**Proiect*: Identificarea literelor dintr-o imagine cu text în funcție de formă.(56)***

**Introducere**

Acest proiect utilizează tehnici de procesare a imaginilor și recunoaștere optică a caracterelor (OCR) pentru a extrage text din imagini. Pentru implementare, folosim bibliotecile **OpenCV** pentru preprocesarea imaginilor și **Tesseract OCR** pentru recunoașterea textului.

**Pașii proiectului**

1. **Încărcarea imaginii**

Imaginea este preluată dintr-o sursă externă și citită folosind OpenCV prin funcția **cv2.imread(image\_path)**, care încarcă imaginea într-un format numeric.

1. **Preprocesarea imaginii**

Conversia imaginii în tonuri de gri se face **cu cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**, ceea ce elimină informațiile de culoare și îmbunătățește contrastul.

* + **De ce grayscale?** Imaginile în grayscale sunt mai ușor de procesat în sarcini de viziune computerizată, deoarece au un singur canal (valori de intensitate), spre deosebire de cele RGB care au trei canale. Acest lucru simplifică pașii de preprocesare ulterioare.

Aplicarea unui filtru Gaussian Blur cu **cv2.GaussianBlur(gray, (5, 5), 0)** ajută la reducerea zgomotului, netezind detaliile inutile. Dimensiunea kernel-ului este (5, 5), ceea ce înseamnă că se aplică un filtru 5x5 pe fiecare pixel, iar deviația standard este 0.

* + **De ce aplicăm un blur?** Blurrarea ajută la eliminarea zgomotului mic, care poate interfera cu recunoașterea textului. Aceasta netezește imaginea, păstrând în același timp structura generală a caracterelor.

Binarizarea imaginii utilizând metoda Otsu cu **cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)** transformă imaginea într-o versiune alb-negru, separând clar zonele de text de fundal. Pixelii care au o intensitate mai mare decât un prag sunt setați la alb (255). Pixelii care sunt sub acest prag sunt setați la negru (0).

Metoda Otsu calculează automat pragul optim pentru a minimiza varianța dintre clasele de pixeli.

* + **De ce binarizăm imaginea?** Binarizarea ajută la separarea textului de fundal, făcând recunoașterea mai ușoară pentru Tesseract.

1. **Detectarea contururilor**

**cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)**:

Această funcție găsește toți conturii externe din imaginea binară (thresh). Contururile sunt pur și simplu limitele componentelor conectate (obiecte sau forme) din imagine.

* + **cv2.RETR\_EXTERNAL:** Aceasta recuperează doar contururile externe (ignora contururile interioare).
  + **cv2.CHAIN\_APPROX\_SIMPLE:** Aceasta stochează doar punctele finale ale contururilor, economisind memorie.
  + **De ce detectăm contururi?** Detectarea contururilor ajută la identificarea regiunilor unde ar putea apărea text în imagine. Acestea sunt folosite pentru a desena dreptunghiuri în jurul regiunilor textuale.

Pentru fiecare contur găsit, **cv2.boundingRect(contour)** determină coordonatele unui dreptunghi care încadrează regiunile cu text. Aceasta returnează colțul din stânga sus (x, y) și lățimea (w) și înălțimea (h) dreptunghiului.

Se elimină zonele prea mici pentru a reduce zgomotul, iar contururile sunt desenate cu **cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)** pentru evidențiere.

* + Dacă lățimea și înălțimea unui contur sunt mai mari de 10 pixeli, se desenează un dreptunghi verde ((0, 255, 0)) în jurul conturului, cu o grosime de 2 pixeli.

Această etapă elimină contururile mici, care ar putea fi doar zgomot (de exemplu, pete mici sau semne care nu sunt text).

1. **Recunoașterea textului cu OCR**

Funcția **pytesseract.image\_to\_string(thresh, lang='eng')** analizează imaginea binarizată și extrage textul folosind motorul OCR Tesseract.

Se setează manual calea către executabilul Tesseract prin **pytesseract.pytesseract.tesseract\_cmd**, pentru a asigura detectarea corectă a programului.

1. **Salvarea rezultatelor**

Textul detectat este scris într-un fișier text utilizând **open("detected\_text.txt", "w", encoding="utf-8")** și metoda **.write().**

Imaginea preprocesată și cea cu contururile textului sunt afișate folosind **cv2.imshow()** pentru verificare vizuală.

Utilizatorul poate închide ferestrele cu **cv2.waitKey(0) și cv2.destroyAllWindows().**

**Bibliografie**

* Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.
* Smith, R. (2007). *An Overview of the Tesseract OCR Engine*. *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 629-633. IEEE.
* OpenCV. (n.d.). *Open Source Computer Vision Library*. Retrieved from <https://opencv.org>
* Tesseract OCR. (n.d.). *Tesseract: An Open-Source OCR Engine*. Retrieved from <https://github.com/tesseract-ocr>
* Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.