısının dilimlenmesi ve karakterlerin tanınması işlemleri üzerinde durmuştur [6].

Juntanasub ve Sureerattanan tarafından 2005.yılında plaka tanıma sistemleri üzerinde çalışma yapılmıştır. Bu çalışmada görüntü ikili hale getirildikten sonra bölütleme

yapılmıştır. Karakterlerin tanınması kısmında ise Hausdorff uzaklığı yöntemi uygulanmıştır [7].

3 PLAKA TANIMA SİSTEMLERİ ARAŞTIRMASI

Plaka tanıma sistemi(PTS) farklı çeşit araç plakalarını insan faktörü olmadan okuyarak ve hızlı kimlik doğrulaması için yüklü bir veritabanıyla karşılaştırıp kontrol eden bir sistemdir. PTS plaka tespitinde kameranın aldığı görüntüyü kullandığı için görüntü tanıma teknolojisidir [8].

Çok rahatlıkla söylenir ki, günlük yaşamımızda otomobil kullanılan birçok alanda, trafik yönetiminde PTS kullanılmaktadır. Genellikle çeşitli otoparklarda, karayollarında, havaalanı gibi giriş çıkış noktaları olan alanlarda, hız kontrolü, araba hırsızlığı gibi olaylarda kullanılmaktadır.

3.1 PTS Avantajları

Genel olarak bakarsak iki avantaja sahiptir: İşlem hızı ve daha az hata yapması. PTS bir araç tespiti için sadece birkaç saniye harcıyor. Otobanda hızla giden bir otomobilin bile yavaşlamasına ihtiyaç kalmadan plaka görüntüsü alına biliyor [8]. Doğal olarak hız limiti olur ama bu limit çok yüksek oluyor ve sistemine göre değişiyor. Plaka tespit edildikten sonra birkaç saniye içerisinde bariyerin açılması, ikaz lambasının yanması, LED ışıkların yanması, siren çalması gibi gerekli işlemler uygulanıyor.

3.2 PTS Kullanım Alanları

Plaka Tanıma Sistemlerine ilave ek yazılım eklenerek, bu sistem çeşitli alanlara uygulanıyor. Bu nedenle otomobilin olduğu birçok sektörlerde PTS görmek mümkündür.

Plakadan elde edilen veriler genel olarak 4 amaç için kullanılıyor [9]:

*Hukuki uygulama***:** Hızlı ve ya tehlikeli araba kullanımı, trafik ihlalleri, çalıntı ve ya aramada olan arabaların tespiti için PTS kullanılıyor [9]. Sisteme yakalanan arabalar polise alarm ile veya başka yöntemle haber verile bilir. Sürücüye kesilen ceza sisteme otomatik olarak giriliyor. Ceza kontrolü internet üzerinden takip edile bilir.

Otomatik Geçiş Kapılarında: Ücretli manuel geçiş kapılarında sürücü arabanı durduruyor ve uygun ücreti ödedikten sonra bariyerin açılmasını bekleyip geçiyor [9]. Otomatik PTS sayesinde sürücü çok fazla beklemeden kapıdan geçe bilir [9]. Araç geçtikten sonra sistem belli bir sınıflandırma yaparak geçiş ücretini hesaplıyor ve PTS'nin kayda aldığı plakanın sahibi kişi tespit edilerek, ona aylık fatura gönderiliyor [9]. Bazı ücretsiz otopark, site, firma veya güvenliği önemli olan alanların giriş çıkış kapılarında da bu sistem kullanılıyor.

Örneğin siteye giriş zamanı sürücülerin kapıda güvenlikçinin kontrolünü ve manuel olarak bariyerin açılamasını beklemesi hoş bir durum değil. Aynı zamanda bazı site garajlarına giriş için araç sahibi site sakinlerinin her birine kumanda vermek ve ya personel çalıştırmak maliyet bakımından kullanışlı değil. Bu bakımdan PTS daha kullanışlı ve gelişmiş bir teknolojidir. PTS uygulanmış bir site giriş çıkışına arabası olan sakinler kapının önünde beklemeden geçe bilirler. Bunun için arabası olan sakinler araçlarının plakasını sisteme kayıt ettirmeleri lazımdır. Site sakini yeni araba aldığında plakayı veri tabanına kayıt ettiriyor, sattığında ise sistemden sildiriyor.

Diğer bir örnek ise firma giriş çıkış kapılarında kulllanılan Plaka Tanıma Sistemidir. Bu sistem sayesinde personellerin işe giriş çıkış saatleri belirlene bilir.

Otoparkında PTS kullanılan avm ve ya market gibi alanlarda giriş çıkış kapılarında araçlar sayıla biliniyor. Park eden ve otoparkı terk eden araçların sayı kontrol edilerek otoparkta kaç yer kaldığı ve ya dolu olduğu belirleniyor ve gerekirse sürücülere dolu olup olmaması gösteriliyor.

Tıkanıklık ve olay algılamada: Trafikteki otomobil kuyruğu, yavaş araçlar ve kazalar yaklaşan sürücüler için potansiyel tehlikedir [9]. Bu tür olaylar PTS ile tespit edile bilinirse, yaklaşan sürücüleri uyarmak amacıyla değişken mesajlı işaretler, hız limiti ayarlana bilir [9].

Yol kapasitesinin artırılması: PTS-den elde edilen veriler sayesinde yol kapasitesinin artırılmasına ihtiyacın olup, olmamasını da belirlemek mümkündür. Yol kapasitesinin artırılmasının en iyi yolu yeni yol inşa etmektir [9].

3.3 Kullanılan Plaka Tanıma Sistemleri

Bildiğimiz gibi dünyada eskiden beri PTS ile ilgili gelişmeler mevcuttur. Gelişmeler arttıkça şirketlerin bir-biriyle rekabetleri de artmaktadır. Hem Türkiye'de, hem de diğer ülkelerde kullanılan örnek PTS'lerden bazılarının çalışma prensiplerine bakalım.

"Otomatik-Müfettiş"- Dış koşullarında güvenilir şekilde çalışan, kolayca güvenlik ekipmanları ile entegre edile bilen, dış veritabanı olan ve harakette olan araçların numaralarını tanımayı sağlayan donanım ve yazılım kompleksidir. Kayıt problemlerinin çözümü, trafik kontrölü ve motorlu aracın güvenliği için etkilidir. Bu sistem gerekli fonksiyonelliği ile çeşitli alanlarda sorunları çözmek için kullanılan uygulamadır. Otopark içindeki araçların güvenliğini sağlamak, işletme genelinde araçların hareketinin kontrolü, tek bir magistral yolu ve ya bütün şehri kontrol etmek için kullanılmaktadır.

"Avtouraqan"- Devlet plakası gibi kayıt olmuş araç numaralarının tanınması için yazılım ve donanım kompleksi, görüntü tanıma uygulamasıdır. Bilgisayara gönderilen video görüntü esasında araç plakaları belirleniyor ve veritabanıyla kontrol ediliyor, aynı zamanda otomobilin o alandan geçtiğini kayd ediyor ve bunun gibi bazı işlemler yürütüle bilir. "Avtouraqan" plaka tanıma sistemi esnek ayarlarıyla saysız sorunları çözmeye izin veriyor. Bu sistem modüler yapıya sahip olduğu için bazı avantajları da var.

- Istemci-sunucu dağıtık ağ sistemi oluşturarak modüller farklı bilgisayarlarda ve ya lokal bir bilgisayarda çalışa bilinir.
- Gerekirse tanıma sistemine yeni fonksiyonlar eklene bilir.

"Platasis"- Türkiyede birçok şehirde birçok alanda kullanılmakta olan PTS'lerden biridir. Söz konusu sistem Türkiye'de geliştirilmiş ilk plaka tanıma ve okuma yazılımıdır. Otoyol, otopark, site gibi farklı alanlara uygulana bilmektedir. Tüm hızlardaki araçları tanıma, tüm veri tabanlarını desteklemesi, gece-gündüz okuma yapa bilmesi, raporlama sunması, POLNET (Emniyet Teşkilatı veri tabanı) uyumu sistemin temel özelliklerindendir.

"Hobi"- Türkiyede kullanılan diğer bir PTS markası da Hobi Bilişimdir. Sunduğu plaka tanıma sistemine başka taşıt tanıma sistemleri entegre ederek aracın plakasından başka rengini, hızını, yönünü, türünü, markasını da belirleme potansiyeline sahiptir. Plaka tanıma sırasında aranan araç tespit edildikte sesli ve görsel alarm üretilebilir.

3.4 PTS İle İlgili Yapılmış Çalışmalar

Optik karakter tanıma ve optik tarayıcılar ortaya çıktıktan sonra plaka tanıma üzerinde çalışmalar başlamıştır. İlk olarak Amerikada 1960.yıllarında bu yönde gelişmeler gerçekleştirilmiştir [10].

1990 yılında Newcastle üniversitesi harekette olan araçları tetikleme ünitesiyle algılayarak resmini çekip plaka kısmından karakterleri tanıya bilen sistem geliştirmiştir. Plaka kısmının bulunması etiketleme yöntemiyle hayata geçirilmiştir. Karakterlerin tanınması modülünde ise yapay sinir ağları kullanılmıştır [11].

1995 yılında Avustralyada CSIRO ve Telstra şirketlerinin beraber geliştirdikleri sistem kaliteli görüntülerden araçların plakalarını bulup merkeze göndermekteydi. Merkezdeyse araç sahiplerinin yolda kaç saat zaman harcadıkları hesaplanmaktaydı. Bu sistemde hem plaka kısmının bulunması, hem de karakterlerin tanınması aşamasında yapay zeka yöntemi kullanılmıştır [12].

1997 yıllarında Bristol üniversitesinden E. L. Dagless ve arkadaşları plaka tanıma sistemleri üzerinde çalışmalar yapmışlardı. Onlar plakadan karakterlerin ayrıştırılması adımında yatayda histogram yöntemini kullanmışlardı. Bu yöntemde histogram çizgilerine göre plakadan karakterler tespit edilmekteydi [13]. Aynı dönemlerde aynı üniversiteden C. John Setchel plaka yerini bulma adımında histogram yöntemini, karakterlerin tanınmasında ise yapay sinir ağlarını kullanmıştı [14].

Türkiye'de plaka tanıma sistemleri alanında 2003 yılında Mustafa Oral ve Umut Çelik tarafından çalışma yapılmıştır. Çeşitli görüntü işleme ve geriye yayılım algoritmaları kullanan yapay sinir ağı kullanılarak karakterlerin ayrıştırılması adımı hayata geçirilmiştir. Ayrıştırma işlemi için yatay ve düşey smearing algoritmaları kullanılmıştır [15].

2006 yılında Hacettepe üniversitesinde plaka yeri saptanmasında matlab uygulamaları kullanılarak %89,09, 2007yılında Süleyman Demirel üniversitesinde yapılan çalışmada ise %90 oranında başarı elde edilmiştir.

3.5 PTS Yöntemleri

PTS-ler genel olarak bakıldığında 3 aşamadan oluşuyor. Plaka Tanıma Sisteminin temel üç aşaması böyledir:

1. Kameradan elde edilen araç resminden plaka bölgesinin bulunması.

2. Plaka bölgesinden gereksiz şeyleri temizleyerek karakterleri ayırt etmek.

3. Ayırt edilen karakterleri karşılaştırma ile belirlemek ve plakanı tanımak.

Tüm aşamalar sonucunda plaka tanındıktan sonra artık yerine ve durumuna göre belirli işlem yapıla bilinir(sisteme kayda alma, kapıyı açma, alarm verme, hızını tespit etme ve benzeri). Temelini oluşturan bu 3 aşamadan önce ise görüntünün elde edilmesi ve görüntü üzerinde önişlemler uygulanıyor.

3.5.1 Görüntü ve görüntü üzerinde yapılan işlemler

Resim piksellerin, yani küçük noktaların birleşmesi sonuçu ortaya çıkan anlamlı dijital görüntüdür. F(x,y) 2 boyutlu uzayda pikseller bir araya gelerek görüntüyü oluşturur [16]. Genel olarak desek görüntü m x n boyutlu piksellerden ibaret bir matristir. Resimler 2 özelliği ile karakterize olunmaktadır.

1. Radyometrik özelliği: Pikselin algılandığı elektromanyetik spektrumdaki gri değeri [17].
2. Geometrik özelliği: Görüntü matrisinde sahip olduğu matris koordinatları [17].

PTS-de de bu özellikler üzerinde bazı önişlemler yapılarak ve çeşitli algoritmalara uygulanarak tanıma işlemleri hayata geçiriliyor. Bunlara morfolojik işlemler deniliyor ve resmin kalitesinin ayarlanması, boyutunun ayarlanması, renk dönüşümü gibi diğer birçok işlemler aittir.

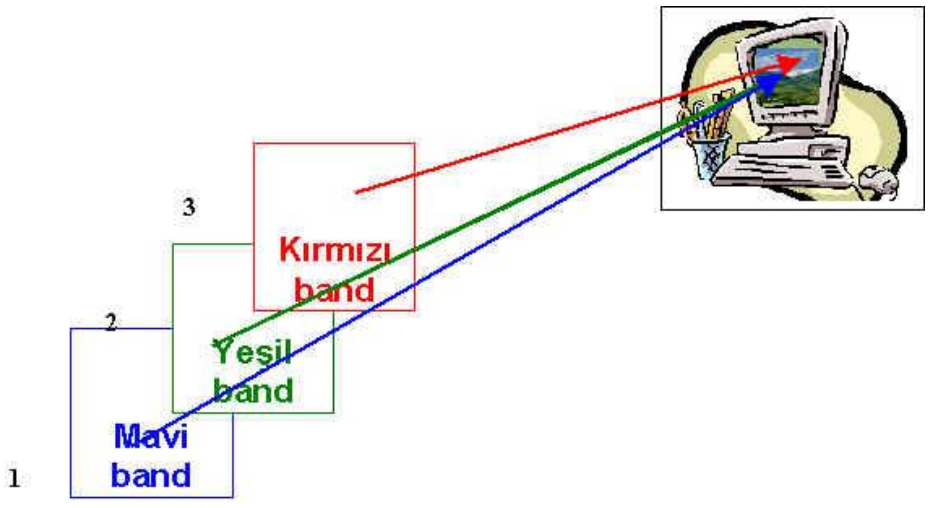
Resmin kalitesi santimetrekareye düşen piksellerin sayısı ile düz orantılıdır. Her bir santimetrekareye düşen piksellerin sayısı az oldukça doğal olarak pikseller daha büyük ölçüde oluyor ve piksellerin büyük oluşu resmin kalitesini azaltıyor. Çünkü resme

bakıldığında pikseller tek tek seçiliyor ve bir bütün resim oluşturmuyor. Pikseller küçük olduğunda ise onların kenar birleşme yerleri belli olmuyor, hepsi bir bütün olarak gözüküyor ve kaliteli görüntü oluşturuyor.

Resmin boyutu büyütüldükçe görüntüde detay yorumlama olanağı gittikçe yok olur. Bir görüntüyü büyütmeye başladıkça büyütme, görüntünün çözünürlüğünün izin verdiği düzeye gelesiye kadar bize objeleri daha ayrıntılı bir şekilde gözleme inceleme olanağı verir. Fakat bu çözünürlüğün üst değerlerine ulaştıkca yani büyütme sayısı bu cözünürlügün el verdiği ölçütlerden daha fazla bir değere yaklaştıkca bu durumda objeler iyice belirginsizleşir ve objeleri ayırt etmemiz olanaksız hale gelir. Bir büyütmede ulaşa bileceğimiz en son nokta tekli pikseldir. Bununla bereaber PTS-de lazım geldiğinde görüntüleri incelerken ve yorumlarken büyütme operasyonu kullanmaktayız. Aynı şekilde resim gittikce küçültüldüğünde de görüntüdeki ayrıntılar yok olacaktır ve bütüncül şekilde izleye bileceğiz.

Piksellerin renk değerlerinin farklılığı sayesinde değişik renk tonlarında görüntüler oluşuyor. Bu bakımdan PTS-de genel olarak 3 çeşit görüntü kullanılmaktadır. Renkli görüntü, gri seviyeli görüntü, ikili görüntü(siyah-beyaz). Diğer iki görüntü renkli görüntüden elde ediliyor.

Renkli görüntü üç ana rengin (RGB-kırmızı, yeşil, mavi) farklı oranlarda eklenmesiyle oluşuyor.



Şekil 3.1 Renkli Görüntü *[17]*

Renkli görüntü kavramı; 1band bir anlamda kırmızı filtrelenmiş, başka bir deyişle orijinal görüntüdeki gri değerler kırmızının tonları şeklinde ifade edilmiş, benzer

şekilde 2 ve 3 bandlar da yeşilin ve mavinin tonları şeklinde ifade edilip üst-üste çakıştırılmış ve oluşan renk karışımından da doğal renkler elde edilmiştir; şeklinde açıklana bilir [17]. R(kırmızı), G(yeşil), B(mavi) renkleri 0 ve 255 arası değerler alıyorlar. Renk koyuysa 0-a daha yakın bir değer alıyor, açık tonlu renkse 255-e yakın değer alacaktır. En koyu renk siyahtır onun değeri 0-dır, en açık tonlu renk ise beyazdır, onun da değeri 255dir. Fakat renkler [0,1] aralığına normalize edildiği [18] için 0-siyah, 1se beyaz rengi ifade etmektedir.

3.5.2 Resimden plaka bölgesinin ayırt edilmesi

Çekilmiş resimden plaka bölgesinin tespiti çok önemli bir aşamadır. Çünkü plaka yerinin saptanmasında yapılan hata plaka tanıma sisteminin genel performansını doğrudan etkileyecektir [19]. Araç plakalarının resim içerisindeki yerini bulmaya yönelik birçok yöntemler mevcuttur.

En çok kullanılan plaka yeri tespiti yöntemleri ayırt temelli ya da bölge temelli yaklaşımlara dayanmaktadır [20]. Bu algoritmaların bazıları ticari amaç güddüğü için şirketler tarafından gizli tutuluyor. Ama literatürde yer alan bazı plaka yeri tespiti için algoritmalar da mevcuttur.

Örneğin, kenar ayrıştırma, Hough dönüşümü, Top-Hat dönüşümü, simetri özelliği, morfolojik işlemler, renk özelliği, Histogram analizi, Gabor süzgeçleri v.s gibi sayısız teknikler önerilmiştir [21].

Hough dönüşümü- Dikdörtgen plakalar düşünülerek, giriş resmine Hough dönüşümü uygulanarak, plakanın sınır çizgilerini bulmaya çalışan çalışmalar mevcuttur [22]. Özellikle çizgilerin bulunması için uygulanan Hough dönüşümü yüksek işlem yükü ve bellek ihtiyacı gerektirdiğinden gerçek zamanlı uygulamalarda kullanılmamaktadır [20].

Top-Hat dönüşümü- Top-hat dönüşümü plakanın ana karakteristiği olan beyaz arka plan üzerindeki siyah pikselleri veya siyah arka plan üzerinde beyaz pikselleri ayrıştırmada çok etkilidir [21]. Bu dönüşüm önişlem yapılarak gri seviyeye getirilmiş resimlere uygulanıyor. Top-Hat dönüşümü açılma işlemi uygulanmış resim ile orijinal resmin farkı alınarak bulunuyor [21].

