BIL 467 Görüntü İşleme

Ödev #4

Kağan Hamzaçebi

151101064

26/07/2021

1. Genel Açıklama

Ödev klasörü içerisinde number_recognizer isimli script dosyası ve dataset (training ve test dataları) klasörü bulunmaktadır. İstenilen tüm fonksiyonlar number_recognizer içinde bulunmaktadır.

1.1. Kullanım

Ödev recognize isimli bir fonksiyon çağırılarak kullanılmaktadır. Bu fonksiyon target isimli bir parametre kabul etmektedir. Bu parametreye test edilmesi istenilen image verilmesi gereklidir. recognize fonksiyonuna test görüntüsü verildikten sonra Command Window'da script'in çalışması tamamlandıktan sonra sonucu görebilirsiniz.

Not: 15. satırda bulunan training dataset'in absolute path'ini kendi bilgisayarınıza göre değiştirmeniz gerek.

Not2: Eğer Test olarak verdiğiniz image Binary ise 4. ve 5. satırlarda uygulanan image'ı binary'e dönüştürme işlemlerini kaldırabilirsiniz. Benim test verilerim Paint uygulamasında oluşturulduğundan dolayı RGB olarak kaydedildi. Bu sebeple verimi ilk olarak Grayscale sonrasında da Binary'e dönüştürmem gerekti. Kısacası recognize isimli fonksiyona verilen Test verisinin Binary olması gerekiyor.

2. Number Recognizer

2.1. Çalışma Yapısı

Bu bölümde yazmış olduğum script dosyasının verilen image'ın hangi rakama ait olduğunu nasıl bildiğini ve hangi metodları kullanarak nasıl bu işlemi gerçekleştirdiğini anlatacağım.

Gerekli tanıma işlemini gerçekleştirebilmek için iki farklı metod tercih ettim. Birincisi Euler Number ve ikincisi ise Shape Number (Chain Code ve First Difference'dan elde

edildi). İlk olarak recognize fonksiyonu çağırıldığında image'ın Euler Number'ını bularak içerisinde kaç adet boşluk (hole) bulunduğunu elde ediyorum. Bu bulmuş olduğum boşluk sayısı 0, 1 veya 2 olabiliyor. Bulmuş olduğum Euler Number'a göre işleme devam ediyorum. Eğer Euler Number 2 ise image'ın sekize ait olduğuna ulaşmış oluyorum, eğer Euler Number 1 ise işlemime bir, iki, üç, beş, yedi rakamları ile devam ediyorum. Eğer Euler Number 0 çıkar ise işleme sıfır, dört, altı ve dokuz ile devam etmem gerektiğini anlıyorum. Bu ilk süzgeçten geçirdikten sonra test ettiğim image'a ait shape number'ımı oluşturuyorum. Shape Number oluşturma işlemini derste anlatıldığı gibi ilk önce chain code, daha sonrasında first difference ve first difference'ı shape number'a dönüştürerek elde ediyorum. Shape Number üretebilmek için script'imin içerisinde yer alan im2sn fonksiyonunu oluşturdum. Shape Number oluşturmadan önce image'ı preprocess'den geçiriyorum. Preprocess kısmı sırasıyla dilation, boundry_detection işlemlerini içeriyor. Dilation uygulamamın sebebi image'da scale işlemi gerçekleştirilirken bozulmalar meydana gelmemesi. Bir image küçültüldüğünde pixel kaybı yaşanabiliyor ancak dilation sonrası boundry pointlerdeki kopmalar engellenmiş oluyor. Bu işlemler kullanılarak bir image'a ait Shape Number elde ediliyor. Daha sonrasında recognize fonksiyonu içerisinde dataset/training path'inde bulunan .bmp uzantılı tüm training datamız (Euler Number ile filtreleme sonrası) ile recognize işleminin gerçekleşmesini istediğimiz Test verimiz karşılaştırılıyor (Shape Number'ları). Karşılaştırma işlemi için Edit Distance (Levenshtein distance) kullandım. Shape Number karşılaştırmaları sonrası elde ettiğimiz Edit Distance sonuçlarından en düşük beş değeri buluyoruz. KNN (K-Nearest Neighbor) kullanarak bu beş değer arasında çoğunluğa bakarak sonucu elde etmiş oluyoruz.

