

Concepțe și aplicații în Vederea Artificială - Tema 1

Calculator automat de scor pentru jocul Double Double Dominoes

Cioclov Maria-Simona

5 decembrie 2023

Task 1 - Identificarea poziției piesei adăugate în runda curentă:

- Extragerea careului

- am folosit scriptul din laboratorul 6 pentru a-mi da seama, în mod experimental, cum să modific imaginea tablei de joc folosind spațiul HSV, astfel încât să rămână definită cât mai clar marginea careului pe care se adăugată piese; pe imaginea rezultată am adăugat un filtru median de reducere a zgomotului, apoi un filtru de erodare, pentru a netezi marginile careului (cele trei procesări sunt ilustrate, în această ordine, în figura 1)

```
1 low_yellow = (20, 0, 0)
2 high_yellow = (130, 255, 255)
3 img_hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)
4 image = cv.inRange(img_hsv, low_yellow, high_yellow)
5
6 image_median_blur = cv.medianBlur(image, 21)
7
8 kernel = np.ones((19,19), np.uint8)
9 eroded = cv.erode(image_median_blur, kernel)
```

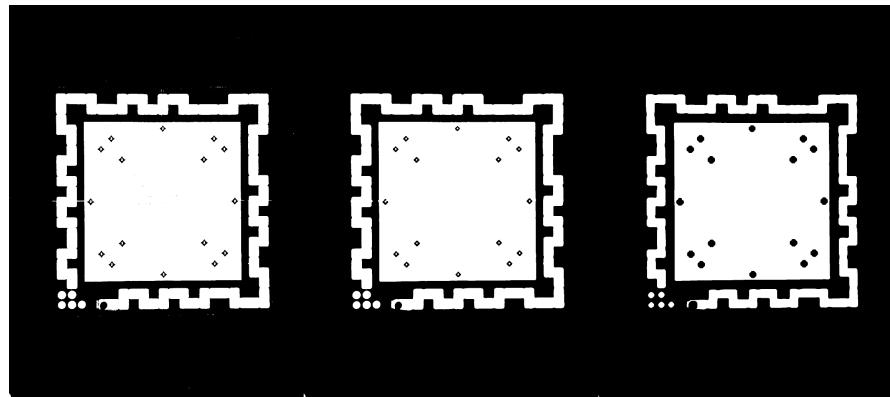
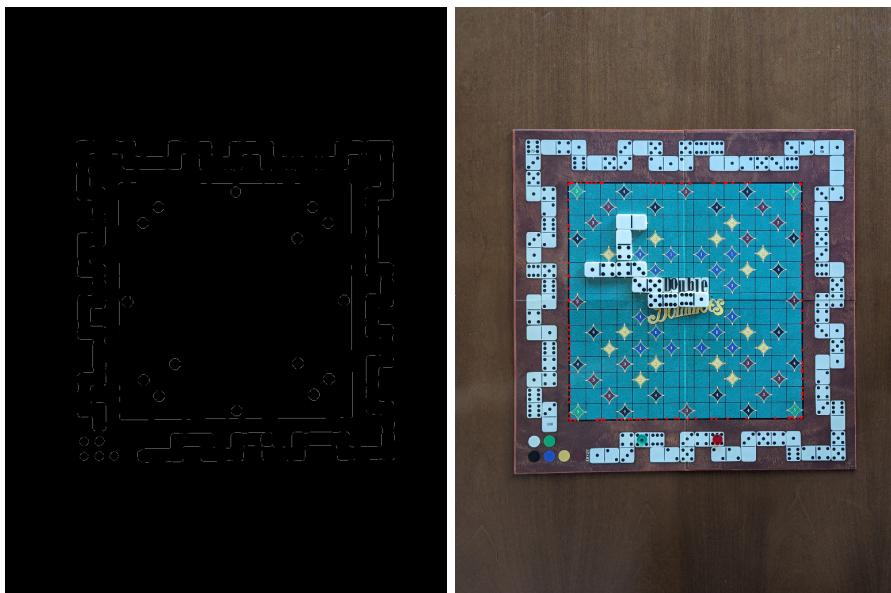


Figura 1: Preprocesare pentru extragerea careului - imaginea în spațiul HSV, imaginea după aplicarea filtrului median, imaginea după aplicarea filtrului de erodare