Gabor Süzgeçleri- Gabor süzgeçler doku analizinde kullanılan en önemli araçlardan bir tanesidir [23]. Bu sistem bulunması istenilen farklı desenlere göre ayarlana bilir. Gabor çekirdeğinin denklemi aşağıdaki gibidir**(3.1):**

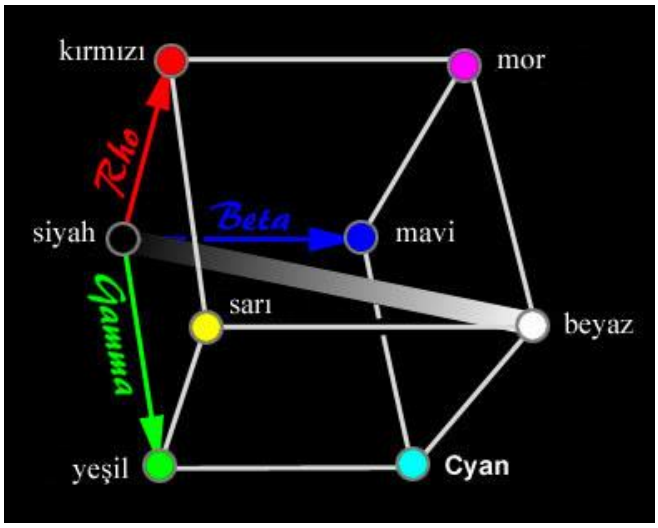
$$g(x, y; \lambda, \theta, \sigma) = \exp(-\frac{x'^2 + y'^2}{2\sigma^2}).\cos(2.\pi.\frac{x'}{\lambda})$$
$$x' = x.\cos\theta + y.\sin\theta \qquad y' = -x.\sin\theta + y.\cos\theta \qquad \textbf{(3.1)}$$

Bu denklemlerde g-gabor çekirdeği, θ-aranan desenin açısı, λ-aranan desenin dalga boyunu göstermektedir. Ϭ değeri büyük olduğunda sadece λ ve bu değere çok yakın dalga boylarına sahip desenler bulunurken, Ϭ değeri küçük olduğunda daha geniş spektrumda dalga boyu içeren desenler bulunur [20].

Histogram analizi- Histogram eşitleme işlemi yapılması için resmin gri seviyeye getirilmesi gerekir. Gri seviyeli resmin renk dağılımını göstermek için kullanılan yöntemdir. Histogram: bir veri setindeki verilerin dağılımını veya başka bir deyişle kullanım sıklığını ya da frekansını gösteren bir tablodur diyebiliriz [24]. Tezde yazılan PTS örneğinde plaka tespit kısmı bu yöntem kullanılarak yapıldığı için histogram eşitleme daha detaylı şekilde aşağıda anlatılacaktır.

3.5.2.1 Renkli resmin gri resme dönüştürülmesi

Bu aşamada yapılan işlem morfolojik işlemlerdendir. Kameradan çekilen resim orijinal halde RGB değerlerine sahip oluyor. Fakat renkli resimde temizlenmesi gereken çok sayıda gereksiz ayrıntıların ve gürültünün olması tanıma işlemini zorlaştırıyor. Bu nedenle resmi gri seviyeli, daha basit resme çevirmek gerekiyor. Yukarda yazıldığı gibi RGB değerleri 0-255 arası değer alıyor ve 0-değeri siyah rengi, 255-değeri beyaz rengi karakterize etmektedir. Siyah ve beyaz renklerinin düz bir çizginin başlangıç ve sonunda olduğunu varsayarsak bu çizgi üzerinde bulunan değerler grinin tonlarının değerleri olacaktır. RGB değerlerine üç boyutlu koordinat sisteminde bakarsak bunu göre biliriz.



Şekil 3.2 RGB Değerlerinin 3 Boyutlu Koordinatlarda Gösterilmesi *[18]*

RGB küpündeki (Şekil 3.2) beyaz-siyah köşegenindeki tüm değerler gri seviyedeki rengi kodlandırır ve bu renk RGB renk kanallarının hepsinin ortak bir değer ile doldurulmasından elde edilir [18]. Örneğin (127, 127, 127) üçlüsü grinin tonu iken (127, 18, 63) beyaz -siyah köşegeninde bulunmadığı için gri seviye değildir [18].

Gri seviyeli resmin her pikselinde R=G=B olduğu için, gri seviyeye indirgemenin en kolay yöntemi renkli resmin R,G,B değerlerinin ortalamasının bulunmasıdır. Bu tezde de resmi gri seviyeye indirgemek için aynı teknik kullanılmıştır. Bu yöntemle RGB resminin gri resme dönüştürülmesi formülü **(3.2)** ve örnek görüntü:

$$Y = (R\ (\text{Kırmızı}) + G\ (\text{Yeşil}) + B\ (\text{Mavi}))/3 \qquad \textbf{(3.2)}$$

Şekil 3.3 Ortalamasının Bulunması Metoduyla Griye İndirgenmiş Görüntü

2007 yılında yapılan deneyler sonucunda insan gözünün renklere duyarlılığı incelenerek; beyaz ışığa olan göz duyarlılığı 1(bir) olarak ve gözün en iyi görebildiği renkler sırasıyla yeşil, kırmızı ve mavi olarak algılanmıştır [16]. Bu yöntemle gri seviyeye indirgeme eşitliği ve elde edilen sonuç **(3.3)** ve şekil 3.4 de gösterilmiştir:

$$Y=0.299R+0.587G+0.114B \; ; \; R=Y \; G=Y \; B=Y \qquad \textbf{(3.3)}$$

Şekil 3.4 Göz Duyarlılığına Göre Bulunmuş Yöntemle Gri Seviyeye İndirgenmiş Görüntü

Eşitlik 3.3 her bir piksel için uygulanır ve ortaya çıkan Y değeri, gri ölçekli resmin yeni RGB değerleridir [18].

3.5.2.2 Histogram eşitleme

Histogram eşitleme gri seviyeli resmin renk dağılımının normalize edilmesidir. Resimler farklı zamanlarda farklı mekanlarda çekildiği için güneş ışığından, hava durumundan, resmi çekme açısından kaynaklanan parlaklık, kontrast gibi sorunlarla sık-sık karşılaşa bilinir. Bu gibi sorunların çözülmesinde kullanılan, gri seviyeli resme uygulanan bir işlemdir. Histogram eşitleme matematiksel olarak böyle gösterilmektedir **(3.4).**

$$p(i) = \frac{n_i}{n} \; ; \; T(r) = round(255 \sum_{i=0}^{r} p(i)) \; ; 0 \leq r \leq 255 \qquad \textbf{(3.4)}$$

i- parlaklık değeri

n_i- i-ninci parlaklık değerinin görüntüdeki sayısı

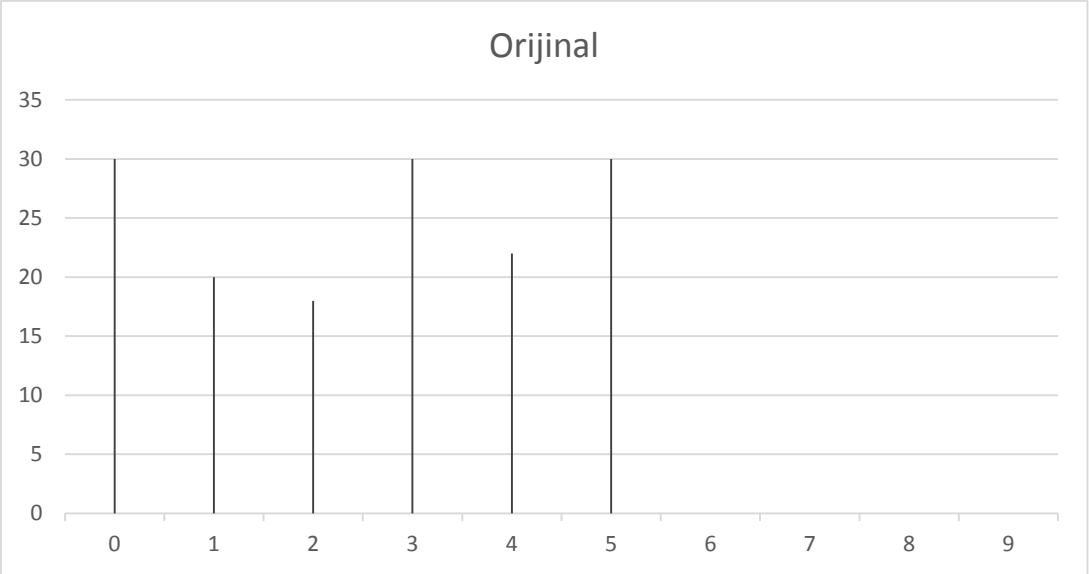
n- görüntüde kullanılan tüm parlaklık değerlerinin toplam sayısı

Histogram Eşitleme örneğine bakalım:

Çizelge 3.1 Renk Dağılım Tablosu

İ	0	1	2	3	4	5	6	7	8	9
n_i	30	20	18	30	22	30	0	0	0	0

Çizelge 3.2 Renk Dağılım Grafiği



p(0)=30/150=0.2 p(1)=20/150=0.13 p(2)=18/150=0.12

p(3)=30/150=0.2 p(4)=22/150=0.14 p(5)=30/150=0.2

p(k)=0/150=0; k=6,7,8,9

T(0)=round 9*p(0)= round 1.8=2

T(1)=round 9*(p(0)+p(1))= round 2.97=3

T(2)=round 9*(p(0)+p(1)+p(2))= round 4.05=4

T(3)=round 9*(p(0)+p(1)+p(2)+p(3))= round 5.85=6

T(4)=round 9*(p(0)+p(1)+p(2)+p(3)+p(4))= round 7.11=7

T(5)=round 9*(p(0)+p(1)+p(2)+p(3)+p(4)+p(5))= round 8.91=9

T(6)=round 9*(p(0)+p(1)+p(2)+p(3)+p(4)+p(5)+p(6))= round 8.91=9

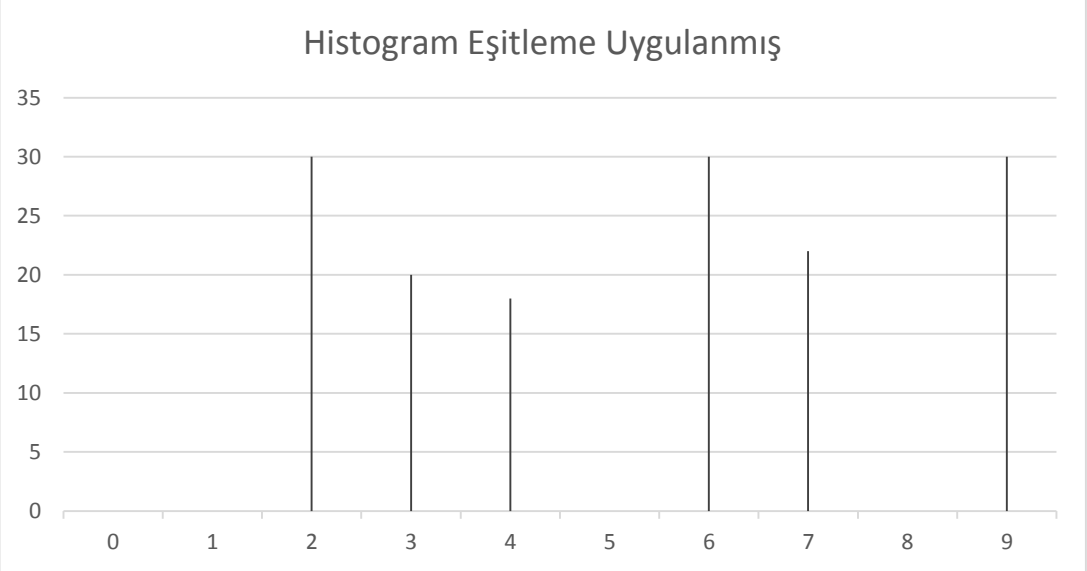
T(6)=T(7)=T(8)=T(9)=9

Histogram eşitleme uygulandıktan sonra görüntünün bir yere kümelenmiş renk parlaklık değerleri paylanılıyor ve bu aşağıdaki tablo ve grafikte göre bilinir:

Çizelge 3.3 Histogram Eşitleme Uygulanmış Renk Dağılım Tablosu

i	0	1	2	3	4	5	6	7	8	9
n_i	0	0	30	20	18	0	30	22	0	30

Çizelge 3.4 Histogram Eşitleme Uygulanmış Renk Dağılım Grafiği



3.5.2.3 Gri resmin siyah-beyaz resme dönüştürülmesi

Gri seviyeli görüntünün pikselleri daha önce anlatıldığı gibi siyah beyaz çizgisi üzerinde yer alan değerleri almaktadır. Siyah-beyaz(ikili) görüntünün pikselleri ise sadece iki değer ala bilir: 255 ve 0. Genel olarak ise ikili resim olduğu için 1(beyaz) ve 0(siyah) rakamlarıyla karakterize ediliyor. Görüntünü ikili resme dönüştürülmesinin nedeni ise plaka kısmının arka fonunun beyaz olmasıdır. Yani plaka kısmının bulunmasında diğer renkler gerekli değildir.

Dönüştürülme işlemi gri seviyeye indirgenmiş görüntüye uygulanıyor. Bu işlem bazen literatürlerde eşiklenme işlemi olarak da geçiyor. Farklı yöntemleri mevcuttur.

Örneğin yerli(local) eşikleme, bütünsel(global) eşikleme, en iyi yaklaşımla(optimal) eşikleme, Otsu algoritmasıyla eşikleme [25]. Tezde kullanılan yöntemlerden bir tanesi Otsu yöntemidir, diğeri ise sabit bir eşik değeri belirlenerek uygulanan tekniktir. En kolay yolu da sabit bir eşik değer belirleyerek yapmaktır. Bu eşik değerden yüksek olan pikseller beyaza, düşük olan pikseller siyaha çevriliyor. Sabit eşik değer belirleyerek ikili resme çevirme formülü **(3.5)** denkleminde gösterilmiştir [26].

$$\mathbf{I}_{bin}(\mathbf{p}) = \begin{cases} 1 & \text{if } \mathbf{I}_{grey}(\mathbf{p}) \geq d \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(3.5)}$$

$I_{bin}(p)$ ikili görüntüsünü elde ettiğimiz bu formülde *d* herhangi eşik değerdir.

Otsu algoritmasını Nobuyuki Otsu 1979 senesinde icat etmiştir. Otsu algoritmasında eşik değeri görüntüye bağlı olaral değişmektedir. Her bir görüntünün aslında 2 kısımdan ön(foreground) ve arka fondan(background) oluştuğu varsayılıyor [27]. Daha sonra tüm eşik değerleri için bu iki renk sınıfının(ön, arka fon)varyans değerleri **(3.9)** ve sınıf içi varyans değerleri **(3.6)** hesaplanıyor [27]. Varyans değerlerinin hesaplanmasında **(3.7)** ve **(3.8)** denklemleri kullanılmaktadır. Bu değerin en küçük olmasını sağlayan eşik değeri, optimum eşik değeridir [27].

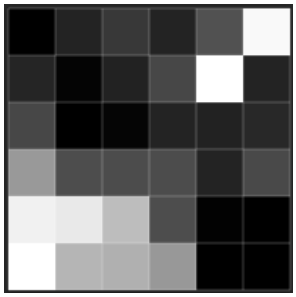
$$\sigma_w^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \textbf{(3.6)}$$

$$W_b(t) = \sum_{i=1}^{t} P(i) \quad ; \quad W_f(t) = \sum_{i=t+1}^{I} P(i) \quad \textbf{(3.7)}$$

$$\mu_b(t) = \sum_{i=1}^{t} \frac{i*p(i)}{W_b} \quad ; \quad \mu_f(t) = \sum_{i=t+1}^{I} \frac{i*p(i)}{W_f} \quad \textbf{(3.8)}$$

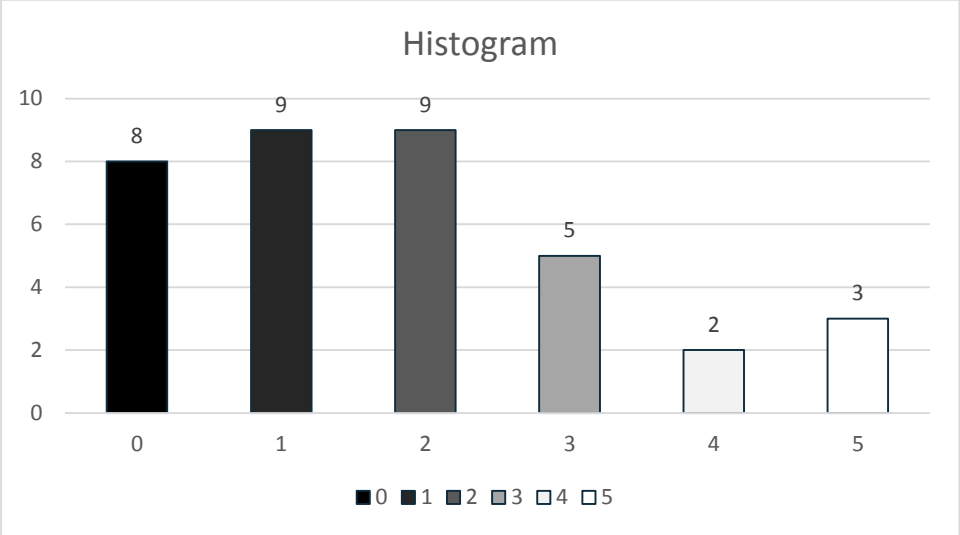
$$\sigma_b^2(t) = \sum_{i=1}^{t} [i - \mu_b(t)]^2 \frac{i*p(i)}{W_b} \quad ; \quad \sigma_f^2(t) = \sum_{i=t+1}^{I} [i - \mu_f(t)]^2 \frac{i*p(i)}{W_f} \quad \textbf{(3.9)}$$

Çok küçük (6x6 matrislik) gri seviyeli görüntüde bu formülleri kullanarak sınıf içi varyanslarını hesaplayıp, görüntüye uygun olarak eşik değerini bulup, resmi eşikleyelim.



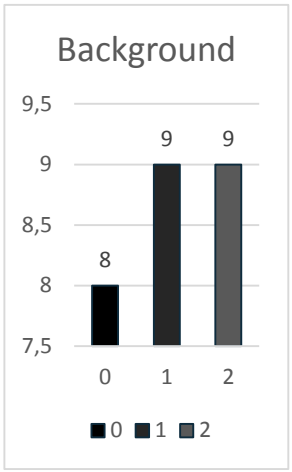
Şekil 3.5 Gri Seviyeli Görüntü

Çizelge 3.5 Gri Seviyeli Görüntünün Histogramı



Hesaplamalar sonucunda eşik değerinin T=3 olduğu anlaşılmıştır. Bu yüzden tüm hesaplamaları kontrol etmeden örnek için sadece T=3 haline bakalım.

Çizelge 3.6 Gri Seviyeli Görüntünün Arka Plan Histogramı

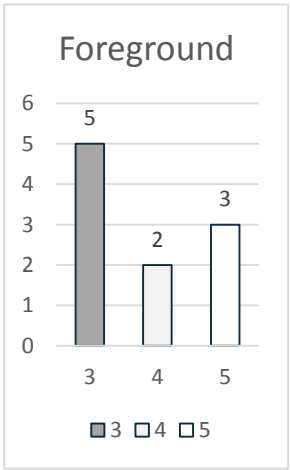


Weight $W_b = \frac{8+9+9}{36} = 0{,}7222$

Mean $\mu_b = \frac{0*8+1*9+2*9}{26} = 1{,}0384$

Variance $\sigma_b^2 = \frac{((0-1{,}0384)^2*8)+((1-1{,}0384)^2*9)+((2-1{,}0384)^2*8)}{26} = 0{,}6523$

Çizelge 3.7 Gri Seviyeli Görüntünün Ön Plan Histogramı



Weight: $W_f = \frac{5+2+3}{36} = 0{,}2777$

Mean: $\mu_f = \frac{3*5+4*2+5*3}{10} = 3{,}8$

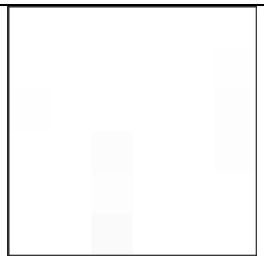
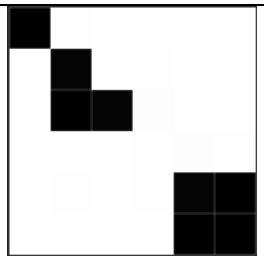
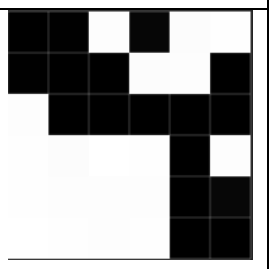
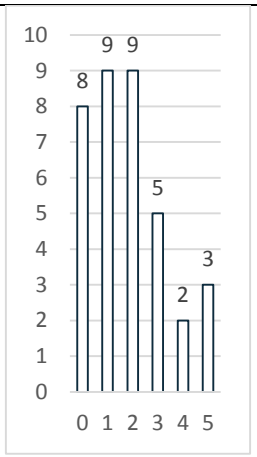
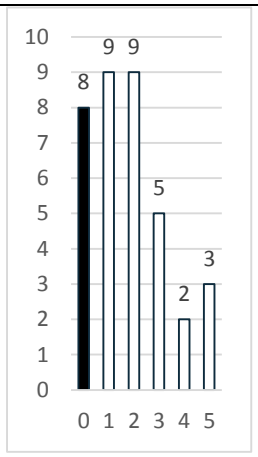
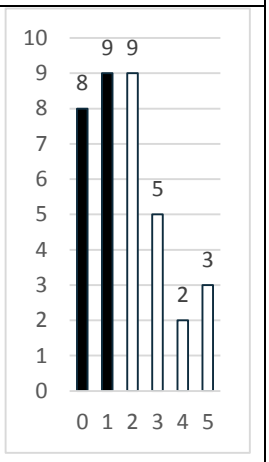
Variance: $\sigma_f^2 = \frac{((3-3,8)^2*5)+((4-3,8)^2*2)+((5-3,8)^2*3)}{10}=0,76$

Sonuncu adım ise sınıf içi varyansın hesaplanmasıdır.