Bu script'in çalışmasında önceki ödevlerde yazmış olduğum fonksiyonlardan yararlandım. Bunlar; boundry_detection (boundry detection için), imgDilate (dilation işlemi için), eulerNum (hole detection için, yazmış olduğum connected components fonksiyonumdan yararlandım) fonksiyonlarıdır.

2.2. Sonuç

Tanıma işlemi gerçekleştirilirken Training ve Test datası birbirinden tamamen bağımsız olarak kullanıldı. Her rakam için 5-10 arası verilen kriterlere uygun image'lar üretildi.

Sonuç olarak bazı rakamlarda çok başarılı sonuç verirken bazı rakamlarda ise daha düşük true positive oranı yakalandı.

Çok başarılı sonuçlara ait rakamlar (90%'dan fazla): 0, 1, 4, 5, 7, 8

Orta başarıda olan sonuçlara ait rakamlar (60-70%): 2, 3

Birbiri ile karıştırılan rakamlar: 6, 9

Bu sonuçlar training verilerimiz arttırıldıkça çok daha başarılı sonuçlar vermektedir.

Ek olarak 1 test datasının 60 küsür veri ile karşılaştırılması da 10 saniyeyi geçmemektedir, bu bakımdan çalışma hızı olarak da yeterince hızlıdır.

Not: Bu sonuçlar dataset klasörü içerisinde kullanılan veriler ile elde edildi.

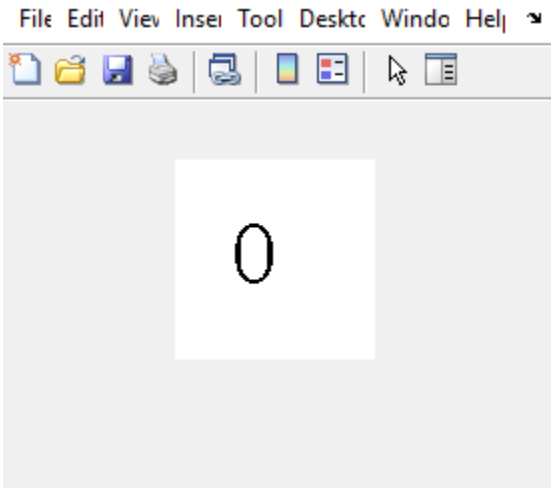
3. Karşılaşılan Problemler – Zorluklar

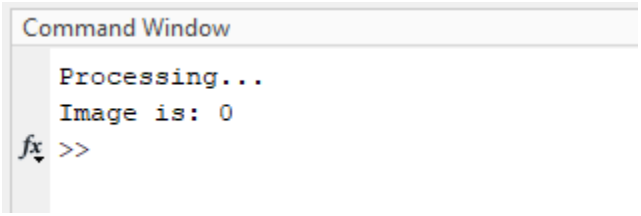
İlk yaşamış olduğum problem image'ları scale (interpolation) ederken veri kayıpları ile karşılaşmış olmam. Bu sorunu image'lara scale işlemi öncesi dilation uygulayarak çözdüm. Bu sayede scale işlemi sonrası boundry detection yaparken boundry'lerdeki kopmayı önlemiş oluyorum ve başarılı bir sonuç alıyorum.

İkinci problem olarak; ilk olarak yapmış olduğum representer, descriptor yapısı chain kod idi. Chain Code'dan Shape Number elde ettim ve bu objelere ait Shape Number'ları birbirleri ile karşılaştırıp tanıma işlemini gerçekleştirmeye çalıştım ancak beklediğimden çok daha başarısız sonuçlar elde ettim. 0, 1, 7 ve 8'I başarılı bir şekilde tanırken 1, 2, 3, 5 sayılarını 4 ile karıştırmaktaydı ve test verimi çoğalttığımda 2 ve 3 gibi rakamlarda hataların diğer rakamlara göre daha fazla olduğunu gördüm. İlk olarak 1, 2, 3 ve 5 rakamlarini 4 ile nasıl ayırabileceğimi düşündüğümde içerisindeki boşluk'lara (hole) bakabileceğimi farkettim. Euler Number algoritmasını ekleyerek bu sorunu aşmış oldum. Daha sonrasında rakamların kendi içerisindeki bulabilme oranını arttırabilmek için de KNN yapısını ekledim. Böylece elde ettiğim en iyi sonuca ulaştım.

