STUDIA UNIVERSITATIS MOLDAVIAE, 2018, nr.7(117)
Seria "Științe exacte și economice" ISSN 1857-2073 ISSN online 2345-1033 p.25-32

CZU: 004.81:159.953.5:[519.68:61]

PROCESAREA IMAGINILOR DIGITALE CU IMPLEMENTAREA ALGORITMILOR ÎNVĂŢĂRII NESUPERVIZATE

Maria CRISTEI

Universitatea de Stat din Moldova

Lucrarea de faţă prezintă elaborarea unei aplicaţii informatice de procesare a imaginilor bazată pe algoritmi de analiză, de segmentare în baza modelului de culori, cu implementarea metodei de învăţare automată nesupervizată *K-means* şi a algoritmului de detectare a contururilor *Canny*, în vederea îmbunătăţirii calităţii imaginilor de înaltă rezoluţie şi extragerii datelor relevante pentru utilizarea ulterioară a acestora în analiza automată sau pentru interpretarea prin afişare. Utilizarea aplicaţiei elaborate la prelucrarea şi analiza imaginilor biomedicale (tomografii computerizate, radiografii, rezonanţă magnetică, ultrasunete şi altele) face posibilă îmbunătăţirea contrastului, codarea intensităţii (nivelurilor de gri) imaginilor monocrome în culori, detectarea contururilor şi recunoaşterea formelor specifice, contribuind astfel la determinarea mai uşoară a unor anomalii. De asemenea, acestea pot fi utilizate pentru monitorizarea pacienţilor şi pentru descoperirea/ identificarea de boli şi tumori, îmbunătăţind astfel actul medical.

Cuvinte-cheie: *segmentare, învăţare automată, învăţare nesupervizată, algoritm, model.*

DIGITAL IMAGE PROCESSING BY IMPLEMENTING UNSUPERVISED LEARNING ALGORITHMS

This paper presents the development of an image processing application using color modeling algorithms with the implementation of the unattended *K-means* automated learning method and the *Canny* contour detection algorithm for the improvement of high-resolution images and extracting data relevant to their subsequent analysis or display interpretation. The use of the application developed in biomedical imaging processing and analysis (computational tomography, radiography, magnetic resonance, ultrasound, etc.) provides contrast enhancement, encoding the intensity (in gray scale) of monochrome images in color, contours determination and recognition of specific forms, contributing to the easier determination of anomalies. They can also be used to monitor patients and discover / identify diseases and tumors, thus improving medical performance.

Keywords: *segmentation, machine learning, unsupervised learning, algorithm, color model*.

Consideraţii generale

În prezent, procesarea imaginilor constituie un domeniu de cercetare major ce oferă soluţii la îmbunătăţirea calităţii imaginilor efectuate cu ajutorul tehnologiilor digitale de ultimă generaţie. La ora actuală, practic în orice domeniu de activitate se pot găsi numeroase aplicaţii, care necesită utilizarea tehnicilor de prelucrare şi analiză a imaginilor, susceptibile să îmbunătăţească informaţia vizuală. Prelucrarea digitală a imaginilor presupune o succesivitate de etape de procesare hardware şi software, precum şi de implementare a unor metode teoretice. Etapele fundamentale necesare pentru realizarea unor prelucrări de imagini sunt ilustrate în Figura 1.

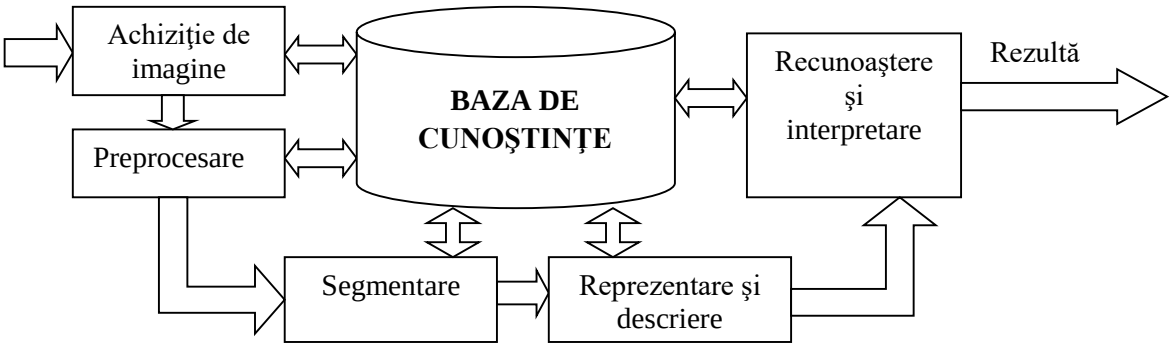

Fig.1. Etape fundamentale în prelucrarea digitală a imaginilor.