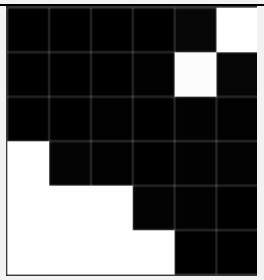
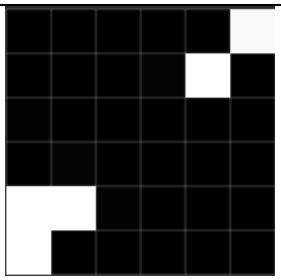
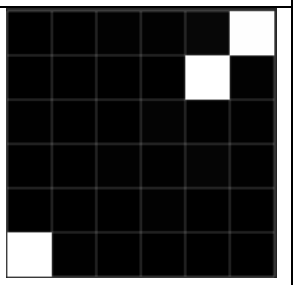
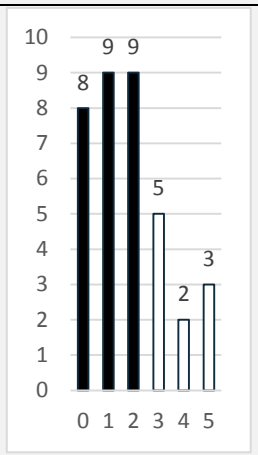
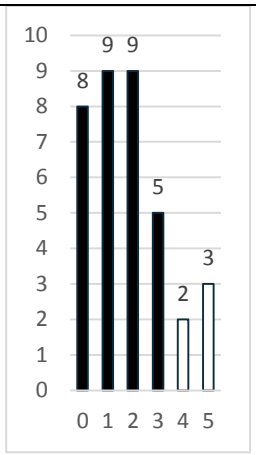
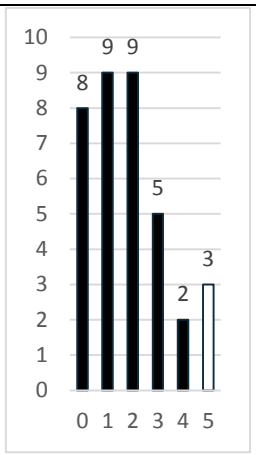
$\sigma_w^2=0,7222*0,6523+0,2777*0,76=0,6821$

Çizelge 3.8 ve Çizelge 3.9 da tüm eşik değerlerle hesaplamaların sonucuna bakalım:

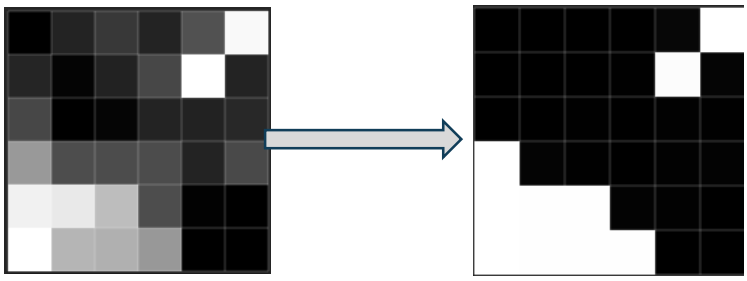
Çizelge 3.8 T=0,1,2 Eşik Değerleriyle Görüntünün Eşiklenmesi

Threshold	T=0	T=1	T=2
Image			
Histogram			
Weight	W_b=0	W_b=0,2222	W_b=0,4722
Main	μ_b=0	μ_b=0	μ_b=0,5294
Variance	σ_b^2=0	σ_b^2=0	σ_b^2=0,2491
Weight	W_f=1	W_f=0,7777	W_f=0,5277
Main	μ_f=1,8055	μ_f=2,3214	μ_f=2,9473
Variance	σ_f^2=2,2121	σ_f^2=1,6466	σ_f^2=1,2077
Within Class Variance	σ_w^2=2,2121	σ_w^2=1,2805	σ_f^2=0,7549

Çizelge 3.9 T=3,4,5 Eşik Değerleriyle Görüntünün Eşiklenmesi

Eşikleme	T=3	T=4	T=5
Görüntü			
Histogram			
Weight	W_b=0,7222	W_b=0,8611	W_b=0,9166
Main	μ_b=1,0384	μ_b=1,3548	μ_b=1,5151
Variance	σ_b^2=0,6523	σ_b^2=1,0613	σ_b^2=1,4012
Weight	W_f=0,2777	W_f=0,1388	W_f=0,0833
Main	μ_f=3,8	μ_f=4,6	μ_f=5
Variance	σ_f^2=0,76	σ_f^2=0,9471	σ_f^2=0
Within Class Variance	σ_w^2=0,6821	σ_w^2=0,9471	σ_w^2=1,2843

Bu tablodan da görüldüğü gibi T=3 eşik değerinde sınıflar içi varyans en küçük değeri alıyor.



Şekil 3.6 Gri Seviyeli Görüntü Ve T=3 ile Eşiklenmiş Görüntü

3.5.2.4 Filtrelemeler

Yumuşatma, keskinleştirme, belirginleştirme, kenar tespiti, yatay/dikey sınırların tespiti yaygın olarak kullanılan filtreleme tekniklerindendir. PTS lerde ise genelde yumuşatma(smooth) filtreleleri kullanılıyor. Buna pürüssüzleştirme de diye biliriz. Görüntüdeki gürültünü azaltmak için kullanılıyor. Söz konusu bu tezde de yumuşatma yöntemlerinden kullanılmıştır. Bir çok pürüssüzleştirme yöntemleri var. Bunlardan ikisine dikkat edilecek.

1.Smooth Gaussian

2.Smooth Median

Filtreleme zamanı filtre olarak çekirdek matris(kernel) kullanılıyor. Matrislerin içeriği ve boyutları filterleme tekniğine göre değişiyor. Resmi pürüssüzleştirmek için kullanılan filtre matrislerinin boyutu birden büyük ve tek olmalıdır [28]. Matrisin boyutu büyüdükce görüntünün bulanıklığı ve işlem süresi daha da artıyor [28].

Görüntüye piksellerin değerlerinden oluşan bir matris gibi bakarsak, bu değerlerle çekirdek matrisinin değerleri üst-üste gelecek şekilde yerleştire ve kaydıra biliriz. Matrisin değerleri ile görüntünün piksellerinin üst-üste düşen değerleri tek-tek çarpılarak toplanıp, çekirdeğin değerlerinin toplamına bölünüyor [27]. Daha sonra elde edilen değer görüntünün eşleştirilmiş merkez pixelinin yerine yazılıyor [27]. Bu işlem görüntü boyunca tekrarlanır. Aşağıda düşük filtreleme için bulanıklaştırma örneği gösterilmiştir.

Çizelge 3.10 Orijinal Görüntü

12	123	60	110	44	23
34	7	44	11	150	77
10	66	53	28	66	99
180	157	9	1	12	61
99	100	201	159	67	83
10	46	87	91	168	190

45

Çizelge 3.11 Çekirdek Matris

1	1	1
1	1	1
1	1	1

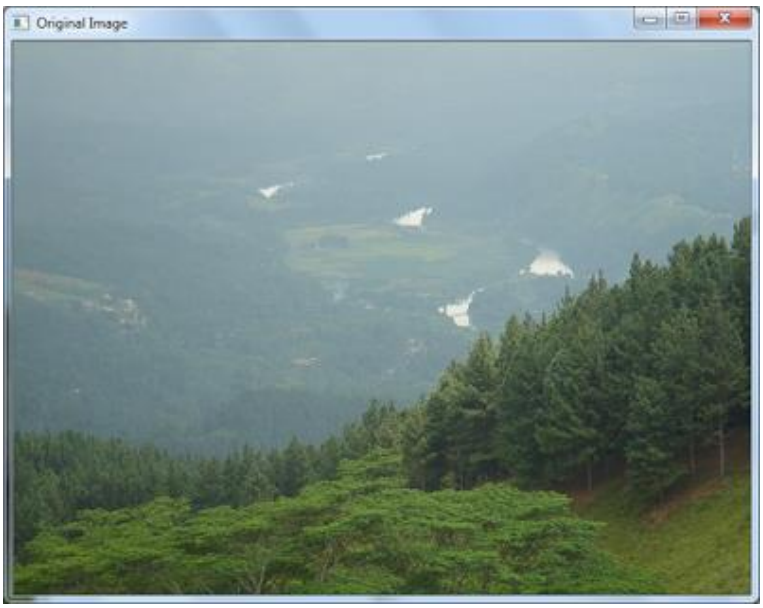
12*1+123*1+60*1+34*1+7*1+44*1+10*1+66*1+53*1=409

Çekirdek matrisin değerlerinin toplamı n olursa n=9dur.

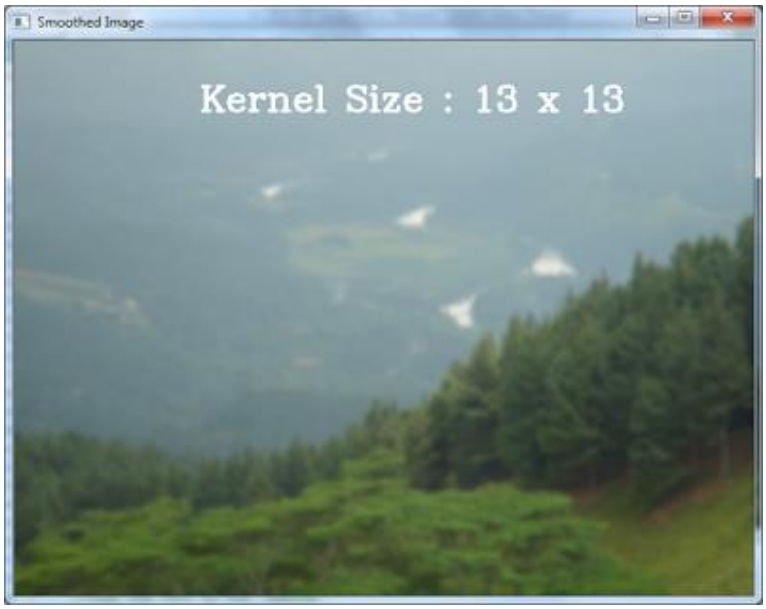
409/n=409/9≈ 45

Görüntüde merkezde bulunan 7 değeri yeni bulunan 45 değeri ile değiştiriliyor. Fakat bu değişim en son ekleniyor. Çekirdek tüm görüntü boyu gezdirildikten sonra eski değerlerle işlem yapıldıktan sonra yeni değerler görüntüde yerlerini alıyor [27]. Yani kısacası eski değerlerle yeni değerler aynı tabloda yer alamaz [27].

Smooth Gaussian- Adını renk değerlerinin değişimini gaussian çanı denilen eğriyle eşleştirmesinden alır [29]. Görüntüdeki gürültüyü aradan kaldırmak için kullanılıyor [28]. Çekirdeği 13x13 olan filtreden geçirilmiş örnek bir resme bakalım:



Şekil 3.7 Orijinal Resim



Şekil 3.8 Smooth Gaussian Uygulanmış Resim

Smooth Median- Girişe verilen resmi Medyan çekirdek ile filtreleniyor. Medyan çekirdek tek ve birden büyük olmalıdır. Aynı zamanda kare matris olması lazımdır. Çekirdeğin elemanlarını matrisde sırası büyükten küçüğe doğru ve ortanca elemanın (2,2) indisinde olması gerekiyor [30]. Bu teknik geniş ölçüde kenar algılama algoritmaları kullanarak, belirli şartlar altında gürültünü yok ederken kenarları koruyor [28]. Genelde "salt-pepper" bozunma türünü yani ikili gürültüleri aradan kaldırmak için kullanılıyor [30]. Çekirdeği 17x17 olan filtreden geçirilmiş örnek bir resme bakalım:

Şekil 3.9 Smooth Median Uygulanmış

3.5.2.5 TopHat dönüşümü

Plaka yeri saptaması yöntemlerinden birisi de Top-hat dönüşümüdür. Bu teknik siyah beyaz kısımların ayrıştırmasında etkilidir. Gri seviyeye indirgenmiş resimler üzerinde kullanılmaktadır. Tepe **(3.10)** veya çukur **(3.11)** bölgeleri belirginleştirme özelliğine sahiptir [21].

Aydınlık bölgeler için [31], $\text{TopHat}\,[A,B] = A - (A \circ B) = A - \max(\min(A))$ **(3.10)**

Karanlık bölgeler için [31], $\text{TopHat}\,[A,B] = (A \bullet B) - A = \min(\max(A)) - A$ **(3.11)**

3.5.2.6 Gabor filtresi

Gabor filtresi görüntü analizinde kullanılan yöntemlerden birisidir. Bu yöntem parmak izi tanıma, iris tanıma, yüz tanıma, plaka tanıma gibi sistemlerde kullanılmaktadır.

3.5.2.7 Genişletme ve aşınma işlemleri

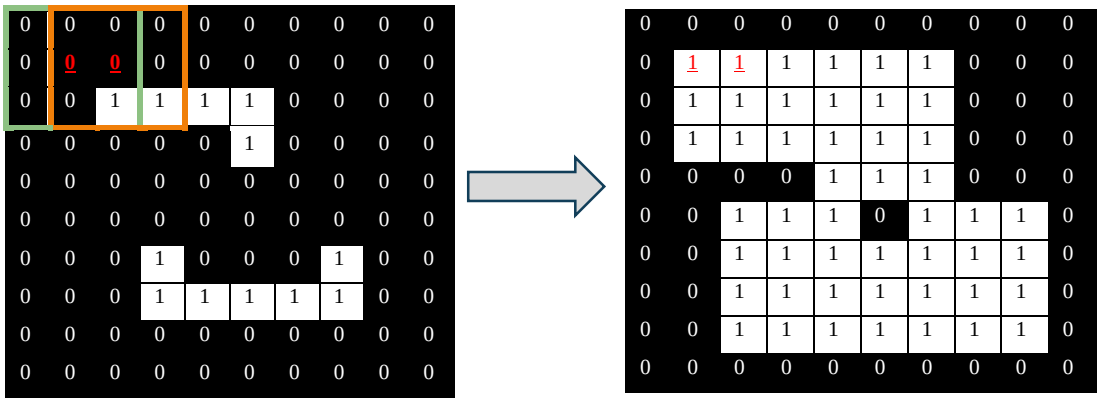
Genişletme ve aşınma işlemleri en önemli morfolojik işlemlerdendir. Diğer morfolojik işlemler bunlarında üzerinden uygulanıyor. Genişletme algoritması ile aşınma algoritması bir-birinin tersi olan işlemleri yapıyor. Örneğin genişletme işlemi görüntüde gürültü ile ayrılmış iki kısımı birleştirirken, aşınma işlemi birleşmiş bir nesneni bir-birinden farklı iki nesne haline dönüştürüyor.

Genişletme işlemini görüntüyü uygulamak için ilk önce 3x3 boyutlu matris yapı elemanı belirleniyor. Bu matris yapı elemanı farklı rakamlardan oluşturula bilir. Şimdi bu matrisin 1-lerden oluşuturulduğunu ve görüntünün de 0-1 den ibaret matris olduğunu farz edelim. Resim üzerinde bu yapı elemanı sırayla kaydırılacak. Tek bir pikselde bile 1-ler üst-üste geldirse yapı elemanının merkezinde olan görüntünün piksel değeri 0 ise onun yerine 1 yazılacak. Kaydırarak değer değişimi yaparken dikkatli olunması gereken bir nokta var. Değiştirelen yeni değerler eski değerlerle işleme girmemeli. Yani tüm görüntü boyunca kaydırılma işlemi bittikten sonra yeni değerlerin görüntüde yerlerini alması lazımdır.

Çizelge 3.12 3x3 Boyutlu Yapı Elemanı Matrisi

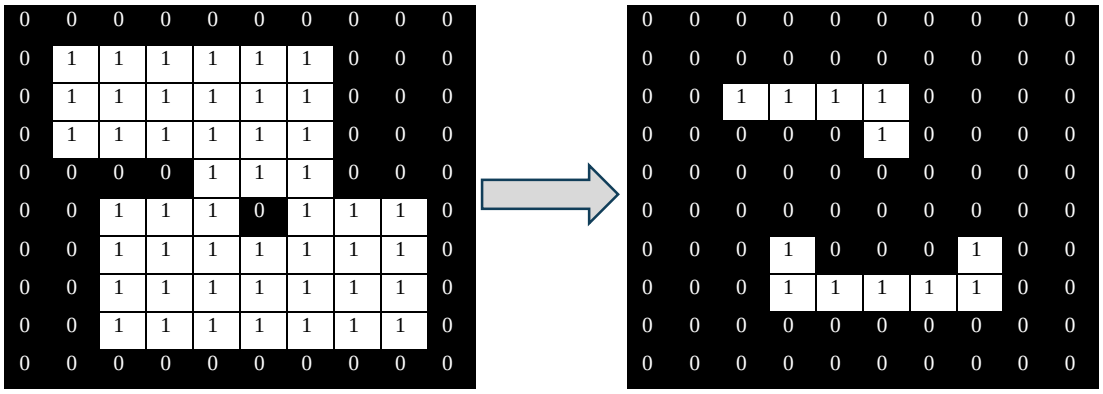
1	1	1
1	1	1
1	1	1

Çizelge 3.13 : Orijinal Görüntü Ve Genişletme Algoritması Uygulanmış Görüntü



Aşınma algoritması da genişletme gibidir. Tek fark 1ile 1in üst-üste gelmesi değil 1ile 0-ın üst-üste gelmesi sonucunda merkezdeki 1 değerinin yerine 0 yazılıyor. Bu teknikte de yine eski değerlerle yeni değerler aynı görüntüde olamaz. İşlem bittikten sonra yeni değerler yerlerini alacaktır. İşlemin amacı ise görüntüde birleşmiş nesneleri bir-biriden ayırmaktır. Yukarıda genişletme algoritması uygulanmış görüntü parçasına aynı 3x3 boutlu 1lik matrisle aşınma algoritmasını uygulayalım:

Çizelge 3.14 Genişletme Ve Aşınma Algoritması Uygulanmış Görüntü



3.5.2.8 Bağlantılı bileşen etiketleme

Bağlantılı bileşen etiketleme görüntü işleme tekniklerindendir. Siyah-beyaz resimlere uygulanıyor [27]. Bu işlemin amacı görüntüde bileşenlerin komşuluk ilişkilerini bulmaktır [27]. Görüntü üzerinde sırayla pikselleri tarayarak uygulanıyor. Tarama işlemi satırlar veya sütünlar üzere yapıla bilinir [27]. Tarama zamanı rastlanan her bir beyaz piksele gerekli etiket(rakam) atanıyor. İşlemin algoritması böyledir:

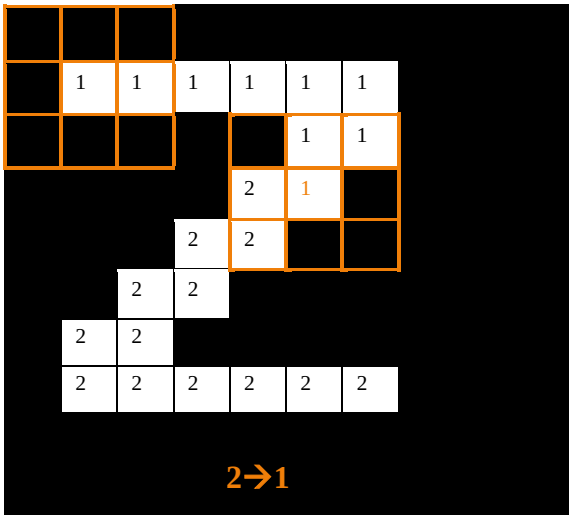
1.Tarama yönü(sütün, satır) belirleniyor ve görüntü üzerinde tarama başlatılıyor [27].

