T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ İŞLEME YÖNTEMLERİYLE OPTİK TARAMA

YÜKSEK LİSANS TEZİ

Berkay ÇETİN

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. Ümit KOCABIÇAK**

Mayıs 2022

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Berkay ÇETİN

15.05.2022

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Prof. Ümit KOCABIÇAK'a teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

CNN	: Evrişimli sinir ağı (Convolutional neural network)
OCR	: Optik karakter tanımlama (Optical character recognition)
OS	: İşletim sistemi (Operating system)
RGB	: Kırmızı-yeşil-mavi renk modeli (Red-green-blue color model)
ReLU	: Doğrultulmuş lineer birim (Rectified linear unit / rectifier activation function)
CLR	: Common language runtime

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: Optik işaret okuma, Görüntü işleme, Derin öğrenme, Evrişimli sinir ağları

Optik işaret okuma çeşitli form veya görüntüler üzerinde tanımlanmış konumlara yapılan işaretlemeleri ve karakterleri, bir bilgisayar sistemine veri girme yöntemidir. Bu yöntem sayesinde kısa süre içerisinde çok sayıda veri toplanabilir. Başlıca soru cevapları, anketler ve dökümanlar gibi kişilerin işaretleme veya yazı ile cevapladığı formlar üzerinde kullanılabilir.

Optik işaret okuma, günümüzde görüntü işleme tekniklerinin gelişmesi ile kullanım açısından daha kolay ve daha kapsamlı veriler üretebilir hale gelmiştir. Ayrıca derin öğrenme yöntemleri sayesinde de oluşturulan evrişimli sinir ağı üzerinden karakter tanıyarak el yazısı ile sağlanan veriler işlenebilir.

Geliştirilmiş olan bu projede cevap kağıdı düzeni, sıkça kullanılan, gereksinimlere uygun şekilde ve sonradan düzenlenebilir olarak tasarlanmıştır. Cevap kağıdı üzerinde işaretlenecek bölgeler veri karmaşasını ortadan kaldırmak için gruplandırılmıştır. Herhangi bir tarayıcı yardımıyla sisteme taratılan cevap kağıdı üzerinde görüntü işleme yöntemleri ve yapay sinir ağları kullanılarak gruplandırılmış alanlar üzerinde gerçekleştirilen işlemler sonucu, sistem verileri çıktı olarak aktarmaktadır.

Bu proje temelde klasik optik okuma işlemi yerine görüntü işleme ve derin öğrenme yardımıyla kullanıcıdan istenen soru sayısı, boş alan bırakma ve doldurulacak alanları tanımlama gibi bilgileri yazılımın kendisi algılayarak kullanıcı bağımlılığını minimuma indirgeyerek gerçekleştirir, böylelikle farklı dizayndaki cevap kağıtları girilse de hızlı bir şekilde kullanıcıya veri sağlayabilir. Ayrıyeten yapay sinir ağı kullanılarak, girilen karakterlerin algılanmasıyla klasik olarak yapılan sınavlarında okunabilmesine olanak sağlar ve daha düzenlenebilir formları tanıtma gereği duymadan esnek bir okuma geliştirir.

ANSWER SHEET DETECTION WITH IMAGE PROCESSING METHODS

SUMMARY

Keywords: Optical mark reading, Image processing, Deep learning, Convolutional neural networks

Optical mark reading is a method of entering data into a computer system, marking the characters and markings made at defined positions on various forms or images. Thanks to this method, a large amount of data can be collected in a short time. It can be used on forms that people answer by markup or text, such as answers sheets, surveys and documents.

Optical mark reading has become easier to use and can produce more comprehensive data with the development of image processing techniques today. In addition, handwritten data can be processed by recognizing characters through the convolutional neural network created by deep learning methods.

In this developed project, the answer sheet layout is designed as frequently used, in accordance with the requirements and can be edited later. The regions to be marked on the answer sheet are grouped to eliminate data confusion. The system outputs the data as a result of the operations performed on the grouped fields using image processing methods and artificial neural networks on the answer sheet scanned to the system with the help of any scanner.

This project basically realizes the information such as the number of questions requested from the user, leaving empty spaces and defining the fields to be filled with the help of image processing and deep learning instead of the classical optical reading process, minimizing the user dependency by detecting the information such as the software itself. In addition, by using the artificial neural network, it allows the entered characters to be read in classical exams by detecting them and develops a flexible reading without the need to introduce more editable forms.

BÖLÜM 1. GİRİŞ

Optik işaret okuma, anketler ve testler gibi belge formları üzerine işaretlenmiş verileri yakalama işlemidir. Optik işaret tanımanın en yaygın kullanımı, insanların çoktan seçmeli bir sınava veya bir ankete doldurdukları cevapları dijital olarak işlemek, depolamak ve cevapların kontrolü içindir. İşaretleme yapılması için önceden belirlenmiş kriterlere sahip cevap kağıtlarına veya formlara cevaplar işaretlenir ve bu formlar daha sonra bir tarayıcı yardımıyla taratılarak sisteme veri girişi sağlanır. Bu veri üzerinden ise yapılan işlemler sayesinde cevaplar kullanıcıya sunulur.

Geliştirilen optik okuma yazılımındaki görüntü işleme yöntemleri sayesinde istenilen sayfa düzeni cevap alanlarının gruplandırılmasıyla sağlanabilir ve böylelikle tek bir kalıp ile sınırlandırılmamış, farklı alanlarda kullanılabilecek bir form oluşturulmasına olanak sağlar.

Bu projede klasik optik okumaya ek olarak aynı form içerisinde kişilerden işaretleme alanının yanında el yazısı ile doldurmaları istenen bölümlerde eklenmiştir, böylelikle klasik sorularında cevaplanmasına olanak sağlanmıştır. El yazısı tanıma alanındaki araştırmalar, derin öğrenme tekniklerine odaklanmıştır ve son birkaç yılda çığır açan bir performans elde edilmiştir. Konvolüsyonel veya evrişimsel sinir ağları (CNN'ler), farklı özelliklerin otomatik olarak çıkarılmasına yardımcı olacak şekilde el yazısı karakterlerin yapısını algılamada çok etkilidir ve CNN'yi el yazısı tanıma problemlerini çözmek için en uygun yaklaşım haline getirir (Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020)).

BÖLÜM 2. KAYNAK ARAŞTIRMASI

2.1. C# Dili

C#, güçlü yazım, bildirimsel, verimli, sınıf tabanlı ve bileşen yönelimli programlama disiplinlerini kapsayan çok paradigmalı, nesne yönelimli bir programlama dilidir. .NET'te çalışan birçok türde güvenli ve sağlam uygulama oluşturmasına olanak tanır. C#'ın kökleri C dil ailesindedir ve C, C++, Java ve JavaScript dillerine benzerdir (https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/). C# 2000 yılında Microsoft tarafından .NET Framework'ün bir parçası olarak ilk tanılıtıldığında kapalı kaynak kodlu bir dildi ve 2004 yılında Microsoftun bir çalışması ile C# çoklu platform derleyicisi ve açık kaynak kodlu kütüphaneler sağlandı.

C# tanımı geliştikçe, tasarımında kullanılan hedefler aşağıdaki gibidir:

- C#'ın basit, modern, genel amaçlı, nesne yönelimli bir programlama dili olması amaçlanmıştır.

- Dil ve uygulamaları, güçlü tip denetimi, dizi sınırları denetimi, başlatılmamış değişkenleri kullanma girişimlerinin algılanması ve otomatik çöp toplama gibi yazılım mühendisliği ilkeleri için destek sağlamalıdır. Yazılımın sağlamlığı, dayanıklılığı ve programcı üretkenliği önemlidir.

- Dil, dağıtılmış ortamlarda dağıtıma uygun yazılım bileşenlerinin geliştirilmesinde kullanılmak üzere tasarlanmıştır.

- Programcı taşınabilirliği kadar kaynak kodu taşınabilirliği de çok önemlidir, özellikle C ve C++'a zaten aşina olan programcılar için.

- Uluslararasılaşmanın desteklenmesi çok önemlidir.

- C#, karmaşık işletim sistemleri kullanan çok büyük sistemlerden özel işlevlere sahip çok küçük sistemlere kadar, hem barındırılan hem de gömülü sistemler için uygulama yazmak için uygun olmayı amaçlamaktadır.

- C# uygulamalarının bellek ve işlem gücü gereksinimleri açısından ekonomik olması amaçlanmış olsa da, dilin performans ve boyut açısından C veya Assembly dili ile doğrudan rekabet etmesi amaçlanmamıştır (C# Language Specification https://www.ecma-international.org/publications-and-standards/standards/ecma-334/).

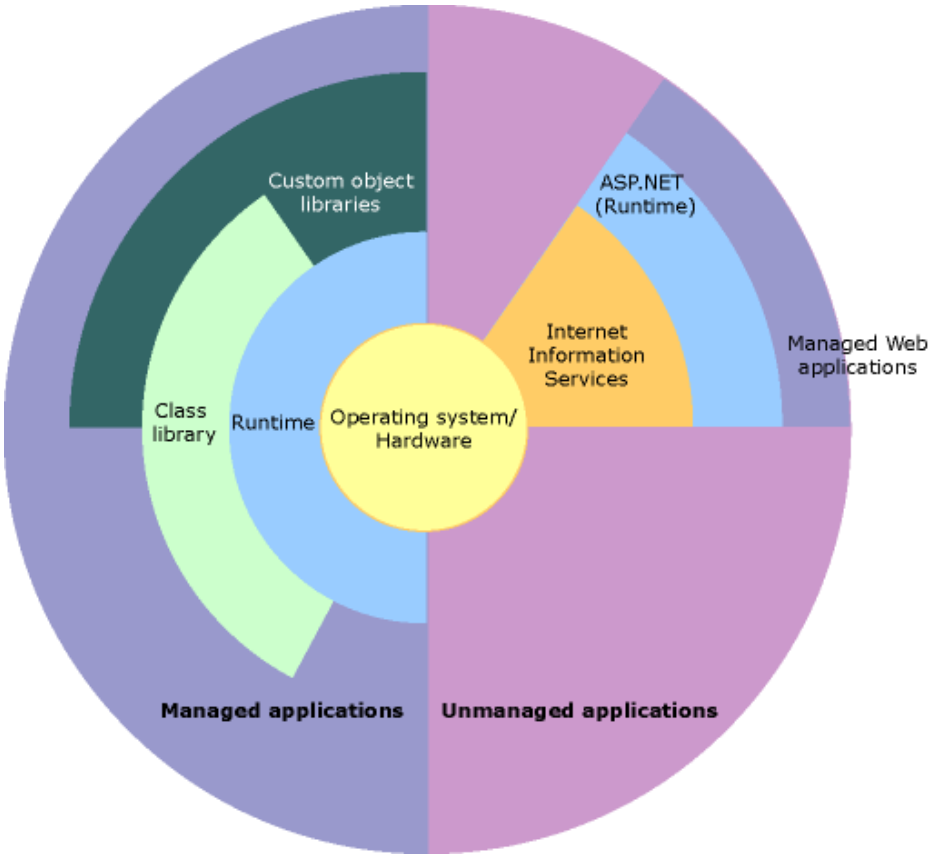
2.2. .NET Framework

Bir yazılım frameworkü olan .NET, Microsoft tarafından geliştirilmiştir, Windows uygulamaları ve web hizmetleri oluşturmayı ve çalıştırmayı destekleyen bir teknolojidir. .NET Framework için yazılan programlar, Common Language Runtime adlı bir yazılım ortamında yürütülür ve CLR güvenlik, bellek yönetimi ve özel durum işleme gibi hizmetler sağlayan bir uygulama sanal makinesidir (https://en.wikipedia.org/wiki/.NET_Framework#Architecture).

Common Language Runtime, yürütme sırasında kodu yöneten, bellek yönetimi, iş parçacığı yönetimi ve uzaktan iletişim gibi temel hizmetler sağlayan ve aynı zamanda katı tür güvenliğini ve güvenliği ve sağlamlığı destekleyen diğer kod doğruluğu biçimlerini uygulayan bir aracı olarak düşünülebilir. Çalışma zamanını hedefleyen kod, yönetilen kod olarak bilinirken, çalışma zamanını hedeflemeyen kod,

yönetilmeyen kod olarak bilinir. .NET Framework, süreçlerine ortak dil çalışma zamanını yükleyen ve yönetilen kodun yürütülmesini başlatan, böylece hem yönetilen hem de yönetilmeyen özelliklerden yararlanan bir yazılım ortamı oluşturan yönetilmeyen bileşenler tarafından barındırılabilir (https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview).

Şekil 2.1.'deki çizim, Common Language Runtime ve sınıf kitaplığının uygulamalar ve genel sistemle ilişkisini gösterir. Çizim ayrıca yönetilen kodun daha büyük bir mimaride nasıl çalıştığını gösterir.



Şekil 2.1. CLR çalışma şeması

2.3. AForge.NET

AForge.NET, C/C++ ile yazılmış bir kütüphanedir. .NET Framework için geliştirilmiş bir bilgisayar görüşü ve yapay zeka kitaplığıdır (https://www.nuget.org/profiles/aforge.net). AForge.NET, Bilgisayarla Görme ve Yapay Zeka, görüntü işleme, sinir ağları, genetik algoritmalar, bulanık mantık, makine öğrenimi, robotik vb. alanlarındaki geliştiriciler ve araştırmacılar için tasarlanmış açık kaynaklı bir çerçevedir.

AForge.NET kitaplığının ana bileşenleri arasından bu projede kullanılmış olanlar aşağıda sunulmuştur:

- AForge.Imaging: Bu kütüphane resimler üzerinde çalışmayı ve çeşitli filtreleri kullanmayı kolaylaştırmaktadır.

- AForge.Math: Sayısal matris ayrıştırma, sınırlı ve sınırsız problemler için sayısal optimizasyon algoritmaları, özel işlevler ve bilimsel uygulamalar için diğer araçlarla birlikte bir matris uzantıları kitaplığı içerir.

Çerçevenin şu ana kadarki en büyük kütüphanesi olan AForge.Imaging, bazı bilgisayarlı görme görevlerinde olduğu gibi görüntü iyileştirme/işlemede olduğu gibi yardımcı olması amaçlanan farklı görüntü işleme rutinleri içermektedir.

2.4. EmguCV

Emgu CV, Çapraz platform olarak OpenCV görüntü işleme kitaplığına .NET uyumlu dillerden çağrılmasına izin verir. Visual Studio ve Unity tarafından derlenebilir, Windows, Linux, Mac OS, iOS ve Android üzerinde çalışabilir (https://www.emgu.com/wiki/index.php/Main_Page).

EmguCV başlıca sağladığı avantaj hazırlanan kodun rahatlıkla platformlar arası kullanılabilmesidir. Emgu CV, C#, VB.NET, C++ ve IronPython dahil olmak üzere birçok farklı dilde kullanılabilir. Emgu CV tamamen C# ile yazılmıştır. iOS, Android, Mac OS X, Linux ve windows dahil olmak üzere herhangi bir platform .NET desteğinde çalıştırılabilir. Ayrıca yazılan kod platformlar arasıdır (https://www.emgu.com/wiki/index.php/Main_Page).

2.5. Tesseract

Tesseract, çeşitli işletim sistemleri için bir optik karakter tanıma motorudur. Tesseract özelleştirilebilir ve çoğu dili destekleyen bir veri setine sahiptir. Bu projede de bu veri setlerinden yararlanılmış ve hali hazırdaki setler tekrardan kullanılarak rakam ve harfler olmak üzere farklı alanlarda kullanılmak üzere ayrıştırılmıştır (https://github.com/tesseract-ocr/tesseract).

