GrowCut under StudierFenster

Alessandra Masur*
Supervised by: Jan Egger†

Institute of Computer Graphics and Vision (ICG)
Graz University of Technology
Graz / Austria
Institute for AI in Medicine (IKIM)
University Hospital Essen
Essen / Germany

Abstract

Segmentation is a crucial procedure in medical image analysis. The usage of automatic algorithms in this field is an attractive alternative to manual segmentation. One promising semi-automatic segmentation tool is the GrowCut algorithm, which allows n-dimensional image segmentation, providing interactive and dynamic features. Currently, using the GrowCut algorithm for medical image segmentation with a user interface is only possible via medical image analysis software, making it device- and platform-dependent. The GrowCut algorithm without a user interface is available via various implementations but requires a lot of technical knowledge of the user.

The aim of this contribution is to provide a user interface for the GrowCut algorithm on the basis of a web application. This is achieved by implementing an adapted version of the GrowCut algorithm, the Fast GrowCut algorithm, into a client/server based, web-hosted 3-dimensional medical image viewer, called StudierFenster. As a result, the Fast GrowCut algorithm can be used directly inside the online environment without installing software and without technical knowledge of the user. It is now possible to use the segmentation tool on any 2-dimensional transverse slice of a 3-dimensional image. The workflow was made user-friendly, allowing input to be drawn with a brush onto the image and loading the output automatically, making it immediately visible.

Keywords: GrowCut, Fast GrowCut, Segmentation, Medical Image Analysis, Web Application, StudierFenster

1 Introduction

Image segmentation is one of the most important and common practices for medical images analysis [1, 2, 3]. Extracting the region of interest can be done in an automatic or semi-automatic process. The GrowCut algorithm is a widely used semi-automatic segmentation algorithm, working on n-dimensional images, and uses cellular automata to calculate the segmentation. As input, seeds are selected manually inside and outside the region to be segmented, marking foreground and background, respectively. During execution of the GrowCut algorithm, all remaining image pixels get sorted into either foreground or background [4]. It is an attractive segmentation tool because of its interactive and dynamic features. The GrowCut algorithm has been further developed into the Fast GrowCut algorithm, by reformulating it as a clustering problem and approximating the solution, making it significantly faster [5]. Different versions of the GrowCut algorithm are available on medical image computing platforms, such as the 3D Slicer software [6], however, not on a web-based tool where no installation of software is needed. Aside from that, a lot of open source implementations of the GrowCut algorithm are available, which do not provide a user interface and therefore require a lot of technical knowledge of the user.

This contribution aims to provide a user interface for segmentation with the GrowCut algorithm on a device- and platform-independent basis. To achieve this, the Fast GrowCut algorithm is implemented in the Medical 3D Viewer of the "StudierFenster" website (`http://studierfenster.at/`, `http://studierfenster.icg.tugraz.at/`), hereafter referred to as StudierFenster. It provides visualization and segmentation tools for medical images and is built as a client-server model. Its main component is the Medical 3D Viewer, which offers various annotation and segmentation tools [7].

As a result of this contribution, the Fast GrowCut algorithm was successfully implemented into the StudierFenster website as a segmentation feature within the Medical 3D Viewer. It is now possible to use the Fast GrowCut algorithm with a user interface without additional software. Its implementation was achieved by creating the input and viewing the output on the client side and running a Python script with the algorithm on the server side. In the Medical 3D Viewer, the Fast GrowCut algorithm can be used to

*alessandra.masur@student.tugraz.at
†egger@icg.tugraz.at

segment 2-dimensional regions of slices in the horizontal plane.

Giving an overview of this paper, Section 2 gives background on the topic of StudierFenster. The related work is presented in Section 3. In Section 4, the methods of implementing the Fast GrowCut algorithm into StudierFenster are elaborated. Lastly, the results of this contribution are given in Section 5, followed by a concluding discussion and an outlook in Section 6.

