**DOCUMENTATIE**

SON ANDREEA MARINA

GRUPA 463

**Cerinte**

1. “*Vom evalua performanta algoritmului vostru pe o multime de 100 de imagini de testare ce reprezinta 5 jocuri de DDD, fiecare joc avand 20 de runde. Pentru fiecare imagine de testare algoritmul vostru trebuie sa furnizeze pozitia piesei care a fost plasata pe tabla in runda respectiva. Veti primi punctajul numai daca algoritmul vostru furnizeaza corect pozitia piesei adaugate pe tabla (ambele capete). Nu exista punctaje partiale. Fiecare configuratie corecta valoreaza 0.05 puncte pentru un total de* ***5 puncte***”
2. “*Vom evalua performanta algoritmului vostru pe aceleasi 100 de imagini de la Task-ul 1, pentru piesa plasata trebuie sa recunoasteti numerele care se afla pe piesa respectiva (pe cele doua capete). Veti primi punctajul numai daca algoritmul vostru recunoaste corect ambele numere. Nu exista punctaje partiale. Fiecare configuratie corecta a numerelor valoreaza 0.02 puncte pentru un total de* ***2 puncte***”
3. “*Vom evalua performanta algoritmului vostru pe aceleasi 100 de imagini de la Task-urile 1 si 2, pentru fiecare imagine trebuie sa calculati scorul obtinut de jucatorul curent. Veti primi punctajul numai daca algoritmul vostru calculeaza corect scorul pentru runda curenta. Nu exista punctaje partiale. Pentru fiecare scor corect calculat primiti 0.02 puncte pentru un total de* ***2 puncte***”

**Rezolvari**

Indiferent de abordarea aleasa, un prim pas in determinarea pozitiei dominourilor de pe tabla si a valorilor de pe fiecare piesa este extragerea tablei de joc. Pentru a reusi extragerea tablei, am decupat initial imaginile si am eliminat astfel bucata din tabla care continea avansul pieselor jucatorilor, intrucat nu este relevanta in determinarea careului principal.

|  |
| --- |
| # coordonatele imaginii decupate  X = 720  Y = 1170  W = 1765  H = 1770 |



*Imaginea decupata*

Mai apoi, au fost aplicate un numar de filtre asupra imaginilor, in principal pentru a crea o versiune *sharpened* a imaginii initiale.

|  |
| --- |
| image\_m\_blur = cv.medianBlur(image,9)  image\_g\_blur = cv.GaussianBlur(image\_m\_blur, (0, 0), 5)  image\_sharpened = cv.addWeighted(image\_m\_blur, 1.2, image\_g\_blur, -0.8, 0) |



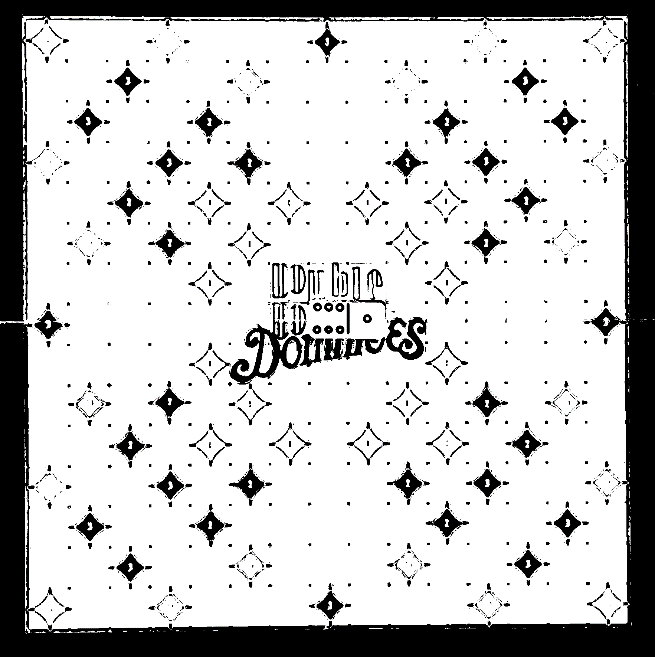
*Imaginea Sharpened*

Cu imaginea *sharpened* obtinuta a fost creata o noua imagine (***image\_HSV***) pentru care culoarea este reprezentata in format HSV (*Hue*, *Saturation* si *Value*). Aceasta va ajuta la eliminarea conturului maro de pe marginea careului prin crearea unei masti (***mask\_HSV***).