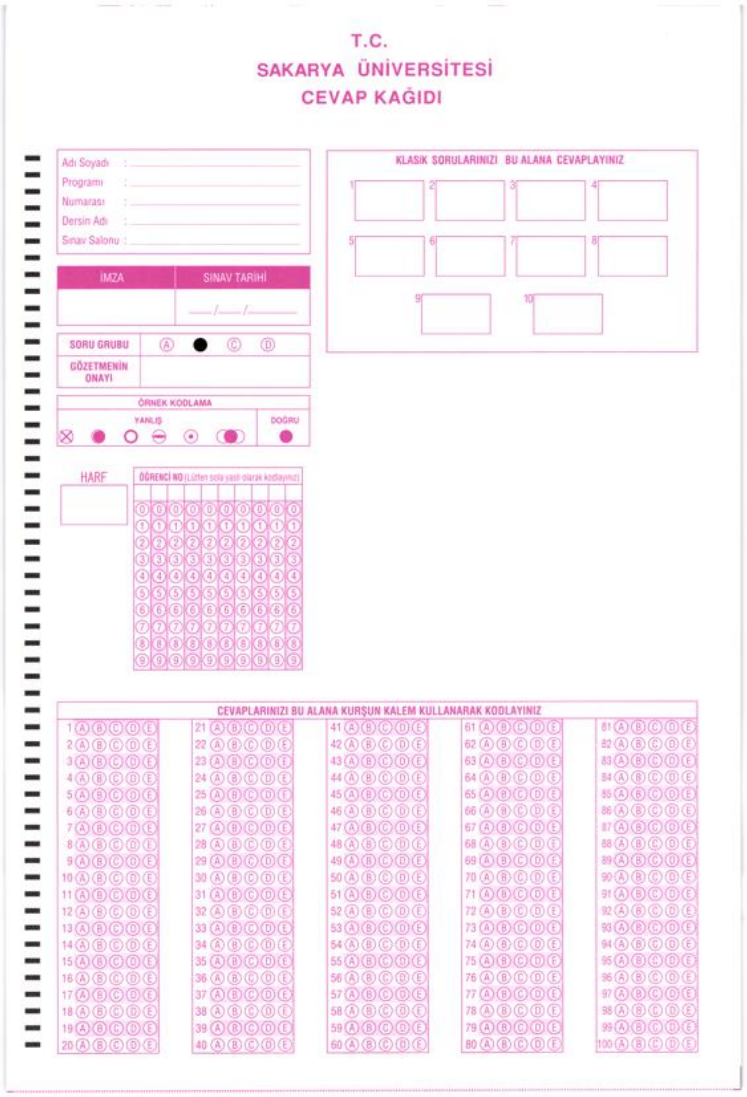
2.6. Evrişimli sinir ağları

Evrişimsel sinir ağları, derin öğrenmede görsel işlemler için en yaygın kullanına yapay sinir ağıdır.

CNN, evrişim katmanları, havuz katmanları ve tam bağlantılı katmanlar gibi çoklu yapı taşlarını kullanarak geri yayılım yoluyla özelliklerin uzamsal hiyerarşilerini otomatik ve uyarlanabilir bir şekilde öğrenmek için tasarlanmıştır (Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi 2018).

BÖLÜM 3. MATERYAL VE YÖNTEM

3.1. Materyal



Şekil 3.1. Örnek Optik Form

Şekil 3.1.'de gösterildiği gibi öğrenci numarası, test cevap işaretleme alanı, soru grubu ve klasik cevap alanı olan örnek bir cevap formu kullanılmaktadır. Bu cevap formu alanların yerleri değiştirilerek düzenlenebilir ve doldurma yapılan alanlar haricindeki alanlar çıkartılarak yeni alanlarda eklenebilir. Bu işlem yapılırken belirtilen doldurma alanları dışındaki alanlarda işlem yapılmadığı için yeni alanların eklenmesi işlem üzerinde ciddi bir maliyet yaratmayacaktır ve isteğe uygun esnek bir form düzenlemesi seçeneği sunmaktadır.

Doldurulan formun üzerinde işlem yapılabilmesi için dijital ortama aktarılması gerekmektedir ve taratılarak okunaklı bir kalitede bilgisayar ortamına geçirilmelidir, bu işlem bir tarayıcı yardımıyla gerçekleştirilebilir.

3.2. Yöntem

3.2.1. Filtreleme

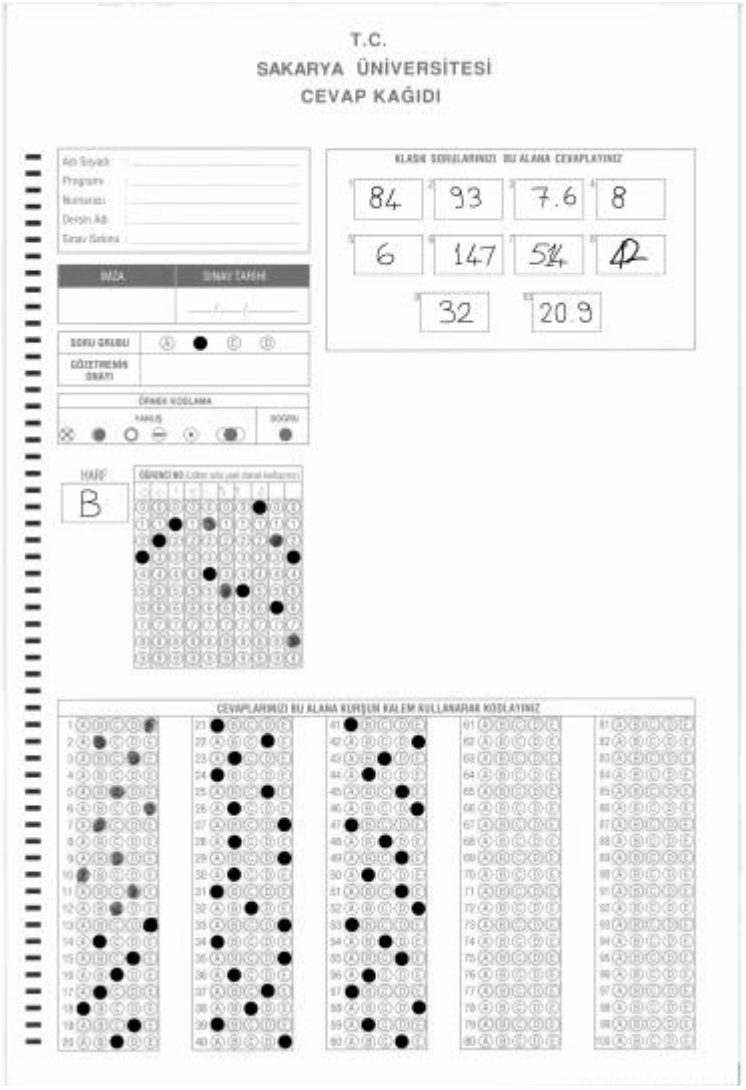
Görüntüler piksel adı verilen küçük adreslenebilir birimlerden oluşmaktadır. Piksellerin konum ve renk yoğunluğu değerleri vardır ve bu değerleri içeren bir matrsite sıralanarak görüntüyü oluştururlar. Kırmızı, yeşil ve mavi olmak üzere 3 kanala sahip olan pikseller bu kanalların değerlerine göre sahip oldukları renkler belirlenir. 1x3 vektörü olarak temsil edilir ve her 3 rengin 0'dan 255'e kadar tamsayı değerleri bulunur. Bu değer aralığına göre toplam 256^3 yani 16777216 farklı renk seçeneği bulunmaktadır (https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html).

Görüntü filtreleme bir görüntünün piksel konumlarına dokunmadan renk değerleri üzerinde değişiklikler yapılmasıdır. Görsel üzerinde işlemler gerçekleştirilmeden önce filtreleme işlemleri uygulanmaktadır. Filtreleme işlemleri görüntü işleme işlemleri öncesinde kirliliği azaltmak, görselin genişlik ve yüksekliğini değiştirmek ve gereksiz renklerden kurtularak görselin dosya boyutunu azalmasını sağlayarak işlem hızını arttırmak için uygulanır.

3.2.1.1. Gri tonlama

İlk olarak taratılmış olan optik gri tonlamalı bir görüntüye dönüştürülür, her pikselin değerinin yalnızca bir ışık miktarını temsil eden tek bir örnek olduğu görüntüdür; yani sadece yoğunluk bilgisi taşır. Bir siyah beyaz veya gri monokrom olan gri tonlamalı görüntüler, yalnızca gri tonlarından oluşur. Kontrast, en zayıf yoğunlukta siyahtan, en güçlü beyaza kadar değişir (Johnson, Stephen; Stephen Johnson on Digital Photography, 2006). Gri tonlamalı görüntülerde, iki tonlu siyah ve beyaz görüntülerden farklı olarak gri tonlarıda bulunmaktadır. Bir pikselin yoğunluğu, 0 ile 1 arasında belirli bir yoğunluğu ifade edecek şekildedir. Bu aralık, siyah için 0 ve beyaz için 1 değerleri arasında herhangi bir kesirli değerle bir aralık olarak soyut bir şekilde temsil edilir (https://en.wikipedia.org/wiki/Grayscale).

Şekil 3.2.'de gösterildiği gibi gri tonlama filtresi kullanılarak RGB (Kırmızı-Yeşil-Mavi) katsayıları değiştirilerek gri tonlanmış hale getirilir.

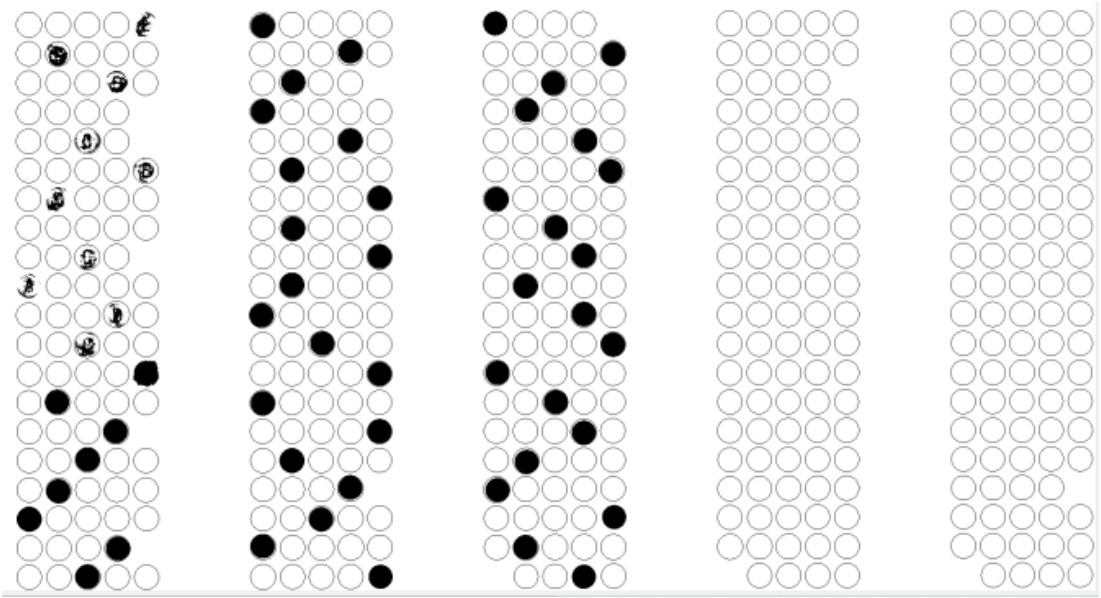


Şekil 3.2. Gri filtreleme uygulanmış optik

3.2.1.2. Eşikleme

Ardından belirtilen eşik (Treshold) değerini kullanarak görüntü ikilileştirme yapılır. Gri tonlamalı bir görüntüden, ikili görüntüler oluşturmak için bu teknik kullanılır. Yoğunluğu eşik değerine eşit veya daha yüksek olan tüm pikseller beyaz piksellere dönüştürülür; yoğunluğu eşik değerinin altında Şekil 3.3. test alanı örneğindeki gibi tüm pikseller yoğunluk oranlarına göre siyah veya beyaz piksellere dönüştürülür. Eşik

filtresi kirliliğin az olduğu, aynı gruptaki piksellerin yoğunluğunun diğer gruplanmış piksellere oranla daha yakın olduğu ve ışığın homojen olduğu görünütülerde daha iyi sonuçlar verir ve aksi durumda yanlış yoğunluk grupları üzerinden hatalı bir ikileme oluşturacaktır.



Şekil 3.3. Eşik değeri uygulanmış test alanı

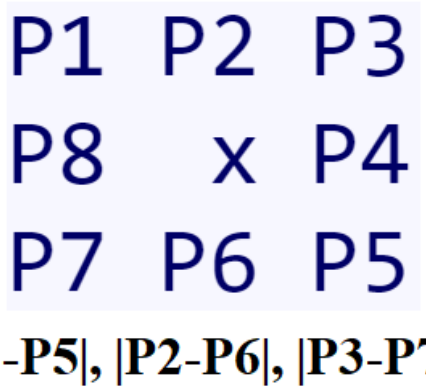
3.2.2. Görsel üzerinde açısal düzeltme

3.2.2.1. Kenar bulma

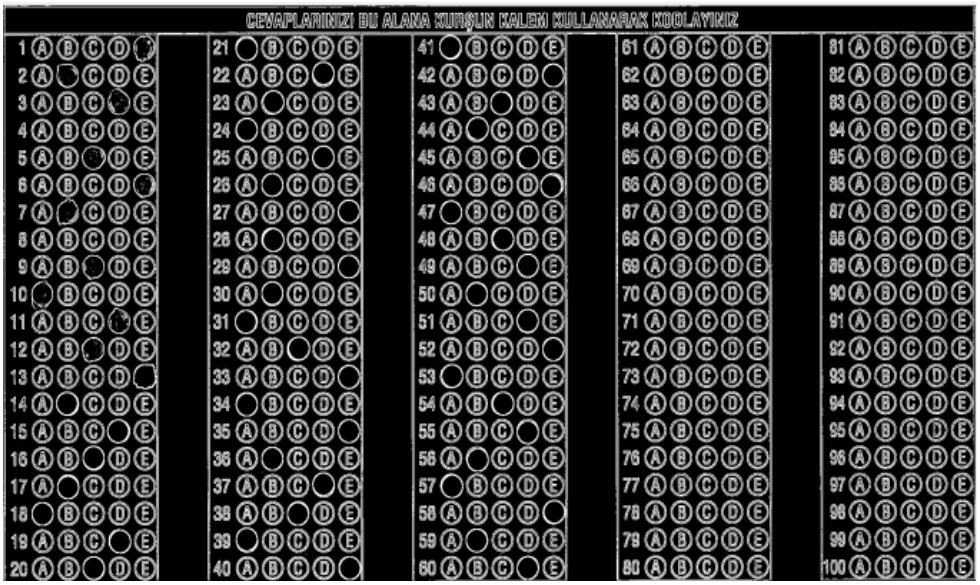
Görsel içerisindeki piksellerin yoğunluklu olarak birleşmesi ve iki nokta arasında bağlantı oluşturmasına kenar denir. Kenarlar filtre uygulanmış görsel içerisinde keskin ayrımlar oluştururlar.

Görüntü işleme teknikleri arasında yer alan kenar bulma, görselin sınırlarının belirlenmesi olarak tanımlanır. Dijital bir görselde piksel yoğunluğunun keskin değişimler yaşadığı alanlarda görselin parlaklığının süreksizlik bölgelerine göre ayırma yöntemine kenar bulma yöntemi denir.

AForge.NET içerisindeki DifferenceEdgeDetector fonksiyonu ile pikseller üzerindeki yerel maksimum işlemi ile pikselin çevresindeki 4 yönde pikseller arasındaki maksimum fark hesaplanarak nesnelerin kenarları bulunur (http://www.aforgenet.com/framework/docs/html/d0eb5827-33e6-c8bb-8a62-d6dd3634b0c9.htm). Şekil 3.4.'te gösterildiği gibi seçilen x pikselinin 4 farklı yöndeki bitişik pikselleri üzerinde maksimum bulma işlemi ile kenarlar bulunur ve Şekil 3.5.'te gösterildiği gibi kenarlar etrafında birer çerçeve oluşacak şekilde bir çıktı verir.



