# Interactive Volumetric Region Growing for Brain Tumor Segmentation on MRI using WebGL

Jonas Kordt
Hasso Plattner Institute - University of Potsdam
Potsdam, Germany
jonas.kordt@student.hpi.de

Paul Brachmann
Hasso Plattner Institute - University of Potsdam
Potsdam, Germany
paul.brachmann@hpi.de

Daniel Limberger
Hasso Plattner Institute - University of Potsdam
Potsdam, Germany
daniel.limberger@hpi.de

Christoph Lippert
Hasso Plattner Institute - University of Potsdam
Potsdam, Germany
christoph.lippert@hpi.de

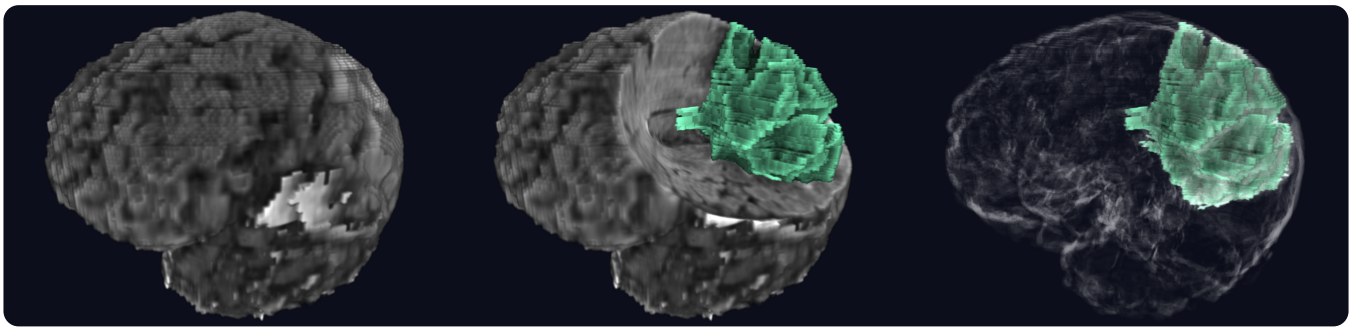

**Figure 1: Brain MRI rendered with 3 different transfer functions. Left: MRI intensities. Middle: Cutting a cone into the MRI to reveal an occluded tumor segmentation. Right: Prominent edges in the MRI for context around the tumor segmentation.**

## ABSTRACT

Volumetric segmentation of medical images is an essential tool in treatment planning and many longitudinal studies. While machine learning approaches promise to fully automate it, they most often still depend on manually labeled training data. We thus present a GPU-based volumetric region growing approach for semi-automatic brain tumor segmentation that can be interactively tuned. Additionally, we propose multidimensional transfer functions for ray tracing that allow users to judge the quality of the grown region. Our implementation produces a full brain tumor segmentation within a few milliseconds on consumer hardware. The visualization uses adaptive resolution scaling and progressive, asynchronous shading computation to maintain a stable 60 Hz refresh rate.

## CCS CONCEPTS

• **Human-centered computing** → **Visualization**; *Human computer interaction (HCI)*; • **Applied computing** → *Life and medical sciences*; • **Computing methodologies** → Computer graphics.


*Web3D '21, November 8–12, 2021, Pisa, Italy*

ACM ISBN 978-1-4503-9095-8/21/11…$15.00
https://doi.org/10.1145/3485444.3487640

## KEYWORDS

medical imaging, MRI, brain tumor, interactive segmentation, data labeling, region growing, progressive rendering, WebGL

**ACM Reference Format:**
Jonas Kordt, Paul Brachmann, Daniel Limberger, and Christoph Lippert. 2021. Interactive Volumetric Region Growing for Brain Tumor Segmentation on MRI using WebGL. In *The 26th International Conference on 3D Web Technology (Web3D '21), November 8–12, 2021, Pisa, Italy.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3485444.3487640

## 1 INTRODUCTION

Medical imaging is an essential diagnostic tool for clinicians and is widely used by researchers to gain statistical insights about a group of individuals. For these purposes, it is common to segment recorded scans, i.e., to assign a label to every pixel or voxel in them. A typical case of this is the segmentation of gliomas, a diverse set of tumors that present the most frequent brain malignancy, i.e., a medical condition that progressively causes an increasing amount of damage to the brain [DeAngelis 2001]. Segmenting gliomas in volumetric magnetic resonance imaging (MRI) scans is a task regularly involved in predicting their aggressiveness and the patient's response to therapy. It can also be a vital asset in conducting pre-operative risk assessments for tumor resection operations [Rosenstock 2018].

In practice, however, segmentation is a time-consuming task. As brain MRIs now frequently hold upward of 100 individual 2D slices, manually segmenting them becomes less feasible, while maintaining a spatial sense of segmented structures becomes impractical

when the segmentation tool is limited to slice-based views. Automated approaches to segmentation are typically based on supervised machine learning [Isensee et al. 2019; Zhang et al. 2021] and thus require vast amounts of manual labeling to generate sufficient training data [Shen et al. 2017]. To speed up this labeling process while maintaining a high data quality, semi-automatic segmentation methods can be used as long as they do not introduce significant bias. Thus, a workflow that captures the user's intent as directly as possible becomes necessary.

In this paper, we therefore

(1) present a GPU-based volumetric region growing workflow for semi-automatic segmentation implemented in WebGL,
(2) outline and discuss a web-based approach to direct volume rendering of medical images, and
(3) introduce progressive rendering strategies for iterative resolution and local ambient occlusion enhancements to balance interactive performance and rendered image quality.

These key contributions are integrated and combined into an interactive annotation system to (1) mitigate common issues associated with volumetric segmentation tasks and (2) speed up the generation and validation of glioma segmentations.