- am detectat muchiile folosind metoda Canny (2a), apoi am extras contururile, careul fiind reprezentat de conturul cu aria maximă (2b)



(a) Detectare muchii

(b) Conturul careului

Figura 2

- am extras colțurile careului (3a)

```

1 top_left = None
2 bottom_right = None
3 for point in careu.squeeze():
4     if top_left is None or point[0] + point[1] < top_left
5         [0] + top_left[1]:
6             top_left = point
7     if bottom_right is None or point[0] + point[1] >
8         bottom_right[0] + bottom_right[1]:
9             bottom_right = point

```

```

8  diff = np.diff(careu.squeeze(), axis = 1)
9  top_right = careu.squeeze()[np.argmin(diff)] #deoarece
   acolo este x maxim si y minim
10 bottom_left = careu.squeeze()[np.argmax(diff)] #deoarece
    acolo este x minim si y maxim

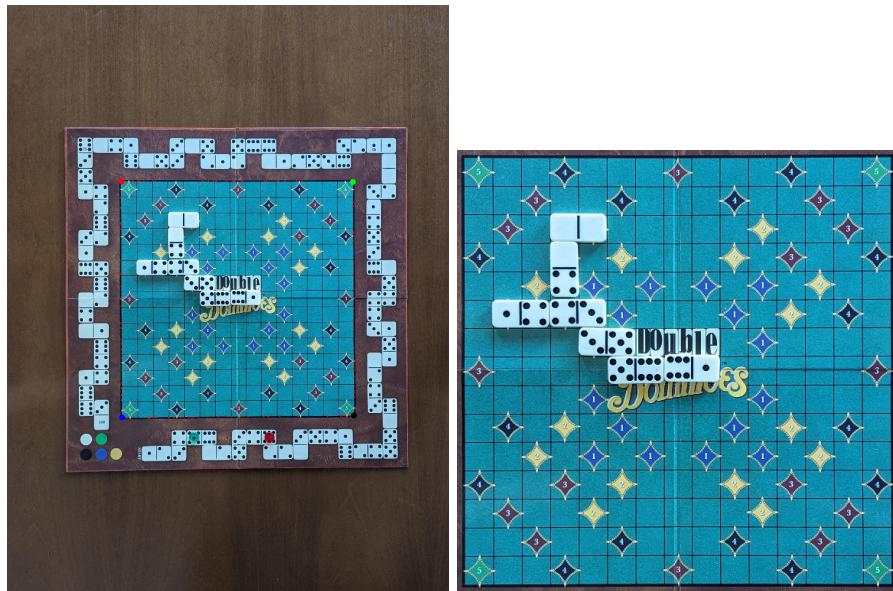
```

și am îndreptat perspectiva (3b)

```

1  size = 150 * 15 + 60 #am adaugat o margine de 30px în
   jurul careului (pentru a nu taia din piesele de la
   marginea)
2
3  puzzle = np.array([top_left-30, top_right+[30,-30],
   bottom_right+30, bottom_left+[-30,30]], dtype="float32")
4  destination_of_puzzle = np.array([[0,0],[size,0], [size,
   size],[0, size]], dtype="float32")
5  M=cv.getPerspectiveTransform(puzzle, destination_of_puzzle
   )
6  result = cv.warpPerspective(original, M, (size, size))