Şekil 3.4. maksimum fark hesaplama (http://www.aforgenet.com/framework/docs/html/d0eb5827-33e6-c8bb-8a62-d6dd3634b0c9.htm)[15G]

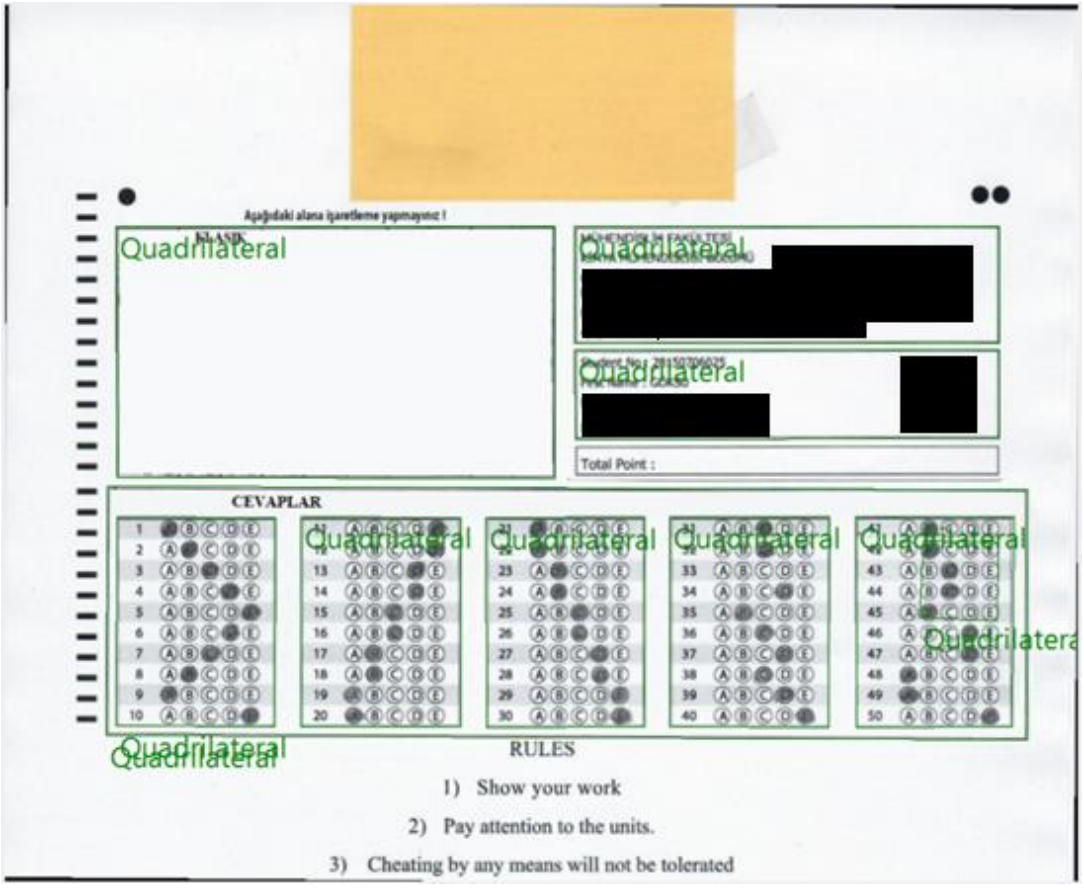


Şekil 3.5. Kenar bulma uygulanmış optik kağıt

3.2.2.2. Şekil bulma

Filtreleme işlemi ve kenar bulma uygulanarak kenar oluşturan piksellerin koordinat noktaları bir dizi içerisinde tutulur. Resim üzerinde AForge BlobCounter sınıfı içindeki bağlı bileşen etiketleme algoritmasını kullanarak görüntülerdeki bağımsız noktaları sayma ve ayıklama işlemi yapılır. Algoritma eşikleme değeri altındaki pikselleri yok sayarak diğer pikseller üzerinden nesneleri tanımlar (http://www.aforgenet.com/framework/docs/html/d7d5c028-7a23-e27d-ffd0-5df57cbd31a6.htm).

Yöntem, ilişkili noktaları bir dizi içerisinde her satırını ve sütununu tarar ve en üst/alt/sol/sağ noktaları bulur. Koordinat değerlerinin bulunduğu dizi içerisinden bu yöntem aracılığıyla koordinat değerleri karşılaştırılarak köşe noktalar bulunarak, köşelerin sayısına göre objenin dörtgen veya daire olduğu bulunur (http://www.aforgenet.com/framework/docs/html/59949f70-2afc-f7a5-1a53-ff99a9208133.htm).



Şekil 3.6. Dörtgen bulma işlemi

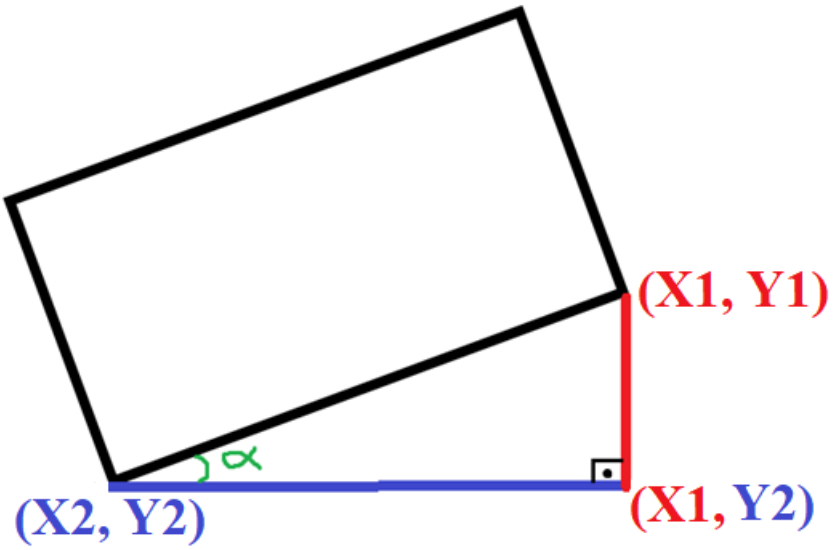
Şekil 3.6.'da gösterildiği gibi piksel kenarları kesişen ve köşe sayısı 4 olan piksel grupları yeşil bir çerçeve içerisine alınarak Dörtgen oldukları görsel üzerine bir string ile yazdırılarak gösterilmiştir.

3.2.2.3. Açı bulma

Taranan optik, dijital resim halinde ister istemez kayma yaşar; sağa veya sola yatık bir şekilde dijitalleşir. Bu kaymalar görsel işlemler yapılırken yanlış koordinat hesaplama ve yanlış bölge tayini gibi hatalar ortaya çıkarır.

Şekil bulma yöntemi sonucunda optik içerisindeki belirlenen dörtgenler içerisinde, ilk dörtgen baz alınarak elde edilmiş olan köşe noktalarının koordinatlarını içeren dizi

içerisinden alt 2 köşe noktası alınarak, y ekseni kıyaslamasıyla taranmış resmin sağa mı yoksa sola mı yatık olduğu belirlenir. Eğer dörtgenin sol alt köşe noktasının y ekseni koordinat değeri, sağ alt köşe noktasının y ekseni koordinat değerinden daha yüksek bir değere sahipse taratılan resim sağ tarafa yatık; tam tersi durumda ise sol tarafa yatık olarak program içerisinde tanımlanır ve işlem bu görselin sağa veya sola yatık olmasına göre devam eder.



Şekil 3.7. İz düşüm üçgeni

Şekil 3.7.'deki gibi görselin içerisinden tespit edilmiş dörtgenin sola yatık olmasına bağlı olarak sağ alt köşe noktasından iz düşüm alınan nokta, iz düşüm alındığı köşe noktasıyla aynı x ekseni koordinat değerine ve diğer alt köşenin y ekseni koordinat değerin sahip olacaktır. Oluşan dik üçgenin her üç köşe noktasının koordinat değerleri böylelikle bulunmuş olur ve üçgenin her iki noktası arasındaki uzaklık matematik fonksiyonları ile hesaplanır. Bu fonksiyon x ve y eksen değerleri bilinen iki nokta arasındaki uzaklığın ölçümünü sağlayarak üçgenin kenar uzunluk değerlerinin bulunmasını sağlar. İz düşüm üçgeni bir dik üçgen olacağı için iz düşüm noktası ve iz düşümün alındığı nokta Şekil 3.7.'de gösterilen (X1, Y2) ve (X1, Y1) iz düşüm olması sebebiyle aynı x ekseni koordinat değerlerini paylaşmaktadırlar dolayısıyla y ekseni

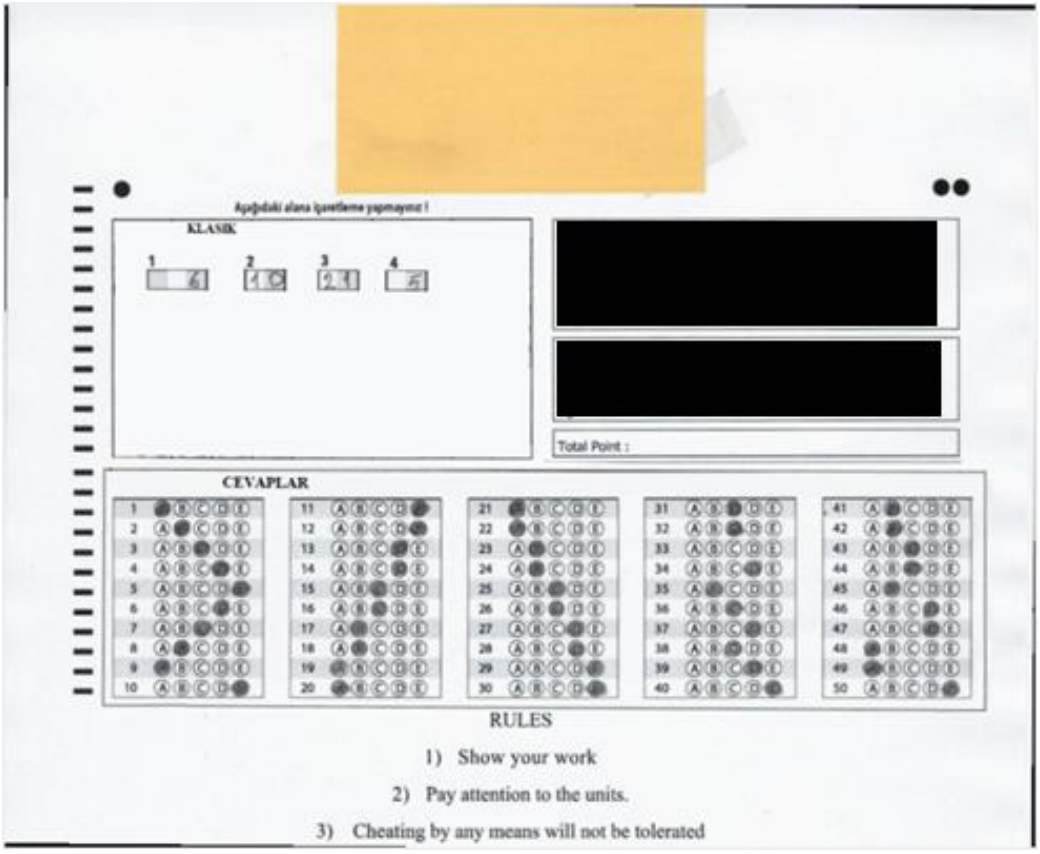
koordinat değerleri farkı bu iki nokta arasındaki uzaklık değerini sağlamış olur. (X2, Y2) ve (X1, Y2) noktaları ise aynı y ekseni koordinat değerlerine sahip olduğu için x eksenlerinin koordinat farkı bu iki nokta arasındaki uzaklık değerinin hesaplanmasını sağlar. Bu işlemler C#.Vector sınıfının içerisindeki lengthSquare fonksiyonu ile belirtilen şekilde noktalar arası fark alma yöntemi ile hesaplanır.

$$\theta = \text{Acos}\left(\frac{a^2 + c^2 - b^2}{2bc}\right)$$

Şekil 3.8. Ark kosinüs formülü

Üç kenarı bilinen üçgenin açısını hesaplamak için Şekil 3.8.'de belirtilen ters trigonometrik bir işlem olan Arkkosinüs formülü kullanılır. Hesaplanmış olan üçgenin kenar değerleri karesi alınarak, önceden belirlenmiş görselin sağa mı yoksa sola mı yatık olması durumuna göre arkkosinüs formülü uygulanır. Bu yöntem uygulanarak elde edilen açı kadar bütün görsel çevirilerek düz hale getirilmiş olur.

Şekil 3.9. Taratılmış görsel



Şekil 3.10. Döndürme işlemi yapılmış görsel

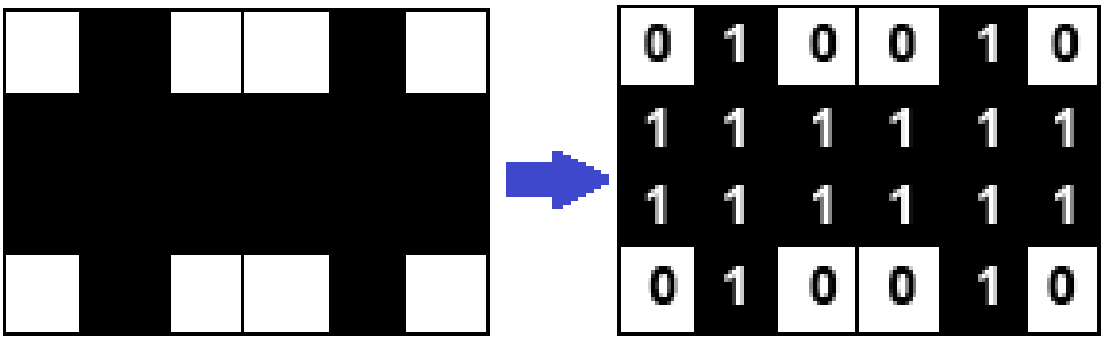
Taratılmış resim Şekil 3.9.'da gösterildiği gibi sola tarafa yatık bir şekilde taratma işlemi uygulanmıştır. Arkkosinüs formülü sonucu elde edilen açı kadar döndürülen bu resim soncunda Şekil 3.10.'da düzleştirilmiş şekilde sonuç verir. Bu döndürme işlemi sonucunda görselin önceden yatık olduğu tarafta siyah alanlar oluşmaktadır. Bu alanlar sonraki işlemler için bir sorun teşkil etmediği için herhangi bir işlem yapılma gereği duyulmamıştır.

3.2.3. Okuma yapılacak alanların belirlenmesi

Okuma yapılacak alanlar taratılmış olan optik kağıdı içerisindeki doldurma işlemi yapılmış alanlardır. Bu doldurma işlemi daire doldurma veya metinsel içerik sağlama şeklindedir. Bu işlemlerin yapıldığı alanlar genellikle optik kağıdı tasarımında belirli

bölgeler olarak ayrılır ve optiği dolduracak kişinin, doldurulacak alanları kolay anlayabilmesi için başlıkla açıklanır. Şekil bulma işlemi ile tanımlanmış olan dörtgenler bir dizi içerisinde koordinatları baz alınarak sıralanır. Bu dizi bulunan dörtgenlerin kırpılmış resimlerini Bitmap şeklinde alt köşe noktalarının koordinat değerleri üzerine sıralayarak saklar.

Bitmap, piksellerinin veya bitlerinin her birinin rengi de dahil olmak üzere bir görüntüleme alanının tanımlandığı nesnedir. Bir bitmap, bir görüntüdeki veya ekrandaki piksellerin değerlerini temsil eden bir ikili veri dizisidir. Görsel bir ekranda görüntülendiğinde, ekranı "boyamak" için hangi renklerin kullanılacağını belirlemek için bitmap'i okur ve okuduğu veriye göre görüntü sağlar (https://www.britannica.com/technology/bitmap). Şekil 3.11.'de ikili bir bitmap grafiği gösterilmiştir. Bitmapler bir veri dizisi oldukları için görsel işlem uygulamalarında sıkça kullanılır.



