Soluție de generare arbori pentru evaluarea de binarizări optime

Alexandru-George ȘERBAN
Facultatea de Automatică și Calculatoare, gr.344C2,
Universitatea "Politehnica" București,
București, România,
alexandru.serban00@stud.acs.upb.ro

Mircea TIMPURIU
Facultatea de Automatică și Calculatoare, gr.342C5,
Universitatea "Politehnica" București,
București, România,
mircea.timpuriu@stud.acs.upb.ro

Abstract— **Binarizarea globală este o metodă de conversie a unei imagini cu tonuri de gri într-una binară cu pixeli albi și negri, prin aplicarea unui singur prag (threshold). La binarizarea locală se utilizează un threshold diferit pentru ferestre determinate din cadrul imaginii. Lucrarea prezintă optimizări pentru ambele tehnici. Pentru binarizarea globală se ilustrează un algoritm de generare a unui arbore care returnează un prag unic. În ambele situații, intrările soluțiilor software sunt fișiere de tip CSV, ce conțin informații de binarizare, referințe ale unui set de imagini (metadate). Pentru metoda locală se prezintă un algoritm pseudo aleatoriu de generare a unui arbore care returnează câte un prag local pentru fiecare porțiune din imagine. Etapele de dezvoltare programatică sunt de antrenare (train), validare (validation) și testare (test). Primele două stagii denotă procesul de construire a arborelui performant, pe când în cadrul celei de test se memorează structura optimă raportată la metrici finale. Totodată, introducerea de noduri rădăcină este etapizată și dependentă aplicării rezultatului de funcții compuse. Pasul anterior este condiționat de atingerea unei mai bune performanțe pentru arbore în cazul adăugării root-ului. Comparația de rezultate nu se rezumă doar la momentul inserării de noduri, ci este efectuată și asupra altor arbori generați aleatoriu, în deosebi în cadrul etapei de validare.**

Keywords— binarizare globală, threshold, binarizare locală, threshold adaptiv, arbori evaluare, validare, testare, metrici de performanță

I. INTRODUCERE

Conversia unei imagini cu tonuri de gri într-o imagine binară cu pixeli albi și negri se numește binarizare. Metoda este utilizată la procesarea imaginilor, în special pentru analiza acestora și identificarea obiectelor conținute.

Cea mai simplă modalitate de binarizare este thresholding. Tehnica thresholding-ului presupune ca primă etapă estimarea unei valori de prag pentru tonurile de gri existente în imagine. În a doua etapă, folosind valoarea de prag găsită, se convertește o imagine alcătuită din pixeli cu tonuri de gri într-o imagine binară, adică una care este formată din pixeli albi și negri. Astfel, tuturor pixelilor pentru care valoarea de gri este mai mare sau egală cu pragul setat li se atribuie culoarea alb (sau valoarea binară 1), iar pentru cei care au valoarea de gri sub prag primesc culoarea negru (binar 0).

Metodele de binarizare pentru imaginile în tonuri de gri pot fi grupate în două mari categorii: binarizare globală, respectiv locală [2], [3].

Metodele globale identifică o singură valoare de prag pentru întreaga imagine. Fiecărui pixel i se asociază o valoare binară (0 sau 1) pe baza nivelului de gri prin comparație cu pragul unic. Metodele globale sunt rapide și dau rezultate bune pentru documentele scanate tipice. Tehnicile cu prag global unic sunt utilizate atunci când există o distribuție distinctă a tonurilor de gri pentru obiecte, respectiv fundal. Timp de mulți ani, binarizarea unei imagini în tonuri de gri s-a bazat pe algoritmi statistici cu prag global [13].

O metodă globală larg utilizată este tehnica Otsu. Se presupun două clase de pixeli (prim-plan și fundal) și se folosește histograma valorilor în tonuri de gri din imagine pentru a alege pragul care maximizează varianța între clase și o minimizează în interior. Metoda Otsu se aplică iterativ. Varianța reprezintă media deviațiilor pătratice pentru o variabilă aleatorie de la media sa. Informal spus, măsoară depărtarea valorilor individuale (aleatorii) dintr-o mulțime, de la media lor [6].

