

İSTANBUL MEDENİYET ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİL370 GÖRÜNTÜ İŞLEME

Dr. Öğr. Üyesi Nurullah ÇALIK

Proje ID: 10

Proje Konusu: Görüntüdeki Metinlerin Tespiti ve Analizi

Ayşenur YÖRÜR 22120205384

Merve MUTLUER 23120205100

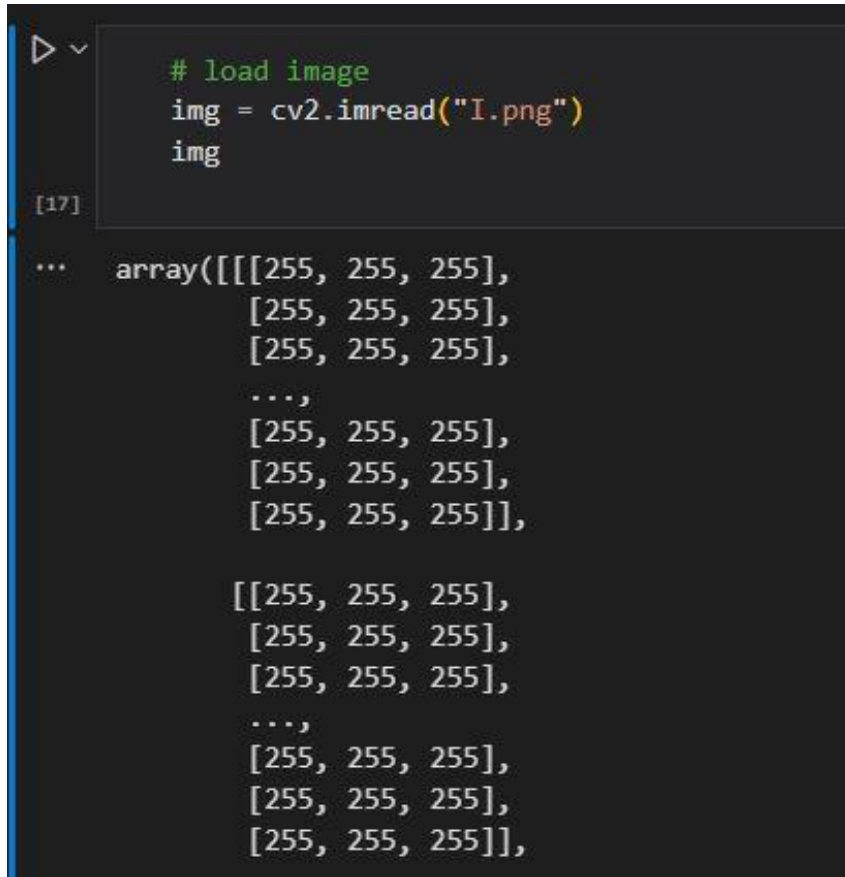
Özet

Bu proje, RGB formatında bir görüntü olan "I.png" dosyasındaki metinlerin otomatik olarak tespit edilmesi ve analiz edilmesi üzerine odaklanmaktadır. Projede, görüntü işleme teknikleri kullanılarak görüntüdeki satırların ve her satırdaki harflerin sayılması gerçekleştirilmiştir. Çalışmada noktalama işaretleri harf olarak ele alınmıştır.

1. Yöntem

1.1. Görüntüyü Yükleme

Şekil 1’de görüldüğü üzere, "I.png" isimli dosya okunmakta ve bu dosya *img* adlı bir değişkene yüklenmektedir. *cv2.imread* fonksiyonu, belirtilen dosya yolundaki görüntüyü okur ve bu görüntüyü bir NumPy dizisi olarak döndürür. Bu NumPy dizisi, görüntünün piksel değerlerini içerir.



```
# load image
img = cv2.imread("I.png")
img
```

[17]

```
... array([[255, 255, 255],
          [255, 255, 255],
          [255, 255, 255],
          ...,
          [255, 255, 255],
          [255, 255, 255],
          [255, 255, 255]],

          [[255, 255, 255],
          [255, 255, 255],
          [255, 255, 255],
          ...,
          [255, 255, 255],
          [255, 255, 255],
          [255, 255, 255]])
```

Şekil 1. Dosya Okuma İşlemi

1.2. Görüntü İşleme Teknikleri

1.2.1. Morfolojik İşlemler

img adlı değişkende tuttuğumuz görüntü gri tonlamalı bir görüntüye dönüştürülmüştür ve elde edilen gri tonlamalı görüntüye ikili (binary) eşikleme işlemi uygulanmıştır. Bu eşikleme işlemi OpenCv'nin resmi dökümanında yer alan Otsu's Thresholding[1] metodu kullanılmıştır. Yapılan işlem, *grayscale* ve *thresh* görüntüleri şekil 2'de verilmiştir.



