

# Documentatie proiect - Mathable

Ciuperceanu Vlad-Mihai  
Grupa 351

## 1 Introducere

Acet proiect are ca scop analizarea unor jocuri de Mathable, extragand informatia din imaginile oferite dupa fiecare mutare si rezolvand cele 3 task-uri date. Primul task este referitor la gasirea pozitiei in care s-a plasat piesa, al doilea este referitor la identificarea piesei plasate, iar al treilea task este referitor la calcularea scorului pentru fiecare jucator. Dintre acestea, primele 2 necesita tehnici de Computer Vision, in timp ce al treilea task doar se foloseste de informatia obtinuta anterior.

In ansamblu, etapele prin care trecem pentru a rezolva cerintele sunt:

- extragerea tablei mari (eliminarea mesei din imagini)
- extragerea tablei mici (eliminarea conturului tablei mari)
- impartirea tablei obtinute intr-un grid de 14x14, obtinand casutele unde vor fi plasate piesele
  - parcurgerea imaginilor si gasirea diferentei intre imaginea curenta si cea precedenta (si gasirea pozitiei unde a fost plasata ultima piesa)
  - identificarea piesei care a fost plasata prin template matching
  - retinerea scorului obtinut pentru aceasta miscare si, ulterior, determinarea scorului pentru jucatori pentru rundele fiecaruia

Primele 4 etape sunt legate de primul task, in timp ce etapa 5 corespunde task-ului 2, iar ultima etapa este legata de task-ul 3.

In continuare, vom prezenta metodele prin care am abordat aceste etape, rezolvand cele 3 task-uri.

## 2 Precalculari si observatii

In primul rand, putem analiza imaginile inainte de a le procesa, extragand cateva informatii generale despre imaginile date.

Ne putem initializa cateva constante cu date din joc si din imagini. De exemplu, putem retine ca tabla mare are dimensiuni de 1975x1975 pixeli, ca marginea dintre tabla mare si tabla mica este de 250 pixeli si ca tabla mica are dimensiuni de 1456x1456 pixeli. Aceste valori sunt aproximative, fiind observate din analizarea unei imagini deschise cu OpenCV. Datorita operatiilor ulterioare, numerele acestea nu trebuie sa fie fixe, doar suficient de apropiate pentru a ne ajuta in procesul de analizare a imaginilor. Totodata, numarul de 1456 a fost ales si pentru ca este un numar divizibil cu 14, numarul de casute pe care le avem in grid-ul nostru, lasand 104 pixeli per casuta (atat pe latime, cat si pe inaltime).

Informatiile despre casute pot fi retinute prin intermediul unei clase, continand toata informatia necesara pentru task-uri:

- pozitia
- valoarea piesei puse (daca exista)
- bonusul de pe acea casuta (1 in cazul in care nu este o casuta speciala)
- conditia care trebuie respectata in crearea ecuatiilor

De asemenea, din moment ce tabla este standard, putem sa ne initializam o matrice de 14x14 care sa reprezinte informatia dintr-o tabla goala, in care nicio piesa nu a fost pusa inca. Bonusul si conditiile sunt statice, deci pot fi retinute din start. Valorile pieselor vor fi si ele retinute de la inceput, pentru a fi folosite in task-ul 2.

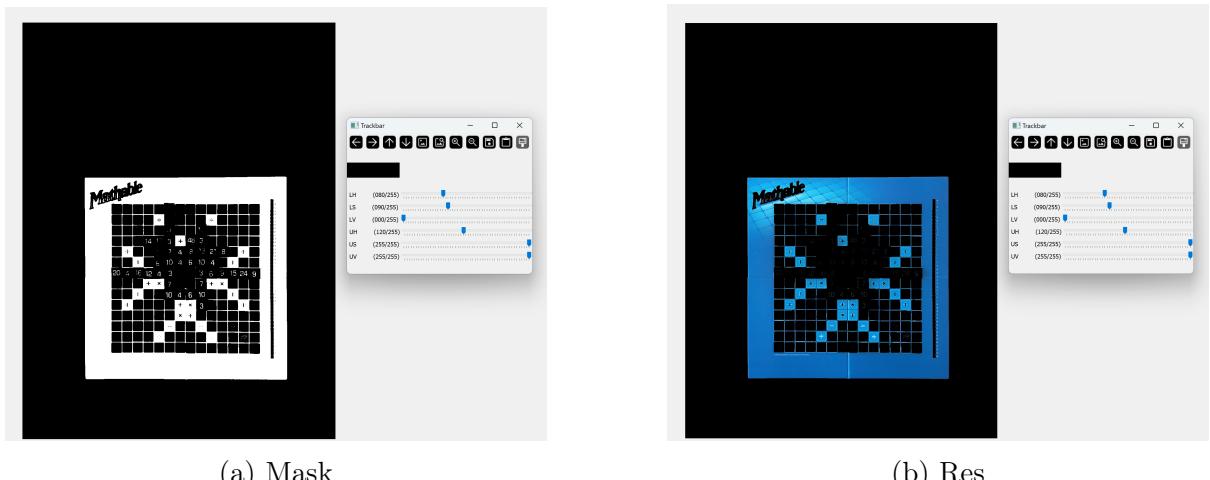
Mai observam si ca, pentru tabla mica, gasim coltul stanga-sus la coordonatele (257, 257), coltul dreapta-jos fiind la coordonatele (1713, 1713) - raportat la colturile tablei mari. Acestea ne vor fi de folos mai tarziu, in extragerea tablei mici.

