

Tema 1

ROBERT-LUCIAN SUTO

November 2021

1 Taskul 1

Am inceput prin a adăuga o margine de culoare gri la poza initială pentru a mă asigura că nu apar probleme la găsirea contururilor din poze.

Exemplu: Poza 17 din colecția jigsaw, avea tabla de joc prea în margine, iar conturul nu era selectat bine.

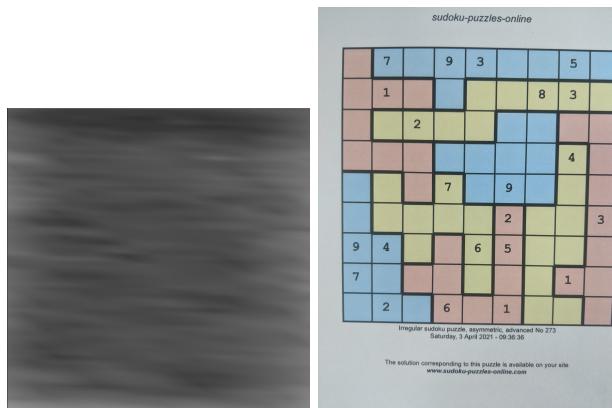


Figure 1: Cel mai mare contur din imaginea 17 fără margini aplicate.

După ce am aplicat un border imaginii, am căutat contururile din imagine, și am selectat cel mai mare contur determinat de 4 puncte. folosind funcția `contur_mare` din fișierul `funcții.py`

```
def contur_mare(contours):
    mare = np.array([])
    max_ = 0
    for i in contours:
        area = cv.contourArea(i)
        if area > 50:
            perimetru = cv.arcLength(i, True)
```

```

        colturi = cv.approxPolyDP(i, 0.02*perimetru, True)
        if area > max and len(colturi) == 4:
            mare = colturi
            max = area
    return mare, max

```

După ce am găsit punctele ce determină tabla, voi da un `getPerspective` și un `warpPerspective`. Aplic un GaussianBlur, și transform tabla de joc într-o imagine alb-negru (nu grayscale) folosind un `threshold` pentru a fi sigur că am doar pixeli alb/negru de intensitate 255/0. Imaginea devine:

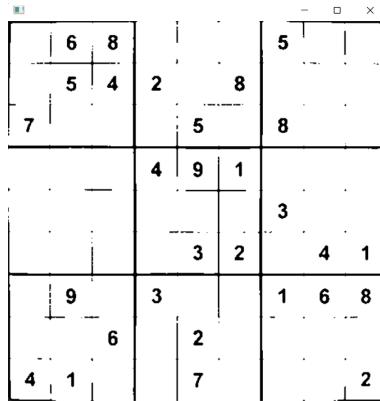


Figure 2: Tabla de joc după modificările de mai sus.

Împart apoi imaginea în 81 de pătrate egale folosind un vertical split și un horizontal split.

```

def patratele(img):
    coloane = np.vsplit(img,9) #despartim imaginea in 9 parti egale
    casute = []
    #parcurgem liniile si le spaltim in 9 dupa coloane pentru a gasi de la stanga la dreapta
    # si de sus in jos casutele
    # impartirea la 9 ar trebui sa fie buna din moment ce un sudoku este 9/9 iar rezolutiile
    for i in coloane:
        randuri = np.hsplit(i,9)
        for casuta in randuri:
            casute.append(casuta)
    return casute

```

Pătratele rezultate după apelarea funcției vor arăta în felul următor.



Figure 3: Exemple de patrate.

În final, verificăm dacă există pixeli de intensitate 0 în pătratele redimensionate (pentru a ne asigura că nu este verificată și marginea carourilor) și afișăm 'x', sau 'o' în funcție de rezultat.

2 Taskul 2

Pentru taskul 2, am început din nou cu adăugarea unei margini gri la poza inițială. Am lucrat cu 2 imagini în paralel:

- Prima, pe care am aplicat operațiile de la taskul 1, pentru a extrage numerele din imagine
- A doua, pe care am aplicat mai multe operații ce vor fi descrise mai jos.

Astfel, am aplicat un **median blur** de mai multe ori pentru a scăpa de contururile subțiri și a rămâne cu cele care delimitizează zonele. De asemenea am aplicat și un **dilate** și un **erode**, pentru a scăpa de linile mai subțiri și apoi pentru a îngroși linile rămase, după care am aplicat un border uniform negru tablei de joc.