O altă metodă globală folosită este cea propusă de Kittler. Acest algoritm, conceput pentru a diferenția un obiect din fundal în imaginile cu tonuri de gri, presupune că funcțiile de densitate de probabilitate condiționată a claselor obiect și fundal sunt distribuții normale. Histograma este folosită în vederea clasificării erorii pentru un ansamblu de două funcții gaussiene. Algoritmul găsește pragul optim care minimizează probabilitatea erorii de clasificare. Această procedură funcționează considerând ipoteza funcțiilor de densitate gaussiană cu varianță inegală și rezolvă o problemă de ajustare a densității gaussiene cu eroare minimă. Prin urmare, se obține un prag de binarizare care permite împărțirea pixelilor de fundal și de obiecte [7].

Astfel de metode statistice sunt inadecvate pentru imagini complexe și pentru documente degradate. Dacă luminozitatea și contrastul unei imagini nu sunt uniforme, este dificilă identificarea unui prag unic, iar metodele de binarizare globală tind să introducă zgomot. Pentru a depăși aceste probleme, au fost propuse tehnici locale de prag pentru binarizarea documentelor [9], [10], [14], [16].

Metodele de binarizare locale estimează un prag pentru fiecare pixel din imagine pe baza informațiilor conținute în vecinătatea acestuia. Una din procedurile de binarizare locală frecvent utilizată este tehnica Sauvola. Aceasta metodă calculează un prag de binarizare pentru pixelul centrat pe o vecinătate, luând în considerare intervalul dinamic al abaterii standard a valorilor tonurilor de gri pentru pixelii din fereastra locală [5], [17].

În diverse lucrări științifice se arată că fiecărei metode de obținere a pragurilor de binarizare i se pot aduce obiecții. Ca o

consecință, au fost dezvoltate noi metode de binarizare. Unele dintre acestea țin seama, în mod dinamic, de ambele tehnici de obținere a pragurilor, atât metodele globale și cât cele locale [1], [4], [8], [11], [12], [15].

II. Descrierea Soluției

Pentru binarizarea globală, soluția propusă constă într-un algoritm pseudo aleatoriu de generare a unui arbore care să returneze un prag (threshold) global de binarizare, cât mai apropiat de imaginea de ground truth.

Algoritmul generează la început, un număr de arbori inițiali (default 5), iar atât threshold-urile de bază folosite în frunzele arborelui (Otsu, Kittler etc.), cât și un număr de prime noduri pe care se aplică funcții, sunt stabilite aleatoriu (random).

Urmează în etapa de antrenare, ca pentru fiecare dintre arbori, să se adauge un nod nou de tip rădăcină (root) în capăt.

Modalitatea prin care se determină viitorul root este următoarea:

- se aleg aleatoriu un număr de funcții distincte (default 5), exprimând valori posibile ale noului nod, și care au argumente random extrase din cele deja existente în arbore;
- se creează câte o copie a arborelui inițial, la care se adăugă în capăt unul dintre aceste noduri posibile;
- se calculează metrica de performanță pentru fiecare dintre arborii nou creați în starea actuală.

Nodul care ilustrează arborele cu cea mai bună performanță rămâne în structură. În caz contrar, dacă pentru arborele inițial se obțin performanțe mai bune decât oricare inserare a unui root, acesta nu se modifică.

Pentru fiecare arbore generat la început, pentru toți pașii algoritmului, se aplică modalitatea de inserare a unui nod descrisă anterior. După ce în fiecare dintre arbori a fost adăugat (sau nu) un root, se compară metricile de performanță ale acestora. Pentru diminuarea spațiului și timpului de rulare la aproape jumătate, în acest stadiu, arborele cu performanța cea mai slabă va fi eliminat, și nu va mai fi luat în considerare în pașii următori.