2.Eğer bileşenin her hangi etiketli komşusu yoksa bu durumda yeni etiket atanıyor piksele. Eğer bileşenin 8(ya da 4) komşusundan en az bir tanesi etikete sahipdirse ozaman piksele komşusunun etiketi atanıyor. Eğer bileşenin komşularında farklı etiketler vardırsa, bu durumda bileşene etiketi küçük olanın etiketi atanıyor ve bu bilgi kayd ediliyor.

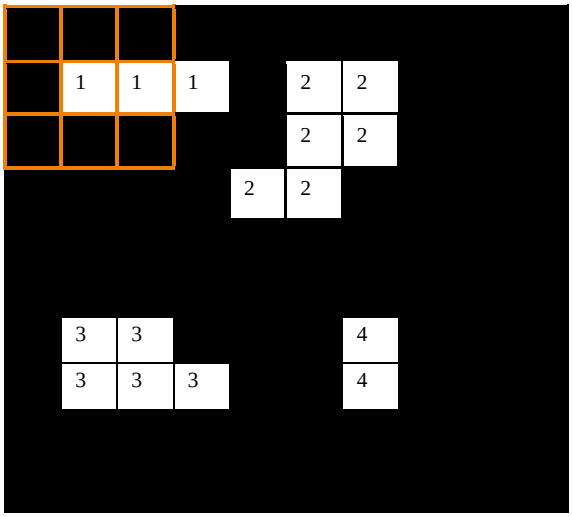
3.Tarama zamanı komşuda farklı etiket durumu olmuşsa eğer bu zaman büyük etiketin değeri küçükle değiştiriliyor. Böylece her bir bileşene bir etiket atanmış oluyor [27].

Bu işlemi tek bileşene sahib ve bir kaç bileşene sahip görüntüler üzerinde uygulayarak deneyelim:

Çizelge 3.15 Tek Bileşenli Z Harfi Görüntüsü. Sütunlar Üzere Tarama



Çizelge 3.16 Çok Bileşenli Görüntü. Satırlar Üzere Tarama



3.5.3 Plaka bölgesinden karakterlerin seçilmesi

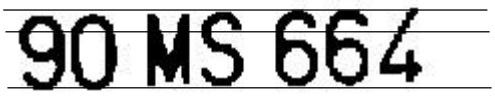
Yapay zekâ yöntemiyle yapılan karakter tanıma işlermi için karakterlerin ayrıştırılması önemli işlemlerdendir. Farklı yöntemlerle bu işlemi hayata geçirmek mümüküdür.

3.5.3.1 Karakterlerin sınırlarının belirlenmesi yöntemi

Karakterlerin çerçevelenmesi yani seçilmesi için ilk önce verilen metnin satırlara ayrılması gerekmektedir [32]. Plaka bölgesinde zaten tek satır olduğu ve karakterler bir birinden ayrık, eğimsiz olduğu için bu durumda ayrıştırma işlemi daha kolaylaşmıştır. Satır başlangıcının belirlenmesi için görüntünün dikey başlangıc noktasından başlamak üzere soldan sağa her bir yatay piksel satırı taranır [32]. İlk siyah piksele rastlanılan piksel satırı plakanın başlangıcı kabul edilir [32]. Bu aşamadan sonra gelen ilk piksel içermeyen piksel satırı, metin satrının kapsadığı yatay piksel satrını, yani metin satrının yüksekliğini belirler [32]. Plakalarda türkçe karakterler (Ö, Ğ, Ü, İ.) kullanılmadığı için onları seçmek için ayrı bir işlem yapmaya gerek de kalmıyor.

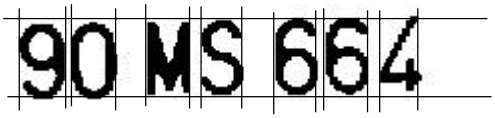
Başlangıç piksel satrı

Bitiş piksel satrı



Şekil 3.10 Karakterlerin Yatay Sınırlarının Belirlenmesi

Karakterlerin çerçevelenmesinde ise satırın başlangıç ve son pikselleri arasında yatay başlangıç noktasından(x=0) başlanarak dikey doğrultuda tarama yapılır [32]. Satır belirlemeye benzer şekilde rastlanan ilk siyah piksel içeren sütun karakterin sol başlangıcı kabul edilir [32]. Bu sütundan sonra ilk siyah piksel içermeyen sütün ise karakterin sonu kabul edilir [32].

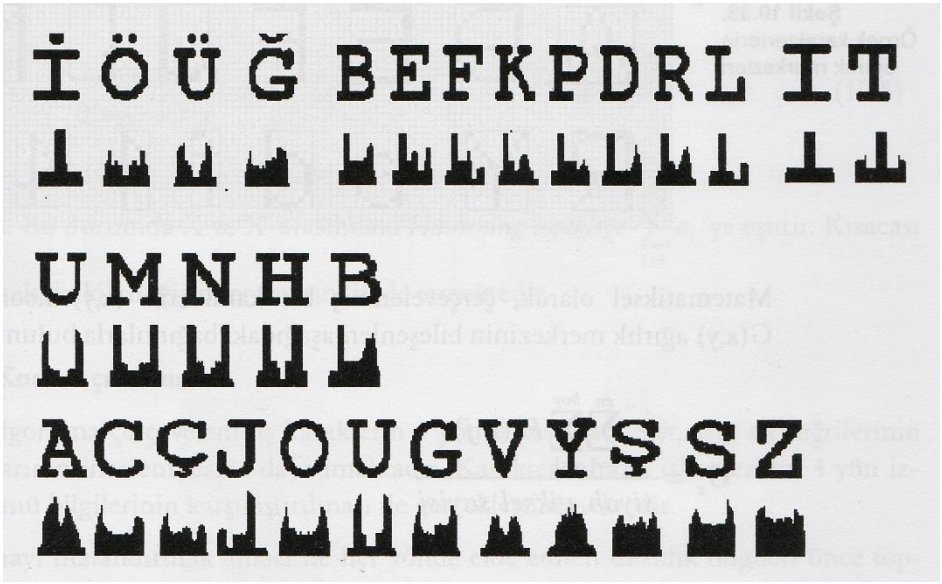


Şekil 3.11 Karakterlerin Dikey Sınırlarının Belirlenmesi

Böylece karakter sol ve sağ kenarından hiç boşluğu olmayan bir çerçevenin içine alınarak karakterin eni belirlenmiş olur [32]. Bu yöntem genelde iyi temizlenmiş, beyaz arka fon üzerine siyahla yazılmış plakalarda hızlı çalışıyor ve iyi sonuç veriyor [33]. Yani bu metodun başarılı olması için önceki adımların sonucunda kaliteli plaka görüntüse elde etmemiz gerekiyor [33].

3.5.3.2 Dikey histogram (izdüşümü) yöntemi

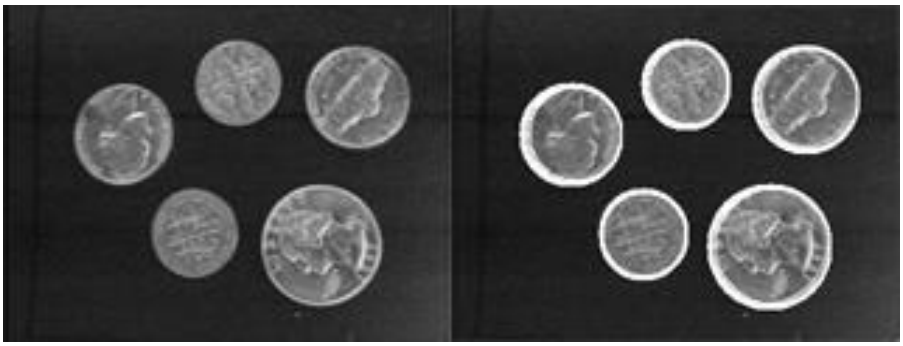
Birbirine değmeyen karakterlerin analizi için çok kullanılan basit metodlardan birisi de dikey histogram yöntemidir. Bu yöntem uygulanırken resmin kesinlikle ikili görüntüde olması ve gürültülerden iyi temizlenmesi gereiyor. Yani daha önce anlatılan önişlemlerin uygulanması zorunludur. Bazı kaynaklarda ikili plaka görüntüsü araba resminden yeterli hassasiyetle lokalize edilmiş olsa bile görüntünün sınırlarını daraltmak gerektiği belirtilmiştir [34]. Bu işlem gereksiz dikey çizgilerin (plaka anahat) histogramını alma durumunu önlemek için uygulanıyor [34]. Daha sonra karakterlerin ayrıştırılması için plaka görüntüsünün dikey histogramı(izdüşümü) alınmıştır(şekil 3.12). Şekil 3.12 da görüldüğü gibi izdüşüm konturu bu karakterler arasındakı pozitif değeri ve aralıklar arasındakı çok düşük değerleri göstermektedir [18]. Yani histogramdakı yerel bir minimum nokta, karakterler arası sınırları belirler [32]. Minimum noktalar arasındakı mesafe de karakterin genişliği olarak kabul edilir [18].



Şekil 3.12 Dikey Histogram

3.5.3.3 Hough transformu

Hough dönüşümü 1962 yılında Paul Hough tarafından geliştirilmiştir. Genelde siyah beyaz görüntüler üzerine uygulanıyor. Hough dönüşümü düzgün doğruların tespiti için geliştirilmiştir. Ama bu yöntem geliştirilerek bunun yanı sıra günümüzde doğrusal olmayan eğrilerin tespiti için de kullanılmaktadır. İris bulma, plaka bulma ve benzeri alanlara uygulanmaktadır. Bu alanlarda görüntüdeki şekillerin tespitinde kullanılıyor. Bazen şekillerde kopukluk olduğu zaman tespit etme zorlaşıyor [35]. Görüntünün tamamının görülebilir olmadığı durumlarda da olası şekiller tespit edile bilmektedir [35]. Aşağıdaki örnekte solda orijinal görüntü, sağda hough dönüşümü ile çemberleri belirlenen şekiller verilmiştir.



Şekil 3.13 Hough Dönüşümü Uygulanmış Resim [22]

3.5.3.4 Genişletme

Karakterler çerçevelendikten ve ya ayrıştırıldıktan sonra her bir karaktere genişletme işlemi uygulanıyor. Genişletme işleminin amacı karakterde olan kopukluğu, boşlukları aradan kaldırmaktır. Doğal olarak bu zaman daha iyi sonuçlar elde edilir. Genişletme işlemi zamanı çerçevelenmiş karakterler üzerinde farklı maskeler kullanılıyor [32]. Plaka kısmının bulunmasında genişletme ve aşınma işleminde olduğu gibi resimin üzerinde satırlar üzre gezdidirilerek karaktere uygulanıyor. Maskeler farklı amaçlarla kullanıla bilir.

- Eğer toplam değer 0 ise elde edilen sonuç kenar algılama ile ilgilidir [32]
- Eğer toplam sonuç değer 1 ise, sonuç orjinal değerlerindedir [32].
- Eğer toplam değer 1den büyük ise burada amaç resmin iyileştirilmesidir [32].

Karakter ayrıştırma için programlarda kullanılan diğer bir yöntem hazır kütüphanelerin kullanılmasıdır. Tüm bu işlemleri kendisinde toplayan hazır karakter ayrıştırma kütüphaneleri mevcutturki bu kütüphaneleri programda çağırmak yeterlidir. Örnek hazır kütüphanelere bakalım: GOCR, PUMA,

3.5.4 Optik karakter tanıma

Üzeri yazı dolu kâğıtlar ve ya plakaya sahip araba resimleri bilgisayarlar için sadece bir görüntü rolü oynuyor. İnsanlar için ise bunlar belli şekle sahip karakter olarak algılanıyor ve tanınıyor. Makinelerin de insan gibi bu görüntüleri karakter olarak algılaması için bunların tek-tek öğretilmesi gerekiyor. Bilgisayar ortamında kullanmak istediğimiz bunun gibi dokümanları bazı işlemlerden geçirmek gerekir. Bu işlemler dizisini gerçekleştiren sistemlere optik karakter tanıma(Optical Character Recognation-OCR) sistemleri deniliyor. Karakter tanıma bir metnin içindeki karakterlerin taranmış sayısal görüntüsünü bunlara karşılık düşen sembolik ifadeye dönüştürme işlemidir [32]. Çoğu zaman bu sembolik ifade, karakterin ASCII kod karşılığıdır [32].

Modern OCR teknolojisi 1951 yılında M. Sheppard ın icadı olan GISMO-Okur, Yazar Robot ile birlikte doğmuştur. 1954 yılında sistem çok gelişmesine rağmen dakikada bir karakter tanıma becerisine sahipti ve sadece büyük sektörler tarafından kullanılan fantastik değerlere alınan bir sistemdi. Günümüzde ise çok fazla gelişerek uygun fiyatlara satılan, hata oranı çok düşük olan ve daha hızlı tanıma becerisine sahip bir sistem halini almıştır. En başarılı sistemlerin hata oranı 2% ye kadar düşmüştür [32],

tanıma becerisi ise saniyede 12 karaktere kadar artmıştır [32]. Uygun fiyat nedeniyle Optik Karakter Tanıma günümüzde en çok kullanılan sistemlerden birisidir. İster PTS'ler, isterse de herhangi bir görüntünün metin şekline dönüştürülmesi için çeşitli alanlarda, farklı yöntemlerle kullanılıyor: havaalanlarında, otoparklarda, sınır geçişlerinde, site girişlerinde ve b.

OCR temelinde Yapay Zekâ mantığı duruyor. Farklı alanlara uygulandığı için farklı tabanlı sistemler meydana gelmiştir. Bunların her biri farklı aşamalardan ibarettir.

Genel olarak karakter tanımanın 2 yöntemi vardır: Şablon tabanlı tanıma ve özelliğe dayalı tanıma [18]. Şablon tabanlı tanıma, şablon adı verilen bir grup bilinen karakter görüntüsüyle her bir girdi karakter görüntüsünü eşleştirir [18]. Fakat özelliğe dayalı (özellik tabanlı) tanımada, bu yaklaşım tanımlama veri tabanıyla girdi görüntü belirleme özelliklerini veya etiketlerini karşılaştırma eğilimindedir [18].

Bu çalışmada PTS için şablon eşleştirme ve öğrenme yöntemleri kullanılmıştır. Bu iki yöntemler bir-birine bağlı şekilde çalışmaktadır [1]. Bu çalışmada aynı zamanda bir önceki başlıkta bahs edildiği gibi hazır kütüphaneler kullanılarak da karakter tanıma işlemi hayata geçirilmiştir.

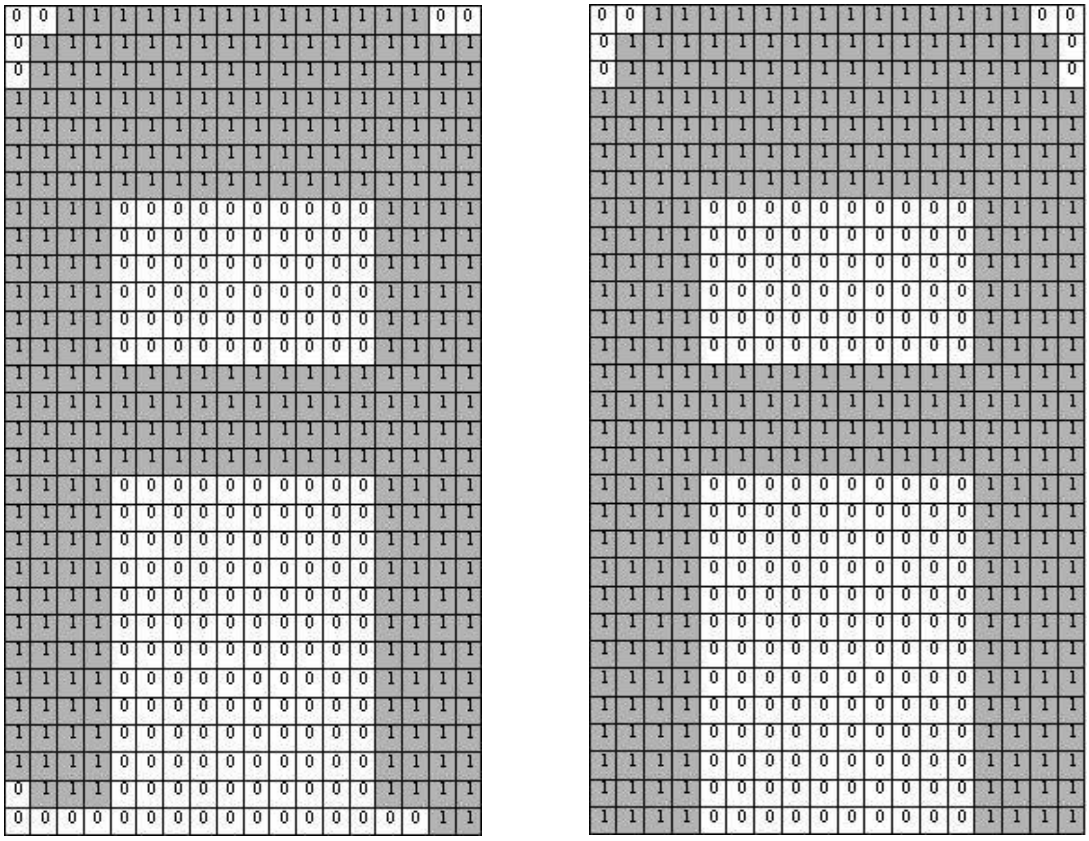
3.5.4.1 Şablon tabanlı tanıma

Şablon tabanlı tanımada gerekli olan 2 bileşen vardır: kaynak görüntü ve şablon görüntü. Önceden veritabanına kayd edilmiş şablon karakterlerle bulduğumuz kaynak karakter görüntüleri karşılaştırılarak bulunuyor. Bulunmadığında ise Öklid Uzaklığı ile bulunuyor. Öklid uzaklığı iki nokta arasındaki doğrusal uzaklıktır.

Veri tabanında kayd edilen karakterlerin sayısının çok olması daha başarılı sonuç elde edilmesine neden olsa da karşılaştırırken çok fazla zaman harcanacak ve sistemi yavaşlatacaktır.

Şablon eşleştirme yöntemi piksel bazında uygulanıyor. Yani şablon eşleştirme yönteminde plaka resmi üzerinde ayrıştırılan karakter resmi veri tabanındaki şablonlarla piksel-piksel karşılaştırılır ve en iyi sonucu üreten yani en çok benzeyen şablon resmin karşılığı olan karakter olarak kabul edilir [6]. Fakat her karakterin tüm pikselleri şablonların tüm pikselleriyle karşılaştırılırsa çok zaman harcanır. Bu nedenle farklı yöntemlerle de karakter ve şablonlar karşılaştırıla bilir. Örneğin karakterin belli

oranda piksellerinin veya satır ve sütunundaki piksel değerlerinin(siyah piksellerin sayısı) karşılaştırılması mümkündür. Bu zaman karşılaştırılmaların sayısı az olduğu için zaman az harcanmış olur. Aynı zamanda harf ve rakam tablolarını veri tabanında ayrı tutarak da zamandan tasarruf etmek mümkündür.



a. b.