## 2 RELATED WORK

*Segmentation Techniques.* Different techniques exist for (semi-) automatic segmentation of medical images. The simplest approach for segmentation is global voxel-based *thresholding*. Here, every voxel of the image is classified as part of the segmentation or the background based on its intensity value and one or more thresholds. A slightly more complex algorithm is *region growing*. Starting from one or more initial "seeds", the region grows to its neighboring voxels if these fit some homogeneity criteria. The initial seeds can be set by the user [Adams and Bischof 1994] or automatically, based on heuristics [Lin et al. 2000]. Another concept is *watershed* segmentation [Vincent and Soille 1991]. The idea is to view grayscale images as a heightmap and simulate water settling in so-called "basins". *Level set* and *active contour* segmentation are based on modeling the edges of an segmentation. *Level set* segmentation uses signed distance fields where the 0-level is the border between segmentation and background. Similarly, *active contour* segmentation represents the edges of the segmentation using a large number of nodes. Several surveys focus on how these techniques can be implemented on the GPU [Eklund et al. 2013; Hadwiger et al. 2004; Smistad et al. 2015]. Region growing, in particular, can be implemented on the GPU using a GPGPU approach such as Nvidia CUDA [Kalaiselvi et al. 2017] or using shaders [Schenke et al. 2005].

Recently, *deep learning* has had a great impact on medical image analysis. Litjens et al. give a comprehensive overview, including segmentation approaches. For more focused applications to brain tumor segmentation, see Isensee et al.; Zhang et al..

*Region Growing Workflows.* The open-source segmentation application ITK-SNAP [Yushkevich et al. 2006] includes a 3D region growing tool but does not allow users to adapt how far the region is allowed to grow interactively. Instead, users have to watch the region growing on 2D slices and stop the process if they see the region escaping the desired area. Another approach for volumetric segmentation was shown by Sherbondy et al.. They show the resulting segmentation volume in 3D, but their 3D visualization does not provide context information around the segmentation. Another approach for dealing with region growing escaping the intended region is using a deformable clipping plane to cut off unintended parts of the segmentation [Chen and Rabadán 2017]. Combined with automatic seed selection and threshold calculation, region growing can also be used for fully automatic segmentation [Wantanajittikul et al. 2016].

*MRI.* Magnetic resonance imaging (MRI) is a non-invasive technique of medical imaging. Magnetic resonance scanners work by applying a strong magnetic field to the subject and measuring the anatomy of organs using radio waves. The natural and widespread visual representation of the resulting intensity values is a grayscale image with higher intensities being represented in a brighter shade. Different machine settings and contrast agents have an impact on the measured intensities and lead to different imaging modalities. While T1-weighted MRIs record a high intensity in fatty tissues such as subcutaneous fat, bone marrow, or blood products, T2-weighted MRI shows high intensity in fatty and water-based tissues, also including cerebrospinal fluid (CSF) and increased water in tumors. T2 Fluid Attenuated Inversion Recovery (FLAIR) shares many of the imaging characteristics of T2, but normal CSF fluid is attenuated so that it is easier to differentiate between abnormalities and CSF [Bakas et al. 2017, 2019; De Coene et al. 1992; Hajnal et al. 1992; Menze et al. 2015].

We use FLAIR scans with the mentioned grayscale mapping for all our images to make tumor locations more obvious to the reader, i.e., the high intensities of the increased water in the tumors show up as a bright shade of gray in our images.

## 3 VOLUMETRIC SEGMENTATION

Segmenting a volumetric structure such as brain tumors slice by slice takes a long time. Computer-assisted annotation tools that perform region growing based on a user-selected seed can accelerate this workflow without additional data requirements [Chen and Rabadán 2017; Top et al. 2011]. However, without any knowledge about the structure the user intends to segment, region growing algorithms tend to easily over-grow the expected area by a large amount. This is why region growing tools are often limited to segment one slice at a time, enabling users to judge the correctness of the grown region accurately.

To enable confident atomic segmentation of a whole 3D volume, we thus use a combination of the standard two-dimensional slice-based rendering and three-dimensional direct volume rendering. This helps users to recognize large over-grown areas rapidly.

Additionally, we do not force the user to discard the whole grown region, should any error occur. Instead, we provide them with a preview that they can interactively tune to scale back how far the region is allowed to grow from the seed. This way, it is possible to limit the grown region in a way that it cannot "escape" the intended bounds while still keeping as much as possible of the desired segmentation. Whether or not the grown region escapes the user-intended bounds can be judged in the context of a visualization of prominent edges in the MRI scan (figure 1, right).

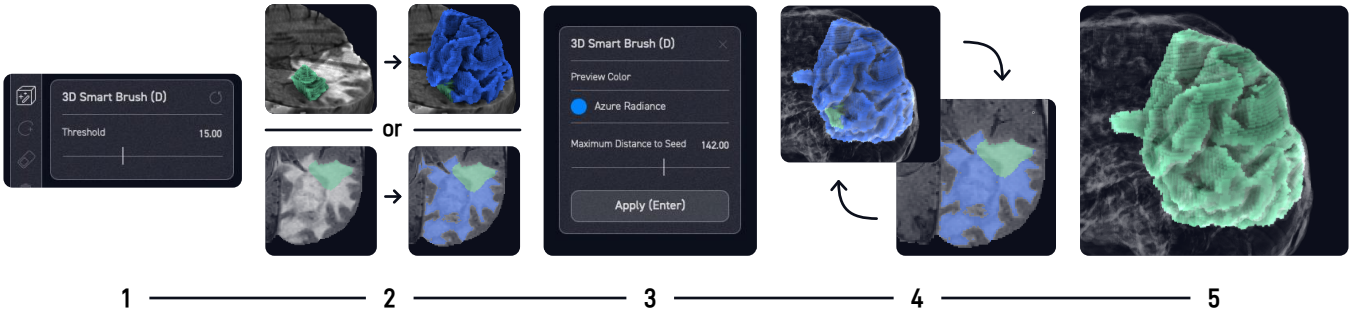


**Figure 2: Workflow for using 3D region growing to segment a brain tumor in an MRI. 1: Select desired threshold. 2: Select seed in 2D or 3D. System performs region growing. Region growing preview is rendered in both 2D and 3D. 3: Panel with tuning settings is shown. 4: Use slider from panel to adjust region growing distance. View in 2D and 3D to find errors. 5: Confirm the segmentation.**