Şekil 3.11. Bitmap grafiği

Kırpılmış bölgelerden oluşturulmuş bitmapleri içeren dizi içerisinde başlığın olacağı bölge kırpılır ve çerçeve kalmaması için 5 milimetre içeriye doğru kesilir.

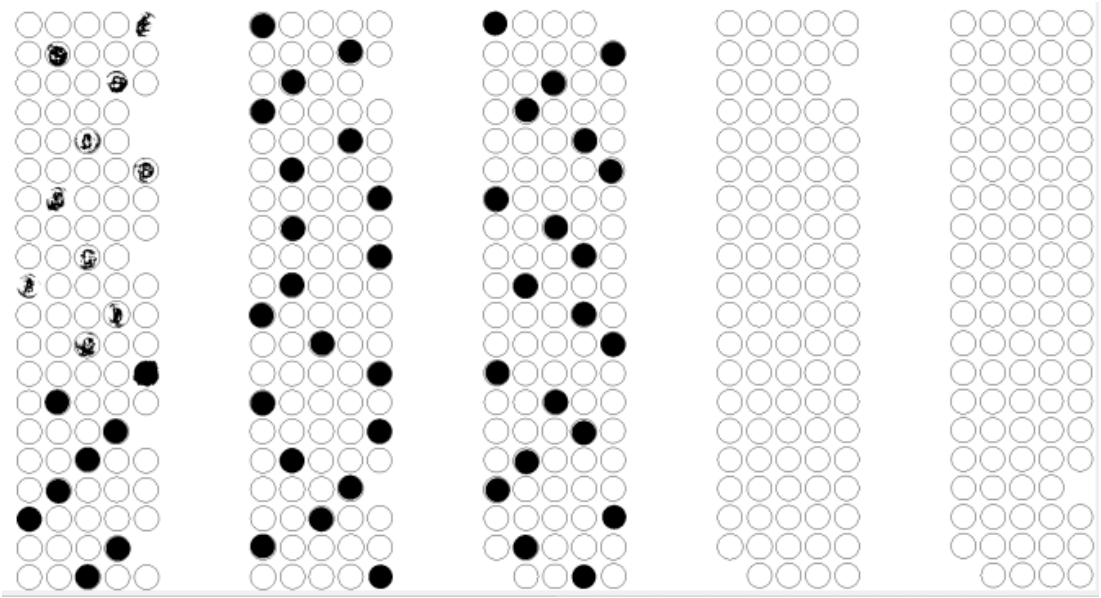
Oluşan bu görsel içerisinde harf algılamak için eğitilmiş veri seti aracılığıyla başlık kontrolü yapılarak Soru Grubu, Öğrenci Numarası, Klasik Sorular Cevap Alanı ve Test Soruları Cevap Alanı olmak üzere tespit edilir.

3.2.4. Doldurulan alanların okunması

3.2.4.1. Daire doldurulan alanların okunması

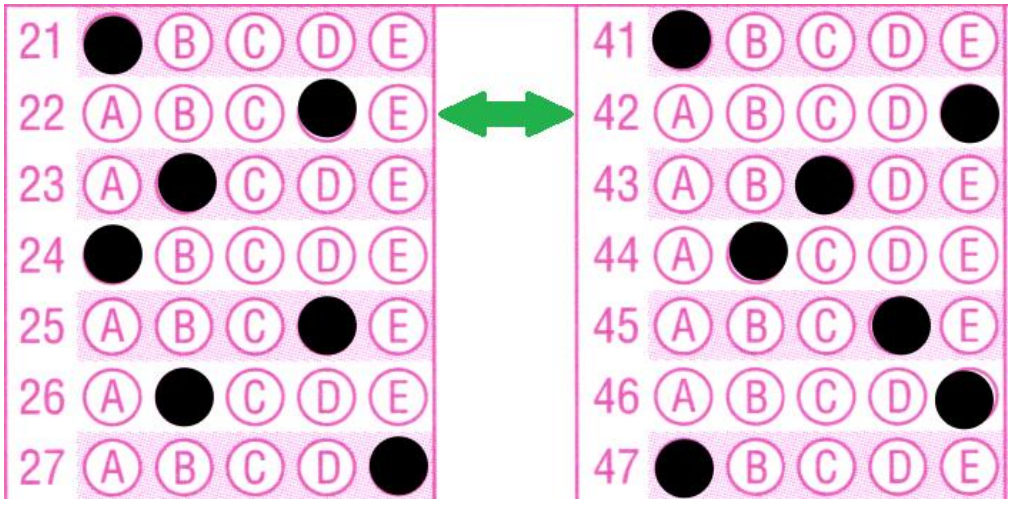
Tespit edilen Soru Grubu, Öğrenci Numarası ve Test Soruları Cevap alanları üzerinde daire doldurma işlemi gerçekleştirilmektedir. Daire doldurulan bu alanlardaki işaretli alanları bulma işlemi aynı şekilde yapılır, sonuçların sınıflandırılması işlem yapılmış alana göre değişiklik gösterir. Soru grubu "A, B, C, D" gibi harf gruplarının tek satırda işaretlenmesi ile doldurulur, öğrenci numarası rakamlardan oluşur yukarıdan aşağıya 10 rakamın ilişkili olan rakamın işaretlenmesiyle sola doğru giderek doldurulur ve test sorularının cevapları için "A, B, C, D, E" şıkları olmak üzere 5 şık olarak gruplandırılmıştır. Soru grubu sayısı 4 farklı grup olacak şekilde gruplandırılmıştır. Öğrenci numarası belirlenen alan kadar veya daha az basamaklı olacak şekilde girilebilir ve numara sola yatık bir şekilde alınır; numaranın daire doldurma işleminin en solda veya boşluk bırakılarak yapılması programda bir sorun teşkil etmez boş basamakları almadan sola yatık bir şekilde öğrenci grubunu okur. Doldurulan alanı belirlendikten sonra test soru sayısının değişkenlik göstermesi programı etkilemez, program test alanındaki toplam daire sayısına bakarak soru sayısını kendisi algılar. Test işaretlemeleri soru sırası tüm test optik kağıtlarında olduğu gib yukarıdan aşağıya ve soldan sağa olacak şekildedir ve 5 farklı şık işaretlemesi yapılacağı varsayılarak okuma yapar.

Filtrelerden geçirilmiş cevap alanı belirlenen Şekil 3.12.' deki gibi treshold (Eşik) değeri uygulanarak bulunan noktaların birleşimi AForge.NET BlobCounter sınıfı aracılığıyla daire şekilleri tespit edilen alanların içerisinde ayrı ayrı bulunur.



Şekil 3.12. Eşikleme uygulanmış test alanı

Eşikleme uygulandıktan sonra Şekil 3.12.'de görülüğü gibi bazı daireler filtreleme sonucu gözükmez ve bu durum soru sayısının hesaplanmasını etkiler. Bu durumda bulunmuş dairelerin merkez noktaları arasında minimum ve maksimum x ve y koordinat değerleri sıralanarak gruplar arası boşluk göz önünde bulundurularak soru sayısı doğrulaması yapılır.



Şekil 3.13. Doldurma alanları arası boşluk

Öğrenci Numarası ve Test Cevapları alanında bulunan dairelerin merkez noktaları baz alınarak tekrardan bir eşikleme uygulanır ve yeni merkez noktaları arasındaki uzaklık kıyaslanarak belirlenen değerden küçük olacak şekilde "A, B, C, D, E" çoktan seçmeli grupları şeklinde 5'e bölünerek toplam soru sayısı, bulunan daire sayısıyla kıyaslanarak yanlış sıralama yapılıp yapılmadığı belirlenir ve aynı zamanda bu teknik sayesinde test soru sayısı belirtilmesine gerek kalmadan dairelerin hepsini hesaba katarak işlem yapılmış olur. Fakat bu işlem cevap kağıdının 5 şıktan oluştuğu varsayılarak yapılmıştır ve daha az veya daha fazla şık içeren cevap formları için doğru sonuç vermeyecektir.

Bulunan her dairenin merkez noktası koordinatları bir liste içerisinde saklanır. Dairelerin x eksen değerlerine göre minimum değerleri alınıp kıyaslanarak Şekil 3.13.'te gösterildiği gibi arada bir boşluk olması durumu incelenir. Bu işlem sonucunda daireler soru sayısına denk gelecek şekilde sıralanır.

Hali hazırda treshold (Eşik) değeri ile filtrelenmiş daireler koordinatları sırasınca bir dizi içerisinde farklı bir fonksiyon ile tekrardan treshold (Eşik) değeri uygulanarak içi doluluk oranına göre belirli bir yüzdeden fazla olması durumunda işaretlenmiş olarak kabul edilir. Dizi içerisinde öncelikle soldan sağa ve yukarıdan aşağıya şeklinde x ve y ekseni değerlerine göre sıralanmış daireler, işaretlenmiş olanlar sıra ve işaretli şıkkı olarak bir string değeri olarak tutulur. İşaretli olan bu cevaplar hangi soru sırasına denk geliyorsa o sayı ile ve hangi şık grubunda ise birleştirilerek soru numarası – cevap şıkkı şeklinde string içerisine yazdırılır.

Öğrenci numarası aynı daire bulma işlemleriyle dairelerin merkez noktalarının kıyaslanması sonucu yerleri saptanır ve yine aynı şekilde boş alanlar değerlendirilmeden dolu daireler soldan sağa sıralanarak karşılık gelen rakamların sıralanması ile öğrenci numarası elde edilmiş olur.

Soru grubu alanı halihazırda 4 grup olduğu varsayılarak tasarlanmıştır ve treshold (Eşik) değeri belirli yüzdenin üzerinde olan dairelerin işaretlenmiş olduğu kabul edilen dairenin merkez noktasının x ekseni üzerinde diğer merkez noktalarıyla sıralanması sonucu elde edilen sıraya göre tutulur.

3.2.4.2. Karakter içeren bölgelerin okunması

Öğrenci numarasının harf kısmı ve klasik soruların bulunduğu doldurulması gereken alanlar, diğer daire doldurma alanlarından farklı olarak karakterler ile cevaplama işlemi yapılır. Bu işlemlerden öğrenci numarasının harf kısmı belirlenmiş olan alanın içerisine harf karakteri yazılarak doldurulur. Klasik soruların cevaplandırılması ise cevap alanının içerisine ayrılmış olan 10 farklı cevap doldurma alanına soruların cevaplarının yazılması ile gerçekleşir; rakamlar ve virgül karakterleri ile doldurulması istenir.

Öğrenci Harf Numarası ve Klasik Soru Cevapları, öncelikle başlık ve alanların çerçeveleri dışarıda kalacak şekilde kırpılır. Bu kırpma işlemi Şekil 3.14.'te gösterildiği gibi 5'er milimetre olacak şekilde resmin kesilmesiyle gerçekleşir. Harf Numarası için başlıkların algılanması için kullanılan harflere ayrılmış veri seti kullanılarak elde edilen harf, daire işaretlenerek elde edilen öğrenci numarasının başına eklenerek öğrenci numarası tamamlanmış olur.

Şekil 3.14. Çerçeve kırpma işlemi uygulanmış harf alanı

Klasik Soru Cevaplama alanı Şekil 3.15.'te gösterildiği gibi alan içerisinde 10 farklı klasik soru cevaplama alanları bulundurur ve karakter ile doldurulması gerekmektedir. Alan içerisindeki ayrı dörtgen alanlar için programın başında kullanılan AForge BlobCounter sınıfı tüm görsel haricinde klasik cevap alanı için kullanılarak dörtgen tespit etme işlemi tekrarlanır ve elde edilen dörtgenler koordinatlarına göre bir dizi içerisine sıralanır. Sıralı dörtgenler çerçeveleri dışarıda kalacak şekilde Şekil 3.16.'da gösterildiği gibi kırpılarak sayılar olarak ayrılmış olan veri seti yardımıyla tanımlanır. Bu süreç esnasında sayılar ve virgül karakterleri tanımlanarak dörtgenlerin dizi içerisindeki sırası baz alınarak içi dolu olan dörtgenlerin sıralamasıyla cevap olarak tutulur. İçi boş bir dörtgen olması durumunda o sorunu boş olduğu belirtmek için soru numarasının yanı boş bırakılır.

String olarak tutulan Soru Grubu, Öğrenci Numarası, Test ve Klasik Soru Cevapları bir bütün şeklinde metin çıktısı olarak arayüzde kullanıcıya sunulur.

Şekil 3.15. Klasik soru doldurma alanı

Şekil 3.16. Çerçeveleri kesilmiş klasik soru cevap alanı

3.2.5. Evrişimli sinir ağı

Evrişimsel sinir ağları, çok sayıda birbirine bağlı nörondan oluşur. Ağ için girdi iki boyutlu resim dizisi şeklindedir. Nöronlar katmanlar halindedir ve girdi katmanı, gizli katmanlar ve çıktı katmanlarından oluşur. Bir katmandaki nöronlar, diğer katmanda bulunan bazı bağlantı kurarak, bias ve ağırlık oluşturarak ilerler. Alt örnekleme veya havuzlama katmanları kullanılarak girdinin boyutlarını azaltma işlemi uygulanabilir.

3.2.5.1. Girdi katmanı

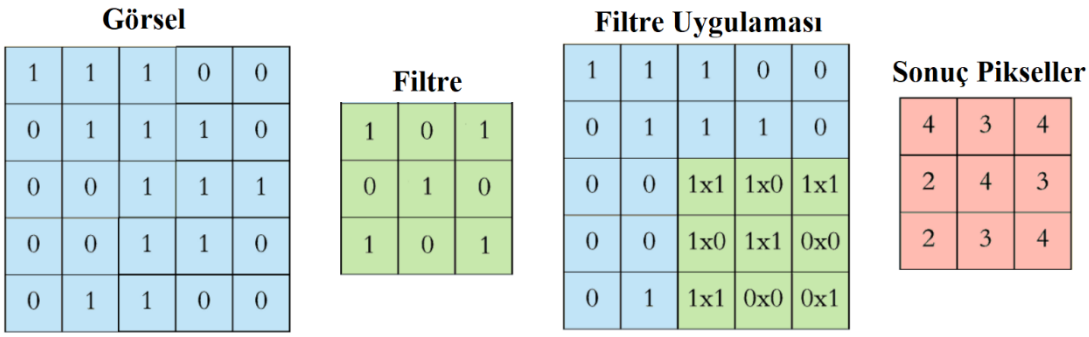
Girdi resmi verileri bir matris şeklinde girdi katmanına yüklenir ve depolanır. Taratılan görselin yüksek, genişlik ve renk değerlerini (RGB bilgisi) bu matris içerisinde tanımlıdır.

3.2.5.2. Gizli katmanlar

Gizli katmanlar, ayırt edici özelliklerin algılandığı evrişim, havuzlama ve aktivasyon işlevlerinin kullanıldığı bir katmandır.

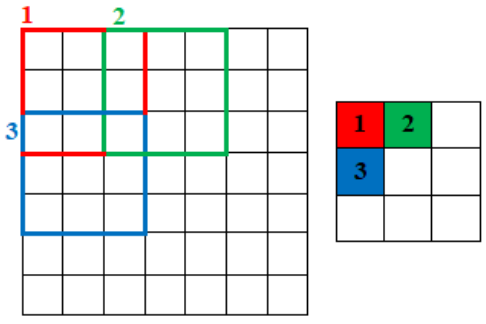
3.2.5.3. Evrişim katmanı

Evrişim katmanında girdi katmanından gelen görselin üzerinde işlem uygulanan ilk katmandır ve girdi katmanındaki nöronlardan gelen yükseklik ve genişlik üzerinde, pikseller arası ilişki kaybı yaşatmadan özellik haritası oluşturmak için filtre uygular. Belirlenen boyutta bir alıcı alan, görsel üzerinde gezdirilerek bu işlemi uygular. 5 × 5 boyutlu görsel üzerinde filtrelemenin görselleştirilmesi ve aktivasyon haritası Şekil 3.17.'de örneklendirilmiştir.



