

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT R

TEHNIČKA DOKUMENTACIJA

Sustav za procesiranje dokumenata pisanih pisaćom mašinom

Student
Lovro Magdić

Nastavnik
Juraj Petrović

Siječanj 15, 2023.

1 Opis razvijenog proizvoda

Cilj ovog projekta bio je razviti sustav za procesiranje dokumenata pisanih pisačom mašinom. Sustav korisniku omogućuje lakše vizualno pregledavanje dokumenata, te računalno iščitavanje s dokumenata. Od korisnika očekuje se datoteka sa slikama nad kojima treba provesti procesiranje, sve ostalo sustav radi automatski. Odabir najboljih parametra, spremanje međurezultata, rotacija i slično.

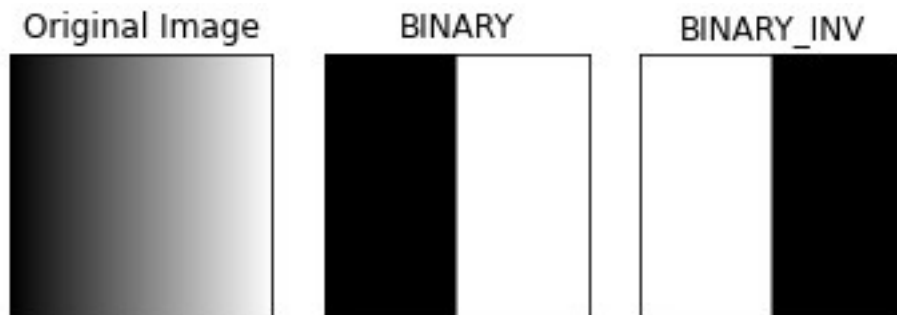
Cijeli proces obrade se odvija se kroz naredbeni redak odnosno u razvojnom okruženju po odabiru korisnika. Za razvijanje sustava korišten je Visual Studio Code (VSCode).

Sustav je razvijen za python3 te zahtjeva razvojnu biblioteku openCV. OpenCV je biblioteka razvojnih funkcija za računalni vid u stvarnom vremenu, njezine funkcionalnosti koristimo za razne obrade, spremanja i čitanja dokumenata.

2 Opis rada sustava

2.1 Binary threshold i Image deskew

U navedenom koraku koristimo navedene funkcije obrade: Binary threshold i Image deskew. Threshold je segmentacijska funkcija pomoću koje postizemo crno-bijelu sliku iz "grayscale" formata ili slike u boji. Preciznije za inicijalni threshold koristimo "Binary Threshold". Kao parametre prima granicu do koje nijanse sive postaju crne i iznad navedene granice nijanse sive postaju bijele.



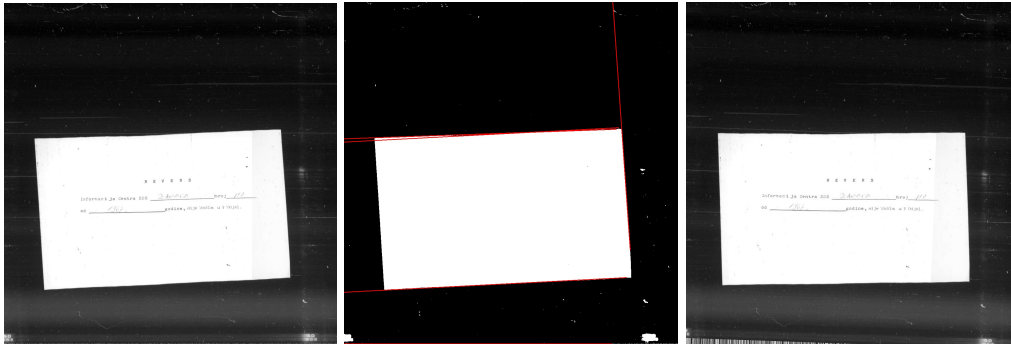
Binary threshold nam omogućuje uklanjanje raznih artefakata koji bi otežali proces "Hough Line Transform".

Hough Line Transform je jednostvani transformacijski proces pomoću kojeg pronalazimo ravne linije na slici, u našem slučaju rubove papira na slici koja je prošla threshold. Nakon detekcije linija, računamo njihovu orijentaciju s obzirom na os apscise i time dobivamo kut zakrivljenosti. Navedeni kut odnosno kutevi nam olakšavaju postizanje uspravnosti papira na slici.