*Mathematical Model.* As mentioned in section 1, we use a very simple mathematical model for region growing in order to avoid additional bias when using the result for training machine learning models. Let $I_{\text{seed}}$ be the MRI intensity of the selected seed voxel, $I_{\text{self}}$ the MRI intensity of a voxel that could become part of the region, $I_{\text{neighbor}}$ the MRI intensity of a neighboring voxel, and $t$ the region growing threshold. Then, the voxel becomes part of the region if the neighboring voxel is part of the region, and $|I_{\text{self}} - I_{\text{neighbor}}| < t$, and $|I_{\text{self}} - I_{\text{seed}}| < 1.5t$. This has to be tested for all six neighboring voxels. Using $I_{\text{seed}}$ ensures that the region cannot grow infinitely far away from the seed intensity in case there are smooth gradients in the MRI.

*Workflow.* The specific workflow of our 3D region growing tool is shown in figure 2. In the first step, the user sets the threshold used for region growing. This threshold controls the maximum difference between neighboring voxels' intensities that allows the region to grow. In the second step, the user selects the seed region. This can be done in both the 2D and the 3D views. In the 2D view, the user can scroll through the slices of the MRI and select the seed voxel using the mouse cursor. In the 3D view, the user can click anywhere in the rendered image and the seed will be set at the first high-opacity intersection of their click with the volume. This works especially well when using the cone transfer function (see figure 2, step 2) as it allows the user to set the seed somewhere in the middle of the tumor and not only on the outside of the brain. After that, the region growing is executed. The grown region is shown as a preview in a distinct color to differentiate it from any pre-existing segmentation parts. Furthermore, a tuning UI panel is automatically opened, giving the user access to a slider that controls the maximum distance the region may grow from the seed. Figure 3 shows the effect of manipulating the slider from within the 3D view. In the fourth step, the user can switch between the 2D and 3D view while manipulating the slider to ensure there are no large errors where the region over-grew the desired part of the volume. As guidance, different transfer functions (detailed in section 4) can be used, e.g., one that shows prominent edges in the underlying MRI, which can be useful to judge whether the grown region has escaped the intended section of the volume or not. Finally, in the fifth step, the user applies the previewed region to merge it into the segmentation volume.

*Integration.* Our 3D region growing workflow is integrated into a 2D segmentation web application called *VISIAN*[1]. This means that smaller clean-up and voxel-level corrections can be performed in 2D after the 3D region growing has been completed. For this, *VISIAN* includes multiple tools such as a simple pixel brush and eraser, 2D region growing, and outlining to add and remove voxels to and from the segmentation. Additionally, interactive dilation and erosion filters can be used to fill small holes or remove unwanted noise in the segmentation. Due to this integration, a hybrid 2D and 3D workflow for segmentation becomes feasible.

## 4 DIRECT VOLUME RENDERING

In this section, we briefly describe the *transfer functions* used in our volume rendering, discuss aspects on *medical correctness* of the rendering itself, and highlight our *progressive rendering* approach.

*Transfer Functions.* To display MRIs and segmentation volumes in 3D, we provide multidimensional transfer functions, namely $H_{\text{dense}}$, $H_{\text{clip}}$, $H_{\text{edge}}$, and $H_{\text{user}}$, used for ray tracing (more specifically, ray marching). Transfer function $H_{\text{dense}}$ maps the MRI intensities to opacity and grayscale of the volume (figure 1, left). The displayed intensity values can be filtered from top and bottom, and the segmentation volumes can be used to mask them. Transfer function $H_{\text{clip}}$ uses selective spatial clipping to enable *focus+context* visualization. The focus is put on the segmentation volumes while the MRI scans are used for context and spatially clipped by a cone (figure 1, middle). The cone can follow the virtual camera or be fixed in place. A similar approach for selective spatial clipping, presented by Díaz et al., used a clipping plane instead of a cone (which can be seen as a special case of a 180° cone). $H_{\text{edge}}$ semi-transparently displays pronounced edges in the MRI scans around the segmentations to give spatial context information while barely occluding the focus volumes (figure 1, right). The edges are detected based on gradient magnitude [Kniss et al. 2002] and can be filtered from top and bottom. To allow for custom mapping of MRI intensity values

[1] *VISIAN* is a web-based image segmentation suite with a focus on machine learning in the medical domain. Please refer to https://visian.org for more information.

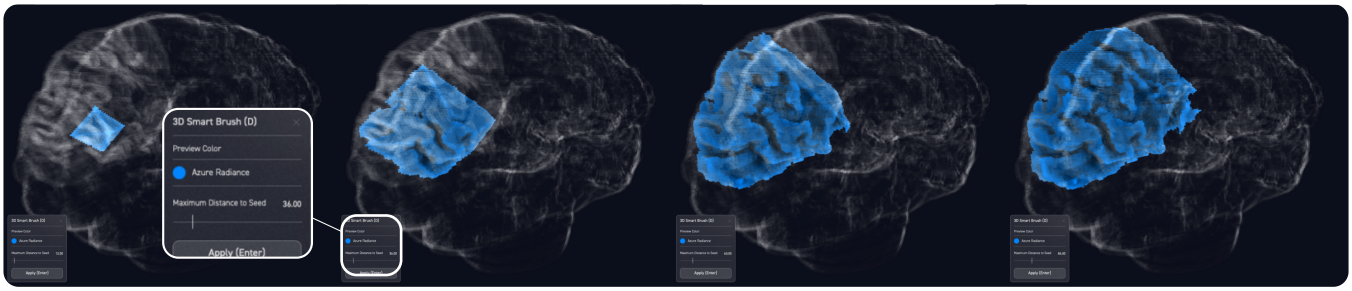


**Figure 3: 3D region growing of a brain tumor segmentation restricted to different numbers of region growing steps; 12, 36, 60, 84 f.l.t.r. respectively. Note that the segmentation is contained in the visible tumor edges.**

to colors and opacities, a fourth transfer function, $H_{\text{user}}$, can be specified by custom, single-pixel-height images uploaded by the user.

