Universitatea Tehnica din Cluj-Napoca
Facultatea de Automatica si Calculatoare
Departamentul Calculatoare

Sudoku Solver

Student: Szabolcs-Marton BAGOLY, 30234

Indrumator: Asist. Dr. Ing. Anca CIURTE

2017

1. Introducere

Sudoku este un puzzle, un joc popular de logica. Puzzle-ul apare de multe ori la sfarsitul revistelor, ziarelor sau chiar in reviste intregi exclusiv cu puzzle-uri Sudoku. Deoarece nu necesita cunostinte avansate in nici un domeniu, jucatorul se bazeaza doar pe logica este un joc popular, indiferent de varsta jucatorului. Adesea, puzzle-urile sunt complicate si solutiile nu sunt disponibile. Exista o varietate de programe si aplicatii web, care ajuta jucatorul in aflarea solutiei, insa aceste programe necesita introducerea manuala a campurilor completate in faza initiala.

O metoda mai prietenoasa si rapida este utilizarea domeniul procesarii imaginilor in rezolvarea acestei probleme. Cu ajutorul unui dispozitiv se fotografiaza puzzle-ul si, in conditiile in care poza reda cu o claritate suficienta continutul puzzle-ului, dupa o procesare scurta programul afiseaza pe un ecran poza modificata, astfel incat pe pozitiile casutelor nemodificate apar cifrele corespunzatoare, satisfacand constrangerile necesare.

2. Studiul bibliografic

In prezent exista diverse aplicatii care rezolva problema anuntata in capitolul precedent. Cele mai raspandite aplicatii sunt cele de aplicatii mobile, fiindca, de obicei, dispozitivele mobile incorporeaza atat un ecran cat si o camera de luat vederi, deci generarea si afisarea imaginii nu ridica probleme. Pasii necesari dezvoltarii aplicatiei denumita **Sudoku Grab** este prezentata pe pagina [2]. O metoda diferita se prezinta in [3].

Primul pas de efectuat este binarizarea imaginii. Apoi se extrage cea mai mare component conexa din imagine. Pentru extragerea celor 81 de casute exista diferite abordari. In general, pentru realizarea acestui task se foloseste transformata Hough pentru detectia liniilor si in continuare se face partitionarea in casute. Aceasta metoda poate ridica problem atunci cand foaia de hartie despre care se face poza nu este pe o suprafata plana si apar curburi in liniile de margine al puzzle-ului. In articolul [4] se prezinta o abordare diferita a detectiei casutelor mici: fiecare celula este identificata individual prin estimarea initiala al centrului celului si apoi corectarea centrului cu ajutorul elementelor structurale. Folosind aceasta metoda se poate imbunatati performanta algoritmului in cazul in care liniile din care este alcatuit puzzle-ul nu sunt perfect drepte.

Dupa pasul de identificarea celulelor individuale, se identifica cifrele din cellule, daca ele exista. Si in acest caz apar implementari diferite: in metoda prezentata in [2] se foloseste o retea neurala pentru recunoasterea celor 9 cifre. In general, aceasta problema se rezolva folosind algoritmi de pattern recognition si optical character recognition.

Ultimul pas al algoritmului consta din rezolvarea efectiva a puzzle-ului avand la dispozitie matricea de 9x9. Multe solutii implementeaza o simpla metoda de backtracking, care intr-adevar genereaza solutia astfel incat aplicatia poate fi utilizata in timp real. Insa, daca tinem cont de regulile jocului, aceasta metoda se poate imbunatati cu constrangeri pe domeniu reducand mult spatiul de cautare. O astfel de metoda implementata apare in articolul [5].

3. Metoda propusa

In *Figura 1* se prezinta pasii prin care imaginea initiala este trecuta pana in faza in care se decupeaza fiecare celula din puzzle-ul Sudoku, se identifica cifra din celula (daca aceasta exista) si se memoreaza. Dupa prelucrarea imaginii, modelul puzzle-ului este complet si urmeaza procesul de gasire a solutiei problemei prin metoda backtracking si afisarea solutiei obtinute.