Şekil 2. Grayscale ve Thresh Görüntüleri

Yataydaki bulanıklığı kaldırmak için morfolojik erozyon kullanılmıştır. Nokta çizgilerinden ince çizgileri kaldırmak için de morfolojik açma kullanılmıştır. Bu konuda github üzerinden yapılan başka çalışmalar incelenmiş ve test edilmiş olup [2]'de yapılan çalışmadaki morfolojik işlemler ile çok daha düzgün bir sonuç elde edilmiştir. Elde edilen sonuçlar şekil 3'te gösterilmiştir.



Şekil 3. Morfolojik İşlemler

1.2.2. Yatay Projeksiyon ile Satır Sayısı Tespiti

Yatay projeksiyon, bir görüntünün her satırındaki beyaz (1) piksellerin toplamını hesaplayarak elde edilen bir vektördür. Bu vektör, her satırın ne kadar "aktif" olduğunu gösterir. Beyaz piksellerin olduğu satırlar yüksek değerlere sahipken, sadece siyah (0) piksellerden oluşan satırlar sıfır değerine sahip olacaktır.

Örnek bir ikili görüntü (5x10 matris) ele alalım:

```
morph = np.array([
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Satır 0
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 1
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 2
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Satır 3
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 4
])
```

`np.sum(morph, axis = 1)` satırdaki toplam 1 sayısını dönecektir. Bu örneğin gerçekleştirilmesi şekil 4'te verilmiştir.

```
morph = np.array([
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Satır 0
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 1
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 2
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Satır 3
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0], # Satır 4
])

horizontal_sum = np.sum(morph, axis=1)

horizontal_sum
....
horizontal_sum = [0, #0. satirdaki toplam 1 sayisi
                  8, #1. satirdaki toplam 1 sayisi
                  8, #2. satirdaki toplam 1 sayisi
                  0, #3. satirdaki toplam 1 sayisi
                  8] #4. satirdaki toplam 1 sayisi
....

array([0, 8, 8, 0, 8])
```

Şekil 4. Yatay Projeksiyon Toplam

Görülmemekte olduğu üzere bazı değerler 0 olmuştur yani bu satırda hiç 1 (beyaz renk) yok demektir. `rows = np.where(horizontal_sum > 0)[0]` satırı aracılığı ile içinde en az bir tane

beyaz renk bulunan satırlar bulunmaktadır ve içinde beyaz renk bulunan satırların indexleri rows değişkeni içinde tutulacaktır.

Çalışmamıza bu işlemleri uygularsak şekil 5'te verildiği gibi bir sonuç elde edeceğiz. Bu görseldeki sayıları incelemiş olursak ardışıklığın belli zamanlarda bozulduğunu göreceğiz. Ardışıklığın bozulduğu anlar img resmindeki satır geçişini ifade etmektedir.

```
# Yatay projeksiyon kullanarak satırları tespit et
horizontal_sum = np.sum(morph, axis=1)
rows = np.where(horizontal_sum > 0)[0]

rows
✓ 0.0s

array([ 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
       26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
       39, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
       80, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
       108, 109, 110, 111, 112, 113, 114, 115, 116, 163, 164, 165, 166,
       167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179,
       180, 181, 182, 183, 184, 185, 186, 187, 188, 199, 200, 201, 202,
       203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215,
       216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228,
       229, 230, 231, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249,
       250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262,
       263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 287, 288, 289,
       290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302,
       303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 328, 329,
       330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342,
       343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 365,
       366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378,
       379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391,
       392, 393, 394, 395, 396, 411, 412, 413, 414, 415, 416, 417, 418,
       419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431,
       432, 433, 434, 435, 436, 437, 446, 447, 448, 449, 450, 451, 452,
       453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465,
       466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477],
      dtype=int64)
```