### 3 Task 1 - Gasirea pozitiei in care s-a adaugat piesa

Asa cum am mentionat anterior, sunt mai multe etape prin care trebuie sa trecem pentru a obtine informatia din fiecare imagine, urmand sa comparam informatiile obtinute pentru a determina pozitia noii piese.

#### 3.1 Extragerea tablei mari

Pentru acest pas, primul lucru de facut este eliminarea mesei din imagine, urmand sa luam conturul mesei mari. Astfel, ne vom folosi de reprezentarea in HSV a imaginii, pentru generarea unei masce si pentru a transforma imaginea data, evidentiind tabla mare. Dupa mai multe incercari, folosind codul din laborator, am ajuns la un set de valori care sa faciliteze aceasta operatie:  $LH = 80$ ,  $LS = 90$ ,  $LV = 0$ ;  $UH = 120$ ,  $US = 255$ ,  $UV = 255$ . Mai jos sunt 2 ilustratii in care se poate vedea efectul acestei transformari asupra imaginii 1\_41.jpg din setul de date *fake\_test*:



(a) Mask

(b) Res

Figura 1: Transformari utilizand reprezentarea HSV

Apoi, vom face sharpening pe imaginea obtinuta, pentru a evidenția mai bine conturul tablei mari, urmat de thresholding pentru a obtine o imagine binara si erodare pentru a elimina zgomotul, in mod asemanator cu laboratorul 6.

Vom aplica Canny Edge Detection si functia *findContours* din OpenCV pentru a obtine conturul tablei mari. Prin parcurgerea sa, vom determina colturile tablei mari, pentru a putea extrage cu totul tabla mare din imagine. Colturile sunt obtinute cautand punctele cu coordonatele ale caror suma sau diferență este minima sau maxima, respectiv. Obtinem rezultatul din Figura 2 - (a). Dacă aplicam funcțiile *getPerspectiveTransform* și *warpPerspective* din OpenCV, putem obține imaginea fără masa, așa cum se vede în Figura 2 - (b).