Şekil 3.17. 5×5 görselin 3x3 boyutunda filtrelenmesi (https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/)

Adımlama, CNN içerisinde kullanılan başka yöntemdir. Şekil 3.18.'de örnek gösterildiği gibi filtrenin her seferinde piksel piksel hareket ettiği adım, 7x7 görselin 2'şer birim adımlama yöntemidir.

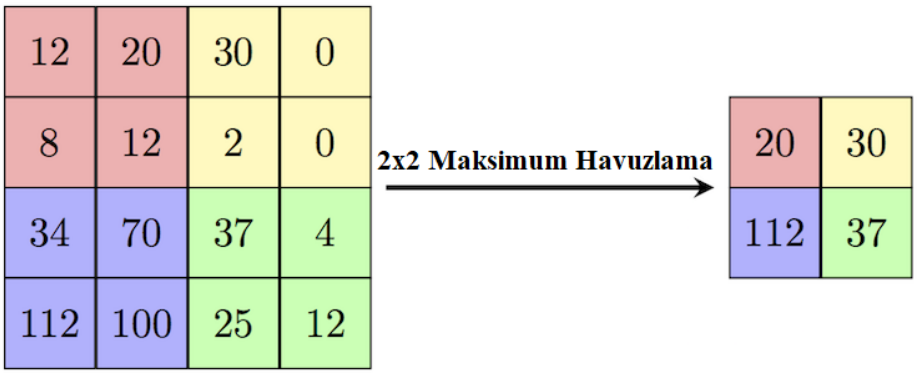


Şekil 3.18 İkili adımlama işlemi (https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/)

Dolgulama işlemi, evrişimli katmandaki çıktının küçültülmesi için kullanılır. Oluşan özellik haritası evrişimsel katmandaki işlemler sonrası köşe piksellerde değer kaybı yaşar, bu nedenle özellik haritasının bozulmaması için matris kenarlarına 0 değerleri eklenir.

3.2.5.4. Havuzlama katmanı

Havuzlama, istenmeyen değerler yerine seçilen değerlerin bir sonraki katmana geçmesine sağlar. Havuzlama katmanı ayrıca özellik seçiminde ve aşırı uyumu kontrol etmede yardımcı olur. Maksimum, minimum veya ortalama gibi filtreleme işlemi uygulanarak filtre sonucu oluşan değerler ile özellik haritası oluşturulur. Şekil 3.19.'daki gibi 2x2 filtre, ikili adımlama ile filtre içerisinde bulunan değerler içerisinde maksimum değer alınarak devam eder. Negatif değerlerden kurtulmak içinse minimum havuzlama kullanılır.

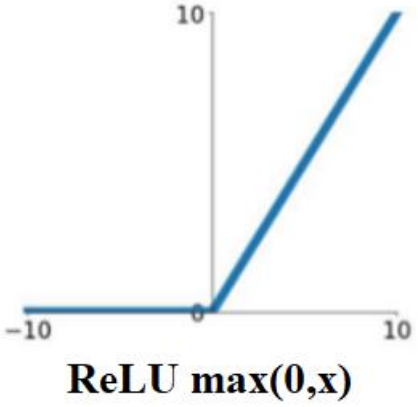


Şekil 3.19. 2x2 Filtre ile Maksimum Havuzlama

3.2.5.5. Aktivasyon katmanı

Aktivasyon işlemi, girdini ağırlık sonucunun düğümler üzerinden bir sonuca ulaşması için bir aktivasyon fonksiyonun kullanılmasıdır ve evrişim katmanlarından sonra

kullanılır. Bu projede negatif değerler için ReLU aktivasyon fonksiyonu kullanılarak kurtulunmuş ve 0'a çevrilmiştir.

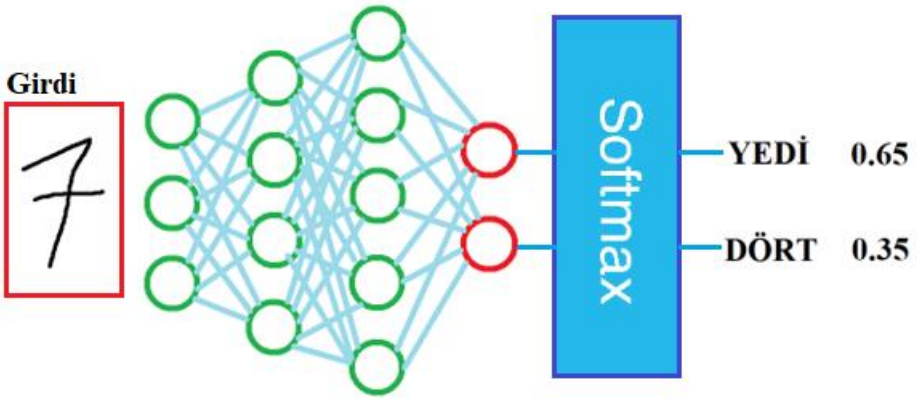


Şekil 3.20. ReLU maksimum sıfırlama fonksiyonu

Havuzlama katmanı, her bir özellik haritasının boyutunu azaltır ve bu da ağdaki hesaplama işlemini azaltır.

3.2.5.6. Tam bağlı katman

Tam bağlı katman, önceki katmanlar tarafından çıkarılan özellik haritaları özelliklerini kullanarak girdi olarak verilen görseli sınıflandırmak veya tahmin etmek için kullanılır. Sınıflandırma problemleri için, tam bağlı katmanı soft-max aktivasyon fonksiyonu Şekil 3.21.'de tasfir edildiği gibi 0 ile 1 arasında oranlar verir, girdi için her sınıfın olasılığını üretir. Oluşan oranlar arasından en yüksek değere sahip olan seçilerek sınıflandırma tamamlanmış olur.



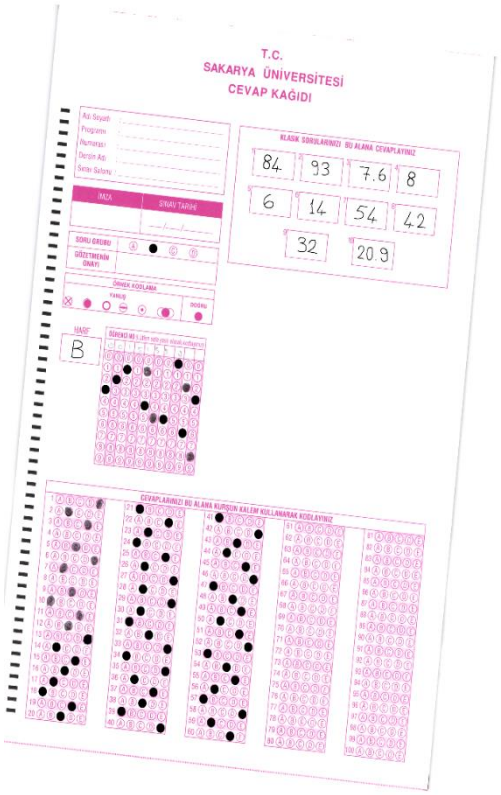
Şekil 3.21. Softmax aktivasyonu

BÖLÜM 4. UYGULAMA

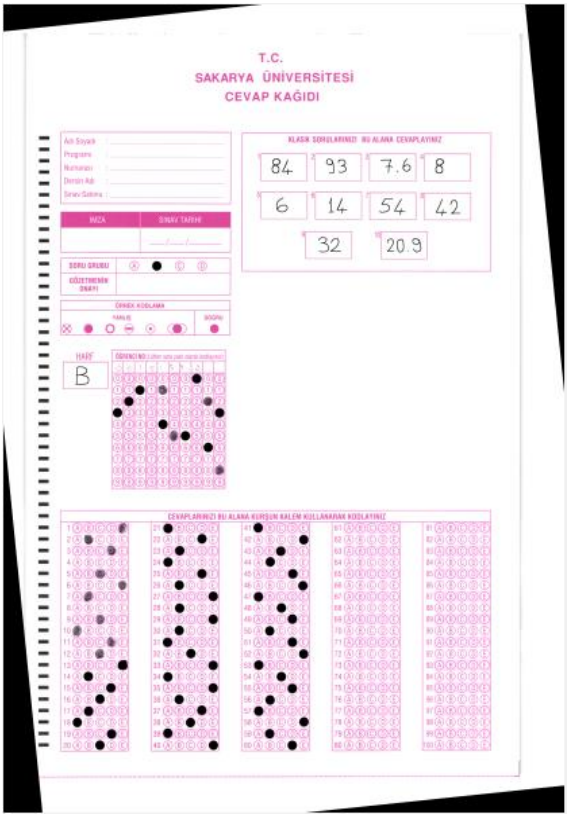
Bu çalışmada farklı cevap kağıtları üzerinde optik okuma işlemi yapılabilmesi amaçlanmıştır. Örnek test optikleri farklı cevap kağıtları üzerinde doldurularak programa girdi olarak sunulmuştur.

Cevaplandırılma yapılmış bir şekilde taratılıp programa girilen cevap kağıdı örneği görseli Şekil 4.1. arkkosinüs fonksiyonu ile 3 noktası bilinen iz düşüm üçgeni üzerinden hesaplanan derece kadar döndürülerek Şekil 4.2. olarak düzeltme sonucu olarak işleme alınır. Döndürülen resim kenararında döndürülen alan kadar siyah bölgeler oluşmaktadır, bu görselde veri kaybı yaşanmaması için görsel formatının genişletilmesi sonucu oluşmuştur ve yapılan işlemlere herhangi bir etkisi bulunmamaktadır.

Ters trigonometrik işlme sonrası elde edilen açı kadar tüm resim döndürülmüştür; alternatif olarak tüm resmi döndürmek yerine hali hazırda kırpılarak elde edilmiş alanlar tek tek bulunan açı kadar döndürülebilir. Bu durumun aksine tüm resmi döndürmek, bulunan açıyı bir kere kullanarak ayrı ayrı tüm bölgeleri dödürmek yerine tüm resmi tek seferde döndürülmesini sağlar ancak bu durumda döndürme işlemi sonrasında alanların koordinat değerleri değişeceği için tekrardan AForge.NET BlobCounter sınıfını kullanarak doldurulma yapılmış alanların bulunması gerekir. Bu durum karşısında hem alan bulma algoritması hemde döndürme algoritması lineer bir zaman karmaşıklığına sahiptir fakat döndürme algoritmasını her alan için tek tek yapmak daha maliyetli olacaktır. Bu nedenle görsel üzerinde bütün olarak döndürme işlemi gerçekleştirilir.



Şeki 4.1. Döndürme işlemi öncesi optik

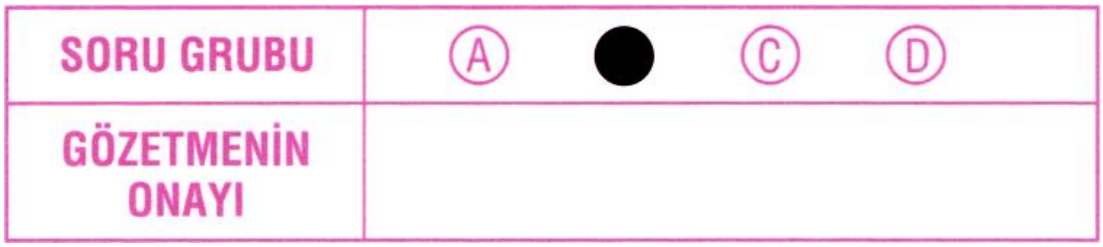


Şeki 4.2. Döndürme işlemi sonrası optik

Döndürme işlemi tam açı hesaplanarak uygulandığı için görseli 90 derecelik dik bir şekle getirmektedir. Görselin 179 derece veya daha fazla şekilde döndürülmesi durumunda algıladığı dörtgen alanların üzerinde ter trigonometrik formül uygularken almış olduğu köşe noktalar ters halde hesaplanacağından resmi ters çevirmektedir ve doğru sonuç vermemektedir. Dolayısıyla görselin tartıldıktan sonra açısal olarak olmasada konum olarak düz bir şekilde programa verilmesi gerekmektedir.

Soru grubu kısmı cevaplanan optik için 4 farklı seçenek sunmaktadır. Seçilen seçenek Şekil 4.3.'te gösterildiği gibi daire doldurularak belirlenir. Bu işlem sırasında dairelerin merkez noktalarının x eksenleri kıyaslanarak seçilen grup belirlenir. Birden fazla işaretleme yapılması durumunda hatalı giriş yapılmış olacaktır ve bu durumda

program x ekseni değeri en düşük olan grubu seçilmiş olarak algılayacaktır. Yani hem "B" hemde "C" grubu işaretlendiği taktirde program soru grubunu B olarak algılayacaktır. Programa daireleri algılama kısmında eşikleme uygulanmış yani doldurulmuş dairelerin sayısının 1 den fazla olma durumunda hata aldığını uyarı şeklinde bildirmesi eklenebilir.

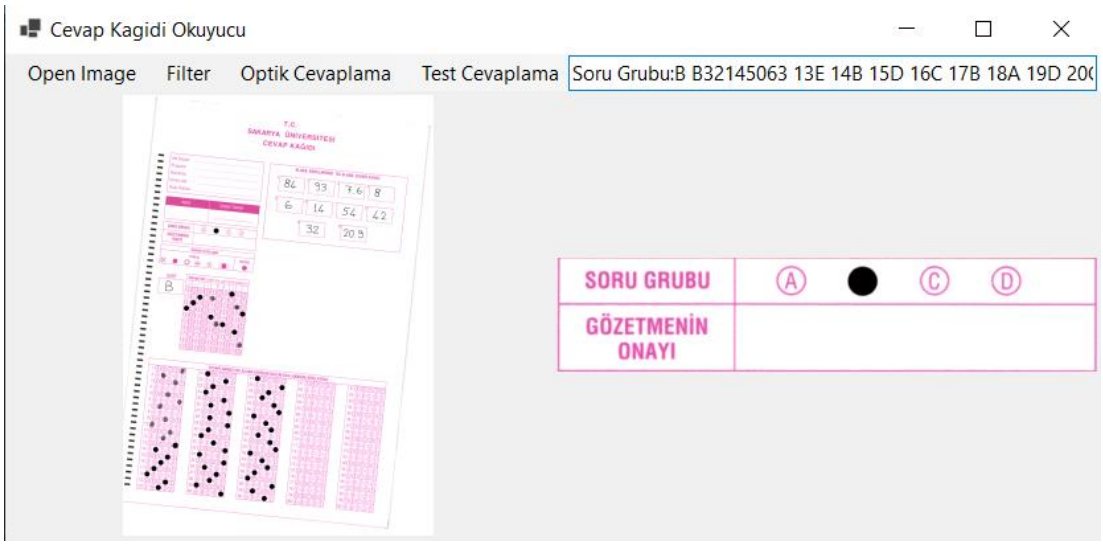


Şekil 4.3. İşaretli soru grubu

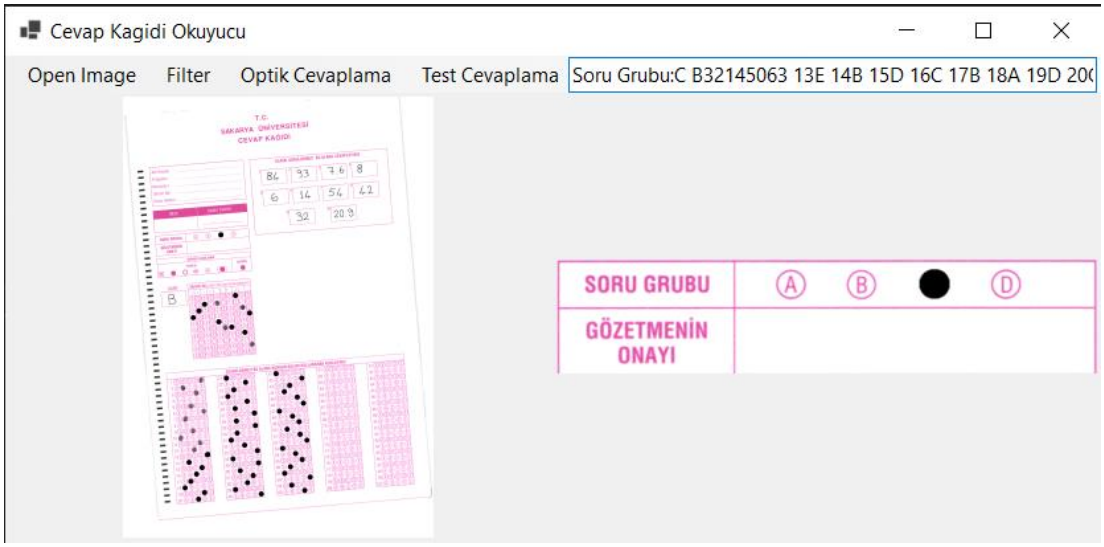
Birden fazla işaretleme öğrenci no ve test cevap alanı içinde gerçekleşebilir. Bu hatalı durum toplam daire sayısında bir değişiklik yaratmayacaktır ve sıralamada bir kayma yaşanmaz. Eşikleme işlemi ardından bulunan doldurulmuş daireler kıyaslandığında aynı x eksen aralığındaki birden fazla işaretlenmiş daire o soru sayısı için x eksen değeri en küçük olan doldurulmuş cevabı kabul edecektir. Böylelikle sistem hata vermeden devam ederek bir sonuç vermiş olur fakat birden fazla işaretleme sonucu hatalı da olsa cevap doğru bir şekilde iletilebilir ve bu durum sorun yaratacaktır. Aynı soru için olsa dahi şıkların dairelerinin x eksenleri değerleri değişiklik gösterebilmektedir, bu durum döndürme algoritması kullanılarak belirli ölçüde giderilmiştir ancak kullanıcının doldurma şekli bu durumda alınacak doldurulmuş dairenin merkez koordinat değerlerini değiştirebilir. Kullanıcıların Şekil 4.4.'te belirtildiği gibi kodlama yapmaları bu durumda büyük önem taşır.