Şekil 5. Beyaz Renk Bulunan Satır Indexleri

Şekil 5'teki görselde görüldüğü üzere satır aralıklarının başlangıç ve bitişlerini görebiliyoruz. Kırmızı ile çizdiğim [13,39] ilk satırın, yeşil ile çizdiğim [55,80] ikinci satırın, sarı ile çizdiğim [96,116] üçüncü satırın başlangıç ve bitişini temsil etmektedir. Bu amaçla çalışmamızda şekil 6'daki işlemleri gerçekleştirmekteyiz.

```

# Satır aralıklarını belirle
row_intervals = []
satir_baslangic = rows[0]
for i in range(1, len(rows)):
    if rows[i] != rows[i-1] + 1: #eğer ardışık değilse
        row_intervals.append((satir_baslangic, rows[i-1]))
        satir_baslangic = rows[i]
row_intervals.append((satir_baslangic, rows[-1]))

row_intervals
✓ 0.0s

[(13, 39),
 (55, 80),
 (96, 116),
 (163, 188),
 (199, 231),
 (240, 272),
 (287, 313),
 (328, 354),
 (365, 396),
 (411, 437),
 (446, 477)]

```

Şekil 6. Satır Aralıkları Tespiti

Şekil 6’da gösterilen row_intervals listesi içerisinde tespit edilmesi istenen satır için satırın başlangıç index’i ve o satırın bitiş indexi’değerlerini tutmaktadır. Şekil 7’de görüldüğü gibi row_intervals listesinin uzunluğu bize toplamda kaç satır olduğunu vermektedir.

```

# Toplam satır sayısını yazdır
print(f"Toplam satır sayısı: {len(row_intervals)}")
✓ 0.0s

Toplam satır sayısı: 11

```

Şekil 7. Toplam Satır Sayısı

1.2.3. Dikey Projeksiyon ile Harf Sayısı Bulma


Şekil 8’deki gösterilen işlem ile her satır kendi içerisinde incelenecek ve her satır için harf sayısı bulunacaktır.

```

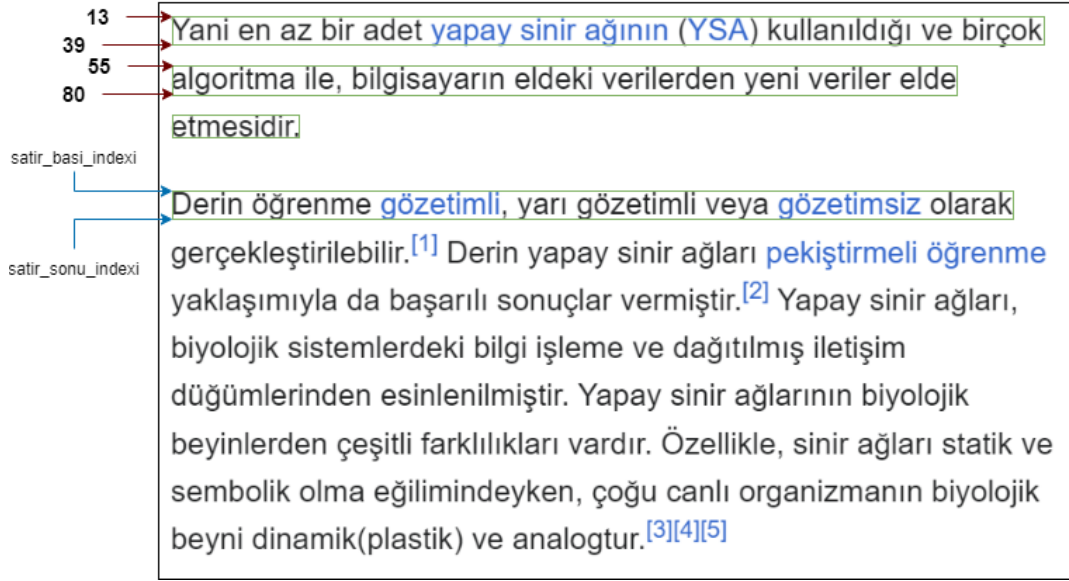
# Her satırdaki harf sayısını bulma
for row_num, (start_row, end_row) in enumerate(row_intervals):

    # Satırın görüntü kesitini oluştur. Böylece her satır ayrı ayrı incelenecektir
    row_image = thresh[start_row:end_row+1, :]

