

BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP – Y SINH



ĐỒ ÁN CAPSTONE
NGÀNH: KỸ THUẬT Y SINH
ĐỀ TÀI: XÂY DỰNG THUẬT TOÁN PHÂN VÙNG ẢNH ĐỂ TÍNH
KÍCH THƯỚC KHỐI U NÃO

GVHD: T.S Nguyễn Thanh Nghĩa

SVTH: MSSV

1. Nguyễn Thị Hoài 19129018

2. Đào Thái Phương Uyên 19129064

Tp. Hồ Chí Minh – 12/2022

NHIỆM VỤ ĐỒ ÁN CAPSTONE

Họ tên sinh viên:	Nguyễn Thị Hoài	MSSV: 19129018
	Đào Thái Phương Uyên	MSSV: 19129064
Chuyên ngành:	Kỹ thuật Y sinh	Mã ngành: 01
Hệ đào tạo:	Đại học chính quy	Mã hệ: 1
Khóa:	2019	Lớp: 191290C

I. TÊN ĐỀ TÀI:

XÂY DỰNG THUẬT TOÁN PHÂN VÙNG ẢNH ĐỂ TÍNH KÍCH THƯỚC KHỐI U NÃO

II. NHIỆM VỤ

1. Các số liệu ban đầu:

- Tài liệu về xử lý ảnh.
- Tài liệu về não và u não.
- Phần mềm huấn luyện: Matlab.
- Tập dữ liệu Brain Tumors được sử dụng để phân vùng u não.

2. Nội dung thực hiện:

NỘI DUNG 1: Tìm hiểu lý thuyết về khối u não, cách tạo hình ảnh MRI và đặc điểm ảnh MRI.

NỘI DUNG 2: Nghiên cứu các phương pháp phân vùng ảnh.

NỘI DUNG 3: Tìm hiểu phương pháp nâng cao chất lượng ảnh.

NỘI DUNG 4: Tìm hiểu phương pháp tìm biên ảnh, công thức tính toán kích thước khối u não bao gồm diện tích và thể tích.

NỘI DUNG 5: Thiết kế giao diện GUI trên Matlab.

NỘI DUNG 6: Chạy mô phỏng trên Matlab, hiệu chỉnh, thu thập kết quả.

NỘI DUNG 7: Đánh giá kết quả thực hiện.

NỘI DUNG 8: Viết báo cáo.

III. NGÀY GIAO NHIỆM VỤ: 12/10/2022

IV. NGÀY HOÀN THÀNH NHIỆM VỤ: 29/12/2022

V. HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN: TS. Nguyễn Thanh Nghĩa

CÁN BỘ HƯỚNG DẪN BM. ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

LỊCH TRÌNH THỰC HIỆN ĐỒ ÁN CAPSTONE

Họ tên sinh viên 1: Nguyễn Thị Hoài

Lớp: 191290C

MSSV: 19129018

Họ tên sinh viên 2: Đào Thái Phương Uyên

Lớp: 191290C

MSSV: 19129064

Tên đề tài: **Xây dựng thuật toán phân vùng ảnh để tính kích thước khối u não.**

<i>Tuần/ngày</i>	<i>Nội dung</i>	<i>Xác nhận GVHD</i>
Tuần 1 (19/09 – 25/09)	- Gặp GVHD nghe phổ biến yêu cầu làm ĐAMH.	
Tuần 2 (26/09 – 02/10)	- Tiến hành chọn đề tài. - GVHD xét duyệt đề tài.	
Tuần 3 + 4 (03/10 – 09/10)	- Đọc và tìm hiểu tài liệu liên quan. - Hoàn thành đề cương chi tiết.	
Tuần 5 + 6 (10/10 – 23/10)	- Tìm hiểu cơ sở lý thuyết về tăng cường biên, tìm kích thước khối u.	
Tuần 7 + 8 (24/10 – 06/11)	- Xây dựng và phân tích sơ đồ khối của hệ thống	

Tuần 9 + 10 (07/11 – 20/11)	- Tiến hành lập trình trên Matlab	
Tuần 11 (21/11 – 27/11)	- Xây dựng giao diện GUI	
Tuần 12 + 13 (28/11 – 11/12)	- Chạy mô phỏng, hiệu chỉnh mô hình (nếu cần). - Lập bảng đánh giá mô hình.	
Tuần 14 (12/12 – 18/12)	- Hoàn thành báo cáo ĐAMH.	
Tuần 15 (19/12 – 25/12)	- Hoàn thành Slide PowerPoint ĐAMH. - Báo cáo đề tài với GVHD.	

GV HƯỚNG DẪN

(Ký và ghi rõ họ và tên)

LỜI CAM ĐOAN

Đề tài này là do chúng tôi tự tìm hiểu, nghiên cứu dựa trên một số tài liệu trước đó dưới sự hướng dẫn của TS. Nguyễn Thanh Nghĩa và không sao chép bất cứ các tài liệu hay công trình nghiên cứu liên quan nào trước đó nhằm mục đích phục vụ cho đề tài. Nếu có, chúng tôi xin chịu hoàn toàn trách nhiệm.

Người thực hiện đề tài

Nguyễn Thị Hoài

Đào Thái Phương Uyên

LỜI CẢM ƠN

Chúng em xin chân thành gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Thanh Nghĩa đã trực tiếp hướng dẫn và tận tình giúp đỡ, góp ý và chia sẻ nhiều kinh nghiệm quý báu để chúng em có thể hoàn thành tốt đề tài. Đồng thời, trong quá trình thực hiện đề tài, nhóm chúng em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo tận tình của Thầy cô và bạn bè.

Do điều kiện khó khăn về thời gian và kiến thức còn hạn chế nên đề tài vẫn còn sai sót. Chúng em rất mong nhận được những lời đóng góp, nhận xét từ Thầy, góp phần giúp cho đề tài báo cáo đồ án của nhóm được hoàn thiện hơn.

Xin chân thành cảm ơn!

Người thực hiện đề tài

Nguyễn Thị Hoài

Đào Thái Phương Uyên

MỤC LỤC

NHIỆM VỤ ĐỒ ÁN CAPSTONE	i
LỊCH TRÌNH THỰC HIỆN ĐỒ ÁN CAPSTONE	iii
LỜI CAM ĐOAN	v
LỜI CẢM ƠN	vi
MỤC LỤC	vii
DANH MỤC HÌNH ẢNH	x
DANH MỤC BẢNG	xi
TÓM TẮT	xii
CHƯƠNG 1. TỔNG QUAN	1
1.1 ĐẶT VẤN ĐỀ	1
1.2 MỤC TIÊU	2
1.3 NỘI DUNG NGHIÊN CỨU	2
1.4 GIỚI HẠN	3
1.5 BỐ CỤC	3
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	5
2.1 MÔ TẢ VỀ BỆNH U NÃO	5
2.1.1 Khái quát về u não	5
2.1.2 U não lành tính và u não ác tính	6
2.1.3 Phân loại khối u theo cấp độ	6
2.2 ẢNH MRI	7
2.2.1 Tổng quan ảnh cộng hưởng từ	7
2.2.2 Nguyên lý của MRI	8
2.2.3 Ưu điểm của chụp MRI	9
2.2.4 Nhược điểm của chụp MRI	10
2.3 PHÂN VÙNG ẢNH	10
2.3.1 Phân vùng dựa trên ngưỡng Otsu	11
2.3.2. Thuật toán biến đổi Watershed	13

2.3.3 Phân vùng dựa trên kỹ thuật phân cụm	14
2.3.3.1 Phân cụm Fuzzy C-Means	14
2.3.3.2 Phân cụm K-means	17
CHƯƠNG 3: TÍNH TOÁN VÀ THIẾT KẾ	20
3.1 GIỚI THIỆU	20
3.2 QUY TRÌNH XÁC ĐỊNH KÍCH THƯỚC KHỐI U NÃO	20
3.3 ĐỌC ẢNH ĐẦU VÀO	21
3.4 TIỀN XỬ LÝ	21
3.4.1 Tách ảnh màu thành ảnh xám	22
3.4.2 Lọc nhiễu	22
3.5 PHÂN VÙNG ẢNH	25
3.5.1 Thuật toán ngưỡng Otsu	25
3.5.2 Thuật toán biến đổi Watershed	30
3.5.3 Thuật toán phân cụm K – Means	35
3.5.4 Phương pháp Phân cụm Fuzzy C Means	40
3.5 XÁC ĐỊNH VỊ TRÍ KHỐI U	47
3.5.1 Bouding Box	48
3.5.2 Tọa độ tâm của khối u	49
3.6 TÍNH KÍCH THƯỚC KHỐI U	49
3.7 THIẾT KẾ GIAO DIỆN GUI	50
3.7.1 Giới thiệu giao diện GUI trên Matlab	50
3.7.2 Lưu đồ giải thuật	51
3.7.3 Tài liệu hướng dẫn sử dụng, thao tác	52
CHƯƠNG 4: KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ	60
4.1 KẾT QUẢ	60
4.2 NHẬN XÉT	78
4.3 ĐÁNH GIÁ	79
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	80
5.1 KẾT LUẬN	80

5.2 HƯỚNG PHÁT TRIỂN	80
TÀI LIỆU THAM KHẢO	81
PHỤ LỤC	83

DANH MỤC HÌNH ẢNH

Hình 2.1 Nguyên lý cơ bản của máy chụp cộng hưởng từ - MRI	9
Hình 3.1 Sơ đồ khối quy trình xác định kích thước khối u	20
Hình 3.2 Ảnh trước và sau khi được tiền xử lý	22
Hình 3.3 Phân vùng khối u não dùng thuật toán Otsu	26
Hình 3.4 Phân vùng ảnh u não dùng thuật toán Watershed	31
Hình 3.5 Phân vùng ảnh u não dùng thuật toán K - Means	36
Hình 3.6 Phân vùng ảnh u não dùng thuật toán Fuzzy C Means	41
Hình 3.7 Lưu đồ viết chương trình giao diện GUI trên Matlab	51
Hình 3.8 Giao diện mô phỏng GUI trên Matlab	52
Hình 3.9 File mô phỏng GUI	53
Hình 3.10 Nhấn Load Image để nhập ảnh	53
Hình 3.11 File ảnh MRI u não	54
Hình 3.12 Nút nhấn Load Image trên GUI	54
Hình 3.13 Nút nhấn Preprocessing trên GUI	55
Hình 3.14 Menu thuật toán Segmentation trên GUI	55
Hình 3.15 Nút nhấn Tumor Alone trên GUI	56
Hình 3.16 Nút nhấn Tumor Area Location trên GUI	56
Hình 3.17 Nút nhấn Tumor Outline trên GUI	57
Hình 3.18 Nút nhấn Detect Area and Perimeter trên GUI	57
Hình 3.19 File ảnh masks của khối u	58
Hình 3.20 Nút nhấn Evaluate trên GUI	58
Hình 3.21 Nút nhấn Reset trên GUI	59
Hình 4.1 Đồ thị đánh giá độ chính xác các thuật toán	78

DANH MỤC BẢNG

Bảng 4.1 Kết quả phân vùng khối u của các thuật toán	62
Bảng 4.2 Kết quả xác định ví của từng thuật toán	64
Bảng 4.3 Kết quả tính kích thước và vị trí của khối u	66
Bảng 4.4 So sánh độ chính xác của các Thuật toán Otsu	69
Bảng 4.5 Đánh giá độ chính xác của Thuật toán Watershed	72
Bảng 4.6 So sánh độ chính xác của các Thuật toán K - Means	74
Bảng 4.7 So sánh độ chính xác của các Thuật toán Fuzzy C Means	76
Bảng 4.8 Đánh giá độ chính xác của các thuật toán	78

TÓM TẮT

Trong tình hình thế giới hiện nay, ung thư não ảnh hưởng gần 1% dân số thế giới. Các tế bào ung thư dưới dạng khối u được hình thành trong mô não là nguyên nhân gây ra ung thư não, gây cản trở hoạt động của não. Các phương pháp có thể được áp dụng trong liệu trình điều trị ung thư não bộ gồm có phẫu thuật cắt bỏ khối u, xạ trị, hóa trị. Trong tất cả các vấn đề liên quan đến chữa trị ung thư hiệu quả như trên đều liên quan đến nâng cao chất lượng hình ảnh và xác định kích thước của khối u. Vì vậy nhóm chúng em đã lựa chọn đề tài “XÂY DỰNG THUẬT TOÁN PHÂN VÙNG ẢNH ĐỂ TÍNH KÍCH THƯỚC KHỐI U NÃO” nhằm tạo ra một giao diện hiển thị các thông số đo hiệu suất được sử dụng để đánh giá độ đặc hiệu, độ nhạy và độ chính xác và hiển thị kích thước khối u não bao gồm diện tích và thể tích.

CHƯƠNG 1. TỔNG QUAN

1.1 ĐẶT VẤN ĐỀ

Theo thống kê trên thế giới mỗi năm có khoảng 10 triệu người chết vì ung thư. Ung thư được xem là nguyên nhân gây tử vong đứng hàng thứ hai trên toàn thế giới. Trong đó, 70% trường hợp tử vong do ung thư xảy ra ở các nước có thu nhập thấp đến trung bình. Ung thư vẫn luôn là mối quan tâm trên toàn cầu. Theo thống kê của GLOBOCAN năm 2020 của Cơ quan nghiên cứu ung thư quốc tế (International Agency on Cancer Research – IACR) trực thuộc Tổ chức Y tế Thế Giới (WHO) tại Việt Nam, ước tính có 182.563 ca mắc mới và 122.690 ca tử vong do ung thư, tình hình mắc và tử vong do ung thư đều có xu hướng tăng [1].

U não là bệnh liên quan đến ung thư gây tử vong cao nhất ở các bệnh nhân dưới 14 tuổi và là nguyên nhân tử vong thứ 2 ở bệnh nhân dưới 20 tuổi. Có khoảng 120 loại u não khác nhau, hầu hết là các khối u trong mô não, ngoài ra là u ở màng não, tuyến yên, dây thần kinh sọ não và nhiều loại khác [2]. U não là tình trạng các tế bào bất thường tăng trưởng trong não. U não gồm hai loại là u não lành tính và u não ác tính. Cả hai loại u não trên đều gây ảnh hưởng đến tế bào não, khiến não bị tổn thương, thậm chí là nguy hiểm đến tính mạng. Xử lý hình ảnh y tế là việc sử dụng các kỹ thuật xử lý hình ảnh để nâng cao hiệu quả sử dụng các thiết bị chuẩn đoán đo lường bằng hình ảnh [3]. Xử lý ảnh trong y tế giúp chẩn đoán thông qua hình ảnh chính xác, hiệu quả, nhanh chóng, hỗ trợ hiệu quả cho bác sĩ trong chẩn đoán bệnh. Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn.

Hình ảnh y khoa là kỹ thuật và quy trình tạo hình ảnh trực quan về bên trong của cơ thể để phân tích lâm sàng và can thiệp y tế, cũng như biểu thị trực quan chức năng của một số cơ quan hoặc mô sinh lý học. Hình ảnh y khoa nhằm tìm kiếm các cấu trúc bên trong được che giấu bởi da và xương cũng như chẩn đoán và điều trị bệnh. Hình ảnh y

khoa sử dụng các công nghệ hình ảnh như X-quang, cắt lớp vi tính (CT), cộng hưởng từ (MRI), chụp ảnh y khoa (medical photography), chụp cắt lớp trở kháng điện (EIT, Electrical impedance tomography) và kỹ thuật hình ảnh y học hạt nhân như chụp cắt lớp phát xạ positron (PET, positron emission tomography) và chụp cắt lớp vi tính phát xạ photon đơn (SPECT, Single-photon emission computed tomography) [4].

Qua quá trình nghiên cứu và học tập tại trường đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh, nhóm đã tìm hiểu về luận văn của thạc sĩ Trần Thị Quỳnh Như, với đề tài “Dùng phương pháp Wavelet tăng cường biên ảnh để xác định kích thước khối u đặc” [5]. Với việc kế thừa và phát triển từ đề tài, nhóm sinh viên quyết định mở rộng thêm với đề tài **“Xây dựng thuật toán phân vùng ảnh để tính kích thước khối u não”** với chất lượng ảnh tốt hơn, hiệu quả chuẩn đoán cao, nhóm còn thiết kế giao diện GUI trên phần mềm Matlab để thể hiện trực quan hóa hình ảnh.

1.2 MỤC TIÊU

Đề tài nhằm mục đích xử lý ảnh để tăng cường ảnh cho ra chất lượng ảnh tốt hơn, xác định kích thước khối u chính xác hỗ trợ tốt hơn cho điều trị ung thư. Do đó trong đề tài này, chúng em xác định kích thước khối u não dùng các kỹ thuật phân vùng ảnh gồm phân vùng dựa trên ngưỡng Otsu, thuật toán biến đổi Watershed, phân vùng dựa trên kỹ thuật phân cụm K – Means và Fuzzy C – Means để tăng cường biên ảnh; kích thước khối u ở đây là diện tích và chu vi khối u. Đồng thời, nhóm thiết kế giao diện GUI trên phần mềm Matlab để thể hiện trực quan hóa hình ảnh.

1.3 NỘI DUNG NGHIÊN CỨU

Trong quá trình thực hiện Đề tài: “Xây dựng thuật toán phân vùng ảnh để tính kích thước khối u não”, nhóm đã tập trung giải quyết và hoàn thành được những nội dung sau:

- NỘI DUNG 1: Tìm hiểu lý thuyết về khối u não, cách tạo hình ảnh MRI và đặc điểm ảnh MRI.

- NỘI DUNG 2: Nghiên cứu các phương pháp phân vùng ảnh.
- NỘI DUNG 3: Tìm hiểu phương pháp nâng cao chất lượng ảnh.
- NỘI DUNG 4: Tìm hiểu phương pháp tìm biên ảnh, công thức tính toán kích thước khối u não bao gồm diện tích và thể tích.
- NỘI DUNG 5: Thiết kế giao diện GUI trên Matlab.
- NỘI DUNG 6: Chạy mô phỏng trên Matlab, hiệu chỉnh, thu thập kết quả.
- NỘI DUNG 7: Đánh giá kết quả thực hiện.
- NỘI DUNG 8: Viết báo cáo.

1.4 GIỚI HẠN

- Chỉ xử lý ảnh 2D và không xử lý ảnh CT hoặc X - quang u não.
- Không tính được thể tích khối u.
- Chỉ xây dựng được 3/5 các kỹ thuật phân vùng ảnh.
- Hệ thống chỉ mô phỏng trên Matlab và không có phần cứng.

1.5 BỐ CỤC

Với đề tài “Xây dựng thuật toán phân vùng ảnh để tính kích thước khối u não” được trình bày với bố cục như sau:

Chương 1: Tổng Quan

Trình bày đặt vấn đề dẫn nhập lý do chọn đề tài, mục tiêu, nội dung nghiên cứu, các giới hạn thông số và bố cục đồ án.

Chương 2: Cơ Sở Lý Thuyết

Trình bày cơ sở lý thuyết liên quan tới đề tài, cũng như nêu một vài phương pháp tăng cường biên ảnh để xác định kích thước khối u đặc hiệu nay.

Chương 3: Tính Toán và Thiết Kế

Trình bày phương pháp tính toán và thiết kế hệ thống, thiết kế sơ đồ khối, chức năng từng khối và thực thi chương trình trên MATLAB.

Chương 4: Kết Quả, Nhận Xét và Đánh Giá

Nêu kết quả đã đạt được, nhận xét đánh giá mô hình.

Chương 5: Kết Luận và Hướng Phát Triển

Trình bày ngắn gọn những kết quả đã thu được dựa vào những phương pháp đề ra. Đồng thời, tiến hành kết luận, tổng kết công việc hoàn thành ngắn gọn của quá trình và đề xuất những phương pháp khác để cải thiện đề tài.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 MÔ TẢ VỀ BỆNH U NÃO

2.1.1 Khái quát về u não

U não là một tập hợp số lượng lớn các tế bào não phát triển bất thường vượt ngoài tầm kiểm soát của cơ thể. Các u não có thể bắt đầu trực tiếp từ tế bào não, tế bào đệm của hệ thần kinh trung ương, hoặc cũng có thể bắt đầu từ các bộ phận khác, rồi theo máu đến não, được gọi là u di căn não.

Cơ chế hình thành: Thông thường, từ lúc sinh ra đến lúc mất đi, không có thêm tế bào thần kinh nào được sinh thêm ra nữa. Khi có đột biến không rõ nguyên nhân trong DNA khiến các tế bào phân chia mất kiểm soát thì sẽ hình thành nên u não.

Ảnh hưởng: Tốc độ phát triển cũng như vị trí của u não quyết định mức độ nghiêm trọng và tầm ảnh hưởng của nó đến chức năng hệ thần kinh, thậm chí là đe dọa đến tính mạng nếu không được chẩn đoán, theo dõi và chữa trị kịp thời.

Mức độ phổ biến: U não chiếm 2% trong tổng số các ca ung thư từ mọi nhóm tuổi. Trong số các trường hợp tử vong do ung thư ở nhóm trẻ em dưới 15 tuổi và nhóm từ 20-39 tuổi, bệnh u não là nguyên nhân gây tử vong cao thứ 2. Người ngoài 85 tuổi có tỉ lệ bị u não cao nhất.

Có khoảng 120 loại u não khác nhau, hầu hết là các khối u trong mô não, ngoài ra là u ở màng não, tuyến yên, dây thần kinh sọ não... Bất cứ dạng u não nào cũng có thể gây nguy hiểm cho người bệnh. Các khối u ở mô não hoặc u não lành tính thường tiến triển chậm, các triệu chứng của u não trong trường hợp này cũng sẽ xuất hiện chậm và diễn biến âm ỉ hơn. Ngược lại, nếu u não phát triển nhanh, người bệnh sẽ thấy các triệu chứng rõ rệt hơn cả về tần suất và mức độ [6].

2.1.2 U não lành tính và u não ác tính

- U não lành tính (không phải ung thư)

Định nghĩa: U não lành tính là loại u não phát triển chậm, không di căn, có thể được điều trị dứt điểm bằng cách phẫu thuật loại bỏ khối u trực tiếp mà không cần xạ trị hay hóa trị.

Đặc điểm: Khối u lành tính tuy có thể chữa lành nhưng vẫn có thể đe dọa tính mạng nếu không được loại bỏ kịp thời nhờ phẫu thuật. Bệnh ít có khả năng tái phát sau phẫu thuật nếu được loại bỏ hoàn toàn. Do đó, bạn cần chủ động khám sức khỏe định kỳ để tầm soát, phát hiện sớm khối u lành tính (nếu có).

- U não ác tính (ung thư)

Định nghĩa: U não ác tính là loại u não chứa tế bào ung thư (tế bào phân chia nhanh vượt mức kiểm soát). Khối u này dễ tấn công và di căn sang các vùng tế bào khỏe mạnh lân cận. Bệnh phát triển nhanh, dễ tái phát và ảnh hưởng trực tiếp đến tính mạng bệnh nhân.

Đặc điểm: Khi một bệnh nhân mắc khối u ác tính, phương pháp điều trị thường là kết hợp xạ trị, hóa trị với phẫu thuật. U não ác tính chiếm gần 30% tổng số ca u não nguyên phát, còn u não thứ phát (do di căn) thì 100% là khối u ác tính [6].

2.1.3 Phân loại khối u theo cấp độ

Chia u não theo 4 cấp độ (hay còn gọi 4 giai đoạn) từ giai đoạn 1 đến giai đoạn 4. Để xác định cấp độ (giai đoạn) của khối u, bác sĩ sẽ căn cứ theo đặc điểm hình dạng của tế bào khối u quan sát được dưới kính hiển vi. Hình dạng mẫu tế bào khối u trông càng bình thường (giống với tế bào khỏe mạnh) thì cấp độ u não càng thấp.

Một số loại u não thường gặp:

Hiện nay, có 04 loại khối u não thường gặp là:

- U não Gliomas: Còn gọi là u thần kinh đệm vì đây là khối u não bắt đầu từ trong các tế bào thần kinh đệm ở não hoặc tủy sống. U não Gliomas là loại u não nguyên phát ác tính, chiếm khoảng 50,4% tổng số các ca u não và 78% các ca có khối u não nguyên phát ác tính.

- U màng não: Là một khối u phát triển chậm, hình thành từ màng não hay lớp màng bao quanh tủy sống. U màng não là loại u lành tính, thường xuất hiện ở nữ giới, chiếm khoảng 20,8% tổng số các ca bị u não với tỉ lệ tái phát sau phẫu thuật thấp (ít hơn 20%).

- U tuyến yên: Là khối u xảy ra trong tuyến yên (nằm ở bề mặt dưới não) với hơn 60% các ca được chẩn đoán là lành tính, 35% là loại khối u có xâm lấn. Theo thống kê, u tuyến yên chiếm từ 10% – 25% trên tổng số các ca u não và tỷ lệ mắc bệnh có thể lên đến 17% dân số.

- Khối u thần kinh ngoại biên: Do các nguyên bào sợi tăng trưởng bao quanh bó thần kinh gây ra và chiếm khoảng 10% tổng số ca bị u não. Hầu hết các khối u thần kinh ngoại biên là lành tính (không phải ung thư). Tuy nhiên, khối u chèn ép thần kinh gây đau và có thể làm mất khả năng kiểm soát cơ bắp [6].

2.2 ẢNH MRI

2.2.1 Tổng quan ảnh cộng hưởng từ

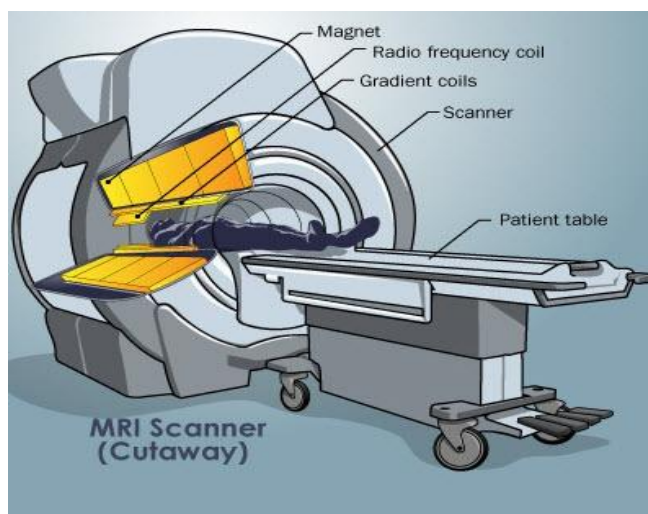
Chụp cộng hưởng từ hay chụp MRI (Magnetic Resonance Imaging) là một trong những kỹ thuật chẩn đoán hình ảnh tiên tiến nhất hiện nay. Đây là phương pháp đưa cơ thể vào vùng từ trường mạnh để đồng hóa chiều chuyển động của các nguyên tử Hydro trong các phân tử nước của cơ thể và một ăng ten thu phát sóng radio tần số thấp (tần số radio này được thay đổi trong vùng từ trường ổn định của nam châm chính tùy theo mục đích khảo sát của sự phân biệt mỡ, nước...) được sử dụng để gửi tín hiệu đến cơ thể gặp

các nguyên tử Hydro của cơ thể sau đó nhận lại tín hiệu về chiều chuyển động của các nguyên tử này, tín hiệu của ăng ten được truyền về trung tâm xử lý tín hiệu và tái tạo thành hình ảnh cấu trúc của các cơ quan, bộ phận trong cơ thể cần khảo sát. Những hình ảnh thu nhận được sẽ cung cấp nhiều thông tin có giá trị trong việc chẩn đoán và điều trị bệnh [7].

2.2.2 Nguyên lý của MRI

Trong cơ thể người chứa rất nhiều nước, đồng nghĩa với có nhiều nguyên tử hydro (proton). Ngoài ra, trong những môi trường khác nhau thì quá trình hồi phục của các hạt nhân diễn ra khác nhau. Do đó, tín hiệu phát ra tại giữa vùng bình thường và khối u, hay giữa mô cứng và mô mềm sẽ không giống nhau. Từ đó ta sẽ có các hình ảnh khác nhau.

Cơ thể chúng ta cấu tạo chủ yếu từ nước (60-70%). Trong thành phần của phân tử nước luôn có nguyên tử hydro. Về mặt từ tính, nguyên tử hydro là một nguyên tử đặc biệt vì hạt nhân của chúng chỉ chứa 1 proton. Do đó, nó có một mômen từ lớn. Từ điều này dẫn tới một hệ quả là: nếu ta dựa vào hoạt động từ của các nguyên tử hydro để ghi nhận sự phân bố nước khác nhau của các mô trong cơ thể thì chúng ta có thể ghi hình và phân biệt được các mô đó. Mặt khác, trong cùng một cơ quan, các tổn thương bệnh lý đều dẫn đến sự thay đổi phân bố nước tại vị trí tổn thương, dẫn đến hoạt động từ tại đó sẽ thay đổi so với mô lành, nên ta cũng sẽ ghi hình được các thương tổn.



Hình 2.1 Nguyên lý cơ bản của máy chụp cộng hưởng từ - MRI

Ứng dụng nguyên lý này, MRI sử dụng một từ trường mạnh và một hệ thống phát các xung có tần số vô tuyến để điều khiển hoạt động điện từ của nhân nguyên tử, mà cụ thể là nhân nguyên tử hydro có trong phân tử nước của cơ thể, nhằm bức xạ năng lượng dưới dạng các tín hiệu có tần số vô tuyến. Các tín hiệu này sẽ được một hệ thống thu nhận và xử lý điện toán để tạo ra hình ảnh của đối tượng vừa được đưa vào từ trường đó [7].

2.2.3 Ưu điểm của chụp MRI

Ảnh MRI cho kết quả cấu trúc các mô mềm trong cơ thể như tim, phổi, gan và các cơ quan khác rõ hơn và chi tiết hơn so với ảnh được tạo bằng các phương pháp khác.

MRI giúp cho các bác sĩ đánh giá được các chức năng hoạt động cũng như cấu trúc của nhiều cơ quan nội tạng trong cơ thể.

Sự chi tiết làm cho MRI trở thành công cụ vô giá trong chẩn đoán thời kì đầu và trong việc đánh giá các khối u trong cơ thể.

Tạo ảnh bằng MRI không gây tác dụng phụ như trong tạo ảnh bằng chụp X quang thường quy và chụp CT.

MRI cho phép dò ra các điểm bất thường ẩn sau các lớp xương mà các phương pháp tạo ảnh khác khó có thể nhận ra.

MRI có thể cung cấp nhanh và chuẩn xác so với tia X trong việc chẩn đoán các bệnh về tim mạch.

Không phát ra các bức xạ gây nguy hiểm cho con người.

2.2.4 Nhược điểm của chụp MRI

Các vật bằng kim loại cấy trong cơ thể không được phát hiện có thể bị ảnh hưởng bởi từ trường mạnh.

Không sử dụng với các bệnh nhân mang thai ở 12 tuần đầu tiên. Các bác sĩ thường được sử dụng các phương pháp tạo ảnh khác, ví dụ như siêu âm, với các phụ nữ mang thai trừ khi thật cần thiết bắt buộc phải sử dụng MRI.

2.3 PHÂN VÙNG ẢNH

Phân vùng ảnh (Image Segmentation) là một phần quan trọng trong lĩnh vực thị giác máy tính (Computer Vision). Nó là quá trình chia nhỏ một bức ảnh thành nhiều phần, với mục tiêu đơn giản hóa hoặc thay đổi biểu diễn của bức ảnh để dễ dàng phân tích. Phân vùng ảnh cũng có một mục tiêu chung với phát hiện vật thể (Object Detection) là phát hiện ra vùng ảnh chứa vật thể và gán nhãn phù hợp cho chúng. Tuy nhiên tiêu chuẩn về độ chính xác của Image Segmentation ở mức cao hơn so với Object Detection khi nó yêu cầu nhãn dự báo đúng tới từng pixel. Trong quá trình này, mỗi pixel trong hình ảnh được liên kết với một loại đối tượng. Có hai kiểu phân đoạn hình ảnh chính – phân đoạn ngữ nghĩa (semantic segmentation) và phân đoạn cá thể (instance segmentation). Từ đó, Phân vùng ảnh có thể chỉ ra thông tin chi tiết của bức ảnh, bao gồm: Vị trí của vật thể trong ảnh, hình dạng của vật thể và từng pixel nào thuộc về vật thể nào [8].

Có 05 kỹ thuật phân vùng ảnh, bao gồm:

- Phân vùng dựa trên ngưỡng (Threshold Based Segmentation).
- Phân vùng dựa trên cạnh (Edge Based Segmentation).
- Phân vùng dựa trên khu vực (Region-Based Segmentation).
- Phân vùng dựa trên kỹ thuật phân cụm (Clustering Based Segmentation).
- Phân vùng dựa trên mạng nơron nhân tạo (Artificial Neural Network Based Segmentation)

Trong báo cáo này, nhóm đã tập trung xây dựng thuật toán ngưỡng Otsu, biến đổi Watershed, phân cụm K – Means, phân cụm Fuzzy C – Means để phân vùng ảnh.

2.3.1 Phân vùng dựa trên ngưỡng Otsu

Phân đoạn ngưỡng ảnh là một dạng phân vùng ảnh đơn giản, giúp tạo ra một hình ảnh nhị phân hoặc nhiều màu dựa trên việc đặt giá trị ngưỡng theo cường độ pixel của hình ảnh gốc.

Trong quá trình xác định ngưỡng, cần xem xét biểu đồ cường độ của tất cả các pixel trong hình ảnh. Sau đó, tiến hành đặt một ngưỡng để chia hình ảnh thành các phần. Đối với một ảnh có nền và đối tượng, có thể chia ảnh thành các vùng dựa trên cường độ của đối tượng và nền. Nhưng ngưỡng này phải được thiết lập hoàn hảo để phân đoạn hình ảnh thành một đối tượng và một nền.

Phân ngưỡng bao gồm các kỹ thuật như ngưỡng toàn cục (Global thresholding); ngưỡng thủ công (Manual thresholding); ngưỡng thích ứng (Adaptive Thresholding);

ngưỡng tối ưu (Optimal Thresholding); ngưỡng thích ứng cục bộ (Local Adaptive Thresholding) [8].

Phương pháp của Otsu sử dụng để thực hiện ngưỡng hình ảnh tự động. Ở dạng đơn giản nhất, thuật toán trả về một ngưỡng cường độ duy nhất tách các pixel thành hai lớp, tiền cảnh và hậu cảnh. Ngưỡng này được xác định bằng cách giảm thiểu phương sai cường độ trong lớp, hoặc tương đương, bằng cách tối đa hóa phương sai giữa các lớp [11].

Thuật toán tìm kiếm toàn diện ngưỡng giảm thiểu phương sai trong lớp, được định nghĩa là tổng phương sai có trọng số của hai lớp:

$$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Trọng lượng ω_0 và ω_1 là xác suất của hai lớp cách nhau bởi một ngưỡng t và σ_0^2 và σ_1^2 là sự khác biệt của hai lớp này.

Xác suất lớp $\omega_{0,1}(t)$ được tính toán từ L Thùng của biểu đồ:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Đối với 2 lớp, giảm thiểu phương sai trong lớp tương đương với việc tối đa hóa phương sai giữa các lớp:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \omega_{\omega}^2(t) = \omega_0(t)(\mu_0 - \mu_T)^2 + \omega_1(t)(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

Được thể hiện dưới dạng xác suất lớp ω và phương tiện giai cấp μ , nơi lớp học có nghĩa là $\mu_0(t)$, $\mu_1(t)$ và μ_T là:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$
$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$
$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

Xác suất xảy ra mức độ i xám:

$$P(i) = \frac{n_i}{n}$$

Trong đó i là giá trị pixel tối đa (255).

Do đó, quy trình của thuật toán chung cho tùy chọn tối đa hóa phương sai giữa các lớp có thể được biểu diễn theo cách sau:

- *Bước 1*: Tính toán biểu đồ và xác suất mức cường độ

Khởi tạo $w_i(0), \mu_i(0)$

- *Bước 2*: Lặp lại trên các ngưỡng có thể: $t = 0, \dots, \text{max_intensity}$.

- *Bước 3*: Cập nhật các giá trị của , trong đó w_i là xác suất và μ_i là giá trị trung bình của w_i, μ_i lớp i

- *Bước 4*: Tính giá trị phương sai giữa các lớp $\sigma_b^2(t)$. Ngưỡng cuối cùng là giá trị tối đa $\sigma_b^2(t)$

2.3.2. Thuật toán biến đổi Watershed

Trong nghiên cứu về xử lý ảnh, Watershed là phép biến đổi được xác định trên ảnh thang độ xám. Phép biến đổi Watershed xử lý hình ảnh mà nó vận hành giống như một bản đồ địa hình, với độ sáng của mỗi điểm biểu thị chiều cao của nó và tìm đường chạy dọc theo đỉnh của các đường vân. Các thuật toán Watershed được sử dụng trong xử lý ảnh chủ yếu cho mục đích phân đoạn đối tượng, tức là để tách các đối tượng khác nhau trong một ảnh. Điều này cho phép đếm các đối tượng hoặc phân tích sâu hơn về các đối tượng phân tách.

Các bước để phân đoạn hình ảnh bằng thuật toán Watershed:

- *Bước 1:* Tìm nền bằng cách sử dụng xử lý hình thái học như Opening và Closing.
- *Bước 2:* Tìm tiền cảnh bằng cách sử dụng biến đổi khoảng cách.
- *Bước 3:* Khu vực không xác định là khu vực không nằm ở tiền cảnh và hậu cảnh, được sử dụng làm điểm đánh dấu cho thuật toán Watershed.

2.3.3 Phân vùng dựa trên kỹ thuật phân cụm

Phân cụm (Clustering) là một loại thuật toán học máy không giám sát, được sử dụng phổ biến trong phân vùng ảnh. Một trong những thuật toán Clustering thường được ứng dụng cho tác vụ phân vùng ảnh là KMeans Clustering. Loại phân cụm này có thể được sử dụng để tạo các phân đoạn trong một hình ảnh có màu.

2.3.3.1 Phân cụm Fuzzy C-Means

Fuzzy c-means (FCM) là một kỹ thuật phân cụm dữ liệu trong đó một tập dữ liệu được nhóm thành *các cụm* N với mọi điểm dữ liệu trong tập dữ liệu thuộc về mọi cụm ở một mức độ nhất định.

Phân cụm FCM hoạt động bằng cách gán thành viên cho từng điểm dữ liệu tương ứng với từng trung tâm cụm trên cơ sở khoảng cách giữa tâm cụm và điểm dữ liệu. Thêm dữ liệu gần trung tâm cụm nhiều hơn là thành viên đối với trung tâm cụm cụ thể. Rõ ràng, tổng hợp tư cách thành viên của mỗi điểm dữ liệu nên được bằng một. Sau mỗi lần lặp lại, các trung tâm thành viên và cụm được cập nhật theo công thức: [9]

$$\mu_{ij} = 1 / \sum_{k=1}^c (d_{ij} / d_{ik})^{(2/m-1)}$$
$$v_j = \left(\sum_{i=1}^n (\mu_{ij})^m x_i \right) / \left(\sum_{i=1}^n (\mu_{ij})^m \right), \forall j = 1, 2, \dots, c$$

Trong đó:

- n là số điểm dữ liệu.
- v_j đại diện cho i^{th} cụm trung tâm.
- m là chỉ số mờ $m \in [1, \infty]$.
- c số lượng trung tâm cụm.
- μ_{ij} đại diện cho các thành viên của i^{th} dữ liệu về j^{th} cụm trung tâm.
- d_{ij} đại diện cho khoảng cách Euclidean giữa i^{th} dữ liệu và j^{th} cụm trung tâm.

Mục tiêu chính của thuật toán c-means mờ là giảm thiểu:

$$J(U, V) = \sum_{i=1}^n \sum_{j=1}^c (\mu_{ij})^m \|x_i - v_j\|^2$$

$\|x_i - v_j\|$ là khoảng cách Euclide giữa i^{th} dữ liệu và j^{th} cụm trung tâm.

Các bước thuật toán cho phân cụm Fuzzy c-means

Cho $x = \{x_1, x_2, x_3, \dots, x_n\}$ là tập hợp các điểm dữ liệu và $V = \{v_1, v_2, v_3, \dots, v_n\}$ là tập hợp các trung tâm.

- *Bước 1*: Chọn ngẫu nhiên các trung tâm cụm ' c '.

- *Bước 2*: Tính toán tư cách thành viên mờ ' μ_{ij} ' sử dụng:

$$\mu_{ij} = 1 / \sum_{k=1}^c (d_{ij} / d_{ik})^{(2/m-1)}$$

- *Bước 3*: Tính toán các trung tâm mờ ' v_j ' sử dụng:

$$v_j = \left(\sum_{i=1}^n (\mu_{ij})^m x_i \right) / \left(\sum_{i=1}^n (\mu_{ij})^m \right), \forall j = 1, 2, \dots, c$$

- *Bước 4*: Lặp lại bước 2 và 3 cho đến khi đạt được giá trị ' J ' tối thiểu hoặc $\|U^{(k+1)} - U^{(k)}\| < \beta$.

Trong đó:

- k là bước lặp lại.
- β là tiêu chí chấm dứt giữa $[0, 1]$.
- $(\mu_{ij})_{N \times C}$ là ma trận thành viên mờ.
- J là chức năng khách quan.

Ưu điểm

- Cho kết quả tốt nhất cho tập dữ liệu chồng chéo và tương đối tốt hơn sau đó thuật toán k-means.
- Không giống như k-means trong đó điểm dữ liệu phải chỉ thuộc về một trung tâm cụm, ở đây điểm dữ liệu được gán tư cách thành viên của mỗi trung tâm cụm do đó điểm dữ liệu có thể thuộc về nhiều hơn một trung tâm cụm.

Nhược điểm

- Với giá trị β thấp hơn, nhận được kết quả tốt hơn nhưng phải trả giá bằng số lần lặp lại nhiều hơn.
- Các biện pháp đo khoảng cách Euclide có thể có trọng lượng không đồng đều các yếu tố cơ bản.

2.3.3.2 Phân cụm *K-means*

Thuật toán **K-means** là một thuật toán lặp đi lặp lại cố gắng phân vùng tập dữ liệu thành các nhóm con (cụm) không chồng chéo riêng biệt được xác định trước trong đó mỗi điểm dữ liệu chỉ thuộc về **một nhóm**. Nó cố gắng làm cho các điểm dữ liệu trong cụm giống nhau nhất có thể trong khi vẫn giữ cho các cụm khác nhau (xa) nhất có thể. Nó gán

các điểm dữ liệu cho một cụm sao cho tổng khoảng cách bình phương giữa các điểm dữ liệu và centroid của cụm sao (trung bình số học của tất cả các điểm dữ liệu thuộc cụm đó) ở mức tối thiểu. Chúng ta càng có ít biến thể trong các cụm, các điểm dữ liệu càng đồng nhất (tương tự) trong cùng một cụm [10].

Cách thức hoạt động của thuật toán kmeans như sau:

Chỉ định số cụm K.

Bước 1: Khởi tạo centroids bằng cách xáo trộn tập dữ liệu trước tiên và sau đó chọn ngẫu nhiên các điểm dữ liệu K cho centroids mà không cần thay thế.

Bước 2: Tiếp tục lặp lại cho đến khi không có thay đổi nào đối với centroids. tức là việc gán điểm dữ liệu cho các cụm không thay đổi.

- Tính tổng khoảng cách bình phương giữa các điểm dữ liệu và tất cả các centroids.
- Gán từng điểm dữ liệu cho cụm gần nhất (centroid).
- Tính toán centroids cho các cụm bằng cách lấy giá trị trung bình của tất cả các điểm dữ liệu thuộc về mỗi cụm..

Cho một tập hợp các quan sát x_1, x_2, \dots, x_n , trong đó mỗi quan sát là một vectơ thực d-chiều, k-means clustering nhằm mục đích phân chia n quan sát thành ($k \leq n$) đặt $S = \{S_1, S_2, \dots, S_K\}$ để giảm thiểu tổng bình phương trong cụm (phương sai).

$$\arg \min_S \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg \min_S \sum_{i=1}^k |S_i| \text{Var} S_i$$

Trong đó μ_i là giá trị trung bình của các điểm trong S_i . Điều này tương đương với việc giảm thiểu độ lệch bình phương cặp của các điểm trong cùng một cụm:

$$\arg \min_S \sum_{i=1}^k \frac{1}{|S_i|} \sum_{x,y \in S_i} \|x - y\|^2$$

Ưu điểm

- **Đơn giản:** Dễ dàng triển khai k-means và xác định các nhóm dữ liệu chưa biết từ các tập dữ liệu phức tạp. Các kết quả được trình bày một cách dễ dàng và đơn giản.

- **Linh hoạt:** Thuật toán K-means có thể dễ dàng điều chỉnh theo những thay đổi. Nếu có bất kỳ vấn đề nào, việc điều chỉnh phân đoạn cụm sẽ cho phép các thay đổi dễ dàng xảy ra trên thuật toán.

- **Thích hợp trong** một tập dữ liệu lớn: K-means phù hợp với một số lượng lớn các bộ dữ liệu và nó được tính toán nhanh hơn nhiều so với tập dữ liệu nhỏ hơn. Nó cũng có thể tạo ra các cụm cao hơn.

- **Hiệu quả:** Thuật toán được sử dụng rất tốt trong việc phân đoạn tập dữ liệu lớn. Hiệu quả của nó phụ thuộc vào hình dạng của các cụm. K-có nghĩa là hoạt động tốt trong các cụm siêu hình cầu.

Nhược điểm

- **Không tập hợp các cụm không tối ưu:** K-means không cho phép phát triển một tập hợp các cụm tối ưu và để có kết quả hiệu quả, bạn nên quyết định các cụm trước đó.

- **Thiếu tính nhất quán:** Phân cụm K-means cho kết quả khác nhau trên các lần chạy khác nhau của một thuật toán. Lựa chọn ngẫu nhiên các mẫu cụm sẽ mang lại các kết quả phân cụm khác nhau dẫn đến sự không nhất quán.

- **Hiệu ứng đồng nhất:** Nó tạo ra các cụm với kích thước đồng nhất ngay cả khi dữ liệu đầu vào có kích thước khác nhau.

- **Độ nhạy với tỷ lệ:** Thay đổi hoặc thay đổi quy mô tập dữ liệu thông qua chuẩn hóa hoặc tiêu chuẩn hóa sẽ thay đổi hoàn toàn kết quả cuối cùng.

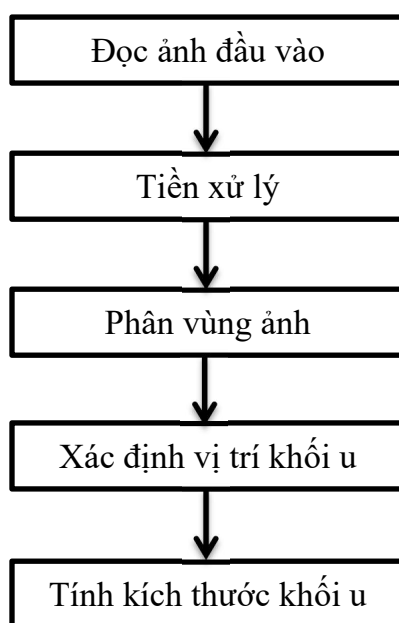
CHƯƠNG 3: TÍNH TOÁN VÀ THIẾT KẾ

3.1 GIỚI THIỆU

Để đáp ứng được yêu cầu đã đặt ra trước đó của đề tài “Xây dựng thuật toán tăng cường biên ảnh để tính kích thước khối u”, phần này sẽ tiến hành thực hiện các nội dung về tính toán và thiết kế hệ thống bao gồm thiết kế sơ đồ khối hệ thống, thiết kế khối đọc ảnh đầu vào, khối tiền xử lý, khối phân đoạn ảnh, khối xác định vị trí khối u, khối tính kích thước khối u,... Đồng thời, chương này sẽ trình bày về thiết kế giao diện GUI trên Matlab.

3.2 QUY TRÌNH XÁC ĐỊNH KÍCH THƯỚC KHỐI U NẪO

Để xác định kích thước khối u nảo, ta thực hiện các công việc cụ thể trong sơ đồ khối sau:



Hình 3.1 Sơ đồ khối quy trình xác định kích thước khối u

Nhiệm vụ các khối như sau:

- **Khối đọc ảnh đầu vào:** Ảnh đầu vào có nhiều định dạng file ảnh, nhiệm vụ của khối này là đọc được nhiều định dạng ảnh khác nhau.

- **Khối tiền xử lý:** Nhiệm vụ khối tiền xử lý là lọc ảnh, tăng cường ảnh. Khối này làm cho ảnh rõ hơn và cho biết các đặc tính của ảnh đầu vào.

- **Khối phân đoạn ảnh:** Nhiệm vụ khối phân đoạn ảnh là đơn giản hóa ảnh, dán nhãn cho từng pixel, tất cả các pixel thuộc cùng một tính chất sẽ có chung một nhãn.

- **Khối xác định vị trí khối u:** Nhiệm vụ khối xác định vị trí khối u là xác định tọa độ tâm của khối u và từ đó chỉ rõ vị trí khối u ở chỗ nào trên ảnh.

- **Khối tính kích thước của khối u:** Nhiệm vụ khối này là khối u sẽ được tính kích thước gồm diện tích và chu vi. Thông số kích thước khối u sẽ hỗ trợ cho việc xác định giai đoạn ung thư hoặc hỗ trợ cho các phương pháp điều trị thích hợp.

3.3 ĐỌC ẢNH ĐẦU VÀO

Các file ảnh có định dạng ảnh đầu vào như: **gif, png, bmp, tiff, pgm**, mỗi định dạng được cấu trúc bằng một ma trận điểm ảnh có kích thước khác nhau, để có thể đọc được nhiều định dạng ảnh khác nhau, cần xây dựng một hàm để có thể hiểu nhiều định dạng ảnh. Người sử dụng chọn file ảnh cần dùng tên của ảnh cần có đuôi ảnh kèm theo, nếu ảnh sử dụng là một trong các đuôi ảnh như trên. Trong phần mô phỏng, ảnh y tế được sử dụng là ảnh MRI có kích thước 512x512, định dạng **png**.

3.4 TIỀN XỬ LÝ

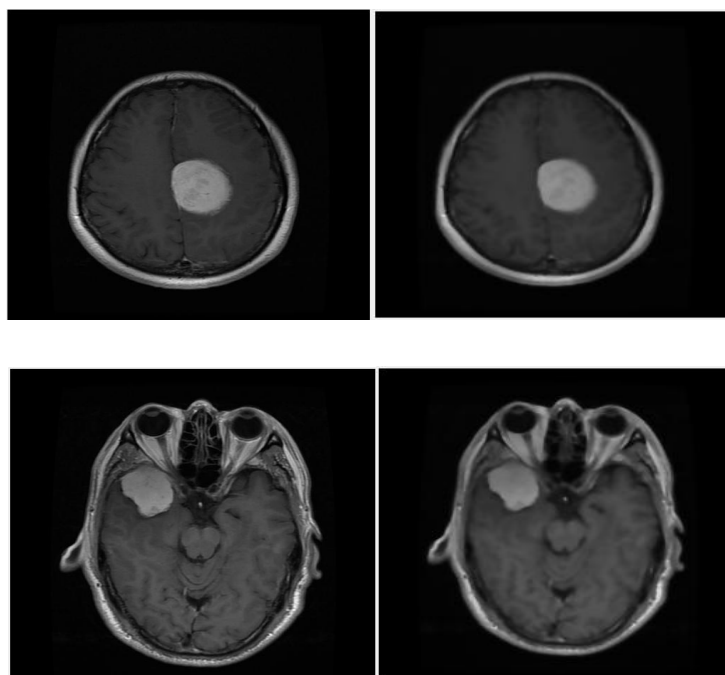
Trước khi tìm biên ảnh để tính kích thước khối u thì ảnh phải được khối tiền xử lý để ảnh đầu vào tốt hơn, dễ dàng quan sát hơn. Trong khối tiền xử lý, cần thực hiện lọc nhiễu và tăng cường ảnh.

3.4.1 Tách ảnh màu thành ảnh xám

Trong xử lý ảnh, việc chuyển đổi ảnh màu sang ảnh xám là công việc vô cùng phổ biến. Ảnh màu thực chất chỉ là tập hợp của những ma trận số có cùng kích thước. Khi muốn xử lý thông tin trên ảnh, sẽ dễ dàng hơn nếu ta chỉ xử lý dữ liệu trên một ma trận số thay vì nhiều ma trận số. Việc biến đổi ảnh màu về ảnh số (Grayscale converting) xuất hiện vì mục đích trên - biến đổi thông tin ảnh về một ma trận số hai chiều duy nhất.

3.4.2 Lọc nhiễu

Trong đề án này, sử dụng hàm Anisotropic để lọc nhiễu ảnh. Sau khi xử lý nhiễu, lấy ảnh sau khi đã tách ảnh I_1 và ảnh sau khi lọc nhiễu I_2 cộng lại với nhau được ảnh I . Sau khi tách ảnh màu thành ảnh xám, xử lý và tăng cường ảnh ta được kết quả ảnh sau khi kết thúc bước tiền xử lý như sau. Ảnh sau xử lý rõ hơn và dễ quan sát hơn, giúp ta xác định tốt hơn vị trí khối u.



Hình 3.2 Ảnh trước và sau khi được tiền xử lý

Chương trình hàm lọc nhiễu Anisotropic:

```
function diff_im = anisodiff(im, num_iter, delta_t, kappa, option)
fprintf('Removing noise\n');
fprintf('Filtering Completed !!');

% Convert input image to double.
im = double(im);

% PDE (partial differential equation) initial condition.
diff_im = im;

% Center pixel distances.
dx = 1;
dy = 1;
dd = sqrt(2);

% 2D convolution masks - finite differences.
hN = [0 1 0; 0 -1 0; 0 0 0];
hS = [0 0 0; 0 -1 0; 0 1 0];
hE = [0 0 0; 0 -1 1; 0 0 0];
hW = [0 0 0; 1 -1 0; 0 0 0];
hNE = [0 0 1; 0 -1 0; 0 0 0];
hSE = [0 0 0; 0 -1 0; 0 0 1];
hSW = [0 0 0; 0 -1 0; 1 0 0];
hNW = [1 0 0; 0 -1 0; 0 0 0];

% Anisotropic diffusion.
for t = 1:num_iter
```

```
% Finite differences. [imfilter(.,,'conv') can be replaced by conv2(.,,'same')]
nablaN = imfilter(diff_im,hN,'conv');
nablaS = imfilter(diff_im,hS,'conv');
nablaW = imfilter(diff_im,hW,'conv');
nablaE = imfilter(diff_im,hE,'conv');
nablaNE = imfilter(diff_im,hNE,'conv');
nablaSE = imfilter(diff_im,hSE,'conv');
nablaSW = imfilter(diff_im,hSW,'conv');
nablaNW = imfilter(diff_im,hNW,'conv');

% Diffusion function.
if option == 1
    cN = exp(-(nablaN/kappa).^2);
    cS = exp(-(nablaS/kappa).^2);
    cW = exp(-(nablaW/kappa).^2);
    cE = exp(-(nablaE/kappa).^2);
    cNE = exp(-(nablaNE/kappa).^2);
    cSE = exp(-(nablaSE/kappa).^2);
    cSW = exp(-(nablaSW/kappa).^2);
    cNW = exp(-(nablaNW/kappa).^2);

elseif option == 2
    cN = 1./(1 + (nablaN/kappa).^2);
    cS = 1./(1 + (nablaS/kappa).^2);
    cW = 1./(1 + (nablaW/kappa).^2);
    cE = 1./(1 + (nablaE/kappa).^2);
    cNE = 1./(1 + (nablaNE/kappa).^2);
    cSE = 1./(1 + (nablaSE/kappa).^2);
    cSW = 1./(1 + (nablaSW/kappa).^2);
    cNW = 1./(1 + (nablaNW/kappa).^2);
```

```
end

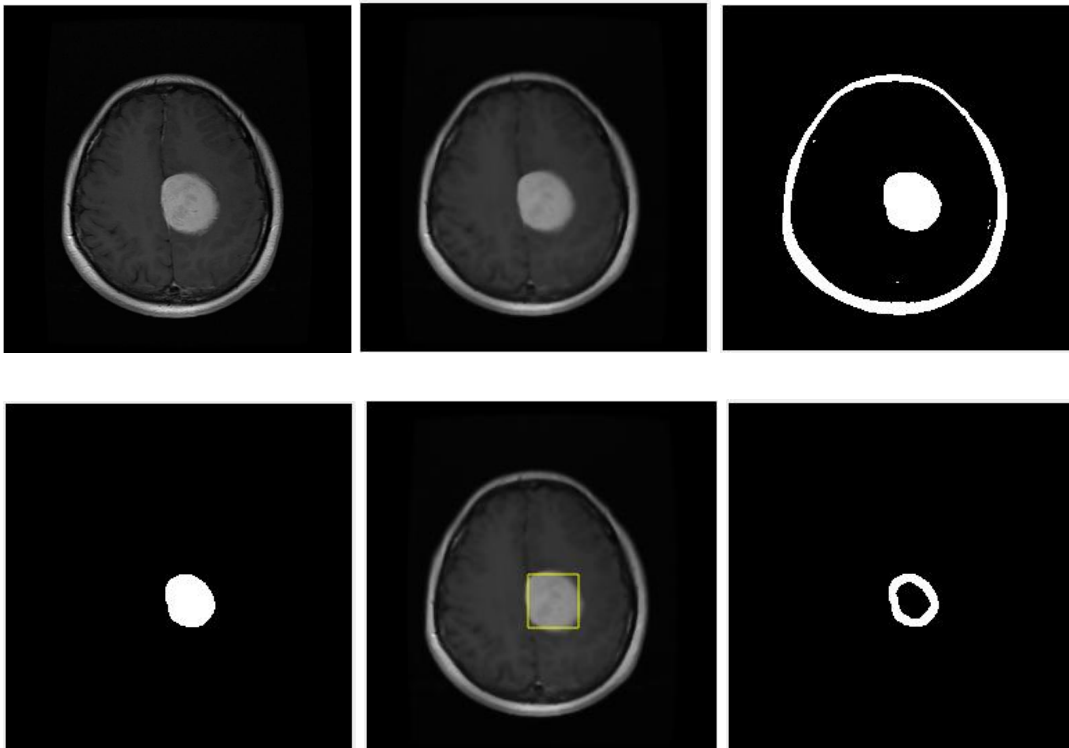
% Discrete PDE solution.
diff_im = diff_im + ...
    delta_t*(...
        (1/(dy^2))*cN.*nablaN + (1/(dy^2))*cS.*nablaS + ...
        (1/(dx^2))*cW.*nablaW + (1/(dx^2))*cE.*nablaE + ...
        (1/(dd^2))*cNE.*nablaNE + (1/(dd^2))*cSE.*nablaSE + ...
        (1/(dd^2))*cSW.*nablaSW + (1/(dd^2))*cNW.*nablaNW );
end
```

3.5 PHÂN VÙNG ẢNH

Sau khi ảnh được tiền xử lý, ta sẽ tiến hành khối phân vùng ảnh để gán nhãn từng vùng ảnh để xác định nổi bật đối tượng (khối u). Trong khối này, ta sử dụng một số phương pháp để so sánh kết quả phân vùng: Otsu, Watershed, K – Means, Fuzzy C Means sẽ được thảo luận dưới đây.

3.5.1 Thuật toán ngưỡng Otsu

Thuật toán Otsu's là tên một nhà nghiên cứu người Nhật đã nghĩ ra ý tưởng cho việc tính ngưỡng một cách tự động dựa vào giá trị điểm ảnh của ảnh đầu vào nhằm thay thế cho việc sử dụng ngưỡng cố định. Ngưỡng Otsu's là phương pháp chuyển đổi ảnh xám sang ảnh nhị phân bằng cách giả sử rằng hình ảnh chỉ có 2 loại pixel, một là các pixel đối tượng và một là các pixel nền. Sau đó, sẽ thực hiện xử lý hình thái học, cụ thể là thực hiện phép Openning – thực hiện phép co rồi giãn với một cấu trúc giúp làm mượt các đường viền và loại bỏ các đối tượng nhỏ. Ta thu được kết quả ảnh được phân vùng dựa trên ngưỡng Otsu.



Hình 3.3 Phân vùng khối u não dùng thuật toán Otsu

Chương trình phân vùng ảnh thuật toán Otsu:

```
clc
close all
clear all

%% Input
[I,path]=uigetfile('*.png','select a input image');
str = strcat(path,I);
I = imread(str);

figure;
imshow(I);
title('Input image','FontSize',20);
```



```
%% Filtered Image
num_iter = 10;
delta_t = 1/7;
kappa = 15;
option = 2;

inp = anisodiff(I,num_iter,delta_t,kappa,option);
inp = uint8(inp);
inp = imresize(inp,[256,256]);
if size(inp,3)>1
    inp = rgb2gray(inp);
end
figure;
imshow(inp);

%% Otsu Segementation
im_thr = imtophat(inp,strel('disk',40));
im_adjust = imadjust(im_thr);
level = graythresh(im_adjust);
figure;
imshow(level);
BW = im2bw(im_adjust,level);
figure;
imshow(BW);
SE_erode=strel('disk',3);
im_erode=imerode(BW,SE_erode);
figure;
imshow(im_erode);
```

```
%% Morphological Operation
SE=ones(3);
im_erode=imopen(im_erode,SE);
imshow(im_erode);
title('Brain Tumor with other small unwanted objects');
im_erode = bwareaopen(im_erode,500);
figure;
imshow(im_erode)
title('Segmented Brain Tumor')

%% Location of Brain tumor
stats=regionprops(im_erode,'Solidity','Area','BoundingBox');
density=[stats.Solidity];
area=[stats.Area];
High_Density_Area=density > 0.5;
MaxArea=max(area(High_Density_Area));
tumor_label=find(area==MaxArea);
tumor=ismember(im_erode,tumor_label);

%Bounding Box
box = stats(tumor_label);
wantedBox = box.BoundingBox;
figure
imshow(inp);
title('Bounding Box','FontSize',20);
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;
```

```
dilationAmount = 5;
rad = floor(dilationAmount);
[r,c] = size(tumor);
filledImage = imfill(tumor, 'holes');

for i=1:r
    for j=1:c
        x1=i-rad;
        x2=i+rad;
        y1=j-rad;
        y2=j+rad;
        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end

tumorOutline = tumor - erodedImage;
figure;
imshow(tumorOutline);
```

```
%% Area and Perimeter of Brain tumor
measurements = regionprops(tumor, ...
    'area', 'Centroid', 'Perimeter');
area_tumor = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];

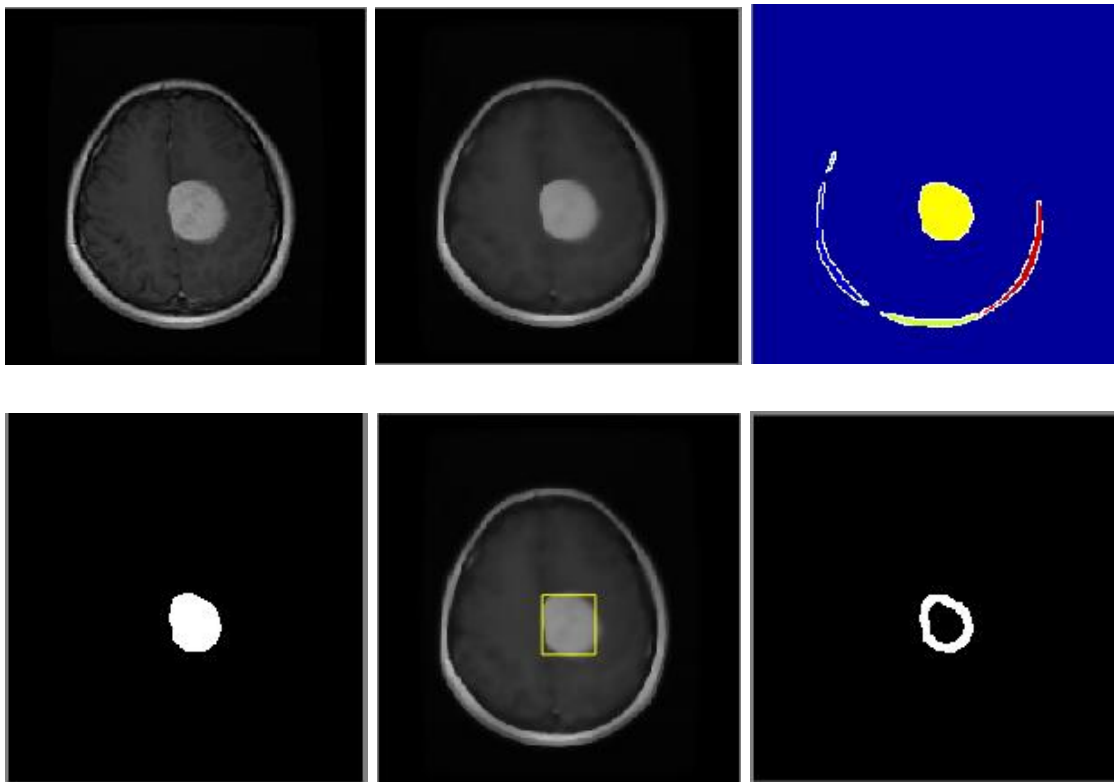
numberOfWhitePixel = bwarea(tumor);
area_tumor = numberOfWhitePixel * (0.26458333)^2;
perimeter_tumor = perimeter * 0.26458333;

structBoundaries = bwboundaries(tumor);
xy=structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.
```

3.5.2 Thuật toán biến đổi Watershed

Thuật toán Watershed là một thuật toán rất hữu ích, có mục đích tìm ra các đường phân cách các vùng trên ảnh, nhóm các vùng ảnh có cùng các thuộc tính liên quan lại thành một vùng đồng nhất.

Trước tiên, ta thực hiện xử lý hình thái học cụ thể là phép Opening, sau đó tăng cường ảnh, chuyển sang ảnh nhị phân. Sau đó ta tiến hành biến đổi Watershed, kết quả ảnh được trả về một ma trận được dán nhãn bao gồm các giá trị dương dọc theo các vùng. Sau đó, để có ảnh được phân vùng ảnh cuối cùng, ma trận được dán nhãn đã được chuyển đổi thành hình ảnh RGB.



Hình 3.4 Phân vùng ảnh u não dùng thuật toán Watershed

Chương trình phân vùng ảnh thuật toán Watershed:

```
clc
close all
clear all

%% Input Image
[filename,path]=uigetfile('*.png','select a input image');
str = strcat(path,filename);
brainImg = imread(str);

figure;
imshow(brainImg);
title('Input image','FontSize',20);
```

```
%% Filtered Image
num_iter = 10;
delta_t = 1/7;
kappa = 15;
option = 2;

inp = anisodiff(brainImg,num_iter,delta_t,kappa,option);
inp = uint8(inp);

inp=imresize(inp,[256,256]);
if size(inp,3)>1
    inp=rgb2gray(inp);
end
imshow(inp);

%% thresholding
sout=imresize(inp,[256,256]);
t0=15;
th=t0+((max(inp(:))+min(inp(:)))/2);
for i=1:size(inp,1)
    for j=1:size(inp,2)
        if inp(i,j)>th
            sout(i,j)=1;
        else
            sout(i,j)=0;
        end
    end
end

%% Watershed segmentation
```

```
hy = fspecial('sobel');
hx = hy';
Iy = imfilter(double(sout), hy, 'replicate');
Ix = imfilter(double(sout), hx, 'replicate');
gradmag = sqrt(Ix.^2 + Iy.^2);
L = watershed(gradmag);
Lrgb = label2rgb(L);
imshow(Lrgb);

%% Location of Brain tumor
label=bwlabel(sout);
stats=regionprops(logical(sout),'Solidity','Area','BoundingBox');
density=[stats.Solidity];
area=[stats.Area];
high_dense_area = density>0.6;
max_area=max(area(high_dense_area));
tumor_label=find(area==max_area);
tumor = ismember(label,tumor_label);

% Bounding Box
box = stats(tumor_label);
wantedBox = box.BoundingBox;
imshow(inp);
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;

dilationAmount = 5;
rad = floor(dilationAmount);
[r,c] = size(tumor);
```

```
filledImage = imfill(tumor, 'holes');

for i=1:r
    for j=1:c
        x1=i-rad;
        x2=i+rad;
        y1=j-rad;
        y2=j+rad;
        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end

tumorOutline=tumor - erodedImage;
imshow(tumorOutline);

%% Area and Perimeter of Brain tumor
measurements = regionprops(tumor, ...
    'area', 'Centroid', 'Perimeter');
```



```
area = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];

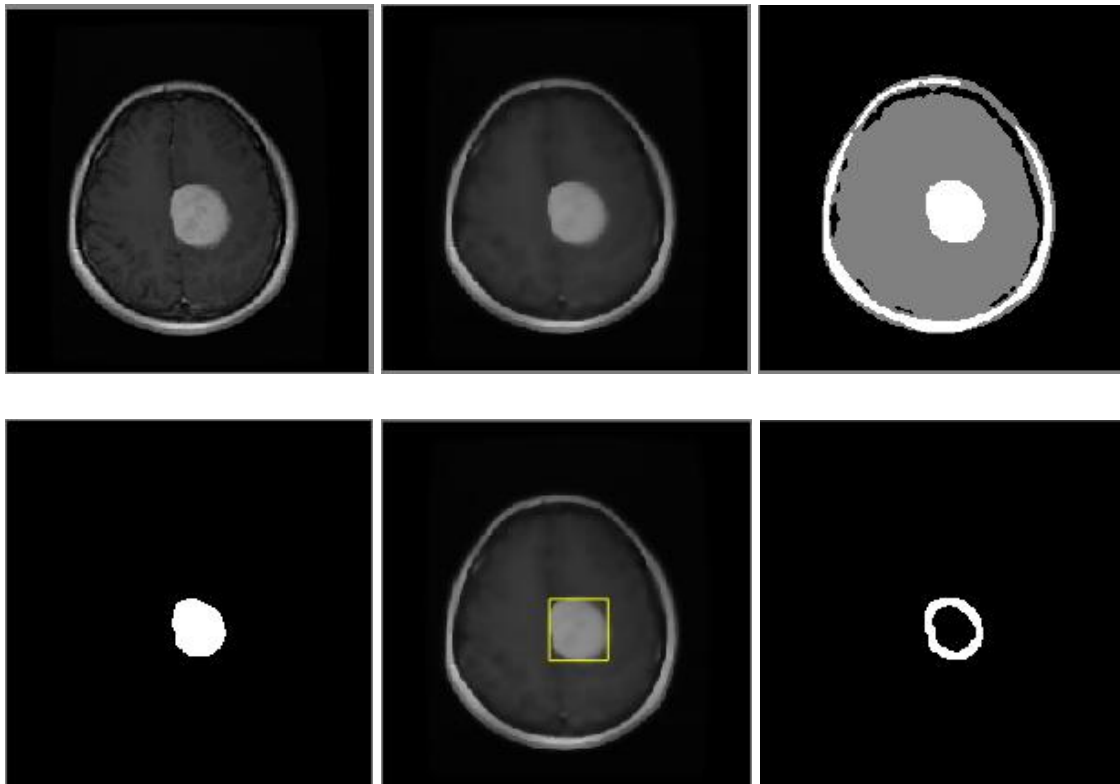
numberOfPixels2 = bwarea(tumor);
area=numberOfPixels2*(0.26458333)^2;
perimeter1=perimeter*0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(tumor);
xy=structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.
```

3.5.3 Thuật toán phân cụm K – Means

Trong trường hợp phân cụm K – Means, tập hợp ban đầu của các cụm (K) cần được xác định vì quá trình phân cụm là phụ thuộc vào giá trị của K hoặc số cụm. Trong nghiên cứu này, giá trị của K được chọn là 4 vì bộ não con người có thể được nhóm thành 4 cụm. Trong đó, báo cáo này chú ý tới khối u được phân cụm là 3.

Lúc đầu, hình ảnh được chuyển thành không gian tuyến tính, sau đó thuật toán K – Means đã được áp dụng, tạo ra một tập hợp của cụm và trung tâm cụm. Hình ảnh không gian tuyến tính sau đó được chuyển đổi lại thành miền không gian. Sau đó, khối u được trích xuất từ cụm 3 bằng cách xử lý hình thái học tương tự như ngưỡng Otsu.



Hình 3.5 Phân vùng ảnh u não dùng thuật toán K - Means

Chương trình phân vùng ảnh thuật toán K – Means:

```
clc
close all
clear all

%% Input Image
[I,path]=uigetfile('*.png','select a input image');
str=strcat(path,I);
I=imread(str);

figure;
imshow(I);
title('Input image','FontSize',20);
```

```
%% Grayscale conversion
num_iter = 10;
delta_t = 1/7;
kappa = 15;
option = 2;
disp('Preprocessing image please wait . . .');
inp = anisodiff_function(I,num_iter,delta_t,kappa,option);
inp = uint8(inp);

inp=imresize(inp,[256,256]);

if size(inp,3)>1
    inp=rgb2gray(inp);
end
figure;
imshow(inp);
title('Filtered image','FontSize',20);

%% Segmentation USING K-means Clustering
Idata=reshape(inp, [],1);
Idata=double(Idata);

[Idx, nn]=kmeans(Idata,4);
Imsame=reshape(Idx,size(inp));
figure;
imshow(Imsame, [])
imshow(Imsame==2, []); % Tumor in this cluster
figure;
subplot(2,2,1); imshow(Imsame==1, []);
subplot(2,2,2); imshow(Imsame==2, []);
```

```
subplot(2,2,3); imshow(Imsame==3, []);
subplot(2,2,4); imshow(Imsame==4, []);

bw = (Imsame == 2);
SE=ones(5); % small
%SE=ones(15); % big
bw=imopen(bw,SE);
figure;
imshow(bw);
title('Brain Tumor with other small unwanted objects');
bw = bwareaopen(bw,400);
figure;
imshow(bw)
title('Segmented Brain Tumor')

%% Location of Brain tumor
stats = regionprops(bw,'Solidity','Area','BoundingBox');
density=[stats.Solidity];
area=[stats.Area];
High_Density_Area=density > 0.6; % reduce to detect small or early stage tumors
MaxArea = max(area(High_Density_Area));
tumor_label = find(area==MaxArea);
tumor = ismember(bw,tumor_label);

% Bounding Box
box = stats(tumor_label);
wantedBox = box.BoundingBox;
figure
imshow(inp);
title('Bounding Box','FontSize',20);
```

```
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;

dilationAmount = 5;
rad = floor(dilationAmount);
[r,c] = size(tumor);
filledImage = imfill(tumor, 'holes');

for i=1:r
    for j=1:c
        x1=i-rad;
        x2=i+rad;
        y1=j-rad;
        y2=j+rad;
        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end
```

```
end
tumorOutline = tumor - erodedImage;
figure;
imshow(tumorOutline);

%% Area and Perimeter of Brain tumor
measurements = regionprops(tumor, ...
    'Area', 'Centroid', 'Perimeter');

area = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];

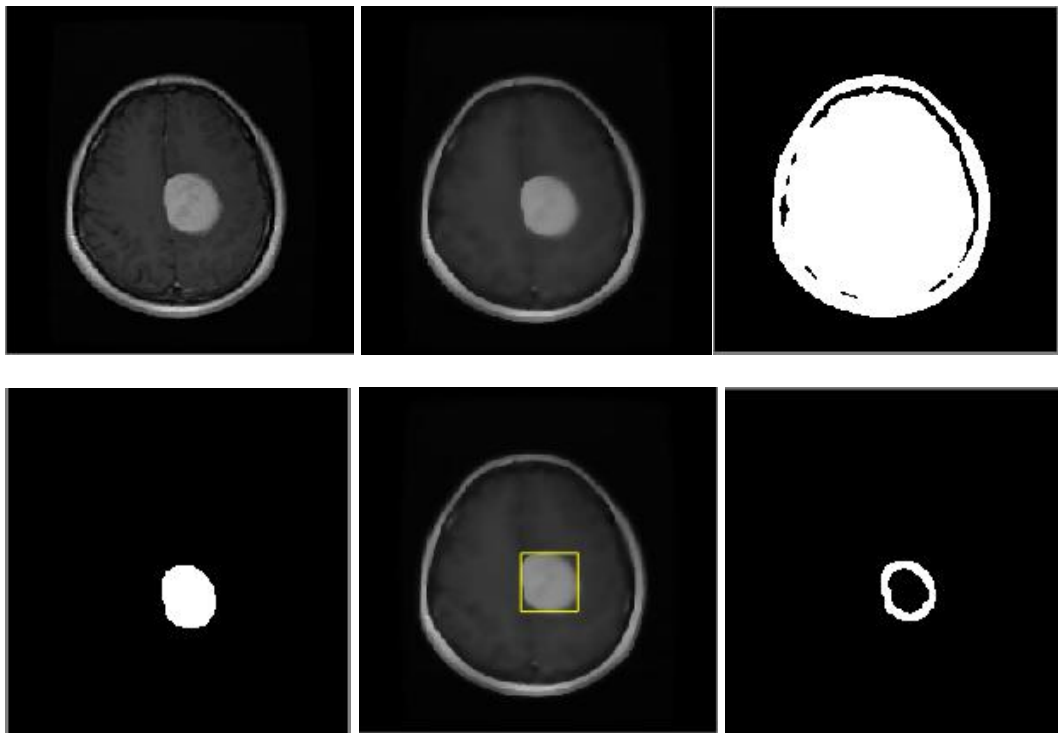
% convert into mm
numberOfWhitePixels = bwarea(tumor);
area_tumor = numberOfWhitePixels * (0.26458333)^2;
perimeter_tumor = perimeter * 0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(tumor);
xy = structBoundaries{1};
x = xy(:, 2);
y = xy(:, 1);
```

3.5.4 Phương pháp Phân cụm Fuzzy C Means

Trong trường hợp phân cụm Fuzzy C – Means, kỹ thuật này gom cụm một tập n vector đối tượng dữ liệu $X = \{x_1, x_2, \dots, x_n\} \subset R^s$ thành các nhóm mờ dựa trên tính toán tối thiểu hóa hàm mục tiêu để đo chất lượng của gom cụm và tìm trung tâm cụm trong mỗi nhóm, sao cho chi phí hàm độ đo độ phi tương tự là nhỏ nhất.

Lúc đầu, hình ảnh được tiền xử lý, sau đó thuật toán Fuzzy C – Means đã được áp dụng, tạo ra một tập hợp của cụm và trung tâm cụm. Qua các công việc tính toán và sắp xếp các điểm dữ liệu cùng độ thuộc lớn nhất của nó vào các cụm theo quy tắc xét độ thuộc của điểm dữ liệu đó với từng cụm, điểm dữ liệu sẽ thuộc vào cụm nào có độ thuộc lớn nhất, nếu có từ hai độ thuộc lớn nhất bằng nhau trở lên thì chọn một trong số các cụm đó để đưa vào. Thuật toán kết thúc. Cuối cùng, ảnh sẽ được xử lý hình thái học như các thuật toán trên để thu được ảnh khối u rõ nhất.



Hình 3.6 Phân vùng ảnh u não dùng thuật toán Fuzzy C Means

Hàm Fuzzy C Means:

```
function [bw,level]=fcmthresh(IM,sw)
%FCMTHRESH Thresholding by 3-class fuzzy c-means clustering
% [bw,level]=fcmthresh(IM,sw) outputs the binary image bw and threshold level of
% image IM using a 3-class fuzzy c-means clustering. It often works better
% than Otsu's method which outputs larger or smaller threshold on
% fluorescence images.
% sw is 0 or 1, a switch of cut-off position.
% sw=0, cut between the small and middle class
% sw=1, cut between the middle and large class
%
% Contributed by Guanglei Xiong (xgl99@mails.tsinghua.edu.cn)
% at Tsinghua University, Beijing, China.
% check the parameters
if (nargin<1)
    error('You must provide an image.');
```

```
elseif (nargin==1)
    sw=0;
elseif (sw~=0 && sw~=1)
    error('sw must be 0 or 1.');
```

```
end
data=reshape(IM,[],1);
[center,member]=fcm(data,3);
[center,cidx]=sort(center);
member=member';
member=member(:,cidx);
[maxmember,label]=max(member,[],2);
if sw==0
    level=(max(data(label==1))+min(data(label==2)))/2;
```



```
else
    level=(max(data(label==2))+min(data(label==3)))/2;
end
bw=im2bw(IM,level);
```

Chương trình phân vùng ảnh thuật toán *Fuzzy C Means*:

```
clc;
clear all;
close all;
%% Input
[I,path]=uigetfile('*.png','select a input image');
str=strcat(path,I);
im=imread(str);

figure;
imshow(im);
title('Input image','FontSize',20);

%% Filter Image
num_iter = 10;
delta_t = 1/7;
kappa = 15;
option = 2;

im = anisodiff(im,num_iter,delta_t,kappa,option);

im = uint8(im);

im = imresize(im,[256,256]);
```

```
if size(im,3)>1
    im = rgb2gray(im);
end
figure;
imshow(im);

fim = mat2gray(im);
level = graythresh(fim);
fim = imresize(fim,[256,256]);
bwfim = im2bw(fim,0.1);

%% Fuzzy C Means Segmentation
[bwfim0,level0]=fcmthresh(fim,0);
[bwfim1,level1]=fcmthresh(fim,1);
subplot(2,2,1);
imshow(fim);title('Original');
subplot(2,2,2);
imshow(bwfim);title(sprintf('Otsu,level=%f,level'));
subplot(2,2,3);
imshow(bwfim0);title(sprintf('FCM0,level=%f,level0'));
subplot(2,2,4);
imshow(bwfim1);title(sprintf('FCM1,level=%f,level1'));

%% Morphological Operation
SE = ones(15);
bwfim1 = imopen(bwfim1,SE);
imshow(bwfim1);
title('Brain Tumor with other small unwanted objects');
bwfim1 = bwareaopen(bwfim1,500);
figure;
```

```
imshow(bwfim1);
title('Segmented Brain Tumor');

%% Location of Brain tumor
stats = regionprops(bwfim1,'Solidity','Area','BoundingBox');
density = [stats.Solidity];
area = [stats.Area];
High_Density_Area = density > 0.6; % reduce to detect small or early stage tumors
MaxArea = max(area(High_Density_Area));
tumor_label = find(area==MaxArea);
tumor = ismember(bwfim1,tumor_label);

%% Bounding Box
box = stats(tumor_label);
wantedBox = box.BoundingBox;
figure
imshow(im);
title('Bounding Box','FontSize',20);
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;

dilationAmount = 5;
rad = floor(dilationAmount);
[r,c] = size(tumor);
filledImage = imfill(tumor, 'holes');

for i=1:r
    for j=1:c
        x1=i-rad;
```

```
x2=i+rad;
y1=j-rad;
y2=j+rad;
if x1<1
    x1=1;
end
if x2>r
    x2=r;
end
if y1<1
    y1=1;
end
if y2>c
    y2=c;
end
erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
end
end

tumorOutline = tumor - erodedImage;
figure;
imshow(tumorOutline);

%% Area and Perimeter of Brain tumor
measurements = regionprops(tumor, ...
    'Area', 'Centroid', 'Perimeter');

area = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];
```

```
numberOfWhitePixels = bwarea(tumor);

%convert into mm
area_tumor = numberOfWhitePixels * (0.26458333)^2;
perimeter_tumor = perimeter * 0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(tumor);
xy = structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.
```

3.5 XÁC ĐỊNH VỊ TRÍ KHỐI U

Sau khi đã phân vùng được khối u thông qua các thuật toán thu được như trên. Tiến hành xác định vị trí của khối u bằng hàm `regionprops()` trong Matlab.

```
stats = regionprops(BW,'Solidity','Area','BoundingBox');
```

Hàm `regionprops()` trả về các phép đo cho tập hợp các thuộc tính cho từng thành phần (đối tượng) được kết nối 8 trong ảnh nhị phân, BW. Có thể sử dụng `regionprops()` trên các vùng liền kề và các vùng không liền kề.

Tham số	Mô tả
Area	Số pixel thực tế trong vùng, được trả về dưới dạng vô hướng.
Solidity	Tỷ lệ pixel trong bao lồi cũng nằm trong vùng, được trả về dưới dạng vô hướng.
Bounding box	Vị trí và kích thước của box nhỏ nhất chứa vùng.
Centroid	Tâm của vùng, được trả về dưới dạng vectơ.
Perimeter	Chu vi của vùng, được bằng cách tính khoảng cách giữa mỗi cặp pixel liền kề xung quanh đường viền của vùng được trả về dưới dạng vô hướng.

3.5.1 Bouding Box

Chương trình xác định vị trí của khối u dùng Bounding Box:

```
% % Location of the Brain tumor
stats = regionprops(bwfim1,'Solidity','Area','BoundingBox');
density = [stats.Solidity];
area = [stats.Area];
High_Density_Area = density > 0.6;
max_area = max(area(High_Density_Area));
tumor_label = find(area == max_area);
tumor = ismember(bwfim1,tumor_label);

% Bounding Box
box = stats(tumor_label);
wantedBox = box.BoundingBox;
figure;
imshow(im);
title('Bounding Box','FontSize',20);
hold on;
rectangle('Position',wantedBox,'EdgeColor','y');
hold off;
```

3.5.2 Tọa độ tâm của khối u

Tương tự, ta cũng sử dụng hàm *regionprops* () để tìm tọa độ tâm của vùng khối u sau khi đã tìm được Bounding Box thông qua tham số ‘Centroid’.

```
%% Get coordinates of the boundary in tumor
measurements = regionprops(tumor, ...
    'Area', 'Centroid', 'Perimeter');
centroid = [measurements.Centroid];
structBoundaries = bwboundaries(tumor);
xy = structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.
```

3.6 TÍNH KÍCH THƯỚC KHỐI U

Sau khi đã thực hiện bước phân vùng ảnh, ta tiến hành tìm diện tích và chu vi khối u lần lượt bằng cách tính tổng số pixel màu trắng (*tumor*(1)) và tính khoảng cách giữa mỗi cặp pixel liên kề xung quanh đường viền của vùng. Công thức tính kích thước như sau:

$$\text{Area_tumor} = \sum_{w=0}^{256} \sum_{H=0}^{256} \text{tumor}(1) * P^2 \text{ (mm}^2\text{)}$$
$$\text{Perimeter_tumor} = \sum_{w=0}^{256} \sum_{H=0}^{256} \text{tumor}(1) * P \text{ (mm)}$$

Trong đó:

- W và H là kích thước 256x256 của ảnh khối u.
- *tumor*(1): Pixel màu trắng của vùng khối u.
- P = 0.26458333 mm (ta có, 1 pixel = 0.26458333 mm)

Trong Matlab, có hỗ trợ tính diện tích và chu vi của 1 một vùng ảnh được xác định - hàm *regionprops()* được giới thiệu ở mục **3.5** sẽ hỗ trợ tính diện tích và chu vi của vùng khối u muốn tìm.

Chương trình tính diện tích và chu vi của khối u:

```
%% Area and Perimeter of Brain tumor
measurements = regionprops(tumor, ...
    'Area', 'Centroid', 'Perimeter');

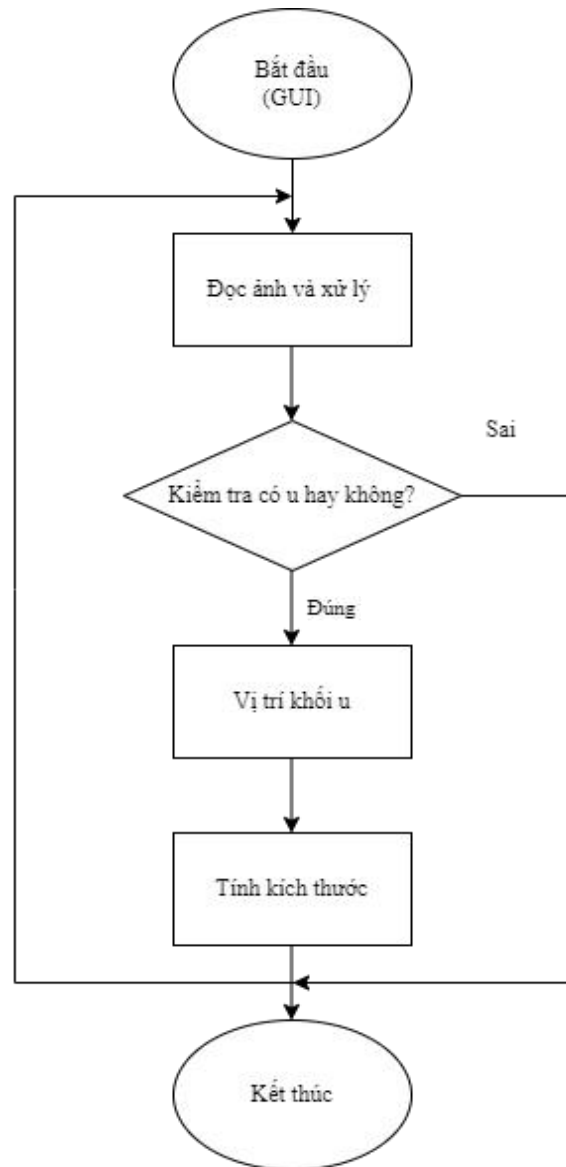
area = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];
% Convert into mm
numberofWhitePixels = bwarea(tumor);
area_tumor = numberofWhitePixels * (0.26458333)^2;
perimeter_tumor = perimeter * 0.26458333;
```

3.7 THIẾT KẾ GIAO DIỆN GUI

3.7.1 Giới thiệu giao diện GUI trên Matlab

GUI được viết tắt của từ “Graphical User Interface” là giao diện người dùng đồ họa. Trong Matlab thì GUI hỗ trợ đầy đủ các chương trình để bạn thực hiện. Như là tính toán với phép toán LOGIC, lập trình không gian 2D, 3D, đọc dữ liệu từ Excel, xử lý hình ảnh,... Việc đó được thực hiện thông qua hàm xây dựng sẵn là CALLBACK.

3.7.2 Lưu đồ giải thuật



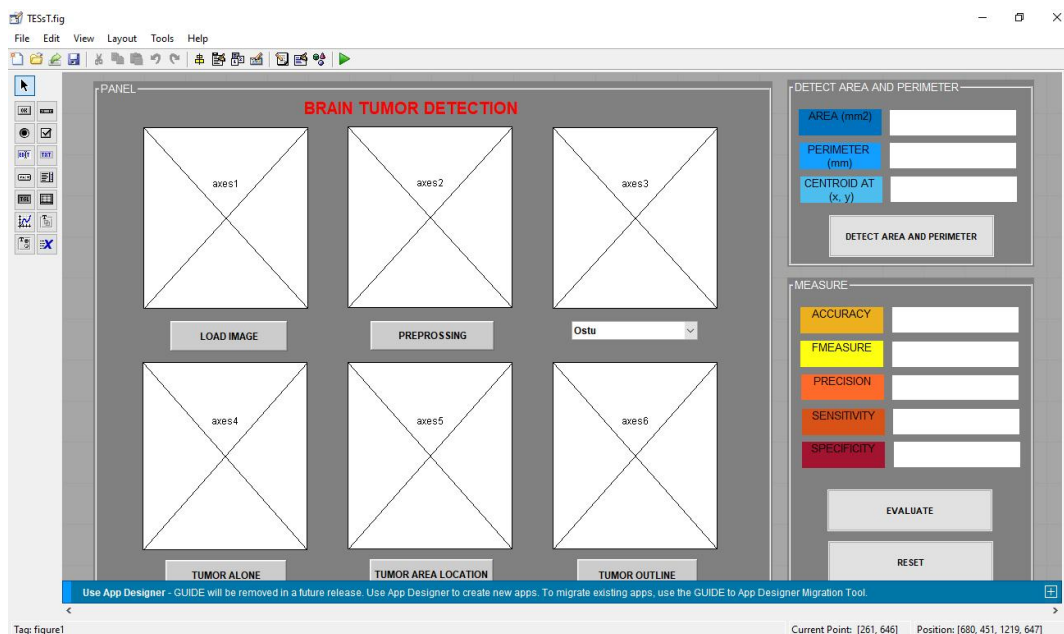
Hình 3.7 Lưu đồ viết chương trình giao diện GUI trên Matlab

Chương trình giao diện GUI:

Toàn bộ chương trình giao diện GUI được viết trong phần **PHỤ LỤC**.

3.7.3 Tài liệu hướng dẫn sử dụng, thao tác

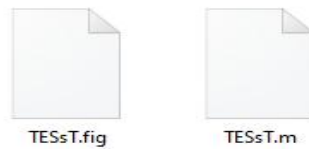
Giao diện GUI bao gồm: 8 nút nhấn (Load Image, Preprocessing, Tumor Alone, Tumor Area Location, Tumor Outline, Detect Area and Perimeter, Evaluate, Reset), thực hiện các chức năng lần lượt: đọc ảnh, tiền xử lý, xử lý hình thái học, xác định vị trí, phác họa đường biên khối u, tính kích thước khối u, đánh giá độ chính xác, và xóa. 1 Menu gồm có 4 thuật toán phân vùng theo ngưỡng Otsu, biến đổi Watershed, phân cụm K – Means hay Fuzzy C), 6 khung Axes để hiển thị ảnh, 3 Panel (Hiển thị và xử lý ảnh u não, Tính kích thước khối u, Đánh giá mô hình).



Hình 3.8 Giao diện mô phỏng GUI trên Matlab

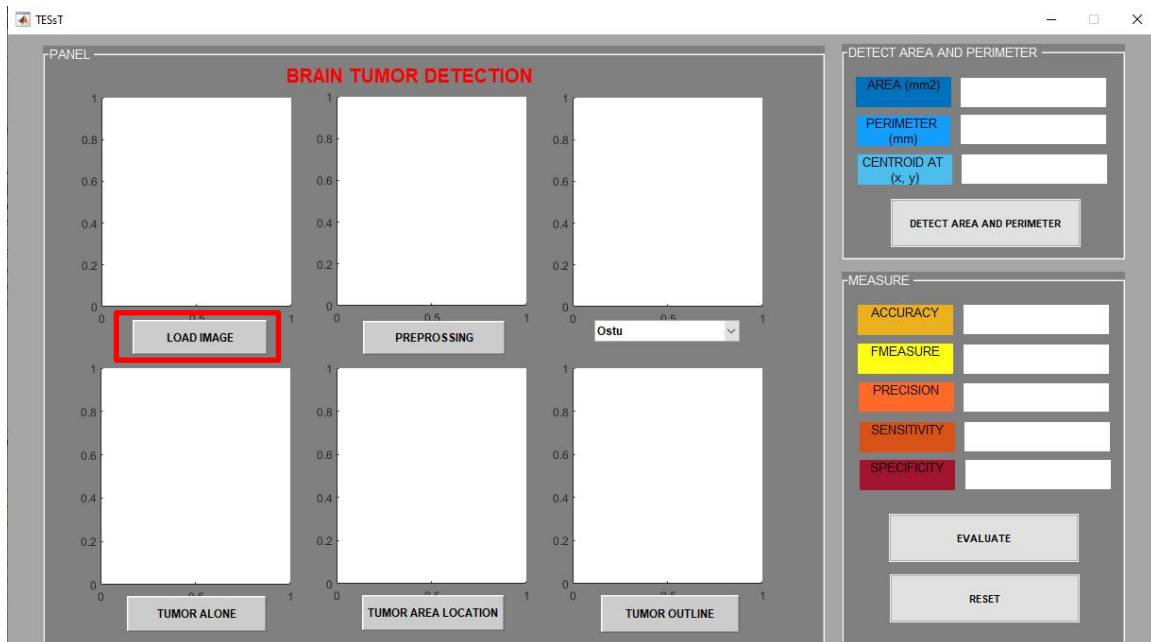
Hướng dẫn chi tiết:

- *Bước 1:* File giao diện GUI bao gồm 2 file có đuôi TESsT.m và TESsT.fig. Mở file TESsT.m giao diện GUI trên MATLAB, nhấn RUN để chạy file TESsT.fig.

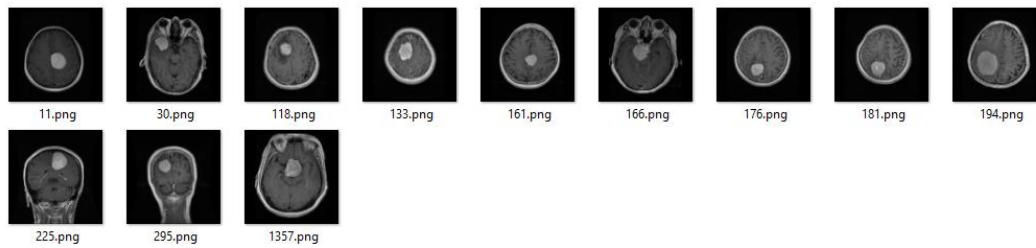


Hình 3.9 File mô phỏng GUI

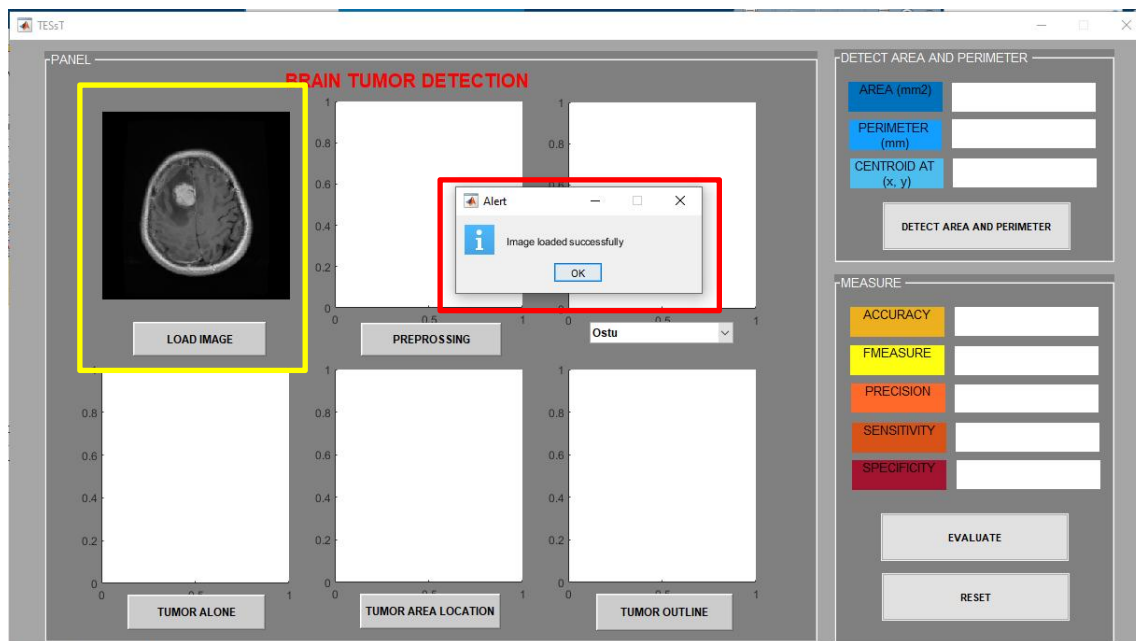
- *Bước 2:* Nhấn Load Image để đọc ảnh file ảnh MRI u não. Sau đó, sẽ xuất hiện hộp thoại Alert thông báo đã Load ảnh thành công và nhấn Ok.



Hình 3.10 Nhấn Load Image để nhập ảnh

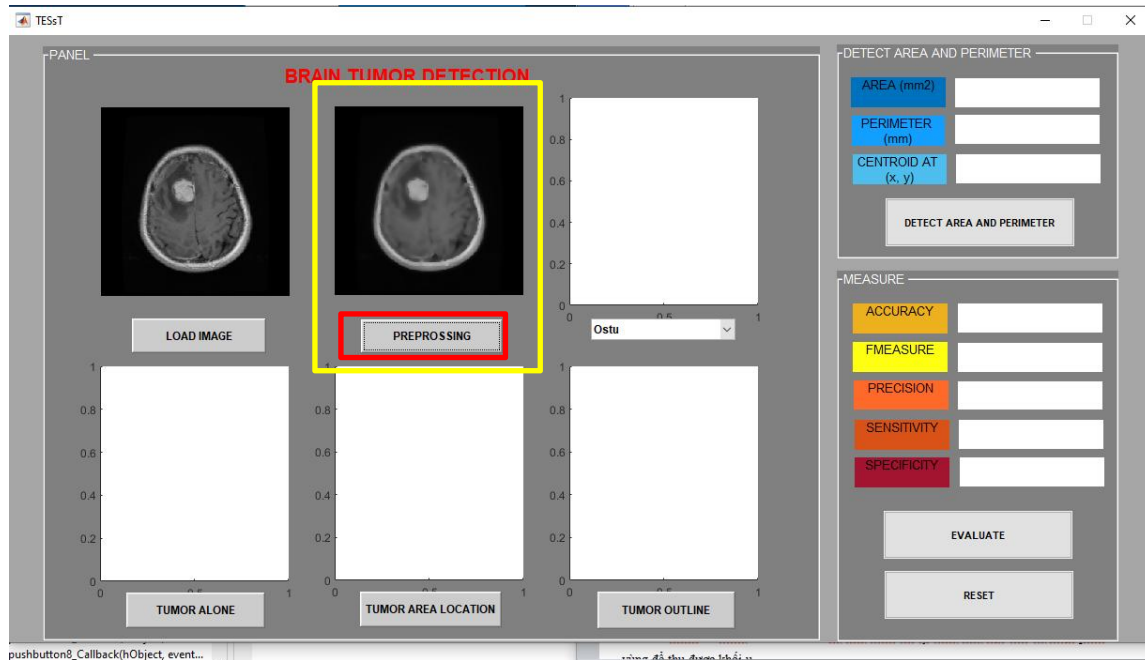


Hình 3.11 File ảnh MRI u não



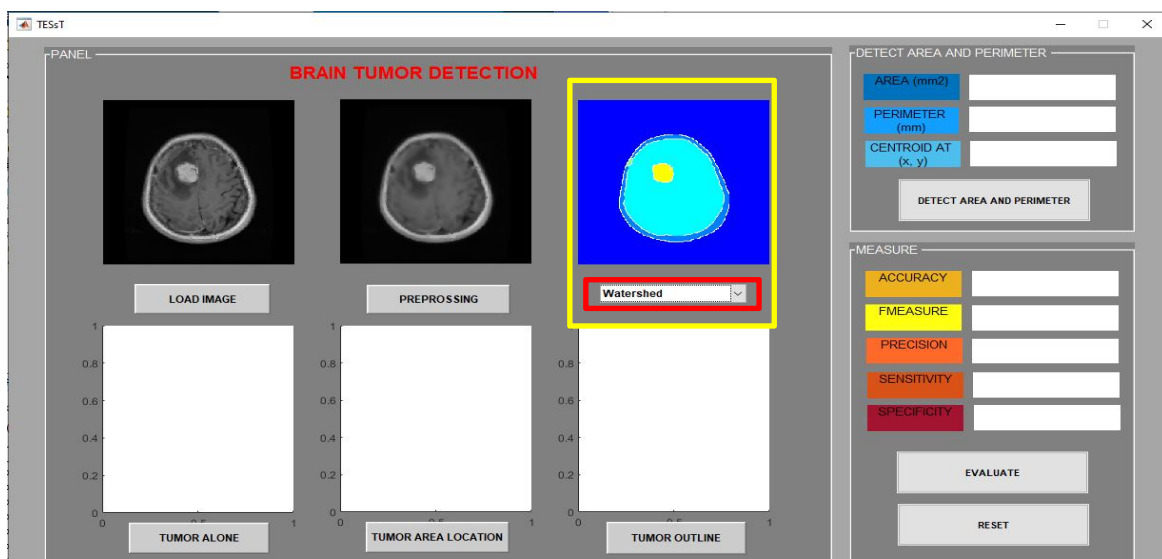
Hình 3.12 Nút nhấn Load Image trên GUI

- Bước 3: Nhấn Preprocessing để tiến hành tiền xử lý ảnh.



Hình 3.13 Nút nhấn Preprocessing trên GUI

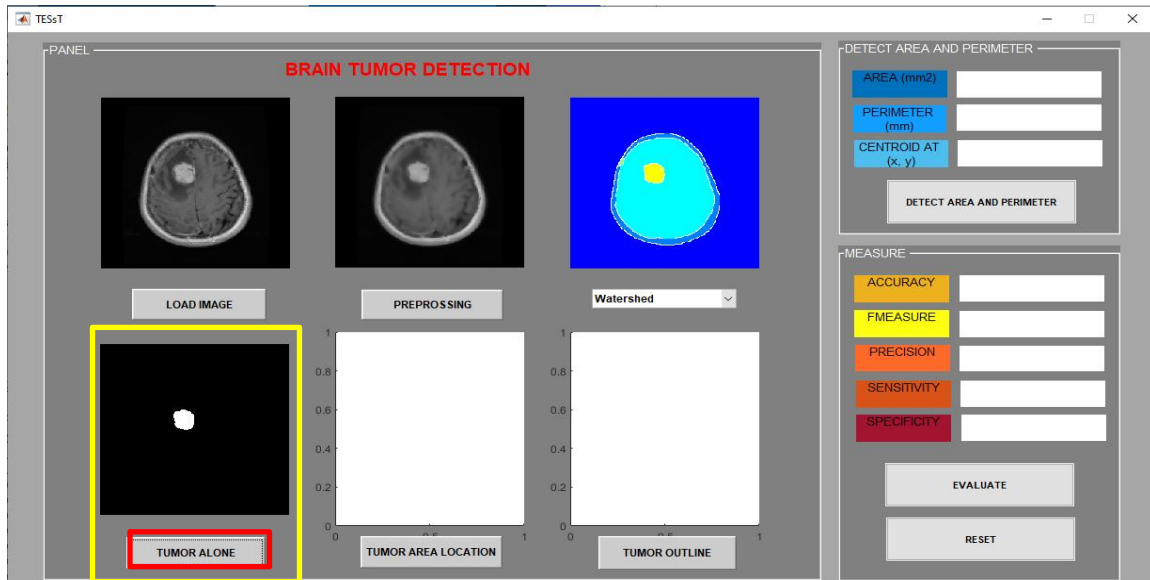
- Bước 4: Nhấn chọn Menu (Otsu, Watershed, K – Means, Fuzzy C Means) để lựa chọn thuật toán phân vùng ảnh đã tiền xử lý.



Hình 3.14 Menu thuật toán Segmentation trên GUI

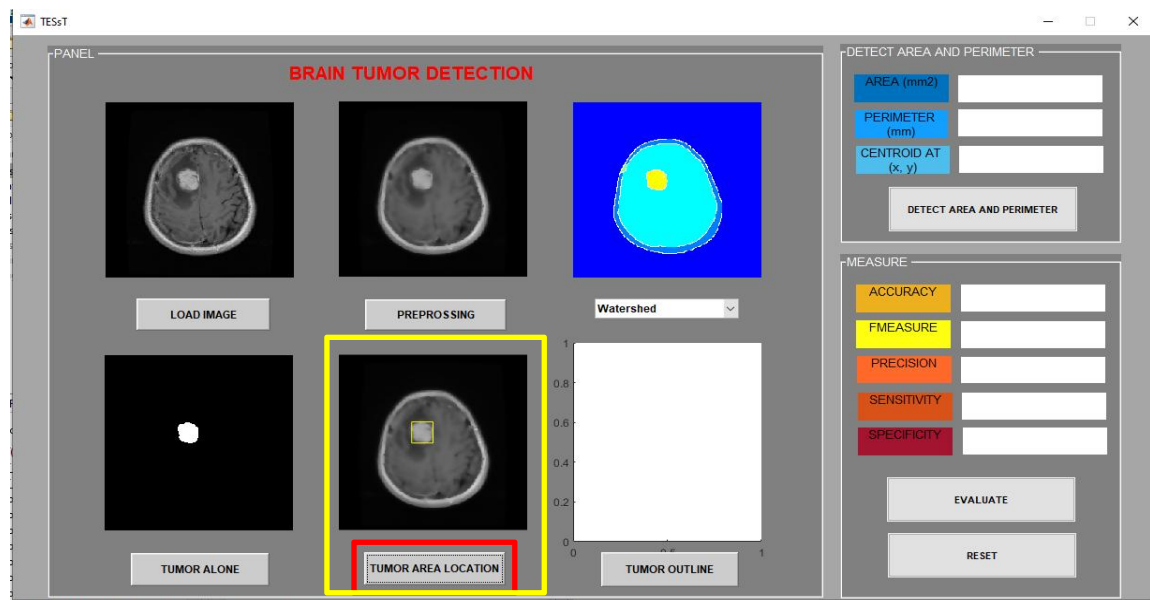
CHƯƠNG 3. TÍNH TOÁN VÀ THIẾT KẾ

- *Bước 5:* Nhấn Tumor Alone để tiến hành xử lý hình thái học ảnh đã được phân vùng để thu được khối u rõ nhất.



Hình 3.15 Nút nhấn Tumor Alone trên GUI

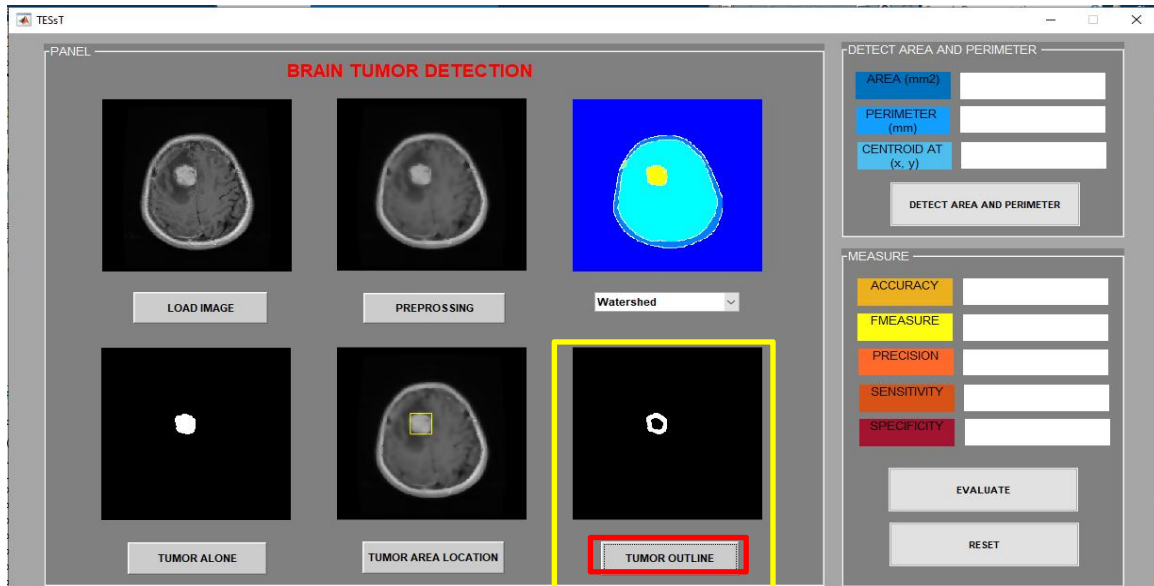
- *Bước 6:* Nhấn Tumor Area Location để xác định vị trí của khối u



Hình 3.16 Nút nhấn Tumor Area Location trên GUI

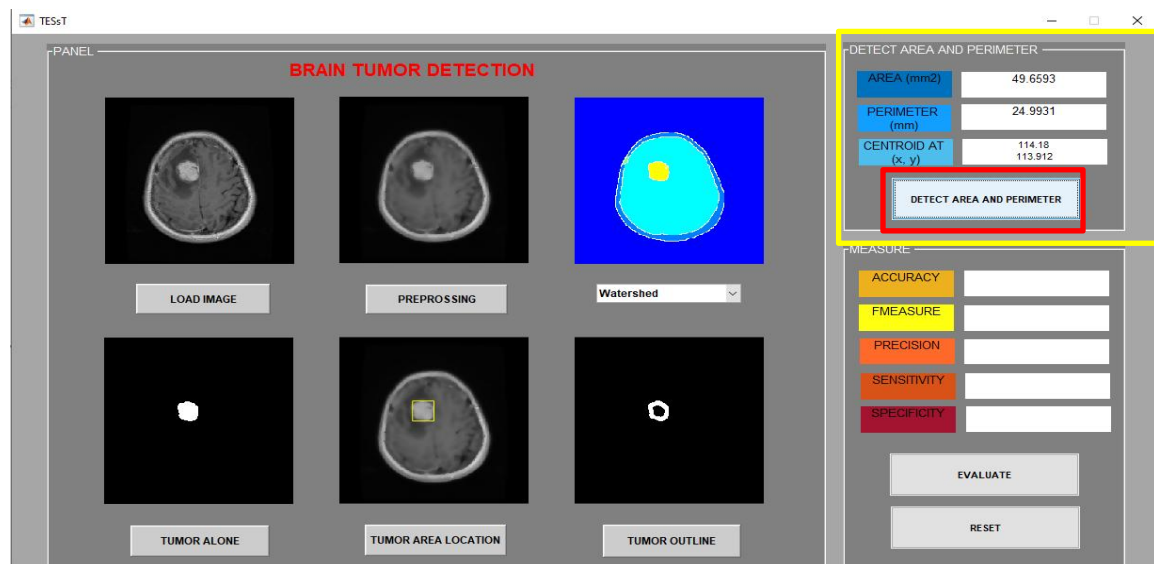
CHƯƠNG 3. TÍNH TOÁN VÀ THIẾT KẾ

- *Bước 7:* Nhấn Tumor Outline để phác họa đường biên khối u sau khi đã được xác định vị trí.



Hình 3.17 Nút nhấn Tumor Outline trên GUI

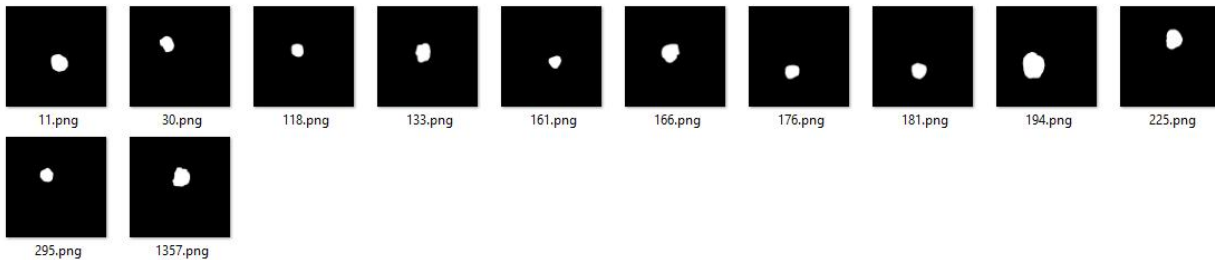
- *Bước 8:* Nhấn Detect Area and Perimeter để tính diện tích, chu vi và xác định vị trí tâm của khối u não.



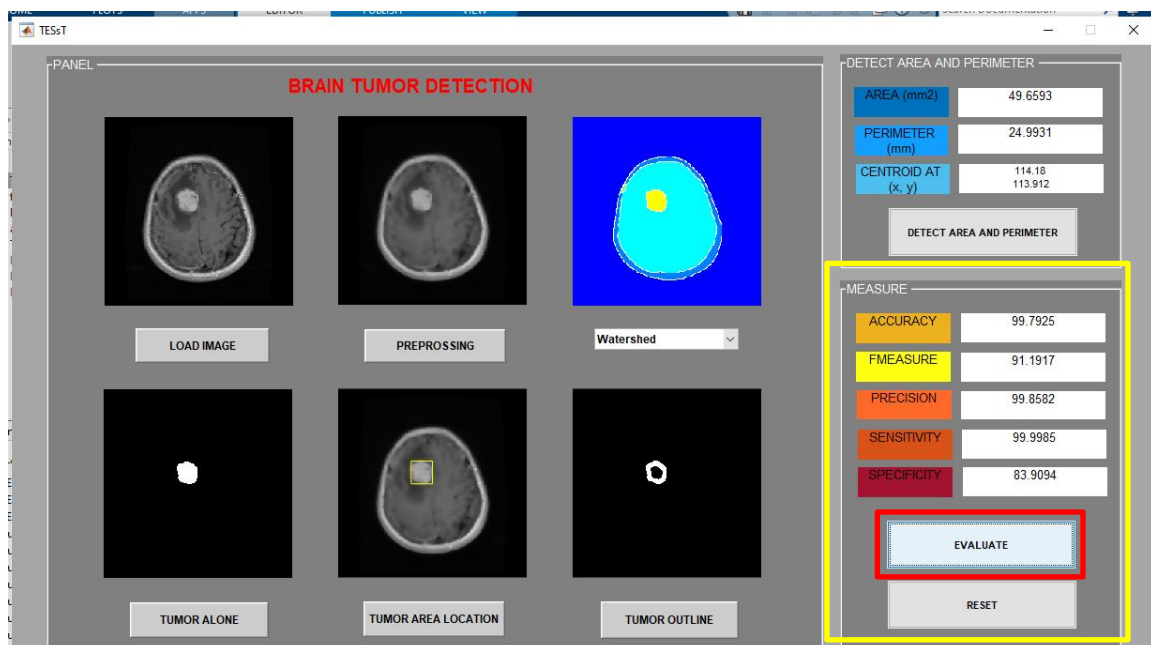
Hình 3.18 Nút nhấn Detect Area and Perimeter trên GUI

CHƯƠNG 3. TÍNH TOÁN VÀ THIẾT KẾ

- *Bước 9:* Nhấn Evaluate để chọn nhãn tương ứng với ảnh MRI u não ban đầu tính toán các thông số đánh giá độ chính xác giữa phân vùng ảnh với nhãn tương ứng. Kết quả đánh giá sẽ hiện thị lần lượt là Accuracy, FMeasure, Precision, Sensitivity, Specificity.

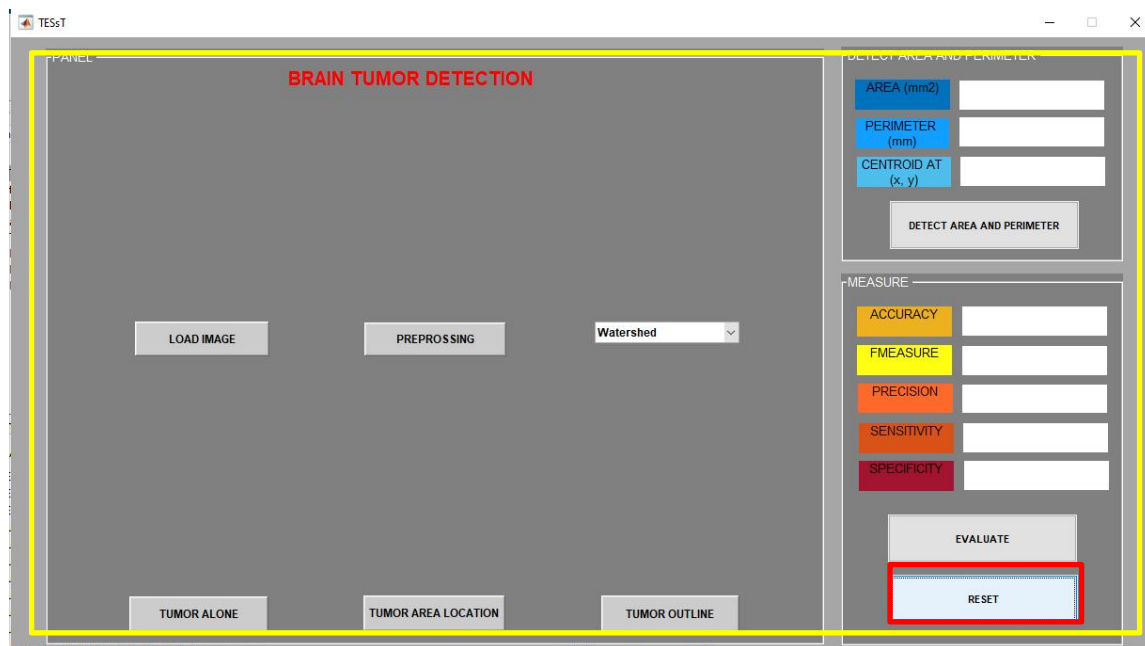


Hình 3.19 File ảnh masks của khối u



Hình 3.20 Nút nhấn Evaluate trên GUI

- *Bước 10:* Nhấn Reset để tiến hành reset và thực hiện xử lý ảnh MRI khác.



Hình 3.21 Nút nhấn Reset trên GUI

CHƯƠNG 4: KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

4.1 KẾT QUẢ

Sau 16 tuần nghiên cứu và học tập, nhóm đã tìm hiểu được về các thuật toán tăng cường biên ảnh và cách phương pháp phân vùng ảnh bao gồm: thuật toán Otsu, Watershed, phân cụm K – Means, Fuzzy C – Means, và cách tính kích thước của khối u não, cũng như cách xây dựng giao diện GUI trên Matlab tương tác với người dùng, vận dụng được kiến thức về các phương pháp xử lý ảnh y tế.

Đánh giá kết quả: Tiến hành thử nghiệm mô hình với hơn 10 ảnh MRI u não, với mỗi ảnh tiến hành thử nghiệm từng thuật toán phân vùng đã nêu trên để đánh giá độ chính xác và hiệu quả thông qua Ma trận lỗi:

- **True Positive (TP):** Số lượng dự đoán chính xác.
- **True Negative (TN):** Số lượng dự đoán chính xác một cách gián tiếp.
- **False Positive (FP):** Số lượng các dự đoán sai lệch.
- **False Negative (FN):** Số lượng các dự đoán một các sai lệch một cách gián tiếp.

Độ chính xác – Accuracy: Tỷ lệ của tất cả trường hợp phân loại đúng (không phân biệt Negative/Positive trên toàn bộ trường hợp trong mẫu kiểm định

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

Chỉ số FMeasure: Đánh giá cân bằng giữa Sensitivity và Precision

$$FMeasure = \frac{2 * TP}{2 * TP + FP + FN}$$

CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

Độ chính xác – Precision: Tỷ lệ thực sự Positive trên tổng số các trường hợp được mô hình dán nhãn Positive

$$Precision = \frac{TP}{TP + FP}$$

Độ nhạy – Sensitivity: Tỷ lệ phân loại Positive đúng trên tổng số các trường hợp Positive

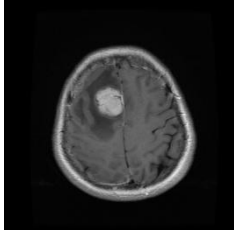

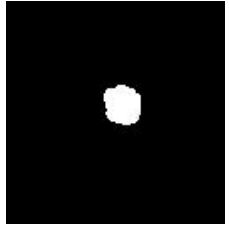
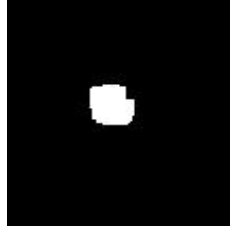
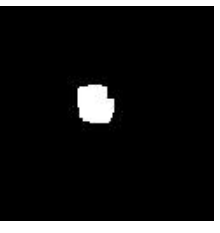
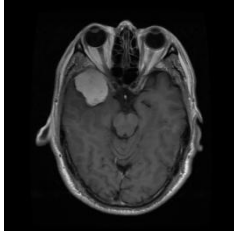

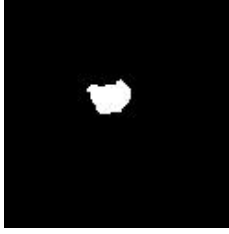


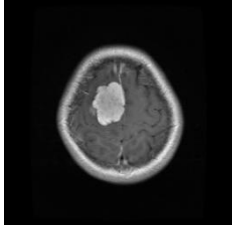

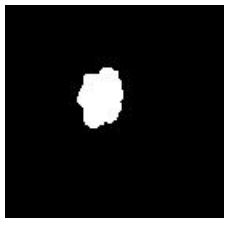


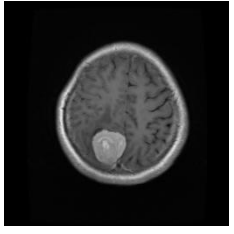
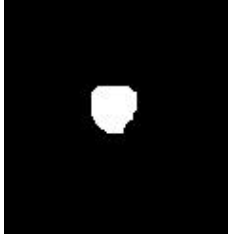
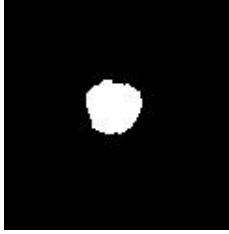
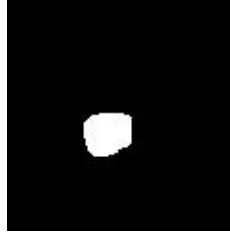
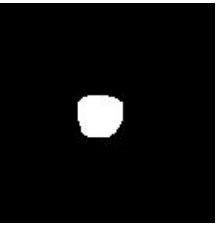
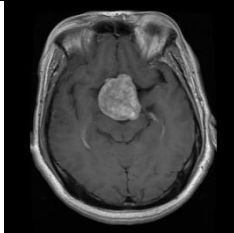
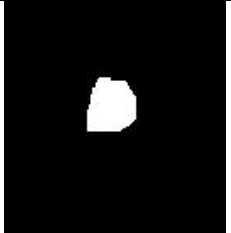



$$Sensitivity = \frac{TP}{TP + FN}$$

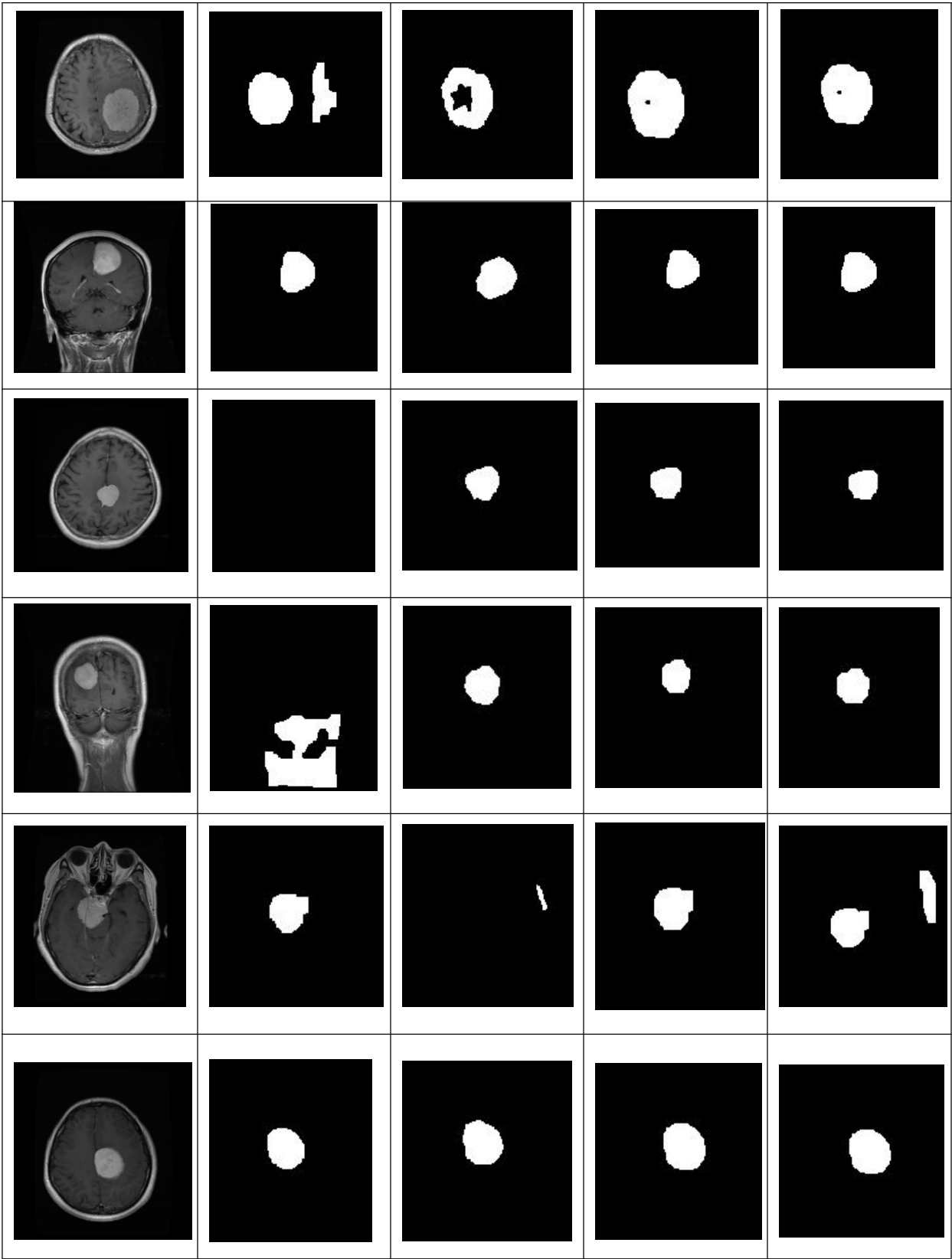
Độ đặc hiệu – Specificity: Tỷ lệ loại trừ đúng trên tổng số các trường hợp Negative

$$Specificity = \frac{TN}{TN + FP}$$

Kết quả thu được sẽ được trình bày ở Bảng 4.1, 4.2, 4.3, 4.5, 4.6, 4.7 dưới đây:

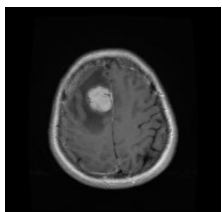
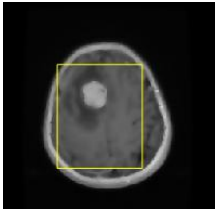
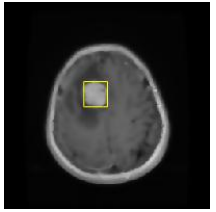
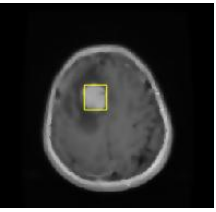
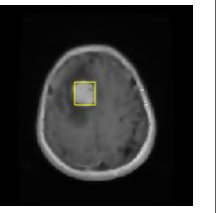
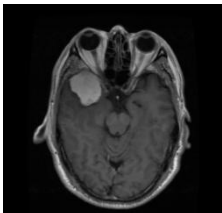
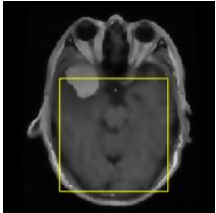
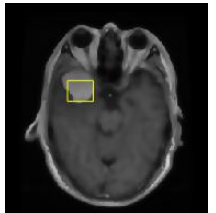
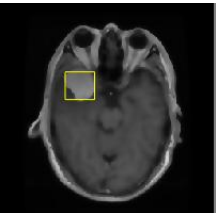
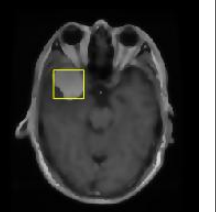
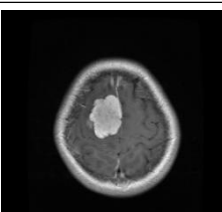
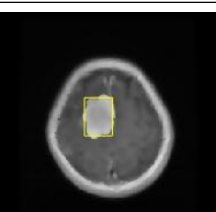
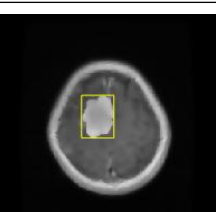
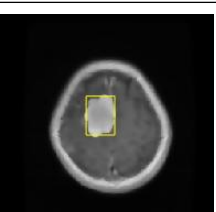
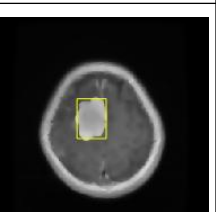
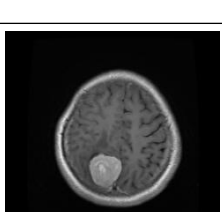


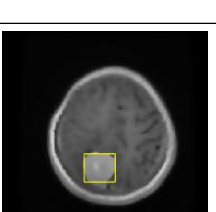

Bảng 4.1 Kết quả phân vùng khối u của các thuật toán

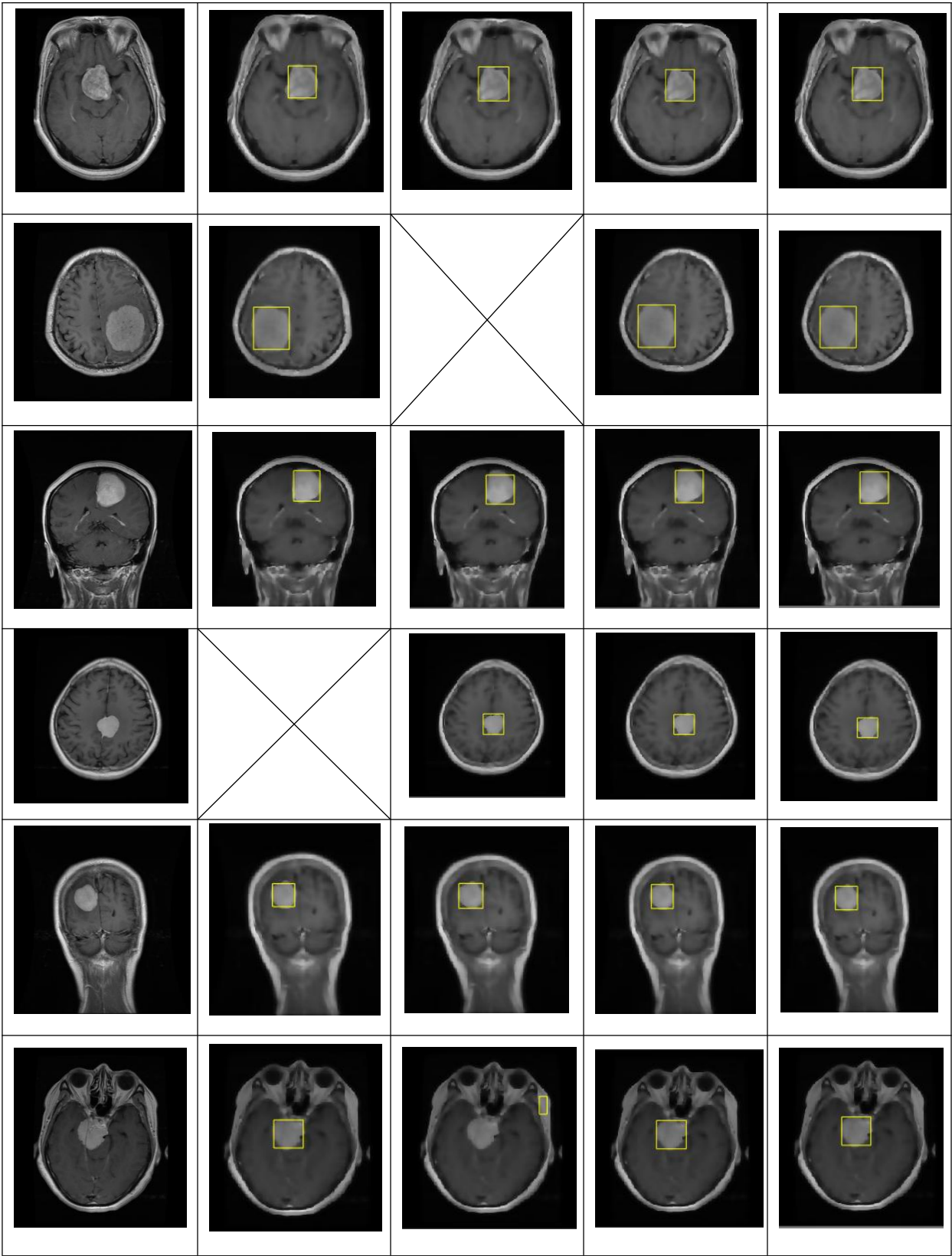
<i>Segmentation</i> <i>Ảnh</i>	<i>Otsu</i>	<i>Watershed</i>	<i>K - Means</i>	<i>Fuzzy C Means</i>
				
				
				
				
				

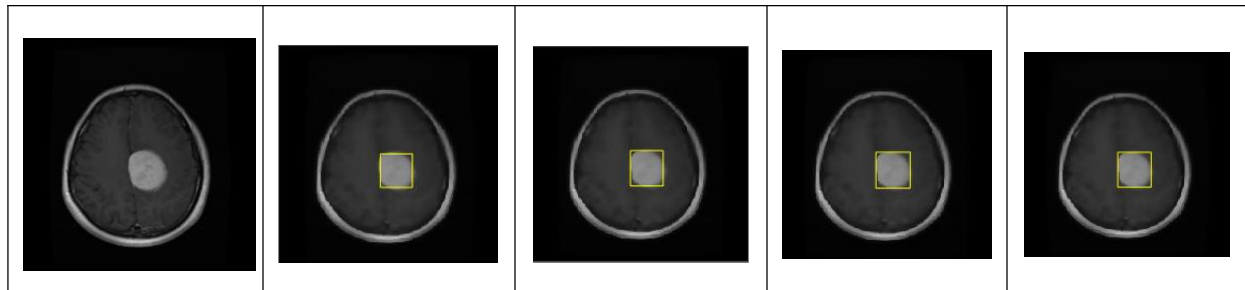


Nhận xét: Từ bảng 1 trên, ta thấy thuật toán Otsu được sử dụng kém nhất trong 4 phương pháp phân vùng ảnh ở đây. Thuật toán K – Means và Fuzzy C Means phân vùng khá hiệu quả và tương đối tốt. Thuật toán Watershed cũng khá tốt nhưng còn phụ thuộc vào đặt ngưỡng t_0 bằng bao nhiêu.

Bảng 4.2 Kết quả xác định ví của từng thuật toán

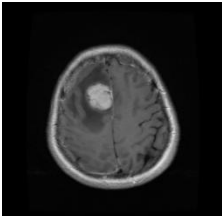
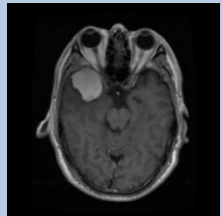
<i>Segmentation</i> <i>Ảnh</i>	<i>Otsu</i>	<i>Watershed</i>	<i>K - Means</i>	<i>Fuzzy C Means</i>
				
				
				
				



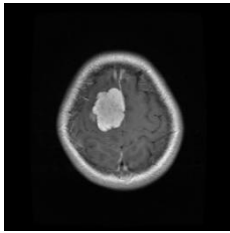
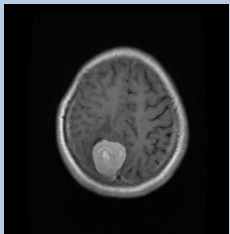
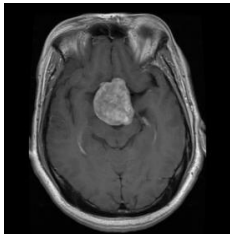
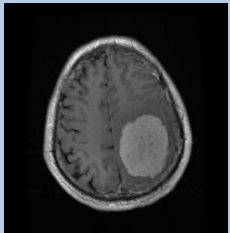


Nhận xét: Từ bảng 4.2, ta thấy sử dụng Bounding Box tìm vị trí khối u khá hiệu quả đối với cả 4 thuật toán. Tuy nhiên, ở thuật toán Otsu xác định vị trí sai khối u 3/11 ảnh và đúng 8/11 ảnh, thuật toán Watershed xác định vị trí sai 2/11 ảnh và đúng 9/11 ảnh, thuật toán K – Means và Fuzzy C xác định vị trí khối đúng 11/11 ảnh.

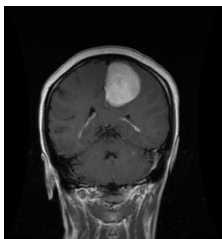
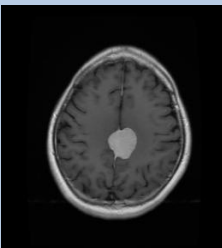
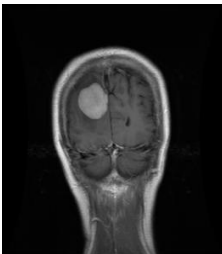
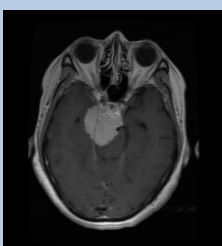
Bảng 4.3 Kết quả tính kích thước và vị trí của khối u

<i>Segmentation</i> <i>Ảnh</i>	<i>Size</i>	<i>Otsu</i>	<i>Watershed</i>	<i>K - Means</i>	<i>Fuzzy C</i> <i>Means</i>
	Area (mm ²)	x	49.6593	44.934	44.5928
	Perimeter (mm)	x	24.9931	24.588	24.4173
	Centroid (x;y)	x	(114.18;113.9 12)	(114.058;1 14.245)	(114.058;1 14.203)
	Area (mm ²)	x	33.9959	90.6994	86.1228
	Perimeter (mm)	x	22.2229	29.1899	28.9623
	Centroid (x;y)	x	(100.768; 104.803)	(97.7731;9 8.438)	(97.8238;9 8.6941)

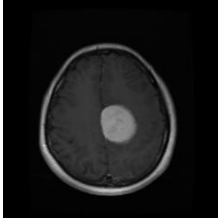
CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

	Area (mm ²)	66.6091	93.4383	87.6454	87.6454
	Perimeter (mm)	30.4281	36.8673	34.9073	34.9073
	Centroid (x;y)	(119.259;19.79)	(119.042;119.971)	(119.261;119.773)	(119.261;119.773)
	Area (mm ²)	57.7886	76.0335	72.9008	71.9732
	Perimeter (mm)	26.783	31.0653	29.9207	29.7675
	Centroid (x;y)	(119.675;166.66.248)	(119.829;166.397)	(119.925;165.51)	(119.696;165.554)
	Area (mm ²)	88.3105	87.7242	113.223	112.034
	Perimeter (mm)	34.2728	57.8416	38.6382	38.3865
	Centroid (x;y)	(133.364;105.05.251)	(130.984;105.25)	(133.412;105.504)	(133.245;105.494)
	Area (mm ²)	171.826	x	194.822	194.822
	Perimeter (mm)	46.7476	x	50.2618	50.2618
	Centroid (x;y)	(96.5368;152.528)	x	(96.8436;152.181)	(96.8436;152.181)

CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

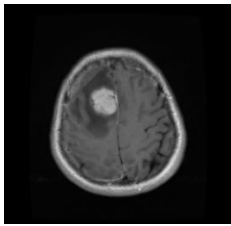
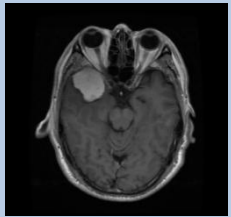
	Area (mm ²)	84.819	84.2765	109.566	100.526
	Perimeter (mm)	32.9295	32.6102	35.8923	35.6976
	Centroid (x;y)	(138.273;86.4805)	(139.241;89.37681)	(138.756;87.0243)	(138.779;87.0804)
	Area (mm ²)	x	52.967	50.6919	50.6219
	Perimeter (mm)	x	26.0675	25.1965	25.1965
	Centroid (x;y)	x	(139.324;143.396)	(139.151;143.476)	(139.133;143.467)
	Area (mm ²)	44.0677	55.6447	52.757	52.6258
	Perimeter (mm)	22.8992	25.9911	25.505	25.4357
	Centroid (x;y)	(105.735;99.8259)	(106.197;100.065)	(106.036;99.9148)	(106.028;99.9186)
	Area (mm ²)	75.7884	x	94.9084	93.8671
	Perimeter (mm)	33.2073	x	35.6558	35.5814
	Centroid (x;y)	(116.928;121.36)	x	(116.695;121.491)	(116.592;121.611)

CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

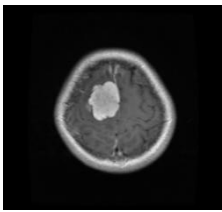
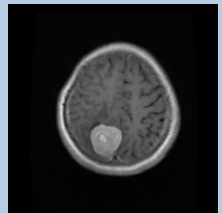
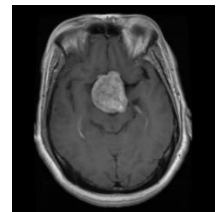
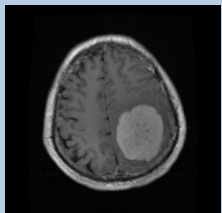
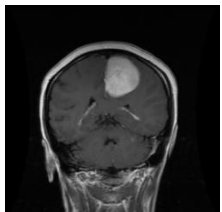
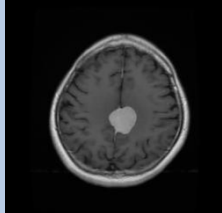
	Area (mm ²)	79.2537	89.413	101.655	101.445
	Perimeter (mm)	31.0653	33.2227	35.2139	35.1658
	Centroid (x;y)	(136.327;1 46.09)	(135.212;145. 983)	(136.39;14 6.081)	(136.359;1 46.06)

Nhận xét: Từ bảng 4.3 trên, tính kích thước và xác định vị trí của khối u dùng thuật toán Otsu rất kém, thuật toán Watershed thì diện tích có độ chính xác không cao, thuật toán Fuzzy C và K Means thì tương đương nhau về diện tích, chu vi và vị trí của khối u.

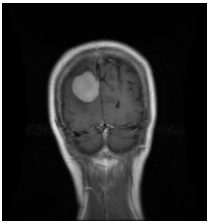
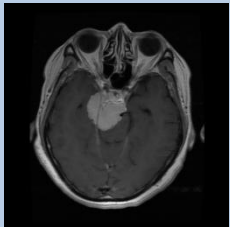
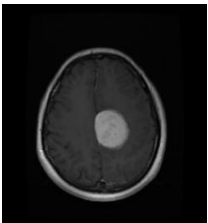
Bảng 4.4 So sánh độ chính xác của các Thuật toán Otsu

<i>Performance</i> <i>Ảnh</i>	<i>Accuracy</i> (%)	<i>FMeasure</i> (%)	<i>Precision</i> (%)	<i>Sensitivity</i> (%)	<i>Specitivity</i> (%)
	89.9368	16.0835	9.00285	90.1263	75.3278
	83.3527	14.4582	7.90941	83.3408	84.0474

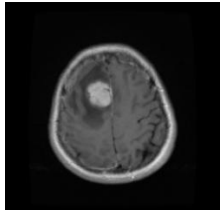
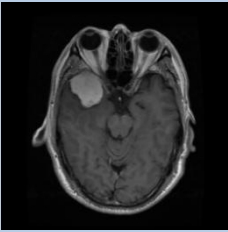
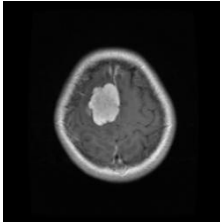
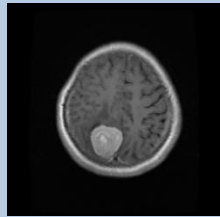
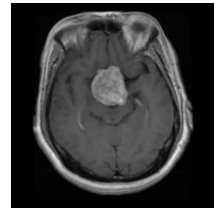
CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

	99.2691	79.8485	100	100	66.4566
	99.4278	81.4264	100	100	68.6717
	99.3652	85.8117	100	100	75.1493
	99.2142	90.4824	100	100	82.619
	99.3088	84.1995	100	100	72.7108
	98.7656	0	NaN	100	0

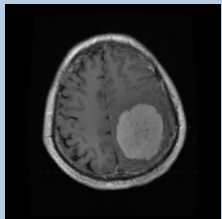
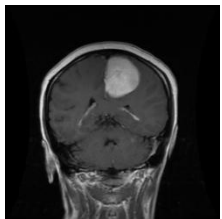
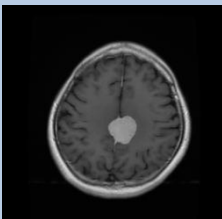

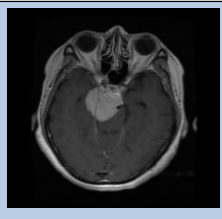
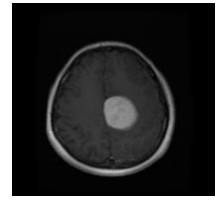
CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

	99.5117	79.6438	100	100	66.1734
	99.1623	79.7043	100	100	66.2569
	99.3774	84.6732	100	100	73.4202
Trung bình (%)	87.9937	63.3029	74.2648	88.497	66.4394

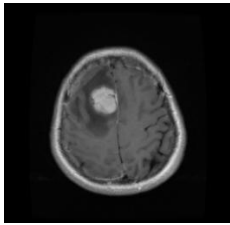
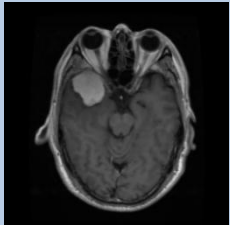
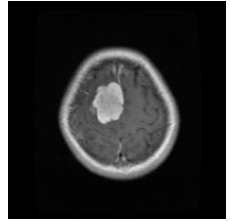
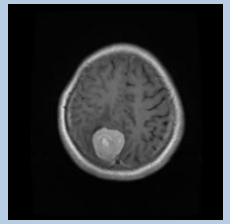
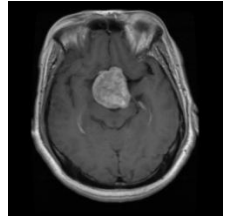
Bảng 4.5 Đánh giá độ chính xác của Thuật toán Watershed

<i>Performance</i> <i>Ảnh</i>	<i>Accuracy</i> (%)	<i>FMeasure</i> (%)	<i>Precision</i> (%)	<i>Sensitivity</i> (%)	<i>Specitivity</i> (%)
	99.7925	91.1917	99.8582	99.9985	83.9094
	99.0616	61.0513	100	100	43.938
	99.8337	96.0464	99.6238	99.9922	92.7171
	99.8169	94.7322	99.815	99.9969	90.142
	99.295	84.1672	98.7138	99.9749	73.3572

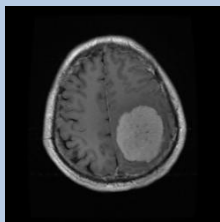
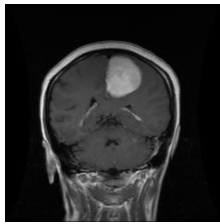
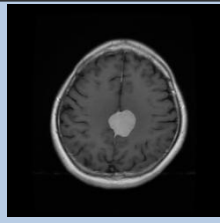
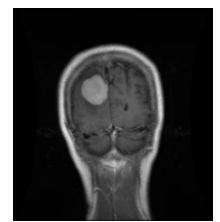
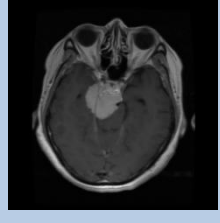
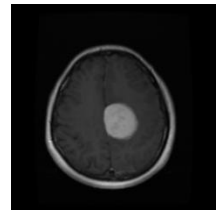
CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

	95.4788	0.336361	50	99.992	0.168748
	99.292	83.7649	99.9165	99.9984	72.1084
	99.8611	94.1704	97.7394	99.9737	90.8529
	99.7559	90.7834	99.7468	99.9969	83.2981
	97.3419	0	0	99.8201	0
	99.5987	90.6306	100	100	82.8664
Trung bình (%)	99.01165	71.53404	85.94668	99.9769	64.85075

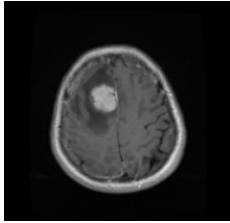
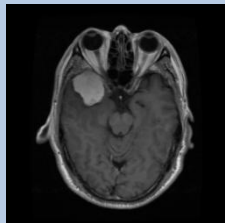
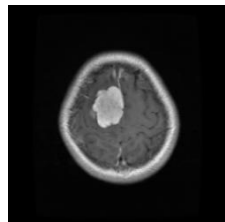
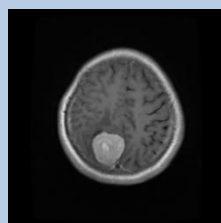
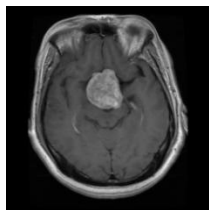
Bảng 4.6 So sánh độ chính xác của các Thuật toán K - Means

<i>Performance</i> <i>Ảnh</i>	<i>Accuracy</i> (%)	<i>FMeasure</i> (%)	<i>Precision</i> (%)	<i>Sensitivity</i> (%)	<i>Specitivity</i> (%)
	99.6933	86.4097	99.8438	99.9985	76.1621
	99.7131	90.7023	99.1351	99.9876	83.5916
	99.7238	93.2387	99.9199	99.9984	87.395
	99.7543	92.7964	99.9037	99.9984	86.6332
	99.733	94.5669	98.4486	99.9624	90.9797

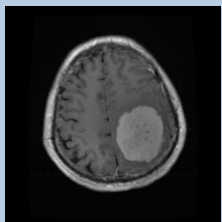
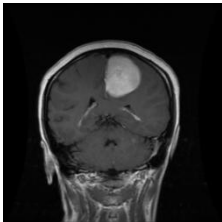
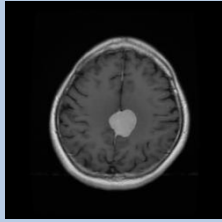
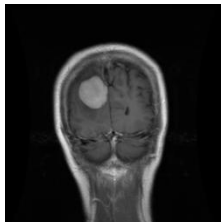
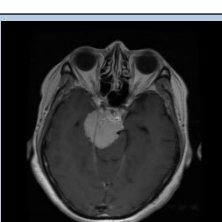
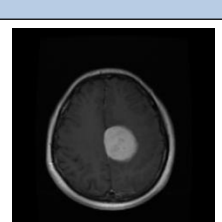
CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

	99.6124	95.5734	98.8108	99.9473	92.5412
	99.6445	92.4863	99.5142	99.989	86.3855
	99.823	92.4183	98.0583	99.9784	87.3918
	99.794	92.3164	100	100	85.7294
	99.5422	89.8785	99.626	99.9922	81.8685
	99.8199	96.0429	98.9634	99.9766	93.2899
Trung bình (%)	99.714	92.403	99.293	99.984	86.543

Bảng 4.7 So sánh độ chính xác của các Thuật toán Fuzzy C Means

<i>Performance</i> <i>Ảnh</i>	<i>Accuracy</i> (%)	<i>FMeasure</i> (%)	<i>Precision</i> (%)	<i>Sensitivity</i> (%)	<i>Specitivity</i> (%)
	99.6887	86.1601	100	100	75.6853
	99.7131	90.7023	99.1351	99.9876	83.5916
	99.7238	93.2387	99.9199	99.9984	87.395
	99.7543	92.7964	99.9037	99.9984	86.6332
	99.8215	96.4448	98.1447	99.953	94.9029

CHƯƠNG 4. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

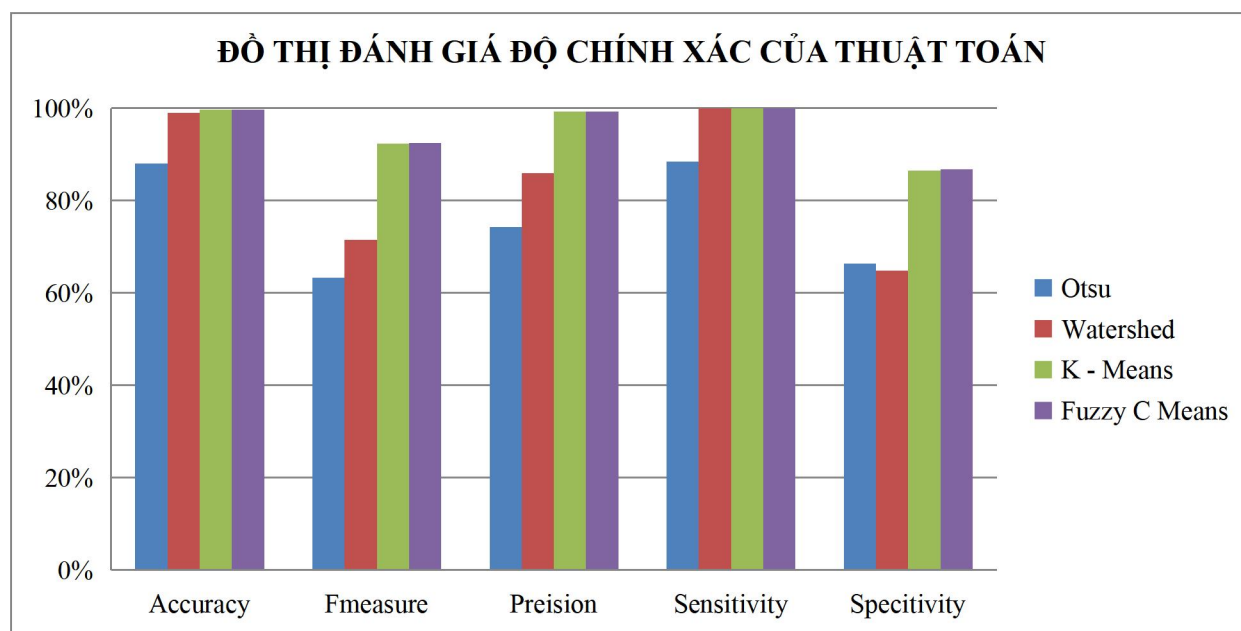
	99.6124	95.5734	98.8108	99.9473	92.5413
	99.6323	92.2032	99.5807	99.9906	85.8434
	99.8245	92.4787	98.1944	99.9799	87.3918
	99.7879	92.0707	100	100	85.3066
	99.5422	89.8785	99.626	99.9922	81.8685
	99.8184	96.0054	99.0305	99.9781	93.1596
Trung bình (%)	99.720	92.505	99.304	99.984	86.756

4.2 NHẬN XÉT

Từ Bảng 4.4, 4.5, 4.6, 4.7, ta tổng hợp thu được bảng 4.8

Bảng 4.8 Đánh giá độ chính xác của các thuật toán

<i>Performance</i> <i>Segmentation</i>	<i>Accuracy</i> (%)	<i>FMeasure</i> (%)	<i>Precision</i> (%)	<i>Sensitivity</i> (%)	<i>Specitivity</i> (%)
Otsu	87.994	63.303	74.265	88.497	66.439
Watershed	99.012	71.534	85.947	99.977	64.851
K - Means	99.714	92.403	99.293	99.984	86.543
Fuzzy C - Means	99.72	92.505	99.304	99.984	86.756



Hình 4.1 Đồ thị đánh giá độ chính xác các thuật toán

Từ đồ thị trên, độ chính xác của thuật toán Fuzzy C – Means và K – Means cao nhất trong cả 4 thuật toán, tiếp đến thuật toán Watershed và cuối cùng thuật toán Otsu có độ chính xác thấp nhất.

4.3 ĐÁNH GIÁ

Ưu điểm:

- Giao diện GUI dễ thao tác, sử dụng và quan sát kết quả.
- Độ sai số tương đối nhỏ.

Nhược điểm:

- Trong quá trình thao tác trên GUI, thường gặp lỗi không hiển thị hình ảnh như ý muốn.
- Thay đổi phép co hoặc giãn trong xử lý hình thái học bằng cách thủ công là gián tiếp thay đổi Code trong Chương trình.
- Chưa tối ưu được các thuật toán để đưa ra kết quả chính xác hơn.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Dựa vào mục tiêu đặt ra ban đầu của đề tài “Xây dựng thuật toán tăng cường biên ảnh để tính kích thước khối u não” và cũng như những nội dung đã làm, nhận thấy kết quả đề tài đạt được một phần mục tiêu đề ra. Kết quả cho thấy, thuật toán được xây dựng đúng quy trình thu được ảnh của khối u được phân qua các bước xử lý nêu trên, xác định đúng vị trí của khối u, nhưng kết quả kích thước của khối u não có độ chính xác không cao. Các thuật toán được thử nghiệm trên phần mềm MATLAB R2020b nhiều lần, tính toán độ chính xác dự đoán u não của thuật toán Otsu, Watershed, K – Means, Fuzzy C – Means. Trong đó, thuật toán K – Means và Fuzzy C - Means có độ chính xác cao hơn 2 thuật toán còn lại.

Kết quả của đề tài là hình ảnh 2D về khối u, phân vùng khối u và kích thước khối u là con số về diện tích và chu vi, làm cho các bác sĩ cũng như nhân viên y tế có cái nhìn tổng thể về khối u.

5.2 HƯỚNG PHÁT TRIỂN

Thêm một số chức năng như:

- Xây dựng thuật toán AI phân vùng ảnh để tính kích thước khối u
- Phân loại từng khối u não.
- Xây dựng đa dạng phương pháp tính kích thước (diện tích, chi vi và thể tích) khối u chính xác hơn.
- Dựng lại ảnh của khối u thành 3D để quan sát hình ảnh khối u giúp bác sĩ chẩn đoán bệnh dễ hơn.

TÀI LIỆU THAM KHẢO

[1] MOH, “Tình hình ung thư tại Việt Nam”, *Cổng thông tin điện tử Bộ Y Tế*, 2021.

(https://moh.gov.vn/hoat-dong-cua-dia-phuong/-/asset_publisher/gHbla8vOQDuS/content/tinh-hinh-ung-thu-tai-viet-nam)

[2] Vinmec, “Triệu chứng cảnh báo u não & cách điều trị”, *Hệ thống Bệnh viện Đa khoa Quốc tế Vinmec*, 2019.

(<https://www.vinmec.com/vi/tin-tuc/thong-tin-suc-khoe/trieu-chung-canh-bao-u-nao-cach-dieu-tri/>)

[3] Nguyễn Thanh Hải, Ngô Quốc Cường, “Giáo trình: Xử lý ảnh”, *Trường ĐHSPKT, Tp.HCM, Nhà xuất bản ĐH Quốc Gia, Tp.HCM*, 2018.

[4] Wikipedia, “Hình ảnh y khoa”, *Bách khoa toàn thư mở Wikipedia*, 2021.

(https://vi.wikipedia.org/wiki/H%C3%ACnh_%E1%BA%A3nh_y_khoa)

[5] Trần Thị Quỳnh Như, “Dùng phương pháp Wavelet tăng cường biên ảnh để xác định kích thước khối u đặc”, *Luận văn thạc sĩ, trường ĐH SPKT, Tp. HCM*, 2014.

(<http://stinet.gov.vn/viewer/?id=164537>)

[6] TâmAnhHospital, “U não: dấu hiệu, nguyên nhân và cách chuẩn đoán bệnh”, *Công ty Cổ phần bệnh viện đa khoa Tâm Anh*, 2022.

(<https://tamanhhospital.vn/u-nao/>)

[7] Choyte, “MRI là gì? Cơ sở lý thuyết, cấu tạo và ứng dụng của máy chụp cộng hưởng từ”, *Công ty Cổ phần Chợ Y tế Việt Nam*, 2015.

(<https://choyte.com/mri-la-gi-co-so-ly-thuyet-cau-tao-va-ung-dung-cua-may-chup-cong-huong-tu-5773.htm>)

[8] VINBIGDATA, “Phân vùng ảnh: Các thuật toán và cơ sở dữ liệu mã nguồn hữu ích”, Công ty cổ phần VinBigData, 2022.

(<https://vinbigdata.com/phan-vung-anh-cac-thuat-toan-va-co-so-du-lieu-ma-nguon-mo-huu-ich/>)

[9] Wikipedia, “Fuzzy clustering”, *Wikipedia The Free Encyclopedia*, 2022.

(https://en.wikipedia.org/wiki/Fuzzy_clustering)

[10] Wikipedia, “k-means clustering”, *Wikipedia The Free Encyclopedia*, 2022.

(https://en.wikipedia.org/wiki/K-means_clustering)

[11] Wikipedia, “Otsu’s method”, *Wikipedia The Free Encyclopedia*, 2022.

(https://en.wikipedia.org/wiki/Otsu%27s_method)

PHỤ LỤC

Chương trình hàm đánh giá thuật toán:

```
function [Accuracy, Sensitivity, Fmeasure, Precision, MCC, Dice, Jaccard, Specitivity] =  
EvaluateImageSegmentationScores(A, B)  
    % Copyright 2019 by Dang N. H. Thanh. Email: thanh.dnh.cs@gmail.com  
    % Visit my site: https://sites.google.com/view/crx/sdm  
    % You need to install the image processing toolbox  
    %  
    =====  
    % A and B need to be binary images  
    % A is the ground truth, B is the segmented result.  
    % MCC - Matthews correlation coefficient  
    % Note: Sensitivity = Recall  
    % TP - true positive, FP - false positive,  
    % TN - true negative, FN - false negative  
  
    % If A, B are binary images, but uint8 (0, 255),  
    % Need to convert to logical images.  
    if(isa(A,'logical'))  
        X = A;  
    else  
        X = imbinarize(A);  
    end  
    if(isa(B,'logical'))  
        Y = B;  
    else  
        Y = imbinarize(B);  
    end
```

<pre> % Evaluate TP, TN, FP, FN sumindex = X + Y; TP = length(find(sumindex == 2)); TN = length(find(sumindex == 0)); subtractindex = X - Y; FP = length(find(subtractindex == -1)); FN = length(find(subtractindex == 1)); Accuracy = ((TP+TN)/(FN+FP+TP+TN))*100; Sensitivity = (TP/(TP+FN))*100; Precision = (TP/(TP+FP))*100; Fmeasure = (2*TP/(2*TP+FP+FN))*100; MCC = (TP*TN-FP*FN)/sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)); % If you use MATLAB2017b+, you can call: Dice = dice(A, B), but A, B % need to be converted to the logical form % If you use MATLAB2017b+, you can call: Jaccard = jaccard(A, B), but % A, B need to be converted to the logical form Dice = 2*TP/(2*TP+FP+FN); Jaccard = Dice/(2-Dice); Specitivity = (TN/(TN+FP))*100; end </pre>
--

Chương trình giao diện GUI trên MATLAB:

<pre> function varargout = TESsT(varargin) % TESsT MATLAB code for TESsT.fig % TESsT, by itself, creates a new TESsT or raises the existing % singleton*. % % H = TESsT returns the handle to a new TESsT or the handle to % the existing singleton*. </pre>
--

```

%
%  TESsT('CALLBACK',hObject,eventData,handles,...) calls the local
%  function named CALLBACK in TESsT.M with the given input arguments.
%
%  TESsT('Property','Value',...) creates a new TESsT or raises the
%  existing singleton*. Starting from the left, property value pairs are
%  applied to the GUI before TESsT_OpeningFcn gets called. An
%  unrecognized property name or invalid value makes property application
%  stop. All inputs are passed to TESsT_OpeningFcn via varargin.
%
%  *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%  instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TESsT

% Last Modified by GUIDE v2.5 17-Dec-2022 13:38:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @TESsT_OpeningFcn, ...
    'gui_OutputFcn', @TESsT_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

```

```

if narginout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before TESSsT is made visible.
function TESSsT_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TESSsT (see VARARGIN)

% Choose default command line output for TESSsT
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TESSsT wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = TESSsT_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global brainImg

[filename, pathname] = uigetfile({'*.png'; '*.jpg'; '*.bmp'; '*.tif'; '*.gif'; '*.jpeg'}, 'Load Image
File');

if isequal(filename,0)||isequal(pathname,0)
    warndlg('Press OK to continue', 'Warning');

else
    brainImg = imread([pathname filename]);
    brainImg = imresize(brainImg,[256,256]);

    axes(handles.axes1);
    imshow(brainImg);
    axis off
    helpdlg(' Image loaded successfully ', 'Alert');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

```

```

% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global brainImg inp

num_iter = 10;
delta_t = 1/7;
kappa = 15;
option = 2;

inp = anisodiff(brainImg,num_iter,delta_t,kappa,option);

inp = uint8(inp);

inp = imresize(inp,[256,256]);
if size(inp,3)>1
    inp = rgb2gray(inp);
end
axes(handles.axes2);
imshow(inp);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global inp im_seg max_area luachonseg im_tumoralone tumor_label label stats density area sout
h im_seg1 level1 fim
luachonseg = get(handles.popupmenu2,'value');

if (luachonseg == 1)
    SE = ones(5);
    im_seg = imopen(im_seg, SE);
    im_tumoralone = bwareaopen(im_seg, 600);

    stats = regionprops(im_seg,'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];
    High_Density_Area = density > 0.5; % reduce to detect small or early stage tumors
    max_area = max(area(High_Density_Area));

    if max_area > 100
        axes(handles.axes4);
        imshow(im_tumoralone);
    else
        h = msgbox('No Tumor!!','status');
        return;
    end

elseif (luachonseg == 2)
    sout = imresize(inp,[256,256]);
    t0 = 10;
    th = t0 + ((max(inp(:)) + min(inp(:)))/2);
    for i = 1:1:size(inp,1)
        for j = 1:1:size(inp,2)

```

```

        if inp(i,j) > th
            sout(i,j) = 1;
        else
            sout(i,j) = 0;
        end
    end
end
label = bwlabel(sout);
stats = regionprops(logical(sout),'Solidity','Area','BoundingBox');
density = [stats.Solidity];
area = [stats.Area];
high_dense_area = density > 0.6;
max_area = max(area(high_dense_area));
tumor_label = find(area == max_area);
im_tumoralone = ismember(label,tumor_label);

if max_area>100
    axes(handles.axes4);
    imshow(im_tumoralone);
else
    h = msgbox('No Tumor!!','status');
    return;
end

elseif (luachonseg == 3)
    im_seg = (im_seg == 3);
    SE = ones(10); % small
    %SE=ones(15); % big
    im_seg = imopen(im_seg,SE);
    im_tumoralone = bwareaopen(im_seg, 400);

```



```

stats = regionprops(logical(im_seg),'Solidity','Area','BoundingBox');
density = [stats.Solidity];
area = [stats.Area];
high_dense_area = density > 0.6;
max_area = max(area(high_dense_area));

if max_area > 100
    axes(handles.axes4);
    imshow(im_tumoralone);
else
    h = msgbox('No Tumor!!','status');
    return;
end
elseif (luachonse == 4)
    [im_seg1,level1] = fcmthresh(fim,1);
    SE = ones(10);
    im_seg1 = imopen(im_seg1,SE);
    im_tumoralone = bwareaopen(im_seg1, 500);

    stats = regionprops(logical(im_seg1),'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];
    high_dense_area = density > 0.7;
    max_area = max(area(high_dense_area));

    if max_area > 100
        axes(handles.axes4);
        imshow(im_tumoralone);
    else
        h = msgbox('No Tumor!!','status');
        return;
    end
end

```

```

    end
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global luachonseg max_area tumor_label im_tumoralone1 inp_sout wantedBox im_seg im_seg1
h
luachonseg = get(handles.popupmenu2,'value');

if (luachonseg == 1)
    stats = regionprops(im_seg,'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];
    High_Density_Area = density > 0.5; % reduce to detect small or early stage tumors

    max_area = max(area(High_Density_Area));
    tumor_label = find(area == max_area);
    im_tumoralone1 = ismember(im_seg,tumor_label);

    if max_area > 100
        box = stats(tumor_label);
        wantedBox = box.BoundingBox;
        axes(handles.axes5);
        imshow(inp);
        hold on;
        rectangle('Position',wantedBox,'EdgeColor','y');
        hold off;
    else

```

```

        h = msgbox('No Tumor!!','status');
        return;
    end

elseif (luachonseg == 2)
    label = bwlabel(sout);
    stats = regionprops(logical(sout),'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];
    high_dense_area = density>0.6;
    max_area = max(area(high_dense_area));
    tumor_label = find(area == max_area);
    im_tumoralone1 = ismember(label,tumor_label);

    if max_area > 100
        box = stats(tumor_label);
        wantedBox = box.BoundingBox;
        axes(handles.axes5);
        imshow(inp);
        hold on;
        rectangle('Position',wantedBox,'EdgeColor','y');
        hold off;
    else
        h = msgbox('No Tumor!!','status');
        return;
    end

elseif (luachonseg == 3)
    stats = regionprops(logical(im_seg),'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];

```

```

high_dense_area = density > 0.6;
max_area = max(area(high_dense_area));
tumor_label = find(area == max_area);
im_tumoralone1 = ismember(im_seg,tumor_label);

if max_area > 100
    box = stats(tumor_label);
    wantedBox = box.BoundingBox;
    axes(handles.axes5);
    imshow(inp);
    hold on;
    rectangle('Position',wantedBox,'EdgeColor','y');
    hold off;
else
    h = msgbox('No Tumor!!!','status');
    return;
end
elseif (luachonseg == 4)
    stats = regionprops(logical(im_seg1),'Solidity','Area','BoundingBox');
    density = [stats.Solidity];
    area = [stats.Area];
    high_dense_area = density > 0.6;
    max_area = max(area(high_dense_area));
    tumor_label = find(area == max_area);
    im_tumoralone1 = ismember(im_seg1,tumor_label);

    if max_area > 100
        box = stats(tumor_label);
        wantedBox = box.BoundingBox;
        axes(handles.axes5);
        imshow(inp);
    end
end

```

```

        hold on;
        rectangle('Position',wantedBox,'EdgeColor','y');
        hold off;
    else
        h = msgbox('No Tumor!!','status');
        return;
    end
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global luachonseg im_tumoralone1 erodedImage tumorOutline filledImage dilationAmount
luachonseg = get(handles.popupmenu2,'value');

if (luachonseg == 1)
    dilationAmount = 5;
    rad = floor(dilationAmount);
    [r,c] = size(im_tumoralone1);
    filledImage = imfill(im_tumoralone1, 'holes');

    for i=1:r
        for j=1:c
            x1=i-rad;
            x2=i+rad;
            y1=j-rad;
            y2=j+rad;
            if x1<1

```

```

        x1=1;
    end
    if x2>r
        x2=r;
    end
    if y1<1
        y1=1;
    end
    if y2>c
        y2=c;
    end
    erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
end
end
tumorOutline = im_tumoralone1 - erodedImage;

axes(handles.axes6);
imshow(tumorOutline);

elseif (luachonseg == 2)
    dilationAmount = 5;
    rad = floor(dilationAmount);
    [r,c] = size(im_tumoralone1);
    filledImage = imfill(im_tumoralone1, 'holes');

    for i=1:r
        for j=1:c
            x1=i-rad;
            x2=i+rad;
            y1=j-rad;
            y2=j+rad;

```

```

        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end
tumorOutline = im_tumoralone1 - erodedImage;

axes(handles.axes6);
imshow(tumorOutline);

elseif (luachonseg == 3)
    dilationAmount = 5;
    rad = floor(dilationAmount);
    [r,c] = size(im_tumoralone1);
    filledImage = imfill(im_tumoralone1, 'holes');

    for i=1:r
        for j=1:c
            x1=i-rad;
            x2=i+rad;
            y1=j-rad;

```

```

        y2=j+rad;
        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end
tumorOutline = im_tumoralone1 - erodedImage;

axes(handles.axes6);
imshow(tumorOutline);

elseif (luachonseg == 4)
    dilationAmount = 5;
    rad = floor(dilationAmount);
    [r,c] = size(im_tumoralone1);
    filledImage = imfill(im_tumoralone1, 'holes');

    for i=1:r
        for j=1:c
            x1=i-rad;
            x2=i+rad;

```



```

        y1=j-rad;
        y2=j+rad;
        if x1<1
            x1=1;
        end
        if x2>r
            x2=r;
        end
        if y1<1
            y1=1;
        end
        if y2>c
            y2=c;
        end
        erodedImage(i,j) = min(min(filledImage(x1:x2,y1:y2)));
    end
end
tumorOutline = im_tumoralone1 - erodedImage;

axes(handles.axes6);
imshow(tumorOutline);

end

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global luachonseg im_tumoralone area perimeter x y centroid numberOfPixels2
luachonseg = get(handles.popupmenu2,'value');

```

```

% handles = guidata(hObject);

if (luachonseg == 1)
    measurements = regionprops(im_tumoralone, ...
    'area', 'Centroid', 'Perimeter');

    area = [measurements.Area];
    centroid = [measurements.Centroid];
    perimeter = [measurements.Perimeter];

    % Calculate the area, in pixels
    numberOfPixels2 = bwarea(im_tumoralone);
    %area = sqrt(numberOfPixels2);

    %convert into mm
    area = numberOfPixels2 * (0.26458333)^2;
    perimeter = perimeter * 0.26458333;

    % Get coordinates of the boundary in tumor
    structBoundaries = bwboundaries(im_tumoralone);
    xy=structBoundaries{1}; % Get n by 2 array of x,y coordinates.
    x = xy(:, 2); % Columns.
    y = xy(:, 1); % Rows.

    set(handles.text10,'String',area);
    set(handles.text11,'String',perimeter);
    set(handles.text12,'String',centroid);

elseif (luachonseg == 2)
    measurements = regionprops(im_tumoralone, ...
    'area', 'Centroid', 'Perimeter');

```

```

area = [measurements.Area];
centroid = [measurements.Centroid];
perimeter = [measurements.Perimeter];

% Calculate the area, in pixels
numberOfPixels2 = bwarea(im_tumoralone);
%area = sqrt(numberOfPixels2);

%convert into mm
area = numberOfPixels2 * (0.26458333)^2;
perimeter = perimeter * 0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(im_tumoralone);
xy = structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.

set(handles.text10,'String',area);
set(handles.text11,'String',perimeter);
set(handles.text12,'String',centroid);

elseif (luachonseg == 3)
    measurements = regionprops(im_tumoralone, ...
    'area', 'Centroid', 'Perimeter');

    area = [measurements.Area];
    centroid = [measurements.Centroid];
    perimeter = [measurements.Perimeter];

```

```

% Calculate the area, in pixels
numberOfPixels2 = bwarea(im_tumoralone);
%area = sqrt(numberOfPixels2);

%convert into mm
area = numberOfPixels2 * (0.26458333)^2;
perimeter = perimeter * 0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(im_tumoralone);
xy = structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.

set(handles.text10,'String',area);
set(handles.text11,'String',perimeter);
set(handles.text12,'String',centroid);

elseif (luachonseg == 4)
    measurements = regionprops(im_tumoralone, ...
    'area', 'Centroid', 'Perimeter');

    area = [measurements.Area];
    centroid = [measurements.Centroid];
    perimeter = [measurements.Perimeter];

% Calculate the area, in pixels
numberOfPixels2 = bwarea(im_tumoralone);
%area = sqrt(numberOfPixels2);

%convert into mm

```

```

area = numberOfPixels2 * (0.26458333)^2;
perimeter = perimeter * 0.26458333;

% Get coordinates of the boundary in tumor
structBoundaries = bwboundaries(im_tumoralone);
xy=structBoundaries{1}; % Get n by 2 array of x,y coordinates.
x = xy(:, 2); % Columns.
y = xy(:, 1); % Rows.

set(handles.text10,'String',area);
set(handles.text11,'String',perimeter);
set(handles.text12,'String',centroid);

end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axes(handles.axes1); cla(handles.axes1); title("");
axes(handles.axes2); cla(handles.axes2); title("");
axes(handles.axes3); cla(handles.axes3); title("");
axes(handles.axes4); cla(handles.axes4); title("");
axes(handles.axes5); cla(handles.axes5); title("");
axes(handles.axes6); cla(handles.axes6); title("");

set(handles.text10,'String','');
set(handles.text11,'String','');
set(handles.text12,'String','');

```

```

set(handles.text20,'String','');
set(handles.text21,'String','');
set(handles.text22,'String','');
set(handles.text23,'String','');
set(handles.text24,'String','');


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2
luachonseg = get(hObject,'Value');
global inp im_thr im_adjust im_seg level BW SE_erode Idata Idx nn level0 fim im_seg0
if(luachonseg ==1)
    inp = imresize(inp,[256,256]);
    im_thr = imtophat(inp,strel('disk',40));
    im_adjust = imadjust(im_thr);
    level = graythresh(im_adjust);
    BW = im2bw(im_adjust,level);
    SE_erode = strel('disk',3);
    im_seg = imerode(BW,SE_erode);


    axes(handles.axes3);
    imshow(im_seg);

elseif (luachonseg == 2)
    sout = imresize(inp,[256,256]);

```

```

t0 = 10;
th = t0+((max(inp(:)) + min(inp(:)))/2);
for i = 1:1:size(inp,1)
    for j = 1:1:size(inp,2)
        if inp(i,j) > th
            sout(i,j) = 1;
        else
            sout(i,j) = 0;
        end
    end
end
end
hy = fspecial('sobel');
hx = hy';
Iy = imfilter(double(sout), hy, 'replicate');
Ix = imfilter(double(sout), hx, 'replicate');
gradmag = sqrt(Ix.^2 + Iy.^2);
L = watershed(gradmag);
im_seg = label2rgb(L);

axes(handles.axes3);
imshow(im_seg);

elseif (luachonseg == 3)
    Idata = reshape(inp, [],1);
    Idata = double(Idata);
    [Idx, nn] = kmeans(Idata, 3);
    im_seg = reshape(Idx,size(inp));

    axes(handles.axes3);
    imshow(im_seg, []);

```

```

elseif (luachonse == 4)
    inp = imresize(inp,[256,256]);
    fim = mat2gray(inp);
    level = graythresh(fim);
    [im_seg0,level0] = fcmthresh(fim,0);

    axes(handles.axes3);
    imshow(im_seg0)
end
% --- Executes on selection change in popupmenu2.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to text4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text4 as text
%     str2double(get(hObject,'String')) returns contents of text4 as a double

```



```

% --- Executes during object creation, after setting all properties.
function text4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to text5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text5 as text
%         str2double(get(hObject,'String')) returns contents of text5 as a double

% --- Executes during object creation, after setting all properties.
function text5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

```

```

%    See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text6 as text
%    str2double(get(hObject,'String')) returns contents of text6 as a double

% --- Executes during object creation, after setting all properties.
function text6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%    See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)

```

```

% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function text10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function pushbutton7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im_tumoralone im_tumoralone2 im_mask Accuracy Sensitivity Fmeasure Precision MCC
Dice Jaccard Specitivity

[filename, pathname] = uigetfile({'*.png'; '*.jpg'; '*.bmp'; '*.tif'; '*.gif'; '*.jpeg'}, 'Load Mask File');
im_mask = imread([pathname filename]);
im_mask = imresize(im_mask,[256,256]);
im_mask = im2bw(im_mask, 0.5);

```

```
im_tumoralone2 = imresize(im_tumoralone,[256,256]);
im_tumoralone2 = im2bw(im_tumoralone2, 0.5);

[Accuracy, Sensitivity, Fmeasure, Precision, MCC, Dice, Jaccard, Specitivity] =
EvaluateImageSegmentationScores(im_mask, im_tumoralone2);

set(handles.text20,'String',Accuracy);
set(handles.text21,'String',Fmeasure);
set(handles.text22,'String',Precision);
set(handles.text23,'String',Specitivity);
set(handles.text24,'String',Sensitivity);
```