```



(a) Colțurile careului

(b) Rezultatul final

Figura 3

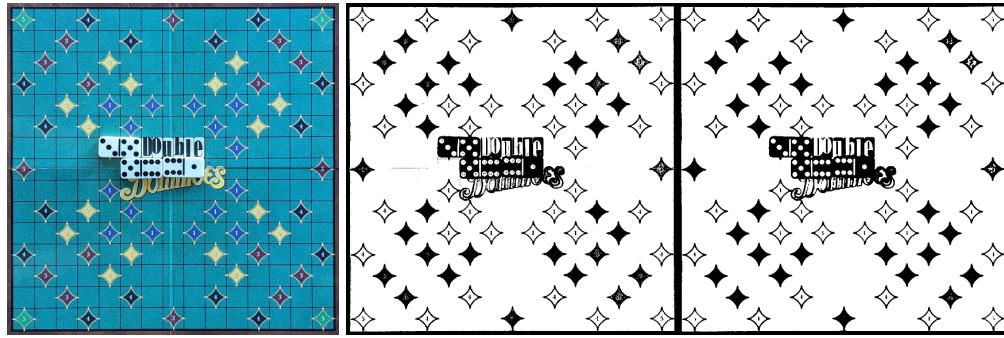
- Identificarea coordonatelor

- procesez imaginea cu starea anterioară și imaginea cu starea curentă a tablei de joc, astfel încât să pun în evidență piesele (folosesc din nou spațiul HSV, apoi un filtru median pentru reducerea zgromotului) - un exemplu de imagine procesată apare în figura 4

```

1 low_yellow = (0, 110, 0)
2 high_yellow = (255, 255, 255)
3 img_hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)
4 image = cv.inRange(img_hsv, low_yellow, high_yellow)
5
6 image_median_blur = cv.medianBlur(image, 9)

```

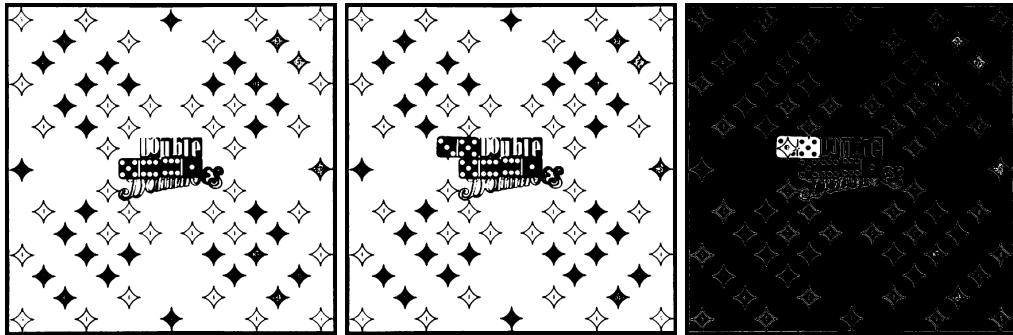


(a) Imaginea inițială

(b) Modificare cu spațiul HSV, apoi aplicarea filtrului median

Figura 4

- fac diferența absolută între cele două imagini procesate ale tablei de joc (cu starea anterioară și cea curentă); piesa nouă a fost plasată în pătrățelele din careu care au suma pixelilor maximă, după cum se observă în figura 5



(a) Starea anterioară

(b) Starea curentă

(c) Diferența

Figura 5

- pentru a itera prin pătrățelele careului am folosit o listă a liniilor orizontale și o listă a liniilor verticale (6); liniile au fost luate din $x \times x$ pixeli, unde x este lungimea unui pătrățel

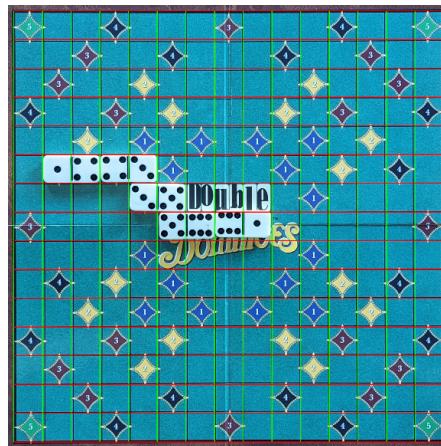


Figura 6: Afisarea liniilor din careu

Task 2 - Recunoasterea numerelor de pe piesă:

- Extragerea piesei



Figura 7: Detectarea piesei

- având poziția piesei, o pot extrage din imagine (8); relativ la aceasta imagine, îmi dau seama care ar fi coordonatele unei porțiuni din mijloc, care să conțină linia despărțitoare a numerelor



