Documentație Proiect 1 - Concepte și Aplicații în Vederea Artificială

Carina Săicu

Noiembrie 2024

1 Formatarea Pozelor

Înainte de a rezolva task-rile propriu-zise, am creat câteva funcții care să proceseze imaginile, astfel încât să le aduc la un format mai ușor de manipulat ulterior.

Prima funcție pe care am implementat-o este ***apply_filter_for_table***, care aplică un filtru asupra imaginii originale. Acest filtru evidențiază masa pe care este așezată tabla de joc, colorând-o în negru. Pentru a determina valoarea optimă a filtrului, m-am bazat pe codul oferit în laboratorul 7, care mi-a permis să analizez efectele diferitelor filtre și să le folosesc în avantajul meu.

Funcția primește o imagine ca input și returnează două rezultate:

- masca (o imagine binară alb-negru ce indică regiunile filtrate);
- imaginea originală, dar filtrată, păstrând doar regiunile luminoase specificate.

Apoi am creat funcția ***extract_table*** care are scopul de a identifica și extrage tabla de joc dintr-o imagine, presupunând că aceasta are o formă dreptunghiulară bine definită.

Procesul începe prin detectarea contururilor regiunilor vizibile din mască, iar din acestea se selectează conturul cel mai mare, am presupus că acesta reprezintă tabla de joc. Ulterior, colțurile acestui contur sunt ordonate corect, pentru a asigura că se identifică cele patru colțuri ale tablei. După aceea, se aplică o transformare de perspectivă care redimensionează tabla într-un pătrat standardizat de dimensiune 810×810 pixeli.

Funcția primește două intrări:

- frame (imaginea originală, utilizată pentru extragerea tablei);
- mask (masca binară creată de un filtru anterior, folosită pentru a izola regiunile relevante).

Rezultatul este imaginea decupată, care conține doar tabla de joc.

După extragerea tablei de joc, am dorit să decupez și careul central, adică doar partea relevantă în care pot apărea noi piese. Pentru acest lucru, am creat funcția ***filter_and_extract_square***, care combină principiile celor două funcții menționate anterior. Aceasta aplică un filtru asupra imaginii decupate cu tabla de joc, transformând în negru zonele care nu mă interesează (partea albastră), iar apoi extrage careul central pe baza filtrului aplicat.

După toate aceste prelucrări, tabla a ajuns într-o formă mult mai simplă, păstrând doar zona care mă interesează, unde se adaugă piesele. Acest lucru face mai ușoară rezolvarea taskurilor următoare.

2 Task 1

Pentru realizarea acestui task am folosit o matrice de dimensiune 14x14, care reprezintă starea actuală a tablei de joc. De exemplu, la prima imagine, matricea conține valoarea -1 în toate celulele (indicând că în acele poziții nu există nicio piesă plasată), cu excepția pozițiilor (7,7) (unde valoarea este 1), (7,8) (unde valoarea este 2), (8,7) (unde valoarea este 3) și (8,8) (unde valoarea este 4). Aceste valori reflectă configurarea inițială a tablei înainte de a fi plasată vreo piesă.

Matricea este actualizată la fiecare pas și este utilizată pentru acest task, deoarece o piesă poate fi plasată doar dacă există deja o altă piesă într-o celulă adiacentă. În plus, funcția verifică dacă o anumită celulă din matricea binară se află într-o poziție "favorabilă", respectând astfel regulile impuse.

Totodată, pentru realizarea acestui task am creat mai multe funcții:

- ***split_into_cells*** - imparte o imagine a careului de joc de dimensiune 810×810 pixeli într-o grilă de 14×14 celule pătrate, fiecare reprezentând un pătrat de pe tablă. Fiecare celulă decupată este adăugată într-o listă bidimensională, reprezentând grila completă;

- ***is_favorable_position*** - Aceasta functie verifică dacă o celulă din matrica mea este într-o poziție "favorabilă". Poziția este considerată favorabilă dacă celula respectivă are valoarea -1 și este adiacentă cel puțin unui vecin care nu are valoarea -1. Această funcție primește drept parametri un rând, o coloană (pentru a verifica o poziție specifică) și matricea de poziții. Funcția returnează true (adevărat) dacă poziția respectivă este validă conform regulilor, sau false (fals) în caz contrar;

- ***compare_cells*** - această funcție compară valorile din celulele unei matrici pentru a identifica poziția cu cea mai mare diferență (compararea se face între celulele a două imagini consecutive pentru a identifica locul în care

a apărut o piesă nouă). După ce găsește celula cu diferența maximă, apelează funcția ***is_favorable_position*** pentru a verifica dacă poziția respectivă este favorabilă. Dacă poziția identificată nu este favorabilă, funcția continuă căutarea, analizând următoarea celulă cu cea mai mare diferență. În final, dacă găsește o celulă care este favorabilă și are diferența maximă, aceasta este poziția căutată.

