Universitatea Tehnică “Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
2022/2023

Medical image segmentation

Raport final

Autor: Alexandru Ștefan-Albert
Grupa: 1410A
Profesor îndrumător: Marius Gavrilescu
Proiectarea Interfețelor Utilizator

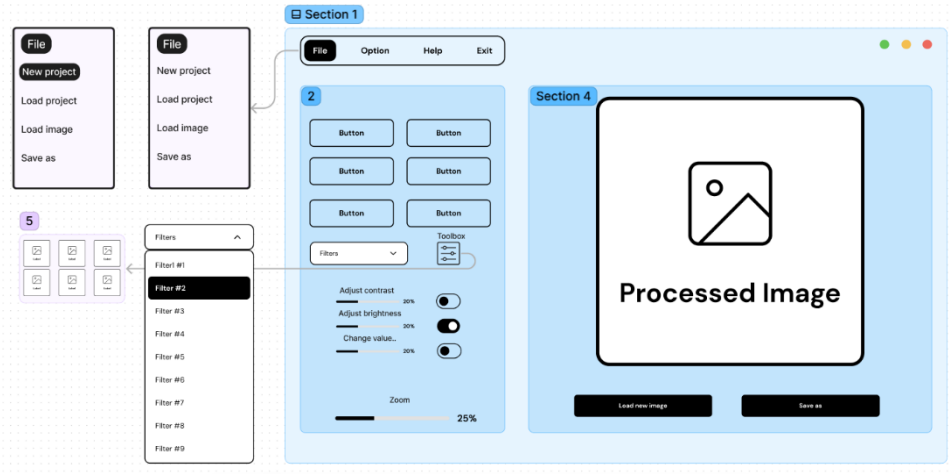
Scopul aplicației:

Dezvoltarea unei aplicatii software care permite utilizatorilor să navigheze peste imaginile medicale, și să efectueze segmentarea și prelucrarea imaginii astfel încât să obțină un cadru vizual cât mai bun pentru analiza imaginii medicale de tip MRI.

Obiective:

- Implementarea unui GUI care sa faciliteze interacțiunea utilizatorului cu aplicația.
- Implementarea unor algoritmi de segmentare și procesare a imaginilor medicale astfel încât utilizatorul să obțina cea mai buna vizualizare asuprea imaginii respective.
- Posibilitatea selectării zonei de vizualizare de către utilizator.
- Aplicarea unor filtre de reducere a zgomotului pentru a prelucrarea eficienta a imaginilor.

Un prim pas spre conceperea aplicației a fost realizarea unei schițe orientative a acesteiea cu ajutorul Figma:



(Prototipul aplicației, realizat în Figma)

Pornind de la design-ul creat in Figma, s-a dezvoltat aplicatia. Pentru realizarea interfetei grafice am utilizat QT.

Pentru implementarea algoritmilor de prelucrare a imaginilor medicale am utilizat libraria OpenCv in limbajul C++.

Aplicatia contine o clasa principala „`Ui_MainWindow`” care extinde clasa „`QObject`” din libraria QT.

Am implementat propriul label in clasa „`myLabel`” care extinde clasa „`QLabel`” si implementeaza metodele *paintEvent(), mousePressEvent(), mouseMoveEvent()* si *mouseReleaseEvent()* pentru desenarea unui QRect pe label-ul ce contine imaginea incarcata ce se doreste a fi prelucrata. Aceasta clasa contine semnalele *mousePress(), paint, mouseMove(), mouseRelease()* ce sunt transmise atunci cand se deseneaza pe Image Label.

In clasa „`FiltersClass`” sunt implementati algoritmii de prelucrare a imaginii medicale. Aceasta clasa contine metodele *convertQImageToMat()* care converteste un obiect Qimage din libraria QT intr-unul de tipul Mat din libraria OpenCv, si *convertMatToQImage()* care converteste un obiect Mat intr-unul Qimage.

Metoda *adjustContrast(**Mat img, double alpha, int beta, double gamma**)* este implementata pentru ajustarea contrasului, brightness-ului si a corectiei gama a imaginii din label.