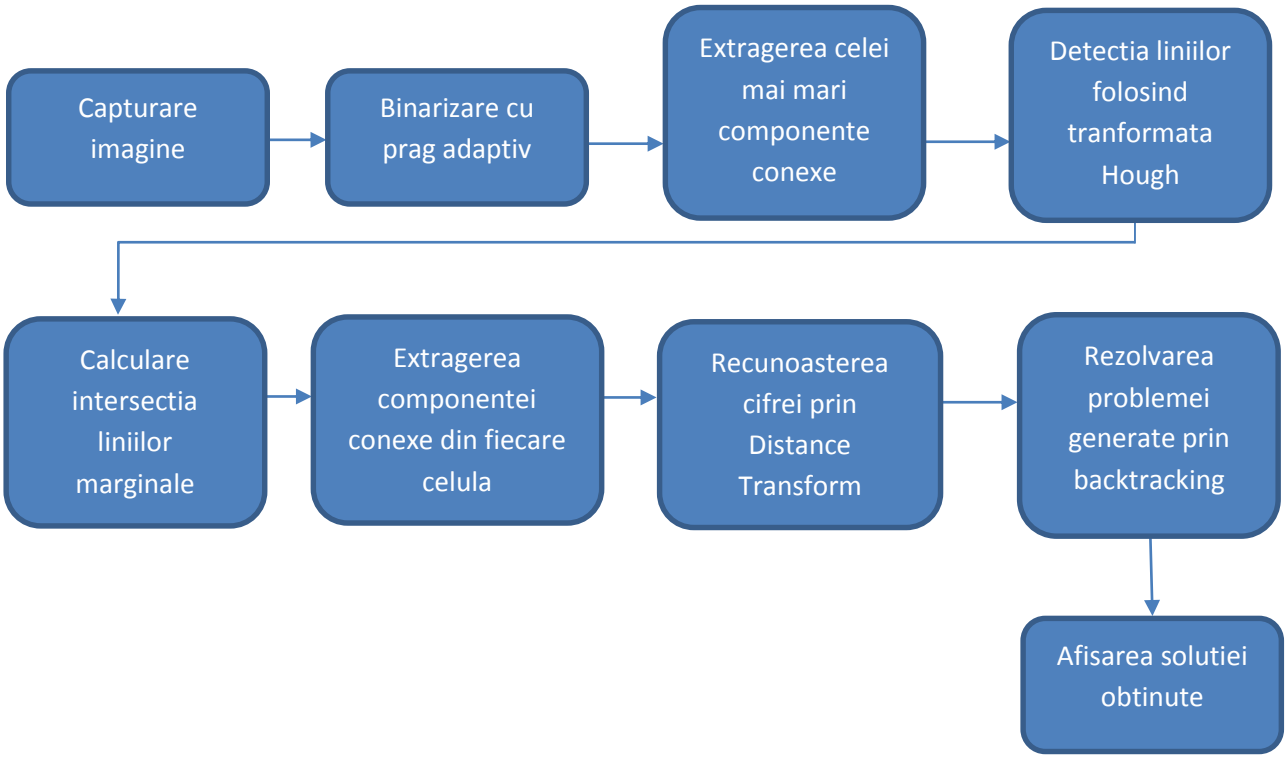


Figura 1. Pasii algoritmului de rezolvare Sudoku

Primul pas dupa incarcarea imaginii in memoria de lucru este binarizarea acesteia folosind metoda pragurilor adaptive. Dupa aplicarea acestei transformari obtinem o imagine cu fundalul negru in care componentele imaginii (linii, cifre, texte auxiliare) au culoarea alba.

Urmatorul pas consta din cautarea celei mai mari componente conexe, parcurgand toti pixelii imaginii si etichetand componentele separate. Se presupune ca imaginea capturata are ca obiectiv principal reprezentarea matricei Sudoku. Avand in vedere aceasta presupunere, consideram ca cea mai mare componenta conexa identificata reprezinta forma in care se afla cifrele matricei Sudoku.