Nakon određenog kuta zakrivljenosti preostaje nam rotacija slike. Za to koristimo definiranu funkciju "rotateImage" koja obavlja rotaciju pomoću integrirane funkcije "warpAffine" iz razvojne biblioteke openCV, slika se rotira oko svoga centra.

```
def rotateImage(cvImage, angle: float):  
    newImage = cvImage.copy()  
    (h, w) = newImage.shape[:2]  
    center = (w // 2, h // 2)  
    M = cv2.getRotationMatrix2D(center, angle, 1.0)  
    newImage = cv2.warpAffine(newImage, M, (w, h), flags=cv2.INTER_CUBIC,  
                             borderMode=cv2.BORDER_REPLICATE)  
    return newImage
```

Rotacija se ne provodi na slici koja je prošla threshold već na originalnoj slici, threshold sliku koristimo za izračun rotacije.

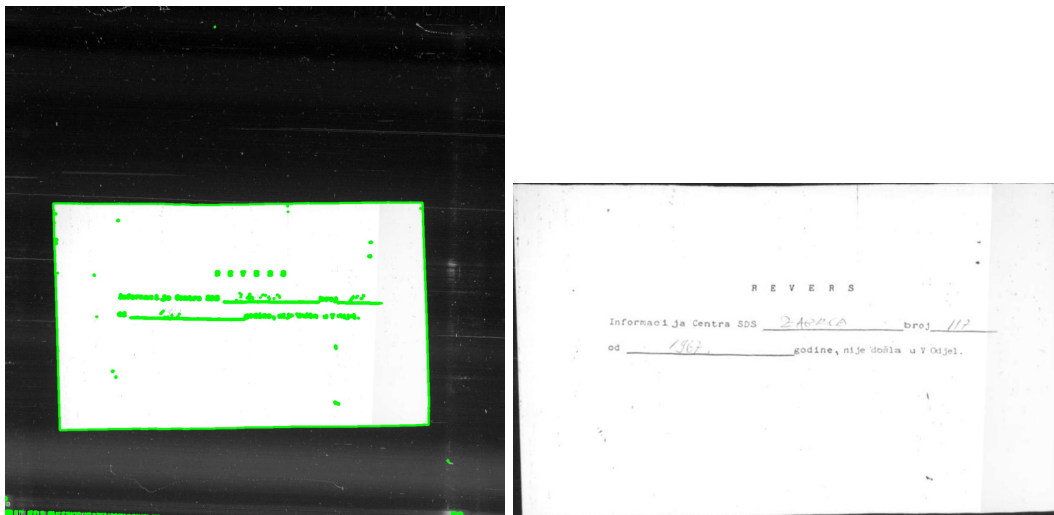


2.2 Contour detection

Sljedeći korak u procesu obrađivanja slika je "contour detection". Bitan je za primjenu "adaptive threshold" funkcije i izuzimanje nepotrebnih artefakata iz slike (crne pozadine i artefakata). Radi na jednostavnom principu označavanja kontinuiranih točaka te lokaliziranje istih u skupinu odnosno konturu.

Detekciju kontura provodimo na threshold slici a konture primjenjujemo na originalnoj odnosno rotiranom originalu.

Kako bi postigli optimalnu konturu za svaku sliku provodimo detekciju optimalnog parametra threshold funkcije za svaku sliku. Koristimo threshold u intervalu (100, 235) i računamo površinu konture koju je detektirao, određujemo razliku maksimalne konture i površine koju je sustav odredio za trenutni threshold. Optimalni threshold imat će najmanju razliku jer u tom slučaju imamo najprecizniju konturu stvarnom papiru na slici.



```

for i in range(100, 235, 5):
    image = cv2.imread(each)
    img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(img_gray, i, 255, cv2.THRESH_BINARY)

    contours, hierarchy = cv2.findContours(image=thresh, mode=cv2.RETR_TREE,
        method=cv2.CHAIN_APPROX_NONE)
    image_copy = image.copy()

    cv2.drawContours(image=image_copy, contours=contours, contourIdx=-1, color=(0, 255,
        0), thickness=2, lineType=cv2.LINE_AA)

    c = max(contours, key = cv2.contourArea)
    x,y,w,h = cv2.boundingRect(c)

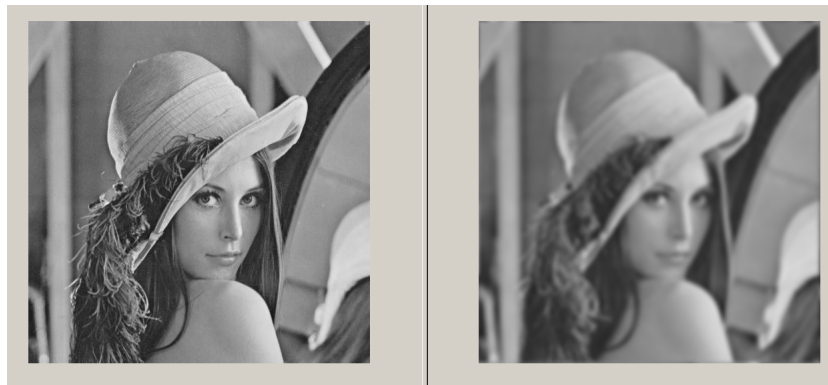
    ar.append([cv2.contourArea(c), w*h, i])
    array.append(int(w*h)-int(cv2.contourArea(c)))

index = array.index(min(array))