*Medical Correctness.* While color, opacity, and shading within our 3D visualization collectively provide an expressive visual representation of the data, they also influence the perception of the data and might cause misrepresentation of the real anatomy [Pommert and Höhne 2002]. Therefore, we give users additional access to the raw MRI data by means of a 3D clipping plane that can be placed and oriented freely within the MRI volume (figure 4). If the plane is axis-aligned, the image displayed on the plane shows a typical MRI slice with square (or rectangular for non-isotropic scans) voxel shapes (figure 4, left). Regardless of the plane orientation, however, nearest neighbor filtering is favored over linear interpolation to display only accurate and factual intensity values (figure 4, right).

*Progressive Rendering.* To achieve interactive frame rates even on low-performance devices, we use progressive rendering for resolution and shading quality. In particular, when the user moves the virtual camera, the resolution is decreased immediately and progressively increased once the movement has stopped. This allows for both a smooth frame rate during interaction and full-quality images once the desired perspective is found. For shading, we use progressive local ambient occlusion (LAO), which determines the occlusion (darkening) of a particular voxel by sampling short rays in multiple directions [Hernell et al. 2010] over multiple frames. For every frame, only a few rays are used, and the results are accumulated over time. Thereby, the shading quality continuously increases. This shading remains valid regardless of camera movement. It is invalidated only when the volume display configuration changes, e.g., through clipping or transfer function changes.

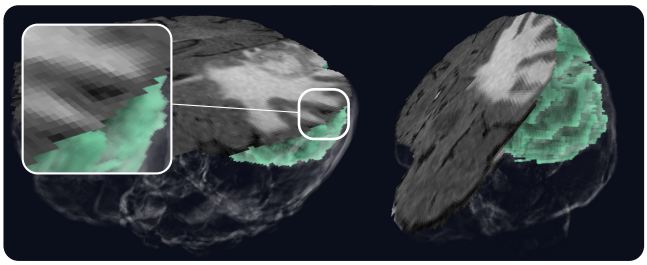

**Figure 4: Raw MRI data is shown on a clipping plane through the volume. Axis-aligned (left) and freely oriented (right).**

## 5 IMPLEMENTATION

*VISIAN* is built in TypeScript using React with MobX for state management and runs exclusively on the client. For graphics, we use Three.js as a wrapper around WebGL. Since we built *VISIAN* with tablet and especially iPad users in mind to enable pen input for segmentation, we decided to use WebGL 1. This means that we cannot use 3D textures. Instead, we use texture atlases that contain all slices of a scan next to each other. This has positive and negative effects. It makes it easy, e.g., to generate textures that contain gradient information used for detecting edges. We can simply render the whole atlas at once with a corresponding fragment shader and do not have to render slice by slice. However, it also means that we cannot use the hardware's trilinear interpolation. Instead, we have to use the hardware's bilinear interpolation and perform a third mix manually in a shader. With the expected upcoming ratification of WebGPU, a switch from WebGL to WebGPU seems very promising since it might resolve the current workarounds and limitations while improving the overall performance of our application.

In the following, implementation details on *ray tracing*, *GPU-based 3D region growing*, and *progressive rendering* are provided.

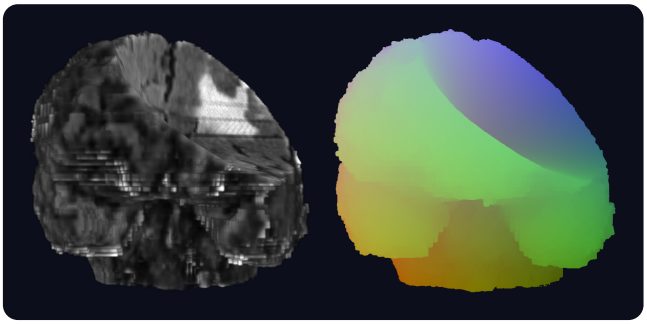

**Figure 5: Brain MRI with a tumor rendered using the $H_{\text{clip}}$ transfer function. Normal volume shader showing the tumor as a potential region growing seed location in white (left) and volumetric picking shader (right).**