Folosind dreptunghiul identificat la pasul anterior, se calculeaza cele 4 puncte de extrem al acestuia. Un dreptunghi este definit de 4 linii drepte. Cu ajutorul transformatei Hough se detecteaza liniile drepte din imagine. Se reduce numarul de linii detectate prin contopirea acelor linii care sunt foarte apropiate unul de celalalt. Dupa reducerea numarului de linii, se calculeaza cele 4 puncte de extrem care definesc dreptunghiul. Un punct de extrem se defineste prin intersectia a doua linii marginale. Folosind ecuatia matematica a liniei se obtin aceste puncte.

Cu ajutorul celor 4 puncte de extrem se identifica pozitia casutei in imaginea originala si se creeaza o noua imagine in care punctele de extrem apar pe marginea imaginii, deci imaginea astfel obtinuta contine doar matricea de cifre. Urmatorul pas este detectarea celulelor individuale in care se afla cifrele cautate. Acest task se realizeaza prin impartirea uniforma a inaltimii/latimii imaginii. Desi aceasta abordare nu este perfecta (apar probleme cand imaginea contine o foaie indoita), ar trebui sa functioneze in majoritatea cazurilor.

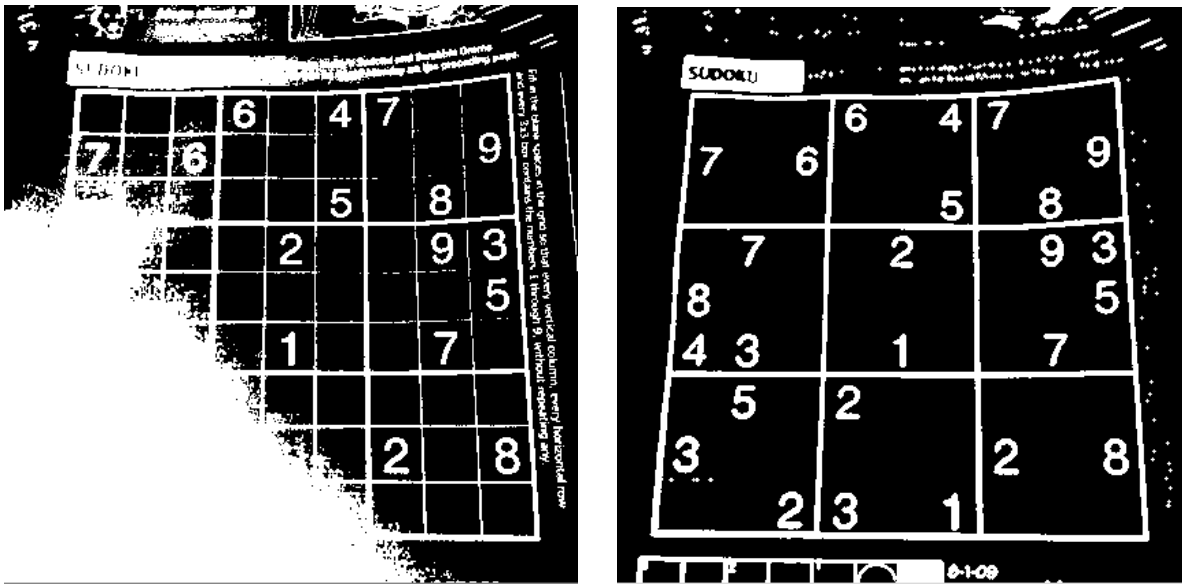
Avand la dispozitie celulele elementare, extragem din ele cea mai mare componenta conexa, aceasta fiind cifra din puzzle. Cifra extrasa se identifica prin metoda pattern recognition, utilizand distance transform, comparand imaginea obtinuta cu template-urile celor 9 cifre cautate. Dupa recunoasterea cifrei aceasta este trecuta in matricea de model al puzzle-ului.

