UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Names Reader

propusă de

dna. Profesor Anca Ignat

Sesiunea: *iulie, 2018*

Coordonator științific

Lector dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Names Reader

Tudor Timofte

Sesiunea: *iulie, 2018*

Coordonator științific

Lector dr. Anca Ignat

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele

_____

Data _____ Semnătura

_____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul(a)

……

domiciliul în ……..

născut(ă) la data de ……..…., identificat prin CNP ……..……..……..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza" din Iași, Facultatea de ……. specializarea ……, promoția ……., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: _____

_____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ……

Semnătură

student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Prenume Nume*

_____

(semnătura în original)

Cuprins

Intorducere

Lucrarea are ca scop îmbunătățirea procesului ce privește efectuarea prezențelor în cadrul cursurilor sau laboratoarelor prezentate în cadrul Facultății de Informatică. Tema licenței a mai fost dezbătută anterior,aceasta se concentrează asupra identificării diverselor caractere scrise de mână,dar utilizează un concept diferit în concretizarea unei soluții.

Procesarea imaginilor este un domeniu relativ recent, care evoluează rapid. Această ramură își găsește aplicabilitate în diverse arii de cercetare: medicină, armată, industrie, artă sau acolo unde informația din mediul înconjurător este reprezentată sub formă de imagini. Principala aplicație o reprezintă îmbunătățirea informației conținute de imagini în vederea interpretării unor date sau obținerea și manipularea unor caractere pentru domeniul informatic.

Recunoașterea scrisului de mână presupune capaciteatea unui calculator de a interpreta și descifra diverse caractere, unice, create de urma unui instrument de scris utilizat de o persoană cu scopul de a transmite informații pe un suport fizic. Programul are rolul de a simplifica procesul de notare al prezențelor prin identificarea fiecărui nume în parte ce a fost scris pe foile de prezență.

Modul de lucru presupune scanarea foilor de prezență și încărcarea lor în aplicație. Pasul următor urmărește identificarea caracterelor scrise în pagină și a modului în care sunt grupate pe cuvinte pentru a alcătui numele studenților. Aceste caractere sunt introduse într-un algoritm de similaritate ce incercă să identifice cu o probabilitate cât mai mare fiecare literă în parte ca apoi acestea să fie grupate în cuvinte. Aceste cuvinte obținute sunt apoi căutate într-o bază de date existentă ce conține toate numele sau prenumele posibile și se va aplica, similar ca în cazul literelor, un alt algoritm ce va selecta într-un final numele și prenumele elevului căutat.

Contribuții

Pentru o abordare corectă a temei a fost necesară o documentare clară asupra domeniului ce presupune procesarea de imagini. Acest domeniu a trebuit parcurs de la noțiuni generale până la cele mai complexe pentru a înțelege fondul problemei. Totodată a trebuit găsită cea mai potrivită soluție în privința recunoașterii scrisului de mână, lucru realizat prin încercări repetate ale diverselor metode deja existente în prelucrarea de imagini. A fost ales un proces relativ simplu la bază, însă care a fost adaptat cerințelor lucrării. Proiectul se bazează pe un algoritm ce funcționează pe principiul incercărilor repetate și pe observarea și alegerea celor mai potrivite rezultate obținute în urma mai multor rulări. De asemenea au fost parcurse diverse studii și articole ce prezintă rolul invațării automate în acest context, al recunoașterii caracterelor scrise de mână, și au fost aplicate unele concepte găsite în acestea.

1.Descrierea problemei

Lucrarea are rolul de a identifica numele diverșilor studenți ce au fost notați pe foile de prezență.

Foaia de prezență este scanată și ulterior introdusă în program. Acesta ar trebui să identifice atât individual bucăți din imaginea dată ce reprezintă de fapt caracterele scrise de mână cât și modul în care acestea, grupate, formează cuvinte. Toate aceste secvențe de imagine sunt apoi transformate cu ajutorul procesării de imagini în matrici care sunt identificate ulterior ca și litere ale alfabetului. Cunoscând caracterele scrise, acum putem recrea foaia de prezență în mod electronic.