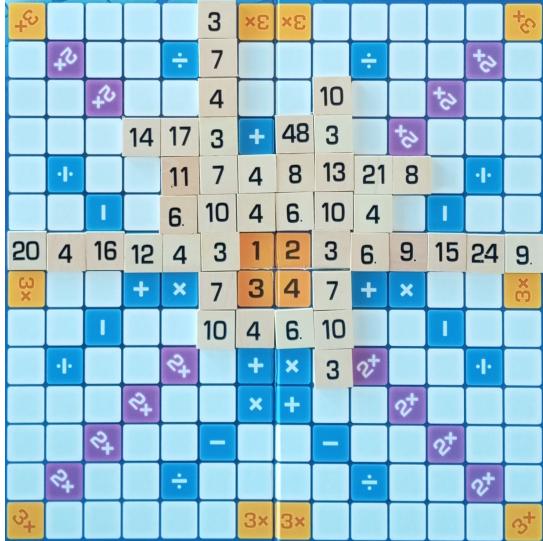


Figura 2: Detectarea colturilor și extragerea tablei mari

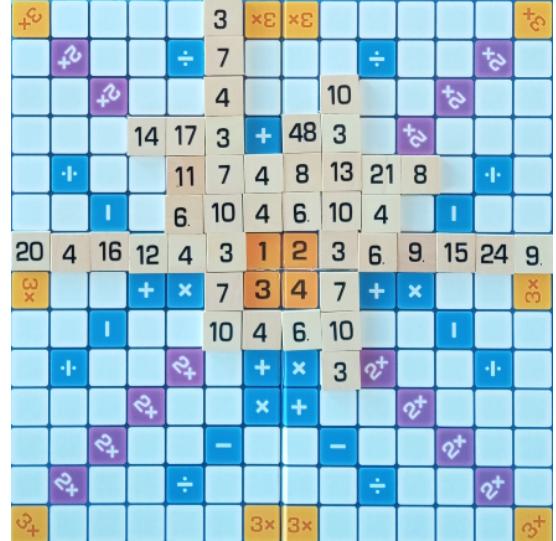
### 3.2 Extragerea tablei mici

Pentru extragerea tablei mici din cea mare, putem avea 2 metode: fie extragem direct tabla manual, folosind coordonatele approximate ale colturilor (cu mici variațiuni), fie folosim o abordare asemănătoare ca mai sus, cu cele 2 funcții din OpenCV. Prima varianta funcționează destul de bine, însă pot apărea pixeli nedoriti la margini datorita erorii de aproximare a coordonatelor, că și a faptului că tabla mică nu este perfect dreapta în imagine. Diferența între cele 2 abordări se poate vedea în figura de mai jos, justificând alegerea celei de-a două metode.

Astfel, ne-am delimitat regiunea de interes maxim din imaginile date, pentru a extrage informația necesară. Din acest punct încolo, vom folosi aceasta prelucrare a imaginii pentru urmatoarele etape, având toate datele despre mutare în aceasta imagine.



(a) Crop manual



(b) Folosirea functiilor din OpenCV

Figura 3: Extragerea tablei mici

### 3.3 Impartirea tablei mici in casutele unui grid

Delimitarea grid-ului va fi facuta manual, construind liniile care vor delimita casutele la distanta fixa de 104 pixeli, dupa cum am facut calculele anterioare despre tabla mica.

Aceasta delimitare ne va ajuta in a identifica patch-uri din tabla, urmarind locurile unde vor fi puse piesele.

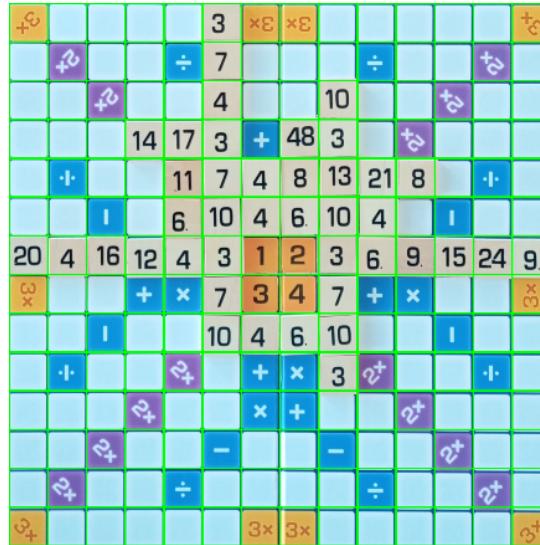


Figura 4: Tabla delimitata

### 3.4 Gasirea pozitiei piesei adaugate

Acum ca ne-am extras tabla mica, am ajuns la etapa in care trebuie sa determinam pozitia piesei adaugate.

Pentru aceasta, vom parcurge imaginile cu fiecare mutare ale unui joc si ne va interesa diferenta intre imaginea curenta si cea precedenta. Evident, prima poza a unui joc va fi

comparata cu imaginea tablei goale, extrase din imaginea auxiliara 01.jpg. Aceasta differenta va da pozitia noii piese adaugate. Astfel, transformam ambele imagini in grayscale, aplicam un threshold si calculam diferența absolută între ele. Mergând prin patch-urile delimitate anterior (care corespund casutelor din tabla), vom căuta diferența maximă dintre cele 2 imagini, pentru a determina pozitia piesei adaugate.

Desigur, vor exista diferențe și în alte locuri decât în locul în care s-a pus piesa, datorită schimbării de luminositate a mediului sau a umbrelor create prin adăugarea unei piese. De aceea, vom taia puțin din marginea patch-ului, uitându-ne mai cu seama la centrul acestuia, unde modificările vor fi mai semnificative și unde nu ar mai trebui să se resimtă atât de tare umbrele sau alte schimbări. Totodată, vom verifica să nu avem deja o piesă pusă în acea casută dintr-o imagine anterioară, pentru a exclude diferențele care nu sunt cauzate de adăugarea unei piese.