Şekil 3.14 Şablon Eşleştirme Yöntemi a. Plaka Üzerinden ayrıştırılan karakter resmi b. veri tabanında karşılaştırma sonucu bulunan karakter (en yakın karakter)

4 GELİŞTİRİLEN PLAKA TANIMA SİSTEMİ

Tezimin konusu olan plaka tanıma sistemi Azerbaycan Devlet plakaları için tasarlanmıştır. Bunun başlınca nedenlerinden bir tanesi Azerbaycan'da plaka tanıma sistemlerine ihtiyaçların olmasına rağmen henüz çok gelişmemesidir. Azerbaycan Devlet plakaları da Türk Devlet plakaları gibi beyaz zemin üzerinde siyah karakterlerden oluşmaktadır. Bu plakalar 3 kısımdan oluşmaktadır: iki rakamlı şehir kodu, rastgele iki harf, rastgele 3 rakamlı adet. Bu kısımlar bir-birilerinden ortadan tek çizgiyle ayırt ediliyor ve plakanın şekli genelde dikdörtgen şeklinde oluyor. Örnek Azerbaycan Devlet Plakası resmine bakalım.

Şekil 4.1 Azerbaycan Plaka Standartlarına Uygun Örnek

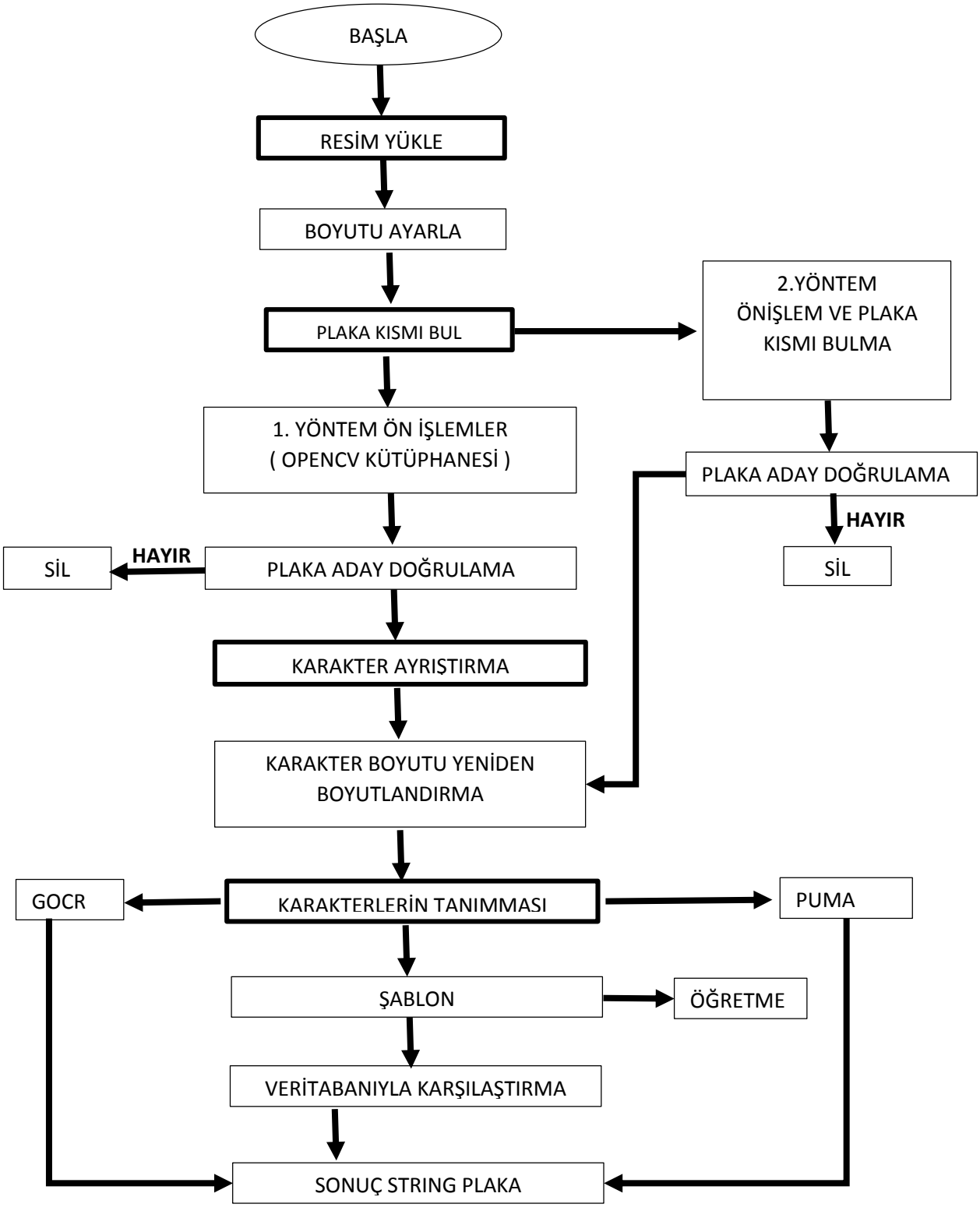
Tez konusu plaka tanıma sisteminde Sony Alpha a390 markalı kameradan ve cep telefonundan çekilmiş aynı zamanda çeşitli internet sitelerinden alınmış Azerbaycan Devlet plakasına sahip araç resimleri kullanılmıştır. Bu resimler farklı zamanlarda, farklı mekânlarda, değişik açılarda, günün çeşitli anlarında çekilmiştir. Sistemin de avantajlarından bir tanesi de farklı çeşit resimlerden plakanı tanımasıdır.

Geliştirilen PTS daha önce de anlatıldığı gibi 3 kısımdan oluşmuştur:

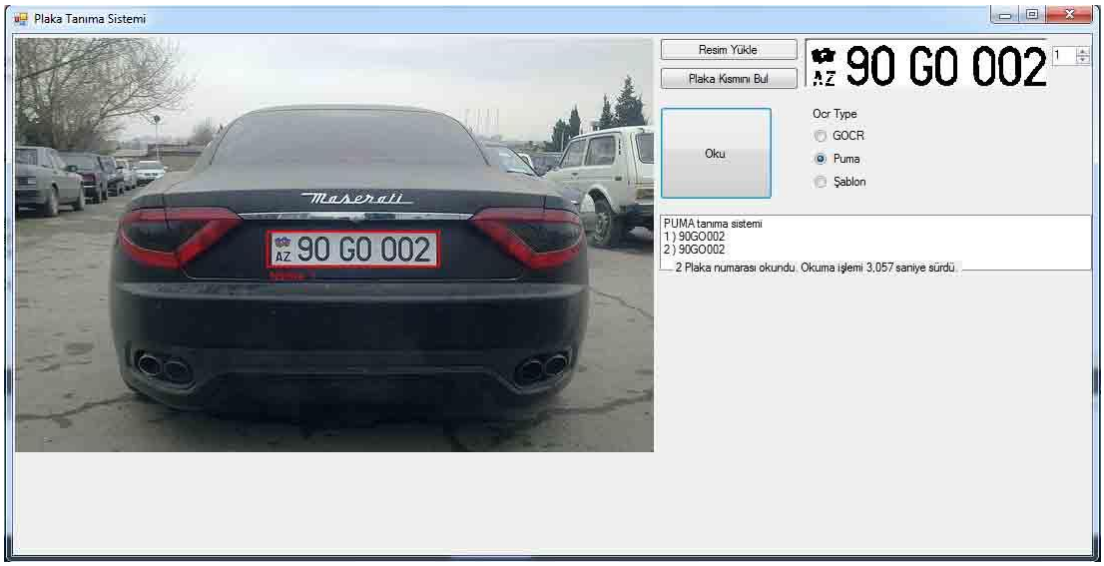
- Plaka bölgesinin bulunması
- Karakterlerin Ayıklanması
- Karakterlerin Tanınması

PTS, Microsoft Visual Studio 2010 paketine dâhil olan C#.NET programlama dilinde yazılmıştır. Daha iyi sonuç için sisteme farklı kütüphaneler entegre edilmiştir. Program başlatıldığında ilk başta resim yükleniyor, daha sonra plaka kısmını bul butonu çalıştırılarak plaka kısmını bulunuyor. Plaka kısmı birkaç tane buluna bilir. Buna neden plaka kısmı bulmak için iki metot kullanılmıştır. Her metot farklı plaka bölge adayları bula bilir. Bulduğu plakaların sayısı yan tarafında kutucukta gösteriliyor. Oku butonuyla ise gerekli sonuç elde ediliyor, yani plakanın karakterleri

metin şeklinde bulunuyor. Okuma algoritmasını farklı yöntemlerle yürüte biliriz. Hazır kütüphane(Puma, GOCR) kullanarak ve ya öğretme algoritması kullanarak(Şablon). Ek olarak da okuma işleminin kaç saniye sürdüğüne dair bilgi de ekranda gözükmektedir. Sistemin ara yüzü Şekil 4.3 de genel olarak algoritması Şekil 4.2 de gösterilmiştir.



Şekil 4.2 Geliştirilen PTS algoritması



Şekil 4.3 Plaka Tanıma Sistemi Ara Yüzü

4.1 Plaka Bölgesinin Bulunması

Bu bölümde çekilmiş resimlerden plaka bölgesinin bulunması için sistemde kullanılan algoritmalar, hazır kütüphaneler anlatılacaktır. Aynı zamanda ilk adım olarak görüntü üzerinde yapılan önişlemler(görüntü üzerinde işlemler) de bu bölümde gösterilecektir. Daha iyi sonuç almak için resimlere görüntü işleme ve plaka tespit etme literatürde araştırılan iki yöntemle uygulanmıştır. Programda kullanılan yöntemlerden bir tanesinde görüntü üzerindeki işlemlerde açık kaynak kodlu, Windows, Linux, Mac OS X, PSP işletim sistemleri üzerinde çalışabilen ve C# programlama dilini destekleyen Open Cv (Open Source Computer Vision Library) Kütüphanesi kullanılmıştır. Bu yöntemde görüntü üzerinde işlemler bu sırayla uygulanmaktadır:

1. Resmin boyutunu ayarlama

2. Gri seviyeye indirgeme

3. Equalized Hist (Histogram Eşitleme)

4. Smooth Gaussian

5. MorphologyEx

6. Threshold

7. Smooth Median

8. Dilate

9. Plaka Bölgesini Doğrulama

Bu sıralama kurulan sisteme daha iyi sonuç vermesine göre değişe bilir. Tezde yazılmış program için ise böyle sıralamanın daha iyi sonuç verdiği anlaşılmıştır. Bu yöntem kullanılarak önişlemlerin yapıldığı fonksiyon aşağıda gösterilmiştir:

```
public void PreProcess()
    {
        IplConvKernel element = Cv.CreateStructuringElementEx(21, 3, 10, 2,
ElementShape.Rect, null);
        timg = new IplImage(src.Size, BitDepth.U8, 1);
        IplImage temp = timg.Clone();
        IplImage dest = timg.Clone();
        src.CvtColor(timg, ColorConversion.RgbaToGray);
        Cv.EqualizeHist(timg, timg);
        pimg = timg.Clone();
        Cv.Smooth(timg, timg, SmoothType.Gaussian);
        Cv.MorphologyEx(timg, dest, temp, element, MorphologyOperation.TopHat,
1);
        Cv.Threshold(dest, timg, 128, 255, ThresholdType.Binary |
ThresholdType.Otsu);
        Cv.Smooth(timg, dest, SmoothType.Median);
        Cv.Dilate(dest, dest, element, 2);
        Cv.ReleaseImage(temp);
        Cv.ReleaseImage(dest);
    }
```

Bu fonksiyonda önişlemler için OpenCV hazır kütüphanesi kullanılmıştır. İlk başta morfolojik işlemler için gerekli boyutu döndüren kod satırı yazılmıştır ve boyutu ayarlanmış orijinal girdi resmi gri seviyeye indirgenmiştir. Sonra gri görüntüye sırasıyla histogram eşitleme, bulanıklaştırma, top-hat dönüşümü işlemleri uygulanmıştır. Elde edilen görüntü Otsu metoduyla ikili resme yani siyah beyaz resme dönüştürülmüştür ve bir daha farklı metotla bulanıklaştırma ve genişletme işlemi

uygulanmıştır. Görüntü üzerinde işlemlerden sonra birinci yöntemin plaka kısmını bulma algoritması uygulanıyor. Programda kullanılan fonksiyon aşağıda gösterilmektedir. Fonksiyonda yapılan işlemler daha detaylı şekilde sonraki bölümde anlatılmıştır.

```
  public int FindPlates()
      {
          IplImage labelImg = new IplImage(src.Size, CvBlobLib.DepthLabel, 1);
          blobs = new CvBlobs();
          plaka.Clear();
          CvBlobLib.Label(timg, labelImg, blobs);
          CvBlobLib.FilterByArea(blobs, 600, 10000);
          IplImage srctemp = src.Clone();
          CvBlobLib.RenderBlobs(labelImg, blobs, src, srctemp,
RenderBlobsMode.BoundingBox | RenderBlobsMode.Angle);
          foreach (var item in blobs)
          {
             item.Value.SetImageROItoBlob(pimg);
             //Cv.ShowImage("s", pimg);
             //MessageBox.Show("ss");
             // ratio values of plate between 3.5 and 5.4
             double ratio = (double)item.Value.Rect.Width / item.Value.Rect.Height;
             double angle = (double)item.Value.CalcAngle();
             if (ratio > 3.5 && ratio < 5.4 && angle > -15 && angle < 15)
             {
                IplImage plakatemp = new IplImage(new CvSize(250, 50), pimg.Depth,
pimg.NChannels);
                Cv.Resize(pimg, plakatemp, Interpolation.Area);
                Cv.Threshold(plakatemp, plakatemp, 128, 255, ThresholdType.Binary |
ThresholdType.Otsu);
                plakatemp = Morfolojik_nomre(plakatemp);
                plakatemp = simvol_ayirma_2(plakatemp);
                plaka.Add(plakatemp);
                src.Rectangle(item.Value.Rect, new CvScalar(0, 0, 255), 2,
LineType.Link4);
```

```
    }
}
src.ResetROI();
return plaka.Count;       }
```

Plaka konumu tespitinde kullanılan ikinci yöntem ise Hüseyin Atasoy [27] tarafından geliştirilen yöntemdir. Burada hem matematiksel hesaplamalar, hem de hazır kütüphaneler kullanılmıştır. Bu yöntemde plaka tespit etmek için uygulanan işlemler böyledir:

1. Gri seviyeye indirgeme
2. Gabor filtresi
3. İkili biçime dönüştürme
4. Yatayda genişletme
5. Bağlantılı Bileşenleri Etiketleme
6. Bileşenlerin Koordinatlarını tespit etme
7. Plaka bölgesini doğrulama

Bu yöntemin program kısmına daha ilerde bölümlerde tek-tek bakılacak.

4.1.1 Plaka bölgesinin bulunması için yapılan önişlemler

Yeniden Boyutlandırma- Resimlerin boyutu pencereden büyük ise resim yeniden boyutlandırılıyor.

Gri Seviyeye İndirgeme- Plaka kısmının bulunması için kullanılan her 2 yöntemde de bu aşama uygulanmıştır. RGB değerlerinde olan girdi resmini gri değerlerine sahip resme çevirme aşamasıdır. Gri seviyeye çevirmenin nedeni plakanın arka zemininin renkli değil de beyaz olmasıdır. Bu durumda renkli kısımlar sadece işi zorlaştırıyor bunun için de griye dönüştürülüyor. Plaka konumu tespitinde kullandığım ilk yöntemde resim OpenCv hazır kütüphane aracılığıyla, ikinci yöntemde ise daha önce belirtilen formülle(Denklem1) gri seviyeye indirgenmiştir. Bu formülde piksellerin üç ana renk değerlerinin ortalaması bulunarak tek boyutlu diziye atanmıştır. Programda yazılmış algoritmasına bakalım:

```
int x, y;
        int genislik = (int)pictureBox1.Image.Width;
        int yukseklik = (int)pictureBox1.Image.Height;
        byte[] gPixeller = new byte[(int)genislik * yukseklik]; // boz piksellər
```

```
int[] rPixeller = new int[(int)genislik * yukseklik]; // rəngli piksellər

Bitmap resim = (Bitmap)pictureBox1.Image;
Color renk;
int sayi;
for (y = 0; y < yukseklik; y++)
   for (x = 0; x < genislik; x++)
   {
      renk = resim.GetPixel(x, y);
      sayi = renk.R;
      sayi += renk.G;
      sayi += renk.B;
      gPixeller[y * genislik + x] = (byte)(sayi / 3);
   }
```

Kurulan bu algoritmalarla griye dönüştürme adımının sonucuna bakalım:

Şekil 4.4 Orijinal Ve Gri Seviyeli Resim

Bu indirgeme sistemin çalışma hızını da belirli bir oranda yükseltmektedir [18]. Çünkü RGB ile 3 farklı düzlemde çalışmak yerine, sadece tek gri düzlem üzerinde çalışılmaktadır [36].

Histogram Eşitleme(Equaliz Histogram)- Bu yöntem birinci metotta kullanılmıştır. OpenCv kütüphanesinin histogram eşitleme fonksiyonundan yararlanılmıştır. Gri seviyeye indirgenmiş resme histogram eşitleme işlemi uygulanmıştır. Böylece resmin gri ton dağılım gösterge çizelgesinde homojenliği sağlanmıştır. Gri seviye ve histogram işlenmiş görüntülere bakalım:

Şekil 4.5 Gri Seviyeli Ve Histogram Eşitleme Uygulanmış Resim

Smooth Gaussian- Birinci metotta OpenCv kütüphanesi kullanılarak resme uygulanmıştır. Histogram eşitleme işleminden sonra bu işlem yapılıyor. Gri görüntüde olan gürültüleri aradan kaldırarak resme yumuşaklık veriyor. Programda kullanılan örnek resme bakalım:

Şekil 4.6 Histogram Eşitleme Uygulanmış Ve Bulanıklaştırılmış Resim

Top-Hat Dönüşümü- Bu yöntem plaka bulma kısmının birinci metodunda kullanılmıştır. Gri seviyeye indirgenmiş resim üzerinde top hat dönüşümü yapılmıştır. Tezde yapılmış bu programda bulanıklaştırılmış(yumuşatılmış) gri seviyeli resim üzerine OpenCv Kütüphanesi aracılığıyla bu işlem uygulanmıştır. Resmin rengi koyulaşarak gereksiz bazı yerler aradan kaldırılmıştır, gerekli yerler ise daha belirginleştirilmiştir. Top-Hat dönüşümü uygulanmış örnek resme bakalım:

Şekil 4.7 Bulanıklaştırılmış Ve Top-Hat Dönüşümü Uygulanmış Resim

İkili Görüntü (Threshold)- Plaka bölgesi saptanması için kurulan her iki yöntemde de kullanılmıştır. İkili resme çevirmenin en hızlı yoludur [37]. Birinci metotta tophat dönüşümünden sonra gri seviyeli resime uygulanarak siyah-beyaz forma dönüştürülüyor. Bu görüntüde tüm pikseller sadece iki değerden birini(siyah, beyaz) ala bilir [38]. Bundan sonra görüntü üzerindeki işlemler iki renk yani siyah(0), beyaz(255) olmak üzere iki bit üzerinde kurulacaktır.(ikinci yöntemde nasıl yapılmıştır hangi sonuc alınmıştır resimli olarak koy) Programda çevrilmiş ikili resme bakalım:

Şekil 4.8 Top-Hat Dönüşümü Uygulanmış Ve İkili Resim

Smooth Median- İlk yöntemde kullanılmıştır. Yaygın olarak kullanılan yumuşatma(smooth) tekniklerindendir [28]. Programda girdi olarak verilen siyah beyaz resimden kenar çizgileri koruma şartıyla, küçük gereksiz beyaz ayrıntıları siliyor. Plaka kısmımız beyaz olduğu için ve görüntüde daha az beyaz ayrıntı kalması için bu işlem işimize yarıyor. Bu tekniğin resme uygulanmış örneğine bakalım:

Şekil 4.9 İkili Resim Ve Smooth Median Uygulanmış Resim

Dilate- ilk metotta kullanılmış morfolojik işlemdir. Genişletme işlemi de deniliyor. Siyah beyaz resim üzerine yapılıyor. Daha önce de anlatıldığı gibi bu işlemin amacı kesik, kopuk kısımları birleştirmektir. Araba plakası beyaz olduğu için programda da bu işlem beyaz kısımları genişleterek plaka kısmının bulunmasına kolaylık sağlıyor. Aşağıda resimden görüldüğü gibi beyaz kısımlar genişletilmiştir.