1.1 Etape

Principapele etape ce trebuie atinse pentru rezolvarea problemei sunt:

- Identificarea caracterelor ce trebuie interpretate
- Identificarea poziției literelor în cuvinte
- Recrearea numelor utilizând interpretarea inițială a caracterelor
- Cautarea și selecționarea numelor potrivite având în vedere rezultatele obținute
- Salvarea numelor obținute ca rezultat final

Principalele impedimente se concentrează asupra creării datelor de antrenament, cu ajutorul cărora sunt identificate caracterele scrise pe foaia de prezență, cât și în procesul efectiv de căutare a literelor, moment în care pot aparea probleme de performanță. Totodată trebuie luat în considerare și unicitatea modului în care fiecare persoană scrie.

1.2 Impedimente

Utilizarea corespunzatoare a bibliotecii OpenCV are un rol foarte important în realizarea cu success a lucrării, o interpretare clară a caracterelor scrise de mână reprezintă un punct crucial în obținerea unor rezultate relevante.

2. Descrierea solutiei

2.1. Procesarea de imagini

Prelucrarea computerizată a imaginii este un domeniu extrem de vast, ce cuprinde arii de dezvoltare teoretică cât și aplicativă, contribuții extrem de variate, de la standarde, concepte și expresii matematice, până la aplicații comerciale și artistice. Prelucrarea imaginii asistată de calculator presupune o serie de principii și procese care fac necesară definirea unor noțiuni specifice. Unul dintre fenomenele cele mai importante care intervine în prelucrarea imaginii cu ajutorul computerului este digitizarea. Principala caracteristică a unei imagini o reprezintă pixelul ce poate fi definit ca cel mai mic element de imagine, acesta determină rezoluția imaginii și are proprietăți relative de intensitate luminoasa și culoare.

Totodată procesarea de imagini prezintă diferite metodologii de lucru, are drept scop realizarea unei imbunătățiri a imaginii prin punerea în evidență a anumitor regiuni, schimbarea luminozității, detecția muchiilor etc., pregătind imaginea în vederea operației de alegere a caracterelor ce trebuie interpretate.Transformările aplicate, în această etapă, unei imagini pot fi de două tipuri: transformări punctuale (modificarea valorii unui pixel se face independent de vecinătatea acestuia) sau transformări locale (noua valoare a pixelului depinde de valorile pixelilor înconjurători).

2.2 Scrisul de mână

Scrisul de mână este scrisul realizat cu un instrument de scris, cum ar fi un stilou sau creion, în mână. Scrierea de mână include atât stiluri de tipar, cât și cursive și este separată de caligrafie formală sau de tip. Cu sistemele informatice și aplicațiile devenind mai inteligente, este logic să ne concentrăm asupra capturii și prelucrării automate a documentelor primite. De-a lungul anilor s-au făcut multe investiții în capacitatea de a recunoaște și de a procesa datele oferite de pe un suport electronic, astfel încât captarea și prelucrarea automată să poată fi realizate. Cu toate acestea, datorită complexității și diferențelor de scriere de mână, același nivel de automatizare este mult mai dificil de realizat.

Tehnologiile de recunoaștere inteligentă a caracterelor (ICR) și tehnologiile de recunoaștere a scrisului de mână (cursiv) și recunoașterea amprentei fără restricții însă acestea pot da rezultate limitate din cauza faptului că scrisul de mână este personal și unic în unele cazuri.

2.3 Pricipala funcție a aplicației, identificarea caracterelor

Compararea caracterelor scrise de mână presupune examinarea mai multor caracteristici:

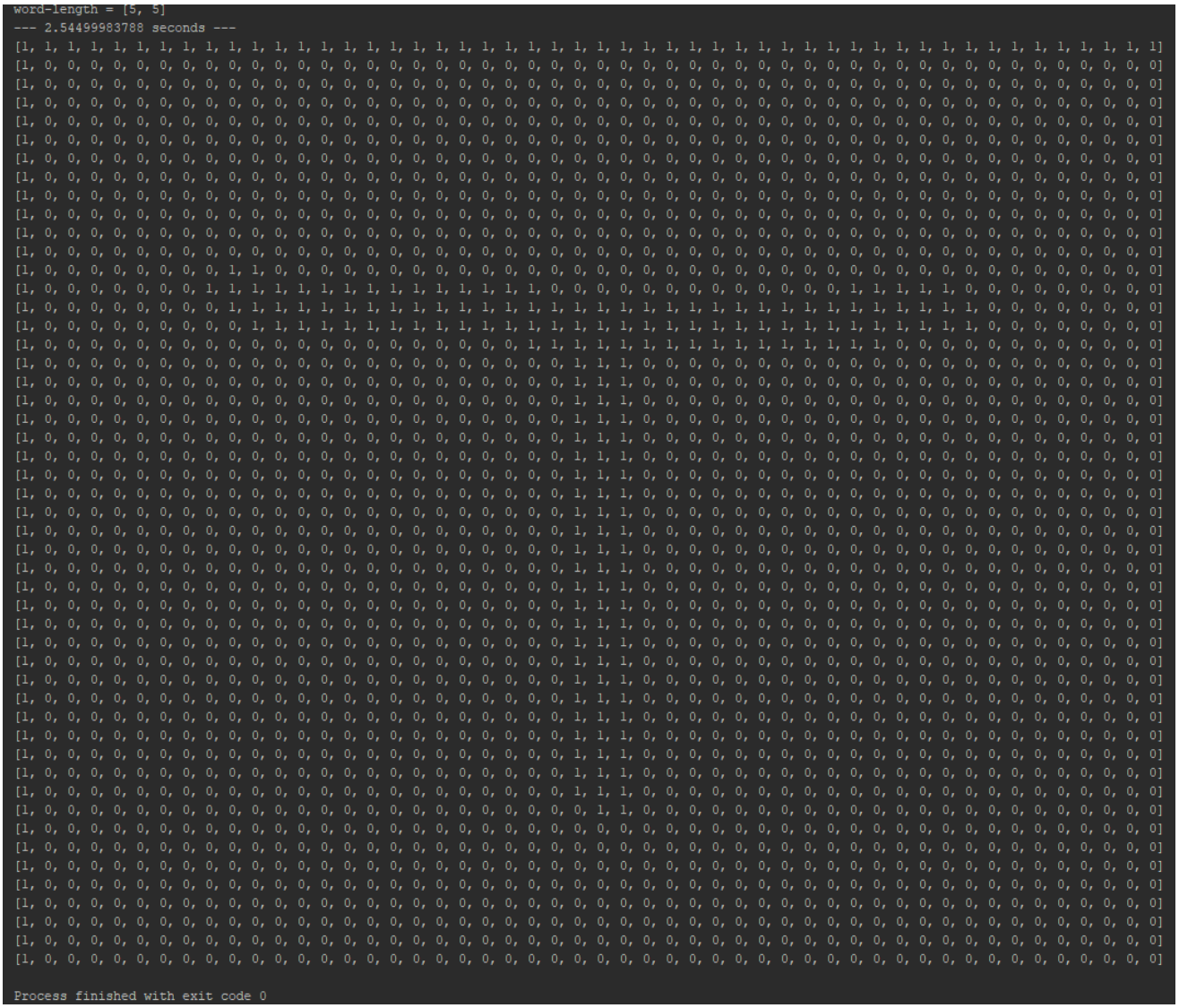
- calitatea liniei
- dimensiunea literelor
- forma neobișnuită sau stilurile unice
- presiunea aplicată instrumentului de scris
- înclinația literei
- plasarea și forma diacriticelor

Acest proces de testare ale proprietăților enumerate mai sus este foarte meticulos și greu de optimizat pentru a obține rezultate relevante pe un mediu informatic. Luând în considerare aceste aspecte s-a decis compararea caracterelor ce trebuie identificate cu o bază de date de antrenament și alegerea unui procent de similaritate între candidații acestui proces.

2.4 Soluția utilizată

Inițial a trebuit găsită o soluție pentru indentificarea formei caracterelor in pagină. Având învedere diversitatea tipurilor de scris, s-a decis implementarea unor standarde ,fapt menționat anterior. Problema nu era reprezentată de forma caracterului în sine, ci de mărimea și indentarea pe care o folosește fiecare persoană. Din cauza continuității scrisului de mână, există situații în care nu putem diferenția informatic două sau mai multe litere. Luând în considerare acest fapt, s-a decis să se apeleze la o liniatură ce desemnează pozițiile în care trebuie scrise literele respective. Acest lucru ușurează foarte mult procesul, pagina scanată poate fi impărțită acum după această liniatură în celule egale cu dimensiuni de 50 pe 50 de pixeli, fiecare având în interior câte un caracter scris.