Şekil 4.4. Doğru kodlama biçimi

Soru grubu alanı döndürme işleminin ardından kırpılarak Şekil 4.5. ve Şekil 4.6.'da olduğu gibi gösterilmiş ve farklı işaretlemeler programın arayüzündeki metin alanına bulunan sonuçlar yazdırılmıştır. Programın optik cevaplama kısmında bulunanan algoritma yardımıyla tasarlanmış cevap kağıdı üzerinde soru grubu bulma işlemi 4 farklı grup baz alınarak yapılmıştır ve 4 farklı daireden birisinin işaretlenmesi sonucu işaretli dairenin diğer dairelerle merkez noktalarının x eksen değerleri kıyaslaması sonucu sıralanarak, oluşan sıradaki konumuna göre hangi grubun işaretlendiği bulunur.

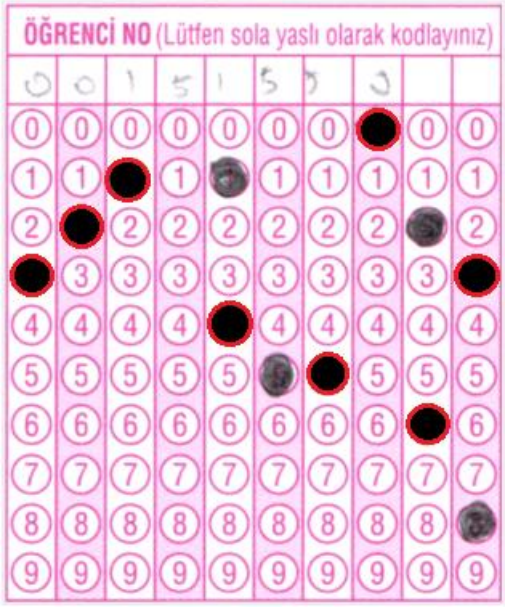


Şekil 4.5. Soru grubu örneği çıktısı



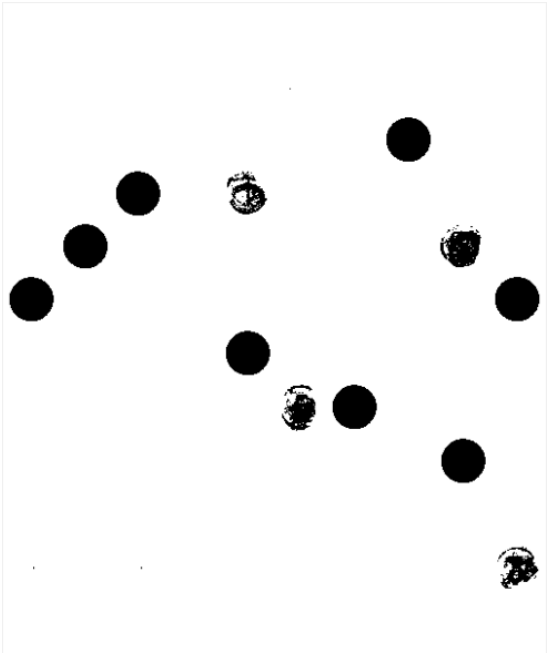
Şekil 4.6. Soru grubu örneği çıktısı

Test alanı soru sayısına bakılmaksızın işlem yapabilmektedir. Toplam daire sayısı AForge.NET BlobCounter sınıfı ile algılanarak test alanındaki toplam daire sayısı bulunur ve şıklar arası boşluklar dikkate alınarak sıralanır. Böylelikle toplam soru sayısı elde edilir. Bu yöntem sayesinde farklı dizayndaki ve soru sayısındaki test alanları aynı algoritma kullanılarak işaretli alan bulma işlemi yapılabilir kılmaktadır.

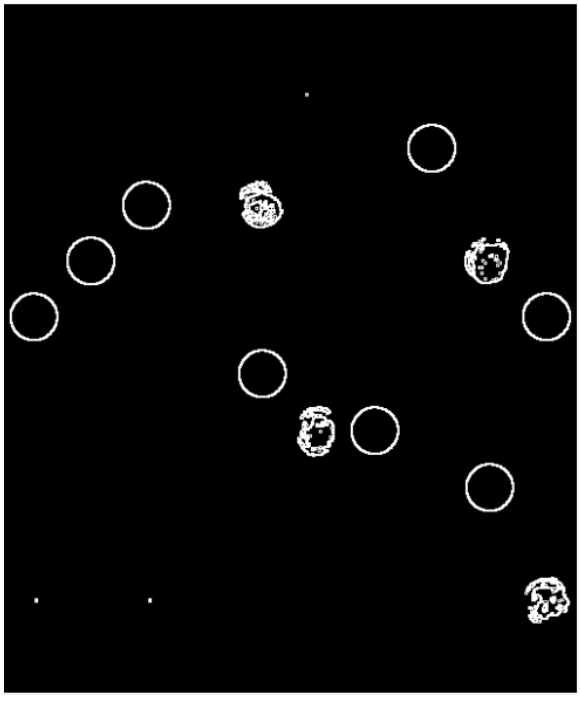


Şekil 4.7. Daire bulma uygulanmış öğrenci no alanı

Şekil 4.7.'de görüldüğü gibi belirli eşik değerinin altındaki doluluk oranındaki daireler işaretlenmiş olarak kabul edilmemiştir. Şekil 4.8.'de gösterildiği gibi eşikleme sonucunda silik olarak işaretlenmiş daireler tam yuvarlak olarak gözükmemektedir. Şekil 4.9.'da eşikleme işlemi sonrası ikili filtre uyglanmış öğrenci no alanı örneğinde tam dolu işaretlenmiş daireler keskin bir yuvarlak oluştururken silik işaretlenmiş daireler şekilsiz ve sürdürülebilirlikten uzak bir dış çerçeve oluşturmuş olarak gözükmektedir. Daireler tam dolu ve tam bir daire şeklinde olmalıdır.



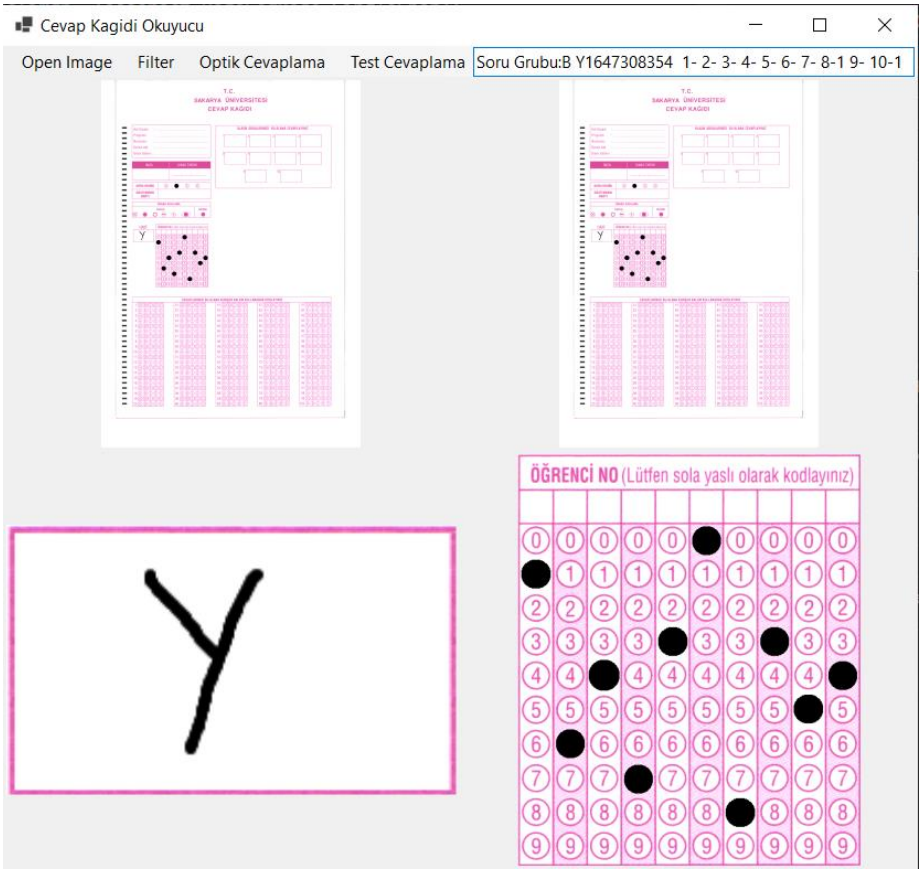
Şekil 4.8. Eşikleme uygulanmış öğrenci no alanı



4.9. İkileme filtresi uygulanmış öğrenci no alanı

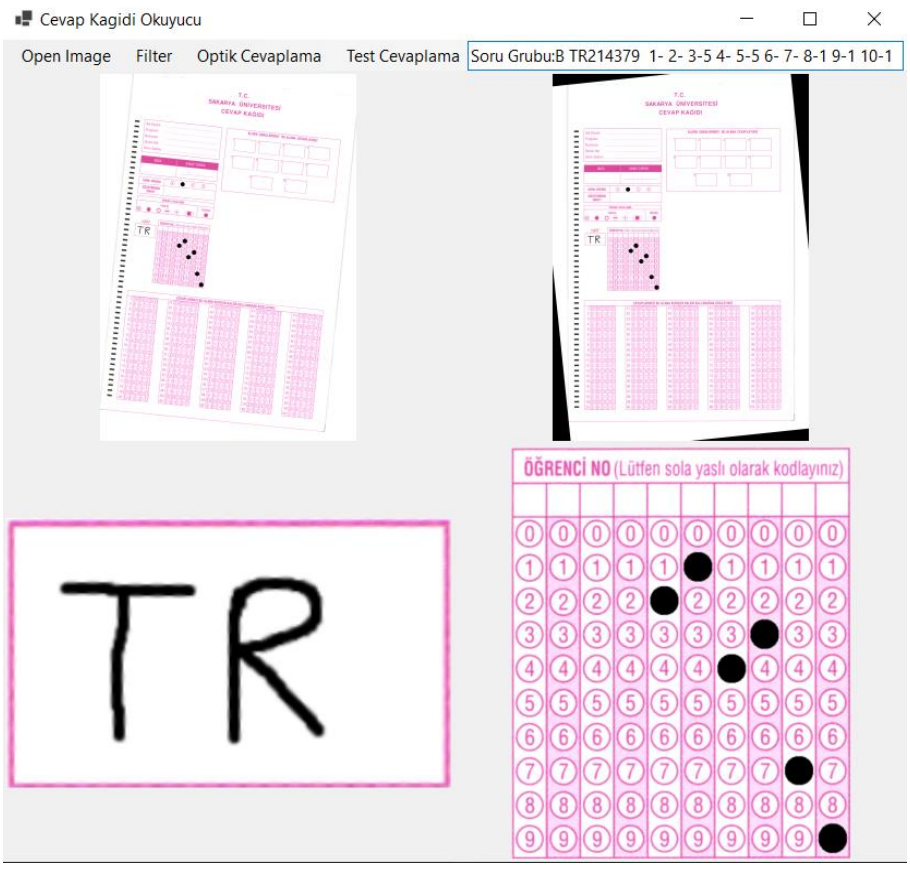
Öğrenci no alanında aynı sütun üzerinde birden fazla işaretleme yapılması durumunda program bu girdilerin hepsini değer olarak almakta ve aynı sütundaki değerleri farklı farklı yazarak hata vermektedir. Silik olarak işaretlenmiş daireleri ise Şekil 4.9.'da gözrüldüğü gibi daire olarak almamakta dolayısıyla işaretleme yapılmamış gibi program devam etmektedir. İşaretleme yapılmamış gibi algılasa dahi bu durum daire merkez değerlerinde bir kayma ve sıra bozukluğu yaratmamaktadır ve diğer işaretli daireler için sonuç doğru bir şekilde devam etmektedir.

Şekil 4.10.'da örnek olarak verilmiş bir optik kağıdında öğrenci numarası doldurulmuştur ve program harf ve öğrenci no alanlarını kırparak göstermek için düzenlenmiş ve sonuç olarak "Y1647308354" olarak dairelerin işaretlendiğini algılayarak metin olarak çıktı sağlamıştır.



Şekil 4.10. Öğrenci no örnek çıktısı

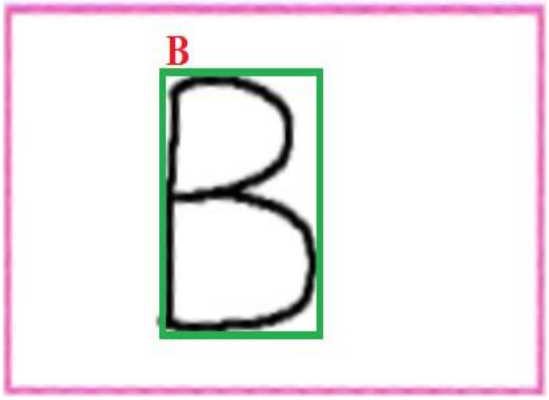
Şekil 4.11.'de öğrenci numarası doldurulmuş ve sağa belirli bir açıda döndürülerek programa verilmiş optik kağıdının optik okuma sonucu öğrenci numarası "TR214379" olarak algılanmıştır. Öğrenci okuma alanına el yazısıyla karakter girme işleminin belirli bir sınırı yoktur, karakterler arası aralıklı ve karakterler okunabilir olması halinde harf kısmı Şekil 4.11.'de olduğu gibi okunarak sonuç sağlanabilecektir. Şekil 4.11.'de gösterildiği gibi optik kağıdında öğrenci numarası sağa kaydırılmış bir şekilde sol tarafında önü boş olacak şekilde kodlanmıştır. Program işaretli daireleri algılayıp işleme tabi tuttuğu için arada boşluk bırakılması, sola veya sağa yapışık bir biçimde numaranın işaretlenmesi bir sorun yaratmayacak ve doğru sonuç sağlayacaktır. Bu durum optik okumada esneklik sağlayacak ve farklı öğrenci numaralarına sahip öğrenciler için aynı format içerisinde hizmet sağlayabilecektir.