Se repetă pasul anterior de adăugare a unui nod pentru toți arborii, eliminându-se cel mai puțin performant dintre ei, până când rămâne unul singur, care este considerat cel optim obținut în etapa de antrenare.

Metrica de performanță este calculată ca fiind media f-measure-urilor pe un set de date, din care se scade de 10 ori timpul de execuție al calculului arborelui. Se iau în vedere atât performanțele din cadrul task-ului, cât și cele de rulare.

În cadrul etapei de validare, se repetă același algoritm ca în stadiul de antrenare, doar că, în loc să se aleagă un număr de arbori inițiali aleatorii, unul dintre aceștia este deja cel optim obținut din etapa de antrenare. Arborele în cauză va fi comparat cu alții care sunt generați aleatoriu, pentru a determina dacă este cel mai performant și după acest stagiu. Dacă însă în această etapă se găsește un arbore mai bun, cel vechi se elimină, și se continuă cu noua configurație.

În stadiul de testare, arborele definitivat după etapele de antrenare și validare se salvează într-un fișier, fiind aplicat pe un set de test pentru a obține metrica finală de performanță, și a compara rezultatele acestuia cu cele aplicate threshold-uri disponibile.

La binarizarea locală, soluția propusă constă într-un algoritm pseudo aleatoriu de generare a unui arbore care sa returneze câte un threshold de binarizare pentru a aproxima cât mai bine valoarea fiecărei porțiuni a unei imagini fata de un ground truth. Se vor genera doi arbori primari, fiecare având pragurile din frunze alese în mod aleatoriu, aplicându-se funcții inițiale pentru combinarea acestora, fiind alese tot random.

În etapa de antrenare, în ambii arbori se adăuga un nod rădăcină (root).

Modalitatea prin care se alege noul nod root al arborelui respectiv determinarea structurii arborescente cu cea mai bună performanță pentru binarizarea locală este asemănătoare cu cea din cadrul tehnicii globale anterior menționate.

Se aplică această metodă asupra ambilor arbori, iar arborele cu criterii potrivite după adăugarea noului rădăcină va avansa în etapa de validare.

Metrica de performanța este calculată ca fiind media f-measure-urilor pe un set de date din care se scade de 1000 de ori timpul de execuție în minute al calculului arborelui. Analog ca la binarizarea globală, se iau în considerare atât performanțele din cadrul task-ului, cât și cele de rulare.

Metrica f-measure în cazul local este calculată după formula $f - measure = TP/(TP + 0.5 * (FP + FN))$, unde:

TP - true positives - pixeli clasificați “albi” care sunt albi în ground truth;

FP - false positives - pixeli clasificați “albi” care sunt negri în ground truth;

TN - true negatives - pixeli clasificați “negri” care sunt negri în ground truth;

FN - false negatives - pixeli clasificați “negri” care sunt albi în ground truth.

Este important de notat etalonul utilizat pentru a determina pixelii albi sau negri, reliefați prin compararea pragului de binarizare cu valoarea de ground truth. În reprezentarea aleasă, 0 semnifică culoarea albă, iar 1 negru. Mai mult, dacă valoarea threshold-ului este peste cea de ground truth, pixelul prezis va fi de tip alb (0), iar daca este mai mic, negru (valoarea 1).

În etapa de validare, arborele avansat de la stadiul de antrenare va fi comparat cu altul nou creat aleatoriu, generat prin aceeași metodă cu a celor doi arbori primari. Se adugă un nou nod rădăcină pentru structurile arborescente provenite de la stagiul de antrenare, alegându-se cea mai bună variantă în funcție de nod, după care cel mai performant dintre cei doi va avansa în etapa de testare.

În cadrul etapei de testare pentru binarizarea locală se efectuează pași identici cu cei ai ultimului stadiu al metodei globale.