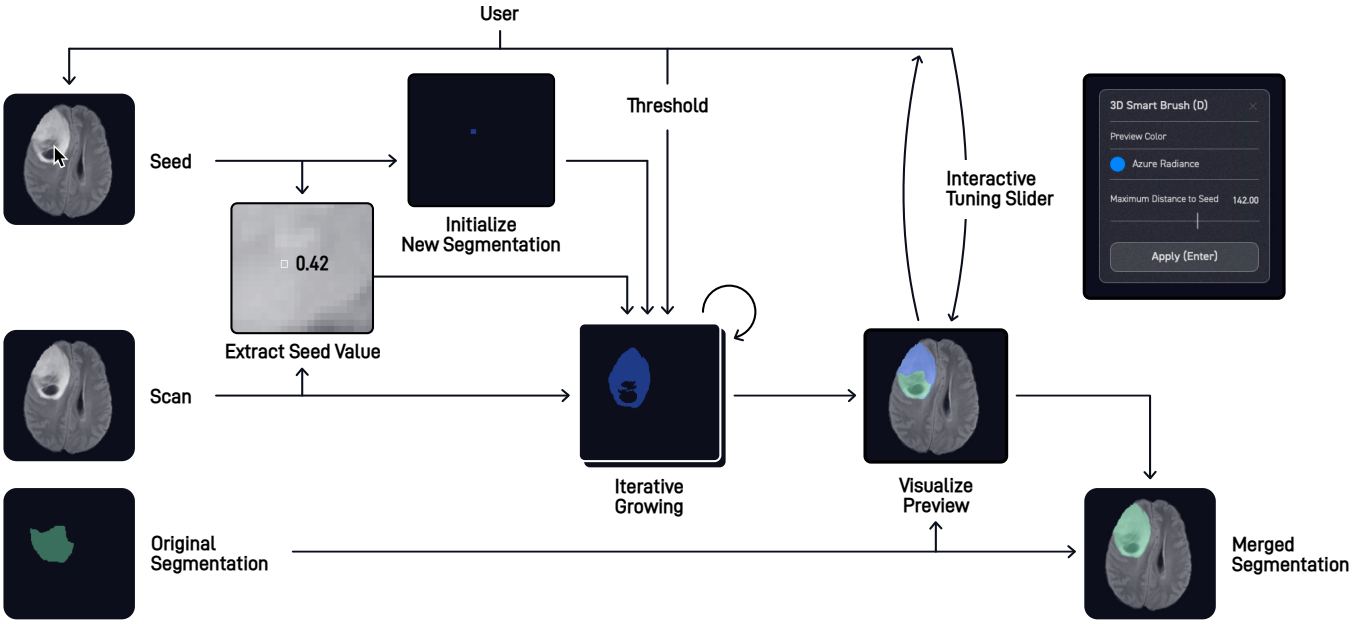


**Figure 6: Data flow of the region growing process. The process is initiated by the user when selecting a seed position. From this, an initial segmentation containing only the selected voxel is created, and the seed value is extracted from the underlying scan. In the iterative growing phase, this initial segmentation then grows iteratively based on the extracted seed value, the underlying scan, and the user-defined threshold. The grown segmentation is then visualized as a preview together with the original segmentation and the user can interactively tune how far the region may grow from the seed using the tuning slider. Finally, the tuned segmentation is added to the original one to form the merged segmentation.**

*Ray Tracing.* For the initial rays, the back faces of the volume's bounding box are rasterized and used for subsequent, custom ray marching through the MRI volume (utilizing fragment shaders for computation). This simplifies further composition of other auxiliary meshes like clipping plane (figure 4) or VR controllers rendered directly using Three.js. To render multiple MRI scans and segmentations simultaneously, we use a stack of layers. During ray marching, we blend the colors and opacities of all layers at every sample point along the rays according to the stack's layer sequence. This allows the user to rearrange layers such that more important segmentations show up more dominantly when overlapping with other segmentations. Furthermore, this unifies the 2D and 3D experience as the layers are blended in the same way for the 2D view.

To push all layers onto the GPU, we use a columnar uniform layout in the fragment shader. However, since WebGL 1 only allows constant array lengths for uniforms, we apply custom regex macros to compose the desired shader code and adjust it to the required number of layers.

*3D Seed Selection.* To place a seed for region growing in the 3D view the user clicks onto the rendered image. To convert the 2D click position into the corresponding 3D position in the volume, volumetric GPU picking is used, rendering the volume again off screen with a modified shader.

This modified shader does the same front-to-back ray marching and color accumulation as the normal volume shader. However, when a small opacity threshold (in our case 0.01) is reached, the ray marching loop is terminated and the current normalized volume coordinates from the ray marching process are rendered into the r, g and b channels of the output buffer. To indicate that the ray marching hit a target, 1.0 is rendered into the alpha channel. See figure 5 for a side-by-side comparison of a brain MRI rendered with the $H_{\text{clip}}$ transfer function and its corresponding volumetric GPU picking shader. If no high opacity is reached, i.e., the user clicked onto the background, this is indicated by a 0.0 in the alpha channel. The same kind of shader is used for the clipping plane (figure 4) during the picking process. However, as this is a normal, rasterized geometry, no ray marching is needed here. The rendered normalized volume coordinates are then converted to the corresponding voxel information, which is then used as the seed for region growing.

*GPU-based 3D Region Growing.* The data flow involved in the tunable 3D region growing process is depicted in figure 6.

The region growing itself iteratively runs on the GPU. We use two frame buffers and render back and forth between them by using them as alternating input and output. In each step, the region can grow one step. As we are using a texture atlas instead of a 3D texture, we render the whole atlas at once. For this, we use a screen-aligned quad and a fragment shader to compute the region growing for each voxel (listing 1). The same procedure is also used to enable multi-step dilation/erosion for segmentation clean up.

To allow for interactive adjustment of the maximum region growing distance (figure 3), we encode the step in which a voxel first became part of the region using 8 bit in one of the texture's channels. Based on this value, we filter the preview using the user-controlled slider value. To render the preview, an additional annotation layer

