Metal Damgalama Görüntüleri Üzerinden Karakter Tanımlaması

Erdem Nayın 180202050 – Melih Yeşilyurt 180202060

Barış Arslan 180202112 – Ali Atilla Aydemir 180202063

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

1.Problem

Günümüzde birçok fabrikada ürün numaraları metal yüzey üzerine yazılmaktadır. Bu firmalara örnek olarak Aygaz ve birçok otomotiv firması tarafından kullanılmaktadır. Bu projede çözülmesi hedeflenen sorun ise yazı metni ile arka plan renginin aynı olmasından dolayı OCR modellerinin yazıyı tespit etmemesi.

2. Karşılaşılan Sorunlar

a. Metin Rengi ve Arka Planın Aynı Renk Olması

Bu projenin zor olmasının en büyük sebebi metin rengi ile arka planın aynı renk olmasıdır. Bu sebeple genel olarak kullanılan OCR kütüphaneleri ve çoğu model bu yazıları tespit etmede yetersiz kalmaktadır.

b. Veri Seti Yetersizliği

İnternette detaylı bir araştırma yapmamıza rağmen ilgili sitelerde veri seti bulamadık. Örnek olarak bir fotoğraf verildiğinden dolayı çalışmalarımızı o fotoğraf üzerinden gerçekleştirdik. Daha sonra dersin hocasıyla iletişime geçtik ama aldığımız veri seti kalitesizdi. Veri setinden 300 fotoğraf bulunmaktaydı. Veri madenciliği dersinden öğrendiğimiz bilgileri kullanarak veri setini temizledik. Elimizde 210 fotoğraf kaldı. Kalan fotoğrafların çok düşük çözünürlüğe sahip olmasından ve sadece sayılardan oluşmasından dolayı işimize yaramadı ve kullanamadık ama daha sonra yaptığımız araştırmalar sonucunda kullanabileceğimiz ve bize uygun olan 2 fotoğraf daha elde ettik. Çalışmalarımızı elimizdeki imkanlarla gerçekleştirmek zorunda kaldık.

c. Tecrübe Yetersizliği

Daha önceden aldığımız veri madenciliği dersinin devamı olarak aldığımız yapay zekâ dersi hakkında daha önceden tecrübemiz ve çalışmamız bulunmadığı için duyduğumuz her kavramı ilk kez duymuş bulunmaktayız. Bu sebeple projemizi geliştirmeden önce kendimizi yapay zeka alanında geliştirdik.

d. Test Edilmiş filtrelerin Kodlarının Bulunmaması

Geliştirdiğimiz programı daha verimli hale getirmek için çeşitli filtreler kullandık. Bu filtreleri hızlı bir şekilde deneyebilmek ve sonuçlarını görebilmek için bazı yararlı internet siteleri bulduk. Ama bu sitelerde denediğimiz her filtrenin kodunu bulamadık. Kodlarını bulabildiğimiz filtrelerin farklı farklı kombinasyonları denemek epey zamanımızı aldı.

e. Benzer Projelerin Bulunmaması

Bu projenin konusu zor olduğundan dolayı internette hazır proje örnekleri bulamadık. Bulduğumuz projeler genellikle normal yazı tespitiyle alakalıydı. Bizim problemimiz ise daha karmaşık olduğundan dolayı o projelerden yeteri kadar faydalanamadık.

3.Denediğimiz Başarısız Çalışmalar

a. Tesseract Kütüphanesi Kullanarak Fotoğraftaki Yazı Karakterlerinin Tespiti

Python dili ve Tesseract kütüphanesi kullanılarak ilgili proje gerçekleştirilmeye çalışıldı. Tesseract kütüphanesinde bulunan fonksiyonlarla ilk önce elimizde bulunan fotoğrafların grayscale halleri elde edildi. Fotoğrafların yeni halleri üzerinde fotoğraflarda bulunan noise kaldırıldı. Daha sonra fotoğraflar Otsu thresholding binarizasyon algoritması kullanılarak filtreden geçirildi. Thresholding, ön plan olarak kabul edilen bir nesneyi arka planından ayırmak için kullanılan çok

