Sudoku

Constantinescu Paul

Explicatie generala:

Am identifical imaginea relevanta (doar sudoku-ul), parcurg fiecare casuta si analizez daca este ocupata si cum este conectata fata de vecini folosind matricea de energie. Aplic un algoritm de colorare si afisez solutiile.

1. show_image:
   Functie ajutatoare pentru afisarea imaginilor
2. preprocess_image:
   Functie destinata prelucrarii imagini pentru a extrage colturile si imaginea sharpened
   Am aplicat filtrele sugerate la laborator, modificand doar medianBlur
   Pentru Task 2 am modificat putin imaginea sharpened pentru a observa mai usor liniile

   ```
   image_sharpened2 = cv.addWeighted(image_m_blur2, 1.0, image_g_blur2, -0.883, 0)
   ```

   Dupa aplicarea filtrelor extrag cele 4 puncte, identificand cel mai mare dreptunghi continuu
3. rotateAndCropImage:
   Functie destinata decuparii si indreptarii pozei
   Construiesc un dreptunghi (mai degraba patrat) din cele 4 colturi identificate la (2)

   ```
   corners = np.array([botLeft, topLeft, topRight, botRight])
   rect = cv.minAreaRect(corners)
   ```

   “Decupez” dreptunghiul si rotesc poza folosind transformarile de perspectiva

   ```
   M = cv.getPerspectiveTransform(src_pts, dst_pts)

   warped = cv.warpPerspective(img, M, (latura, latura))
   ```
4. compute_energy:
   Functie destinata calcularii energiei imaginii
   Aplic Sobel pentru ambele axe pentru a obtine energia imaginii (unde am valori/ nu am)
5. Get_results:
   Functie destinata identificarii patratelor ocupate

Parcurg patch cu patch (casutele), iau centrul lor (12 pixeli a fost val identificata de mine ca fiind cea mai utila). Daca valorea energiei depaseste pragul (130, ales arbitrar) marchez ca fiind ocupata.

6. Get_results2:
   Functie destinata identificarii liniilor verticale
   Similar get_results, doar ca verific daca linia e ingrosata sau nu
   Parametrii au fost alesi arbritar(43, 175)

7. Get_results3:
   Functie destinata identificarii orizontale
   Similar get_results, doar ca verific daca linia e ingrosata sau nu
   Parametrii au fost alesi arbritar(40, 160)
8. showMatrix:
   Functie destinata afisarii matricei sub forma din exemplu
9. buildColoringMatrix:
   Functie destinata crearii matricei de colorare
   Folosind matricele de la (6) si (7) construiesc o matrice in care reprezint prin ‘n’ casutele, ‘=’ liniile care permit trecerea (casutele sunt conectate) si 0 lniile ingrosate (care nu permit trecerea)

```
for i in range(0, 16, 2):
    for j in range(0, 16, 2):
        matrix[i][j] = 'n'
        if matrixVertical[i//2][j//2 + 1] == 'X':
            matrix[i][j + 1] = '0'
        else:
            matrix[i][j + 1] = '='
        if matrixHorizontal[i//2 + 1][j//2] == 'X':
            matrix[i + 1][j] = '0'
            matrix[i + 1][j + 1] = '0'
        else:
            matrix[i + 1][j] = '='
            matrix[i + 1][j + 1] = '0'
    for i in range(0, 16, 2):
        matrix[i][16] = 'n'
        if matrixHorizontal[i//2 + 1][8] == 'X':
            matrix[i + 1][16] = '0'
        else:
            matrix[i + 1][16] = '='
    for i in range(0, 16, 2):
        matrix[16][i] = 'n'
        if matrixVertical[8][i//2 + 1] == 'X':
            matrix[16][i + 1] = '0'
        else:
            matrix[16][i + 1] = '='
    matrix[16][16] = 'n'
```

10. fill:
    Functie destinata colorarii matricei
    Folosesc un algoritm clasic de flood fill pentru a colora matricea (9)
11. transformMatrix:
    Functie destinata transormarii matricei (9) in matrice normala (9 pe 9)
    Elimin toate elementele ajutatoare adaugate la (9): ‘=’, ‘0’
12. task1:
    Apeleaza functiile mentionate mai sus pentru generarea predictiilor necesare pt task-ul 1
    Pe langa apelarea functiilor singurul lucru mentionabil pe care il rezolva este “marcarea” liniilor si coloanelor, proces indispensabil pt (5, 6, 7)

```
lines_vertical = []
space = 55
i = 3
for j in range(9):
    l = []
    l.append((i, 0))
    l.append((i, h))
    lines_vertical.append(l)
    i += np.uint(space)
lines_vertical.append([(h - 3, 0), (h - 3, h)])

lines_horizontal = []
space = 55
i = 3
for j in range(9):
    l = []
    l.append((0, i))
    l.append((w, i))
    lines_horizontal.append(l)
    i += np.uint(space)
lines_horizontal.append([(0, w - 3), (w, w - 3)])
```

13. task2:
    Acelasi lucru pt task1 plus apelarea functiilor pt rezolvarea task-ului 2
14. GeneratePredictionTask:
    Functie ajutatoare pentru a parcurge toate testele deodata