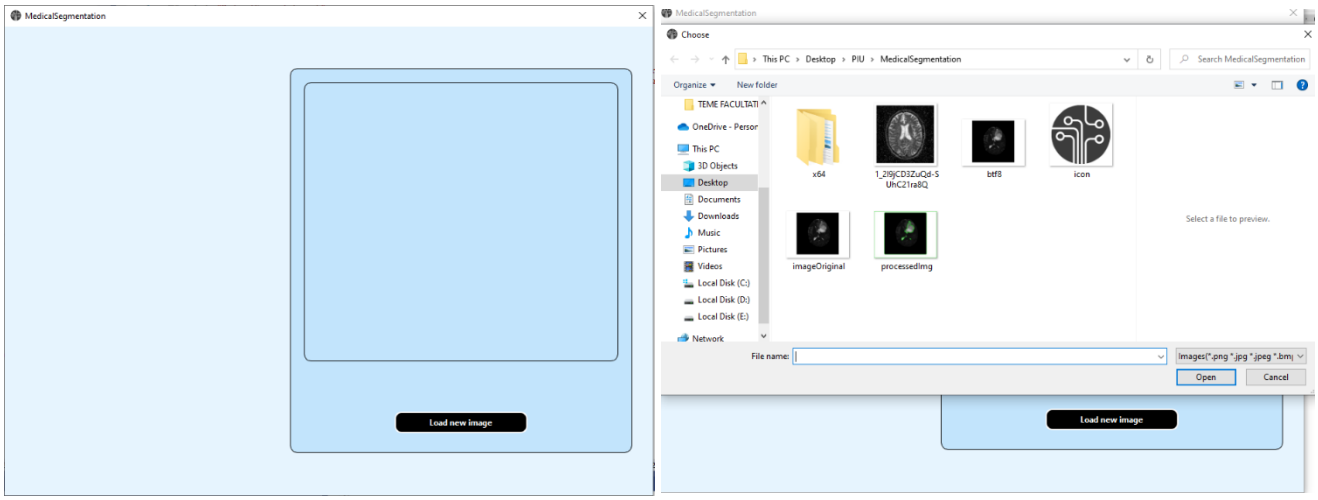
Metoda *noiseReduceMedian()* utilizeaza functia cv::medianBlur() din libraria OpenCv pentru aplicarea unui filtru median de reducere a zgomotului de tip sare si piper.

Metoda *noiseReduceGaussian()* utilizeaza functia cv::GaussianBlur() din libraria OpenCv pentru aplicarea filtrului Gaussian.

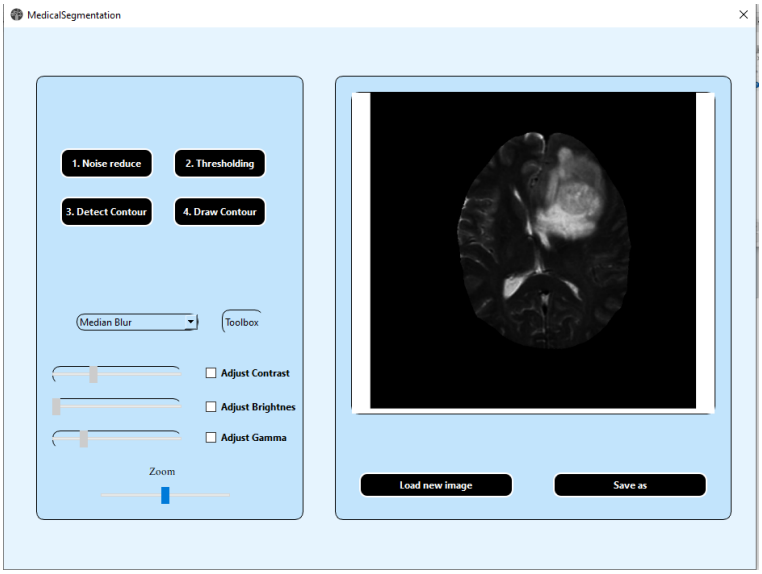
Metoda *thresholdButton()* utilizeaza functia cv::threshold() cu o valoare de threshold stabilita la 200 pentru segmentarea imaginii medicale.

detectContour() detecteaza conturul obiectului dupa aplicarea threshold-ului, iar *drawContour()* deseneaza peste imaginea initiala conturlui detectat.

In momentul rularii programului, este initializata o fereastra de dimensiune 930x670 ce contine interfata principala care permite incarcarea imaginii ce se doreste a fi prelucrata.