Şekil 4.11. Öğrenci no örnek çıktısı

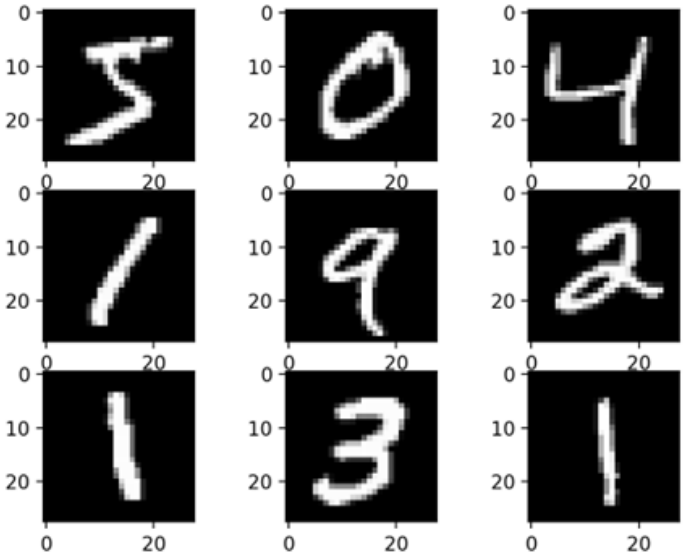
Öğrenci numarasının harf algılama işlemi haricinde sayı olan kısmının algılanmasında sadece dairelere yapılan işaretleme baz alınmıştır bunun dışındaki alanlara yapılan işaretleme veya yazılar belirlemeye bir etki sağlamamaktadır.

Şekil 4.12. Çerçeve kesme işlemi



Şekil 4.13. Harf algılama

Öğrenci Numarasının harf kısmının algılanması sırasında çerçeve harfin algılanmasında sorun teşkil ettiği için dörtgenin kenarlarından 5'er milimetre içeriye doğru kesilerek Şekil 4.12.'de oluşan kırpılmış resmin üzerinde işlem gerçekleştirilir.



Şekil 4.14. Veri seti örneği (https://machinelearningmastery.com/wp-content/uploads)

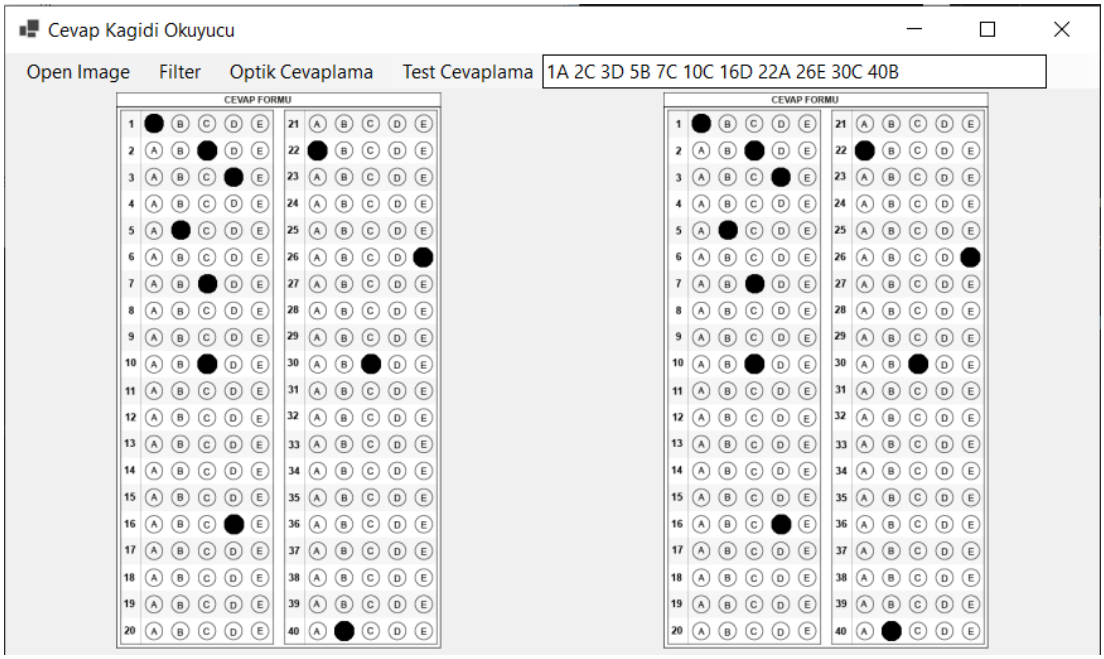
Bu proje için halihazırda tesseract ile mnist veri seti içerisinde harf ve rakam olmak üzere iki farklı sınıflandırma için ayrı olarak kullanılmıştır. Harf veri seti öğrenci no harf algılamasında ve rakam veri seti ise klasik soru cevapları kısmında kullanılmıştır.

Karakter algılama MNIST veri seti yardımıyla yapılmaktadır. MNIST veri seti el yazısı veritabanı, 60.000 örneklik bir eğitim setine ve 10.000 örneklik bir test setine sahiptir. NIST tarafından sağlanan daha büyük bir kümenin alt kümesidir (http://yann.lecun.com/exdb/mnist/).

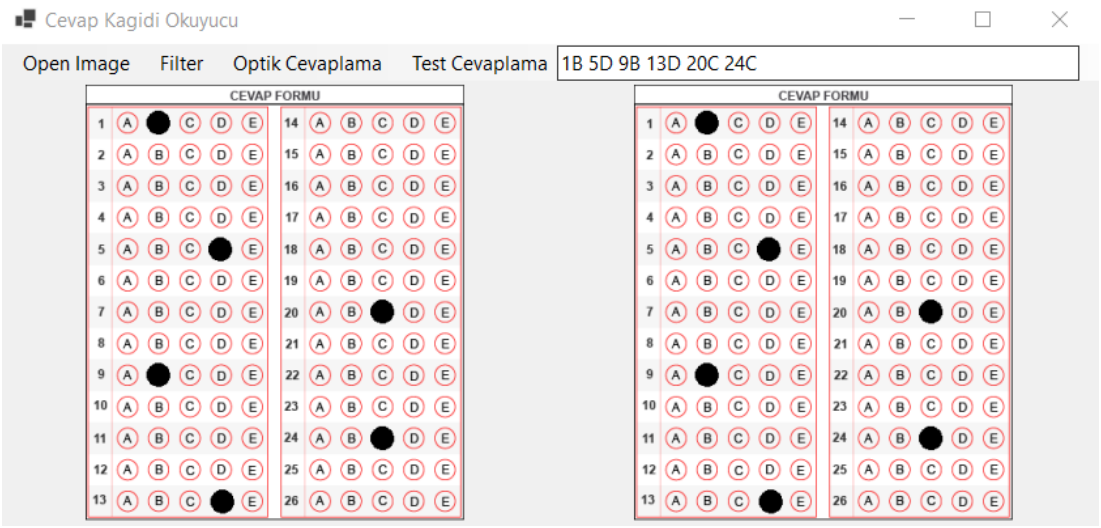
Şekil 4.15.'te rakamların örneklemeleri bulunmaktadır. MNIST veri seti program içerisinde daha iyi sonuçlar alabilmek için el yazısı harfleri ve rakamları olmak üzere iki ayrı sete ayrılarak kullanılmaktadır. Başlık algılama ve öğrenci numarasının harf kısmı için el yazısı harfleri veri seti; klasik soruların cevaplandığı alan için ise el yazısı rakamları veri seti ayrılarak kullanılmıştır.

Şekil 4.15. Veri seti örneği (https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/MnistExamples.png)

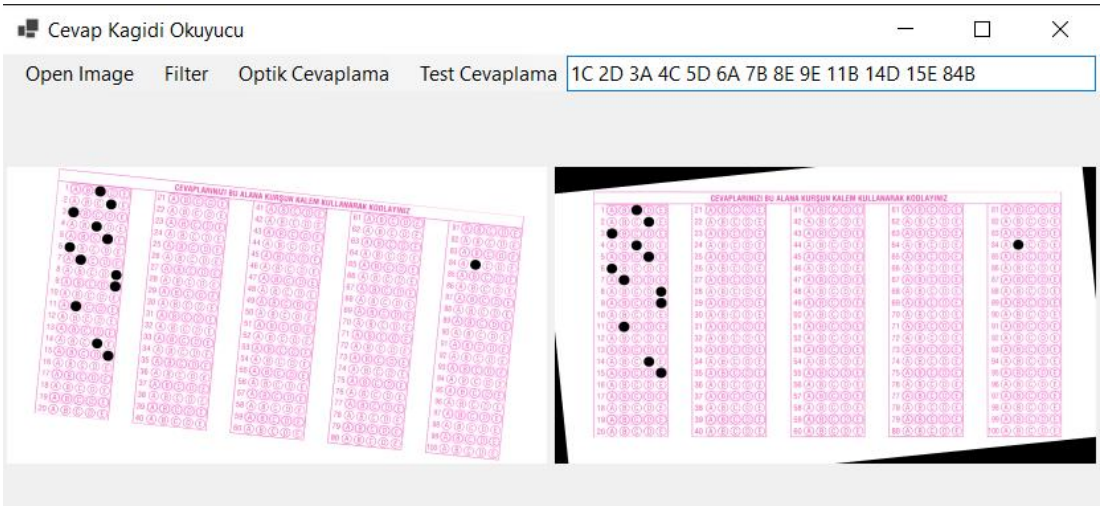
Cevap kağıdı okuyan programın, dizayn edilmiş cevap kağıdı dışındaki optikler için test okuma kısmında örneklerde verildiği gibi test alanlarında doldurma yapılmış optik kağıtlar taratılarak görsel programa girdi olarak sağlanmıştır. Bir Windows Forms uygulaması arayüzünde test cevaplama butonu ile test alanı içeren görsel sırasıyla gri tonlama ve eşikleme filtreleri uygulanır, sağa veya sola yatık olma durumuna göre görseli düzeltilir, daire içeren kısım için BlobCounter sınıfı ile şekil bulma işlemi uygulanarak dairelerin koordinat değerleri bulunur ardından eşikleme yapılarak dolu olan daireler bulunur ve dairelerin koordinat düzlemleri kıyaslanarak sırayla işaretlenmiş cevaplar Windows Forms uygulaması alanına metin olarak yazdırılır. Farklı test örnekleri ve soru sayılarıyla program çalıştırılarak sonuçlar alınmıştır.



Şekil 4.16. Test alanı örnek çıktısı



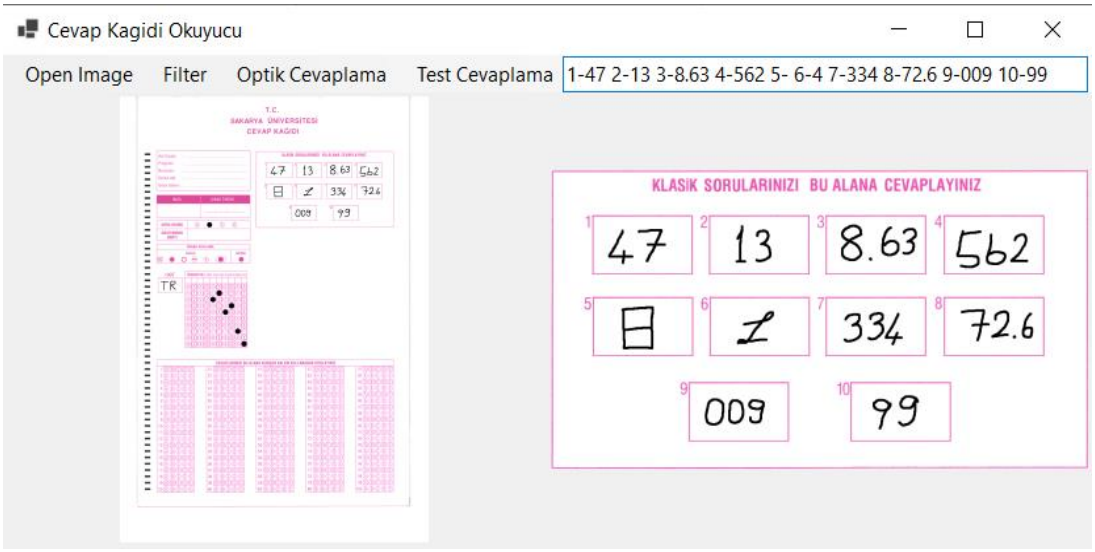
Şekil 4.17. Test alanı örnek çıktısı



Şekil 4.18. Test alanı örnek çıktısı

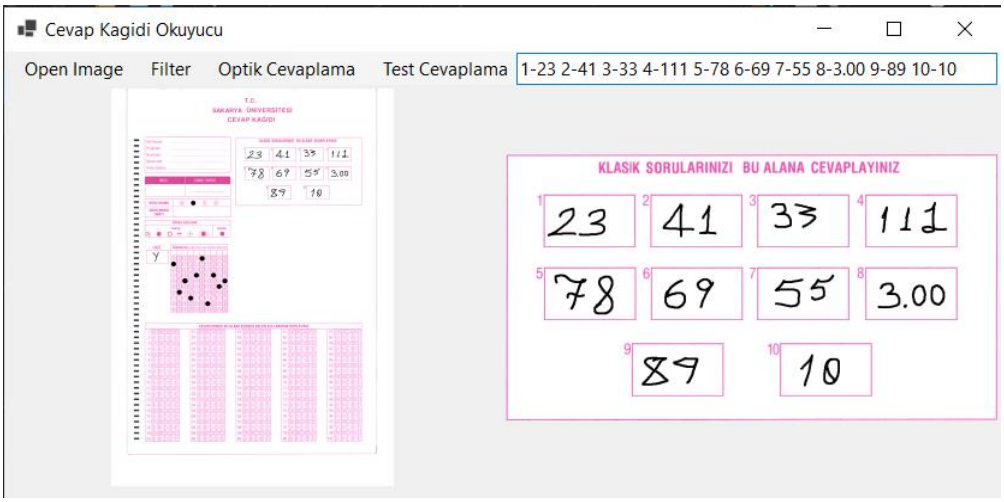
Şekil 4.19.'da klasik soru alanı doldurulmuş bir optik cevap kağıdı programa girdi olarak sağlanmıştır. Cevap alanlarından 4 numaralı alan için 562 sayısı 6 rakamı "b" harfi gibi gözükecek biçimde yazılarak programa verilmiştir. Program klasik soru karakter algılama işlemi için kullandığı veri seti sadece rakamları algılayacak biçimde ayıklandığı için herhangi bir harf girildiğinde onu ya benzer bir rakam olarak algılayacak yada algılamayacaktır. 5 numralı cevap alanına girilen 8 rakamı algılanamamış ve boş olarak geri dönüş sağlanmıştır. 6 numaralı alan için 1 rakamı

eğik bir biçimde yazılmış fakat program bu rakamı 4 olarak algılamıştır. Sonuç olarak "1-47 2-13 3-8.63 4-562 5- 6-4 7-334 8-72.6 9-009 10-99" cevabını bir string olarak dönmektedir.