2 Background

StudierFenster StudierFenster [1] is a website that was developed in previous work at the Graz University of Technology in association with the Medical University of Graz. It is an online environment providing visualization tools for medical data, manual segmentation tools for medical images, and tools for calculations. The tools can be used directly within a web browser which makes them available to more people and platform- and device-independent. Furthermore, adaptions or updates to the software can be deployed directly on the website, without having to distribute changes to each user as it would be the case with software. StudierFenster is built as a client–server model, where some calculations of segmentations are done by the server part. This makes it unique from other existing web-based tools that only use a client-oriented approach. Adding a server back-end provides more complex functionality. A user study conducted by Weber et al. in 2019 generated positive feedback and StudierFenster has been adapted and worked on since its release [8]. The current functionalities, as of November 2022, include a Digital Imaging and Communications in Medicine (DICOM) browser and converter, the Medical 3D Viewer with 2D and 3D data visualization and various manual annotation tools, automatic aortic landmark detection, aortic dissection inpainting [9], centerline tracking [10], 3D skull reconstruction [11], 3D face reconstruction and registration [12], medical virtual reality viewer, and finally the calculation of the Dice coefficient and Hausdorff distance [8]. The client side is written with Hypertext Markup Language (HTML) and JavaScript, also using the Web Graphics Library (WebGL). The server side is written in C, C++, and Python, using libraries like Insight Toolkit (ITK), Visualization Toolkit (VTK), X Toolkit (XTK), and Slice:Drop. Server requests are processed by a Python Flask server [7].

GrowCut Algorithm The GrowCut algorithm is a semi-automatic image segmentation algorithm. It works on n-dimensional images and the segmentation is an iterative process, in which the user has the possibility to give additional input after each iteration. It uses cellular automata which allows fast and parallel computation. Every pixel in the GrowCut algorithm has a corresponding three-tuple $(l_p, \theta_p, \overrightarrow{C}_p)$, where l_p is the label of a pixel, θ_p is the strength of a pixel and $\overrightarrow{C}_p$ stands for the color value [4]. For the initialization of the seed pixels, some pixels are initialized as foreground $l_p = 1$ and some as background $l_p = 0$. Additionally, all initialized seed pixels are assigned the strength value $\theta_p = 1$. After initialization, the algorithm begins with the iteration process of sorting all other pixels into either foreground or background [13]. Simplified, at each iteration step t, each cell tries to "attack" its neighbors with the intention of spreading the foreground or background labels. The force of the cells is dependent on the strength values θ_p and θ_q and the distance between the vectors $\overrightarrow{C}_p$ and $\overrightarrow{C}_q$ of the attacking and defending cells. If an attacker has the greater attack force, its labels are spread onto its weaker neighbor cells [4]. The computation is finished when every pixel in the region of interest is assigned one of two labels [13].

Fast GrowCut Algorithm The Fast GrowCut was developed by reformulating the GrowCut algorithm as a clustering problem, to which the Fast GrowCut computes a fast, approximate solution. The clustering problem which is based on finding the shortest path, can be solved by applying an adapted version of the Dijkstra algorithm. The adaptation is introduced because the Dijkstra algorithm is static, not allowing any user input after it was initiated. To keep the Fast GrowCut algorithm dynamic and allow editing, only local regions that are affected by a new input are updated with the adapted version of the Dijkstra algorithm [5].

3 Related Work

Different versions of the GrowCut algorithm are implemented in the 3D Slicer platform, where they can be used for 2D and 3D segmentation. 3D Slicer is an open-source medical image computing platform, offering various segmentation tools [13, 14].

GrowCut in 3D Slicer The GrowCut algorithm is implemented in 3D Slicer as a module called "GrowCutSegmentation". It is an editor effect that is based directly on the original GrowCut algorithm first introduced by Vezhnevets and Konouchine [4]. The user can color foreground and background, which are used to compute the segmentation. After the computation is finished, the user can further adapt the segmentation by adding additional edits to the segmentation [6].

One application example is the use of GrowCut in 3D Slicer in research about vertebral body segmentation. There it was found that, depending on the use case, the GrowCut algorithm in 3D Slicer can be a more efficient way to segment medical images than manual analysis. In a study conducted by Egger et al. from 2017 [13], results

[1] http://studierfenster.at/

were compared for segmentation of vertebral bodies in T2-weighted magnetic resonance images (MRI). The GrowCut in 3D Slicer was shown to only require significantly less segmentation time and effort than a manual assessment with similar accuracy [13].

Fast GrowCut in 3D Slicer The Fast GrowCut is implemented in 3D Slicer as a module called "FastGrowCutEffect". It is based on the Fast GrowCut algorithm that was introduced by Zhu et al. in 2014 [5]. The user can segment an image by selecting foreground and background seeds in 3D. After initial segmentation, there is the possibility to refine the segmentation repeatedly until the user is satisfied with the result. Furthermore, the module allows multi-label segmentation [15].

The latest version of 3D Slicer, as of October 2022, uses the Fast GrowCut Segmentation under the name "Grow from Seeds", which is an editor that can be found among other segmentation tools inside the "Segment Editor" module of 3D Slicer [16].