```



Şekil 8. Satır Görüntü Kesiti



Şekil 9. Satir Analizi

Çalışmamızın bir önceki bölümünde yatay projeksiyon analizi detaylı olarak anlatılmış ve Şekil 9’da görülen şekilde satırlar elde edildi. Bu aşamada elde edilen her satır için ayrı ayrı dikey projeksiyon analizi uygulanacaktır.

```
# Dikey projeksiyon kullanarak harfleri tespit et
vertical_sum = np.sum(row_image, axis=0)
cols = np.where(vertical_sum > 0)[0]

# Harf aralıklarını belirle
letter_intervals = []
start_col = cols[0]
for j in range(1, len(cols)):
    if cols[j] != cols[j-1] + 1:
        letter_intervals.append((start_col, cols[j-1]))
        start_col = cols[j]
letter_intervals.append((start_col, cols[-1]))
```

Şekil 10. Satir Kesiti için Dikey Projeksiyon Uygulaması

Şekil 10’da verilen işlemler kullanılarak satır kesiti için dikey projeksiyon analizi yapılır. vertical_sum düşeyde siyah renk olup olmadığını incelemektedir. Satır 11’de anlatılmak istendiği üzere siyah her sütun için düşeyde renk değerleri toplanmıştır ve eğer 0 değilse (beyaz renk içeriyorsa) sütun başlangıç değeri daha sonra da sütun bitiş değeri kaydedilir.



Şekil 11. Dikey Projeksiyon

```
# Her satırdaki harf sayısını bulma
for row_num, (start_row, end_row) in enumerate(row_intervals):

    # Satırın görüntü kesitini oluştur. Böylece her satır ayrı ayrı incelenecektir
    row_image = thresh[start_row:end_row+1, :]

    # Dikey projeksiyon kullanarak harfleri tespit et
    vertical_sum = np.sum(row_image, axis=0)
    cols = np.where(vertical_sum > 0)[0]

    # Harf aralıklarını belirle
    letter_intervals = []
    start_col = cols[0]
    for j in range(1, len(cols)):
        if cols[j] != cols[j-1] + 1:
            letter_intervals.append((start_col, cols[j-1]))
            start_col = cols[j]
    letter_intervals.append((start_col, cols[-1]))

    # Satırdaki harf sayısını yazdır
    print(f"Satır {row_num + 1} - Harf sayısı: {len(letter_intervals)}")
```

✓ 0.0s

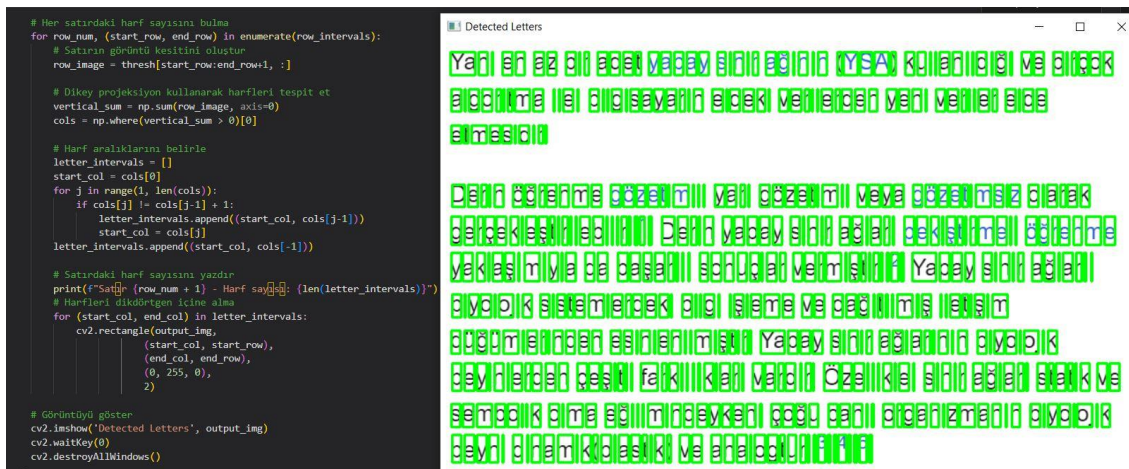
Satır 1 - Harf sayısı: 55
Satır 2 - Harf sayısı: 56
Satır 3 - Harf sayısı: 10
Satır 4 - Harf sayısı: 55
Satır 5 - Harf sayısı: 64
Satır 6 - Harf sayısı: 59
Satır 7 - Harf sayısı: 52
Satır 8 - Harf sayısı: 55
Satır 9 - Harf sayısı: 66
Satır 10 - Harf sayısı: 56
Satır 11 - Harf sayısı: 42

Şekil 12. Her Satır için Harf Sayısı Çıktılar

Yapılan işlemler sonucu Şekil 12’de gösterilen çıktı elde edilmiştir.