popüler bir bölütleme tekniğidir. Bu algoritmada sağlanan threshold(eşik) değerlerine göre fotoğrafta bulunan pixel değerlerinin ataması yapılır. Thresholding'de, her pixel değeri eşik değer ile karşılaştırılır ve pixel değeri eşik değerinden küçükse 0, aksi halde ise maksimum değere ayarlanır. Eşik, her iki tarafında iki bölge bulunan, yani eşiğin altında veya üstünde olan bir değerdir. Computer Vision'da, bu eşikleme tekniği gri tonlamalı görüntüler üzerinde yapılır. Bu nedenle, başlangıçta görüntünün gri tonlamalı renk uzayına dönüştürülmesi gerekir. Bu yüzden fotoğraf bir önceki adımda grayscale hale çevrilmişti. Bu yöntemle istediğimiz sonuca ulaşılamadı çünkü yazı karakterlerinin rengi ile arka plan rengi aynıydı ve dataset'teki fotoğraflar çok küçüktü. Filtreler ve kütüphane uygun bir sonuca ulaşamadı.

b. EasyOCR ve Torch Kütüphanesi Kullanılarak Fotoğraftaki Yazı Karakterlerinin Tespiti

Python dili ve EasyOCR ve Torch kütüphaneleri kullanılarak ilgili proje gerçekleştirilmeye çalışıldı. EasyOCR, adından da anlaşılacağı gibi, computer vision geliştiricilerinin Optical Character Recognition işlemini zahmetsizce gerçekleştirmesini sağlayan bir Python paketidir. Proje gerçekleştirilirken ilk olarak fotoğraf kaynakları programa okundu. Daha sonra bu kütüphanelerde bulunan hazır fonksiyonlar kullanılarak yazının tespiti halinde etrafının dikdörtgen ile çevrilmesi için kodlar yazıldı. Bu alan içerisinde kalan yazıların tespiti ve ekrana bastırılması için yine hazır fonksiyonlar bu kütüphanelerden temin edildi ve kullanıldı. Bu yöntemin başarsız olmasının nedeni fotoğraftaki tespit edilmesi gereken yazı karakterlerinin rengi arka plan rengiyle aynı olduğu için hazır kütüphane fonksiyonları bunları tanımakta yetersiz kaldı ve dataset'teki fotoğraflar çok küçüktü.

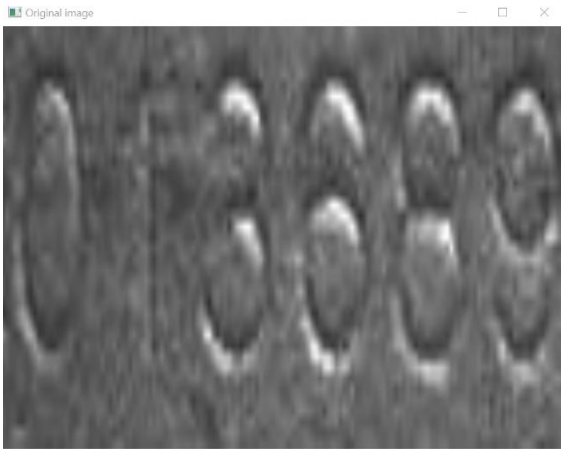
c. OpenCV ve Tesseract Kütüphanesi Kullanılarak Fotoğraftaki Yazı Karakterlerinin Tespiti

Python dili ve OpenCV ve Tesseract kütüphaneleri kullanılarak ilgili proje gerçekleştirilmeye çalışıldı. Proje gerçekleştirilirken ilk olarak fotoğraf kaynakları programa okundu. Sonra openCV kütüphanesi kaynaktan fotoğrafı okurken BGR olarak ters matris şeklinde okuduğu için ilgili fotoğraf dosyası RGB matris haline dönüştürüldü. Daha sonra bu kütüphanelerde bulunan hazır fonksiyonlar kullanılarak yazının tespiti halinde etrafının dikdörtgen ile çevrilmesi için kodlar yazıldı. Bu alan içerisinde kalan yazıların tespiti ve tespit edilen her bir karakterin altına ilgili karakterin ekrana bastırılması için yine hazır fonksiyonlar bu kütüphanelerden temin edildi ve kullanıldı. Bu yöntemin başarsız olmasının nedeni fotoğraftaki tespit edilmesi gereken yazı karakterlerinin rengi arka plan rengiyle aynı olduğu için hazır kütüphane fonksiyonları bunları tanımakta yetersiz kaldı ve dataset'teki fotoğraflar çok küçüktü.

