Extragerea de informatii din imagini cu careuri sudoku

Stanciu Andrei Calin

Solutia realizata in cadrul acestui proiect se imparte in doua sectiuni logice: rezolvarea taskului 1 si rezolvarea task ului 2; aceasta secionare se imparte la randul ei in mai multe componente (unele comune intre task-uri), dar in principiu primitivele de prelucrare a imaginilor au fost utilizate in mod omogen pe tot parcursul implementarii.

In cele ce urmeaza, voi prezenta pe rand implementarea primului task, urmata de implementarea celui de-al doilea task, prezentata in mod analog.

Primul task

O reprezentare simplista a rezolvarii primului task ar fi urmatoarea:

pentru fiecare imagine img:

extragerea chenarului sudoku ----> impartirea in chunk-uri (9x9) ----> verificare daca chunk este gol sau nu (---> daca nu este gol, template matching pentru cifre) ----> salvare output

Extragerea chenarului:

Aceasta etapa se ocupa cu extragerea centrului imaginii (strict careul sudoku) dintr-o imagine initiala. La randul ei, aceasta extragere poate fi impartita in doua subetape: extragerea colturilor careului, rotirea imaginii pentru a alinia cu axele de coordonate muchiile careului, si trunchierea imaginii pentru a pastra numai chenarul.

Extragerea colturilor:

OBSERVATIE: codul acestei functii a fost adaptat dupa ideea prezentata in timpul laboratorului de CAVA.

Cautarea incepe prin a aplica o secventa de filtre asupra imaginii initiale, urmata de identificarea de muchii si cautarea unor minime / maxime:

* se filtreaza imaginea cu un filtru median, in paralel cu un filtru gausian, si se realizeaza o diferenta peste cele doua

* se aplica un threshold care transforma imaginea in una binara - va contine marginile cele mai groase, si bucati din cifre

* aplicarea algoritmului Canny, pentru detectarea acestor muchii

* gasirea unor coordonate pe baza muchiilor anterioare

* extragerea punctelor de coordonate minime (colturile), si a celor care defineau o arie cat mai mare

Parametrii pentru filtrele aplicate initial au fost cautati cu ajutorului unui gridsearch, desi o alegere la nivel intuitiv a lor oferea rezultate satisfacatoare in anumite situatii, selectarea lor manuala nu s-a dovedit suficient de precisa pentru nevoile curente.

Rotirea imaginii si extragerea chenarului:

A fost realizata prin aplicarea unei matrice de rotatie construita pe baza pantei dreptei care defineste latura superioara a conturului, dupa care aceasta imagine rotita a fost trunchiata cu ajutorul tuturor punctelor de coordonate determinate la pasul anterior.

Impartirea in chunk-uri:

Din moment ce toate imaginile prelucrate anterior au chenarul cu laturile paralele cu axele de coordonate, si in plus chenarul ocupa in totalitate imaginea, imartirea s-a putut realiza in mod trivial, luand in calcul doar un padding pentru a compensa cu grosimea conturului, si cu un usor bias observat empiric, in care perspectiva imaginii era usor "aplecata" a.i chunk-urile de pe ultimele linii erau cu 1-2 pixeli mai inalte decat late.

Verificarea unui chunk:

Fiecarui chunk i s-a aplicat un threshold, urmat de un blur median, pentru a "netezi" fundalul, astfel incat (pe cat posibil) chunk-urile goale sa fie in cea mai mare parte albe in interior, iar cele care contineau cifre sa pastreze "urme" de contur negru in interior.

Dupa prelucrarea chunk-ului, se aplica un matching de template uri intre chunk si un template "empty" care are dimensiunea 0.7 x 0.7 relativa la dimensiunea unui chunk; functia folosita pentru matching este distanta L2 intre pixeli, cu normalizare: in momentul in care un chunk este gol, template ul empty are sanse foarte mari sa se suprapuna paste o zona complet alba a chunk ului, drept pentru care in imaginea rezultata din matching, minimul sa fie aproape de 0, pe cand daca imaginea contine seminificativ de multi pixeli de culoare neagra in mijloc, template ul empty (aproape ca) nu va avea ocazia sa se suprapuna peste o regiune complet alba, si deci minimul rezultat din template matching va fi cu mult peste 0. APlicand un threshold determinat tot prin grid search, chunk urile se pot clasifica cu o acuratete foarte mare ca fiind "pline" sau "goale".

Determinarea cifrelor:

In cazul in care se opteaza pentru identificarea cifrelor, in urma identificarii chunk-urilor "pline", asupra lor se va aplica inca un template matching care determina cifra corespunzatoare:

* template urile pentru fiecare dintre cifre au fost selectate manual dintre sample-urile de antrenare, avandu-se in vedere chunk-uri cu marginea cat mai subtire, si o claritate cat mai mare a cifrelor din interior.