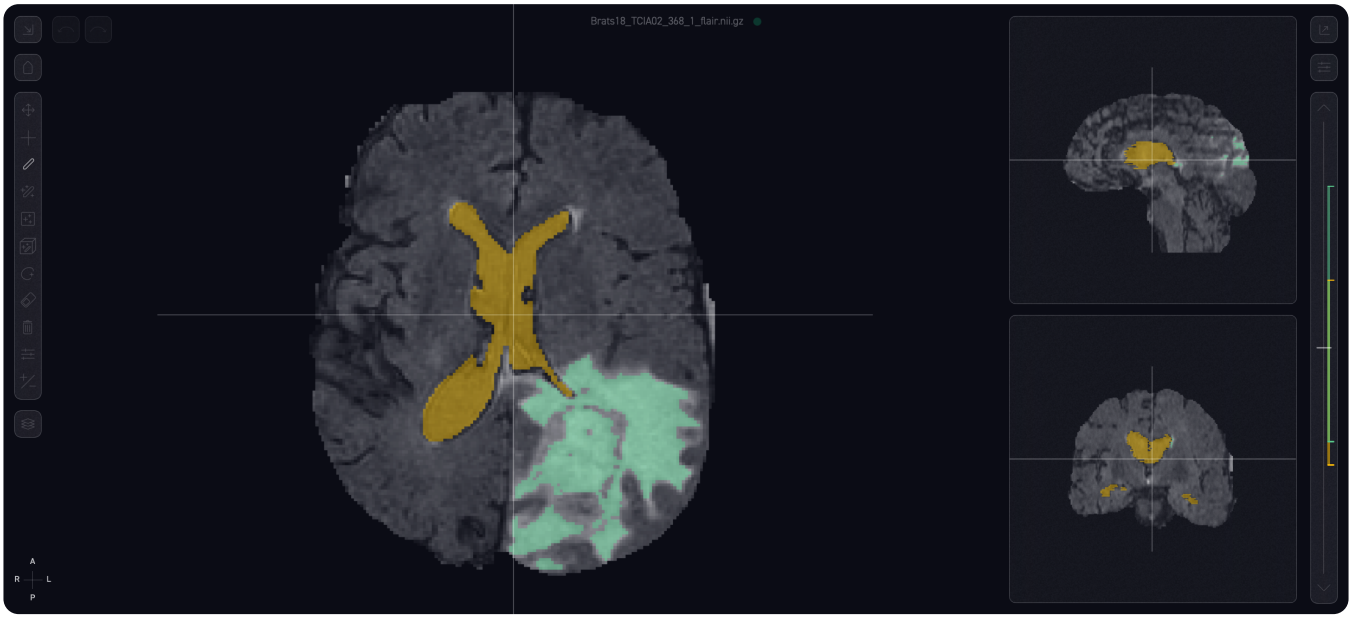


**Figure 7: *VISIAN's* 2D interface showing a brain MRI with segmentations of a tumor (green) and the ventricles (yellow).**

is placed on top of the layer stack. When applying the previewed segmentation to the existing one, we use the filter and render the voxels additively into the segmentation's texture atlas. Encoding the step number in 8 bits allows us to grow the region by 254 steps (0 is reserved for voxels not part of the region, and 255 is the seed). More channels could be used to encode more steps. However, this would also linearly increase the time it takes to perform the region growing, as would increasing the size of the scan. For common scan resolutions used in brain tumor segmentation, 254 steps are typically sufficient to segment whole tumors.

```
// inputs excluding texture atlas information
vec2 uv, sampler2D uIntensity, sampler2D uRegion,
float uEncodedStep, float uSeedInt, float uThreshold

const float two_over_three = 2.0 / 3.0;
bool grow(float ownInt, float nInt, float nReg) {
    return all(lessThan(vec3(-nReg, abs(ownInt - uSeedInt)
        * two_over_three - uThreshold,
        abs(ownInt - nInt) - uThreshold), vec3(0.0)));
}

void main() {
    vec4 region = texture2D(uRegion, uv);
    if(region[0] > 0.0) { gl_FragColor = region; return; }
    // ... texture atlas magic to get uv0 to uv5
    float int0 = texture2D(uIntensity, uv0)[0];
    float reg0 = texture2D(uRegion, uv0)[0];
    // ... the same for uv1 to uv5
    float ownInt = texture2D(uIntensity, uv)[0];
    if(!(grow(ownInt, int0, reg0) || ... )) discard;
    gl_FragColor = vec4(vec3(uEncodedStep), 1.0);
}
```

**Listing 1: GLSL shader that decides for each voxel whether the region should grow onto it from one of its six neighbors.**

The region growing itself operates on the raw MRI intensities. It uses the selected seed's and the six neighboring voxels' intensities for thresholding when growing the region to adjacent voxels (see section 3 for the mathematical model and listing 1 for an implementation in a GLSL fragment shader). Whenever the user changes the region growing distance, voxels are added or removed from the preview (figure 3) without recomputation. However, this means that the LAO for all voxels needs to be recomputed. Since this is a computationally intensive operation, we momentarily turn off shading updates while the user changes the distance.

*Progressive Rendering.* For progressive rendering, tiled rendering is used. In particular, rendering a higher resolution of the image is not done all at once. Instead, smaller tiles are rendered and stitched together at the desired resolution. This allows aborting rendering of higher resolutions quickly, e.g., when the user moves the virtual camera again. The LAO texture atlas is rendered tile-based as well. Here, the results of the previous rendering step are reused and accumulated. This way, we can start with only a few LAO rays, e.g., eight, and progressively produce a full resolution image with this first shading iteration. Then, another batch of LAO rays is rendered and accumulated. With the resulting LAO texture atlas, a full resolution image is rendered off-screen and swapped with the "low-cost" screen image once it is ready. This process is repeated until a target number of LAO rays is reached (32 by default).

## 6 RESULTS

Our approach achieves an end-to-end 3D segmentation workflow that extends the 2D user interface of the image annotation system *VISIAN* as seen in figure 7.