Achiziţia de imagine este prima etapă în procesul de prelucrare digitală. Achiziţia constă în digitalizarea/ capturarea imaginilor prin intermediul diferitor echipamente hardware, fapt ce denotă dependenţa dimensiunii

acestora de produsele hardware utilizate. Astfel, de exemplu, în cazul capturării imaginilor de la aparatura medicală se obţine, de regulă, mari dimensiuni ale acestora, fiind necesară segmentarea şi reprezentarea 3D a acestor imagini.

Segmentarea imaginilor

Segmentarea imaginilor digitale constituie o etapă principală în majoritatea aplicaţiilor de procesare a imaginilor. Segmentarea este procesul de descompunere a unei imagini (sau scene) în elementele sale constituente sau în obiecte distincte. În general, segmentarea este una dintre cele mai dificile etape în prelucrarea digitală de imagini. Pe de o parte, *algoritmii sofisticaţi de segmentare, cu şanse mari de selectare a regiunilor de interes, necesită calcule complicate* şi, deci*, un timp îndelungat*. Pe de altă parte, însă, *algoritmii simpli sau insuficient elaboraţi nu pot fi aplicaţi cu şanse mari de succes*. Adesea, segmentarea este strâns legată de algoritmii de analiză, al căror scop este *de a realiza măsurători cantitative sau evaluări calitative asupra unor anumite categorii de obiecte prezente în imaginea dată*. Prelucrările de imagini care includ *recunoaştere* şi *interpretare* (spre exemplu, interpretarea radiografiilor, tomografiilor sau a altor tipuri de imagini biomedicale, prelucrarea automată a amprentelor, recunoaşterea caracterelor, viziunea artificială industrială folosită pentru asamblarea şi inspecţia vizuală a produselor) *sunt asociate cu aplicaţii de analiză a imaginilor* şi *au ca obiectiv extragerea automată sau semiautomată de informaţii din imagini*. Cu alte cuvinte, aplicaţiile de analiză a imaginilor au ca scop final extragerea unor caracteristici importante din imagine, care să ajute calculatorul să înţeleagă, să descrie sau să interpreteze o scenă. În acest caz, rezultatele analizei prezintă asemănări foarte reduse cu caracteristicile obişnuite ale imaginilor, indispensabile ochiului uman pentru interpretare sau înţelegere. Extragerea caracteristicilor imaginii presupune separarea caracteristicilor spaţiale şi de tip transformare, de formă şi de textură, a muchiilor şi a contururilor, precum şi a caracteristicilor statice ale imaginii. Este de menţionat că, după caracteristicile de extras, tehnicile de segmentare se clasifică în două categorii fundamentale: *tehnici de segmentare orientate pe regiuni* şi *tehnici de segmentare orientate pe contur*.

În cadrul segmentării orientate pe regiuni se disting câteva categorii principale de tehnici: *etichetarea imaginilor binare; segmentarea pe histogramă; creşterea şi fuziunea regiunilor; segmentarea texturilor; segmentarea prin metode de clustering*. Tehnicile principale de segmentare orientată pe contururi sunt: *extragerea contururilor prin metode de gradient şi derivative; extragerea contururilor prin metode neliniare; extragerea contururilor prin metode liniare optimale; extragerea contururilor prin modelare matematică*.

De fapt, prelucrarea imaginilor digitale presupune utilizarea unor tehnici specifice exprimate, de obicei, sub forma unor algoritmi. Însă, pentru o prelucrare pe calculator este necesară conversia datelor într-o formă potrivită. Alegerea modalităţii de reprezentare a datelor este urmată de specificarea metodei de descriere a acestora, astfel încât caracteristicile de interes să fie scoase în evidenţă cu prioritate. În prezent există un număr semnificativ de programe de prelucrare; cu toate acestea, metodele care duc la rezultate foarte bune în unele aplicaţii sunt total nepotrivite în altele. Ceea ce pun, de fapt, la ora actuală la dispoziţie *metodele şi tehnicile* din domeniul inteligenţei artificiale *este un punct de pornire pentru dezvoltarea unor aplicaţii inovative, care necesită încă o muncă de cercetare şi dezvoltare foarte serioasă*.