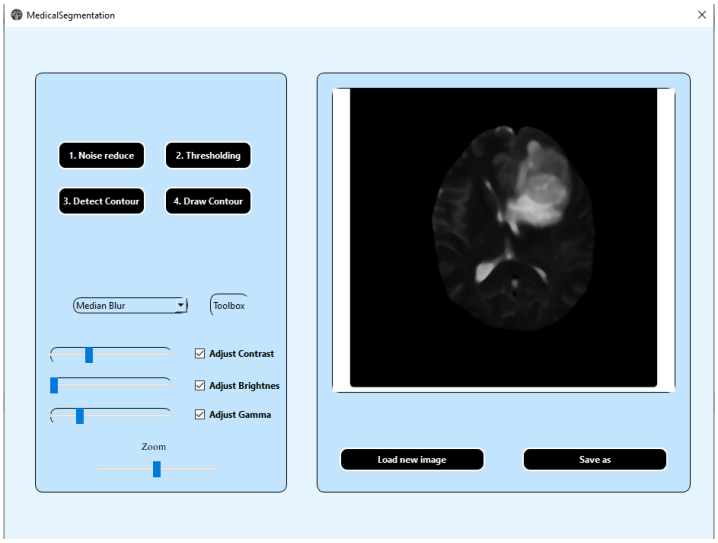


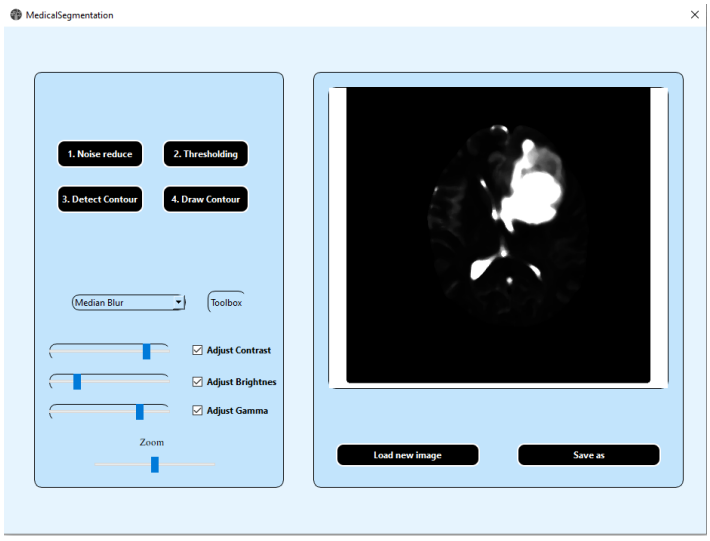
Dupa apasarea butonului Load New Image se deschide fereastra de unde vom selecta imaginea pe care dorim sa incarcam. Dupa acest pas va deveni vizibil si layout-ul din stanga ce contine butoanele si slider-ele dar si butonul de Save As.



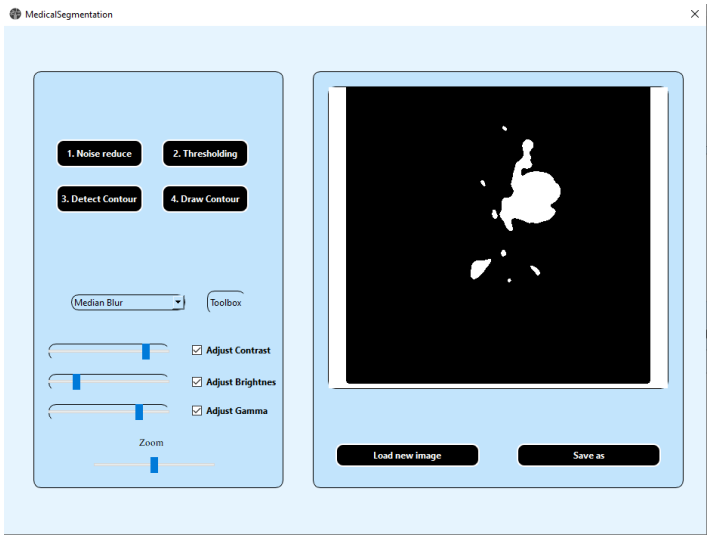
Exista un select bar de unde se selecteaza filtrul ce se doreste a fi utilizat (Median Blur sau Gaussian Blur), iar dupa apasarea butonului 1.Noise reduce se aplica filtrul peste imagine.

Dupa bifarea casutelor Adjust Contrast, Brightness sau Gamma, slide bar-urile devin active.