III. Arhitectura Soluției

Ierarhia de programe pentru ambele soluții relevă procedurile de generare de noduri rădăcină, în scopul obținerii etapizate a arborelui cu metrici performante. Astfel, se elimină succesiv structurile arborescente, care prin inserarea unui nou

root, nu au potențialul de a translata imaginile mai aproape de ground-truth decât pentru threshold-urile canonice.

Arhitectura este implementată în Python 3 și soluții conexe precum Jupyter notebook, iar ieșirea obținută este formatată pentru a fi comprehensibilă utilizatorilor finali (*end users*).

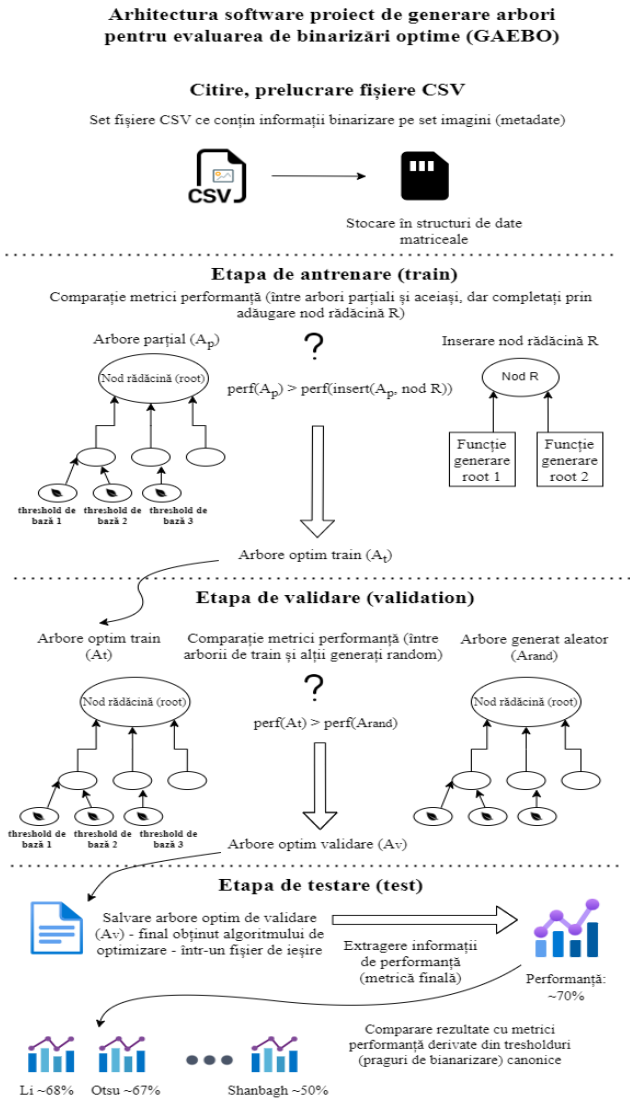

Fig. 1. Arhitectura soluției de binarizare optimală

Datele arborelui performant după stadiile de antrenare și validare sunt salvate în fișiere de output, pentru a extrage informații de performanță și a compara cu efectele aplicării pragurilor clasice pe imagini.

Sursele cod enumerate în ierarhie sunt:
generator.py: fișier care generează programul functions.py, acesta din urmă conținând implementarea unui număr fix (determinist) de funcții care au fost generate aleator.

Fișierul functions.py este creat doar dacă nu există deja. Pentru generarea de funcții, se utilizează 16 proceduri de bază. Următoarele funcții au ca rezultat o combinație de operații între 16 proceduri șablon. Pentru funcțiile compuse, s-a implementat o restricționare cu o gardă (guard), ca o condiție suplimentară, astfel încât rezultatul să rămână în intervalul 0-1.