Dupa extragerea tuturor informatiilor relevante din celule, matricea de model este pregatita pentru a fi solutionata utilizand metoda backtracking imbunatatita cu constrageri pe domeniul valorilor posibile pentru a reduce timpul de cautare a solutiei. Fiindca scopul acestui proiect nu consta din metode algoritmice de rezolvare a puzzle-ului Sudoku, ci se axeaza mai mult pe recunoasterea si generarea problemei din imagine, pentru a genera solutia problemei de Sudoku, se utilizeaza un algoritm de backtracking deja implementat, care se gaseste accesand referinta [6] din bibliografie. Ultimul pas consta din afisarea solutiei obtinute.

4. Rezultate experimentale

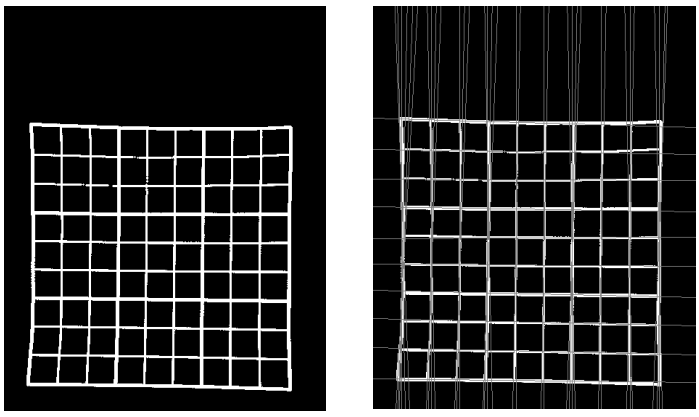
In acest capitol se prezinta rezultatele partiale obtinute in fiecare pas al metodei prezentate in *Figura 1*. Se prezinta si un sumar al concluziilor obtinute dupa fiecare pas si metodele de imbunatatire care s-au aplicat metodei initial propuse.

In primul pas, se aplica o binarizare asupra imaginii citite in modul grayscale. Rezultatele experimentale demonstreaza faptul ca o binarizare simpla, cu un *threshold* fix, in unele cazuri nu poate genera rezultate acceptabile, dupa cum se vede si in figura din stanga:

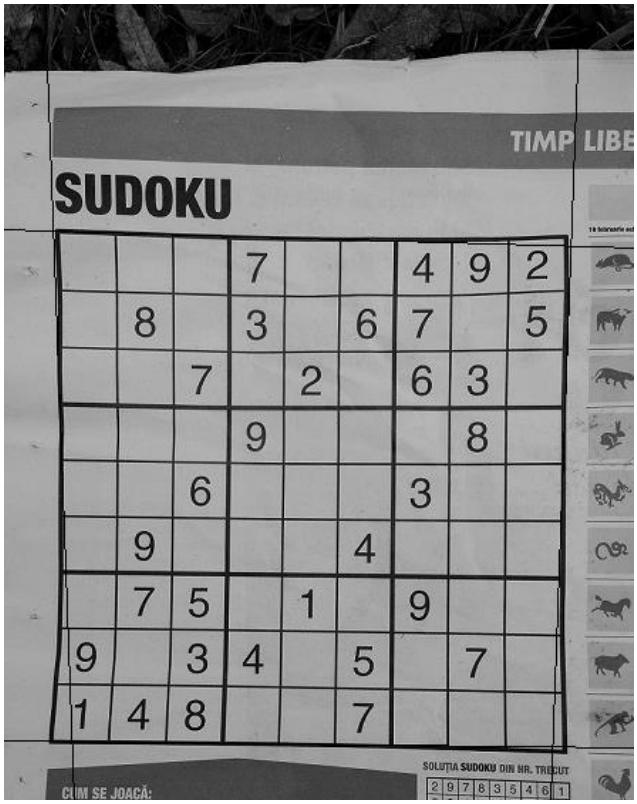


In figura din dreapta, este afisata rezultatul unei binarizari mai complexe: imaginea originale se imparte in 4 segmente egale. Pentru fiecare segment se calculeaza media pixelilor si pentru fiecare segment se calculeaza in *threshold* in functie de media pixelilor din acel segment.