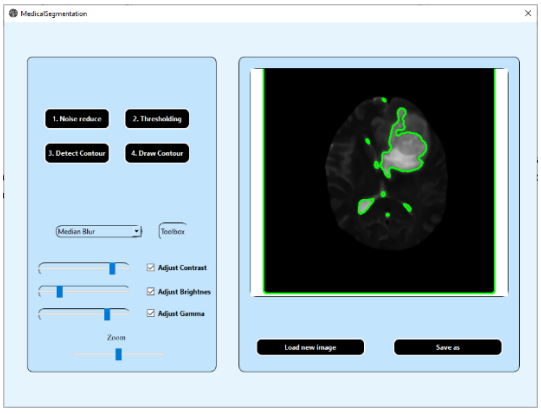




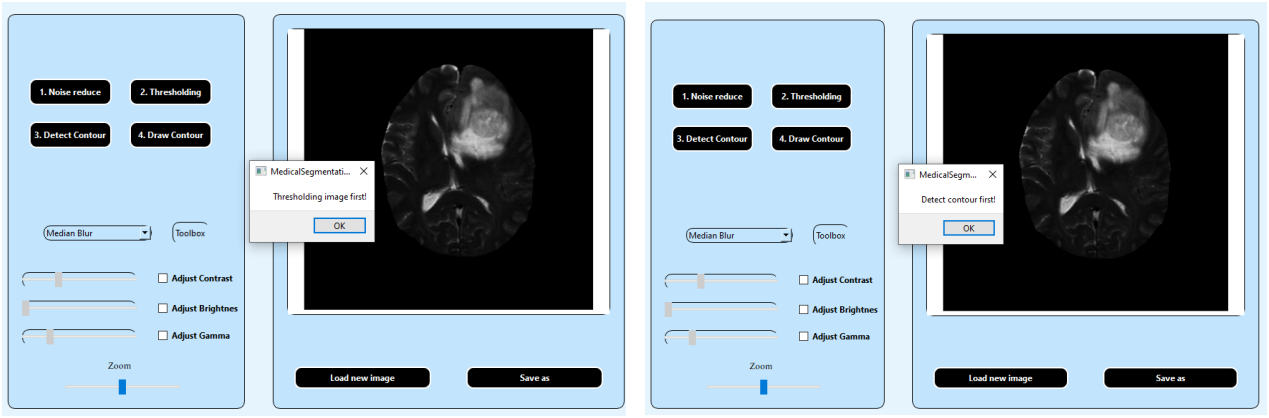
Butonul de Thresholding segmenteaza astfel imaginea:



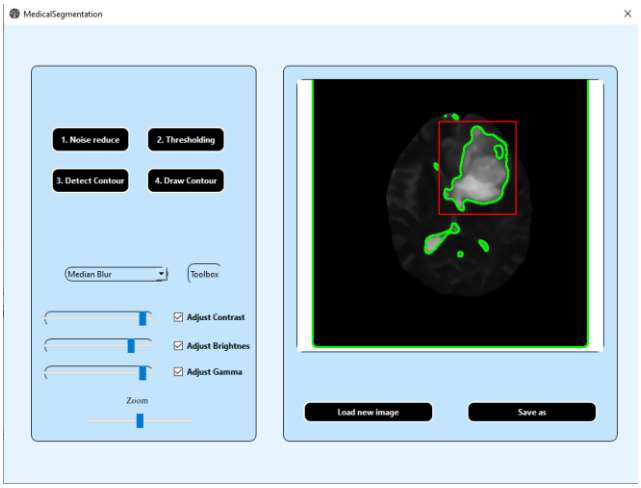
Dupa ce sa obtinut segmentarea imaginii se poate detecta conturul, iar dupa aceasta se poate desena conturul peste imaginea principala. In caz de reusita se va primi un message box cu textul „Contour detected”.



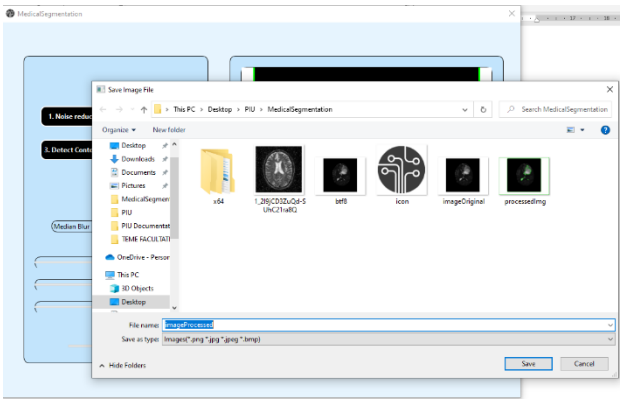
In cazul in care imaginea nu a fost segmentata anterior si se apasa butonul 3.Detect Contour, se va primi un mesaj de atentionare. Iar daca se apasa butonul 4. Draw Contour fara sa fi fost detectat anterior un contur, se va afisa pe ecran un Message box de atentionare.