La soluția de binarizare globală intervin:
functions.py: program ce conține corpurile funcțiilor ce vor fi folosite pentru calcularea nodurilor;
Global_binarisation.ipynb: notebook-ul Jupyter în care este implementat algoritmul principal de obținere a arborilor de threshold-uri. Acesta conține 3 segmente:
1. Segmentul de funcții, în care sunt implementate toate funcțiile necesare rulării, și anume:
- parse_csv, extrage informațiile din fișierele CSV oferite, pe care le stochează în structuri de date matriceale care vor fi prelucrate în continuare;
- choose_random_starting_thresholds, generează un subset aleator de praguri (threshold-uri) frunză pentru un arbore;
- generate_starting_tree_structure, creează un arbore inițial;
- generate_new_node_in_tree, mecanismul de adăugare a unui nou nod de tip rădăcină (root) în arbore;
- solve, aplică funcțiile din arbore pentru a obține un threshold în root, și returnează f-measure-ul threshold-ului obținut pentru o imagine;
- solve_over_dataset și get_final_metrics, aplică funcția solve pe un întreg set de date, returnând metrica de performanță pe întreg setul de date;
- train, execută etapa de antrenare;
- validate, execută etapa de validare;
- write_results_to_file, formatează output-ul pentru a-l face prezentabil;
- get_known_threshold_metric, calculează performanțele threshold-urilor consacrate (Otsu, Kittler, etc.) asupra setului de test.
2. Segmentul de creare a seturilor de date, unde se citesc datele din fișierele CSV și se împarte setul total în mulțime de antrenament (50%), de validare (20%) și de test (30%).
3. Segmentul de determinare a arborilor, în care se aplică etapele de antrenare și validare, scriindu-se cei mai buni arbori în fișiere de output. Se folosește și stadiul de test, unde se compară performanțele arborilor obținuți cu rezultatele specifice celorlalte threshold-uri disponibile.

La algoritmul de binarizare locală intervin:
local_binarisation.ipynb: notebook-ul Jupyter în care este implementat algoritmul local de obținere a arborilor. local_binarisation.ipynb. Fluxul rulării programului arată în felul următor:
- fișierele oferite ca input vor fi parsate în întregime, și, spre deosebire de etapa de binarizare globală, datele vor fi salvate într-o structură de tip dataframe a modulului pandas, prin această opțiune ușurându-se efortul de procesare a unui set de date atât de mare; dataframe va fi împărțit in mulțime de antrenament (60%), de validare (35%) și de test (5%);
- se alege cu ajutorul unei funcții un număr de threshold-uri pentru a fi folosite ca frunze într-un arbore;
- se creează un arbore folosind datele obținute din procedeul anterioar și se aplică funcții matematice asupra nodurilor frunză pentru a crea noduri intermediare;
- se introduc noi noduri rădăcină în arbori și se calculează performanța acestora folosind funcții auxiliare care au scopul de a calcula mai întâi threshold-ul pe o porțiune de imagine. Procedura urmează a fi scalată pe tot setul de date oferit pentru a obține performanța finală;

- toate elementele sunt încorporate în funcția care se ocupă de stagii specifice ale proiectului, fie că este vorba de etapa de antrenare sau de validare, fiind aplicate pentru a obține arborele final, cel mai performant;
- după obținerea soluției finale, arborele va fi scris într-un fișier și vor fi calculate performanțele celorlalte threshold-uri pentru a fi comparate cu soluția implementată.

IV. Rezultate ale Soluției Software

Algoritmii proiectați au fost testați folosind un sistem PC cu procesor Intel Core i5-11400H @ 2.70GHz și 8GB RAM.

La soluția de binarizare globală, pe setul de test partiționat întregii mulțimi de date, algoritmul propus cu valori default reușește să obțină un rezultat f-measure mediu de aproximativ 70%, într-un timp de rulare de aproape 1s. Încă din această etapă, algoritmul este capabil să obțină performanțe mai bune asupra imaginilor oferite, similare de valoarea ground truth, față de aproape toate aplicările celorlalte threshold-uri consacrate pe imaginile din setul de test. Această performanță este deosebită, având în vedere că mulțimea de test este aleasă aleatoriu de fiecare dată. Singurele praguri care se apropie de performanța arborelui sunt threshold-urile Otsu și Li (67-68%), celelalte plasându-se între 50% și 64%. Datele obținute pe un set de test pot fi observate în următorul grafic (*Fig. 2*):