1.2.4. Görselleştirme

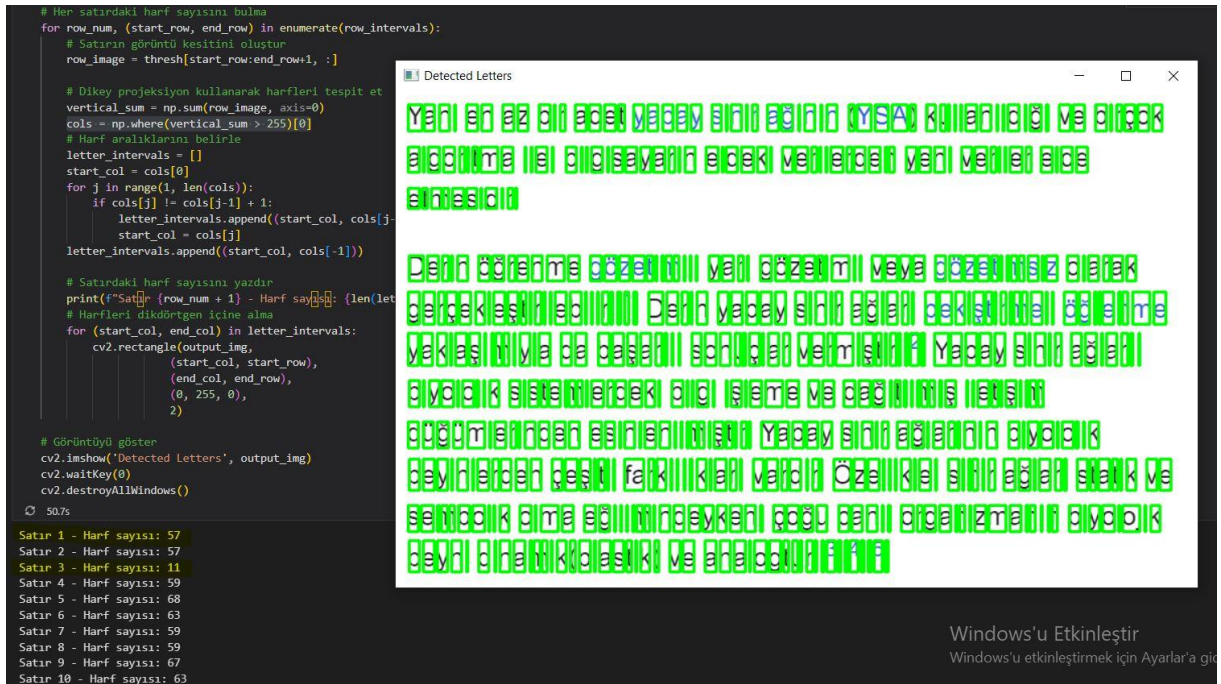
Tespit edilen harfler şekil 13’teki gibi görselleştirilmiştir.



Şekil 13. Tespit Edilen Harflerin Görselleştirilmesi

2. Sonuç ve Tartışma

Çalışmamız farklı renk tonlarında test edilmiştir ve sorun yaşanmamaktadır. “Ya” harflerinin kullanımı durumlarında sorun yaşanmaktadır. Pikseller incelendiğinde Y harfinin kuyruğu a harfine düşmektedir dolayısı ile dikey projeksiyonda boşluk tespit edilememektedir. Bu amaçla `cols = np.where(vertical_sum > 255) [0]` şeklinde test edilmiştir. 255 dışında diğer farklı değerler de test edildi lakin değişimin görüldüğü nokta 255’tir.