Imaginea este totodată transformată cu ajutorul bibliotecii OpenCV în alb-negru, culorile reale utilizate în alcătuirea foii de prezență fiind total neimportante. După aplicarea acestei modificări, vom parcurge toate celulele selectate din pagini și vom salva, cu ajutorul bibliotecii numpy, toate matricile asociate acestor spații cu valori de “0” reprezentând spațiile goale și valori de “1” pentru spațiile ce reprezintă urma instrumentului de scris.



În acest punct există un mod de a interpreta și procesa datele, moment în care putem compara fiecare celulă de pe foaia de prezență (o celulă conținând un caracter reprezentat prin valori de 0 și 1 pe o matrice de 50 x 50 pixeli) cu datele de antrenament. Este de menționat că datele de antrenament sunt obținute prin completarea a diverse foi ce se supun acelorași standard și respectă un proces identic cu foaia de prezență în ceea ce privește modul de interpretare a imaginii.

Compararea presupune numărarea tutor valorilor de “1” ce se află pe poziții identice în cadrul ambelor celule ca mai apoi să se aplice un calcul matematic simplu ce returnează un procent prin care se poate determina cât de similare sunt caracterele celor 2 celule.

```
def compare(value1, value2):
    numbersIdentical = 0
    for idx1 in range(0, 50):
        for idx2 in range(0, 50):
            if value1[idx1][idx2] == value2[idx1][idx2] and value1[idx1][idx2] == 1 and
value2[idx1][idx2] == 1:
                numbersIdentical += 1

    if np.count_nonzero(value1) == numbersIdentical:
        return 100
    else:
        if np.count_nonzero(value1) > numbersIdentical:
            arithmeticalValue = (numbersIdentical * 100) / np.count_nonzero(value1)
        else:
            arithmeticalValue = (numbersIdentical * 100) / np.count_nonzero(value2)
        return arithmeticalValue
```

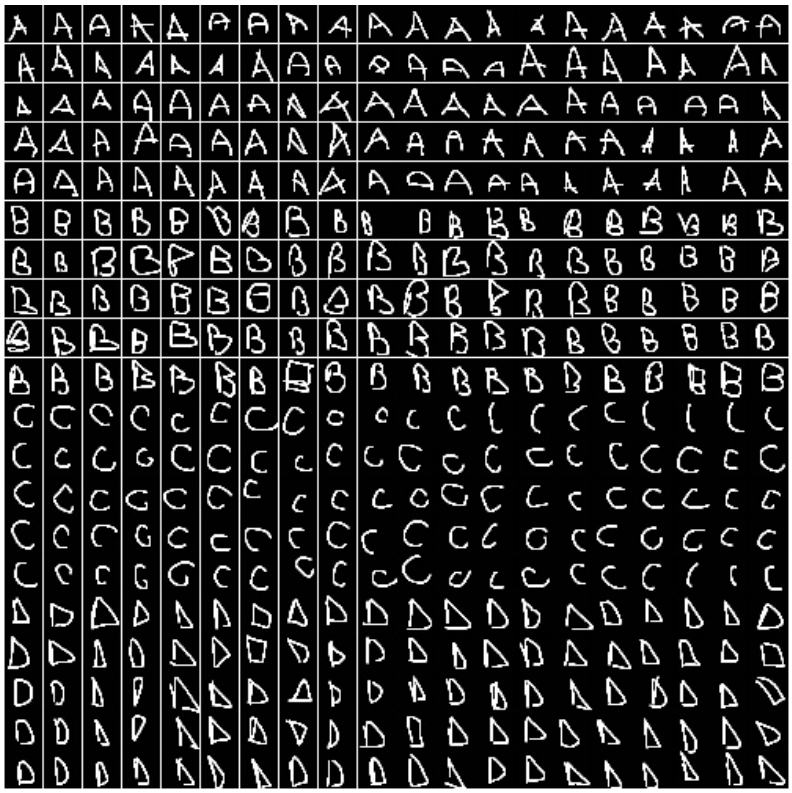
Următorul pas a presupus preluarea corectă a caracterelor ce trebuie identificate și a pozițiilor acestora în cuvinte.