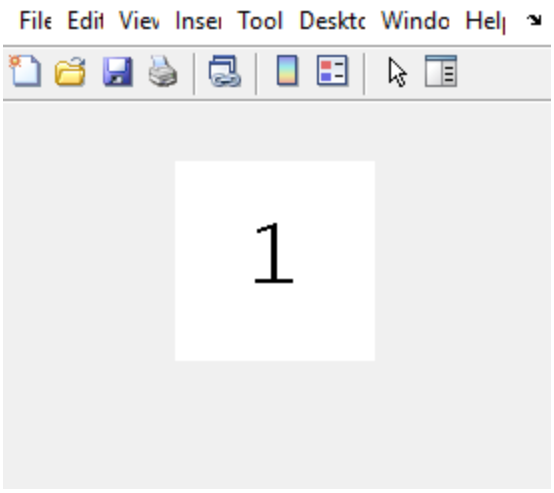
Yaşamış olduğum diğer bir problem rotation'dan bağımsız olan Shape Number'i kullandığım için 6 ve 9 rakamlarını birbirine karıştırıyor. Test image olarak 6 veya 9 verdiğimde o image'a 6 veya 9 diyor ancak bazen 6'ya 9 bazen de 9'a 6 diyebiliyor. Bu probleme çözüm üretemedim.