```

2.3 Gaussian blur

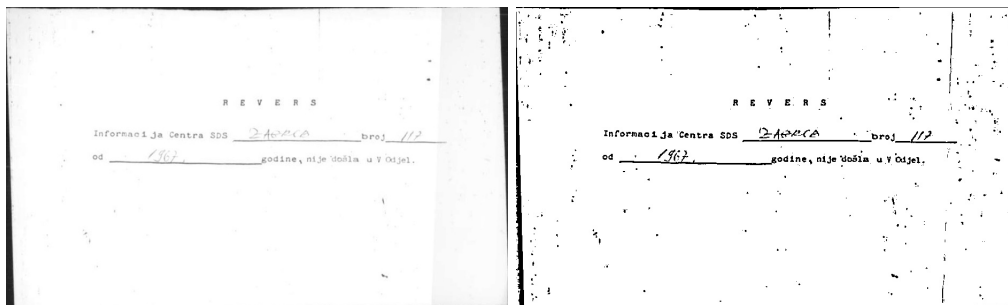
Primjenom Gaussove funkcije u procesu zamućivanja slike smanjujemo detalje i šum prisutan na slici. Postignuti produkt nazivamo "Gaussian blur".



2.4 Adaptive threshold

Slike mogu sadržavati više razina različitog osvjetljenja i to nam otežava određivanje optimalnog parametra thresholda.

Kako bi riješili taj problem koristimo funkciju "adaptive threshold" koja izračunava threshold parametar za piksele s obzirom na susjedne piksele. Time dobivamo različite vrijednosti thresholda za područja različitog osvjetljenja.

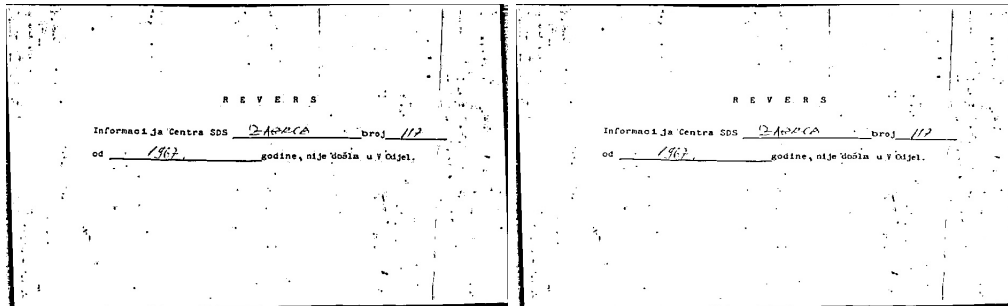


```
img = cv2.imread(each, 0)
image = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                             cv2.THRESH_BINARY, 25, 4)
```

2.5 Font thinning

Kako bi olakšali optičko prepoznavanje znakova primjenjujemo proces stanjivanja fonta kojim također izbacujemo nepotreban šum oko znakova i postizemo detaljniju sliku.

```
def thin_font(image):
    image = cv2.bitwise_not(image)
    kernel = np.ones((2,1), np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.bitwise_not(image)
    return (image)
```



3 Upute za korištenje

Najnovija stabilna verzija dostupna je na [github-u](https://github.com).

```
git clone git@github.com:LovroMagdic/Image-preprocessing.git
```

Sustav je razvijen za python3 te zahtjeva razvojne biblioteke openCV i Tesseract Python.

```
pip install opencv-python
pip install pytesseract
```

Nakon kloniranja repozitorija korisnik postavlja slike nad kojima želi provesti procesiranje u mapu "dataset" te pokreće "image_preprocessing.py". Sustav automatski kreira preostale mape, rezultat obrade nalazi se u mapi "dataset_final" odnosno isčitani podatci u "ocr".

4 Literatura