Astfel, vom gasi pozitia piesei care a fost adăugată în imaginea curentă.

## 4 Task 2 - Identificarea piesei adăugate

După ce am gasit patch-ul în care a fost adăugată o piesă nouă, ramane să o identificăm folosind template matching.

### 4.1 Obținerea template-urilor

Mai întâi, este nevoie să obținem o bază de date cu template-urile pieselor, pentru a le compara cu patch-ul gasit anterior.

Aceasta se regăsește în folder-ul *templates*, unde avem câte un template pentru fiecare piesă. Ele au fost obținute prin extragerea manuală a fiecărei piese în imagini de 103x103 pixeli, folosind imaginea auxiliara 04.jpg, unde există spațiu între piese, asigurând o bună separare și poziționare a acestora.



Figura 5: Template-urile pieselor

S-a încercat extragerea lor cat mai aproape de conturul piesei, cu numarul cat mai central, pentru a facilita template matching-ul, insa, evident, se poate observa eroarea umana in aceasta operatie. Luminozitatea si umbrele sunt, de asemenea, un alt factor in aceasta operatie. De aceea, in momentul efectuarii template matching-ului, vom taia puțin din marginile template-urilor, pentru a evita marginile care pot aduce pixeli nedoriti care sa contribuie la nivelul de corelatie.

### 4.2 Template matching

După ce am obținut patch-ul în care a fost adăugată piesa, vom trimite patch-ul respectiv către o funcție care va face template matching cu template-urile obținute anterior.

Inainte de a aplica functia *matchTemplate* din OpenCV, vom face cateva transformari ale patch-ului si ale template-urilor.

In primul rand, ambele vor fi transformate in grayscale si se va aplica thresholding pentru a obtine o imagine binara. Pragul impus este de 65, obtinut dupa mai multe incercari in asupra datelor de test. Este interesant de observat ca, fara acest threshold, template matching-ul functiona destul de bine, avand doar 5 imagini in care returna o valoare gresita. Alegerea unui prag gresit ducea la rezultate posibil si mai slabe. Am incercat si folosirea metodei Otsu, pentru a gasi automat pragul, mergand aproape perfect, avand o singura imagine (1\_42.jpg din setul de antrenare) in care template matching-ul a dat gres.

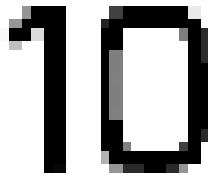


Figura 6: Template-ul pentru piesa 10 dupa operatiile descrise

In al doilea rand, patch-ul trimis catre template matching va fi foarte putin taiat in portiunea de margini, din acelasi motiv, de a nu aduce pixeli nedoriti care sa influenteze corelatia. De asemenea, vom mari patch-ul cu o margine de pixeli albi, folosind functia *copyMakeBorder* din OpenCV. Acest lucru este extrem de important din moment ce template-urile sunt glisate pe patch-ul dat, astfel ca este nevoie de o dimensiune suficienta. Totodata, template-urile pot contine numerele putin decalate de centrul patratului, asa ca este nevoie de o marja suficienta pentru a le gasi si a face potrivirea, lasand, astfel, mai mult loc pentru pozitionare.

In final, vom folosi functia *matchTemplate* si vom gasi similaritatea maxima obtinuta cu unul din template-uri pentru a identifica numarul corect al piesei. Este de mentionat faptul ca verificările facute in acest punct, la fel ca in cazul identificarii pozitiei noii piese, sunt sensibile la rezultatele precedente, influentandu-se una pe cealalta si putand propaga greselile mai departe.

## 5 Task 3 - Calcularea scorului

Pentru aceasta parte, vom retine scorul adus de adaugarea fiecarei piese in cadrul unei mutari a unui joc, pentru a calcula ulterior scorurile.

Verificările ecuațiilor sunt facute dupa regulile descrise ale jocului, retinand, intr-o matrice, fiecare scor adus de o mutare in cadrul unui joc. Vom avea grija sa verificam toate ecuațiile posibil obtinute din cele 4 directii, tinand cont si de conditionarea sau bonusul pozitiei pe care ne aflam.

Apoi, parcurgand fisierele cu rundele jucatorilor, putem determina scorurile obtinute prin insumarea elementelor din matricea respectiva pentru un interval de mutari. In acest punct devine folositoare retinerea tuturor informatiilor din tabla prin matricea cu elemente de tipul clasei mentionate la inceput.

La final, scriem rezultatele obtinute in fisierele de scor, alaturi de mutarile identificate anterior.