*Performance.* We tested our implementation on a local development deployment of *VISIAN* running in Chrome 92.0.4515.107

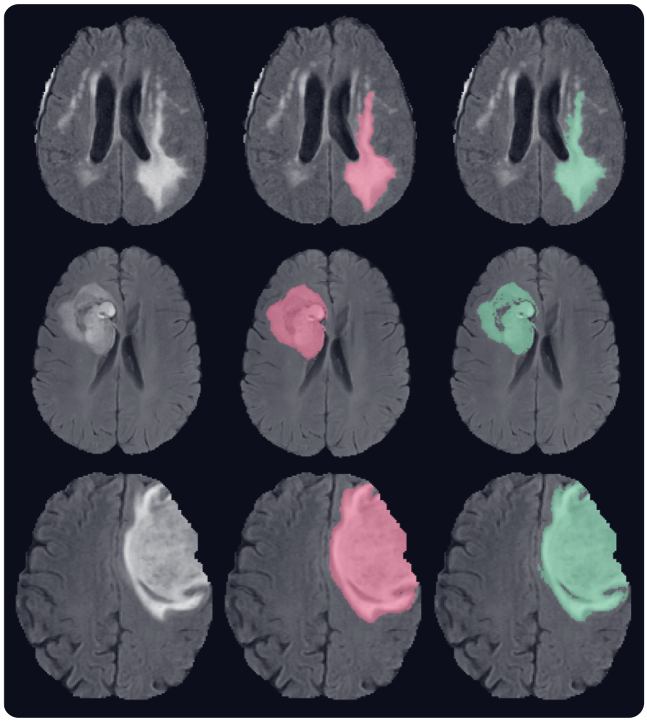

**Figure 8: Comparison of expert ground truth annotations (red, middle) and amateur annotations using the volumetric region growing tool (green, right) on the raw scans (left). Note the small holes in the region growing annotations (green, right) in the 2nd and 3rd rows. These occur because parts of the tumor core appear darker in FLAIR MRI which stops the region growing algorithm. Using the dilate/erode post-processing can fill such holes.**

(64-bit) on a workstation equipped with an i7-7700K running at 4.50 GHz, 32 GB of 3000 MHz DDR4 RAM, an Nvidia GeForce RTX 2070 SUPER, and a 4K display.

Table 1 shows the computation times for all 254 region growing steps on a scan from the BraTS 2018 dataset [Bakas et al. 2017, 2019; Menze et al. 2015] of resolution $240\times240\times155$, resulting in a texture atlas resolution of $3120 \times 2880$. As evident, the first run completed in just below $40ms$ with precompiled shaders. In our tests, it took $10 - 20ms$ longer if the required shaders had not been precompiled. Due to caching effects, the following consecutive region growing runs on the same scan completed in, on average, $5ms$.

All proposed transfer functions produced stable 60 fps video output using our progressive rendering on the above hardware. Without our progressive resolution, the frame rate, depending on the zoom level, occasionally dropped to as low as 20 fps.

*Segmentation Quality.* A first qualitative evaluation of our region growing approach on FLAIR-modality BraTS scans showed that it can produce glioma segmentations that require very little tweaking to achieve accurate outer boundaries. Figure 8 shows a comparison of three expert-annotated ground truth segmentations (red, middle) and amateur segmentations using our volumetric region growing

| **Run Number** | #1 | #2 | #3 | #4 | Mean of #5–#50 |
|---|---|---|---|---|---|
| **Computation Time in** ms | 37.3 | 26.8 | 11.2 | 6.6 | $4.93 \pm 0.74$ |

**Table 1: Wall clock times for consecutive, user-initiated runs of all 254 region growing steps on a $240 \times 240 \times 155$ scan.**

workflow (green, right) on the raw scans (left). All three scans and ground truth segmentations come from the BraTS 2018 dataset. The ground truth segmentations were created by multiple experts using multiple scan modalities (see [Bakas et al. 2017] for more details), while the region growing segmentations were created by an amateur using only the volumetric region growing workflow described in figure 2 and only the shown FLAIR modality.

However, as the core of gliomas can yield fundamentally different image intensities from its surroundings, the generated segmentations often contained a hole in their center (see figure 8, rows 2 and 3). Placing a second seed or using the dilation/erosion post-processing described in section 3 could typically fill these holes.

Especially small holes in the generated segmentation that can often be fully occluded proved hard to find in the volumetric 3D view. We deduce that our current transfer functions can speed up the segmentation proofing process, but they cannot yet replace the 2D view for the task of voxel-exact proofing. However, they were particularly helpful when tuning the maximum number of region growing steps as changes to it became immediately visible on the outside of the generated segmentation.

## 7 CONCLUSION AND FUTURE WORK

Volumetric region growing can support quick segmentation of 3D structures in medical images such as brain tumors. By allowing the user to interactively scale how far a region can grow, unintended over-growing can be mitigated. Another key to a successful 3D segmentation workflow is the tight integration of volumetric rendering with *focus+context* visualizations into the segmentation process. This way, users can judge the volumes they are annotating in 3D and spot errors more quickly and easily. Additionally, the 2D tools of a segmentation application such as *VISIAN* can be used for effective clean up of grown regions.

Our progressive rendering approach is essential to (1) maintain interactive frame rates on low-end devices while generating images of consistently high quality, (2) enable a responsive segmentation workflow, and (3) provide high resolution and high-quality shading to increase comprehension of volumetric structures.

In the future, *VISIAN's* VR support could be used to set a seed region in 3D using controller input. Additionally, the presented 3D workflow could be extended by a "3D sculpting"-like brush tool that could add and remove voxels from the surface of a segmentation in 3D. While we do not find it likely that such a 3D correction workflow can fully replace voxel-level correction and proofing using the 2D view, we do believe it can save time when larger parts of a segmentation that span multiple slices have to be removed. Finally, a comprehensive evaluation of our tool in real-world MRI segmentation tasks performed by clinicians is planned and important to steer subsequent research.

## ACKNOWLEDGMENTS

We thank Maximilian Lindner for assistance with the graphics and *VISIAN*'s user interface as well as Clara Uktar for help with *VISIAN*'s development.