Figura 8: Extragerea piesei din imagine

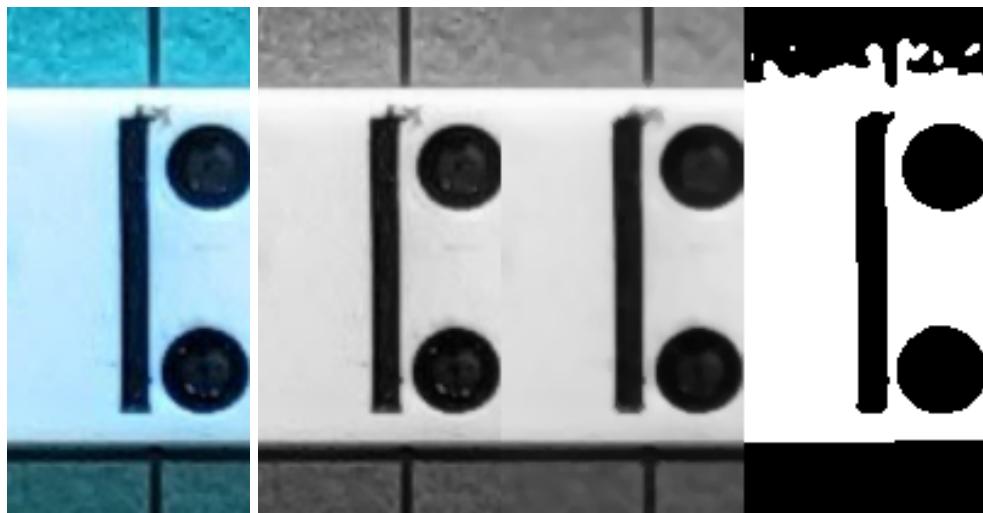
- Împărțirea piesei în jumătăți

- extrag portiunea din mijlocul piesei care conține linia despărțitoare (9a); procesez imaginea astfel: o transform în imagine grayscale, aplic un filtru median pentru a estompa zgromotul, aplic un threshold pentru a obține o imagine binară (9b)

```

1 portiune_linie = cv.cvtColor(portiune_linie , cv.
2   COLOR_BGR2GRAY)
3 image_median_blur = cv.medianBlur(portiune_linie ,5)
4 - , thresh = cv.threshold(image_median_blur , 150, 255, cv.
5   THRESH_BINARY)

```

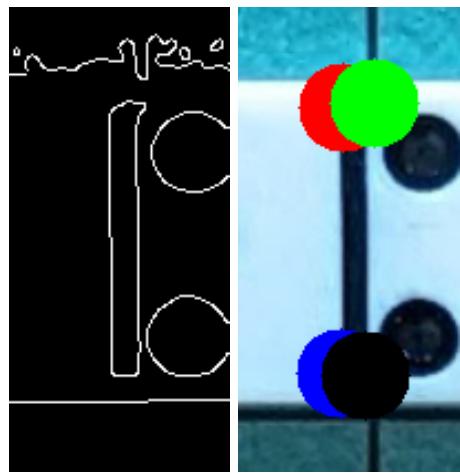


(a) Linia despărțitoare

(b) Grayscale; filtru median; threshold

Figura 9

- pe imaginea procesata aplic detectorul de muchii Canny (10a), apoi extrag contururile, cel de arie maximă fiind conturul liniei; detectez colțurile acestui contur, la fel cum am facut pentru careul de joc (9b)
- știind coordonatele portiunii care conține linia despărțitoare relativ la imaginea completa a piesei, pot calcula coordonatele colțurilor linei în imaginea completă a piesei (11)
- presupunând că a fost extras corect conturul liniei despărțitoare și știind cât este latura unui pătrătel, încerc să scot din imagine patch-urile cu cele două jumătăți de piesă (12)



(a) Muchiile detectate
(b) Colțurile conturului liniei

Figura 10

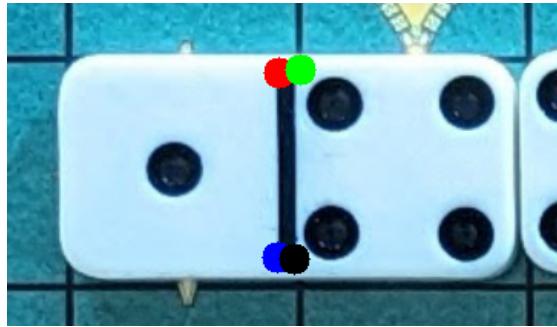


Figura 11: Colțurile conturului liniei în imaginea completă a piesei



(a) Prima jumătate a piesei (b) A doua jumătate a piesei

Figura 12

- Procesarea patch-urilor

- cele două jumătăți de piesă și template-urile (care au fost tăiate să fie puțin mai mici decât jumătățile de piesă) sunt procesate astfel: se transformă imaginea în grayscale și se aplică un threshold; se reduce zgomotul cu un filtru median; se dilată

puțin imaginea pentru a netezi marginea bulinelor negre; pentru template-uri, după procesare am mai dilatat încă puțin imaginea, pentru că bulinele erau puțin mai mari decât în bucătile de piesă extrase

```

1 def proceseaza_piesa(image):
2     image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
3     _, thresh = cv.threshold(image, 150, 255, cv.
4         THRESH_BINARY)
5
6     image_median_blur = cv.medianBlur(thresh, 3)
7
8     kernel = np.ones((3,3), np.uint8)
9     dilated = cv.dilate(image_median_blur, kernel)
10
11    return dilated