Prelucrarea imaginilor şi inteligenţa artificială

Prelucrarea imaginilor şi inteligenţa artificială sunt domenii ce se pătrund reciproc. Un număr important de algoritmi performanţi folosiţi la prelucrarea imaginilor utilizează metode şi tehnici din domeniul inteligenţei artificiale, cum ar fi: reţele neuronale, arbori de decizie, învăţare automată, vectori suport maşină, clasificatori parametrici. Pe de altă parte, inteligenţa artificială presupune zona de intersectare şi construirea de sisteme care pot să realizeze funcţii ale intelectului uman: învăţarea prin experienţă, înţelegerea limbajului natural, utilizarea unui raţionament pentru rezolvarea unor probleme sau luarea unor decizii. Toate acestea presupun însăşi acumularea unei anume cantităţi de informaţie (bază de cunoştinţe, informaţii din mediu etc.) [1]. Această informaţie este preluată de sistemele inteligente şi creează o imagine a mediului în momentul preluării datelor. Din imaginea astfel obţinută trebuie extrase informaţiile utile.

Metodele şi tehnicile de achiziţionare a cunoştinţelor sunt *manuale* (analistul de cunoştinţe) şi *automate* (machine learning). Domeniul învăţării automate formalizează aceste metode şi caută aplicarea lor în sistemele automate. Sistemele de învăţare automată reprezintă varianta de inteligenţă artificială aplicabilă în prezent: în baza unor algoritmi matematici, ce pot să opereze cu un volum foarte mare de date, să înveţe singure date, reguli, algoritmi şi să-şi perfecţioneze acţiunile. Deci, învăţarea automată presupune în primul rând

identificarea şi implementarea unei modalităţi cât mai eficiente de reprezentare a informaţiilor, în sensul facilitării căutării, reorganizării şi modificării acestora [2].

Învăţarea nesupervizată reprezintă o categorie din machine learning, care are ca scop principal descoperirea structurii ascunse într-un set de date fără parametri etichetaţi, însumarea şi gruparea optimă a datelor găsite [3]. În general, învăţarea nesupervizată presupune existenţa unor instanţe neclasificate, un set de reguli euristice pentru crearea de noi instanţe şi evaluarea unor concepte deduse, eventual un model general al spaţiului de cunoştinţe în care se găsesc aceste instanţe. Un algoritm de învăţare nesupervizată construieşte concepte pentru a clasifica instanţele, le evaluează şi le dezvoltă pe cele considerate „interesante" de regulile euristice. În general, concepte interesante sunt considerate cele care acoperă majoritatea din instanţe, dar nu în totalitate. Unii algoritmi de învăţare nesupervizată includ gruparea *K-means*, analiza principală şi independentă a componentelor şi regulile de asociere [4]. Deşi învăţarea nesupervizată este mai dificilă la aplicarea în cazuri simple, ea oferă soluţii pentru rezolvarea problemelor pe care oamenii nu le-ar rezolva în mod normal.

Implementarea algoritmului de învăţare nesupervizată *K-means clustering*

O tehnică de învăţare nesupervizată este *clustering-ul*, care determină o structură intrinsecă într-o mulţime de date. Clustering-ul reprezintă clasificarea datelor existente neetichetate în grupuri distincte, astfel încât membrii aceluiaşi grup sunt asemănători unul cu celălalt şi diferiţi de membrii altor grupuri. Prin clustering se încearcă obţinerea de grupări care sunt: (a) *semnificative*: clusterele trebuie să surprindă natura structurală a datelor, (b) *utile*: sumarizarea unui volum mare de date, furnizarea de explicaţii sau ambele. Una dintre provocările principale la utilizarea clustering-ului îl constituie procesul de alegere a numărului de clustere care necesită a fi create, iar identificarea unui număr optimal este foarte dificilă.

Algoritmul *K-means clustering* este o metodă de determinare a clusterelor pe care le formează mai multe pattern-uri. Algoritmul *K-means* clasifică o mulţime de N date (puncte), descrise de vectori de dimensiune *p*, într-un număr *K* stabilit *a priori* de clustere (grupuri, clase). Fiecare cluster are un centroid. Fiecare dată (punct) este asociată clusterului determinat cu cel mai apropiat centroid, acesta servind ca prototip al grupului. Distanţa dintre punct şi centroid poate fi calculată ca distanţă euclidiană, dar se poate opta şi pentru alte variante. Algoritmul are ca rezultat gruparea mulţimii de date în celulele *Voronoi*. *K-means* este un algoritm iterativ şi este următorul:

1. Se aleg aleatoriu *K* puncte în spaţiul definit de obiectele din baza de date ce urmează a fi clusterizate. Valoarea *K* poate fi aleasă şi prin acordarea unor puncte iniţiale de către utilizator. Acestea sunt *centroizii* iniţiali;
2. Se asignează fiecare obiect din mulţimea de date celui mai apropiat centroid (în sensul distanţei considerate), obţinând astfel un prim grupaj. Fiecare colecţie de puncte asignată unui centroid este un cluster;
3. Se calculează centrele de greutate determinate de punctele clusterelor obţinute, definind astfel noii centroizi;
4. Se repetă paşii 2 - 3 până când nu există nicio schimbare în clustere (sau, posibil, până când nu este îndeplinită o altă condiţie de oprire).