d. Tesseract Kütüphanesi Kullanılarak Metal Yüzeyindeki Kabartmalı Yazı Karakterlerinin Okunması

Python dili ve OpenCV, Tesseract, Numpy, imutils kütüphaneleri kullanılarak ilgili proje gerçekleştirilmeye çalışıldı. Projede ilk olarak resim programa okundu. Sonra fotoğraf grayscale hale getirildi ve üzerindeki noise'ın kaldırılması blur eklendi. Sonra fotoğraftaki yazı ve benzer sivri kenarlı objelerin etrafındaki köşeler tespit edildi. Sonra dilation filtresi ve erosion filtresi sırayla uygulandı. Daha sonra konturlar bulundu ve çizildi ve Otsu thresholding uygulandı. Daha sonra tesseract kütüphanesi OCR uygulaması için çağrıldı ve yazı karakterleri okunmaya çalışıldı. Bu yöntemin başarsız olmasının nedeni fotoğraftaki tespit edilmesi gereken yazı karakterlerinin rengi arka plan rengiyle aynı olduğu ve dataset'teki fotoğraflar çok küçük olduğu için hazır kütüphane fonksiyonları bunları tanımakta yetersiz kaldı ve yapılan filtrelere rağmen güzel ve okunaklı görüntüler elde edilemedi.

- Orijinal Fotoğraf

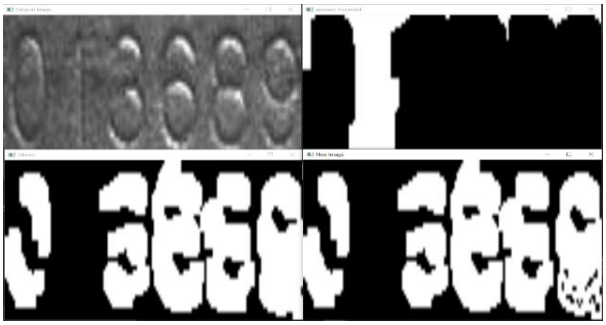


- Dilated Uygulanmış Fotoğraf

- Inverted Threshold Uygulanmış Fotoğraf

- Sürecin Tamamlandığı Final Fotoğraf

- Adımların Karşılaştırmalı Olarak Gösterimi



e. Knn kullanılarak harflerin ve sayıların tek tek tespit edilip analizlerinin yapılması.

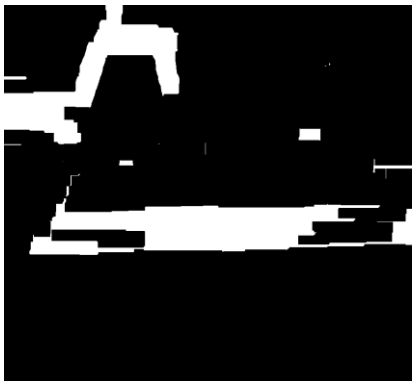
Python dili ve OpenCv kütüphaneleri kullanılarak proje geliştirilmeye çalışıldı.

Burada yapmamız gereken ilk işlem fotoğraflarda sadece yazının olduğu kısmı tespit etmek. Yani fotoğrafta metal parçası dışındaki kısımları kesmemiz gerekiyor. Bu işlemlerin başında main() fonksiyonunun içinde exception kontrolü yapılıyor. Bir hata varsa bu yazdırılıyor, yoksa begin() fonksiyonu çalışıyor ve işlemlerimiz başlıyor. Fotoğrafımız dosyadan okunuyor ve img adında bir değişkene atılıyor. Imutils kütüphanesi ile fotoğrafımızı genişliği 900 piksel olacak şekilde ayarlanıyor. Fotoğrafımızı bu işlemden sonra dosyamıza kaydediyoruz.