Pentru realizarea acestui lucru s-au parcurs toate celulele aparținând foii de prezență și au fost luate în considerare doar acele zone care aveau în componență măcar o valoare de “1” în întreaga matrice ce o reprezenta, altfel erau considerate zone “goale” și au fost utilizate pentru a marca finalul cuvintelor. Totodată aici au trebuit eliminate și marginile ce delimitau aceste spații pe foaie deoarece biblioteca OpenCV le identifica și pe acestea în cadrul procesării de imagine.

```
if i < 19 and j < 19:
    numberOfZero = 2500 - np.count_nonzero(x[i][j]) + 99
else:
    if i == 19 and j < 19 or i < 19 and j == 19:
        numberOfZero = 2500 - np.count_nonzero(x[i][j]) + 148
    else:
        if i == 19 and j == 19:
            numberOfZero = 2500 - np.count_nonzero(x[i][j]) + 196
```

Datorită acestui proces am obținut clar pozițiile caracterelor scrise pe foaia de prezență și modul în care acestea grupate alcătuiesc cuvinte. Următorul pas a fost identificarea lungimilor numelor trecute pe foaie.

Cu toate informațiile necesare la dispoziție, se poate trece la algoritmul de identificare propriu-zis. Celulele extrase de pe foaia de prezență sunt comparate cu celulele de pe foile de antrenament. (de menționat, foile de antrenament grupeaza câte 4 litere ale alfabetului englez în ordine și prezintă câte 100 de repetiții ale fiecărui caracter).



2.5 Prelucrarea rezultatelor obținute + optimizare proces

În urma acestor comparații sunt obținute procentele de similaritate maximă dintre caracterul ce trebuie identificat ca litera și caracterele de pe foile de antrenament. În total se vor obține 7 astfel de valori, câte una pentru fiecare pagină de caractere și va fi aleasă valoarea maximă dintre acestea. Știind valoarea maximă, putem stabili care a fost litera de antrenament cea mai apropiată ca formă de literă scrisă pe foaia de prezență, astfel putem recrea cuvintele.

Tot aici a fost intalnită și o problemă majoră de performanță. Dacă o foaie de prezență are în medie 25 de candidați, acest lucru presupune existența a măcar 50 de cuvinte, aproximativ 300 de caractere. Totodată în datele de antrenament, cum am menționat și anterior, există 26 de caractere ce se repetă de 100 de ori fiecare sub diverse forme, de unde

rezultă un număr de 2600 de litere salvate. Atât cele 300, cât și cele 2600 de date de antrenament sunt reprezentate pe matrici de 2500 de pixeli (50 x 50). Fiecare comparare a două caractere ar presupune parcugerea acestui număr mare de valori afate într-o matrice, lucru ce ar trebui să se repete de măcar 780.000 pentru identificarea tuturor literelor. Având în vedere aceste aspect, s-a decis optimizarea algoritmului.
S-a efectuat inițial o comparare parțială, caracterul ce trebuia identificat a fost căutat doar în primele 5 date de antrenament ale fiecărei litere pentru a determina o similaritate inițială. Maximul valorilor găsite fiind utilizat apoi pentru a releva ce date de antrenament prezintă probabilitatea cea mai mare de a identifica cu succes litera corespunzătoare. În acest mod, numărul de operații de comparare efectuate a fost redus la aproximativ 230 însă acest lucru reprezintă un factor favorizant al identificărilor fals pozitive.

În acest punct există posibilitatea de a fi interpretat greșit mai multe caractere. Pentru a elimina această problemă, am utilizat liste de nume și prenume preexistente și am comparat similaritatea dintre acestea și cuvintele identificate. Se vor alege din acestea, procentele de similaritate maximă, astfel obținând numele persoanei scrise pe foaia de prezență.

2.6 Optimizare mod de lucru

Timpul de lucru a fost optimizat prin minimizarea căutărilor. Cum s-a mai specificat, numărul de comparații efectuate a fost redus foarte mult, însă acest lucru a avut un impact negativ asupra rezultatelor obținute la identificarea caracterelor.