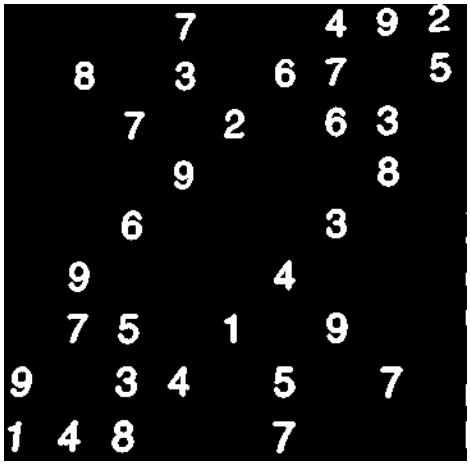
Dupa binarizarea imaginii se extrage componenta conexa cea mai mare. Presupunand ca imaginea, in cea mai mare parte reprezinta puzzle-ul Sudoku, putem presupune ca cea mai mare componenta conexa reprezinta grid-ul puzzle-ului in care se afla cifrele de interes. Mai jos se prezinta rezultatul acestui pas:



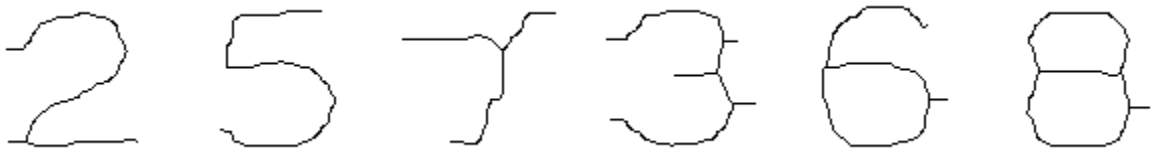
In continuare folosind transformata Hough se detecteaza toate liniile din imaginea cu componenta conexa cea mai mare obtinuta anterior. Dupa ce se extrag toate liniile, se continua cu combinarea liniilor care seamana foarte mult, astfel reducand semnificativ numarul de linii. Se gasesc liniile marginale - aceste linii vor define conturul grid-ului in care se afla cifrele de interes. Prin gasirea intersectiilor acestor linii marginale se obtin cele 4 puncte, care reprezinta cele 4 colturi al dreptunghiului in care se afla puzzle-ul.



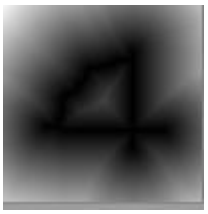
Utilizand aceste 4 puncte de margine, din imaginea originala se extrage o image, in care se afla doar grid-ul. Dupa binarizarea imaginii si ascunderea componentei celei mai mare – adica liniile tabelului, imaginea arata astfel:



Avand la dispozitie imaginea cu grid-ul binarizat, se parcurge imaginea pentru a extrage din fiecare celula componenta cea mai mare conexa – adica cifra pozitiei din grid. Fiindca toate template-urile sunt imagini in care cifrele sunt centrate, si aici se centreaza toate cifrele. Dupa incercarea mai multor experimente cu construirea template-urilor de distance transform din imagine, metoda cea mai precisa s-a demonstrat a fi construirea unui template pe baza scheletului cifrelui. Astfel dupa extragere si centrare, se aplica o metoda de scheletizare asupra imaginii cu cifre. Se obtin rezultate de forma:

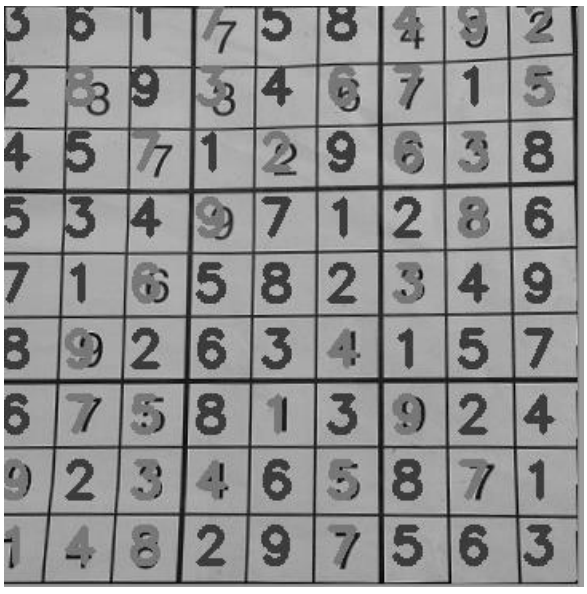


Se incarca in memorie imaginile template si se construiesc matricile de distance transform pentru



