National University of Singapore

School of Computing

Master of Computing CP5103 Project Report

Title: Segmentation and Reconstruction of Jaw Lesions from 3D medical images

Student Name : Pang Weicong

Student ID. : A0297005E

Supervisor : Leow Wee Kheng

Nov, 2024

1: Motivation

Jaw Lesions are growths that occur in the jawbone or the soft tissues of the mouth and face. These lesions can be classified as either odontogenic or nonodontogenic based on their origin. They exhibit a wide range of sizes and degrees of severity. While most of these growths are benign and asymptomatic, they can exhibit aggressive behavior, leading to the expansion, displacement, or destruction of adjacent bones, tissues, and teeth. [1] [2]

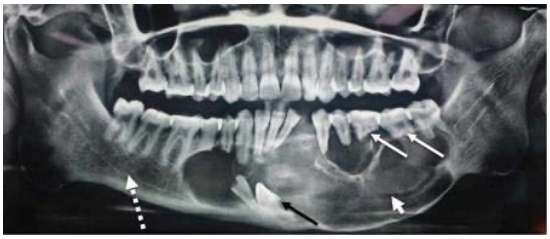

Figure 1.1 Overview of Jaw lesion in Panoramic radiograph

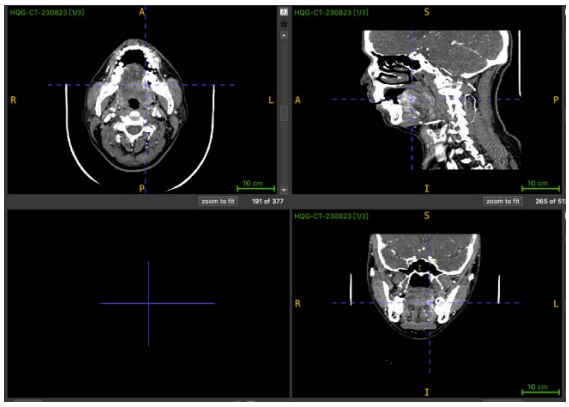

Figure 1.2 CT Scan of a Patient who exists jaw lesion

To effectively treat jaw lesions, it is essential to have a clear understanding of the characteristics about the tumors, including their shapes, sizes, and specific features. This necessitates accurate segmentation and reconstruction of the lesions, which are crucial for effective diagnosis, surgical planning, and treatment of the diseases. Early detection and intervention are important, as most of the jaw lesions can be identified at an earlier stage through a routine radiographic exam called the panoramic radiograph or orthopantomogram. [3]

Traditional methods for jaw lesion segmentation often rely on manual delineation by doctors, which is time-consuming and prone to significant variability among different practitioners. While deep learning approaches like U-Net[4] and TransFuse[5] have demonstrated high accuracy in medical image segmentation, they require large datasets and substantial computational resources. Unfortunately, the datasets of the Jaw Lesions are quite small and it makes deep learning models infeasible to evaluate our model.

Additionally, the Jaw Lesion Segmentation is based on 3D medical images. 3D medical images contain additional spatial information. The volume of data requires advanced algorithms that can effectively process and analyze the interrelationships between each slice. Define boundaries in 3D is often more challenging than in 2D, as objects can be partially obscured or connected in complex ways. Moreover, 3D segmentation often involves integrating data from multiple image modalities (CT, T1mD, T2mD). Each modality has its own characteristics and require different processing steps.

To overcome these challenges, various semi-automatic and fully automatic segmentation methods have been explored. In image segmentation fields, there exists three primary approaches: region-based segmentation, boundary-based segmentation, and hybrid-based segmentation. Boundary-based segmentation, such as level-set segmentation [6], have proven effective for delineating clear edges of anatomical structures. However, boundary-based methods can struggle with complex, ambiguous boundaries where adjacent structures may have similar intensities, making segmentation challenging.

Hybrid-based methods, which combine aspects of both region-based and boundary-based techniques, have gained popularity due to their flexibility and accuracy in handling such complexities. Graph-based approaches, such as GraphCut [7] and GrabCut [8], belong to this hybrid category. GraphCut and GrabCut effectively model relationships between pixels or voxels by incorporating both regional and boundary information, allowing for precise segmentation even in challenging cases where boundaries are not well-defined. The methods are computationally feasible and can be adapted to deal with 3D data, offering a promising solution for the segmentation of jaw lesions.

In this study, we investigate the application of GraphCut and GrabCut algorithms for precise and efficient segmentation of jaw lesions. Additionally, we use ITK-SNAP [9] software to previsualize the position of jaw lesions and to compare our results using ITK-SNAP's semi-automatic snake segmentation algorithm. ITK-SNAP is widely used in medical imaging for its user-friendly interface and capability to perform manual segmentation and automatic segmentation(snake-algorithm). ITK-SNAP allows for interactive visualization and segmentation, facilitating accurate delineation of complex anatomical structures such as lesions. A critical step following segmentation is the 3D reconstruction of the segmented lesions, essential for surgical planning and outcome assessment. For this purpose, we utilize the marching cubes algorithm [10], which is well-suited for extracting a polygonal mesh from the segmented volume. This approach provides high-quality 3D models, enabling clear visualization of the anatomical features. By integrating segmentation and 3D reconstruction, we present a comprehensive pipeline that supports clinicians in visualizing and planning interventions with high precision.

Overall, the contributions of this project are:

- Develop efficient user-interacted GraphCut and GrabCut algorithms for jaw lesion segmentation
- Utilize segmented 2D slices to construct 3D volume, apply marching cubes algorithm to generate 3D jaw lesion mesh

2: Related Work

The advancements in various approaches and algorithms for medical image segmentation significantly enhance diagnostic accuracy and efficiency. By integrating and refining these techniques, the medical imaging process is optimized, offering more precise data support for clinical decision-making.

Above, Section 2.1 will describe the commonly used medical image segmentation software: ITK-SNAP, Section 2.2 will describe the thresholding segmentation algorithm, Section 2.3 will describe the boundary-based segmentation algorithm, Section 2.4 will describe the region-based segmentation algorithm, Section 2.5 will describe the Hybrid-Based Segmentation Method, Section 2.6 will describe the Classification-Based Segmentation Method using Neural Network.

2.1 Thresholding Segmentation Method

Thresholding is one of the simplest segmentation techniques, where pixel intensity values are compared against a global or local threshold to separate objects from the background. Among these methods, Nobuyuki Otsu [11]'s approach provides a nonparametric, unsupervised thresholding mechanism by optimizing the threshold to maximize the separability of pixel intensity classes. This method relies on discriminant analysis principles, using zeroth and first order cumulative moments of the greyscale histogram to enhance inter-class variance. But the method faces challenges with images that exhibit overlapping intensity values or significant inhomogeneity.

2.2 Boundary-based Segmentation Method

Boundary-based segmentation method focus on identifying object boundaries by leveraging image gradients and intensity differences. Among these methods, level set methods [6] and active contour models like the Snake Algorithm have proven effective in accurately capturing complex structures.