Renk paletini gri hale getiriyoruz. Gauss bulanıklaştırma yöntemi uygulanıyor. Kamera sensörü nedeniyle resimde parazit oluşabilmektedir. Gaus yöntemini uygulama sebebimiz de fotoğraftaki gürültüyü azaltılarak bu parazitlerin önüne geçmektir. Her bir elemanı 1 olan kernel adında 25'e 25'lik bir matris oluşturuyoruz. Dilate operatörüne blurlanmış son görüntümüzü ve matrisimizi veriyoruz. Bu operatör giriş olarak verilen görüntü üzerinde parametreler ile verilen alan içerisindeki sınırları genişletmektedir. Bu genişletme sayesinde piksel grupları büyür ve pikseller arası boşluklar küçülür. Blurlanmış fotoğraf gray adlı nesnemize atılıyor. Numpy yardımıyla manuel yapılandırma elemanı oluşturmaya yarar çekirdeğin şeklini ve boyutunu giriyoruz ve matris içi tamamen 1 dolduruyor. MorphologyEx fonksiyonu yardımıyla giriş görüntüsü ve görüntünün açıldıktan sonraki halini elde ediyoruz. Sobel filtresi yardımıyla kenarları tespit ediyoruz. 1. sırada fotoğraf, 2. sırada image matrisindeki dışta kalan piksellerin nasıl üstesinden gelineceği işlemi yapılıyor. Sobel

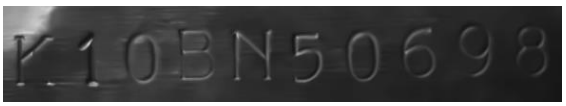
fonksiyonuna verdiğimiz parametreler sayesinde dikey yönde kenarları tespit ediyoruz.

Daha sonra fotoğrafı kaydedip devam ediyoruz. MorphologyEx ve Threshold fonksiyonları yardımıyla dilation ile genişletilen fotoğrafta, yazının olabileceği yatay bölgelerin belirtildiği fotoğraf elde ediliyor.



Burada kullandığımız FindContours yöntemi sebebiyle fotoğraf kullanılamaz hale geliyor bu yüzden fotoğrafımızı yedekliyoruz. Konturlar aynı renk ve yoğunluğa sahip olan tüm kesintisiz noktaları sınır boyunca birleştiren bir eğri olarak basit bir şekilde açıklanabilir. Çeşitli fonksiyonlar sayesinde en büyük kontur alanını tespit ediyoruz. Çünkü kontur alanının en büyük olduğu yerlerde yazı olma ihtimali en yüksek oluyor. Bu alanın sınırları tespit edilir ve fotoğraf bu sınırlara göre kesilir, kaydedilir ve yazdırılır.

Fotoğrafın gürültüsü azaltılır ve kaydedilir. Son hali aşağıdaki fotoğraftaki gibidir.



Fotoğrafın gürültüsü tekrar azaltılır ve artık kenarların yerleri canny kenar tespit yöntemiyle tespit edilir.

2'den 3'e geçerken yapılan morphologyEx yöntemi ve dilation işlemleri uygulanır anca burada bu işlemler dikey yön için yapılmıştır. Harflerin yerlerini bu şekilde tespit edilmiş oluyor. Beyaz alanlar harflerin olduğu yerlere denk düşüyor.

Bu alanlar yani harfler tek tek kesilerek beyaz arka plan ile birlikte dosyaya kaydediliyor. Tespit edilen her harf Knn modelinin çalıştığı sınıfa gönderiliyor ve hangi harf veya sayı olduğu tespit ediliyor. Sonuçlar belirlenmiş karakterlerin listesi aktarılıyor ve liste programın sonunda çıktı olarak. Bu modelin problemi ise sadece bu fotoğraf ve buna çok benzeyen diğer fotoğraf için çalışmasıdır.

4. Bulduğumuz Çözümler

a. Tesseract Ocr Kullanılarak Yazı Tespitinin Gerçekleştirilmesi

Bu çözümü yaparken daha önceki denemelerimizden faydalandık. İlk önce OCR teknolojisini araştırdık ve nasıl gerçekleştirildiğini öğrendik. OCR kütüphanelerini incelediğimiz zaman bize uygun olanın Tesseract olduğuna karar verdik. Veri setimiz kalitesiz olduğundan dolayı internette kaliteli fotoğraflar aradık ve sadece 3 tane kaliteli ve uygun olan fotoğraf bulabildik.