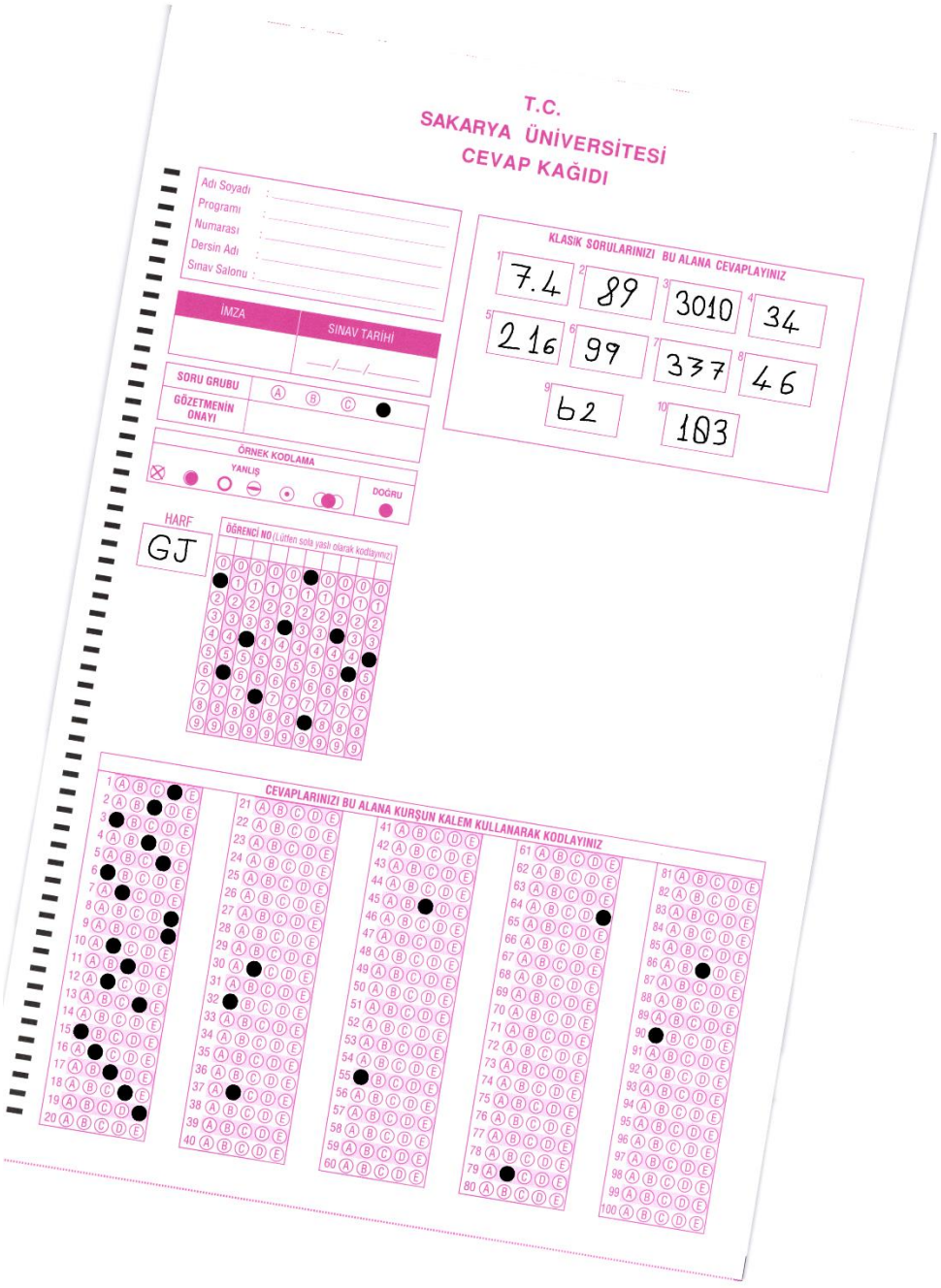


Şekil 4.19. Klasik soru alanı örnek çıktısı

Şekil 4.20.'de farklı bir şekilde doldurulmuş klasik alan cevapları "1-23 2-41 3-33 4-111 5-78 6-69 7-55 8-3.00 9-89 10-10" olacak şekilde doğru algılayarak geri dönüş sağlamıştır.

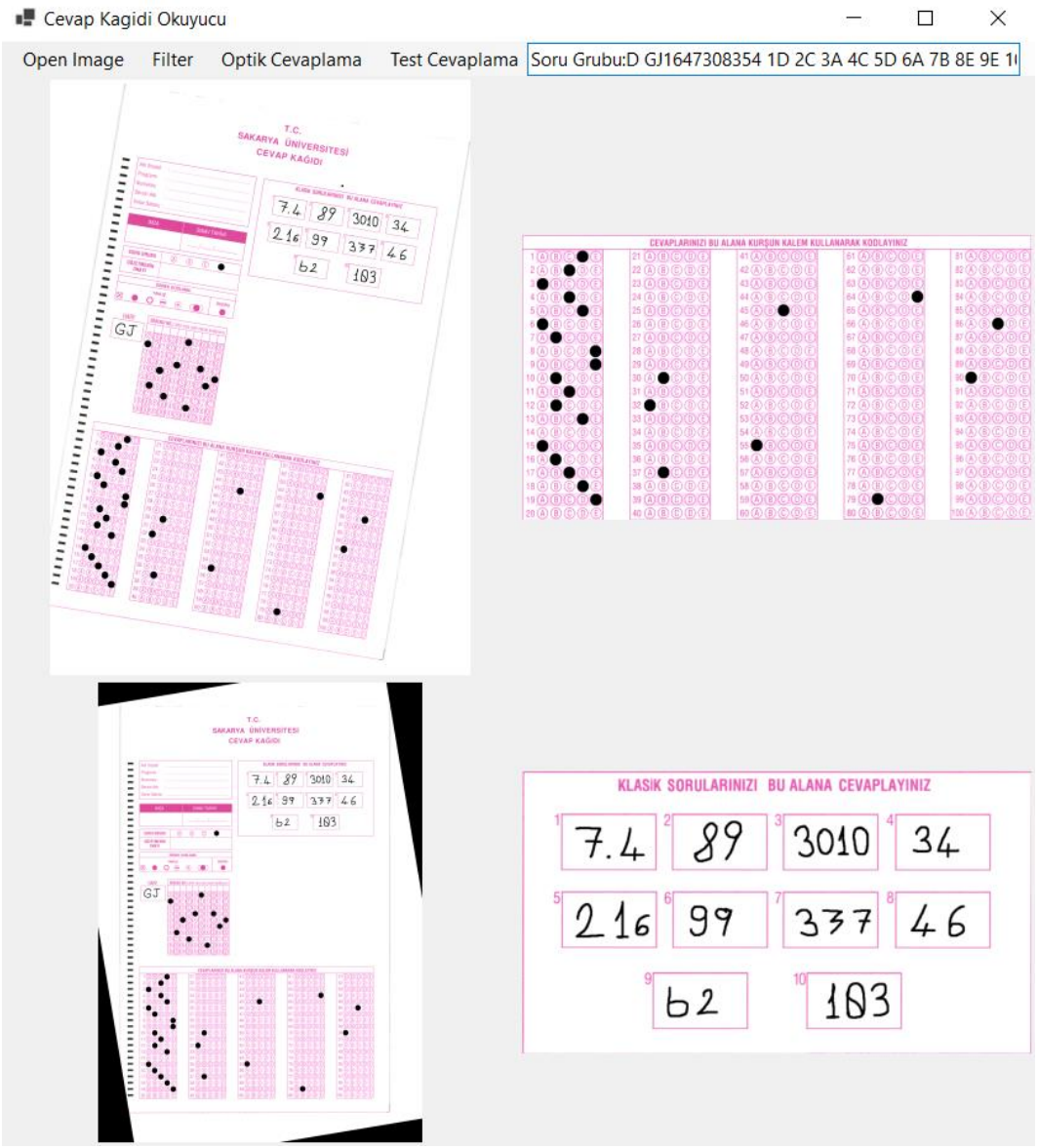


Şekil 4.20. Klasik soru alanı örnek çıktısı



Şekil 4.21. Örnek doldurulmuş cevap kağıdı

Şekil 4.21. örnek olarak doldurulmuş ve sağa yatık bir şekilde programa girdi olarak verilmiş optik kağıdı döndürme filtreleme ve cevapların algılanması işlemlerinden gerçirilmiştir.



Şekil 4.22. Örnek cevap kağıdı program çıktısı

Girilen görsel üzerinde işlemler yapıldıktan sonra oluşan çıktı;

Soru Grubu:D GJ1647308354

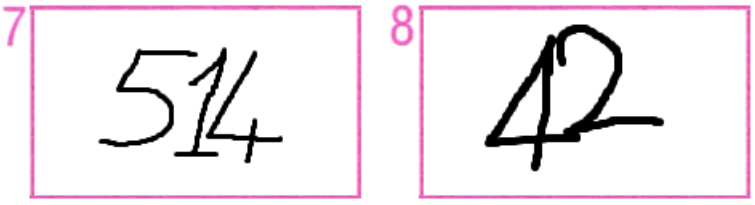
1D 2C 3A 4C 5D 6A 7B 8E 9E 10B 11C 12B 13D 15A 16B 17C 18D 19E 30B 32A 37B 45C 55A 64E 79B 86C 90A

1-7.4 2-89 3-3010 4-34 5-2 16 6-99 7-337 8-46 9-62 10-183

sırasıyla soru grubu, öğrenci numarası, test işaretlemeleri ve klasik soruların cevapları şeklinde bir string olarak kullanıcıya sunulur.

Şekil 4.23. Doldurulmuş örnek klasik soru alanı

Şekil 4.23. sayılar düzgün ve boşluklu bir şekilde yazılmıştır. İstenilen şekilde okunaklı ve boşluklu yazıldığı takdirde 6 farklı deneme sonucu doğru algılama oranı %73,33 olarak hesaplanmıştır.



Şekil 4.24. Yanlış formatta yazılmış klasik soru cevapları

Şekil 4.24.'teki gibi bir yazım karşısında sistem sırasıyla 514 ve 42 olması gereken değerlere 50 ve 147 cevaplarını vermiştir, dolayısıyla yüksek doğrulukta sonuçlar elde

edebilmek yazılan rakamlar arasındaki boşluklara ve okunaklı yazılmasına büyük ölçüde bağlıdır.

Cevap kağıdı üzerinde başlık okuma; cevaplama alanları şekil bulma yöntemiyle bulunduktan sonra gerekli alanların kırpılıp çerçeve çıkarma işlemi yapıldıktan sonra başlık alanlarında karakter tanımlayarak alan belirlenmesi yapılmaktadır. Bu işlem yerine bulunan alanlar arasında daire bulma işlemi yapılarak test alanı bulunabilir ve daire sayısına göre öğrenci numarası alanı ve test cevaplama alanı ayrılabilir ve işlem yükünü azaltılabilir fakat bu yöntem sonucunda klasik cevap alanını tanımlayabilmek zorlaşacaktır. Alanlar içerisine üçgen, beşgen vb. farklı şekiller konularak alanlar birbirlerinden ayrım yapılabilir fakat bu yöntemde kullanıcı için kafa karışıklığı yaratabilir ve bütün optik kağıtları için bu işlem çözüm olamaz. Bunun yerine her optikte tanımlama amaçlı bulunan başlık baz alınarak işlem gerçekleştirilmektedir.

BÖLÜM 5. TARTIŞMA VE SONUÇ

Standart cevap kağıtları baz alınarak tasarlanan bir cevap kağıdı üzerinde doldurulan alanların incelenmesi işlemi gerçekleştirilmiştir ayrıyeten tasarlanan cevap kağıdı dışındaki cevap kağıtları için de optik üzerinde doldurulmuş olan test alanlarının okunarak sırasıyla işaretlenen soru alanlarının cevaplarını metin olarak kullanıcıya bir arayüzde sunar.

Doldurulan alanların dörtgenler olarak ayrılmasına göre belirlenmesi esas alınmıştır. Bu durum bazı optikler için belirleyici özelliklerinin dörtgen biçimindeki alanların dışında kalarak algılama işlemi sonrası veri kaybı yaşanmasına sebep olabilir. Bu nedenle daha efektif bir şekilde bölge algılama işlemi olarak kağıt üzerindeki daire sayılarının kağıt üzerindeki yoğunluğuna göre bölge sınıflandırılması yapılarak doldurulan alanların dörtgen temeline oturtulmadan bulunabilmesini sağlayabilir.

Daire doldurarak işaretlenen alanlarda test alanları için aynı satır; öğrenci numarası alanı için aynı sütunda yapılacak hatalı işaretlemeler sonucu program işaretlenen dairelerin merkez noktalarına göre işleme yapmaktadır. Bu durum hatalı işaretleme yapılsa dahi sistemin o anki sütun veya satırdaki yapılan hatalı işaretlemeyi kabul ederek birden fazla işaretleme yapılmış olsada yanlış cevabı kabul etmektedir. Bu durumun önüne geçmek için belirlenen alana bağlı olarak test alanı olması durumunda satır; öğrenci numarası için satırda birden fazla işaretleme yapıldığını algılayarak sonucu boş veya hatalı döndürmesi sağlanabilir.

Program ileride cevapların tanıtılarak gerçek zamanlı okuma yapacak şekilde geliştirilebilir. Bu işlem oldukça basit bir string kıyaslama işlemi ile programın çıktısını aynı formatta girilen cevap anahtarı ile karşılaştırarak eşleşen sonuçlara göre

gerçek zamanlı puanlama yaparak sınav sonrası süreci kısaltacaktır. Bu işlem için yüksek çözünürlükte bir kamera yardımıyla optik hemen sınav sonrasında taratılarak algoritma çıktısının kıyaslanacağı ortam sağlanabilir.

KAYNAKLAR

[1] Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I.; Application of the residue number system to reduce hardware costs of the convolutional neural network implementation; 2020.

[2] https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ , Erişim Tarihi: 20.04.2022.

[3] https://www.ecma-international.org/publications-and-standards/standards/ecma-334/ , Erişim Tarihi: 23.04.2022.

[4] https://en.wikipedia.org/wiki/.NET_Framework#Architecture , Erişim Tarihi: 22.04.2022.

[5] https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview , Erişim Tarihi: 22.04.2022.

[6] https://www.nuget.org/profiles/aforge.net , Erişim Tarihi: 11.05.2022.

[7] https://www.emgu.com/wiki/index.php/Main_Page , Erişim Tarihi: 07.05.2022.

[8] https://www.emgu.com/wiki/index.php/Main_Page , Erişim Tarihi: 07.05.2022.

[9] https://github.com/tesseract-ocr/tesseract , Erişim Tarihi: 18.05.2022.

[10] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi 2018.

[11] https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html , Erişim Tarihi: 12.05.2022.

[12] Johnson, Stephen (2006). Stephen Johnson on Digital Photography; 2006.

[13] https://en.wikipedia.org/wiki/Grayscale , Erişim Tarihi: 18.05.2022.

[14] http://www.aforgenet.com/framework/docs/html/d0eb5827-33e6-c8bb-8a62-d6dd3634b0c9.htm , Erişim Tarihi: 20.05.2022.

[15] http://www.aforgenet.com/framework/docs/html/d7d5c028-7a23-e27d-ffd0-5df57cbd31a6.htm , Erişim Tarihi: 20.05.2022.

[16] http://www.aforgenet.com/framework/docs/html/59949f70-2afc-f7a5-1a53-ff99a9208133.htm , Erişim Tarihi: 20.05.2022.

[17] https://www.britannica.com/technology/bitmap , Erişim Tarihi: 21.05.2022.

[18] https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/ (Görsel) , Erişim Tarihi: 16.05.2022.

[19] https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/ (Görsel) , Erişim Tarihi: 16.05.2022.

[20] https://computersciencewiki.org/index.php/File:MaxpoolSample2.png (Görsel) , Erişim Tarihi: 16.05.2022.

[21] https://machinelearningmastery.com/wp-content/uploads (Görsel) , Erişim Tarihi: 17.05.2022.

[22] http://yann.lecun.com/exdb/mnist/ , Erişim Tarihi: 23.05.2022.

[23] https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/MnistExamples.png/480px-MnistExamples.png (Görsel) , Erişim Tarihi: 23.05.2022.

ÖZGEÇMİŞ

Adı Soyadı **:** Berkay ÇETİN

ÖĞRENİM DURUMU

Derece	Eğitim Birimi	Mezuniyet Yılı
Yüksek Lisans	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar ve Bilişim Mühendisliği	Devam ediyor
Lisans	İstanbul Medipol Üniversitesi / Mühendislik ve Doğa Bilimleri Fakültesi / Bilgisayar Mühendisliği	2019
Lise	Sakarya Şahin Anadolu Lisesi	2014

İŞ DENEYİMİ

Yıl	Yer	Görev
2021-Halen	Tarım Kredi Yem	Yazılım Geliştirici
2019-2021	NOTOK Games	Mobil Geliştirici

YABANCI DİL

İngilizce

ESERLER (makale, bildiri, proje vb.)

1.

2.

HOBİLER

……. , …….. , …….