Double Double Dominoes

CAVA Tema 1

Trifan Robert-Gabriel

- Grupa 352 -

Contents

1 Introducere **2**

2 Extrage tabla de joc **2**
2.1 Descriere 2
2.2 Algoritm 3

3 Extragerea pieselor de domino **4**
3.1 Descriere 4
3.2 Extragerea tuturor pieselor de domino 4
3.3 Extragerea piesei nou aparute 5
3.4 Curatarea noise-ului din imagine 6
3.5 Detectia cercurilor de pe piesa de domino 7
3.6 Clasificarea piesei de domino 8

4 Regulile jocului **9**

1 Introducere

Entry point-ul aplicatiei este `game.py`. Pentru a rula aplicatia, am facut un `README.md` care contine instructiuni pentru a instala si rula aplicatia.

Flow-ul aplicatiei este urmatorul:

1. Extrage tabla de joc din imaginea primita ca input folosind `board_extractor.py`
2. Detecteaza toate piesele de pe fiecare tabla de joc folosind `domino_extractor.py`
3. Detecteaza piesa nou aparuta la fiecare mutare
4. Extrage piesa nou aparuta si curata noise-ul din imagine folosind `domino_cleaner.py`
5. Clasifica piesa folosind `domino_classifier.py`
6. Dupa ce avem toate piesele clasificate, se aplica regulile jocului pentru a calcula punctajul fiecarui jucator

2 Extrage tabla de joc

2.1 Descriere

`board_extractor.py` este o clasa care primeste la initializare calea catre un dictionar serializat care contine parametrii folositi de algoritmul de extragere a tablei de joc.
Aici este dictionarul (pe care il voi numi de acum `hparams`) care contine parametrii folositi de algoritmul de extragere a tablei de joc:

Key	Value
guassian_blur_kernel	21
median_blur_kernel	21
erode_kernel_size	1
erode_iterations	3
hue_min	90
hue_max	160
saturation_min	130
saturation_max	255
value_min	30
value_max	255

`board_extractor.py` are o metoda `extract_board` care primeste ca parametru o imagine, aplica filtrele din `hparams` si returneaza o imagine care contine doar tabla de joc.

2.2 Algoritm

Experimental am ajuns la concluzia ca pentru a extrage tabla de joc dintr-o imagine, trebuie sa aplicam urmatorii pasi:

1. Convertim imaginea din RGB in HSV
2. Aplicam un filtru gaussian si unul median pentru a elimina noise-ul din imagine
3. Aplicam un filtru de erodare pentru a elimina noise-ul din imagine
4. Impunem limite superioare si inferioare pentru valorile H, S si V
5. Gasim conturul cu cea mai mare arie din imagine
6. Extragem zona care contine conturul cu cea mai mare arie si facem warp perspective pentru a o transforma intr-un dreptunghi
7. De asemenea, expandam zona extrasa pentru a evita taieri accidentale

Figure 1: Imaginea originala

Figure 2: Tabla de joc extrasa

Figure 3: Din figura 4 extragem tabla de joc si obtinem 2.

3 Extragerea pieselor de domino

3.1 Descriere

Asemanator cu `board_extractor.py`, `domino_extractor.py` este o clasa care primeste la initializare calea catre un dictionar serializat care contine parametrii folositi de algoritmul de extragere a pieselor de domino.

Key	Value
median_blur_kernel	41
erode_kernel_size	1
erode_iterations	8
dilate_kernel_size	1
dilate_iterations	12
hue_min	90
hue_max	255
saturation_min	40
saturation_max	255
value_min	220
value_max	255

`domino_extractor.py` are o metoda `extract_dominoes` care primeste ca parametru o imagine, aplica filtrele din `hparams` si obtine o imagine cu pixeli albi sau negri. Aceasta imagine este impartita in 15x15 patratele. Daca un patrat are mai mult de un threshold setat pixeli albi, atunci patratul este contine o piesa de domino. Functia returneaza un numpy array de 15x15 care contine 1 daca patratul contine o piesa de domino si 0 altfel.

3.2 Extragerea tuturor pieselor de domino

Tot experimental, am obtinut urmatorul algoritm pentru a extrage piesele de domino dintr-o imagine:

1. Convertim imaginea din RGB in HSV
2. Aplicam un filtru gaussian si unul median pentru a elimina noise-ul din imagine
3. Aplicam un filtru de erodare pentru a elimina noise-ul din imagine

4. Aplicam un filtru de dilatare pentru a umple gaurile din imagine
5. Impunem limite superioare si inferioare pentru valorile H, S si V
6. Impartim imaginea in 15x15 patratele
7. Daca un patrat are mai mult de un threshold setat pixeli albi, atunci patratul contine o piesa de domino
8. Returnam un numpy array de 15x15 care contine 1 daca patratul contine o piesa de domino si 0 altfel