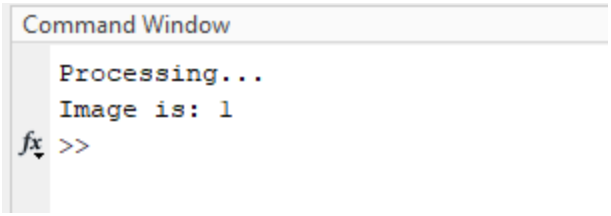
4. Girdi ve Çıktılar



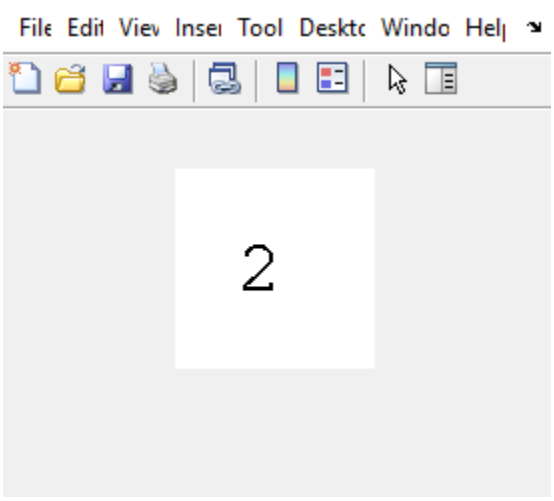


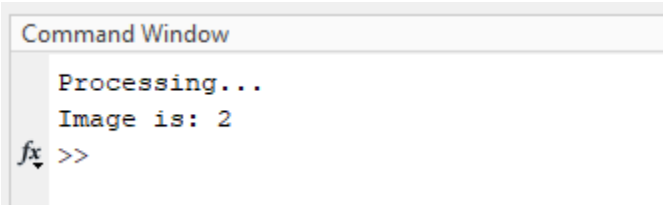
Test Input and Output – 0



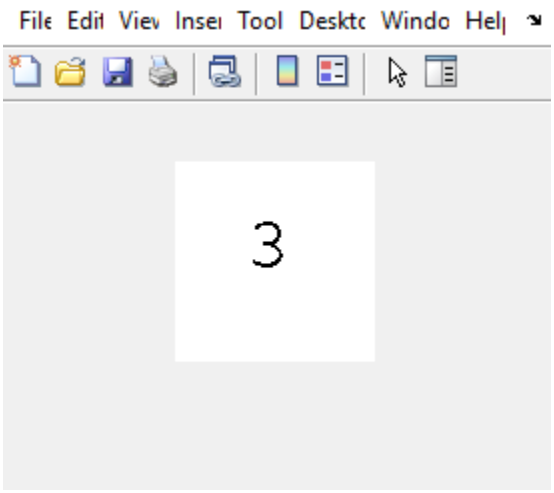


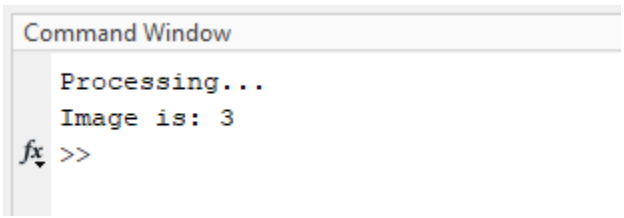
Test Input and Output – 1



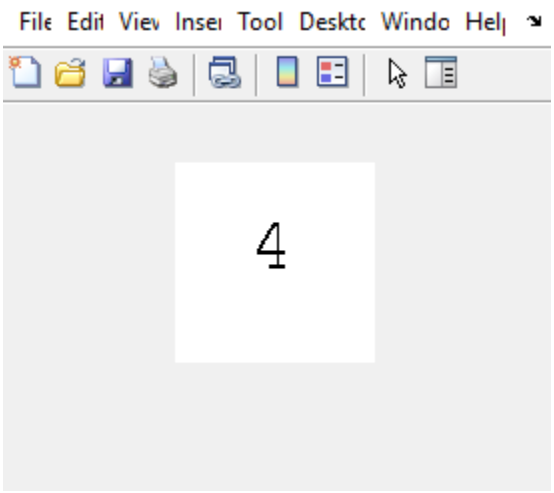


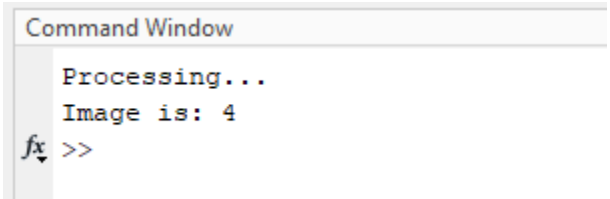
Test Input and Output – 2



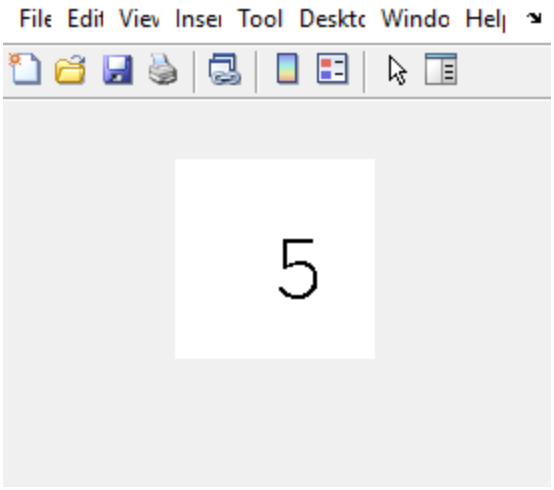


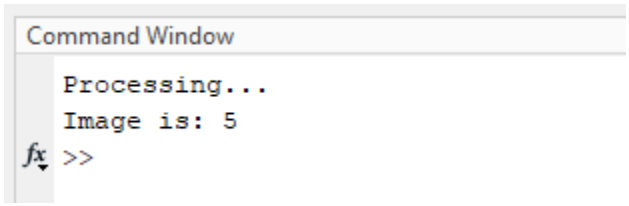
Test Input and Output – 3