Diagrama fluxului iterativ al algoritmului *K-means clustering* este prezentată în Figura 2.

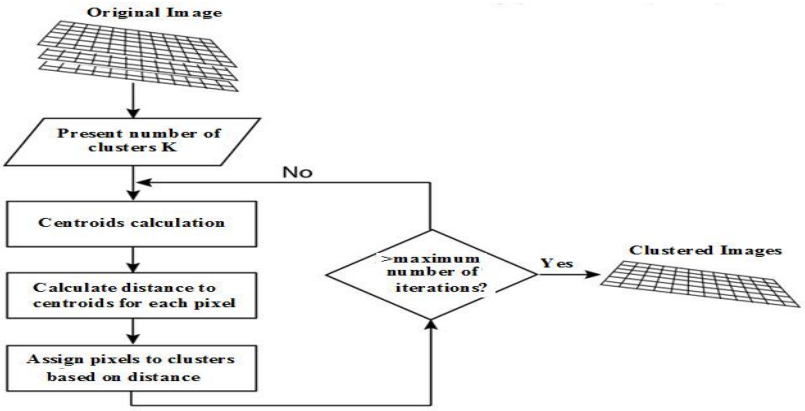

Fig.2. Diagrama fluxului algoritmului *K-means clustering*.

În utilizarea metodei *K-means* există o problemă majoră, şi anume – necesitatea definirii mediei, ceea ce implică faptul că este inaplicabilă la date nenumerice.

Alegerea optimă a valorii *K*

Algoritmul urmăreşte minimizarea *funcţiei obiectiv*, definită de:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} \left\| x_i^{(j)} - c_j \right\|^2,$$

unde $\left\| x_i^{(j)} - c_j \right\|^2$ este distanţa aleasă între punctul $x_i^{(j)}$ (vectorul x_i din *clusterul j*) din baza de date şi centroidul c_j al clusterului *j*.

Funcţia obiectiv, care în acest caz reprezintă funcţia erorii pătratice, este un indicator al distanţei dintre cele N puncte din baza de date şi centrele clusterelor corespunzătoare.

Deoarece algoritmul nu găseşte întotdeauna configuraţia optimă, corespunzătoare minimului global al funcţiei obiectiv şi este sensibil la selectarea iniţială a centroizilor, se rulează de mai multe ori algoritmul, pentru reducerea acestor efecte. Rezultatele pot fi comparate prin examinarea clusterelor sau printr-o măsură numerică, cum ar fi distorsiunea clusterelor, care este suma diferenţelor la pătrat dintre fiecare dată şi centroidul său corespunzător. În cazul distorsionării clusterului, gruparea cu cea mai mică valoare de distorsiune poate fi aleasă ca cea mai bună.

Pentru alegerea unei valori adecvate *K*, executăm experimentul pentru diferite valori ale acestuia şi evidenţiem acele valori ce produc rezultate bune. În acest sens, rezultatele clustering-ului trebuie examinate pentru a determina clusterele care au sens. Valoarea *K* poate fi micşorată în cazul în care unele clustere conţin prea puţine date sau când numărul clusterelor este prea mare [5].

Din această perspectivă, *ne-am propus* (autoarea împreună cu Olga Soltan, studentă la anul III, Facultatea de Matematică şi Informatică, USM), *prin aplicarea tehnicilor de învăţare automată, a metodelor de detectare a contururilor, formelor şi a spectrului de culori, elaborarea unei aplicaţii informatice de procesare, care are ca scop îmbunătăţirea calităţii imaginilor de înaltă rezoluţie şi extragerea datelor relevante pentru utilizarea ulterioară a acestora în vederea analizei automate sau interpretării prin afişare.* Implementarea aplicaţiei elaborate la prelucrarea şi analiza imaginilor biomedicale (tomografii computerizate, radiografii, imagistica prin rezonanţă magnetică şi altele), obţinute în urma monitorizării pacienţilor, contribuie la identificarea cu uşurinţă a unor anomalii. Astfel, facilitează depistarea timpurie de tumori şi boli şi, prin urmare, prevenirea dezvoltării sau tratarea corespunzătoare a acestora.