## REFERENCES

R. Adams and L. Bischof. 1994. Seeded Region Growing. 16, 6 (1994). https://doi.org/10.1109/34.295913

Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, et al. 2017. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific Data* 4 (09 2017). https://doi.org/10.1038/sdata.2017.117

Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, et al. 2019. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. arXiv:1811.02629 [cs.CV]

Andrew X. Chen and Raúl Rabadán. 2017. A Fast Semi-Automatic Segmentation Tool for Processing Brain Tumor Images. In *Towards Integrative Machine Learning and Knowledge Extraction*. Springer International Publishing, 170–181. https://doi.org/10.1007/978-3-319-69775-8_10

B. De Coene, J. V. Hajnal, P. Gatehouse, D. B. Longmore, S. J. White, et al. 1992. MR of the brain using fluid-attenuated inversion recovery (FLAIR) pulse sequences. *AJNR Am J Neuroradiol* 13, 6 (1992), 1555–1564.

Lisa M DeAngelis. 2001. Brain tumors. *New England journal of medicine* 344, 2 (2001), 114–123. https://doi.org/10.1056/NEJM200101113440207

Jose Díaz, Eva Monclús, Isabel Navazo, and Pere-Pau Vázquez. 2012. Adaptive Cross-sections of Anatomical Models. *Computer Graphics Forum* 31, 7 (2012), 2155–2164. https://doi.org/10.1111/j.1467-8659.2012.03208.x

Anders Eklund, Paul Dufort, Daniel Forsberg, and Stephen M. LaConte. 2013. Medical image processing on the GPU – Past, present and future. *Medical Image Analysis* 17, 8 (2013), 1073–1094. https://doi.org/10.1016/j.media.2013.05.008

Markus Hadwiger, Caroline Langer, Henning Scharsach, and Katja Bühler. 2004. State of the art report 2004 on GPU-based segmentation. *VRVis Research Center* 3 (2004).

J. V. Hajnal, B. De Coene, P. D. Lewis, C. J. Baudouin, F. M. Cowan, et al. 1992. High signal regions in normal white matter shown by heavily T2-weighted CSF nulled IR sequences. *J Comput Assist Tomogr* 16, 4 (1992), 506–513.

Frida Hernell, Patric Ljung, and Anders Ynnerman. 2010. Local Ambient Occlusion in Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics* 16, 4 (2010), 548–559. https://doi.org/10.1109/TVCG.2009.45

Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H. Maier-Hein. 2019. No New-Net. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Springer International Publishing, 234–244. https://doi.org/10.1007/978-3-030-11726-9_21

T. Kalaiselvi, P. Sriramakrishnan, and K. Somasundaram. 2017. Survey of using GPU CUDA programming model in medical image analysis. *Informatics in Medicine Unlocked* 9 (2017), 133–144. https://doi.org/10.1016/j.imu.2017.08.001

J. Kniss, G. Kindlmann, and C. Hansen. 2002. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (2002), 270–285. https://doi.org/10.1109/TVCG.2002.1021579

Zheng Lin, Jesse Jin, and Hugues Talbot. 2000. Unseeded Region Growing for 3D Image Segmentation.. In *ACM International Conference Proceeding Series*, Vol. 9. 31–37.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, et al. 2017. A survey on deep learning in medical image analysis. *Medical Image Analysis* 42 (Dec 2017), 60–88. https://doi.org/10.1016/j.media.2017.07.005

Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, et al. 2015. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging* 34, 10 (2015), 1993–2024. https://doi.org/10.1109/TMI.2014.2377694

Andreas Pommert and Karl Heinz Höhne. 2002. Evaluation of Image Quality in Medical Volume Visualization: The State of the Art. In *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2002*. Springer Berlin Heidelberg, 598–605. https://doi.org/10.1007/3-540-45787-9_75

Tizian Rosenstock. 2018. *Risk stratification in motor area-related glioma surgery based on navigated transcranial magnetic stimulation data*. Ph.D. Dissertation. Charité - Universitätsmedizin Berlin. https://doi.org/10.17169/REFUBIUM-6070

Stefan Schenke, Burkhard C Wuensche, and Joachim Denzler. 2005. GPU-based volume segmentation. In *Proceedings of IVCNZ*, Vol. 5. Citeseer, 171–176.

Dinggang Shen, Guorong Wu, and Heung-Il Suk. 2017. Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering* 19, 1 (2017), 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442

A. Sherbondy, M. Houston, and S. Napel. 2003. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *IEEE Visualization, 2003.* 171–176. https://doi.org/10.1109/VISUAL.2003.1250369

Erik Smistad, Thomas L. Falch, Mohammadmehdi Bozorgi, Anne C. Elster, and Frank Lindseth. 2015. Medical image segmentation on GPUs – A comprehensive review. *Medical Image Analysis* 20, 1 (2015), 1–18. https://doi.org/10.1016/j.media.2014.10.012

Andrew Top, Ghassan Hamarneh, and Rafeef Abugharbieh. 2011. Active Learning for Interactive 3D Image Segmentation. In *Lecture Notes in Computer Science*. Springer, 603–610. https://doi.org/10.1007/978-3-642-23626-6_74

L. Vincent and P. Soille. 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 6 (1991), 583–598. https://doi.org/10.1109/34.87344

Kittichai Wantanajittikul, Nipon Theera-Umpon, Suwit Saekho, Sansanee Auephanwiriyakul, Arintaya Phrommintikul, et al. 2016. Automatic cardiac T2* relaxation time estimation from magnetic resonance images using region growing method with automatically initialized seed points. *Comput Methods Programs Biomed* 130 (2016), 76–86. https://doi.org/10.1016/j.cmpb.2016.03.015

Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, et al. 2006. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage* 31, 3 (2006), 1116–1128. https://doi.org/10.1016/j.neuroimage.2006.01.015

Dingwen Zhang, Guohai Huang, Qiang Zhang, Jungong Han, Junwei Han, et al. 2021. Cross-modality deep feature learning for brain tumor segmentation. *Pattern Recognition* 110 (Feb. 2021), 107562. https://doi.org/10.1016/j.patcog.2020.107562