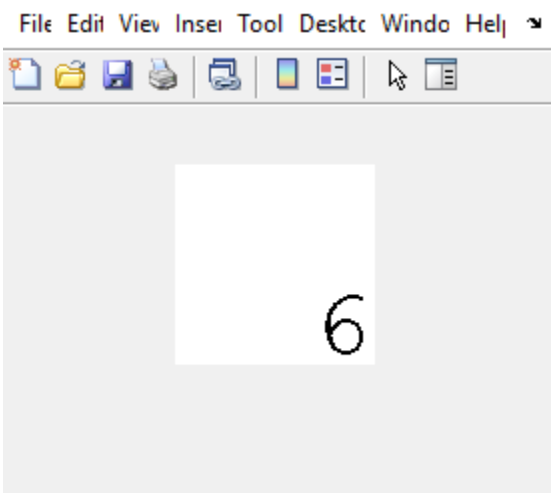


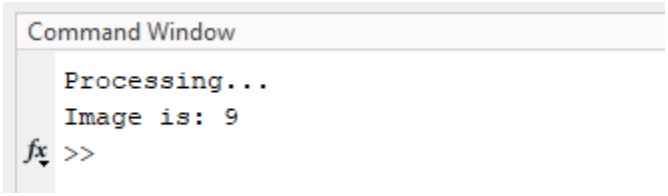
Test Input and Output – 4



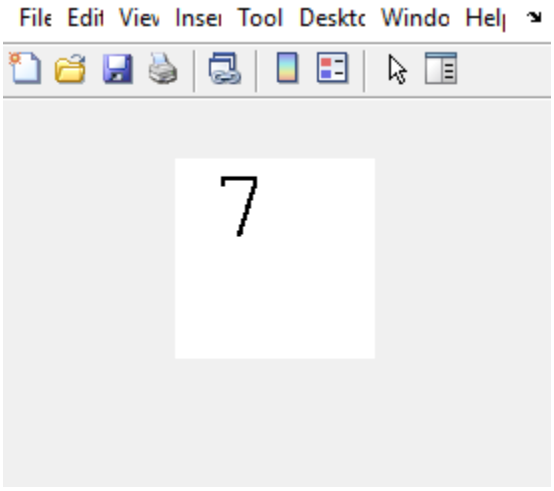


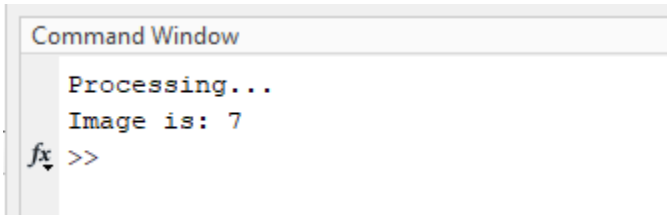
Test Input and Output – 5



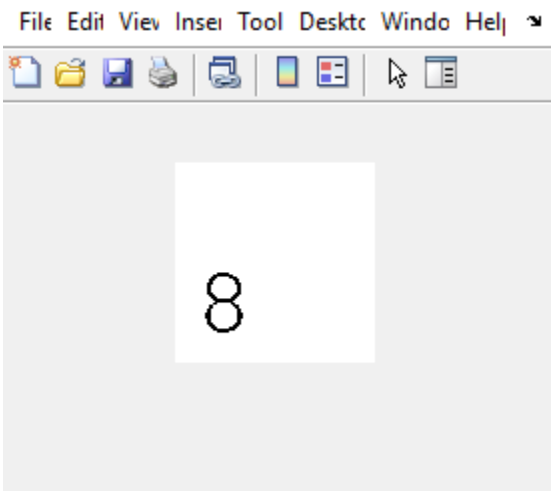


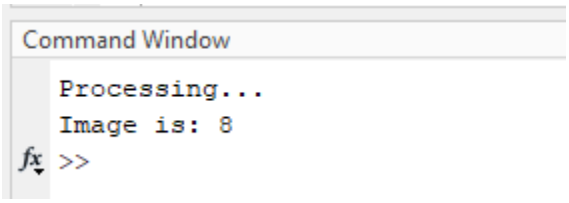
Test Input and Output – 6



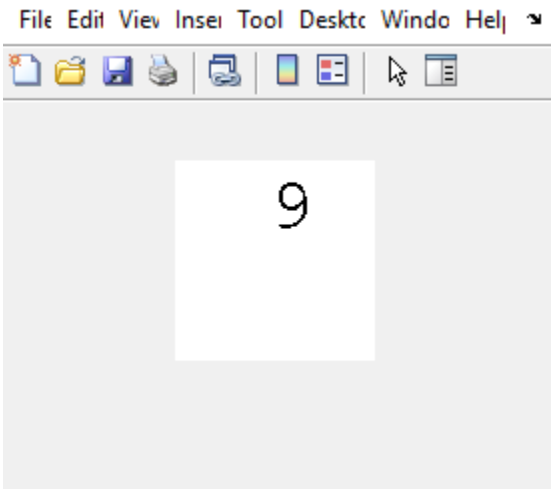


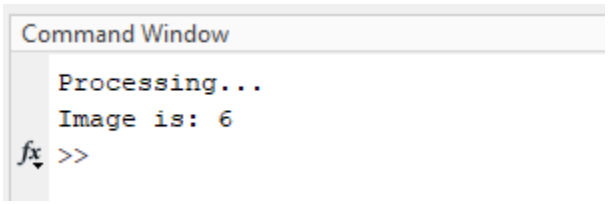
Test Input and Output – 7





Test Input and Output – 8





Test Input and Output – 9