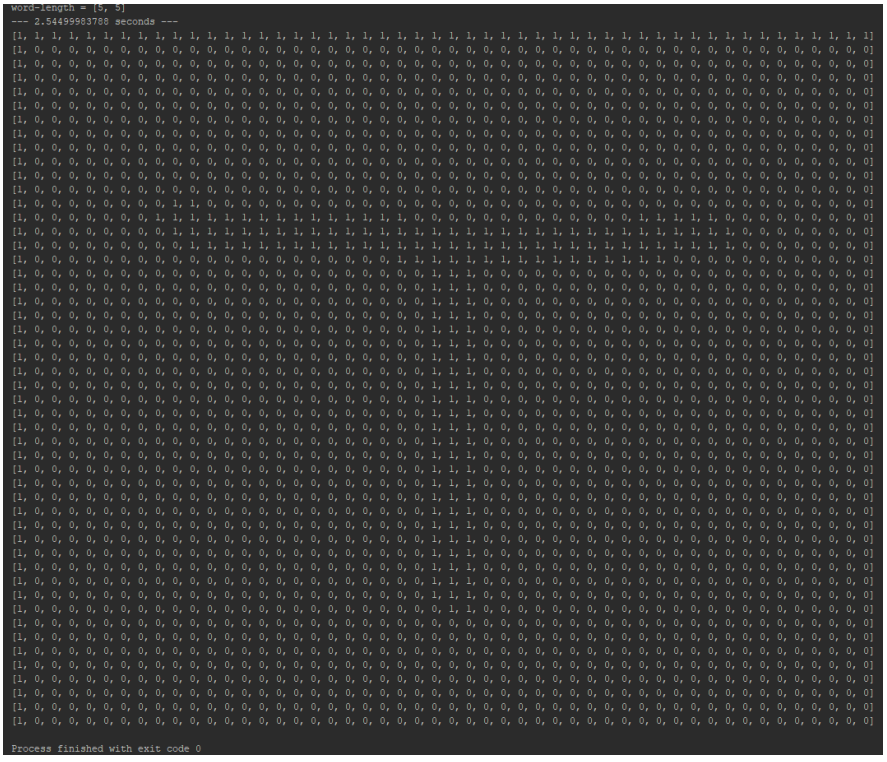
```
C:\Python27\python.exe F:/Licenta/digitRecogOpenCV/TestInstall.py
namePositions = [0, 4, 0, 4]
word-length = [5, 5]
[0, 0, 0, 57, 0, 61, 0, 0, 91, 0, 0, 0, 0, 0, 0, 54, 0, 0, 0, 88, 0, 59, 0, 0, 0, 0]
[0, 0, 62, 0, 0, 0, 0, 59, 0, 0, 0, 67, 0, 0, 64, 0, 0, 0, 57, 0, 75, 0, 0, 0, 0, 0]
[0, 0, 0, 95, 70, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 75, 0, 0, 65, 0, 62, 0, 0, 0, 0, 0]
[0, 0, 74, 0, 0, 0, 70, 0, 82, 0, 0, 0, 0, 0, 99, 0, 0, 0, 0, 66, 59, 0, 0, 0, 0, 0]
[0, 59, 0, 0, 0, 0, 0, 72, 78, 0, 0, 0, 0, 0, 0, 76, 0, 97, 0, 0, 69, 0, 0, 0, 0, 0]
['I', 'U', 'D', 'O', 'R']
[0, 0, 58, 0, 0, 59, 0, 0, 97, 0, 0, 0, 0, 0, 0, 51, 0, 0, 0, 63, 0, 53, 0, 0, 0, 0]
[0, 0, 58, 0, 0, 57, 0, 0, 0, 0, 0, 67, 0, 0, 93, 0, 0, 0, 62, 0, 53, 0, 0, 0, 0, 0]
[93, 0, 0, 0, 60, 0, 0, 0, 77, 0, 0, 0, 0, 0, 69, 0, 0, 0, 67, 0, 58, 0, 0, 0, 0, 0]
[0, 0, 62, 0, 0, 0, 0, 75, 0, 0, 0, 71, 0, 94, 0, 0, 0, 67, 0, 0, 76, 0, 0, 0, 0, 0]
[99, 0, 0, 0, 0, 65, 0, 0, 92, 0, 0, 0, 0, 65, 0, 0, 0, 0, 0, 65, 0, 62, 0, 0, 0, 0]
['I', 'U', 'D', 'O', 'R', 'I', 'O', 'A', 'N', 'A']
--- 152.720000029 seconds ---

Process finished with exit code 0
```

În imaginea anterioară se poate observa că au fost identificate 9 din 10 caractere cu succes întru-un timp de aproximativ 152 de secunde. Valorile ce aparțin array-urilor de mai sus sunt valorile maxime de similaritate ce au fost descoperite pe fiecare fișier de antrenament în parte. Aceste valori ocupă și poziția literei în alfabetul englez.

De asemenea se poate observa caracterul identificat greșit, dar care are o similaritate de 91% cu litera ‘I’. Acest caracter ar fi trebuit să fie găsit ca ‘T’, cum se poate observa și în fișierul de test.