Şekil 4.10 Smooth Median Ve Dilate Uygulanmış Resim

Gabor Filtresi- Bu işlem ikinci metotta kullanılmıştır. GaborFiltresi.dll kütüphanesi aracılığıyla uygulanmıştır. Resimde olan yatay ayrıntılar gabor süzgeçlerinden geçiriliyor. Resimde geriye dik ve dike yakın açısı olan ayrıtlar kalıyor [27]. Programda gabor filtresi kütüphanesi gönderilen parametreler; yönelim açısı 90, faz açısı 0, en-boy oranı 0.9, band genişliği 1, standart sapma 0(0değeri standart sapmanın kütüphane tarafından hesaplanmasını sağlar), dalga boyu 4 [27]. İşlemin kod parçasına bakalım.

```
try
        {
           GaborFiltresi(90, 0, (float)0.9, (float)1, (float)0, 4, ref gPixeller[0],
genislik, yukseklik);
        }
        catch (DllNotFoundException e) { MessageBox.Show(this, "GaborFiltresi.dll
kutuphanesi bulunamadi.", "HATA", MessageBoxButtons.OK,
MessageBoxIcon.Error); lblBilgi.Text = "Hata!"; return; }
```

Ikili Biçime Dönüştürme ve Yatayda Genişletme- İkinci yöntemde kullanılmıştır. Görüntü sabit bir eşik değeri ile eşiklenerek ikili biçime dönüştürülür [27]. Programda kullanılan eşik değer;70 [27]. İkili biçime dönüştürme sonucu siyah-beyaz resim elde ediyoruz. Yatay genişletme işleminde ise görüntüde olan beyaz kısımlar yatay şekilde genişletiliyor. Programda hem ikili seviyeye dönüşüm hem de genişletme işlemleri .NET in Grapics sınıfından yararlanılarak gerçekleştirilmiştir [27]. İkili seviye dönüşümü için basit bir eşikleme yapılmış, genişletme işlemindeyse görüntü üzerinde rastlanan her bir beyaz pixel için, pixeli orta nokta kabul eden 16 pixel genişliğinde yatay çizgiler çizilmiştir [27].

```
resim = new Bitmap(genislik, yukseklik);
        Graphics grafik = Graphics.FromImage(resim);
        Pen kalem = new Pen(Brushes.White);
        grafik.Clear(Color.Black);
        for (y = 0; y < yukseklik; y++)
           for (x = 0; x < genislik; x++)
              if (gPixeller[y * genislik + x] > esik) // Eşikleme
              grafik.DrawLine(kalem, x - genisletmeMiktari, y, x +
genisletmeMiktari, y);
```

Bağlantılı Bileşenleri Etiketleme Ve Koordinatlarını Tespit Etme- Programda ikinci plaka tespit etme metodunda kullanılmıştır. Bu işlemler BaglantiliBilesenEtiketleme.dll kütüphanesi aracılığıyla uygulanmıştır [27]. Görüntüde bileşenler bağlantılı bileşen etiketleme yöntemiyle etiketlenir ve

koordinatlarını tespit etme kolaylaşır [27]. Her bir etiketlenmiş bileşenin en sol-üst ve en sağ-alt pixellerinin konumu tespit edilir ve bu değerler kullanılarak en-boy bilgileri hesaplanır [27]. Programda kullanılan fonksiyonu aşağıda yazılmıştır:

```
int bilesenSayisi = 0;
        try
          {
             Etiketle(ref gPixeller[0], genislik, yukseklik, false, ref bilesenSayisi, ref
rPixeller[0]);
          }
          catch (DllNotFoundException e) { MessageBox.Show(this,
"BaglantiliBilesenEtiketleme.dll kutuphanesi bulunamadi.", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error); lblBilgi.Text = "Hata!"; return; }
          int[] x1 = new int[bilesenSayisi], y1 = new int[bilesenSayisi], x2 = new
int[bilesenSayisi], y2 = new int[bilesenSayisi];
          try
          {
             KoordinatTespit(ref gPixeller[0], genislik, yukseklik, bilesenSayisi, ref
x1[0], ref y1[0], ref x2[0], ref y2[0]);
          }
          catch (DllNotFoundException e) { MessageBox.Show(this,
"BaglantiliBilesenEtiketleme.dll kutuphanesi bulunamadi.", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error); lblBilgi.Text = "Hata!"; return; }
          grafik = pictureBox1.CreateGraphics();
          kalem = new Pen(Brushes.Red, 2);
          int en, boy, tespitEdilen = 0;
          for (x = 0; x < bilesenSayisi; x++)
          {
             x1[x] += genisletmeMiktari;
             x2[x] -= genisletmeMiktari;
             en = x2[x] - x1[x] - genisletmeMiktari;
             boy = y2[x] - y1[x];
```

4.2 Plaka Bölgesini Doğrulama ve Karakterleri Ayrıştırma

4.2.1 Plaka bölgesini doğrulama

1.yöntemde- Önişlemler sonrası plaka kısmını bulmak için elde edilen en son dilate işlemi uygulanmış görüntü FindPlates() fonksiyonuna gönderiliyor. Bu fonksiyon plaka aday kısımlarını bloblarda tutuyor ve önce belli bir aralığın (600-10000 bu reqemlerin ne olduğunu oyren) dışında olan blobları listeden siliyor. Geri kalan dikdörtgen plaka aday kısımlarını tek-tek ayrı bir görüntü olarak kontrol ediyor. Onların genişliğinin yüksekliğine olan oranını ve açısını hesaplıyor. Bu oran 3.5-5.4 Aralığındaysa ve açısı -15 ve 15 Aralığındaysa eğer plaka bölgesi bulunmuş sayılıyor. Bulunan plaka bölgesinin orijinal haliyle ve boyutlarıyla yeni resim oluşturuluyor. Örnek olarak bulunan plaka görüntüsüne ve programda yazılmış kodlarına bakalım:

Şekil 4.11 Bulunan Orijinal Plaka Bölgesi

```
public int FindPlates()
    {
        IplImage labelImg = new IplImage(src.Size, CvBlobLib.DepthLabel, 1);
        blobs = new CvBlobs();
        plaka.Clear();
        labelImg.SaveImage(@"c:\ss1.jpg");
        CvBlobLib.Label(timg, labelImg, blobs);
        CvBlobLib.FilterByArea(blobs, 600, 10000);
        IplImage srctemp = src.Clone();
        CvBlobLib.RenderBlobs(labelImg, blobs, src, srctemp,
RenderBlobsMode.BoundingBox | RenderBlobsMode.Angle);
        foreach (var item in blobs)
        {  item.Value.SetImageROItoBlob(pimg);
           Cv.ShowImage("s", pimg);
           MessageBox.Show("Plaka bolgesi adayi");
           double ratio = (double)item.Value.Rect.Width / item.Value.Rect.Height;
           double angle = (double)item.Value.CalcAngle();
           if (ratio > 3.5 && ratio < 5.4 && angle > -15 && angle < 15)
```

2.yöntemde- Doğal olarak uygulanan önişlemler sonucu bir kaç tane plaka bölgesi adayları oluşacaktır. Bunların hangisinin plaka olduğunu tespit etmek için programda yazılmış plakaOlabilirmi() fonksiyonu kullanılıyor. Bileşenlerin en ve boy değerleri plakaOlabilirmi() fonksiyonuna yollanır ve burada basit bir en-boy oranı kontrölünden geçirilir [27].

```
private bool plakaOlabilirMi(int en, int boy)
    {
       if (boy < 5) return false;
       if (Math.Abs((float)en / boy - 4.5) < 2 && en > 40) return true;
       return false;
    }
```

Plaka sağa veya sola eğilimle olabileceği için en/boy oranına +-2 kadarlık fark töleransı tanınmalıdır [27].

4.2.2 Karakterlerin ayrıştırması

Bir önceki modüller ile plaka kısmı doğrulandıktan sonra orijinal resimden plaka kısmını koparıyoruz. Üzerinde işlemler uygulanmış resmi kullanmamamızın nedeni resmin kalitesiz yani detaylı olmamasıdır. Çünkü plaka kısmı bulma işlemlerinin daha hızlı uygulana bilinmesi için resmin boyutu 320x240 boyutuna kadar küçültülmüştür. Karakterlerin ayrıştırması modülünde orijinal resimden plaka kısmı koparıldığı için tekrar olarak plaka kısmına ilkin algoritmalar uygulanıyor. Programın bu modülünde de yine daha iyi sonuç almak için iki farklı sıralamayla bu algoritmalar görüntüye uygulanmıştır. Daha önce resmin tamamına uygulanan bu algoritmalar 1.yöntemde ve 2.yöntemde bu şekilde uygulanmıştır:

1.yöntem	**2.yöntem**
Boyutunu ayarlama	Gri seviyeye indirgeme
İkili görüntüyü indirgeme	Histogram beraberleme
	Boyutunu ayarlama
	İkili görüntüyü indirgeme

Her iki yöntemde de elde edilen sonuç görüntü morfolojik filtreleme işlemi yapan *Morfolojik_nomre()* daha sonra simvol_ayirma_2() fonksiyonuna gönderiliyor. Bu filtrelemeler plaka karakterlerin ayrıştırılmasında önemli rol oynamaktadır. Çünkü karakterlerin sağlam bir şekilde arka plandan ve bir-birlerinden ayrılmaları için görüntüdeki gürültülerin mümkün olduğu kadar karakterlere dokunmadan yok edilmeleri lazımdır. Bu fonksiyonlarda bazı istatistik ölçülendirmeler kullanılmaktadır. Örneğin Azerbaycan plaka standartlarındaki sivil plakaları 7 karakterden oluşuyor. Karakterler arasındakı boşlukları göze aldıkta plakalardaki hiç bir karakterin yatay genişliği plakanın toplam genişliğinin 1/7-dən fazla ve alanı 100-den az değildir. Bu nedenle yatayda 1/7 nömrə ölçüsünü aşan tüm siyah satırlar beyaz renge çevriliyor. Aynı zamanda plaka üzerindeki "-" sembolu da gereksiz bir detal olduğundan ve alanı, yani piksellerinin sayı 100-den az olduğu için haric edilecektir. Bu işlemler sonrası plaka üzerindeki gürültüler büyük oranda yok olucaktır. Aynı zamanda plakadaki yatay ve dikey uzunluğu 2 pikselden küçük genişliğe sahip siyah kısımlar da beyaza çevrilmelidir. Bu yöntemlerle siyah-beyaz görüntüdeki gereksiz detallar ve gürültüler büyük bir oranda yok edilecektir.

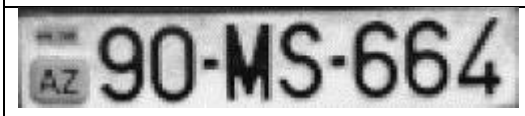
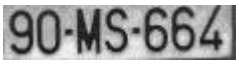
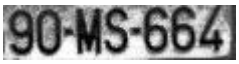
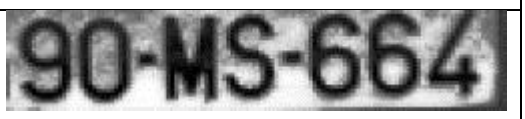
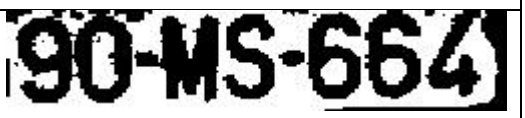
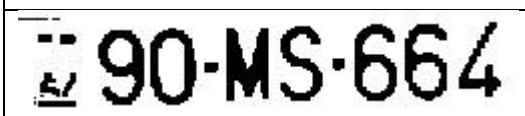
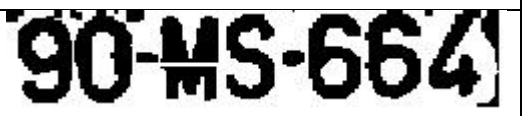
C# dilinde programda yazılmış fonksiyona bakalım:

```
            {
                IplImage plakatemp = new IplImage(new CvSize(250, 50), pimg.Depth,
pimg.NChannels);
                pimg.SaveImage(@"c:\ss2.jpg");
                Cv.Resize(pimg, plakatemp, Interpolation.Area);
                plakatemp.SaveImage(@"c:\ss3.jpg");
                Cv.Threshold(plakatemp, plakatemp, 128, 255, ThresholdType.Binary |
ThresholdType.Otsu);
                plakatemp = Morfolojik_nomre(plakatemp);
                plakatemp = simvol_ayirma_2(plakatemp);
                plakatemp.SaveImage(@"c:\ss4.jpg");
                plaka.Add(plakatemp);
                src.Rectangle(item.Value.Rect, new CvScalar(0, 0, 255), 2,
LineType.Link4);
            }
        }
```

```
            src.ResetROI();
            return plaka.Count;
        }
```

Bu algoritmalardan geçerek oluşan sonuç plaka görüntülerine bakalım:

Çizelge 4.1 Plaka Kısmına Uygulanan İşlemlerin Sonuçları

1.yöntem sonucu plaka kısmı	*2.yöntem sonucu plaka kısmı*
	
	
	
	
Morfolojik Filtreleme sonuçları	
	
	

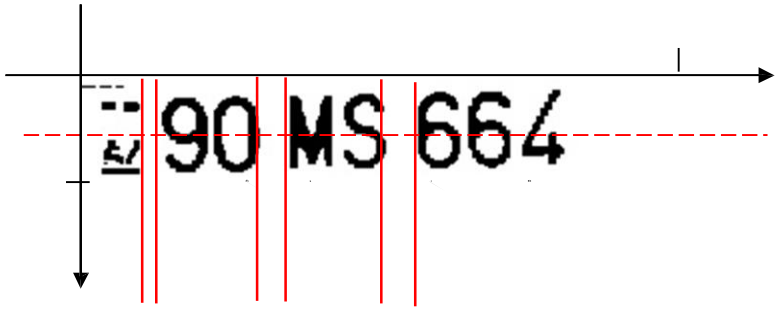
4.2.3 Karakterlerin yeniden boyutlandırılması(boşlukları belirleme)

Bu modül karakterlerin tanınması kısmında Şablon eşleştirme yönteminde kullanılacaktır. Karakterlerin tanınmasında kullanılan diğer iki yöntem ise bir önceki modül olan karakterlerin ayrıştırması modülünün sonuçlarını kullanarak tanıma işlemini hayata geçiriyor. Çünki bu yöntemlerde resmi karaktere dönüştüren hazır kütüphaneler kullanılmıştır.

Azerbaycan plaka standartlarındaki sivil plakaları üç kısımdan oluşuyor. İlk kısımda plakanın bağlı olduğu bölgenin kodunu gösteren iki haneli adet, ikinci kısımda 2 tane

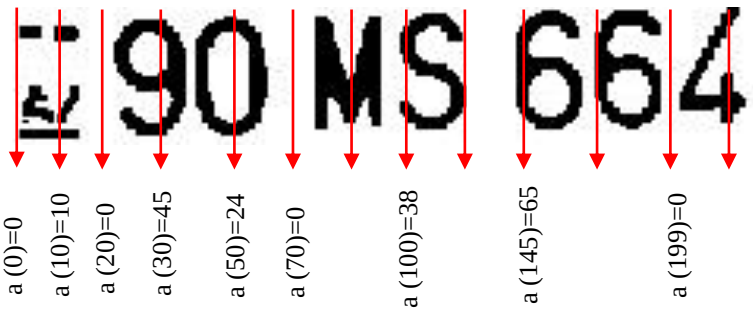
harf və üçüncü kısımda 3 haneli adet oluyor. Bu bilgi daha sonra yapılacak işlemlere yol gösterecektir. Yani plakalarda yataydaki en büyük boşluklar plakanın başında olan "az" logosu ile adet ve adet ile harfler arasında olacak.

Burada dikkat edilmesi gereken şey boşlukları görüntünün dikey okunun hangi noktasında aranacağıdır [18]. Bunun için bir çizgi belirlenmesi ve o çizgi boyunca boşluk aranması lazımdır ki en uygun satır da görüntünün dikey uzunluğunun aritmetik ortalaması olan satırdır [18]. Plakanın ortasından yatay çizgi çizerek boşlukları belirleyelim.



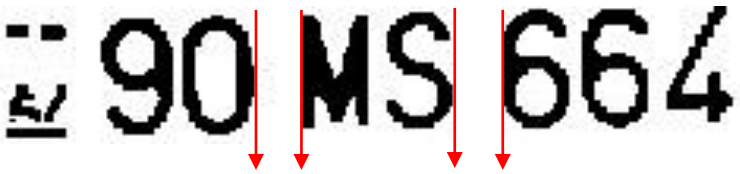
Şekil 4.12 Boşlukların Tespit Edilmesi

Plakadaki büyük boşluklar bulunduktan sonra dikey arama işlemine geçilir. Dikey arama sol yukarı köşede ilk pikselden yukarıdan aşağıya doğru yani x=0-dan x=n değerine kadar her x değeri için y=0-dan y=h değerine kadar aramadır. Bu işlem zamanı diziler kullanılıyor. Yani her x değeri için her sütundakı siyah piksellerin sayısı fazla olursa (0-2 arası gürültü kabul edilmektedir) o sütunda karakterin bir kısmının piksellerinin bulunduğu kabul edilir ve bu sütündakı bulunan piksellerin sayısı a dizisine aktarılıyor. Programda kullandığımız plaka örneği üzerinde aşağıdakı şekilde arama işlemi gerçekleştirilir (resimde verilen piksel değerleri gerçeği yansıtmamaktadır).



Şekil 4.13 y Oku Boyunca Siyah Piksellerin Sayısı(örnek değerlerdir)

Yapılan dikey arama işleminden sonra plakada boş olan kısımlar diziye 0 olarak aktarılıyorki, böylece boşluklara esasen karakterlerin yerleri kolaylıkla buluna bilinir. Harfler kısmının da yani plakanın ikinci kısmının yatayda başlanğıç ve bitiş noktalarının koordinatlarını diziye aktarılan değerler sayesinde kolaylıkla bulmak mümkündür. Harfler kısmının sol yanındaki bu büyük boşluktan(2-ci boşlukdan) sola doğru yapılan dikey arama plakanın şehir kodunun yataydaki bitiş noktasının koordinantı veriyor. Sağ yanındaki büyük boşluktan(3-cü boşluk) sağa doğru yapılan dikey arama sonucu ise plakanın 3cü rakamlar kısmının başlanğıc koordinatını göstermektedir.



(1) (2) (3) (4)

Şekil 4.14 Plakada Bulunun 3Kısmın Başlama Ve Bitme Noktaları

1) Birinci kısmın son noktası 2) İkinci kısmın başlanğıc noktası

3) İkinci kısmın son noktası 4) Üçüncü kısmın başlanğıç noktası

Plaka görüntüsüne ait sütunların taşıdıkları piksel değerlerini kendisinde tutan a dizisi temelinde diğer bir b dizisi de oluşturuluyor. Bu dizide pikseller ikili sistemde olduğu gibi 0 ve 1 lerle ifade olunuyor. Yani a dizisinde 0-dan farklı değer taşıyan sembolleri b dizisinde aynı indeksli yere değeri "1" olarak kayd ediliyor. b dizisinin diğer sembollerinin değeri ise "0" olarak kalıyor.

b(x)01111001111011110001111101111000111101111001111 0

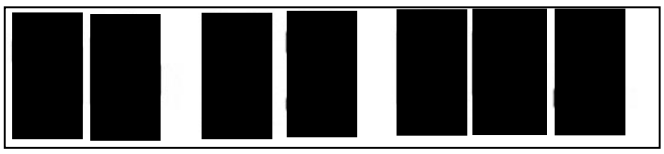
Şekil 4.15 Plakanın b Dizisine 0 Ve 1 İle Yazılması

B dizisi artık plaka karakterlerine ait bilgileri içermektedir. Yani dizide birinci büyük boşluğu içeren “0” lar kümesinden önce gelen “1” ler kümesi şehir kodunu göstermektedir. Azerbaycan Devlet plakalarında bildiğimiz gibi şehir kodunun önünde olan AZ sembolü vardırki, bu aşamaya kadar bazen bu sembol kalmış durumda oluyor. Bunun için bu aşamada ilk “1” ler kümesini oluşturan şehir kodundan önceki

“1” ler gürültü olarak algılanıyor ve siliniyor. 1-ci büyük boşluk ile 2-ci büyük boşluk arasındakı kısım harflerin olduğu kısmın başlangıc və son noktalarını gösteriyor. 2-ci boşluktan sonra gelen “1” ler kümesi ise üçüncü kısmı, rakamların olduğu kısmı gösteriyor. Tüm bu işlemlerin sonucunda plakadaki tüm karakterlerin yatay eksende başlangıc ve bitme noktalarının yerini bulmuş oluruz.