Öncelikle Çözümümüzde fotoğraflardaki yazının konumunu tespit etmek gerekiyordu. Bu yüzden onunla ilgili detectTextForCrop() methodu hazırlandı. Kullandığımız fotoğrafın orijinal hali aşağıda bulunmaktadır.

detectTextForCrop() methodu öncelikle fotoğrafın boyutunu hesaplıyor eğer genişlik 180'den büyük ve yükseklik 60'tan büyük ise kırpma işlemi yapıyor. Kırpma işlemi yaparken ilk önce genişliği 90'a ayarlıyor. Daha sonra renk paletini gri hale getiriyor ve Gauss bulanıklaştırma yöntemi uygulanıyor. Dilate operatörüne blurlanmış son görüntümüzü veriyoruz. Bu operatör giriş olarak verilen görüntü üzerinde parametreler ile verilen alan içerisindeki sınırları genişletmektedir. Bu genişletme sayesinde piksel grupları büyür ve pikseller arası boşluklar küçülür. Blurlanmış fotoğraf gray adlı nesnemize atılıyor. Numpy yardımıyla manuel yapılandırma elemanı oluşturmaya yarar çekirdeğin şeklini ve boyutunu giriyoruz ve matris içi tamamen 1 dolduruyor. MorphologyEx fonksiyonu yardımıyla giriş görüntüsü ve görüntünün açıldıktan sonraki halini elde ediyoruz. Sobel filtresi yardımıyla kenarları tespit ediyoruz. 1. sırada fotoğraf, 2. sırada image matrisindeki dışta kalan piksellerin nasıl üstesinden gelineceği işlemi yapılıyor. Sobel fonksiyonuna verdiğimiz parametreler sayesinde dikey yönde kenarları tespit ediyoruz.

Daha sonra MorphologyEx ve Threshold fonksiyonları yardımıyla dilation ile genişletilen fotoğrafta, yazının olabileceği yatay bölgelerin belirtildiği fotoğraf elde ediliyor ve yazdırılıyor. Burada kullandığımız FindContours yöntemi sebebiyle fotoğraf kullanılamaz hale geliyor bu yüzden fotoğrafımızı yedekliyoruz. Konturlar aynı renk ve yoğunluğa sahip olan tüm kesintisiz noktaları sınır boyunca birleştiren bir eğri olarak basit bir şekilde açıklanabilir. Çeşitli fonksiyonlar sayesinde en büyük kontur alanını tespit ediyoruz. Çünkü kontur alanının en büyük olduğu yerlerde yazı olma ihtimali en yüksek oluyor. Bu alanın sınırları tespit edilir ve fotoğraf bu sınırlara göre kesilir, kaydedilir ve yazdırılır. En son aşağıdaki fotoğrafı elde ediyoruz.

- Kırpılmış Hali:

Kırpma işlemleri bittikten sonra yazının tespit edilmesi için detectLettersFromImage() isimli bir method hazırladık.

Method içerisinde ilk önce fotoğraftaki gürültüyü gidermek için cv2.fastNlMeanDenoisingColored() kullanarak fotoğrafı bulanıklaştırıyoruz. Daha sonra renk paletini gri yapıyoruz.

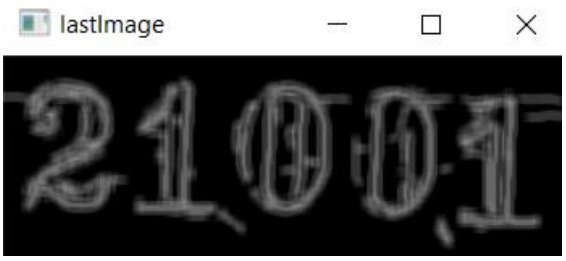
- Bulanıklaşmış ve Renk paleti değişmiş hali:

Daha sonra kenarlarını belirginleştirmek cv2.Canny uyguluyoruz. Canny'in ilk threshold değerini 40 ikincisini ise 20 olarak ayarlıyoruz.

- Canny uygulanmış hali:

Daha sonra oluşan görüntüdeki gürültülerin azaltılması için tekrar bulurlama işlemi uyguluyoruz ama bu bulurlama işlemini daha farklı şekilde yapıyoruz. Kendi ayarladığımız bir kernel matrisi kullanarak bulanıklaştırıyoruz.