Astfel de interpretări greșite apar din pricina formei unice a fiecărui caracter, precum și din infinitatea de poziții în care acesta poate fi scris în celula sa.



În acest caz se poate observa foarte clar fenomenul descries anterior. Cele două poze conțin, în reprezentarea utilizată în rezolvarea lucrării, un caracter ‘T’ ce trebuie testat și caracterul ‘I’ cu care acesta are o similaritate foarte mare. Acest lucru are loc deoarece ‘piciorul’ ambelor caractere ocupă aproape perfect aceleași poziții în celulă, marcând astfel, similar, valorile de 1 în matrice.

Având în vedere că în fișierele de antrenament nu mai există o alta litera ‘T’ care să ocupe aceiași poziție în matrice, gradul de similaritate este favorabil literei ‘I’ și aici intervine alegerea eroantă a programului.

Cu toate acestea, o rată de succes în valoare de 90% arată un comportament promițător al programului. După introducerea optimizărilor ce au ca scop îmbunătățirea timpilor de execuție, se poate observa însă o scădere a performanțelor algoritmului.

```
C:\Python27\python.exe F:/Licenta/digitRecogOpenCV/TestInstall.py
namePositions = [0, 4, 0, 4]
word-length = [5, 5]
[0, 0, 0, 57, 0, 61, 0, 0, 91, 0, 0, 0, 0, 0, 0, 54, 0, 0, 0, 88, 0, 59, 0, 0, 0, 0]
[0, 0, 62, 0, 0, 0, 0, 59, 0, 0, 0, 67, 0, 0, 64, 0, 0, 0, 57, 0, 75, 0, 0, 0, 0, 0]
[0, 0, 0, 94, 70, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 75, 0, 0, 65, 0, 62, 0, 0, 0, 0, 0]
[0, 0, 74, 0, 0, 0, 70, 0, 82, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 66, 59, 0, 0, 0, 0, 0]
[0, 59, 0, 0, 0, 0, 0, 72, 78, 0, 0, 0, 0, 0, 0, 76, 0, 94, 0, 0, 69, 0, 0, 0, 0, 0]
['I', 'U', 'D', 'O', 'R']
[0, 0, 58, 0, 0, 59, 0, 0, 94, 0, 0, 0, 0, 0, 0, 48, 0, 0, 0, 63, 0, 53, 0, 0, 0, 0]
[0, 0, 58, 0, 0, 57, 0, 0, 0, 0, 0, 67, 0, 0, 57, 0, 0, 0, 62, 0, 53, 0, 0, 0, 0, 0]
[93, 0, 0, 0, 60, 0, 0, 0, 77, 0, 0, 0, 0, 0, 63, 0, 0, 0, 67, 0, 58, 0, 0, 0, 0, 0]
[0, 0, 62, 0, 0, 0, 0, 75, 0, 0, 0, 71, 0, 70, 0, 0, 0, 67, 0, 0, 76, 0, 0, 0, 0, 0]
[94, 0, 0, 0, 0, 65, 0, 0, 92, 0, 0, 0, 0, 64, 0, 0, 0, 0, 0, 65, 0, 62, 0, 0, 0, 0]
['I', 'U', 'D', 'O', 'R', 'I', 'L', 'A', 'U', 'A']
--- 45.4100000858 seconds ---

Process finished with exit code 0
```

În imaginea anterioară este prezentat rezultatul algoritmului după adăugarea optimizărilor necesare. Se remarcă faptul că există o scădere de performanță, fiind identificate corect doar 7 din 10 caractere în cadrul acestui test. Pe de altă parte se poate observa o îmbunătățire considerabilă a timpului de rulare, acesta fiind de aproximativ 45 de secunde, la o treime din valorile înregistrate anterior. Totodată, caracterele interpretate greșit sunt ‘I’, ‘L’ respectiv ‘U’ care ar fi trebuit de fapt să aibe valorile ‘T’,’O’ și ‘N’.

Scăderea de performanță a fost însă combătută prin compararea rezultatelor obținute cu nume reale, deja existente în fișierele aplicației.