Selectarea tehnologiilor informatice pentru realizarea aplicaţiei

Metodologia folosită în realizarea informatică a aplicaţiei a fost mixtă, bazându-se pe tehnologii de ultimă oră. Decizia noastră a avut ca prim argument utilizarea pe scară largă a instrumentelor de dezvoltare a aplicaţiilor cu tehnologiile machine learning şi analiza datelor, fără necesitate de mari resurse. În acest sens, astăzi cele mai populare tehnologii sunt: mediul de dezvoltare *Microsoft Visual StudioCod,* limbajul de programare *Python,* pentru procesarea imaginilor – biblioteca de algoritmi machine learning şi funcţii de programare pentru computer vision *OpenCV* şi biblioteca *Scikit-image*. De asemenea, menţionăm că *OpenCV* este o bibliotecă dezvoltată în limbajul C++ şi că *Python* oferă posibilitatea utilizării acesteia datorită funcţionalităţii de extindere a codului cu C/C++. Acest lucru oferă, în primul rând, codului rapiditate la fel ca şi codul original C/ C++; în al doilea rând, este foarte uşor de programat în *Python*. Astfel, modulul *OpenCV-Python* funcţionează, reprezentând un modul *Python* în jurul implementării C++ original [6]. Aplicaţia grafică a fost elaborată prin intermediul instrumentelor de dezvoltare a interfeţei grafice *toolkit-ul cross-platformă* pentru *Python PyQt,* a mediului de dezvoltare grafică a interfeţei *QtDesigner*, iar partea logică a fost dezvoltată în totalitate în limbajul *Python* cu utilizarea funcţionalităţilor oferite de bibliotecile *OpenCV* şi *Sklearn*.

Rezultate şi discuţii

Aplicaţia informatică elaborată are o interfaţă grafică prietenoasă pentru gestionarea procedurilor. În pagina de start a aplicaţiei (Fig.3) suntem invitaţi să selectăm, prin apăsarea consecutivă a butonului „*Choose image"*, şi să importăm imaginile pentru care se va efectua procesarea.

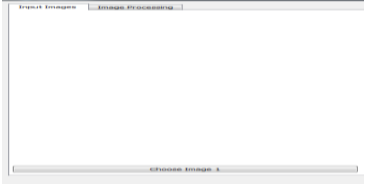

Fig.3. Importarea imaginii.

Codul sursă pentru inserarea imaginilor este prezentat în următorul tabel

Tabelul 1

Codurile funcţiilor de importare a imaginilor supuse procesării

Codul funcţiei de importare a primei imagini	`def choose_image(self):`
	`  self.image1FileName = QtWidgets.QFileDialog.getOpenFileName(None,'OpenFile','D:\\',"Image file(*.png *.jpg *.jpeg)")`
	`  self.image1FilePath = str(self.image1FileName[0])`
	`  self.image1 = QtGui.Qimage(self.image1FilePath)`
	`  self.finalImage1 = QtGui.Qpixmap().fromImage(self.image1)`
	`  labelWidth = self.input_image1.width()`
	`  labelHeight = self.input_image1.height()`
	`self.input_image1.setPixmap(self.finalImage1.scaled(labelWidth, labelHeight,QtCore.Qt.KeepAspectRatio))`

Codul funcţiei de selectare a imaginii a doua	`def choose_second_image(self):`
	`  self.image2FileName = QtWidgets.QFileDialog.getOpenFileName(None,'OpenFile','D:\\',"Image file(*.png *.jpg)")`
	`  self.image2FilePath = str(self.image2FileName[0])`
	`  self.image2 = QtGui.QImage(self.image2FilePath)`
	`  self.finalImage2 = QtGui.QPixmap().fromImage(self.image2)`
	`  labelWidth = self.input_image2.width()`
	`  labelHeight = self.input_image2.height()`
	`self.input_image2.setPixmap(self.finalImage2.scaled(labelWidth, labelHeight,QtCore.Qt.KeepAspectRatio))`
	`self.image_2.setPixmap(self.finalImage2.scaled(labelWidth,labelHeight,QtCore.Qt.KeepAspectRatio))`

După identificarea imaginilor necesare, acestea sunt ajustate la dimensiunile ferestrei, păstrând astfel dimensiunea şi calitatea după cum este reprezentat în Figura 4:

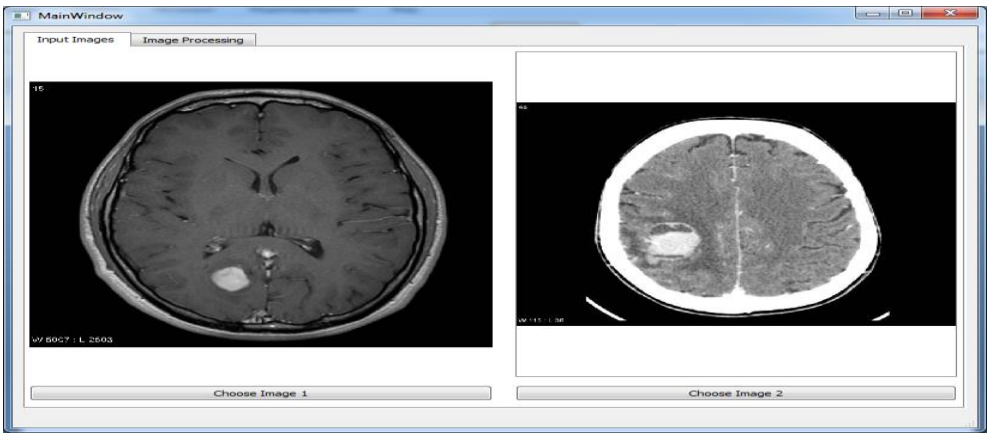

Fig.4. Fereastra de import a două imagini.

Importând imaginile, acestea sunt automat adăugate în a doua fereastră "*Image Processing*" a aplicației (Fig.5), prin apelarea funcției ***setPixmap()*** (Fig.6), care ia ca parametri dimensiunea imaginii și ajustează widget-urile.

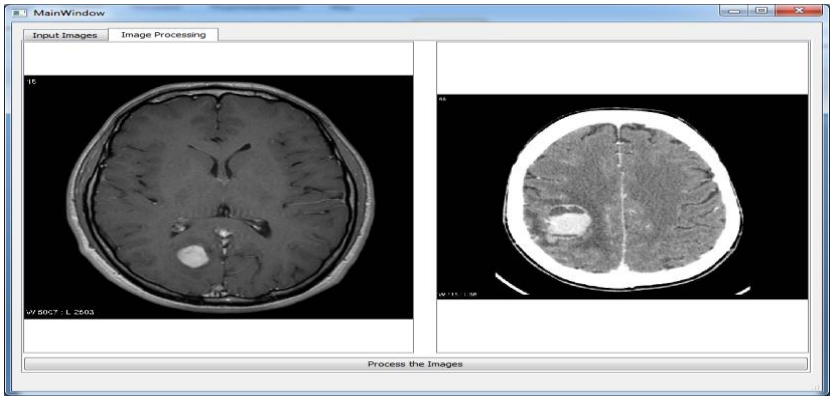

Fig.5. Interfața ferestrei *Image Processing*.

```
self.image_1.setPixmap(self.finalImage1.scaled(labelWidth,labelHeight,
                            QtCore.Qt.KeepAspectRatio))
self.image_2.setPixmap(self.finalImage2.scaled(labelWidth,labelHeight,
                            QtCore.Qt.KeepAspectRatio))
```

Fig.6. Codul pentru importarea automată a imaginilor în a doua fereastră.

Funcția bibliotecii OpenCV ***imread()*** (Fig.7) primește ca parametru calea spre imagine și o citește ca un tablou multidimensional cu spațiul de culori în formatul BGR. Pentru a modifica spațiul de culori, se utilizează funcția ***cvtColor()***, convertind imaginea în spațiul HSV (Hue, Saturation, Value).

```
self.firstGray = cv2.imread(self.image1FilePath)
        width = self.input_image1.width()
        if width > 0:
            height = int((width / self.firstGray.shape[1]) *
                         self.firstGray.shape[0])
            self.firstGray = cv2.resize(self.firstGray, (width, height),
                interpolation=cv2.INTER_AREA)
self.firstGray = cv2.cvtColor(self.firstGray, cv2.COLOR_BGR2HSV)
```

Fig.7. Codul citirii imaginii și conversiei în spațiul HSV.