- Bulanıklaştırma uygulanmış hali:



Fotoğrafın son hali her ne kadar göze hoş gelmese de fotoğraf artık Tesseract tarafından tespit edilebilir hale gelmiştir. Tesseract'ın tespit edebilmesi için image_to_string() fonksiyonunu çalıştırmak yeterlidir. Son olarak çıkan yazıyı konsola çıktı olarak vermekteyiz.

b. Yolov4 Kullanılarak Yazı Tespitinin Sağlanması

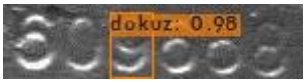
Yolov4'ü seçme sebebimiz rakiplerine kıyasla en başarılı olan olması. Makesense.ai adlı internet sitesini kullanarak yolo4 formatında fotoğraf etiketledik. Test için 31 fotoğraf kullandık. Veri setimizde 300 adet fotoğraf vardı, bu fotoğrafları veri madenciliği dersinde öğrendiğimiz bilgilere göre temizledik.

Eğitim için ise 125 fotoğraf kullandık. Toplamda model için 156 fotoğraf kullandık yaklaşık 25 fotoğrafta kendimizin modeli test etmesi için ayırdık. Modeli eğitmek için Google colab kullandık. En çok yaşadığımız problem Google colab kullanırken işlemci kullanma hakkımızın bitmesiydi. Bunun için birden fazla hesap açtık.

Kodlarımız Google Colab içerisinde çalıştırdık yani bulut ortamında çalıştık. Veri setimiz çok kaliteli ve fazla olmadığından dolayı pek başarılı olamadık. Ama sonuç olarak bu problemin çözümünün bu şekilde sağlanabileceğini kanıtlamış olduk.

Modelin tespit ettiği fotoğraflar:

- Tespit etmesi gereken sayı: 699668



- Tespit etmesi gereken sayı: 913867

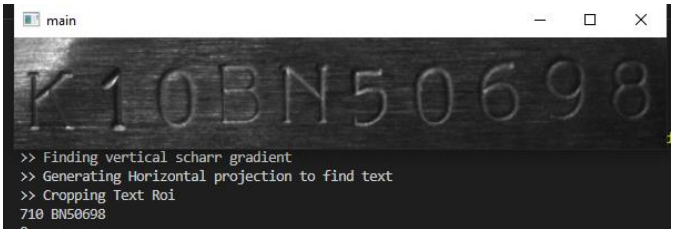
- Tespit etmesi gereken sayı: 139708

- Tespit etmesi gereken sayı: 478378

5. Deneysel Sonuçlar

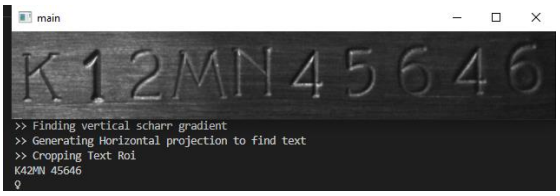
- Orijinal Fotoğraf

- Elde ettiğimiz Sonuç



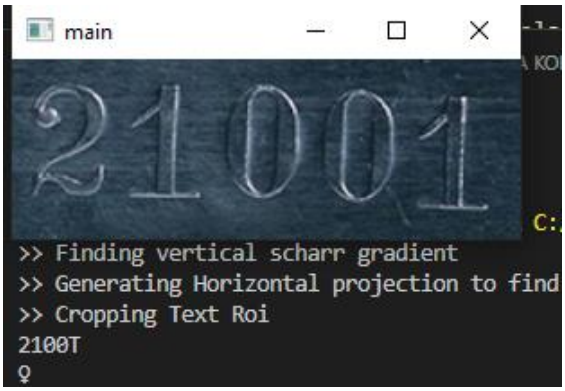
- Orijinal Fotoğraf

- Elde ettiğimiz Sonuç



- Orijinal Fotoğraf

- Elde ettiğimiz Sonuç



6. Kaynakça

- https://www.youtube.com/watch?v=6DjFscX4I_c
- https://pyimagesearch.com/2020/09/14/getting-started-with-easyocr-for-optical-character-recognition/#:~:text=EasyOCR%2C%20as%20the%20name%20suggests,with%20a%20single%20pip%20command
- https://github.com/JaidedAI/EasyOCR
- https://www.youtube.com/watch?v=ZVKaWPW9oQY
- https://towardsdatascience.com/how-to-extract-text-from-images-with-python-db9b87fe432b
- https://www.youtube.com/watch?v=4DrCIVS5U3Y
- https://www.youtube.com/watch?v=9nUNPrvCFAE
- https://stackoverflow.com/questions/29380355/what-thresholding-binarization-algorithm-is-used-in-tesseract-ocr
- https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/#:~:text=Thresholding%20is%20a%20technique%20in,maximum%20value%20(generally%20255)