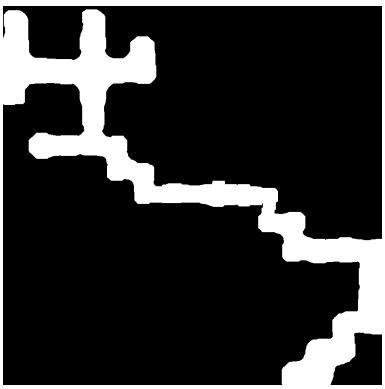


Figure 4: Imaginea originala

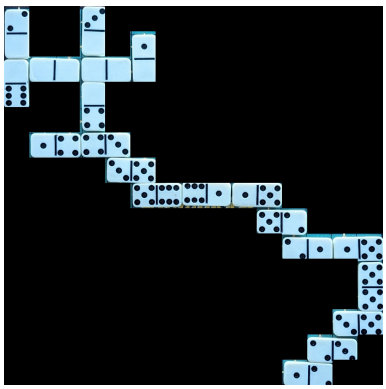


Figure 5: Piesele de domino extrase

Figure 6: Figura 4 surprinde zonele unde se pot afla piesele de domino si figura 5 contine piesele de domino extrase.

3.3 Extragerea piesei nou aparute

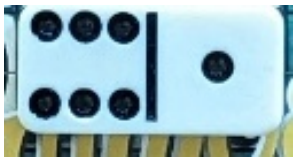
Folosindu-ne de numpy array-ul returnat de `extract_dominoes`, putem extrage piesa nou aparuta la fiecare mutare. Stim ca pentru o piesa nou aparauta, vor aparea 2 patratele noi cu 1 in numpy array-ul returnat. Deci putem extrage usor piesa nou aparuta. Deoarece piesele de domino pot fi puse stramb, functia `extract_domino` expandeaza zona extrasa pentru a evita taieri accidentale.

3.4 Curatarea noise-ului din imagine

Folosim clasa din `domino_cleaner.py` pentru a curata noise-ul din imaginea cu un domino, cu ajutorul parametrilor din `hparams`.

Key	Value
erode_kernel_size	5
erode_iterations	1
hue_min	0
hue_max	255
saturation_min	0
saturation_max	255
value_min	220
value_max	255

Aici sunt cateva exemple de imagini cu un domino inainte si dupa curatarea noise-ului:



(a) Imaginea originala 1

(b) Imagina procesata 1

(c) Imaginea originala 2

(d) Imaginea procesata 2

(e) Imaginea originala 3

(f) Imaginea procesata 3

Figure 7: Algoritmul de curatare a noise-ului.

3.5 Detectia cercurilor de pe piesa de domino

Pentru a clasifica piesa de domino, avem clasa `DominoClassifier` din `domino_classifier.py` care foloseste HoughCircles din OpenCV pentru a detecta cercurile de pe piesa de domino. Parametrii folositi de HoughCircles au fost gasiti experimental, am incercat sa restrictionez cercurile corecte folosind minRadius, maxRadius, minDist param1 si param2.

Key	Value
minDist	10
param1	100
param2	10
minRadius	5
maxRadius	8
dp	1
blur_kernel	5
circle_threshold	0.5

Aici sunt cateva detectii ale algoritmului:

(a) Imagina procesata 1

(b) Imaginea procesata 2

(c) Imaginea procesata 3

Figure 8: Algoritmul de detectie a cercurilor din piesa de domino.

3.6 Clasificarea piesei de domino

Pentru a clasifica piesa de domino, am taiat imaginea in 2 si am numarat cercurile din fiecare jumatate. Pentru piesele orizontale am taiat piesa pe verticala, iar pentru piesele verticale am taiat piesa pe orizontala.

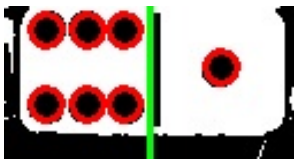


Figure 9: Imagina procesata 1

Figure 10: Imaginea procesata 2

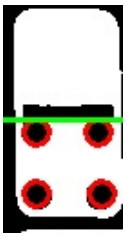


Figure 11: Imaginea procesata 3

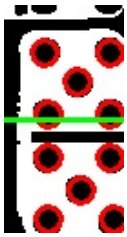


Figure 12: Imaginea procesata 4

Figure 13: Algoritmul de clasificare a piesei de domino.

4 Regulile jocului

Acum ca am terminat de clasificat toate piesele de domino, putem sa aplicam regulile jocului pentru a calcula punctajul fiecarui jucator.

Am hardcodat traseul pe care il urmeaza pionii si stelele de pe tabla de joc. Pentru a calcula punctajul fiecarui jucator, am parcurs fiecare piesa de domino si am calculat punctajul fiecarei piese in functie de traseul pe care il urmeaza.

Salvez fiecare mutare facuta intr-un txt in care scriu pozitia piesei de domino (folosind notatiile din enunt), valorile de pe fiecare piesa de domino si punctajul fiecarui jucator.