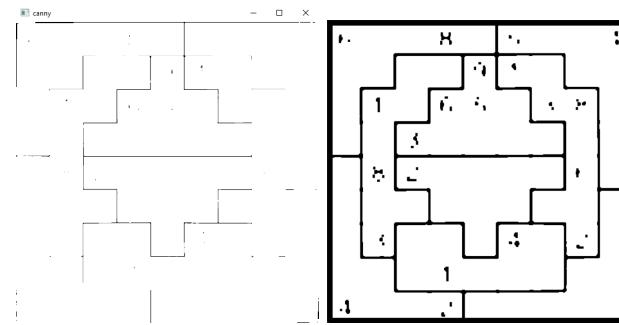


Figure 4: Imaginea 36, înainte și după operațiile de dilate și erode.

Acum că avem conturul fiecărei zone, am apelat funcția **findContours**, iar apoi am sortat aceste contururi stânga dreapta și sus jos folosind cheia:

`key = lambda x: (cv.boundingRect(x)[1] // 10 * 10 * latime_tabla + cv.boundingRect(x)[0])`

Unde `tabla_joc` este a doua imagine din Figura 4.

După sortare, folosind funcția **drawContours** colorez fiecare imagine după un vector de culori care reprezintă ordinea. Astfel imaginea mea devine:

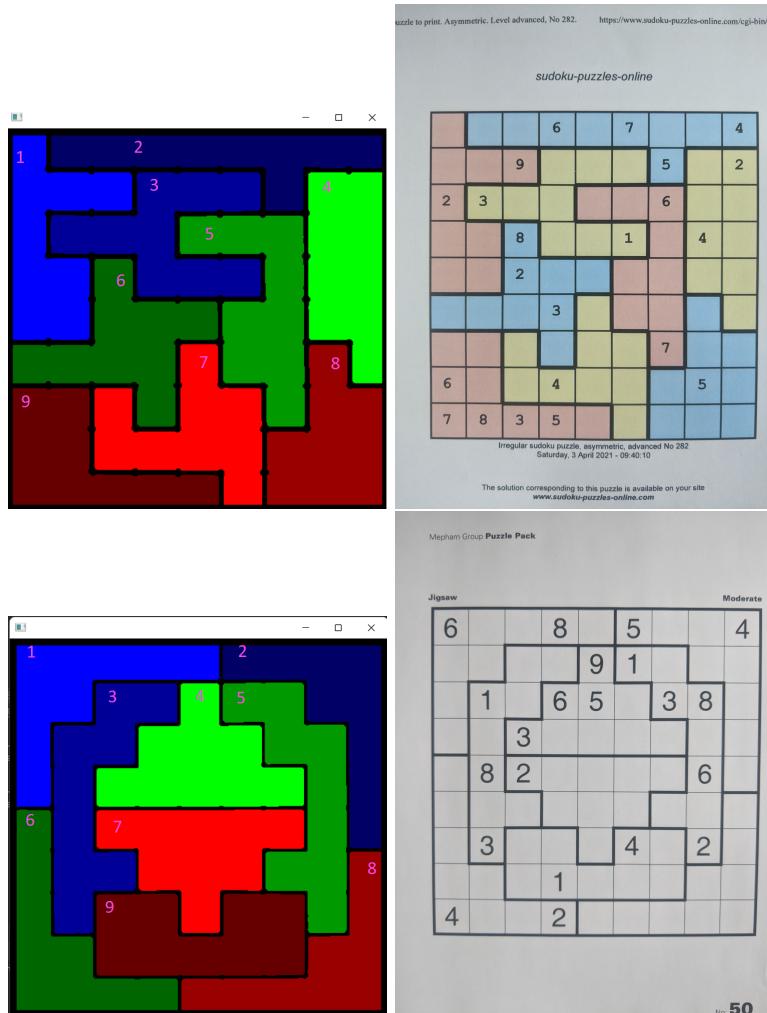


Figure 5: Imagini colorate, unde numărul reprezinta numărul de ordine corespunzător culorii

În final, am aplicat aceeași metodă de la taskul 1 pentru a despărți imaginea colorată în 81 de pătrate la care am verificat apoi culoarea RGB a unui pixel și l-am numerotat corespunzător. Apoi din imaginea 1, menționată la început, am extras numerele folosind metoda din taskul 1 și am concatenat cele două siruri, în aşa fel încât să obținem outputul dorit.