Şekil 14. Test

Şekil 14'te görüldüğü üzere “Ya” artık ayrı ayrı harf olarak tespit edilmiştir. Lakin artık “m” gibi bazı harfler iki kere sayılmaktadır. Dolayısı ile doğru bir çözüm olmamıştır. Bu problemi çözmek amacı ile morfolojik işlemler ile görüntü üzerinde oynama yapılarak bu sorun belki çözülebilir lakin bu işlemler farklı ayrıntıların kaybolmasına da sebep olabilir. Dolayısı ile çalışmamızı Şekil 13'te elde ettiğimiz sonuç ile bırakmaya karar verdik.

3. Kaynaklar

[1] https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[2] <https://github.com/srividya-p/TIRF-Gujrati-Script-Recognition/blob/59fce111a179a954e7289e825944997cf6db24d2/detect-lines/lines.py#L14>

4. Ekler

[1] proje.ipynb ek olarak gönderilmiştir. Gerekli kodları içermektedir. Kodlar aşağıdaki bölüme de eklenmiştir.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# load image
img = cv2.imread("I.png")
# convert to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# threshold the grayscale image
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)[1]

# Görüntüleri matplotlib ile plot etme
plt.figure(figsize=(10, 5))

# Grayscale görüntü
plt.subplot(1, 2, 1)
plt.title('Grayscale Image')
plt.imshow(gray, cmap='gray')
plt.axis('off')

# Eşiklenmiş görüntü
plt.subplot(1, 2, 2)
plt.title('OTSU\'s Thresholded Image')
plt.imshow(thresh, cmap='gray')
plt.axis('off')

plt.show()

# use morphology erode to blur horizontally
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (151, 3))
morph1 = cv2.morphologyEx(thresh, cv2.MORPH_DILATE, kernel)

# use morphology open to remove thin lines from dotted lines
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 17))
morph = cv2.morphologyEx(morph1, cv2.MORPH_OPEN, kernel)

# Görüntüleri matplotlib ile plot etme
plt.figure(figsize=(20, 5))

# Grayscale görüntü
plt.subplot(1, 3, 1)
```

```

plt.title('Threshold Edilmiş Görüntü')
plt.imshow(thresh, cmap='gray')
plt.axis('off')

# Eşiklenmiş görüntü
plt.subplot(1, 3, 2)
plt.title('Morfolojik Erzonyon')
plt.imshow(morph1, cmap='gray')
plt.axis('off')

# Eşiklenmiş görüntü
plt.subplot(1, 3, 3)
plt.title('Morfolojik Açma')
plt.imshow(morph, cmap='gray')
plt.axis('off')

plt.show()

# Yatay projeksiyon kullanarak satırları tespit et
horizontal_sum = np.sum(morph, axis=1)
rows = np.where(horizontal_sum > 0)[0]
# Satır aralıklarını belirle
row_intervals = []
satir_baslangic = rows[0]
for i in range(1, len(rows)):
    if rows[i] != rows[i-1] + 1: #eğer ardışık değilse
        row_intervals.append((satir_baslangic, rows[i-1]))
        satir_baslangic = rows[i]
row_intervals.append((satir_baslangic, rows[-1]))
# Toplam satır sayısını yazdır
print(f"Toplam satır sayısı: {len(row_intervals)}")
# GÖRSELLEŞTİRME
# Her satırdaki harf sayısını bulma
output_img = img.copy()
# Her satırdaki harf sayısını bulma
for row_num, (start_row, end_row) in enumerate(row_intervals):
    # Satırın görüntü kesitini oluştur
    row_image = thresh[start_row:end_row+1, :]

    # Dikey projeksiyon kullanarak harfleri tespit et
    vertical_sum = np.sum(row_image, axis=0)
    cols = np.where(vertical_sum > 0)[0]
    # Harf aralıklarını belirle
    letter_intervals = []
    start_col = cols[0]
    for j in range(1, len(cols)):
        if cols[j] != cols[j-1] + 1:
            letter_intervals.append((start_col, cols[j-1]))
            start_col = cols[j]
    letter_intervals.append((start_col, cols[-1]))

```

```
# Satırdaki harf sayısını yazdır
print(f"Satır {row_num + 1} - Harf sayısı: {len(letter_intervals)}")
# Harfleri dikdörtgen içine alma
for (start_col, end_col) in letter_intervals:
    cv2.rectangle(output_img,
                  (start_col, start_row),
                  (end_col, end_row),
                  (0, 255, 0),
                  2)

# Görüntüyü göster
cv2.imshow('Detected Letters', output_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```