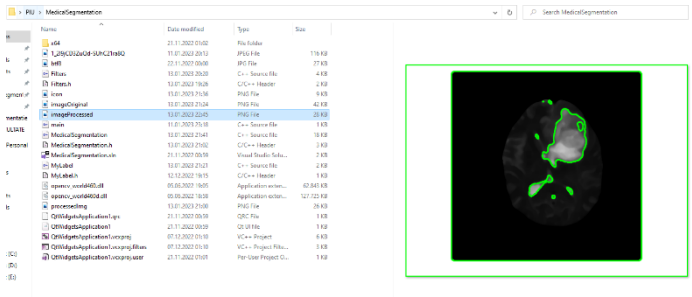


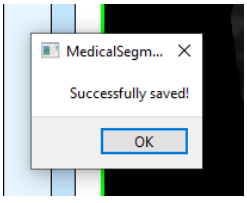
Se poate de asemenea selecta o zona din imagine pentru evidentiere, desenandu-se de catra utilizator un dreptunghi rosu, tinand click-ul apasat pe imagine si navigand dintr-un colt intr-altul pentru selectarea zonei.



Ultimul pas consta in salvarea imaginii prelucrate. Dupa apasarea butonului Save as se dechide o fereastra unde vom selecta folder-ul unde dorim sa salvam imaginea.





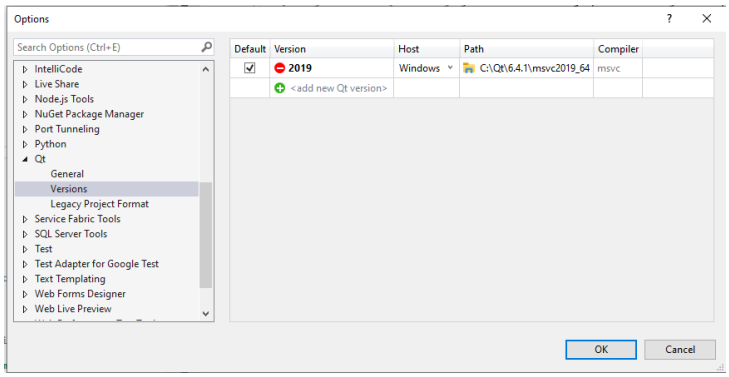


Tehnologii utilizate:

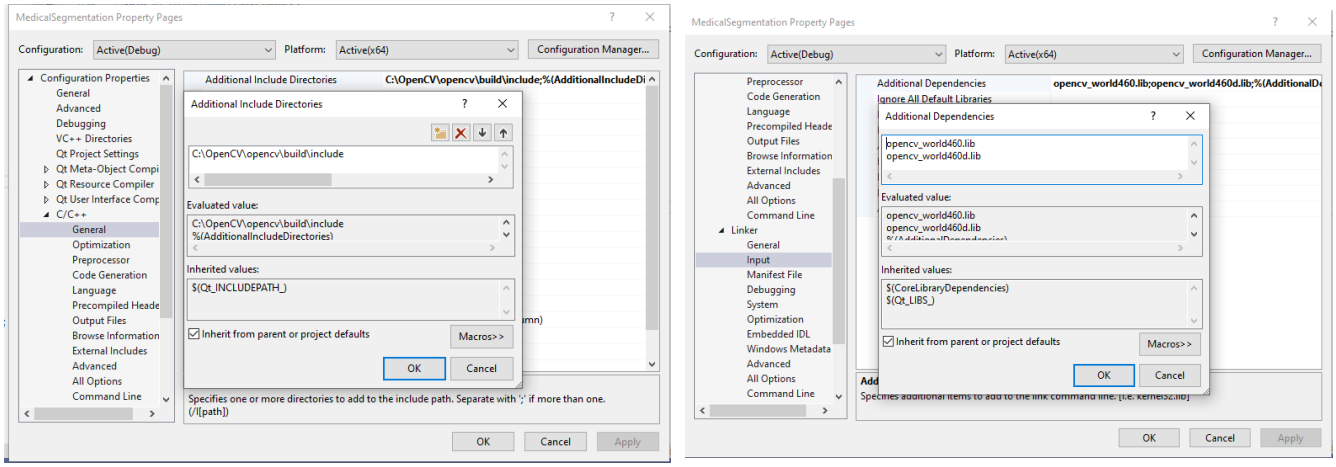
Pentru implementarea algoritmilor de prelucrare a imaginii am utilizat biblioteca **OpenCv**, iar pentru dezvoltarea interfeței grafice **Qt** in C++.

Utilizez mediul de dezvoltare **Visual Studio 2022** iar suplimentar am instalat **Qt versiunea 6.4.1** și **OpenCv versiunea 4.6.0**.

Pentru functionarea si rularea aplicatiei este necesara adaugarea dependintelor OpenCv si instalarea Qt-ului pentru Visual Studio.



(Versiunea de QT)



(Dependintele necesare pentru libraria OpenCv)