* pentru fiecare imagine initiala, se incarca template urile si se redimensioneaza a.i. dimensiunea unui template sa fie fixa, raportata la dimensiunea oricarui chunk din imaginea curenta

* cifra este aleasa pe baza valorii maxime dintre maximele pentru fiecare matching in parte, realizat cu functia care determina coeficientul de corelatie, peste imagini mormalizate

Template urile au fost redimensionate pentru a avea o dimensiune strict mai mica decat chunk-urile verificate, pentru ca matching-ul sa fie invariant la usoare descentrari ale locatiilor cifrelor, in raport cu coordonatele ce definesc acel chunk.

Aceasta modalitate de determinare a cifrelor funcioneaza datorita variatiei destul de mici (spre non-existenta) intre chunk-urile care contin aceeasi cifra.

Al doilea task

In mod asemanator cu primul task, solutionarea celui de-al doilea task arata in felul urmator:

pentru fiecare imagine jigsaw img:

extragerea chenarului sudoku ----> determinarea zonelor ----> impartirea in chunk-uri ----> verificare daca chunk este gol sau nu (---> daca nu este gol, template matching pentru cifre) ----> salvare output

Extragerea chenarului:

Se utilizeaza exact aceeasi implementare cu cea de la task ul 1.

Determinarea zonelor:

Detectarea zonelor are loc in doua sub-etape: detectarea contururilor interioare, si "umplerea" pe baza acestor contururi.

Detectarea contururilor interioare:

Pentru detectarea contururilor s-a utilizat la baza tot procedeul de template matching:

* initial, s-au selectat doua template uri, manual, dintre sample-urile de antrenare, care contin aproximativ "la mijloc" o bordura, peste care s-au aplicat mai multe filtre, astfel incat, la final, bordurile din template-uri sa apara albel, usor blurate, pe un fundal complet negru

* pentru fiecare imagine de jigsaw, s-au aplicat mai intai cateva filtre, aproape identice cu cele folosite pentru determinarea chenarului din imaginile initiale: filtru median, gaussian, diferenta, threshold, Canny, urmate de detectia muchiilor cu ajutorului spatiului Hough - acestea sunt menite sa detecteze muchiile interioare mai groase, care delimiteaza zonele

* pe baza muchiilor descoperite anterior, o parte dintre ele sunt indepartate pe baza unghiului pe care acestea il formeaza raportat la axele de coordonate, dupa care imaginea este reconstruita exclusiv pe baza acestor muchii

* imaginea nou obtinuta se imparte in chunk-uri cu un offset de (chunklen // 2), pentru ca fiecare chunk sa "cada peste o muchie verticala / orizontala"

* pentru fiecare astfel de chunk, se aplica matching cu template-ul pentru muchii orizontale / verticale (in functie de caz): matching ul are loc cu L2 peste imagini normalizate, iar valoarea minima a matching ului este comparata cu un threshold determinat cu grid search, in urma caruia se decide daca chunk-ul respectiv contine o muchie sau nu (procedeul este analog cu cel al determinarii daca un chunk este gol sau plin)

* rezultatele se retin intr-o matrice

Umplerea:

Fill-ul se realizeaza pe baza contururilor determinate anterior, prin parcurgerea imaginii impartite in chunk uri 9x9 in latime (BFS). In urma umplerii, regiunile sunt complet determinate, numerotate corespunzator cerintei.

Verificarea unui chunk:

A doua parte a rezolvarii celui de-al doilea task se aseamana cu rezolvarea primului task, si anume detectarea chunk urilor goale, si eventual cifrele din cele pline.

Imaginea se inparte tot in chunk-uri de 9x9, iar verificarea de tip "chunk gol/plin" se realizeaza identic cu verificarea de la task-ul 1.

Pentru determinarea cifrei din interior, se aplica un procedeu asemanator cu cel de la task-ul 1, cu o diferenta: imaginile jigsaw fiind de doua tipuri (color, sau color dar cu nuante predominat gri) - este nevoie de o diferentiere intre aceste doua sub-tipuri de imagini, realizata in felul urmator:

* pentru fiecare imagine, se alege o zona arbitrara (concret, in implementare se alege un pixel aproape de coltul "stanga-sus") si se compara valorile de pe canalele RGB: daca diferenta depaseste un threshold (ales empiric egal cu 7), imaginea este jigsaw color, altfel, este jigsaw gri - in functie de aceasta diferentiere, se folosesc seturi diferite de template-uri pentru cifre