3 Task 2

Pentru a realiza acest task, am creat un director (*'cifre'*) care conține șabloane pentru fiecare număr în parte. M-am bazat pe imaginile auxiliare furnizate ca materiale. Am desenat grid-ul pentru celule și am extras fiecare celulă într-un director separat. Ulterior, am aplicat o serie de transformări asupra imaginilor pentru a le face mai ușor de comparat. Transformările utilizate sunt detaliate în funcțiile ***trim_edges***, ***apply_filter_for_token*** și ***crop_token***.

Pentru realizarea acestui task am creat mai multe funcții:

- ***find_valid_neighbors*** - această funcție primește ca argumente rândul și coloana unde a fost detectată cea mai mare diferență între imagini (indicând poziția în care a fost plasată piesa curentă) și matricea de poziții. Funcția construiește o listă de liste, fiecare sublistă conținând una sau două valori care reprezintă vecinii valizi pentru poziția dată. Un vecin este considerat valid dacă valoarea sa este diferită de -1;

- ***calculate_operations*** - pentru această funcție, am folosit logica celei anterioare. Am parcurs lista de liste returnată de funcția ***find_valid_neighbors*** și am verificat dacă fiecare sublistă conține cel puțin două elemente consecutive care pot fi considerate valide pentru un calcul. Apoi, am transmis acești parametri funcției mele, care a generat un vector cu toate numerele posibile ce pot apărea la acea poziție;

- ***find_best_match*** - această funcție compară o imagine a unei piese plasate pe tabla de joc cu un set de șabloane salvate, pentru a determina care șablon se potrivește cel mai bine cu piesa. Rezultatul este valoarea piesei care corespunde cel mai bine imaginii și o valoare numerică ce indică diferența minimă dintre piesă și șablonul ales. Pentru a obține o precizie mai mare în procesul de comparare, imaginile au fost supuse unor prelucrări înainte de a fi comparate. Aceste prelucrări sunt detaliate în funcțiile descrise în secțiunile următoare;

- ***trim_edges*** - această funcție decupează marginile unei imagini, reducând dimensiunea acesteia. Am fost nevoită să utilizez această funcție pentru ca, în funcțiile următoare, decuparea imaginilor să se realizeze cu precizie exactă în jurul numerelor;

- ***apply_filter_for_token*** - această funcție aplică un filtru care face fundalul negru și cifrele albe, pregătind imaginea pentru decupare. Acest filtru este aplicat asupra imaginii rezultate în urma apelului funcției ***trim_edges***;
- ***crop_token*** - această funcție decupează imaginea astfel încât să păstreze doar numărul (elementul de interes), eliminând cât mai mult din fundal.

4 Task 3

Am rezolvat acest task parcurgând liniile fișierului cu ture pentru a determina intervalele de fișiere asociate fiecărei ture. Apoi, pentru fiecare fișier din interval, am extras valoarea numerică utilizând fișierele generate anterior (din taskurile 1 și 2). Pentru a aplica multiplicatorii specifici, am definit două liste de poziții speciale, asociate multiplicării cu 2 sau 3. Valoarea ajustată este adăugată la un scor cumulativ, care se resetează la începutul fiecărei ture. Rezultatele sunt stocate și scrise într-un fișier de scoruri pentru fiecare joc.

5 Concluzie

Proiectul a avut ca scop dezvoltarea unui sistem automat pentru calcularea scorului în jocul Mathable, un joc similar cu Scrabble, dar bazat pe formarea de ecuații matematice. Soluția propusă procesează imaginile tablei de joc, identifică piesele plasate, verifică validitatea pozițiilor și calculează punctajele ținând cont de bonusurile și constrângerile pătratelor speciale.

Prin integrarea funcțiilor de prelucrare a imaginilor și logică specifică jocului, sistemul determină scorurile finale pentru fiecare tură, oferind rezultate precise și automate.