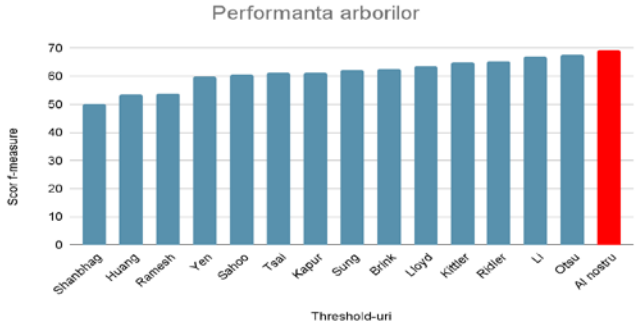

Fig. 2. Comparație performanță arbore obținut prin binarizare globală

Pentru soluția de binarizare locală, pe setul de test care reprezintă 5% din întreaga mulțime de date, algoritmul propus cu valori default reușește să obțină o performanță f-measure medie de aproximativ 78.6%, într-un timp de rulare de aproape 1 minut, pe aceleași specificații de procesor și memorie menționate anterior. Algoritmul propus este capabil să obțină performanțe mai bune asupra imaginilor oferite, cât mai aproape de ground truth, decât o majoritatea de threshold-uri canonice pe imaginile din setul de test. Threshold-urile asemănătoare care reușesc sa depășească la limită performanța arborelui propus sunt Phansalkar si Nick (78.8%), celelalte dintre ele plasându-se între 59% și 78.57%. Datele obținute pe setul de test pot fi observate în următorul grafic (*Fig. 3*):

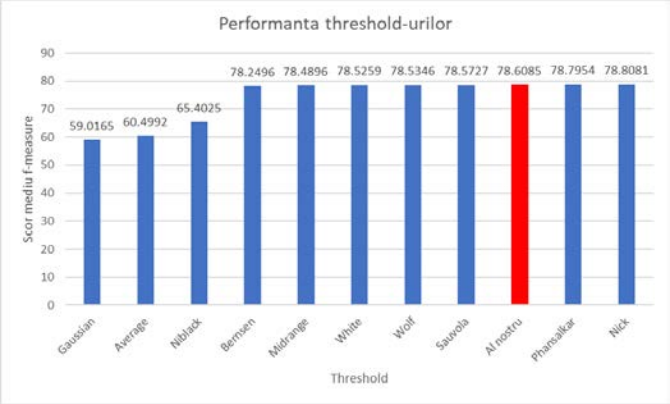

Fig. 3. Comparație performanță threshold obținut prin binarizare locală

V. Concluzii

O problemă întâlnită în dezvoltare și în special, testarea algoritmului, a constat în dimensiunea mare a setului de date oferit spre analiză. Astfel, în cazul binarizării globale, durata de parsare și extragere a datelor din fișiere poate atinge o oră în unele cazuri. Deși folosind pandas dataframes s-a reușit diminuarea semnificativă a timpului de parsare, acest pas durând acum maxim 1-2 minute, creșterea semnificativă a cantității de date duce la probleme temporale în cadrul calculării și testării algoritmului în sine, antrenarea si validarea putând dura între 1-2 ore pe un sistem în configurația menționată anterior. Însă, în aceste condiții, un timp de rulare mare al algoritmului de găsire a celui mai potrivit arbore pentru binarizarea globală, este compensat printr-o performanță mai bună decât restul tuturor celorlalte threshold-uri.

S-a observat faptul că pentru un arbore care nu satisface cerințele într-un pas al generării, probabilitatea de îmbunătățire după adăugarea de noduri tip rădăcină (root) la capătul acestuia este redusă. Astfel, s-a avut în vedere în implementare, eliminarea celui mai puțin performant arbore la fiecare pas. Prin urmare, în cazul binarizării globale, s-a redus la jumătate numărul de operații de căutare a celui mai potrivit nod de inserat.