Şekil 4.16 Plakadaki Tüm Karakterlerin Başlangıç Ve Bitiş Noktası

Karakterlerin yeniden boyutlandırılması işlemlerinde şimdiye kadar sadece yatay eksende her bir karakterin başlama ve son kordinatları tespit edildi ve yatay eksende olan gürültüler yok edildi. Bundan sonra dikey eksende de karakterlerin başlangıç ve son noktalarının tespit edilmesi gerekiyor. Bu işlem için smearing alqoritması kullanılıyor. Her karakterin yataydaki minimum ve maksimum koordinatları (xmin, xmax) bellidir. Her karakter aralığı y=0-dan y=h noktasına kadar yatayda aranarak karakterin dikeydeki en küçük ve en büyük değerleri bulunmuş oluyor [18]. Böylelikle her karakterin yatay ve dikey okta başlangıç ve son noktaları belilenerek karakterler bir-birlerinden ayrılmış oluyor ve her bir karakterin koordinatları son modül olan karakter tanıma modülüne gönderiliyor [18].



Şekil 4.17 Karakterlerin Dikeyde Sınırlarının Tespiti

4.3 Karakterlerin Tanınması

Karakterlerin tanınması bölümünde 3 farklı yöntem kullanılarak plaka okunuyor. Yöntemlerden iki tanesi yani *GOCR* ve *Puma* hazır kütüphaneleri kullanarak plakanı okuyor, bir tanesi ise yani *Şablon eşleştirme* yöntemi yapay zeka ile öğrenme metodu kullanarak plaka kısmını okuyor.

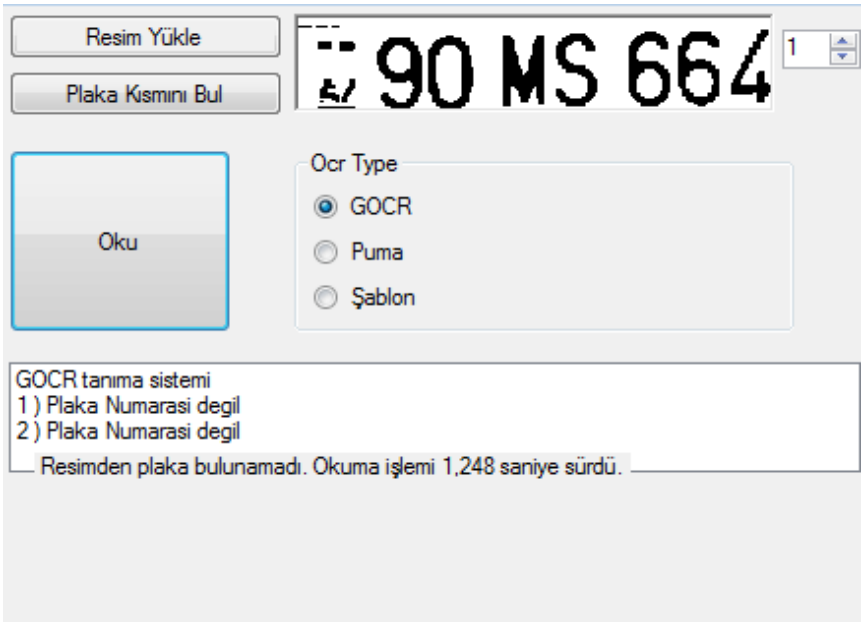
İlk başta yöntemlerden birisi seçilerek Oku butonuna tıklamak lazımdır. Oku butonu tıklandığında program hangi yöntemin seçildiğini tespit ederek o yöntemin fonksiyonunu çağırıyor.

4.3.1 Hazır kütüphaneler ile karakter tanıma

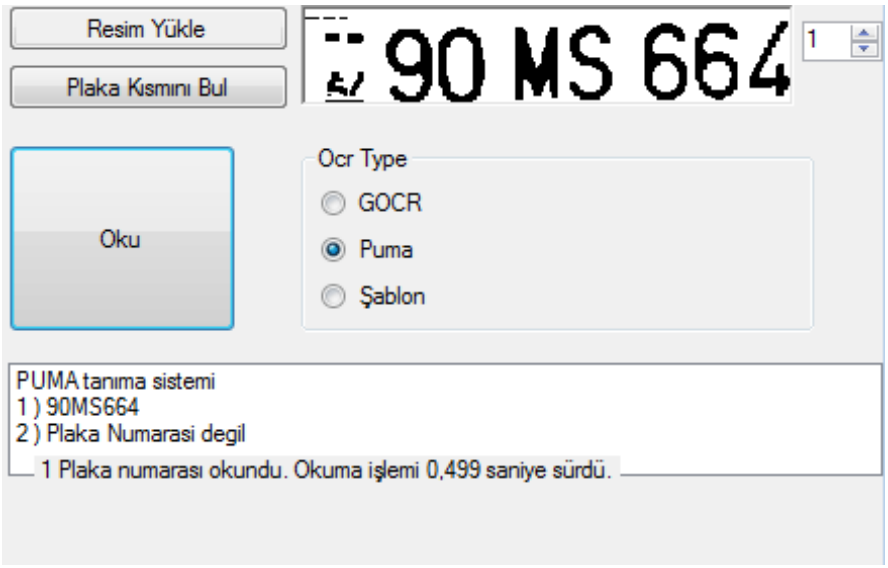
İlk başta GOCR ve Puma hazır kütüphanelerinin bulduğu plaka numaralarını string-e yazarak kontrol ediyor. Azerbaycan plakalarında ilk başta rakam geldiği için. İlk kontrol ilk karakterin rakam olup olmamasıdır. Bu tüm Azerbaycan plakalarında ilk başta eklenmiş AZ sembolünü aradan kaldırmak için kullanılıyor. Çünki AZ sembolü de karakter olduğu için onu plaka tespiti sonrasında, karakterlerin ayrıştırılması algoritmalarıyla plakadan temizlemek bazen başarısız oluyor. Daha sonra ise hazır kütüphanenin bulduğu plaka adaylarının Azerbaycan plaka standartlarına(Şekil4.1) uygun olup olmaması kontrol ediliyor. Şekil4.1 den göründüğü gibi bu plakalarda ilk başta iki rakam şehir kodu, daha sonra iki harf ve üç rakam geliyor. Standartlara uygunluğu da bu şekilde kontrol ediliyor. Eğer standartlara uygun plaka bulunmuşsa ise ekranda plaka gözükecek. Aksi takdirde yalnış harf rakam sıralaması olursa ”plaka numarası değil” şeklinde ekranda gözükecek.

```
public int tan = 0;
    public void ReadPlates(OcrType ocr)
    {
        tan = 0;
        plakaList.Clear();
        int i = 1;
        foreach (var plkimg in plaka)
        {
            bool first_num_finded = false;
            string s = "";
            foreach ( char c in RunOcr(plkimg, ocr))// RunOcr hazir kutuphaneni
cagiriyor burda
            {
                string str = c.ToString();
                if (!first_num_finded) //birinci reqem olub olmamasina baxr AZ in
silinmesi ucun genelde
                {
                    double Num;
                    bool isNum = double.TryParse(str, out Num);
                    if (isNum)
```

```
                {
                    first_num_finded = true;
                    s = s + str;
                }                }
            else
            {
                if (char.IsLetterOrDigit(c))
                    s = s + str;
            }
        }
        if (s.Trim().Length >= 7) s = s.Substring(0, 7); // Runocr den aldigi
karakter neticelerinin Azerbaycan nomre formatina uygunlugunu yoxlayir
        if (Regex.IsMatch(s.ToUpper(), @"^[0-9]{2}[A-Z]{2}[0-9]{3}$"))
        {
            plakaList.Add(i.ToString() + " ) " + s);
            tan++;
        }
        else plakaList.Add(i.ToString() + " ) Plaka Numarasi degil");
        i++;
    }       }
```



Şekil 4.18 GOCR Hazır Kütüphanesi Kullanılarak Okunan Ve Bulunamayan Plaka Örneği



Şekil 4.19 Puma Hazır Kütüphanesi Kullanılarak Okunan Ve Bulunan Plaka Örneği

4.3.2 Şablon eşleştirme yöntemi ile karakter tanıma

Hazır kütüphane kullanılmadığı için plakada olan harfleri ve rakamları ayrıştırmak gerekiyor. Bunun için bu modülde girdi olarak yukarıda anlatılan karakterlerin ayrıştırılması işlemleri sonucunda elde edilen bilgiler kullanılacak. Bu modül hem şablon yöntemiyle karakterlerin tanınması, hem de karakterlerin sisteme öğretilmesi işlemini hayata geçirmektedir. Karakterlerin öğretilmesi hem de tanınması için bir veritabanı kullanılmaktadır. Modülün uygulanması sonucunda karakter görüntülerinden metinsel plaka numarası elde edilecek. Şablon eşleştirme modülü kendisi de genel olarak 2 aşamadan oluşuyor:

1. Karakterlerin Boyutlarının Ayarlanması

2. Şablon Eşleştirme

Hem bu aşamalar, hem de veritabanının özellikleri hakkında ilerki bölümlerde bahs edilecektir.

4.3.2.1 Karakterlerin yeniden boyutlandırılması

Şablon tabanlı tanıma yöntemi için tanıma işlemi uygulanacak karakterlerin genişlik ve yükseklik boyutları sabit bir şekle getirilmelidir [16]. Aynı zamanda karakter ayrıştırma modülünden aldığımız ve girdi olarak kullandığımız karakterlerin üstünde, altında, sağında ve solunda boşlukların olması ihtimali yüksektir [16]. Bu yüzden

tanıma işlemi uygulanmadan önce karakterlerin kesin sınırları belirlenerek bu sınırlar içindeki karakter kullanılır [16]. Bunun için karakterleri kırparak belli bir sabit boyuta getirirsek ihtimal olunan boşluklar da aradan kalkmış olacaktır.

Tasarlanan PTS de karakter şablonları 16x32 boyutunda kullanılmıştır. Bu boyutun seçilmesine neden ise araştırılan kaynaklarda en yüksek performans göstericisi olmasıdır. Araştırma sonucu farklı karakter ebatlarının şablon eşleştirmesi zamanı ortaya çıkan tabloya bakalım:

Çizelge 4.2 Aynı Karakter Eğitim Seti Kullanılarak, Şablon Eşleştirmede Kullanılan Farklı Karakter Ebatlarının Sonuçları [1]

Toplam Karakter Sayısı	Toplam İşlem Süresi (İşlemci: intel 7@1.6 Ghz)	Tarama Yöntemleri	Eşleştirmede Kullanılan Karakter Ebatları	Doğru tanınan Karakter Sayısı	Yanlış Tanınan Karakter Sayısı	Başarı Yüzdesi
6764	21 saniye	YT, DK, SOCT, SACT	10x20	6136	628	%90,71
6764	24 saniye	YT, DK, SOCT, SACT	12x24	6298	466	%93,11
6764	29 saniye	YT, DK, SOCT, SACT	14x28	6379	385	%94,30
6764	30 saniye	YT, DK, SOCT, SACT	15x30	6418	346	%94,88
6764	**29 saniye**	YT, DK, SOCT, SACT	**16x32**	**6480**	**284**	**%95,80**
6764	32 saniye	YT, DK, SOCT, SACT	17x34	6425	339	%94,98
6764	43 saniye	YT, DK, SOCT, SACT	18x36	6433	331	%95,10
6764	32 saniye	YT, DK, SOCT, SACT	19x38	6455	309	%95,43
6764	36 saniye	YT, DK, SOCT, SACT	20x40	6433	331	%95,10
6764	21 saniye	YT, DK	10x20	5585	1179	%82,56
6764	19 saniye	YT, DK	12x24	5779	985	%85,43
6764	21 saniye	YT, DK	14x28	5859	905	%86,62
6764	27 saniye	YT, DK	15x30	5927	837	%87,62
6764	27 saniye	YT, DK	16x32	5939	825	%87,80
6764	26 saniye	YT, DK	17x34	5927	837	%87,62
6764	31 saniye	YT, DK	18x36	5924	840	%87,58
6764	33 saniye	YT, DK	19x38	5936	828	%87,75
6764	35 saniye	YT, DK	20x40	5951	813	%87,98

YT: Yatay tarama, DK: Dikey tarama, SOCT: Sol üsten sağ alta çapraz tarama, SACT: Sağ üsten sol alta çapraz tarama

Bu tablodan görüldüğü gibi resimlerin boyutları büyüdükçe eşleşme süresi düzenli olarak artmamıştır. Bunun nedeni ise karakterin eşleşmesi zamanı şablonda bulunan 1/3 lük kısmı kontrol edilen karaktere %60 benzemiyorsa, bu karakter aranılan karakter değildir [16]. Bu tablodan da en iyi göstericinin 16x32 karakter boyutunda elde edildiği görülüyor. Buna göre girdi olarak verilen karakterlerin 16x32 piksel

büyüklüğüne eşitlenmiştir. Karakterlerin ayrıştırılması yönteminde belirtildiği gibi sonuç görüntüler dizilere aktarılmıştır. Dizilerde kayda alınan karakterleri boyutlandırmak için ise C# dilinin hazır fonksiyonları kullanılmıştır.

İlk başta görüntü Bitmap resmine dönüştürülüyor ve daha sonra *Resize* fonksiyonu ile boyutlandırılıyor. C# dilinde boyutlandırma için kullanılan *Resize* fonksiyonu:

```
public Bitmap Resize(Bitmap limg, int width, int height)
    {
       Image img = limg;
       int originalW = img.Width;
       int originalH = img.Height;
       int resizedW = width;
       int resizedH = height;
       Bitmap bmp = new Bitmap(resizedW, resizedH);
       Graphics graphic = Graphics.FromImage((Image)bmp);
       graphic.InterpolationMode = InterpolationMode.HighQualityBicubic;
       graphic.DrawImage(img, 0, 0, resizedW, resizedH);
       graphic.Dispose();
       return bmp;
     }
```

Yeniden boyutlandırma sonrası elde edilen karakter görüntüleri:

Şekil 4.20 Yeniden Boyutlandırılan Karakterler

Şekil 4.20 den görüldüğü gibi artık elde edilen karakterler tanıma işlemi için hazır hale gelmiştir.

4.3.2.2 Veritabanının geliştirilmesi

Uygulamada karakterlerin tanınması adımında şablon eşleştirme yönteminde kullanılan veritabanı benim tarafımdan MS SQL Serverde oluşturulmuştur. Veritabanı bir "Learn" tablosundan oluşmaktadır. Şablon yöntemiyle tanıma zamanı sisteme öğretilen karakterler bu veritabanına yazılarak burada tutuluyor. Eğer bir karakterin çok fazla şablonları bulunuyorsa veritabanında, tanıma zamanı sistemin plakadan

ayrıştırılan karakteri tanıması olasılığı artmaktadır. Yani bir karakter ne kadar çok öğretilirse ve veritabanında şablonu çok olursa o karakterin tanınma başarısı daha yüksek olmaktadır. Tanınma için ise sistem bulunan karakterle şablon karakterlerin projeksiyon değerlerini karşılaştırıyor. Daha önce de söylenilen gibi karakterlerin tek-tek pikselleri karşılaştırılarak da tanıma gerçekleştirile bilir, ama piksel sayısı fazla olduğu için bu zaman çok fazla zaman harcanmış olur. Veritabanında kullandığım "Learn" tablosunun içeriğine bakalım:

- Proyeksiya
- Simvol

Proyeksiya satrında öğretilen karakterlerin projeksiyon değerleri yer almaktadır. Simvol alanında ise karakterin kendisi tutuluyor. Veritabanından örnek bir parça aşağıda gösterilmiştir.

Learn: Query(C:\Us...NTS\Database1.sdf)

	Proyeksiya	simvol
▶	0,0,3,4,5,5,5,29,...	1
	0,0,22,25,8,9,6,5...	0
	0,3,7,10,11,14,1...	z
	0,2,18,19,9,9,8,8...	5

Şekil 4.21 Veritabanından Örnek Görüntü

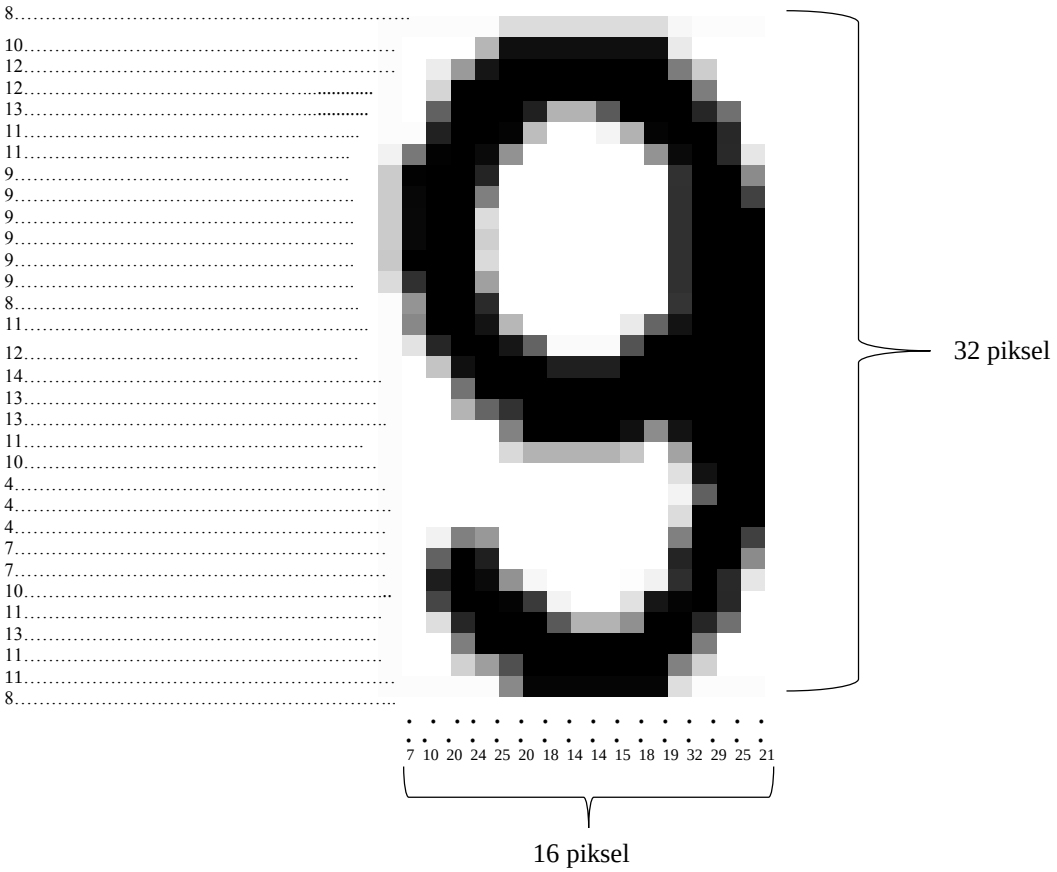
4.3.2.3 Şablon eşleştirme

Şablon eşleştirme modülü, adından da anlaşıldığı gibi karakterlerin şablonlarının yani piksel değerlerinin, daha önceden sisteme tanıtılmış ve hangi karakter olduğu bilinen şablonlarla yani piksel değerleriyle karşılaştırılmasıyla yapılan tanımayı hayata geçiriyor [16]. Bu modül kendisi 3 adımdan oluşmaktadır: Projeksiyon işaretleme, tümevarımsal öğrenme, şablon eşleştirme.