Level set methods [6], which extend the principles of active contours, implicitly represent evolving curves or surfaces within a higher-dimensional space and manage topological transformations such as merging and splitting naturally. These methods utilize curvature-dependent speed functions and advanced numerical techniques such as PSC algorithms to simulate complex front propagation accurately. By solving Hamilton-Jacobi-type equations [12], level set methods can capture sharp gradients and cusps in dynamic boundaries, making them well-suited for segmentation tasks.

ITK-SNAP also provides a boundary-based segmentation method called **the Snake Algorithm**. ITK-SNAP is a free, open-source, multi-platform biomedical images segmentation software. ITK-SNAP can be used to segment a variety of 3D medical images, including CT type, MRI type, and PET type. It supports a range of file formats like DICOM [13] and NIFTI [14]. It presents the users with four panels, three of which show orthogonal slice views (axial, sagittal, and coronal), and the fourth, located at bottom left, shows the three-dimensional view of the segmentation.

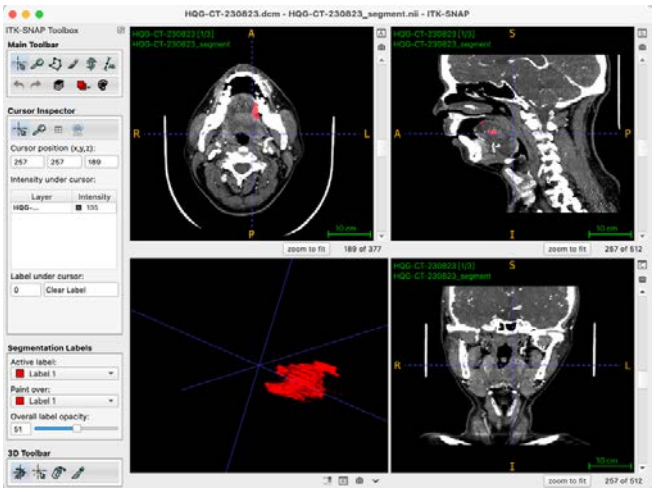

Figure 2.1 Interface of ITK-SNAP

The Snake Algorithm in ITK-SNAP [9], also known as active contour segmentation, is a semi-automatic boundary-based segmentation method that uses an iterative approach to delineate structures based on boundary information. This algorithm initializes with a user-defined contour within the target region and progressively refines the boundary to align with high-contrast edges, where there is a significant intensity difference between the lesion and surrounding tissue. The algorithm relies on both internal forces, which maintain the smoothness and continuity of the contour, and external forces, derived from image gradients, to attract the contour toward object boundaries. The Snake Algorithm is particularly advantageous for medical imaging applications, as it minimizes manual intervention while preserving precision around complex anatomical boundaries.

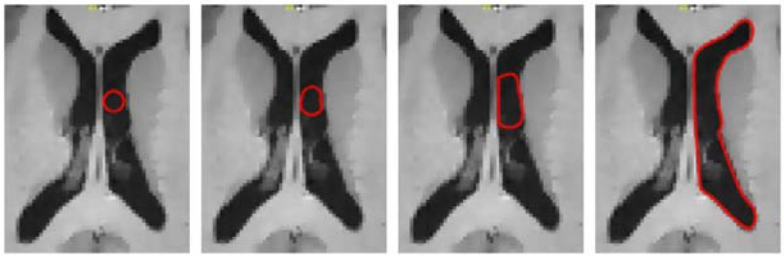

Figure 2.2 Evolution of Contour in Snake Algorithm [9]

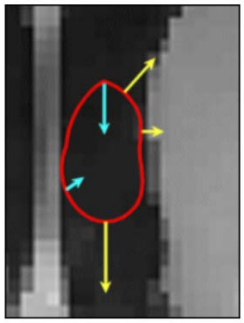

Figure 2.3 Forces acting on the contour in Snake Algorithm [9]

2.3 Region-Based Segmentation Method

Region-based segmentation method, such as region growing [15], are widely used for segmenting regions with homogeneous intensity or texture characteristics. These methods initiate from user-defined seed points and progressively add neighboring pixels that meet specific similarity criteria, typically based on pixel intensity, statistical measures (e.g., average intensity and standard deviation), or other local features such as histograms. Region growing is particularly effective in capturing contiguous regions with similar properties, making it suitable for medical image analysis where anatomical structures often exhibit consistent intensity or texture patterns.

Region-growing techniques assess a local neighborhood around each seed point, typically within a 3*3 window, to determine whether adjacent pixels should be included in the growing region. This neighborhood-based assessment helps ensure that only pixels with similar characteristics are added, thus preserving the boundary integrity of the region. However, these methods can be sensitive to noise, which may lead to unintended region merging if pixel intensity fluctuates in surrounding areas. Accurate seed point selection is crucial, as poorly chosen seeds can result in over-segmentation, capturing areas beyond the intended structure. In cases of low contrast or noisy images, manual adjustments to the seed points are often required to refine the segmentation outcome.

2.4 Hybrid-Based Segmentation Method

Hybrid-based Segmentation method combines region-based and boundary-based approaches to leverage the strengths of both techniques. Graph-based methods belong to hybrid-based segmentation.

Graph-based methods formulate the segmentation problem as a graph partitioning task, where pixels or voxels are represented as nodes connected by weighted edges. The Graph Cuts algorithm [7], proposed by Boykov and Jolly, segments images by finding the minimal cut on a graph that separates foreground and background regions based on function incorporating both boundary and region information.

The Grabcut algorithm [8], which is an extension of Graph Cut that iteratively refines the segmentation with minimal user interaction by incorporating Gaussian Mixture Models [16] to model the foreground and background. It efficiently segments complex images with few user inputs.

2.5 Classification-Based Segmentation Method

One prominent example of using neural Networks for single image segmentation is the **U-Net** architecture proposed by Ronneberger et al. [4]. U-Net (shown in Figure 2.4) was specifically designed for biomedical image segmentation and has become a widely adopted model for single image segmentation tasks due to its effectiveness in learning fine-grained spatial information while preserving contextual features. The architecture consists of a symmetric encoder-decoder structure, where the encoder captures high-level features through a series of down-sampling convolutions, and the decoder reconstructs the segmented output through up-sampling layers. A distinctive feature of U-Net is its skip connections, which link corresponding layers in the encoder and decoder, allowing detailed spatial features from shallow layers to be integrated into the reconstruction process.

This approach enables the model to retain both semantic and location-specific information, making it well-suited for tasks that require precise boundary delineation, such as segmenting organs

or lesions in medical images. Due to its ability to handle limited training data and produce highly accurate segmentation maps, U-Net has been extensively used in medical imaging applications where annotated data are scarce. In subsequent years, U-Net has inspired a variety of modifications and extensions, solidifying its role as a foundational model for single image segmentation tasks in medical imaging.

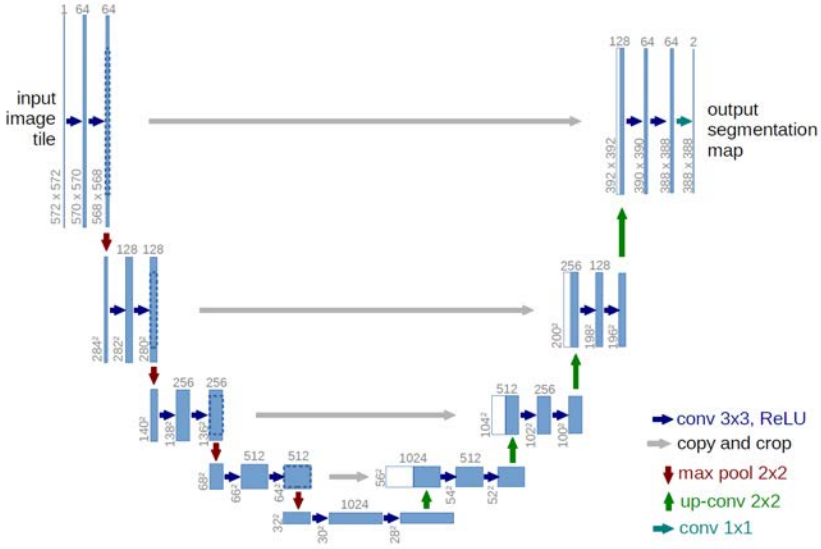

Figure 2.4 Structure of U-Net[4]

3: Methods

Our objective is to develop an effective method for the segmentation and three-dimensional reconstruction of jaw lesions. We employ both the GraphCut and GrabCut Algorithm for segmentation on 2D image slices and utilize the Marching Cube Algorithm for reconstructing 3D model from the segmented slices. Our approach is divided into several stages: data preprocessing, 2D image segmentation, and 3D reconstruction.

In Section 3.1, we introduce the overall workflow of our method and describe the function of each component. Section 3.2 introduces VTK software that is used for manipulation and visualization of medical image data. Section 3.3 and Section 3.4 introduces the implementation of GraphCut and GrabCut algorithm in our jaw lesion segmentation. Section 3.5 introduces the marching cube algorithm that we use for 3D reconstruction.

3.1 Overall Workflow

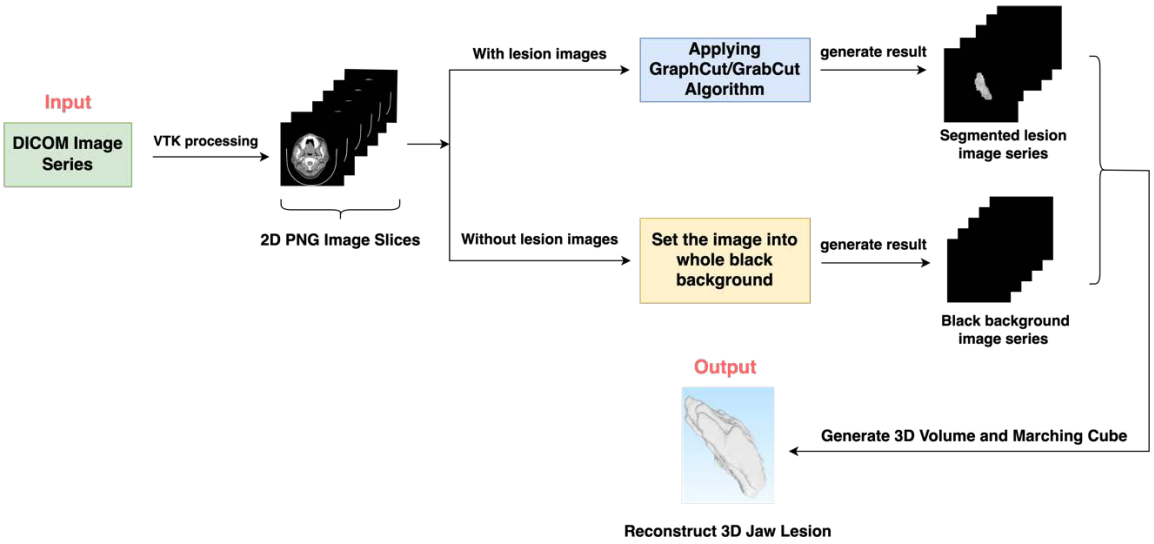

Figure 3.1 Overall of the Jaw Lesion Segmentation and Reconstruction

The workflow for jaw lesion segmentation and 3D reconstruction begins with an input series of DICOM series images. The DICOM series images in our datasets include CT Type, T1mD Type and T2mD Type. The DICOM image series are processed using the **vtkDICOMImageReader**, which reads all slices in the specified directory and produces 3D volumetric images. Then these images are processed slice-by-slice to extract individual 2D slices, which are saved as PNG images to serve as the input for segmentation.

These 2D slices are then segmented using GraphCut or GrabCut algorithms. The segmentation process involves applying user-defined markings to separate foreground from the background. For slices that do not contain jaw lesions, a black background image is generated to ensure consistency across all slices in the volume.

After segmentation, all processed 2D slices including segmented and not segmented are stacked to form a 3D volumetric jaw lesion representation. The stack operation arranges the 2D images along a depth axis. Then the generated 3D volume is passed to the Marching Cube algorithm to extract the 3D surface of the jaw lesion. The Marching Cube algorithm generates a mesh composed

of vertices and triangular faces, and finally the mesh will be processed to allow for advanced jaw lesion mesh visualization and smoothing.

3.2 VTK

The VTK (Visualization Toolkit) is an open-source software suite for the manipulation and visualization of scientific data, offering advanced tools for 3D rendering, interactive 3D widgets, and robust 2D plotting capabilities [17]. While VTK was originally implemented in C++, Python bindings are available, providing convenient access to VTK's powerful functionalities within Python.

In our study, Python VTK was employed to import, manage, and process DICOM Image Series, facilitating the handling of 3D jaw lesion data. Through Python VTK, DICOM series files are loaded as volumetric data, allowing for efficient visualization and manipulation of the jaw lesion scans.

3.3 GraphCut Segmentation Method

GraphCut is widely recognized in computer vision, particularly for tasks such as foreground-background segmentation, stereo vision, and image matting. The technique links image segmentation with the **min-cut algorithm** on a graph, where each pixel in the image corresponds to a vertex, and edges with specific weights represent relationships between neighboring pixels and foreground or background labels.

To formulate the segmentation task as a graph-cut problem, we represented the image as an undirected graph $G = (V, E)$ where V represents vertices corresponding to image pixels and E represents edges between these vertices. Additionally, GraphCut includes two terminal nodes, S and T, which represent the foreground and background classes respectively. Each pixel node is connected to these terminals via two types of edges. The two edges in GraphCut can call them **n-links** and **t-links**. n-links connect neighboring pixels in the image grid and are weighted based on the similarity between connected pixels. t-links connect each pixel node to the terminals S and T, weighted to encourage or discourage the assignment of pixels to either the foreground and background.

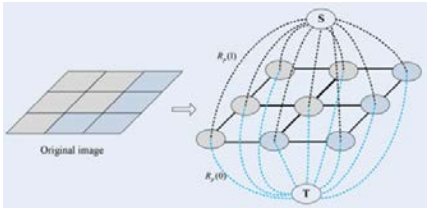

Figure 3.2 Initialization of GraphCut

GraphCut applies energy minimization to achieve segmentation. The segmentation is to find a cut through the graph that minimizes the energy function as listed in Equation 3.1.

$$E(L) = \alpha R(L) + B(L)$$

Equation 3.1 Energy Function in GraphCut

$R(L)$ is the regional term, penalizing the assignment of each pixel to the foreground or background based on its similarity to the respective pixel histograms. $B(L)$ is the boundary term, encouraging smoothness by penalizing large differences between neighboring pixels assigned to different labels. α is a balancing parameter that controls the influence of the regional and boundary terms.

To set these terms, we assigned a label $L = \{l_1, l_2, \dots, l_p\}$ for each pixel p, where $l_p = 1$ for foreground and $l_p = 0$ for background.

The Regional Term $R(L)$ is defined in the following equation:

$$R_p(1) = -\ln P(I_p|obj)$$
$$R_p(0) = -\ln P(I_p|bkg)$$

Equation 3.2 The Regional Term Equation

$P(I_p|bkg)$ and $P(I_p|bkg)$ are the probabilities of pixel p belonging to the foreground and background, respectively, based on intensity histograms. This term penalizes assigning pixels to regions with low probabilities to regions with low probabilities, minimizing when pixels are assigned to the most probable label.

The Boundary Term $B(L)$ is defined by weights $B_{p,q}$ between neighboring pixels p and q:

$$B_{p,q} = \exp\left(-\beta(I_p - I_q)^2\right)$$

Equation 3.3 The Boundary Term Equation

β is a constant based on image intensity statistics. This term enforces smoothness by setting high weights between similar neighboring pixels and low weights between dissimilar ones, thus favoring cuts along high-gradient boundaries.

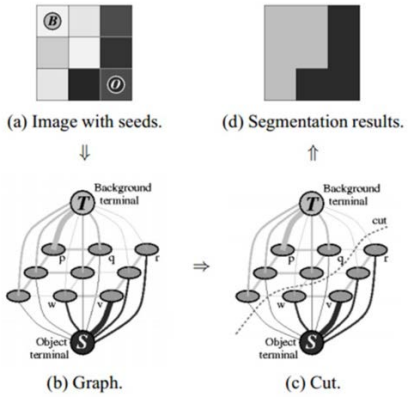

Figure 3.3 The GraphCut representation [7]

The overall GraphCut algorithm can be simplified in the following procedures, as depicted in Figure.. First, the initial image with foreground and background seeds labeled as 'O'(Object) and 'B'(Background), provides initial cues for segmentation. Then, the image is transformed into a graph with each pixel as a node. Two terminals, S (source for foreground) and T (sink for background) are connected to pixel nodes via t-links, while neighboring pixels are connected by n-links. T-links represent pixel association with foreground or background, and n-links enforce spatial smoothness. Next, apply the min-cut algorithm to partition the graph, separating S and T to define the optimal boundary between foreground and background. Last, the final segmentation delineates the object boundary by assigning pixels to foreground or background based on the min-cut result.

To apply the GraphCut algorithm in our jaw lesion segmentation, we include ***numpy, opencv-python, and maxflow*** these software packages. We define the ***InteractiveGraphCut*** class in our code. The user manually labels foreground and background regions by selecting seed points. This labeling initializes the t-links of the graph, assigning high weights to connections between foreground or background pixels and their respective terminals. We build a graph using

maxflow.Graph, where nodes correspond to pixels and n-links and t-links were added based on the intensity similarities and user-provided seeds. The min-cut is determined using the max-flow/min-cut algorithm, dividing the graph into two sets, S and T, representing the foreground and background pixel clusters. The segmentation result is obtained by labeling pixels in foreground and those in background.

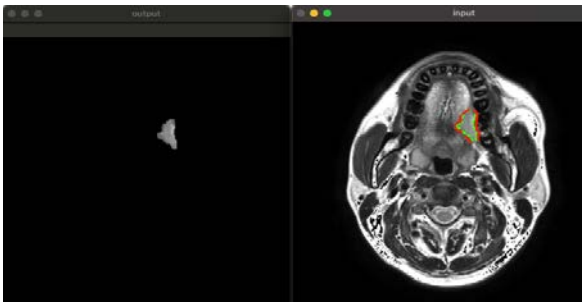

Figure 3.4 GraphCut labeling in our program

3.4 GrabCut Segmentation Method

GrabCut is an extension segmentation algorithm based on the previous GraphCut algorithm. The process begins with the user drawing a rectangle around the foreground region, where pixels inside are initially considered part of the foreground, while those outsides are treated as background. This rectangle provides a rough foreground assumption, which the algorithm refines iteratively by distinguishing probable foreground and background pixels within it.

Within this rectangle, GrabCut utilizes a Gaussian Mixture Model (GMM) [16] to model the color distributions of both the foreground and background. Based on the GMM models, pixels are classified as probable foreground or background, allowing the algorithm to adapt to the unique color characteristics of each region.

Next, GrabCut constructs a graph and applies energy minimization, similar to GraphCut. Each pixel connects to its neighbors and to terminal nodes for foreground and background, with edge weights determined by color similarity and calculated from GMM probabilities. Each pixel p is assigned a probability of belonging to the foreground $P(I_p|obj)$ or background $P(I_p|bg)$, setting the t-link weights accordingly. The min-cut/max-flow algorithm then optimizes the segmentation by minimizing the energy function.

After the initial segmentation, the user can refine the results by marking pixels as definite or probable foreground and background. The algorithm updates the GMMs and segmentation based on these new markings, allowing iterative refinement for improved accuracy.

In our jaw lesion segmentation, we also include ***numpy*** and ***opencv-python*** software packages in our code. The main GrabCut usage is based on the opencv-python.

```
void cv::grabCut ( InputArray          img,
                   InputOutputArray    mask,
                   Rect                rect,
                   InputOutputArray    bgdModel,
                   InputOutputArray    fgdModel,
                   int                 iterCount,
                   int                 mode = GC_EVAL
                 )
Python:
    cv.grabCut( img, mask, rect, bgdModel, fgdModel, iterCount[, mode] ) -> mask, bgdModel, fgdModel
```

Figure 3.5 OpenCV realization of GrabCut Algorithm [18]

The segmentation process begins with a user-defined rectangle around the jaw lesion, where

pixels outside the rectangle are immediately classified as background. This initial rectangle-based segmentation is handled using the **cv.GC_INIT_WITH_RECT** mode.

After the initial segmentation, our implementation allows interactive refinement. Users can label specific regions as definite or probable foreground and background, and the algorithm re-runs in the **cv.GC_INIT_WITH_MASK** mode to update the segmentation based on these labels. This iterative refinement enhances the segmentation accuracy, enabling precise delineation of complex structures within the lesion.

An example of GrabCut usage in our jaw lesion segmentation are listed in Figure 3.6.

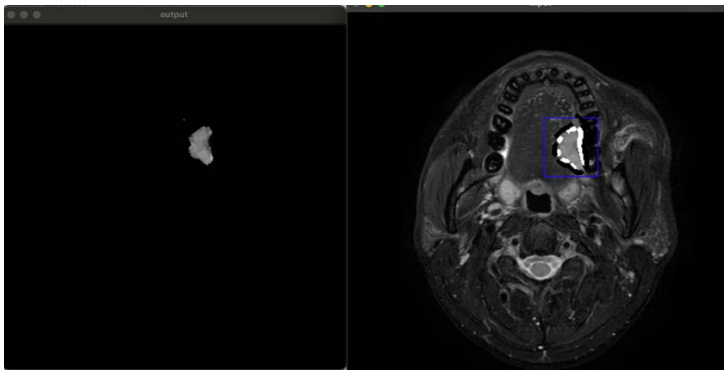

Figure 3.6 GrabCut labeling in our program

3.5 Marching Cube Algorithm

The Marching Cube Algorithm is a classical algorithm for extracting isosurfaces in discrete fields of three-dimensional data. It mainly implements visualization scenes in the medical field, such as 3D reconstruction of CT scans and MRI scans.

In 2D level, the process can be understood through the concept of Marching Squares. Imagine a grid overlay on a 2D image where each grid cell represents a square containing four nodes. Each node is labeled as inside or outside the contour based on a threshold value. This can yield 16 possible configurations for the square ($2^4 = 16$). The algorithm then marches through each square in the grid, identifying edges that intersect the contour. Based on the configuration, it creates line segments within the square to approximate the contour. In the image provided, each square represents different configurations, and the pink regions represent the generated contours within each square configuration.

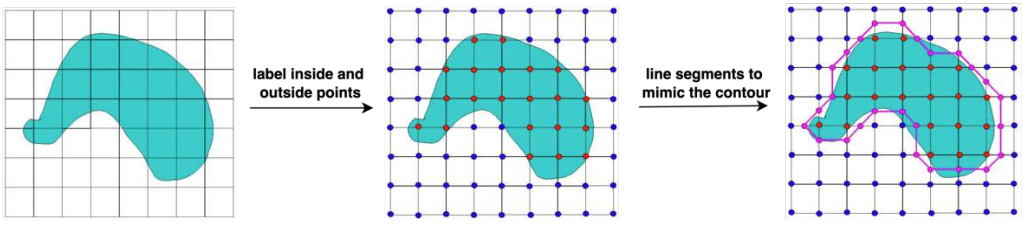

Figure 3.7 2D Marching Squares Algorithm simplified workflow [19]

In 3D level, the algorithm extends this concept to cubes. Each cube has eight nodes, and each node can be classified as inside or outside the surface based on a threshold value. This result in 256 configurations for the cube ($2^8 = 256$). However, due to symmetries, such as rotations, mirroring, and opposite classifications that yield identical triangulations, these configurations can be reduced to 14 unique triangulations.

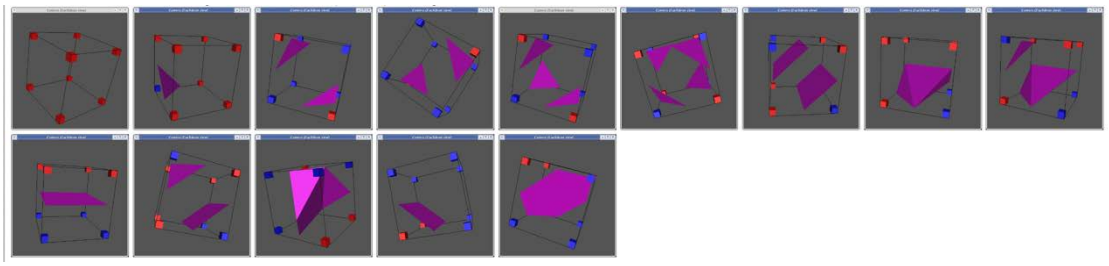

Figure 3.8 14 Unique Triangulations in 3D cube [20]

The marching cube algorithm proceeds in two steps. First, it detects surface intersections within each cube by identifying where the surface crosses the cube's edges. Using the configuration of inside and outside nodes, it references a predefined lookup table containing the 14 unique triangulations. This table specifies the arrangement of triangles required to represent the surface within the cube.

Then using the lookup table, the algorithm generates triangular faces along the intersecting edges. These triangles are created for each cube that contains a portion of the surface, and when all cubes are processed, the triangles are connected to form a continuous 3D surface.

In our jaw lesion 3D reconstruction, we implement the Marching Cube by using the **measure.marching_cubes** function from the **skimage** package. We first load 2D segmented jaw lesion slices and create a 3D Volume which serve as the input for the Marching Cube algorithm. The 3D Volume has pixel intensities that indicate the segmented areas, suitable for surface extraction.

Then we apply the ***marching_cubes*** function to specify the threshold level that determines the surface boundary within the volume. The function outputs vertices and faces to represent the 3D surface. It creates an initial 3D mesh structure from the voxel data in the volume.

Next, we use **PyVista** to process and visualize the mesh. **PyVista** is a powerful Python library that provides an interface to VTK (Visualization Toolkit), offering extensive 3D visualization and mesh manipulation capabilities. It simplifies handling and rendering of complex 3D data in Python. We use PyVista to convert the vertices and faces from generated initial mesh into a PyVista mesh, allowing further processing for smoothing the mesh to enhance the surface quality.

Finally, we apply the rendering method to visualize the final 3D mesh with options for setting the mesh color, background, and opacity.

4: Tests and Discussions

4.1 Dataset and Environment

In our experiments, the dataset only contains two patients. The dataset in each patient consists of three medical image types, with one CT type, one T1mD type, and one T2mD type.

Our implementation was developed using Python 3.11.5. We use PyCharm 2023.3.2 Professional Edition to develop and run our work, as it provides an interactive environment that is ideal for data analysis and visualization. Table 4.1 shows the Python packages and corresponding versions we used in our experiments.

Packages	**Version**
python	3.11.5
vtk	9.3.1
opencv-python	4.10.0.84
numpy	1.26.2
pyvista	0.44.1
scikit-image	0.22.0
maxflow	0.0.1

Table 4.1 List of Packages

4.2 Result and Comparison

In our study on jaw lesion segmentation and 3D reconstruction, we evaluate our reconstructed result using three metrics: Dice Similarity Coefficient (**DICE**), Intersection over Union (**IOU**), and **Hausdorff Distance (HD)**. We compare our result with the doctor's manually segmented result and the Snake Algorithm achieved in ITK-SNAP. While the doctor's result does not serve as an absolute ground truth, it offers a valuable benchmark for assessing the accuracy of our segmentation. The calculations of DICE, IOU, and HD are based on 3D voxel grids, where both the reconstructed and reference segmentations are converted into binary 3D voxel representations. These metrics provide quantitative insights into the overlap, similarity, and boundary alignment, capturing the 3D volumetric characteristics of the lesions and ensuring that our evaluation reflects the full spatial context of the reconstructions.

4.2.1 Evaluation Metrics

DICE measures the overlap between our own segmentation result and manually segmentation result. It is defined in Equation 4.1.

$$DICE = \frac{2|X \cap Y|}{|X| + |Y|}$$

Equation 4.1 Dice Coefficient

where X references our own segmented lesion volume and Y references the doctor's manually segmented lesion volume. A Dice score of 1 indicates perfect overlap, while 0 indicates no overlap.

IOU is another measure of overlap, focusing on the ratio between the intersection and union of the predicted and reference result. It is defined in Equation 4.2.

$$IOU = \frac{|A \cap B|}{|A \cup B|}$$

Equation 4.2 Intersection over Union

IOU provides a stricter overlap measure than DICE by normalizing the intersection over the union of both volumes. IOU score of 1 indicates the best precise segmentation with less background inclusion, while 0 indicates no overlap.

The Hausdorff Distance (HD) measures the degree of boundary misalignment between two sets by identifying the farthest point from one set to the closest point in the other set. Specifically, it calculates the maximum of the shortest distances between each point in one set and the other set. It captures the worst-case scenario of boundary discrepancy, providing an upper bound on the alignment error between the segmented and reference boundaries. It is defined in Equation 4.3.

$$d_{\mathrm{H}}(X,Y) := max\left\{\sup_{x \in X} d(x,Y), \sup_{y \in Y} d(X,y)\right\}$$

$$d(a,B) = \inf_{b \in B} d(a,B)$$

Equation 4.3 Hausdorff Distance Representation

where X and Y are sets of points representing the boundary of the segmented and reference volumes, respectively, sup represents the supremum operator, inf represents the infimum, and where $d(a,B) = \inf_{b \in B} d(a,B)$ quantifies the distance from a point $a \in X$ to the subset $B \in X$.

4.2.2 Evaluation Result

Evaluation Result of All the Data Samples

			GraphCut			GrabCut			ITK-SNAP Snake		
	Data Type	Subject	Dice	IOU	HD	Dice	IOU	HD	Dice	IOU	HD
(1)	CT	HQG	0.42	0.26	19.52	0.43	0.28	16.67	0.73	0.57	20.04
		ZCN	0.41	0.26	7.07	0.34	0.2	12.57	0.76	0.63	6.16
(2)	T1mD	HQG	0.7	0.54	7.81	0.64	0.47	14.56	0.77	0.64	7.14
		ZCN	0.67	0.51	6.71	0.64	0.47	7.81	0.74	0.58	8.25
(3)	T2mD	HQG	0.63	0.46	7	0.62	0.45	7.07	0.67	0.5	13.56
		ZCN	0.62	0.45	8.06	0.65	0.49	7.07	0.74	0.58	10
			(a)			(b)			(c)		

Table 4.2 Results of Various Methods.

(1) CT, (2) T1mD, (3) T2mD. (a) GraphCut, (b) Grabcut, (c) ITK-SNAP Snake

The performance of the GraphCut and GrabCut algorithms was assessed on two jaw lesion datasets. Each dataset is equipped with three medical image types: CT, T1mD, and T2mD.

In comparing the GraphCut and GrabCut implementations, we observe that GraphCut generally demonstrates slightly better performance across most metrics, though the differences are relatively minor. For example, in the HQG-T1mD-230825 dataset, GraphCut achieves a Dice score of 0.701, an IOU of 0.5397, and a Hausdorff Distance of 7.8102, all of which outperform GrabCut's results for the same dataset. Similarly, in the ZCN-T1mD-240403 dataset, GraphCut performs marginally better with a Dice score of 0.6746, an IOU of 0.509, and a Hausdorff Distance of 6.7082 compared to GrabCut.

However, in certain datasets, such as HQG-CT-230823 and ZCN-T2mD-240403, GrabCut achieves slightly better results than GraphCut, indicating that GrabCut can be competitive in specific cases. Overall, while GraphCut tends to perform slightly better, both algorithms yield comparable results, with only small variations across datasets.

In comparing the medical image types, CT scans exhibit the lowest Dice and IOU scores, coupled with the highest Hausdorff Distances, indicating a lower segmentation accuracy compared to MRI-based images. Both T1mD and T2mD images consistently achieve Dice scores above 0.6 and IOU scores around 0.5. In contrast, none of the CT datasets achieve a Dice score above 0.5 or an IOU score above 0.3, and they also present greater Hausdorff Distances. This is likely due to the lower soft tissue contrast in CT. Although T1mD and T2mD both perform better than CT, they both show not good result in IOU, which suggests further improvements are needed in capturing the lesion boundaries.

In comparison with the ITK-SNAP Snake Algorithm, it is evident that ITK-SNAP consistently yields higher DICE and IOU scores across all data types. For instance, in the CT datasets, ITK-SNAP achieves a Dice score of 0.73 and an IOU of 0.57 for HQG-CT-230823, which is significantly higher than the results from both GraphCut and GrabCut. This trend is observed in the T1mD and T2mD Datasets as well, where ITK-SNAP's Dice and IOU scores are higher than those of GraphCut and GrabCut. However, compared the metrics in Hausdorff Distance, the performance of the Snake Algorithm generally is little worse than the GraphCut and GrabCut, but the difference is not too much. The Snake Algorithm also exists the problem that it classifies the background area into

foreground.

4.2.3 Qualitative Comparison of different Methods

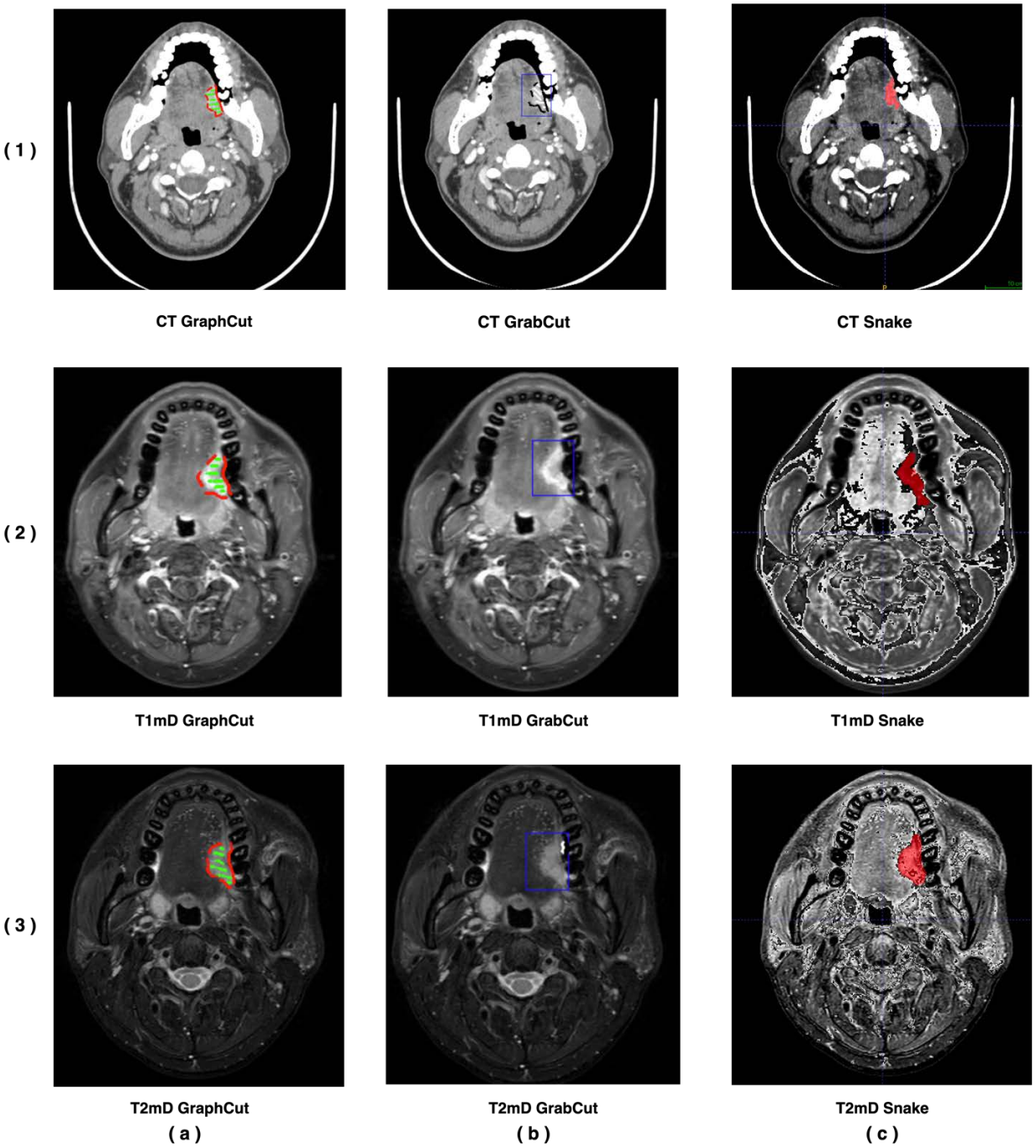

Figure 4.1 Qualitative Results of different methods based on HQG Dataset
(1) CT, (2) T1mD, (3) T2mD. (a) GraphCut, (b) GrabCut, (c) Snake.

Figure 4.1 depict the segmentation results obtained using GraphCut, GrabCut, and the ITK-SNAP Snake algorithm on the HQG dataset. All three algorithms require user-defined input for foreground and background to initiate the segmentation process. In GraphCut, the green color is used to denote the foreground regions, while the red color represents the background, allowing

precise control over the segmentation boundary. For GrabCut, a blue rectangular box is drawn to define the region of interest. In most cases, this initial rectangular selection suffices to accurately segment the lesion. However, in certain situations, additional user intervention is required to specify the foreground and background regions, where white is used for foreground and black for background, to refine the segmentation further. The ITK-SNAP Snake algorithm is employed as a comparison, relying on a boundary-based approach to expand and refine the segmentation. This method is implemented in the ITK-SNAP software, providing an effective way to capture lesion boundaries with minimal user input.

Figure 4.2 presents the results from applying segmentation algorithms using HQG-T1mD dataset, including graphcut, grabcut, and snake algorithm, along with the marching cubes algorithm to generate the 3D reconstructed jaw lesion model.

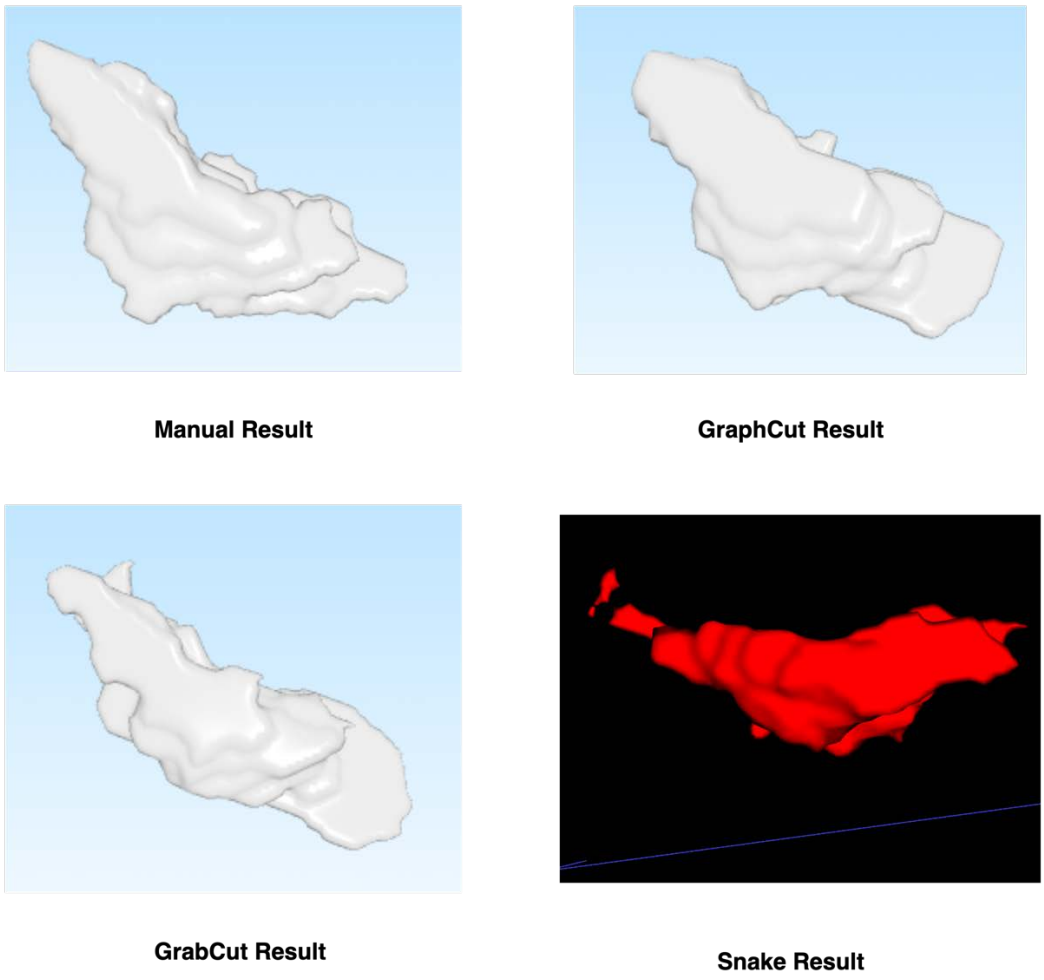

Figure 4.2 3D reconstruction of segmentation results of HQG-T1mD images.
(a) Manual Segmentation, (b) GraphCut, (c) GrabCut, (d) ITK-SNAP Snake

4.3 Discussion

The above results and observations lead to several key insights.

The GraphCut and GrabCut algorithms both provide interactive capabilities, allowing users to define foreground and background regions with flexibility. This interactive approach is beneficial for complex anatomical structures, as it enables the user to guide the segmentation process.

GrabCut's automatic, rectangle-based initialization is particularly advantageous, often yielding good results with minimal input when the foreground is distinct from the background. However, both algorithms encounter challenges in handling high-noise images and low-contrast regions, such as those found in CT scans where the contrast between jaw lesions and surrounding tissue is poor. In such cases, accurate boundary delineation is difficult to achieve. Additionally, GrabCut's rectangle initialization can sometimes mistakenly classify background elements as part of the foreground, requiring further manual refinement to correct the segmentation.

The Marching Cubes algorithm enables the effective reconstruction of 3D structures from segmented 2D slices, producing detailed surface meshes that facilitate spatial analyses crucial for applications like surgical planning. Its robustness, simplicity, and relatively low computational cost make it a widely adopted choice for isosurface extraction. However, a notable limitation of Marching Cubes is the quality of its generated mesh. The algorithm tends to produce a significant number of degenerate and skinny triangles, resulting from its method of classifying grid vertices based on a threshold (isovalue). [21] When a vertex's scalar value matches the isovalue exactly, it can create triangles with all vertices at the same point, leading to degenerate triangles. Skinny triangles, where two vertices are near a grid vertex and one is far, also frequently arise. These mesh artifacts can reduce the overall quality and smoothness of the 3D model, potentially necessitating post-processing steps to correct these issues. Additionally, the Marching Cubes algorithm is highly sensitive to the accuracy of each slice's segmentation; any imperfections in the 2D segmentation can propagate into 3D, impacting the reconstruction fidelity.

In examining the results, it is evident that the CT data consistently shows lower Dice and IOU scores compared to the T1mD and T2mD MRI datasets. This can likely be attributed to the inherent limitations of CT imaging in visualizing soft tissue structures, which results in low contrast between the jaw lesions and surrounding tissues. Unlike MRI, which offers enhanced soft tissue contrast, CT is primarily optimized for visualizing denser structures like bones. Consequently, the low contrast makes it challenging for both GraphCut and GrabCut algorithms to accurately delineate lesion boundaries, leading to suboptimal segmentation performance.

The relatively low IOU scores across all data types further underscore challenges in achieving high overlap between the segmented and reference results. IOU is particularly sensitive to boundary accuracy and overlap, and even minor segmentation inaccuracies can result in significant drops in the IOU score. This suggests that while the algorithms may successfully capture the general shape of the lesions, they struggle to capture the precise boundaries required for high IOU scores.

In comparing the performance of our GraphCut and GrabCut implementations with the Snake Algorithm in ITK-SNAP, it is apparent that Snake Algorithm consistently achieves higher Dice and IOU scores across all datasets. This outcome highlights ITK-SNAP's enhanced capability to capture precise lesion boundaries, especially in complex anatomical regions. The Snake Algorithm's effectiveness can be attributed to its use of internal and external forces that iteratively adjust the contour to align with high-gradient boundaries, an approach that is particularly beneficial in low-contrast regions like those found in CT images. While both GraphCut and GrabCut offer flexibility and interactivity in segmentation, they lack the iterative boundary refinement present in the Snake Algorithm. Consequently, our algorithms may yield good general shapes but are often less precise in complex or low-contrast areas.

5: Conclusions

This study focuses on the jaw lesion segmentation and 3D reconstruction, employing GraphCut, GrabCut, and Marching Cube algorithms to achieve the segmented lesion results. Our work begins with the acquisition of DICOM image series. These DICOM series are processed by Python VTK and converted into 2D PNG slices for GraphCut and GrabCut segmentation. The two algorithms are applied to delineate the lesion boundaries, producing a segmented image series that isolates the lesion region. Once the segmented series is generated, these 2D images are stacked to form a 3D volume and the marching cube algorithm is then applied to this volume to create 3D mesh of the jaw lesion. We then use our own segmentation and reconstruction results to compare with the manual result done by the doctors and the Snake Algorithm in ITK-SNAP to do the evaluation on DICE, IOU, Hausdorff Distance.

Medical image segmentation has always been a highly valuable and challenging area of research in the field of medicine. This study also needs some further improvements.

First, current dataset is too small with only two patients. We should expand the dataset to include a wider range of cases and imaging conditions to better improve the algorithm's generalizability, making them more reliable across diverse clinical scenarios. If we have enough datasets, then we can apply the deep learning approaches, such as U-Net and Transformer, to better capture the information of the images and deal with different modalities.

Second, we should include some advanced preprocessing techniques before we apply GrabCut and GraphCut to the 2D slices images. We can do contrast enhancement to improve lesion visibility in low-contrast images, particularly for CT data. This may be helpful in improving the low score in IOU of all image types. Additionally, refining the user interaction for region marking in GraphCut and GrabCut may improve boundary accuracy by reducing the chance of including extraneous background regions. In terms of postprocessing, smoothing and denoising methods would help refine 3D reconstructions, addressing artifacts introduced during segmentation.

In summary, this study demonstrates that while traditional segmentation and reconstruction algorithms can provide valuable insights into jaw lesion segmentation, further advancements in data diversity, preprocessing techniques, and integration with deep learning models hold significant potential to improve outcomes in jaw lesion segmentation and 3D reconstruction.

References:

[1] Gonzalez-Alva, P. et al. Keratocystic odontogenic tumor: A retrospective study of 183 cases. J. Oral Sci. 50, 205-212 (2008)

[2] Meara, J. G., Shah, S., Li, K. K. & Cunningham, M. J. The odontogenic keratocyst: A 20-year clinicopathologic review. *Laryngoscope* **108**, 280–283 (1998).

[3] Choi, J.-W. Assessment of panoramic radiography as a national oral examination tool: Review of the literature. *Imaging Sci. Dent.* **41**, 1 (2011).

[4] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv, abs/1505.04597*.

[5] Zhang, Y., Liu, H., & Hu, Q. (2021). TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation. *ArXiv, abs/2102.08005*.

[6] Xin Jiang, Renjie Zhang, Shengdong Nie, Image Segmentation Based on Level Set Method, Physics Procedia, Volume 33, 2012, Pages 840-845, ISSN 1875-3892, https://doi.org/10.1016/j.phpro.2012.05.143.

[7] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proceedings of the Eighth IEEE International Conference on Computer Vision*, vol. 1, pp. 105–112, 2001.

[8] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2023. GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. Seminal Graphics Papers: Pushing the Boundaries, Volume 2 (1st ed.). Association for Computing Machinery, New York, NY, USA, Article 62, 593–598. https://doi.org/10.1145/3596711.3596774

[9] http://www.itksnap.org/pmwiki/pmwiki.php

[10] William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. SIGGRAPH Comput. Graph. 21, 4 (July 1987), 163–169. https://doi.org/10.1145/37402.37422

[11] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

[12] Stanley Osher, James A Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics, Volume 79, Issue 1, 1988, Pages 12-49, ISSN 0021-9991, https://doi.org/10.1016/0021-9991(88)90002-2.

[13] *https://www.dicomstandard.org/*

[14] https://nifti.nimh.nih.gov/

[15] R. Adams and L. Bischof, “Seeded region growing,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 6, pp. 641-647, June 1994, doi: 10.1109/34.295913

[16] *Sung H G. Gaussian mixture regression and classification[M]. Rice University, 2004*

[17] https://github.com/Kitware/VTK

[18] https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

[19] https://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

[20] Custodio, L., Pesco, S. & Silva, C. An extended triangulation to the Marching Cubes 33 algorithm. J Braz Comput Soc 25, 6 (2019). https://doi.org/10.1186/s13173-019-0086-6