Pe baza analizelor de la binarizarea globală, în care s-a observat că în cele mai multe cazuri, un arbore care obține performanțe mai slabe decât restul într-un anumit punct al execuției are șanse slabe să revină și să obțină performanțe mai bune decât ceilalți după adăugarea anumitor noduri root în capăt, s-a luat decizia că cea mai bună cale de acțiune este de a lăsa doar 2 arbori să aleagă din câte 2 funcții disponibile în fiecare etapă a algoritmului.

Bibliografie

[1] Xiaoyi Jiang, "Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, nr. 1, pp. 131–137, Jan. 2003.

[2] R. C. Gonzalez and R. E.Woods, *Digital Image Processing*, 4th ed., New York, NY: Pearson, 2018.

[3] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–165, Jan. 2004.

[4] S. Venkatesh and Paul Rosin, "Dynamic threshold determination by local and global edge evaluation," *Graphical models and image processing*, vol. 57, nr. 2, pp. 146-169, Mar. 1995.
[5] J. Sauviola, T. Seppanen, S. Haapakoski and M. Pietikainen, "Adaptive document binarization," *Pattern recognition*, Vol. 33, issue 2, pp.225-236 Feb. 2000.
[6] Nobuyuki Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on sistems, man, and cybernetics*, Vol. SMC-9, nr. 1, Jan. 1970.
[7] Er.Nirpjeet kaur, Er Rajpreet kaur, "A review on various methods of image thresholding", *International journal on computer science and engineering*, Vol. 3, nr. 10, pp. 3443-3443, Oct. 2011.
[8] W. A. Mustafa and M.M. Abdul Kader, "Binarization of document image using optimum threshold modification," *Journal of Physics: Conf. Series*, nr. 1019, 2018.
[9] P. Stathis, E. Kavallieratou and N. Papamarkos, "An evaluation technique for binarization algorithms", *Journal of universal computer science*, vol. 14, nr. 18, pp. 3011-3030, Oct. 1, 2018.
[10] T. Kalaiselvi, P. Nagaraja and V. Indhu, „A comparative study on thresholding techniques for gray image binarization," *International Journal of advanced research in computer science*, Vol. 8, nr. 7, pp. 1168-1172, Jul.-Aug. 2017.
[11] T.R. Singh, S. Roy, O.I. Singh, T. Sinam, K.M. Singh, "A new local adaptive threshoulding technique in binarization," *International Journal of Computer Science Issues*, Vol. 8 issue 6, nr. 2, pp. 271-277, Nov. 2011.
[12] N. Senthilkumaran and S. Vaithegi, "Image segmentation by using thresholding techniques for medical images," *Computer Science & Engineering: An international juournal*, Vol. 6, nr. 1, pp. 1-13, Feb. 2016.
[13] M. Athimethphat, "A review on global binarization algorithms for degraded document images," *AU Journal of technology*, Vol. 14, nr. 3, pp. 188-195, Jan. 2011.
[14] P.K. Sahoo, S. Soltani, A.K.C. Wong and, Y.C. Chen, "A survey of thresholding techniques," *Computer vision, graphics, and image processing*, Vol. 41, nr. 2, pp. 233-266, Feb. 1988.
[15] E. Zemouri, Y. Chibani, and Y. Brik, "Enhancement of Historical Document Images by combining global and local binarization technique," *International journal of information and electronics engineering*, Vol. 4, nr. 1, pp. 1-5, Jan. 2014.
[16] O.D. Trier and T. Taxt, "Evaluation of binarization methods for document images," *IEEE Transactions on Pattern Analysis and Machine Inteligence*, Vol. 17, Nr. 3, pp. 312-315, Mar. 1995.
[17] G. Lazzara and T. Geraud, "Efficient Multiscale Sauvola's Binarization," *International Journal on Document Analysis and Recognition*, Vol. 17, Nr. 2, pp. 105-123, Jun. 2014.