```

- Template matching
 - template matchingul cu parametrul TM_SQDIFF_NORMED mergea cel mai bine

Task 3 - Calcularea scorului pentru runda curentă:

- Construirea matricei cu punctajele marcate pe tabla și a listei cu traseul parcurs de pioni

```

1 #matricea punctelor de pe tabla
2 m = np.zeros((15, 15), dtype=int)
3
4 m[0][0] = m[14][14] = m[0][14] = m[14][0] = 5
5 m[0][3] = m[0][11] = m[1][5] = m[1][9] = m[3][0] = m[11][0] =
6     m[5][1] = m[9][1] = m[14][3] = m[14][11] = m[13][5] = m
7     [13][9] = m[3][14] = m[11][14] = m[5][13] = m[9][13] = 4
8 m[0][7] = m[1][2] = m[1][12] = m[2][1] = m[2][13] = m[3][3] =
9     m[3][11] = m[7][0] = m[7][14] = m[11][3] = m[11][11] = m
10    [12][1] = m[12][13] = m[13][2] = m[13][12] = m[14][7] = 3
11 m[2][4] = m[2][10] = m[3][5] = m[3][9] = m[4][2] = m[4][12] =
12     m[5][3] = m[5][11] = m[9][3] = m[9][11] = m[10][2] = m
13     [10][12] = m[11][5] = m[11][9] = m[12][4] = m[12][10] = 2
14 m[4][4] = m[4][6] = m[4][8] = m[4][10] = m[5][5] = m[5][9] =
15     m[6][4] = m[6][10] = m[8][4] = m[8][10] = m[9][5] = m
16     [9][9] = m[10][4] = m[10][6] = m[10][8] = m[10][10] = 1
17
18 #traseul de pe margine
19 margine = [1, 2, 3, 4, 5, 6, 0, 2, 5, 3, 4, 6, 2, 2, 0, 3, 5,
20     4, 1, 6, 2, 4, 5, 5, 0, 6, 3, 4, 2, 0, 1, 5, 1, 3, 4, 4,
21     4, 5, 0, 6, 3, 5, 4, 1, 3, 2, 0, 0, 1, 1, 2, 3, 6, 3, 5,
22     2, 1, 0, 6, 6, 5, 2, 1, 2, 5, 0, 3, 3, 5, 0, 6, 1, 4, 0,
23     6, 3, 5, 1, 4, 2, 6, 2, 3, 1, 6, 5, 6, 2, 0, 4, 0, 1, 6,
24     4, 4, 1, 6, 6, 3, 0]

```

- Calcularea efectivă a scorului

```

1  def calculeaza_scor(scor_jucatori, coord, numere,
2      pozitie_jucatori, jucator_curent):
3      scor = [0,0]
4      #adun punctele de pe tabla atinse de piesa
5      scor[jucator_curent] = m[coord[0][0]-1][coord[0][1]-1] +
6          m[coord[1][0]-1][coord[1][1]-1]
7      #daca piesa este dubla, dublez punctele
8      if(numere[0] == numere[1]):
9          scor[jucator_curent] *= 2
10
11     #daca pionul unui jucator se afla pe un numar care se
12     #regasesc si pe piesa, ii adaug 3 puncte
13     if pozitie_jucatori[0] != -1 and (margine[
14         pozitie_jucatori[0]] == numere[0] or margine[
15         pozitie_jucatori[0]] == numere[1]):
16         scor[0] += 3
17     if pozitie_jucatori[1] != -1 and (margine[
18         pozitie_jucatori[1]] == numere[0] or margine[
19         pozitie_jucatori[1]] == numere[1]):
20         scor[1] += 3
21
22     #actualizez pozitiile si scorul
23     pozitie_jucatori[0] += scor[0]
24     pozitie_jucatori[1] += scor[1]
25     scor_jucatori[0] += scor[0]
26     scor_jucatori[1] += scor[1]
27
28     return scor[jucator_curent], scor_jucatori,
29         pozitie_jucatori

```