Caracterele identificate au fost regrupate pe cuvinte, cum au fost scrise și pe foaia de antrenament. Aceste nume obținute temporar ce conțin greșeli au fost comparate apoi cu toate numele de aceiași lungime ce se află în evidențele programului. Acest proces elimină aproape în totalitate posibilitatea unei greșeli. În toate testele efectuate, a fost gasită o singură greșeală ce înlocuia numele ‘Ioana’ , care ar fi fost corect identificat, cu numele ‘Alexa’. Având în vedere faptul că alegerea caracterelor are o rată aproximativa de 70% de success și că grupările de caractere găsite sunt apoi comparate doar cu alte cuvinte ce au aceiași lungime, pot spune că există o rată de succes în identificarea numelor de cel puțin 95%.

2.7 Abordări anterioare

Având în vedere parametrii problemei discutate anterior soluția prezentată s-a dovedid ca fiind cea mai eficientă. Inițial s-a încercat aplicarea unui algoritm de invățare automată fundamentat pe o bază de date ce salvează un număr de 17 caracteristici diferite ale literelor și a modului în care acestea pot identifica cu success caracterele. Fiecare caracteristica a literelor

a fost notată cu o valoare de la 0 la 15. Acești parametri acoperă mai multe puncte referitor la caracterul scris, de la înălțimea sau lățimea literei până la fondul utilizat de acesta. S-a renunțat la utilizarea acestui algoritm din două motive: unicitatea scrisului de mână și a posibilităților puține de a alcătui cuvinte, fiind limitați numai la nume și prenume.

3.Tehnologii și librării folosite

Soluția a fost redactata în Python 2.7. Python este un limbaj de programare interpretat la nivel înalt pentru programare generală. Creat de Guido van Rossum și lansat pentru prima dată în 1991, Python are o filozofie de design care accentuează lizibilitatea codului, folosind în special spațiu alb. Oferă construcții care permit o programare clară atât pe scări mici cât și pe scări mari.

A fost ales acest limbaj de programare datorită compatibilității sale cu principala bibliotecă responsabilă cu procesarea de imagini, OpenCV. Aceasta prezintă o multitudine de funcții ce au rol în procesarea și manipularea imaginilor. A fost inițial dezvoltata de Intel și este open-source.

4. Concluziile lucrării

4.1 Îmbunătățiri

Proiectul își atinge scopul de a indentifica cu succes numele studenților ce se regăsesc pe foile de prezență, însă reușește acest lucru cu probleme de performanță. Consider că metoda utilizată pentru a citi cu succes litere este potrivită problemei ce trebuie rezolvată însă apar impedimente pe un volum mare de date. De asemenea, programul poate fi utilizat pentru orice tip de caracter atât timp cât există date de antrenament în acest scop, nu doar pentru majuscule.

Îmbunătățiri pot fi aduse atât pe partea de performanță cât și din punct de vedere al procesării foii de prezență. În acest moment, lucrarea utilizează un standard din punct de vedere al spațierii însă, ideal, nu ar trebui să existe limitări în ceea ce privește modul de completare a foii inițiale. Dificultatea găsirii unui mod de a înlatura aceste standarde impuse constă în nenumăratele modalități în care literele pot fi asezate în pagină și în caracterul unic al scrisului de mână pentru fiecare persoană în parte.

4.2 Funcționalități viitoare

Aplicația poate fi utilizată într-o multitudine de situații atât timp cât datele de antrenament sunt specifice scopului pentru care se dorește folosirea ei. Ca funcționalități viitoare pot fi enumerate recunoașterea scrisului cursiv sau a cifrelor. De asemenea, aplicația ar putea fi utilizată și pentru identificarea sau corelarea a diverse obiecte sau forme din variate imagini (atât timp cât datele de antrenament se supun standardelor aplicate la momentul actual în proiect). Poate fi luată în considerare recunoașterea tuturor obiectelor de un anumit tip dintr-o imagine atât timp cât acestea sunt selecționate corect de pe foaia de test și introduse în celule similare celor descries în soluția actuală. De asemenea, un comportament net superior față de cel actual poate fi identificat în momentul utilizării unor date clare, existent unui tipar bine definit (Ex.: caractere introduse de la calculator).

Bibliografie

1.http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_opencv/py_knn_opencv.html

2. https://mxnet.incubator.apache.org/tutorials/python/mnist.html

3. http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

4. https://docs.opencv.org/3.4.1/d5/d26/tutorial_py_knn_understanding.html

5. https://www.python.org/download/releases/2.7/

6. https://github.com/seatgeek/fuzzywuzzy

7. https://opencv.org/

8. http://www.numpy.org/