Projeksiyon işaretleme: Projeksiyon işaretlemeye dayalı şablon çıkarma yönteminde her bir karakterin yatay ve dikey projeksiyonlarına bakılır [18]. Bir önceki karakterlerin yeniden boyutlandırılması modülünde karakter görüntülerinin 16x32 şeklinde ayarlandığı gösterilmiştir. Bu sebeple her bir karakter görüntüsünde 16x32=512 adet piksel bulunmaktadır. Eşleştirme yöntemi olduğu için de girdi

karakter görüntülerinin ve şablon karakterlerinin piksellerini karşılaştırmak gerekiyor. Her bir uygun piksel bir-biriyle karşılaştırılırsa ise çok fazla zaman harcayarak gereksiz bir sistem yapılmış olur. Bu nedenle tez çalışmasında projeksiyon işaretleme yöntemi kullanılarak daha az zamanda tanıma işlemi hayata geçirilmiştir. Yani işlemler 512 piksel üzerinde değil 16+32=48 piksel üzerinde yürütülmüştür.

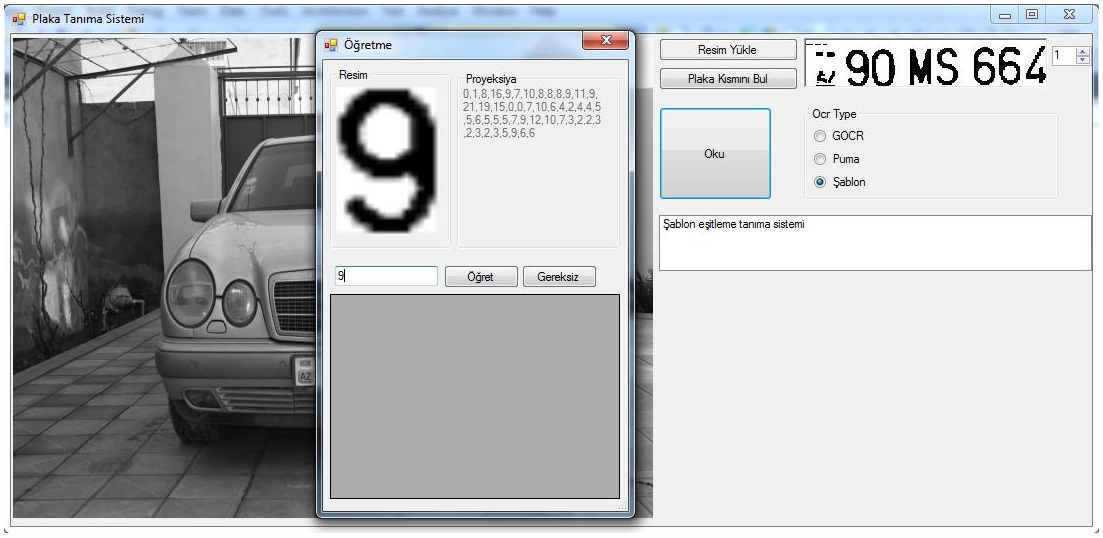
Daha önceki boyutların ayarlanması modülünde a dizisine kayıt edilen karakterlerin yataydaki piksel değerlerini bulma işlemi gibi projeksiyon işaretleme modülünde de her bir karakterin yataydaki ve dikeydeki projeksiyonları bulunarak bir diziye atanacak. Yatay ve dikey projeksiyona bağlı şablon çıkarma yönteminde karakter önce ymin'den ymax' a kadar yatay olarak satır-satır taranır ve her satırın taşıdığı siyah piksel sayısı yatay projeksiyonun 32 adet değerini oluşturur [18]. Daha sonra karakter xmin' den xmax' a kadar dikey olarak sütun-sütun taranır ve her sütunun taşıdığı siyah piksel sayısı dikey projeksiyonunun 16 adet değerini oluşturur [18]. Bulunan projeksiyon değerleri ise hem karakterin öğretilmesi, hem de şablon ile karşılaştırılması zamanı kullanılacaktır.



Şekil 4.22 Karakterin Yatay Ve Dikey Projeksiyonlarının Çıkarılması

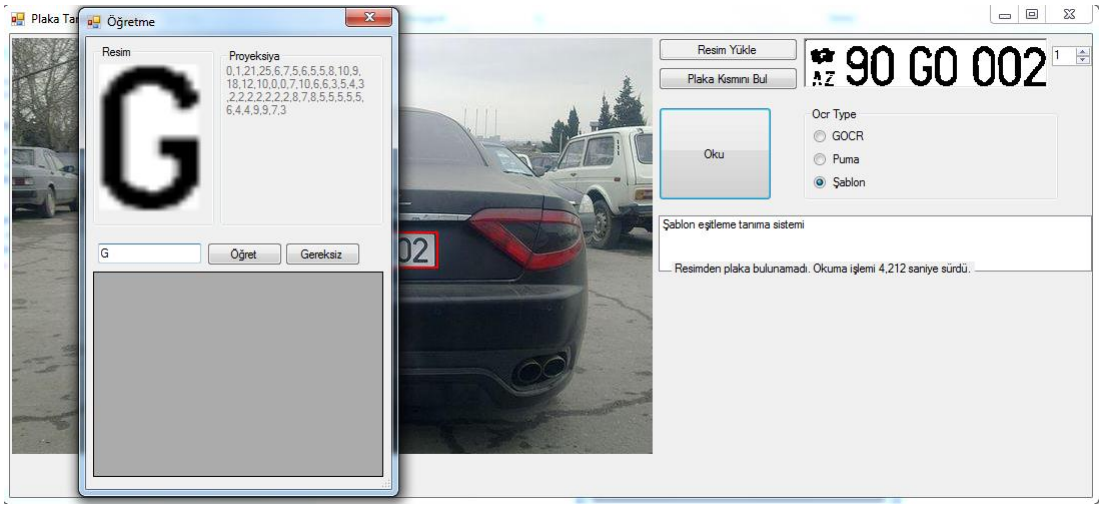
Tümevarımsal öğrenme ve şablon eşleştirme: Sistem çalışmaya başlamakla beraber öğrenmeye de başlamaktadır. Plakadan bulunan karakter görselleri veritabanındaki şablonlarla karşılaştırılıyor eğer bulunursa ekrana metin şeklinde yazdırılıyor. Eğer karşılaştırma sonucunda benzer şablon bulunamazsa o karakter sisteme öğretilerek veritabanına kayıt ediliyor. Böylece bir yandan tanıma işlemi yürütülmekle beraber bir yandan da sisteme karakterler öğretilmektedir. Veritabanında kayıt edilen karakter örneklerinin sayısı arttıkça sistemin doğru sonuç üretme kalitesi de artmaktadır. Bir karakterden ne kadar çok örnek şablon kayıt edilirse sistemin o karakteri tanıma olasılığı da o kadar artıyor. Örneğin 3-9, 0-8, 5-6 gibi bazı karakterler bir-birlerini çok benziyor. Bu yüzden bu karakterlerin örneklerinin veritabanında daha çok kayd edilmesi gerekiyor.

Sistemin daha iyi sonuç vermesi için yapılan adımlardan bir tanesi de rakam ve harf karakterlerini farklı tablolarda tutmaktır. Bildiğimiz gibi Azerbaycan devlet plaka standartlarında ilk ve 3.kısımda sadece rakamlar, ikinci kısımda ise sadece harfler bulunuyor. Bu yüzden plakanın rakam bulunan kısımlarında karakterler veritabanının rakamlar tablosuyla, harf bulunan kısımlarında ise karakterler harfler tablosuyla karşılaştırılıyor. Bu sebepten dolayı bir-birine benzeyen harf ve rakamlarda sistemin benzetme dolayısıyla yanlış sonuç verme ihtimali 0-a iniyor. Aynı zamanda karşılaştırma sayısı azaldığı için zamana da tasarruf yapılmış oluyor. Tez boyunca adımların uygulandığı plakanın şablon eşleştirme yöntemiyle karakterlerinin tanınması adımında 9 rakamının öğretilmesi penceresi aşağıda gösterilmiştir.

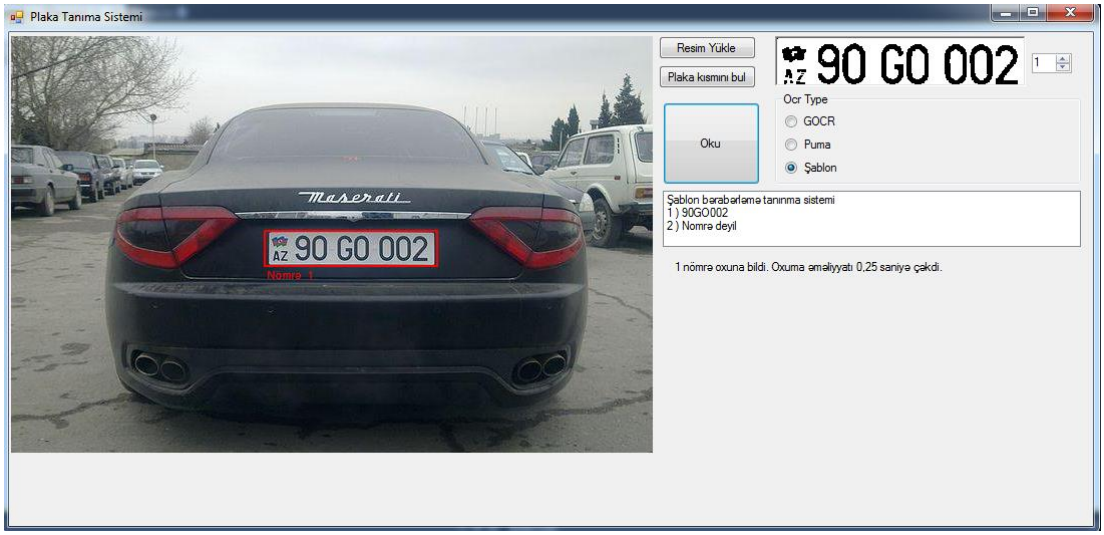


Şekil 4.23 9 Rakamının Öğretilmesi

Diğer bir araba görüntüsünün şablon yöntemiyle başarılı şekilde plaka numarasının bulunması örneğine bakalım. İlk resim öğretilme sırasında, ikinci resim öğretildikten sonra tekrar sisteme giriş yapılarak plaka numarası okunarak başarılı sonuç elde edilmiştir.



Şekil 4.24 G Harfinin Öğretilmesi



Şekil 4.25 Önceden Öğretilmiş Plakanın Şablon Yöntemiyle Bulunması

5 SONUÇ

Araştırılan diğer çalışmalar gibi bu çalışmada geliştirilen uygulamada da fiziki şartlar uygun olduğu zaman yani görüntünün çekilme açısı, arabanın temiz olma durumu gibi şartlar sağlandığı sürece sistem yüksek başarı göstermektedir. Geliştirilen uygulamanın başlıca avantajı izdüşümü yöntemle karakter tanıma kısmındadır. Yani izdüşümü yöntemiyle tanıma işlemi yürütüldüğü zaman tüm pikseller karşılaştırılmadığı için zamandan tasarruf etmiş olunuyor. Sonuçta kısa zamanda tanıma işlemi gerçekleştiriliyor. Araştırmada geliştirilen sistemin Azerbaycan plakalarına uygulanmasının yansıra, yenilik olarak hem plaka kısmı bulunma, hem de karakterlerin tanınması adımında birkaç algoritmanın kullanılmasıdır. Adım tamamlandıktan sonra ise tüm algoritmaların sonucu ekranda gözükmektedir.

Plaka kısmı bulma adımı için yapılan önişlemler 2 farklı algoritmayla hayata geçirilmiştir: Şimdiye kadar birçok yerde kullanılan hazır OpenCv kütüphanesi ile geliştirilmiş algoritma ve Hüseyin Atasoy'un [27] geliştirdiği algoritma.

Karakterlerin tanınması adımı için ise GOCR ve Puma hazır kütüphaneleri ve şablon eşleştirme yöntemi kullanılmıştır. Şablon eşleştirme yönteminin algoritması daha önce söylenildiği gibi izdüşümü yöntemine dayanmaktadır.

Bu tezin temel amacı, Azerbaycan'da plaka tanıma sistemi kullanılan alanlar için genel uygulama geliştirmektir. Genel uygulama olduğu için küçük ilaveler veya sistemler entegre edilerek farklı alanlara uygulana bilir: Otoparklara, site giriş-çıkışlarına, havalimanı ve benzeri yerlerde geçiş kapılarına.

KAYNAKÇA

[1] N. D. Beytullah YALIM, «Türk Tasıt Plaka Standartları İçin Plaka Tanıma Sistemi,» *Bilişim Teknolojileri,* cilt 1, no. 1, pp. 47-50, 1 Ocak 2008.

[2] A. Yayık, «Yapay Zeka ve Uygulamaları,» Hatay, 2014.

[3] F. K., «Necognitron – a hierarchical neural network capable of visual-pattern,» *Neural Networks,* cilt 1, pp. 199-130, 1998.

[4] E. A. v. F. G. H.Levent Akın, «Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümünde Yapay Sinir Ağları Üstüne Yapılan Araştırmalar,» *EMO,* 1993.

[5] Y.LeCun, «Backpropagation applied to handwritten zip code recognition,» *Neural Computation,* cilt 1, pp. 541-551, 1989.

[6] M. Sarıcaoğlu, «Yapay Sinir Ağlarıyla El Yazısının Dilimlenmesi Ve Karakterlerin Tanımlanması,» Trabzon, 1996.

[7] S. Juntanasub. R, «Car license plate recognition through Hausdorff distance technique,» %1 içinde *Tools with Artificial Intelligence*, Hong-Kong, 2005.

[8] Barkod Enerji Sistemleri, «parkyazilim.com,» Barkod Enerji Sistemleri, 23 05 2011. [Çevrimiçi]. Available: http://www.parkyazilim.com/PTS-Bilgiler.html. [Erişildi: 23 01 2015].

[9] C. J. Setchell, «Applications of Computer Vision to Road-Traffic,» University of Bristol, Bristol UK., 1997.

[10] R. Hausle, «The Promise Of Automatic Vehicle identification,» *Transactions on Vehicular Technology,* cilt 26, no. 1, pp. 30-388, 1977.

[11] M. Fahmy, *Computer Vision Application to Automatic Number-Plate Recognition,* Aachen-Germany: In Proceedings of 26th. International Symposium on Automotive Technology and Automation, 1993.

[12] G. P. D. P. J. M. M. I. M. D. N. H. S. B. Auty, *An Image Acquisition System for Traffic Monitoring Applications,* California: Cameras and Systems for Electronic Photography and Scientific Imaging, 1995.

[13] J. R. D. E. L. B.-C. J. Barroso, *Number Plate Reading Using Computer Vision,* Julho, Portugal: IEEE-International Symposium on Industrial Electronics ISIE'97, Universidade do Minho, 1997.

[14] C. Setchell, *Application of Computer Vision to Road-Traffic Monitoring,* Bristol, UK: University of Bristol, 1997.

[15] U. Ç. Mustafa Oral, «Motorlu Araç Plaka Görüntülerinden Karakter Ayrıştırma Ve Tanıma,» %1 içinde *IJCI Proceedings of International Conference on Signal Processing*, Hatay, 2003.

[16] A. BAKKALOĞLU, «ARAÇ PLAKA TANIMA SĠSTEMĠ,» T.C. Selçuk Üniversitesi, Konya, 2011.

[17] Y. D. D. B. Bayram, «Sayısal Görüntü Işleme,» Yıldız Teknik Üniversitesi, Istanbul.

[18] B. Yalım, «Türk Sivil Plaka Standartları İçin Araç Plaka Tanıma Sistemi,» Gazi Üniversitesi, Ankara, 2008.

[19] G. M. Kahraman F., *GABOR Süzgeçler Kullanılarak Taşıt Plakalarının Yerinin Saptanması,* İstanbul: 11. sinyal İşleme ve İletişim Uygulamaları Kurultayı, 2003.

[20] A. C. S. Muhammet BALCILAR, «Geometrik Düzeltme ve Gabor Filtreleriyle Araç Plaka Tespiti,» Yıldız Teknik Üniversitesi, İstanbul.

[21] M. F. Ç. M. M. Halime BOZTOPRAK, «Alternatif Morfolojik Bir Yöntemle Plaka Yerini Saptama,» Süleyman Demirel Üniversitesi, Isparta.

[22] V. G. S. Kamat, «An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S,» %1 içinde *Real-time Technology and Applications, Symposiumı*, 1995.

[23] M. G. Fatih Kahraman, *GABOR SÜZGEÇLER KULLANILARAK TAŞIT PLAKALARININ YERİNİN SAPTANMASI,* İstanbul: 11.Sinyal İşleme Ve Iletişim Uygulamaları Kurultayı, 2003.

[24] M. G. Cinsdikici, «Akıllı Trafik Sistemleri İçin Araç Plakası Tanıma Modülü Tasarımı ve Geliştirilmesi,» Ege Üniversitesi Fen Bilimler Enstitüsü, İzmir, 2004.

[25] Y. D. D. A. Kızılkaya, «Görüntü Bölütleme,» Pamukkale Üniversitesi, Denizli, 2008.

[26] C. I. T. E. K. K. Sinan YILDIRIM, «Görüntü Işleme,» Ege Üniversitesi, Izmir, 2003.

[27] H. Atasoy, «Programlama Günlüğü,» atasoyweb, 2008. [Çevrimiçi]. Available: http://www.atasoyweb.net/Arac-Plakasi-Konum-Tespiti. [Erişildi: 25 02 2015].

[28] S. Fernando, «OpennCV Tutorial C++,» [Çevrimiçi]. Available: http://opencv-srf.blogspot.com.tr/2013/10/smooth-images.html. [Erişildi: 23 02 2015].

[29] Y. D. D. İ. Aydın, «Komşuluk İlişkili İşlemleri-Bölgesel İşlemler-Uzaysal Filtreleme,» Elazığ Üniversitesi, Elazığ, 2013.

[30] T. Şusur, «Order-Statistical Filter».

[31] J. J. G. L. J. v. V. Ian T. Young, «Fundamentals of Image Processing,» Delft University of Technology , Delft.

[32] P. D. V. V. NABİYEV, Yapay Zeka, Ankara: Seçkin Yayıncılık, 2010.

[33] C. E. L. G. O. G. Parisi R., «Car Plate Recognition by Neural Networks and Image Processing,» %1 içinde *IEEE Proceedings of Int. Symposium on Circuits and Systems*, Çin, 1998.

[34] A. V. Ilya Kozhenkov, «AZOFT,» 29 05 2014. [Çevrimiçi]. Available: http://rnd.azoft.com/instant-license-plate-recognition-in-ios-apps/. [Erişildi: 30 05 2015].

[35] L. Işıkdoğan, «Işıkdoğan,» 11 12 2010. [Çevrimiçi]. Available: http://www.isikdogan.com/turkce-blog/hough-donusumu-ile-dairesel-sekil-tespiti.html. [Erişildi: 29 04 2015].

[36] A. C. Y. Y. Ö. Muhammed CİNSDİKİCİ, *PARTITIONED GENERALIZED EUCLIDEAN DISTANCE,* Antalya: The Proceedings of the Twelfth International Symposium on Computer and Information Sciences, 1997.

[37] K. Tsiri, «Number Plate Recognation System,» University of the Western Cape, Africa, 2015.

[38] E. Aksoy, «LICENSE PLATE SEGMENTATION IN IMAGE,» Dokuz Eylul Üniversitesi, İzmir, 2008.

[39] J. H. F. Z.Q.S. Shi, *Method of License PlateLocation Based on Corner Feature,* Dalian: Proc. IEEE 6thWorld Congress on Intelligent Control and Automation, 2006.

[40] I. ARI, «Matlab ile imge işlemeye giriş,» 2008.

[41] Harran üniversitesi, «Görüntü işleme,» Şanlıurfa.

ÖZGEÇMİŞ

Ad-Soyad **:** Günay Musayeva

Doğum Tarihi Ve Yeri : 25.12.1990 Azerbaycan

E-Posta **:** mrsgunay@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2008-2012, Azerbaycan Devlet Neft Akademisi, Bilgisayar Matematiği
- **Yüksek Lisans :** İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği