Biomedical Signal Processing and Control
Volume 91, May 2024, 105901

3D reconstruction of brain tumors from 2D MRI scans: An improved marching cube algorithm

Ruchi Mittal [a], Varun Malik [a], Geetanjali Singla [a], Amandeep Kaur [a], Manjinder Singh [b], Amit Mittal [b]

Reimplementation of a 3D Algorithm (Improved Marching Cubes and CNN)

"3D reconstruction of brain tumors from 2D MRI scans: An improved marching cube algorithm" (Mittal et al., 2024)

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

GBM6700E : 3D Reconstruction from Medical Images

Professor : Lama Séoud

Polytechnique Montréal

Student: Yassine BEN AMMAR
Student ID: 2484854

Contents

1 Introduction

Brain tumors represent a rare but particularly serious pathology ; in 2022, it is estimated that there were more than 320 000 new cases of primary central nervous system tumors and nearly 250 000 associated deaths worldwide [1]. In 2024, approximately 3 300 people per year receive this diagnosis in Canada, with 2 600 deaths [2]. In this context, obtaining a reliable estimation of the tumor's volume and geometry is necessary for staging, surgical or radiotherapy planning, and treatment follow-up.

Magnetic Resonance Imaging (MRI) is the reference examination for studying gliomas. It produces a series of parallel 2D slices that sample a continuous 3D volume. However, representing an intrinsically three-dimensional anatomy through projected 2D views leads to information loss and makes it difficult to perceive complex shapes, even for experienced clinicians. This difficulty in mentally visualizing the tumor in 3D becomes even more pronounced when explaining the situation to a patient or a multidisciplinary team. An explicit 3D visualization of the tumor volume, ideally in the form of a compact triangulated surface, facilitates measurement, planning, and communication.

To address this need, Mittal *et al.* propose a complete two-phase detection–reconstruction pipeline [3]. Phase 1 performs preprocessing of the 2D slices, segmentation using M-BIRCH, followed by a *tumor / non-tumor* classification via a hybrid CNN + Bi-LSTM model whose weights are optimized by a meta-heuristic algorithm (SA-CHOA). Phase 2 relies on an *Improved Marching Cubes* (IMC) method to reconstruct the 3D tumor surface from the positively classified slices. The IMC modifies the classical Marching Cubes in three ways :

(i) intersections are placed at the middle of edges instead of by linear interpolation;

(ii) vertex indices are reused between adjacent cubes through an edge cache;

(iii) a *tightening* step merges certain vertices under angle and distance constraints to produce a lighter mesh without degrading geometry.

In this project, we draw inspiration from this end-to-end philosophy but adapt it to the available data. The public `3d-brain-mri` dataset (subset *UCSF-PDGM*) already provides, for each subject, preprocessed 3D MRI volumes (co-registered, skull-stripped) as well as a reference tumor mask [4, 5]. This allows us to focus our efforts on surface reconstruction rather than detection. We therefore faithfully reimplement the IMC : the reference Marching Cubes, the improved IMC without merging (IMC0), and variants with moderate or aggressive tightening. By using the data of 30 subjects, we compare these methods in terms of mesh complexity (number of vertices and faces) and computation time, with a qualitative analysis through 3D visualization on one representative case.

In parallel, we develop an automatic 2.5D segmentation module inspired by the Phase 1 of Mittal *et al.*, but using a different dataset (*MSD Task01 BrainTumour*) and a simpler architecture [7, 8]. The model ingests sequences of FLAIR MRI slices and combines a per-slice *ContextCNN*, a bidirectional Bi-LSTM to aggregate inter-slice context, and a 2D U-Net modulated by *FiLM* to predict the mask of the central slice [11, 12, 13]. This module serves as a proof-of-concept : it is not quantitatively integrated into the reconstruction on UCSF-PDGM, but it illustrates the feasibility of a future complete pipeline : segmentation $\rightarrow$ IMC $\rightarrow$ 3D surface.

The main contributions of this work are as follows :

1. A detailed reimplementation of Improved Marching Cubes, compatible with the standard Marching Cubes convention, and a systematic study of several tightening configurations on

real glioma data.

2. The development of a 2.5D CNN + BiLSTM segmentation model adapted to the project's hardware constraints, producing 2D masks usable for subsequent reconstruction.

3. An experimental evaluation combining geometric metrics (HD, MSD, area, volume), mesh complexity, and computation time, along with a critical discussion of the trade-offs between geometric fidelity and surface simplification.

2 Materials and Methods

2.1 Pipeline Overview

The project comprises two parts :

- **A) 3D Reconstruction :** From the 3D binary masks of the dataset `3d-brain-mri`, we generate triangulated surfaces of the brain and the tumor. Marching Cubes (MC) serves as the reference. We then reimplement the *Improved Marching Cubes* (IMC) described by Mittal *et al.* from scratch [3]: a version close to the authors (IMC0), followed by two variants simplified by edge merging (IMCdef, IMCext).
- **B) 2.5D Segmentation :** On *MSD Task01 BrainTumour* [7, 8], we train a 2.5D CNN + BiLSTM + U-Net model to predict 2D tumor masks from sequences of FLAIR slices. This module illustrates the detection/segmentation part of the pipeline by Mittal *et al.*, but is not used for the quantitative evaluation of IMC.

The following sections detail the datasets, the segmentation architecture, and the surface reconstruction algorithm.

2.2 Datasets

2.2.1 `3d-brain-mri` (UCSF-PDGM) Dataset for Reconstruction

For 3D reconstruction, we use the public dataset `determined-ai/3d-brain-mri` available on *Hugging Face* [4]. It is a subset (87 subjects) of the *UCSF-PDGM* cohort on pre-operative diffuse gliomas [5]. For each patient, the dataset provides four co-registered 3D MRI volumes, already *skull-stripped*, as well as a reference 3D tumor mask.

In each "`*_nifti`" folder, we select:

- a brain parenchyma mask;
- one or more tumor masks.

Files are loaded in `NIfTI` format with the `nibabel` library [9]. Voxel spacings (in mm) are read from the header and preserved throughout the pipeline in order to work in physical units. Masks are binarized (strict threshold > 0). In the presence of multiple tumor sub-masks, we take their binary union to obtain a global tumor.

2.2.2 MSD Task01 BrainTumour for Segmentation

To train the segmentation module, we use *MSD Task01 BrainTumour* [7, 8], which provides multi-sequence volumes (FLAIR, T1, T1c, T2) and glioma masks. Here we retain only the FLAIR channel, sufficient to highlight the tumor hyperintensity while limiting memory consumption.

Volumes are split into 80 % training cases and 20 % validation cases (patient-level split). This split is performed once with a fixed seed to ensure reproducibility.

2.3 2.5D CNN + BiLSTM Segmentation

2.3.1 Preprocessing and Sequence Construction

Each FLAIR volume is :

- resampled to an isotropic resolution of 1.0×1.0×1.0 mm by linear interpolation (images) or nearest neighbor (labels) [10];
- normalized by intensity *clipping* between the 1st and 99th percentiles within the brain, then standardized (*z-score*) on these voxels;
- cropped and resized : each axial slice is centered, optionally padded, then resized to 224×224 pixels.

To leverage inter-slice context without resorting to a 3D U-Net, we form 2.5D sequences : for a target slice of index z, the model input is the stack of $T = 7$ slices $(z-3, \ldots, z+3)$. The ground truth is the mask of the central slice. To present a sufficient number of slices containing a tumor, sampling favors positions where the mask has positive voxels (over-sampling of tumor slices).

Data augmentations include horizontal and vertical flips, as well as small rotations ($\pm 10°$), applied consistently to the entire sequence.

2.3.2 2.5D Architecture

The architecture follows the spirit of the pipeline by Mittal *et al.* (CNN for visual features, Bi-LSTM for sequential context), but with a lightweight design :

- **Per-slice ContextCNN** : each normalized slice $X_t \in R^{1\times H\times W}$ is processed by a small 2D CNN (conv–ReLU–pool blocks) followed by *global average pooling*, yielding an embedding vector $e_t \in R^d$.
- **Inter-slice aggregation by Bi-LSTM** : the sequence $(e_t)_{t=-3}^{+3}$ is passed to a bidirectional LSTM. We retain only the hidden state corresponding to the central slice, concatenated in both directions:
$$c = [\overrightarrow{h_0}; \overleftarrow{h_0}] \in R^{2h}.$$
The internal LSTM equations (input, forget, and output gates) are standard and are not detailed here.
- **2D U-Net modulated by FiLM** : the central slice X_0 is segmented by a classic 2D U-Net (four encoder/decoder levels) [11]. To inject axial context without exploding the number of parameters, we apply a *FiLM* modulation at the *bottleneck* [12] :
$$B' = \gamma(c) \odot B + \beta(c),$$
where B denotes the feature maps at the deepest level, and γ, β are two linear layers applied to the context vector c, then spatially broadcast. This modulation conditions the 2D segmentation on neighboring slices at a moderate memory cost [13].

The U-Net output is a *logits* map $\hat{y} \in R^{1\times H\times W}$ per slice.

2.3.3 Training and Metrics

The model is trained with the Adam optimizer (*learning rate* 2×10^{-4}, *weight decay* 1×10^{-4}), a *batch size* of 4, for 5 epochs. Training is performed in *mixed precision* (AMP) on GPU [14].

The loss function combines BCE and Dice loss:

$$\mathcal{L}_{\text{tot}} \;=\; 0{,}5\,\mathcal{L}_{\text{BCE}}(\hat{y}, y) \;+\; 0{,}5\,\mathcal{L}_{\text{Dice}}(\hat{y}, y),$$

with

$$\mathcal{L}_{\text{Dice}} \;=\; 1 \;-\; \frac{2\sum \hat{p}\, y}{\sum \hat{p}^{\,2} + \sum y^2 + \varepsilon}, \qquad \text{where } \hat{p} = \sigma(\hat{y}) \text{ and } y \in \{0,1\}^{H \times W}.$$

On validation, we report the binary Dice (threshold 0,5) on 2D slices and save the model with the best mean Dice. For a full 3D volume, predictions can be stacked slice by slice ; this step is used here only illustratively for a reconstruction example.

2.4 3D Surface Reconstruction

2.4.1 Reference Marching Cubes

As a baseline, we use the Lewiner implementation of Marching Cubes provided by `skimage.measure.marching_cubes`. Surfaces are extracted at threshold 0,5. The algorithm is applied directly to the binary masks (brain and tumor) while preserving anisotropic voxel spacing in order to obtain coordinates in millimeters.

By construction, this implementation follows the standard Marching Cubes convention (cube with 8 vertices, 12 edges, index tables from the original description by Lorensen and Cline, relayed by Paul Bourke's tables [6]).

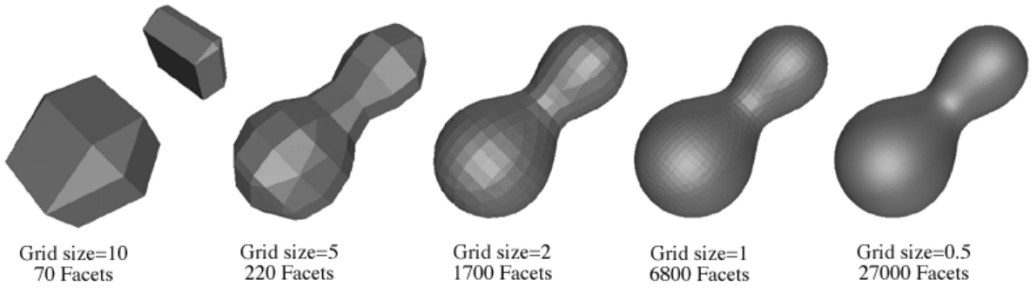


Figure 1: Exemple of marching cubes [6].

2.4.2 Improved Marching Cubes IMC0 (without tightening)

We then reimplement IMC according to the description by Mittal *et al.* [3], explicitly relying on Paul Bourke's `edgeTable` and `triTable` [6]. For each cell with integer coordinates (x, y, z) :

1. Compute the cube index by reading the 8 voxels at the vertices (binary values 0 or 1) and building an 8-bit integer.
2. Ignore cases 0 and 255, where the surface does not cross the cell.
3. Place intersection vertices at the middle of edges : for each edge referenced in `triTable`, the point is positioned exactly at the center of its two corresponding corners, then converted to world coordinates via spacings (s_x, s_y, s_z). In accordance with the paper, no scalar

interpolation is performed ; the high resolution of the data makes this approximation negligible.

4. Reuse vertices between adjacent cubes via a global edge cache : each edge is identified by the two integer coordinates of its corners, sorted to guarantee uniqueness. If a vertex has already been created for this edge, its index is reused, which stitches the cells and avoids duplications.

The resulting variant, without further simplification, is denoted **IMC0**.

2.4.3 Mesh Tightening (IMCdef, IMCext)

To reduce the number of triangles without significantly changing the geometry, we apply a tightening by edge fusion, inspired by Section 4.4.3 of Mittal *et al.* [3]

The process proceeds in iterative passes :

1. Compute face normals by cross product, then vertex normals by area-weighted averaging over adjacent faces.

2. Extract the list of unique edges (triangles $\rightarrow$ 3 edges $\rightarrow$ deduplication).

3. For each edge (i, j), evaluate three constraints :

   - **Angle constraint (MAC)** : the angle $\alpha(n_i, n_j)$ between the vertex normals must be below a threshold MAC.
   - **Distance constraint (MDC)** : we use the robust distance proposed in the paper (equation 30), which combines Euclidean distance and maximum per-coordinate difference. Fusion is allowed only if $d \leq$ MDC.
   - **Boundary constraint** : vertices belonging to a boundary edge (touching a single face) are not merged.

4. If at least two out of the three constraints are satisfied, the two vertices are merged at their midpoint $(v_i + v_j)/2$.

To avoid inconsistent fusion chains, we use a *Union–Find* structure that ensures a vertex participates in at most one merge per pass. After each pass, degenerate faces are removed, indices are compacted, and duplicates eliminated.

We evaluate three configurations :

- **IMC0** : IMC without tightening (IMC reference).
- **IMCdef (moderate)** : MAC = 20°, MDC = $1.2 \times \min(s_x, s_y, s_z)$, 2 passes.
- **IMCext (aggressive)** : MAC = 35°, MDC = $2 \times \min(s_x, s_y, s_z)$, 6 passes.

These parameters are expressed in millimeters to remain consistent with the physical units of the volumes.

2.4.4 Geometric Measures and Surface Distances

For each surface (brain, tumor) and each variant (MC, IMC0, IMCdef, IMCext), we compute with `trimesh` :

- the number of vertices V and faces F;

- the total area (mm^2);
- the volume (mm^3), when the mesh is *watertight*;
- the number of connected components.

Geometric fidelity is assessed relative to MC via :

- the **Hausdorff distance (HD)** (symmetric);
- the **Mean Surface Distance (MSD)**, the average of directed distances in both directions.

To approximate these distances, we sample $N = 50{,}000$ points on each surface. Sampling proceeds in two steps :

1. over-sampling by area ($\approx 6N$ points) via `trimesh.sample.sample_surface`;
2. *Poisson-disk* selection based on a grid : the 3D space is discretized into voxels of size r derived from the area and N according to :

$$r \approx \sqrt{\frac{\text{area}}{\pi N}} \times 0.9,$$

then points are selected so as to respect a minimum distance r.

Nearest distances are then computed by *k-d trees* (`scipy`) in each direction, and aggregated (maximum for HD, mean for MSD).

2.4.5 Visualization and Software Environment

Surfaces are visualized with `PyVista` as 2×2 panels (MC / IMC0 / IMCdef / IMCext). The brain mask is rendered in semi-transparent gray, the tumor in opaque red, and the views are linked to allow direct visual comparison.

The implementation is carried out in `Python 3` (VS Code) with the following libraries : `numpy`, `scipy`, `nibabel`, `scikit-image`, `trimesh`, `pyvista`, `torch`, and `matplotlib`.

Experiments are conducted on a computer with a CUDA-compatible GPU (`RTX4060Laptop`). All sources of randomness (split, surface sampling) are initialized with the same seed (`RANDOM_SEED = 42`), a value chosen according to the "machine learning tradition" inspired by Douglas Adams, in order to ensure reproducibility.

3 Results

3.1 2.5D CNN+BiLSTM Segmentation (MSD Task01 BrainTumour)

Training of the 2.5D model is performed on the 80/20 split described in Section 2.2. The combined BCE+Dice loss decreases steadily over the epochs, while the validation Dice stabilizes at 0.721. The time of training was about 10 hours in this case (2h per epoch), due to material constraints (limited performance to avoid overheating for a long time).

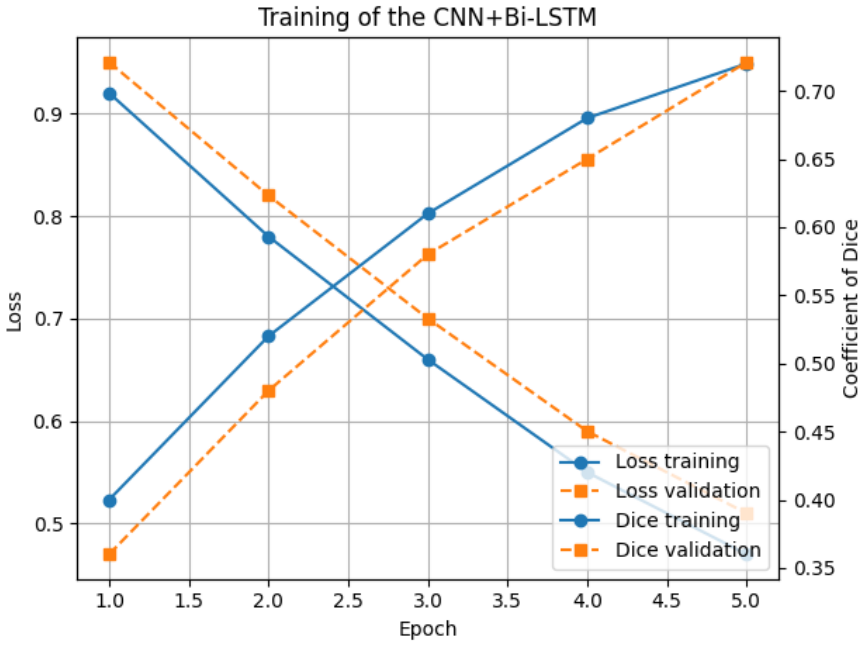
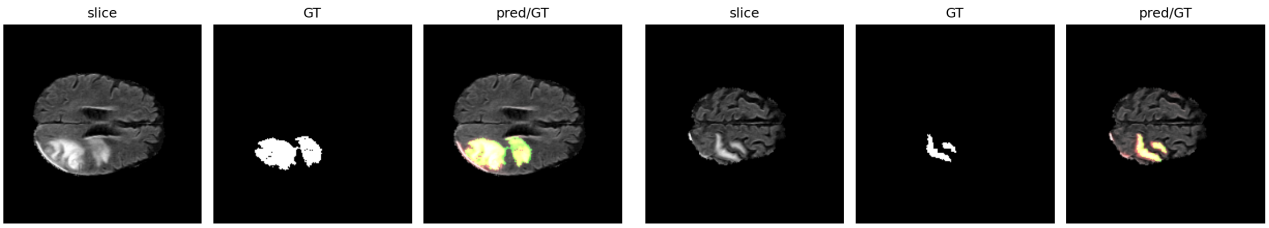
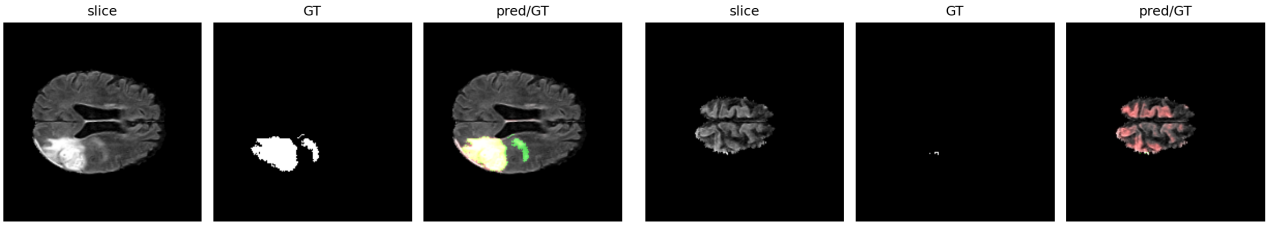


Figure 2: Loss and Dice per epoch



(a) Case 1 : good overall overlap

(b) Case 2 : slight false positives



(c) Case 3 : region with false negatives

(d) Case 4 : over-segmentation

Figure 3: 2D segmentation (prediction/GT overlay). GT in green, prediction in red.

Figure 3 illustrates several qualitative examples on the validation set. For each case, we show :

- the input FLAIR slice (*"slice" column*);
- the ground-truth mask (*"GT" column*);
- the prediction / GT overlay (*"pred/GT" column*), where GT is in green and the prediction in red.

The model correctly localizes the main tumor region and covers most of the GT, with some false negatives at the lesion boundary (white areas not covered). However, it sometimes produces slight false positives in the periphery, typically on FLAIR hyperintense areas close to the tumor or on the ventricular wall.

Figure 3d shows a typical failure case : the GT contains only a very small focus (a few pixels), while the model segments large bilateral cortico-subcortical regions. This behavior suggests that the network learned a large tumoral brain bias from dominant examples and struggles with very small or low-contrast lesions, which limits its direct clinical usability.

Overall, these results indicate that the 2.5D module manages to capture the general tumor location in many cases, but remains insufficiently robust and would require refinement (architecture, losses, modality mix) before being used for systematic 3D reconstruction.

3.2 IMC Reconstruction on a UCSF-PDGM Subject

3.2.1 Mesh Statistics

Table 1 and Table 2 summarize the extracted mesh complexity for the brain and tumor of a UCSF-PDGM subject, for the MC, IMC0, IMCdef, and IMCext methods. Table 3 shows the means of the reduction factors (30 subjects).

Table 1: Brain : mesh complexity.

Method	**Verts**	**Faces**	**Area (mm^2)**
MC	805 888	1 661 856	557 548
IMC0	787 902	1 559 064	533 715
IMCdef	236 700	463 236	417 089
IMCext	41 778	79 802	302 701

Table 2: Tumor : mesh complexity.

Method	**Verts**	**Faces**	**Area (mm^2)**
MC	11 249	22 550	8 259
IMC0	11 240	22 476	8 243
IMCdef	3 310	6 613	7 606
IMCext	299	593	6 711

Table 3: Reduction factors based on MC, average for 30 cases.

Brain			
Method	Verts	Faces	Area (mm^2)
IMC0	1.02	1.07	1.04
IMCdef	3.40	3.59	1.34
IMCext	19.29	20.82	1.84

Tumor			
Method	Verts	Faces	Area (mm^2)
IMC0	1.003	1.002	1.00
IMCdef	3.40	3.41	1.09
IMCext	37.62	38.03	1.23

Observations :

- **IMC0 vs MC :** IMC0 is very close to MC : the number of faces decreases slightly ($-6,5\,\%$ for the brain, $-0{,}2\,\%$ for the tumor) and the area remains almost identical.
- **IMCdef (moderate) :** Reduction by a factor of $\approx 3{,}6$ for the brain and $\approx 3{,}4$ for the tumor, with a similar area (about $-25,4\,\%$ on the brain, $-8,3\,\%$ on the tumor).
- **IMCext (aggressive) :** Much stronger simplification : factor ≈ 21 on the brain and ≈ 38 on the tumor. The area decreases by about $45,7\,\%$ (brain) and $18,7\,\%$ (tumor), indicating more aggressive smoothing.

Topology (watertightness and components) : Watertightness varies little ; brain meshes remain non-watertight regardless of the method, directly reflecting the input masks. For the tumor, IMC0 produces a watertight mesh with 2 components, whereas MC, IMCdef, and IMCext show 5 to 7 components, indicating the appearance/disappearance of small islands (often related to noise in the masks).

3.2.2 Surface Distances and Runtimes

Table 4 presents the surface distances (symmetric HD and MSD) between MC and the IMC variants, and the runtimes are summarized in Table 5.

Table 4: Distances to MC (symmetric HD and MSD, in mm).

Region	Method	HD (mm)	MSD (mm)
Brain	IMC0	6,79	1,22
	IMCdef	11,27	1,22
	IMCext	17,71	1,34
Tumor	IMC0	0,63	0,18
	IMCdef	1,48	0,21
	IMCext	2,98	0,51

For the **brain**, the MSD remains remarkably stable around 1,2 mm, on the order of a voxel, even for IMCext. The HD increases with the level of tightening (from 7 mm to 18 mm), corresponding to a few distant vertices (highly curved areas or small islands disappearing after merging).

For the **tumor**, distances are much smaller : MSD remains $< 0{,}6$ mm for all variants, with a maximum HD < 3 mm for IMCext. In other words, even very aggressive simplification preserves geometry close to MC at the scale of a few millimeters.

Table 5: Surface extraction times (in seconds) for MC and IMC0 on the same subjects.

Region	MC (s)	IMC0 (s)
Brain	0,54	19,60
Tumor	0,11	0,37

Runtimes : The Python implementation of IMC is significantly slower than the optimized C implementation of MC in `skimage` : MC takes 0,54s for the brain and 0,11s for the tumor, versus 19,6s and 0,37s for IMC0. This gap mainly reflects the difference in implementation optimization levels rather than an intrinsic shortcoming of the IMC algorithm.

3.2.3 Qualitative Visualization

Figure 4 shows the reconstruction of the brain and tumor for MC, IMC0, IMCdef, and IMCext, rendered in 3D (brain in semi-transparent gray, tumor in opaque red). Figure 5 shows the brain only, in opaque gray, for better visualization.

Visually:

- **IMC0** is indistinguishable from MC to the naked eye.
- **IMCdef** already smooths part of the surface noise while preserving the tumor's overall shape ; contours remain detailed.
- **IMCext** produces a much smoother surface, with the disappearance of many small irregularities and islands. The global shape remains correct but fine details (local indentations, lobulations) are attenuated.

These observations are consistent with the area and HD/MSD measurements described above.

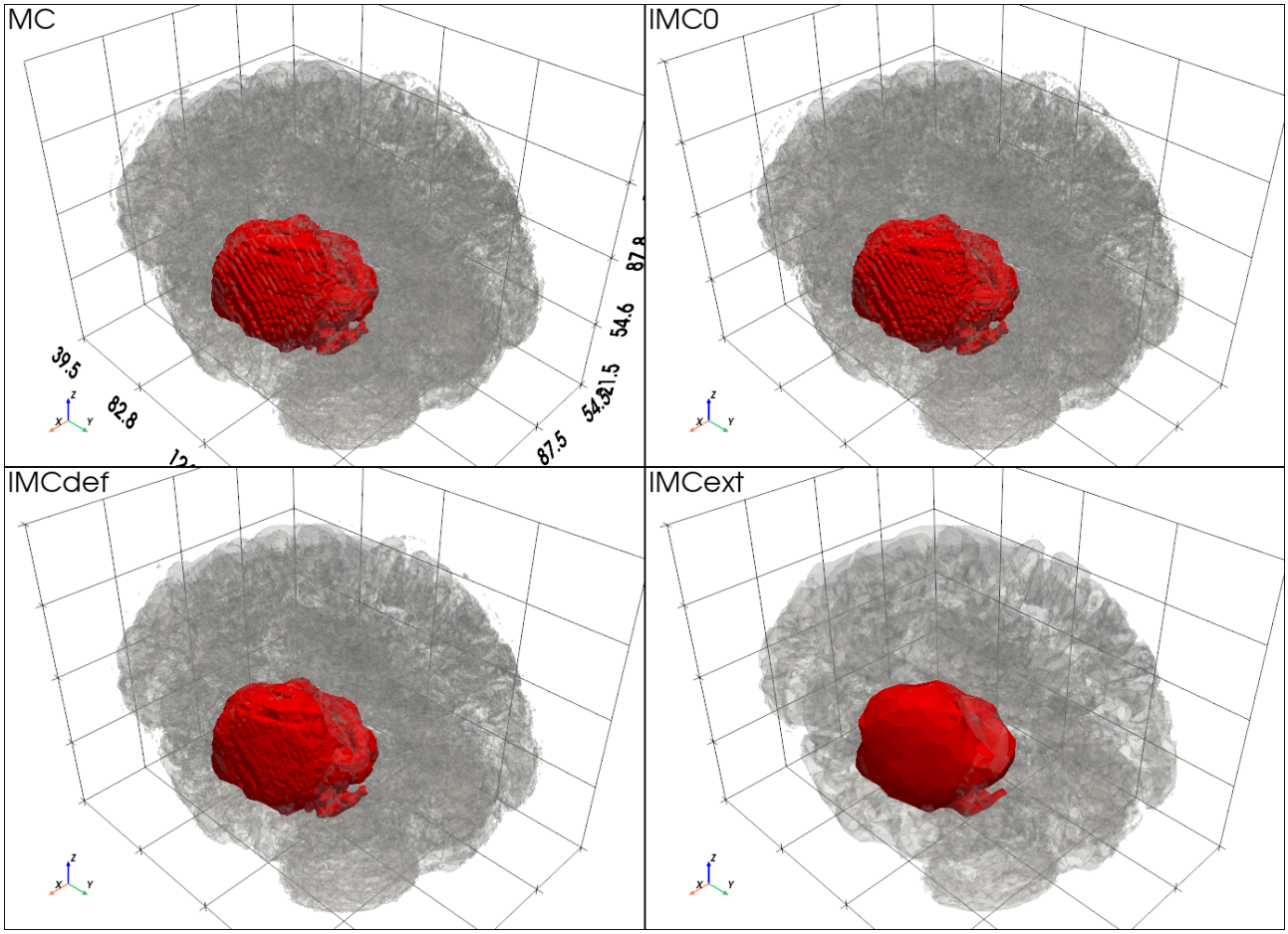


Figure 4: IMC reconstruction (tumor in red).

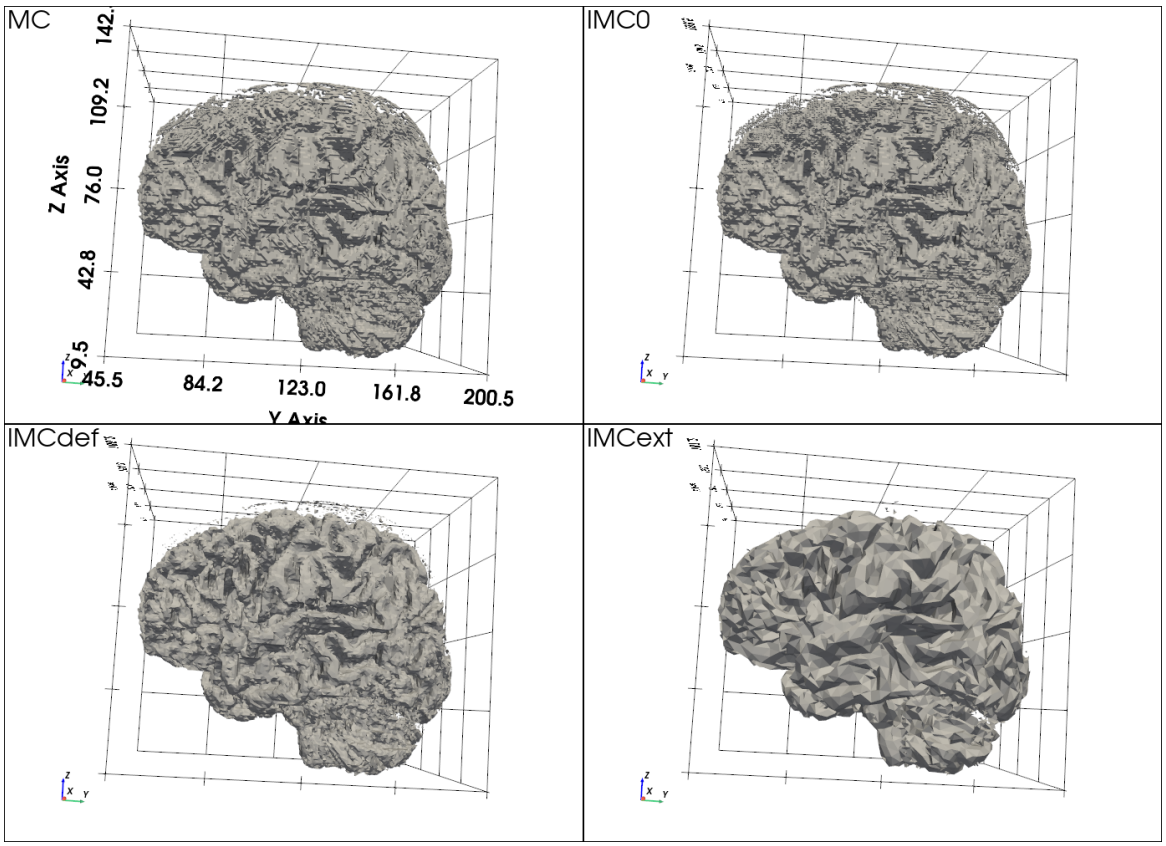


Figure 5: Brain reconstruction.

3.3 Attempted Reconstruction with the CNN

To verify that the complete chain 2.5D segmentation to 3D reconstruction works in practice, we carried out a reconstruction trial from a mask predicted by the CNN on a validation volume.

Figure 6 shows the reconstructed surface of the tumor predicted by the CNN (and the associated brain). We obtain a coherent three-dimensional structure, with relief and indentations comparable to those observed on manually segmented tumors: the algorithm therefore behaves as expected on a mask produced by the network.

However, the overall shape appears larger in volume and much smoother than that reconstructed from the GT, reflecting the false positives and spillovers observed in the 2D overlays.

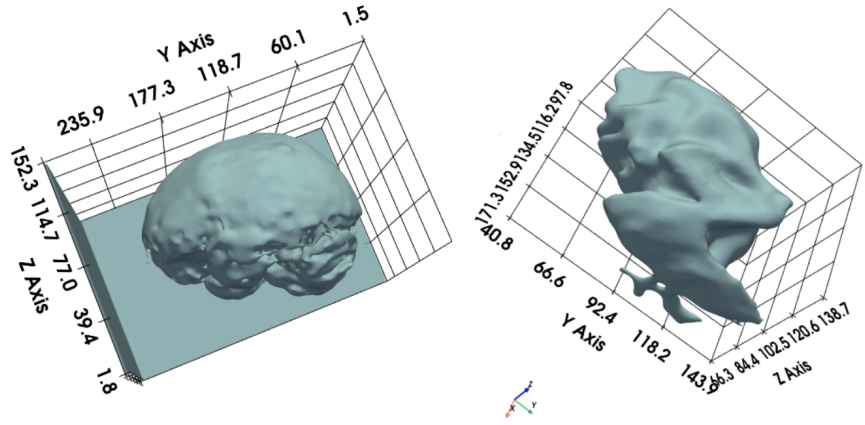


Figure 6: Reconstruction from the CNN predictive mask (left : brain, right : tumor).

4 Discussion

4.1 Assessment of the 2.5D Segmentation

The qualitative examples show that the 2.5D CNN+BiLSTM+U-Net model effectively exploits inter-slice context to track the tumor from one slice to the next, and often manages to cover the main lesion mass, which is promising for future 3D reconstruction.

However, it remains fragile for small or low-contrast lesions (under-segmentation or complete failure), or for non-tumoral hyperintensities (diffuse edema, artifacts, superficial cortex) leading to extensive false positives.

These limitations are consistent with the choice of a single modality (FLAIR only) and a deliberately compact architecture to respect memory constraints. Several immediate avenues for improvement are :

- adding T1/T1c/T2 channels to enrich multimodal contrast;
- using a more suitable loss (Dice + boundary term);
- morphological post-processing of predictions to remove noise;
- moving to a lightweight 3D U-Net on downsampled patches.

In the context of this project, the CNN mainly serves as a *proof-of-concept* illustrating the feasibility of a complete "segmentation $\rightarrow$ IMC" pipeline. A systematic volumetric evaluation (3D Dice per patient, Hausdorff on reconstructed surfaces) would be necessary to assess its clinical relevance.

4.2 Geometry / Complexity Trade-off in IMC

The results show that the IMC algorithm, as reimplemented, achieves its main objective :

- **IMC0** reproduces a surface almost identical to MC (MSD on the order of the voxel, area almost unchanged) while being conceptually better suited to subsequent triangle merging (mid-edge vertex placement, global cache).
- **IMCdef** offers an attractive compromise : a reduction by a factor of $\sim$3,5 in the number of faces for an unchanged MSD and still moderate HD. This is a natural configuration for visualization and simple volume/area computations.
- **IMCext** illustrates the extreme limit of tightening : extremely compact meshes but higher HD and a clearly reduced area. This configuration could be useful for applications where only the global shape matters (educational context or interactive rendering on very limited hardware).

The slightly chaotic behavior of the number of components (more islands for certain variants) underlines the sensitivity of topology to small irregularities in the binary masks. Volumetric pre-smoothing or removal of very small components (by volume threshold) prior to reconstruction could improve topological consistency.

4.3 Positioning Relative to the Reference Paper

Compared to the pipeline of Mittal *et al.*, this project faithfully reproduces the IMC part: Paul Bourke's convention, mid-edge vertex placement, global edge cache, and tightening guided

by the same angle and distance constraints [3, 6]. Conversely, it greatly simplifies the segmentation part : no M-BIRCH or SA-CHOA optimization, but a more classical 2.5D architecture (CNN+BiLSTM+U-Net) trained by standard gradient descent. Finally, the data context differs: here, 3D masks are provided for UCSF-PDGM, which allows us to focus the analysis on the geometric quality of the reconstruction rather than on tumor detection [4, 5].

This difference makes a direct comparison of numerical performance tricky, but shows that the idea of IMC remains relevant and transferable to other datasets and segmentation pipelines.

Quantitative Comparison : In their Table 4, Mittal *et al.* compare the number of vertices generated by MC and by IMC on five test cases. For each volume, IMC slightly reduces the number of vertices compared to MC : for example, from 177 728 to 176 218, or from 186 091 to 182 736. The reported gain ranges from 0,8 % to 2,9 % fewer vertices, so about 1,85 % on average [3].

On our UCSF-PDGM subject, IMC0 (IMC without tightening) is in the same ballpark : for the brain, moving from MC to IMC0 reduces the number of vertices from 805 888 to 787 902, so a reduction of about 2,2 %. This consistency of gain, despite different data and resolutions, confirms that our reimplementation of the pure IMC phase (mid-edge placement + cache following Bourke's convention) matches the description in the paper.

The main difference comes from our tightening step. Mittal *et al.* merely show qualitatively that edge fusion simplifies the mesh (MC/IMC comparison figure), without detailed quantitative measurement (no number of faces after simplification, nor surface distance relative to MC). In our work, we explicitly explore two regimes :

- **IMCdef** : about a 70 % reduction in the number of brain vertices, while keeping MSD = 1,2 mm relative to MC and HD compatible with sub-voxel variation at the tumor scale.
- **IMCext** : more than a 94 % reduction in vertices, at the cost of a moderate increase in HD and a visible smoothing of details.

Thus, whereas the paper limits itself to showing that IMC produces slightly fewer triangles than MC, our project precisely quantifies the complexity–geometric fidelity trade-off over a much broader spectrum, introducing surface distance metrics (HD/MSD) and making explicit the role of the MAC and MDC parameters. We therefore qualitatively confirm the observations of Mittal *et al.* (cleaner meshes, smoothing effect) while demonstrating that, under suitable settings, IMC can go far beyond the slight complexity gain reported in the original article.

4.4 Limitations and Future Work

Among the main limitations, note that the IMC implementation in Python is not optimized and does not reflect the runtime that could be achieved with a multi-threaded C++ version. Moreover, the UCSF-PDGM masks are used as is, without volumetric smoothing or systematic removal of small components. The topological differences (number of islands, watertight nature) between MC and the IMC variants are therefore partly due to label noise. More advanced preprocessing would likely yield more regular surfaces and better isolate the intrinsic effect of tightening.

Conversely, the project opens several avenues to generalize IMC reconstruction to other structures (edema, necrosis, ventricles) and other modalities. One could also integrate the 2.5D segmentation into an end-to-end pipeline applied directly to UCSF-PDGM, with joint Dice/HD/MSD evaluation. Finally, it would be possible to explore more advanced versions of IMC (adaptive

fusion criteria, explicit curvature preservation) or alternative reconstruction methods (level sets, implicit surfaces, surface networks).

5 Conclusion

This project aligns with the objective of better leveraging 3D MRI data for the management of brain tumors. Using public datasets, we reimplemented the *Improved Marching Cubes* (IMC) algorithm proposed by Mittal *et al.* [3], including mid-edge vertex placement, a global edge cache, and tightening controlled by angle and distance constraints. We then quantitatively studied several simplification settings on a UCSF-PDGM case [4, 5], comparing MC, IMC0, IMCdef, and IMCext in terms of complexity, area, volume, surface distances, and runtime. Finally, we developed a 2.5D CNN+BiLSTM+U-Net segmentation module [11, 12, 13], illustrating the role of a sequential model for automatic tumor detection from sequences of FLAIR slices.

The results show that a moderate IMC setting (IMCdef) can reduce the number of triangles by a factor of $\sim$3,5 while maintaining an MSD on the order of a voxel and an area close to that of MC, making it a good compromise for visualization and geometric measurements. The more aggressive version (IMCext) illustrates the possibility of extremely compact meshes at the cost of more pronounced smoothing.

The 2.5D segmentation module, still imperfect, nevertheless confirms the value of exploiting inter-slice context and lays the groundwork for a complete pipeline :
raw MRI $\rightarrow$ automatic segmentation $\rightarrow$ 3D reconstruction, which would be interesting to develop further.

References

[1] Ferlay J, Ervik M, Lam F, Laversanne M, Colombet M, Mery L, Piñeros M, Znaor A, Soerjomataram I, Bray F (2024). *Global Cancer Observatory: Cancer Today*. Lyon, France: International Agency for Research on Cancer. Disponible sur: `https://gco.iarc.who.int/today`. Consulté le 2025-10-04.

[2] Canadian Cancer Society. *Brain and spinal cord cancer statistics*. Disponible sur: `https://cancer.ca/en/cancer-information/cancer-types/brain-and-spinal-cord/statistics`. Consulté le 2025-10-05.

[3] Mittal R, Malik V, Singla G, Kaur A, Singh M, Mittal A. *3D reconstruction of brain tumors from 2D MRI scans: An improved marching cube algorithm*. Biomedical Signal Processing and Control (2024). `https://doi.org/10.1016/j.bspc.2023.105901`.

[4] determined-ai. *3d-brain-mri — Datasets at Hugging Face*. `https://huggingface.co/datasets/determined-ai/3d-brain-mri`.

[5] The Cancer Imaging Archive (TCIA) (14 septembre 2025). *UCSF-PDGM — The Cancer Imaging Archive (TCIA)*. `https://www.cancerimagingarchive.net/collection/ucsf-pdgm/`.

[6] Bourke P. *Polygonising a scalar field (Marching Cubes)*. `https://paulbourke.net/geometry/polygonise/`.

[7] Antonelli M, et al. *The Medical Segmentation Decathlon*. Nature Communications, 13:4128 (2022).

[8] Medical Segmentation Decathlon — site officiel. `https://medicaldecathlon.com`.

[9] NiBabel Documentation. *SpatialImage & API*. `https://nipy.org/nibabel/`.

[10] SciPy Documentation. *ndimage.zoom*. `https://docs.scipy.org/`.

[11] Ronneberger O, Fischer P, Brox T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv:1505.04597 (2015).

[12] Perez E, Strub F, de Vries H, Dumoulin V, Courville A. *FiLM: Visual Reasoning with a General Conditioning Layer*. arXiv:1709.07871 (2017).

[13] Emre R, et al. *Pretrained Deep 2.5D Models for Efficient Predictive Modeling from Retinal OCT*. MICCAI OMIA Workshop (2023).

[14] PyTorch Documentation. *torch.amp*. `https://pytorch.org/docs/stable/amp.html`.