După conversia în spațiul de culori dorit, izolăm doar coordonatele culorilor care vor fi preluate ca argumente la implementarea algoritmului. Pentru izolarea coordonatei *Hue*, variabila ***channelsRaw*** (Fig.8) va fi egală cu 0, pentru *Saturation* se va alege 1, iar pentru *Value* se va alege 2. Dacă este selectată doar o singură coordonată, tabloul devine bidimensional. Pentru a restabili dimensiunea tridimensională, unde primul indice reprezintă rândul, al doilea reprezintă coloana, iar al treilea coordonata spațiului de culori, se utilizează funcția ***reshape()*** (Fig.8), care adaugă un al treilea index fals, astfel restabilind dimensiunea tabloului.

```
channelsRaw = '02'
        channels = cv2.split(self.firstGray)
        channelIndices = []
        for char in channelsRaw:
            channelIndices.append(int(char))
        self.firstGray = self.firstGray[:,:,channelIndices]
        if len(self.firstGray.shape) == 2:
            self.firstGray.reshape(self.firstGray.shape[0],
self.firstGray.shape[1], 1)
```

Fig.8. Codul pentru izolarea coordonatelor culorilor.

Pentru implementarea algoritmului, tabloul anterior se converteşte într-un vector de carcteristici *MxN*, unde *M* reprezintă numărul de pixeli, iar *N* – numărul de coordonate din spaţiul de culori.

Algoritmul are ca parametri de intrare: *numărul de clustere în care vor fi grupaţi pixelii* (**numClusters**), *numărul de rulări ale algoritmului cu diferiţi centroizi* (**n_init)**, *numărul de reînnoiri ale centroidului pentru fiecare cluster* (**max_iter**). După execuţie, se efectuează redimensionarea tabloului în formatul imaginii originale. În final, se iniţializează imaginea cu clusterele formate în urma execuţiei algoritmului, după cum este reprezentat în Tabelul 2.

Tabelul 2

Coduri elaborate

Crearea vectorului de caracteristici	`reshaped = self.firstGray.reshape(self.firstGray.shape[0] * self.firstGray.shape[1], self.firstGray.shape[2])`
Codul pentru executarea algoritmului *K-means*	`numClusters = 4` `kmeans = KMeans(n_clusters=numClusters, n_init=40, max_iter=500).fit(reshaped)`
Codul pentru restabilirea dimensiunii imaginii	`clustering = np.reshape(np.array(kmeans.labels_, dtype=np.uint8), (self.firstGray.shape[0], self.firstGray.shape[1]))`

Codul pentru crearea imaginii clusterizate	`kmeansImage =np.zeros(self.firstGray.shape[:2],dtype=np.uint8)`
	`for i, label in enumerate(sortedLabels):`
	`kmeansImage[clustering == label] = int(255 / (numClusters - 1)) * i`

În urma procesării imaginii cu ajutorul algoritmului *K-means* cu spaţiul de culori HSV, se observă îmbunătăţirea contururilor şi contrastelor, ceea ce permite analiza şi identificarea uşoară a anomaliilor. Imaginea segmentată poate fi observată în Figura 9. În urma execuţiei algoritmului K-*means*, imaginea este segmentată în 3 culori distincte, uşurând procesul de citire a imaginilor biomedicale. Mai mult ca atât, aplicaţia elaborată permite procesarea imaginilor cu conversia acestora în alte spaţii de culori (CIE(International Comission of Illumination) Lab, YCbCr).

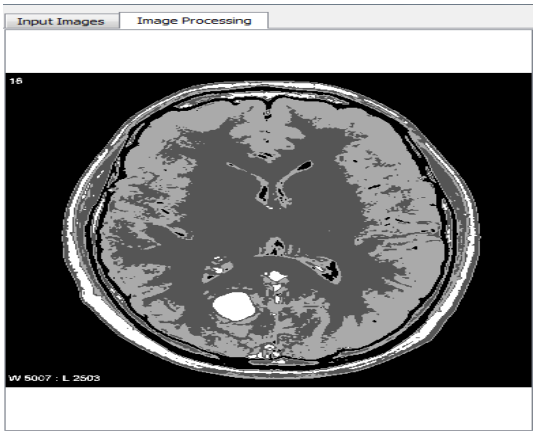

Fig.9. Rezultatul final.

Eficientizarea procesării imaginii prin detectarea contururilor

Pentru a creşte eficienţa algoritmului învăţării automate, s-a implementat detectorul de contur *Canny*, care constă în:

- reducerea zgomotului de pe imagine cu ajutorul filtrului Gaussian;
- găsirea intensităţii gradientelor pe imagine;
- suprimarea non-maximului pentru a exclude pixelii care nu pot crea contururi;
- aplicarea unui interval dublu pentru determinarea potenţialelor contururi.

În Figura 10 este prezentat codul sursă pentru implementarea algoritmului *Canny*.

```
self.firstEdged = cv2.Canny(kmeansImage, 5, 250)
          self.firstKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
          self.firstClosed   =   cv2.morphologyEx(self.firstEdged,   cv2.MORPH_CLOSE,
self.firstKernel)
         self.firstEdged = Image.fromarray(self.firstClosed)
```

Fig.10. Codul sursă pentru implementarea algoritmului Canny.

Concluzii

În concluzie, putem afirma că prelucrarea şi analiza imaginilor reuşeşte să câştige teren ca urmare a abordărilor moderne de dezvoltare software, a evoluţiei spectaculoase a instrumentelor inovative pentru suportul dezvoltării programelor, precum şi a particularităţilor unor medii de dezvoltare de ultimă generaţie. De asemenea, datorită succeselor vizibile înregistrate în domeniul inteligenţei artificiale, metodele învăţării automate devin tot mai utilizate, oferind soluţii multor probleme din diverse domenii. Algoritmi sofisticaţi ai metodelor de învăţare nesupervizată sunt cu succes implementaţi în domeniul procesării grafice. Aceştia sunt folosiţi la clasificarea imaginilor după anumite criterii, la identificarea şabloanelor, la monitorizarea unor obiecte/procese în timp real.

Principala idee a lucrării este prezentarea cercetărilor pentru elaborarea unei aplicaţii informatice pentru procesarea imaginilor biomedicale, utilizând algoritmii şi tehnicile inteligenţei artificiale, care să contribuie la diagnosticarea timpurie a bolilor pacienţilor cu simptome vagi sau greu de observat. Prin urmare, este important să elaborăm şi să furnizăm medicilor instrumente cât mai inovative pentru procesarea imaginilor biomedicale, şi nu numai.

Aplicaţia elaborată este complet funcţională. Procesul de elaborare a aplicaţiei informatice include utilizarea de noi instrumente de dezvoltare. Funcţionalităţile permit inserarea imaginilor şi procesarea acestora cu ajutorul algoritmului învăţării automate *K-means*, a algoritmului de detectare a contururilor *Canny*. Rezultatele obţinute pot fi utilizate la îmbunătăţirea calităţii imaginii, la detectarea şi clasificarea obiectelor identificate pe imagini. De asemenea, este prezentat rezultatul procesării imaginilor la modificarea modelului de culori din *RGB* în *HSV*, *LAB* şi *YcbCr*.

În consens cu rezultatele obţinute, ne propunem, pe viitor, extinderea funcţionalităţilor aplicaţiei, prin integrarea de reţele neuronale artificiale, arbori de decizie şi alte concepte ale inteligenţei artificiale pentru a găsi similitudinile dintre mai multe imagini şi crearea unui model susceptibil să genereze imagini noi în baza celor procesate anterior. Rezultate ce vor permite să se prezică/simuleze dezvoltarea tumorilor sau a maladiilor, precum şi vizualizarea grafică a datelor medicale.

Referinţe:

1. NICOLAU, D., BARBU, D., CIOCOIU, L., SMADA, D. Tehnologii avansate pentru procesarea imaginilor biomedicale, utilizând algoritmi de recunoaştere a formelor. Studiu de caz: afecţiuni ale ficatului. În: *Revista Română de Informatică şi Automatică, vol. 21, nr. 2*. Bucureşti: Institutul Naţional de Cercetare Dezvoltare în Informatică - ICI, 2011, p.57-70. Disponibil: https://rria.ici.ro/wp-content/uploads/2018/02/08-art-Laura.pdf
2. CRISTEI, M., MARIN, Gh., STELEA, V. Prezicerea performanţelor studenţilor folosind învăţarea automată (Machine Learning). În: *Studia Universitatis Moldaviae. Seria „Ştiinţe exacte şi economice"*. Chişinău: CEP USM, 2017, p.43-49.
3. *Machine Learning for humans.* [Accesat: 15.12.2017]. Disponibil: https://medium.com/machine-learning-for-humans/unsupervised-learning-f45587588294
4. CASTLE, N. *Supervised vs Unsupervised Machine Learning Algorithms*. 2017. [Accesat: 09.02.2018]. Disponibil: https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms
5. *OpenCV Official Web Site*. OpenCV team, ©2018 [citat 14.02.2018]. Disponibil: https://opencv.org
6. *Advantages and Disadvantages of Python Programming Language*. [Accesat: 14.02.2018]. Disponibil: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

Date despre autor:
Maria CRISTEI, doctor, lector universitar, Departamentul Informatică, Facultatea de Matematică și Informatică, USM.
E-mail: cristeimusm@yahoo.com

Prezentat la 16.10.2018