fiecare template. In acest pas trebuie aleasa un comprosim intre viteza si precizie: cu cat incarcam si folosim mai multe tipuri de templates, cu atat detectarea cifrelui este mai precise, insa viteza de lucru creste semnificativ. Utilizand 1 singur set de templates, timpul de lucru este aproximativ **2.3 secunde.** In cazul in care s-au utilizat 5 seturi de templates acest numar a crescut la **5 secunde.**

Dupa ce cifrele grid-ului sunt detectate, printr-o metoda de backtracking se incearca rezolvarea puzzle-ului. In mod logic, in cazul in care cifrele nu sunt recunoscute in mod corect, algoritmul nu gaseste o solutie reala a problemei si se afiseaza un mesaj de eroare. Daca operatia are success, rezultatul obtinut se va tipari pe imaginea initiala. In imaginea de mai jos se prezinta rezultatul programului in caz de succes: cifrele mai deschise sunt cele recunoscute din grid, iar cele mai inchise la culoare se genereaza prin algoritmul de rezolvare a problemei Sudoku.



5. Concluzii

Prezentul proiect se conformeaza cerintei initial propuse. Incarcand o imagine, care in cea mai mare parte reprezinta un puzzle Sudoku, programul detecteaza cifrele care alcatuiesc puzzle-ul si, in cazul favorabil se poate ajunge la afisarea solutiei problemei. Constrangerile sub care ruleaza programul este ca fonturile cifrelor sa fie asemanatoare cu cele din care sunt construite template-urile si grid-ul sa fie capturat in conditii bune: adica foaia sa nu fie indoita si luminozitatea imaginii sa fie relativ constanta. In caz contrar, s-au nu se recunosc cifrele in mod corect, sau se poate intampla ca programul sa se comporte intr-un mod total neasteptat, daca la primii pasi nu se face o detective corecta a componentei conexe celei mai mari.

Probleme care apar in mod frecvent, care rezulta in detectarea incorecta a cifrelor apare la calcularea scorului de pattern matching dintre imaginea necunoscuta si imaginea template. Cele mai dese confuzii apar dintre cifrele **3 si 8,** sau cifrele **1 si 7.** O singura greseala in detectarea cifrelor rezulta in imposibilatea generarii unei solutii.

Directiile de dezvoltare in viitor cuprind atat imbunatatirea preciziei algoritmului cat si a vitezei de lucru. Pentru a imbunatati precizia, la recunoasterea cifrelui ar trebui applicate mai multe seturi de templates (acest lucru are ca consetinta cresterea vitezei de lucru) sau implementarea unei alte metode bazata pe metode de clasificare sau retele neurale. Pentru a imbunatati viteza de lucru, s-ar putea beneficia de faptul ca cele mai multe calculatoare moderne sunt sisteme cu procesoare multiple. Astfel, in situatia in care algoritmul permite acest lucru (cum ar fi cazul binarizarii in segmente separate sau calcularea scorurilor de template matching) aplicatia ar putea rula in modul multi-threading astfel imbunatatind timpul de lucru.

Bibliografie

[1] http://sudokumeu.ro/info/

[2] http://sudokugrab.blogspot.ro/2009/07/how-does-it-all-work.html

[3] https://www.codeproject.com/articles/238114/realtime-webcam-sudoku-solver

[4] http://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Wang.pdf

[5] http://byteauthor.com/2010/08/sudoku-solver/

[6] http://www.geeksforgeeks.org/backtracking-set-7-suduku/

http://users.utcluj.ro/~igiosan/Resources/PRS/L4/lab_04e.pdf

http://users.utcluj.ro/~ancac/Resurse/labPRS/L03_hough/prs_lab_03r.pdf