4 Methods

4.1 Workflow

The workflow of using the Fast GrowCut in StudierFenster can be summarised in three steps "Mask Creation", "Executing Fast GrowCut" and "Result Handling". The flowchart in Figure 1 displays the usage of the Fast GrowCut in StudierFenster and gives an overview how the three mentioned steps can be divided further. In Figure 1, blue fields represent user input, and green fields represent steps that are executed automatically.

The workflow starts with loading a nearly raw raster data (NRRD) file in the Medical 3D Viewer in StudierFenster, as it can be seen in Figure 1. The selected file is loaded in the viewer and shown in four different views: a 3D view, a sagittal, a coronal, and a horizontal view in which the Fast GrowCut must be executed. In the horizontal view, the user can start with the creation of the input by coloring the desired foreground area, which is tinted red, and the background area, which is tinted green, with the underlying image still visible. Colored areas can be erased with an erase brush or the whole segmentation can be reset. Both brush modes and the eraser mode feature a round brush with variable size, which can be adjusted by dragging the "Brush Size" slider.

To start the segmentation, both foreground and background need to be specified, which is checked by the implementation. After successful calculation of the Fast GrowCut, its output is automatically shown to the user. The user-specified coloration disappears and instead, the output mask is loaded, coloring the calculated foreground and background areas in red and green, respectively.

The user has the option to delete the colored background, leaving only the segmented foreground on display.

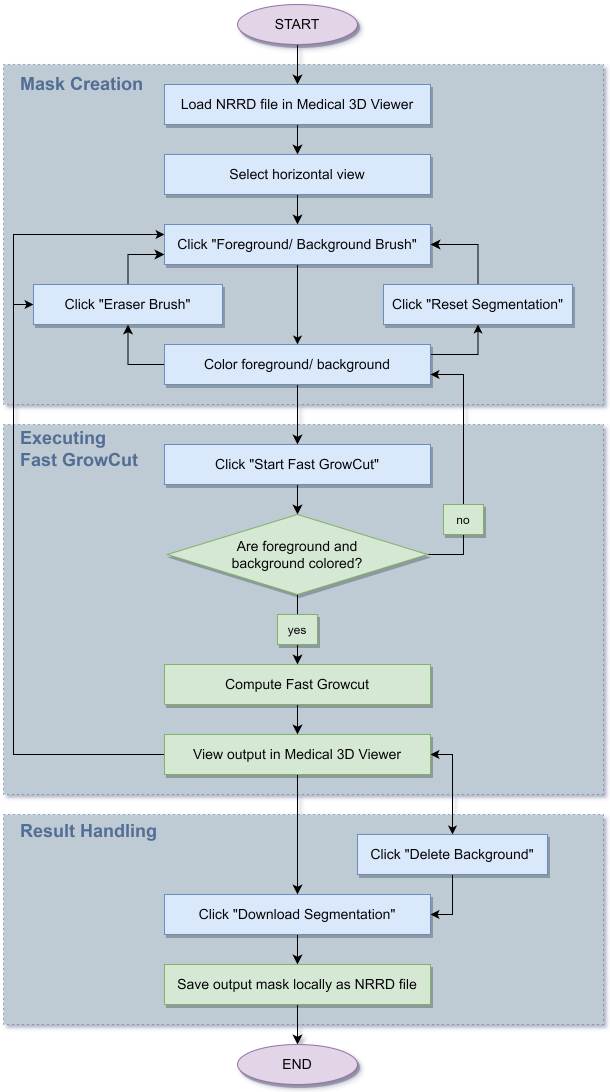


Figure 1: Flowchart demonstrating the Fast GrowCut workflow in StudierFenster. Blue fields represent user input, green fields represent steps executed by the implementation.

The user can further edit the output mask and potentially calculate the Fast GrowCut again if the first output was not satisfactory. Finally, the output mask can be saved locally as an NRRD file where the segmented foreground is colored white and everything else is black.

The Fast GrowCut in StudierFenster can be used through the GrowCut side menu, which contains eight user interface elements, seven buttons, and one slider. A screenshot of the menu is included in Figure 2. It is located with the other segmentation tools in the Medical 3D Viewer side menu and its style, and the implementation of some functions was inspired by other segmentation tools [17].

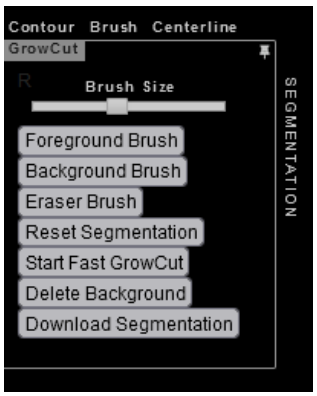


Figure 2: The Fast GrowCut side menu in the Medical 3D Viewer.

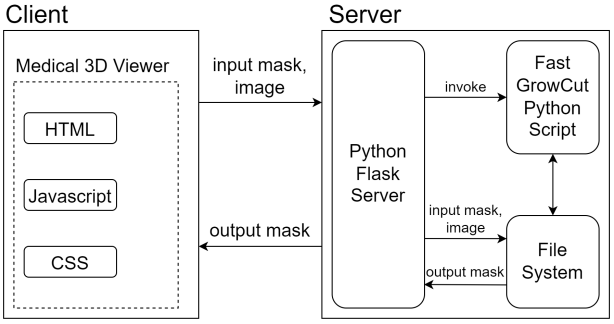


Figure 3: Client/server architecture of the Fast GrowCut implementation in StudierFenster.

4.2 Implementation

4.2.1 Client/Server Architecture

The Fast GrowCut algorithm is implemented in StudierFenster in two parts: a front-end part that is executed on the client side, and a back-end part that is executed on the server side. The client/server architecture is displayed in Figure 3. The client side consists of its interface in the Medical 3D Viewer, which was implemented as HyperText Markup Language (HTML), JavaScript, and Cascading Style Sheets (CSS). The server side consists of the Fast GrowCut implementation and the Flask Server, which is used to communicate between the two sides. The Fast GrowCut is implemented on the server side, because its computation is very intensive and therefore faster and more reliable when done on the server. Furthermore, the algorithm is implemented in a Python script, which cannot be executed in a browser.

The communication between client and server works as followed: The input image and the mask, which are generated on the client side are converted into one JavaScript Object Notation (JSON) object that is sent to the server via an Asynchronous JavaScript and XML (AJAX) request. On the server side, the JSON object is parsed and each of the files is saved individually as an NRRD file in a temporary folder in the file system. The Fast GrowCut Python script then loads the NRRD files from the file system, computes the segmentation, and saves the output mask as an NRRD file in the same folder as the input files. The Python Flask Server waits for the Fast GrowCut to finish, loads the output mask from the file system, and converts it into a JSON object, which is sent back to the client side. Finally, on the client side, the segmentation mask is automatically loaded in the Medical 3D Viewer.

4.2.2 Fast GrowCut Python Implementation

The Python implementation of the Fast GrowCut was adapted from the Python script "growcut_cpu.py" by Shen [2] [18]. It is a Python implementation of Fast GrowCut based on the algorithm by Zhu et al. [5]. The script requires the image data and a seed label array as input [18]. As the Fast GrowCut in StudierFenster is computing two-dimensional segmentations, only the current layer of the image is used for input. The seed label array is of the same size as the two-dimensional input image and contains different labels "1" for foreground and "2" for background. It is generated from the foreground and background information that is drawn by the user that is stored in an HTML canvas element [19]. The output of the Python script is an array of the same size as the seed label array, containing a corresponding "foreground" or "background" label for each array element [18]. This information is mapped to an HTML canvas element, again, for viewing and editing in the Medical 3D Viewer.

5 Results

Figure 4 shows the Medical 3D Viewer with the GrowCut menu opened on the left-hand side, showing the user interface. The loaded image shows an MRI scan of a human brain, the horizontal plane being visible in a big window and the three other views being visible on the right side in small windows.

Results of the Fast GrowCut algorithm used in StudierFenster to segment an image can be seen in Figures 5 and 6. The input masks are visible in Figures 5(a), 6(a) and 6(c) and the corresponding Fast GrowCut results are shown in Figures 5(b), 6(b) and 6(d). In all sub-figures, the foreground is tinted red and the area surrounding the region of interest is tinted green. Figure 5 displays the segmentation of a brain tumor in an MRI head scan and Figure 6 shows the segmentation of a vertebral body in a CT chest scan.