Spre deosebire de formatul RGB (*Red*, *Green*, *Blue*), in formatul HSV culorii ii revine un singur canal, *Hue*, in timp ce *Saturation* si *Value* determina cat de vibranta/terna si luminoasa/intunecata este culoarea. Astfel, in contextul aplicarii unei masti, exista posibilitatea de a include un interval cat mai realist/cuprinzator pentru culorile care se doresc a fi eliminate.

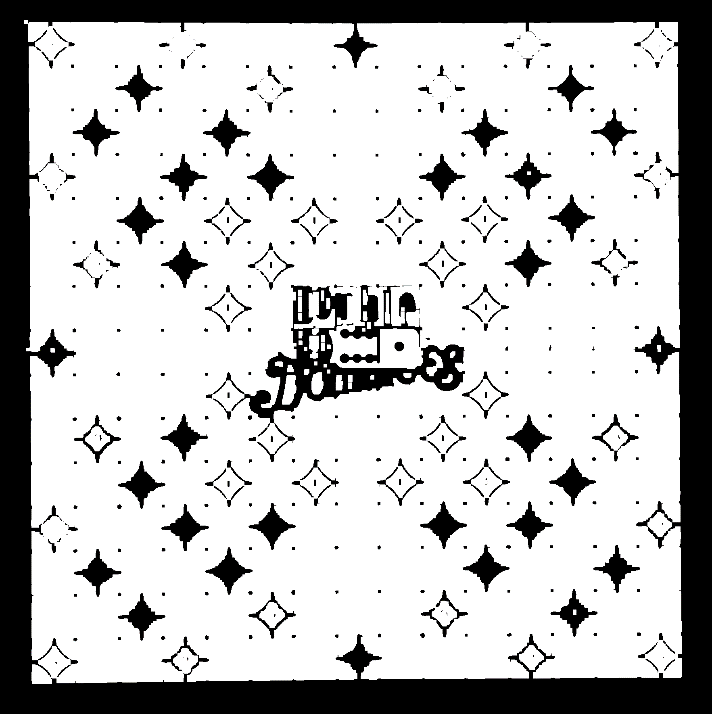
Pentru a crea masca, a fost definit intervalul HSV in care se afla culoarea maro care va fi eliminata:

|  |
| --- |
| low\_HSV = (50, 0, 0)  high\_HSV = (120, 255, 255) |

****

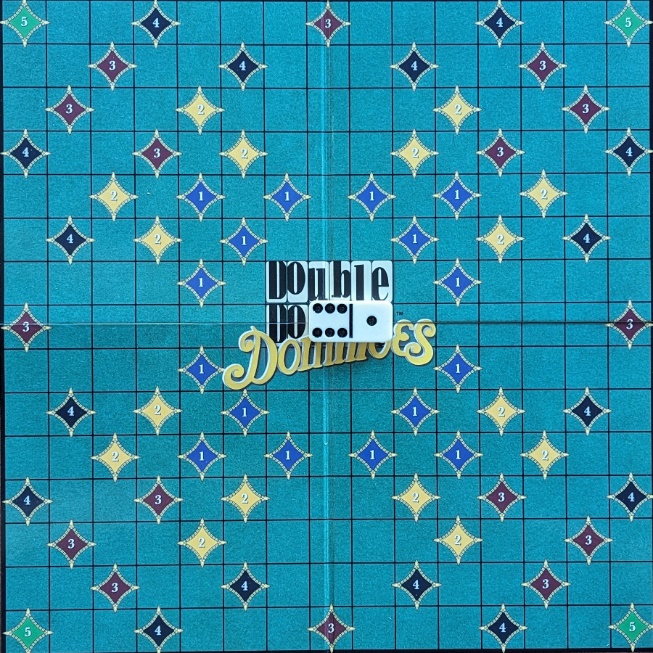
*Masca HSV*

Pentru a elimina zgomotul (*noise*) din imagine, imaginea a fost erodata (*cv.erode*) si dilitata (*cv.dilate*) folosind un nucleu (*kernel*) patratic, de dimensiune 9x9, prin intermediul unei singure functii (*cv.morphologyEx*)



*Imaginea open*

Apoi, a fost determinat cel mai mare contur din imagine, iar coordonatele sale au fost indentificate in imaginea initiala si mapate in intervalul 0-900px, pentru a crea imaginea finala de dimeniune 900x900px.



*Tabla extrasa*

Dupa extragerea tablei, se vor trasa liniile orizontale si verticale care determina casutele (15x15 casute). Apoi, pentru a indentifica dominourile si valorile lor a fost creata functia ***get\_domino\_matrix()***, care ia ca parametrii tabla de joc extrasa, careia i s-a aplicat un threshold (pentru a deveni o imagine alb-negru), coordonatele liniilor orizontale si ale liniilor verticale. Functia calculeaza media pixelilor fiecarei casute din tabla si daca este mai mare de 100, cosidera ca este un domino plasat acolo si il marcheaza in matricea finala cu 1, alfel cu 0.

Functia ***get\_patches\_from\_matrix()*** ia ca parametrii diferenta dintre matricea returnata de functia ***get\_domino\_matrix()*** la iteratia curenta si matricea iteratiei anterioare (iteratie = mutare), returnand toate posibilitatile de a grupa cate 2 casute alaturate din tabla, care formeaza posibile *patch*-uri de dominouri (functia de determinare a matricei tablei poate marca gresit dominourile, care se pot confunda cu diferite alte detalii ale tablei, de aceea *patch*-urile nu sunt exacte).

Am ales sa calculez diferenta dintre matricea de la iteratia curenta si cea de la iteratia anterioara pentru ca aceasta diferenta poate oferi coordonatele exacte ale dominoului din mutarea curenta.

Pentru a determina valorile dominourilor si a filtra casutele indentificate drept dominouri in mod incorect, a fost scrisa functia ***classify\_domino()***. In interiorul functiei, a fost folosit c*v2.matchTemplate()*, avand ca parametrii *patch*-urile gasite anterior si *template*-ul fiecarei piese de domino. Fiecare rezultat al *cv2.matchTemplate()* va fi comparat pentru a obtine cel mai bun *match* cu un domino. In ceea ce priveste *template*-urile, acestea au fost generate pentru a avea toate aceeasi dimensiune si detalii (cercuri) situate la distante precise unele de altele (codul de generare nu a fost inclus in solutie, ci doar *template*-urile generate).

Functia ***classify\_domino()*** va returna gradul de similitudine dintre *patch* si *match*-ul final (returnat de *cv2.matchTemplate()*), impreuna cu valorile piesei cu care s-a obtinut cel mai bun *match*. Apoi, in iteratia din ***main()*** printre mutarile fiecarui joc se va face comparatia intre *patch*-uri, iar *patch*-ul cu cel mai mare grad de similitudine cu o piesa de domino va corespunde pozitiei corecte a dominoului plasat pe tabla curenta.

Pentru a determina scorul la fiecare mutare, functia ***calculate\_score()*** are salvate coordonatele fiecarei stelute de pe tabla de joc care ofera puncte bonus jucatorilor, cat si valoarea fiecarei piese de domino desenate pe laturile tablei. Functia tine scorul pentru ambii jucatori si returneaza scorul amandurora, indiferent cine este la mutare, urmand ca scorul jucatorului curent si informatiile despre dominoul identificat sa fie scrise intr-un fisier text, respectand formatul cerut.