In Figure 6 it is shown that the Fast GrowCut output can be edited before running the Fast GrowCut again with the additional input. Figure 6(a) shows the initial user input and Figure 6(b) shows the corresponding Fast GrowCut

[2] https://github.com/Sherry-SR/fastgc_python/blob/master/modules/growcut_cpu.py

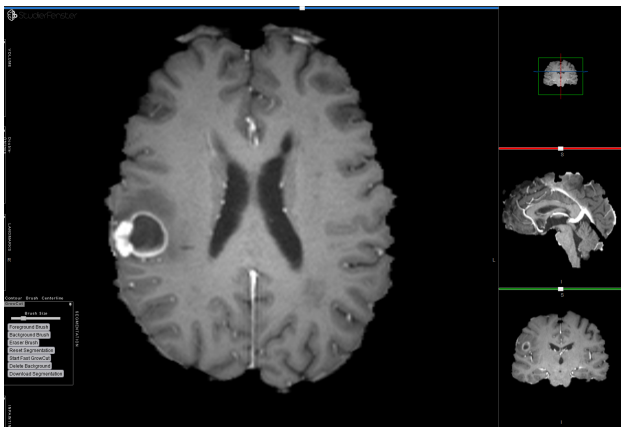


Figure 4: Screenshot of the Medical 3D Viewer with the GrowCut side menu on the left.

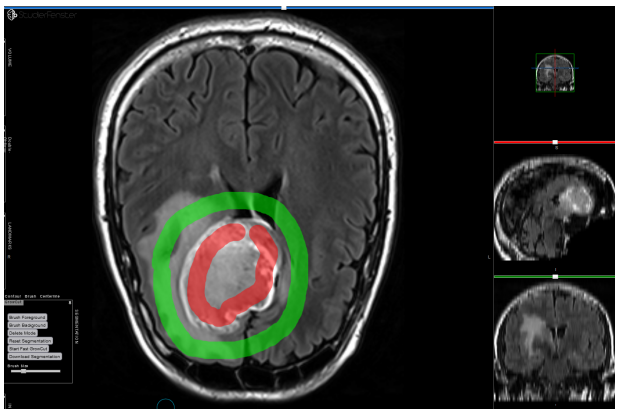
output. The output was edited with the "Delete Mode" and foreground and background brushes, as seen in Figure 6(c) and the Fast GrowCut was run again. Its output is visible in Figure 6(d).

6 Conclusion and Outlook

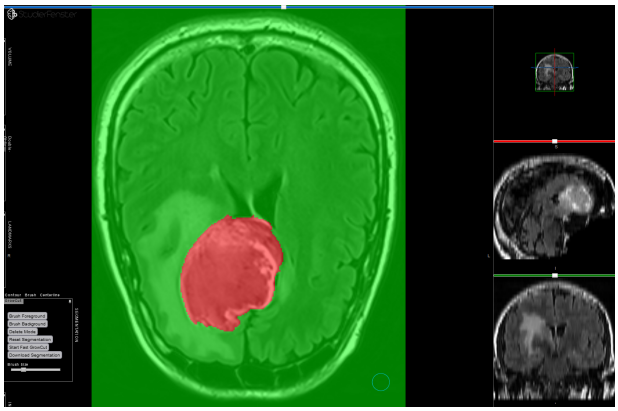
As a result of this contribution, the Fast GrowCut segmentation algorithm was added to StudierFenster, extending the existing segmentation tools, thereby extending the possible use cases.

The Python implementation of the Fast GrowCut by Shen [18] was successfully implemented on the server side of StudierFenster. It was decided to use this implementation of the GrowCut because it could be adapted to consist only of one Python script, which made its applicability simple. Adding this functionality to the architecture of StudierFenster was possible without having to change much of the original Fast GrowCut code. Its straightforward implementation made it stand out against other open-source GrowCut implementations, like the 3D Slicer's "Grow from Seeds" algorithm, for example. The "Grow from Seeds" code is depending heavily on other 3D Slicer modules and functions, which would have made it nearly impossible to extract only the "Grow from Seeds" code, without having to change most of the implementation [16].

After loading an image into the Medical 3D Viewer of StudierFenster, Fast GrowCut can be used to segment any region in the 2-dimensional transverse plane, that has a different color than its surrounding area. For example, a specific use case could be using the Fast GrowCut to segment vertebral bodies, as it was found in a study conducted by Egger et al. in 2017 [13], in which the GrowCut in 3D Slicer was used. Some benefits of using the Fast GrowCut in StudierFenster are its platform and device independency, compared to other implementations like the "Grow from Seeds" algorithm in 3D Slicer [16]. The Fast GrowCut in StudierFenster can be used on various devices, from various browsers. This Fast GrowCut implementation in StudierFenster allows a user to use a graphical interface to select the mask and the output is automatically visible in the original image. This method does not require the user to have any knowledge of Python programming, as opposed to directly using the Fast GrowCut Python script by Shen [18].



(a) User input for tumor segmentation.

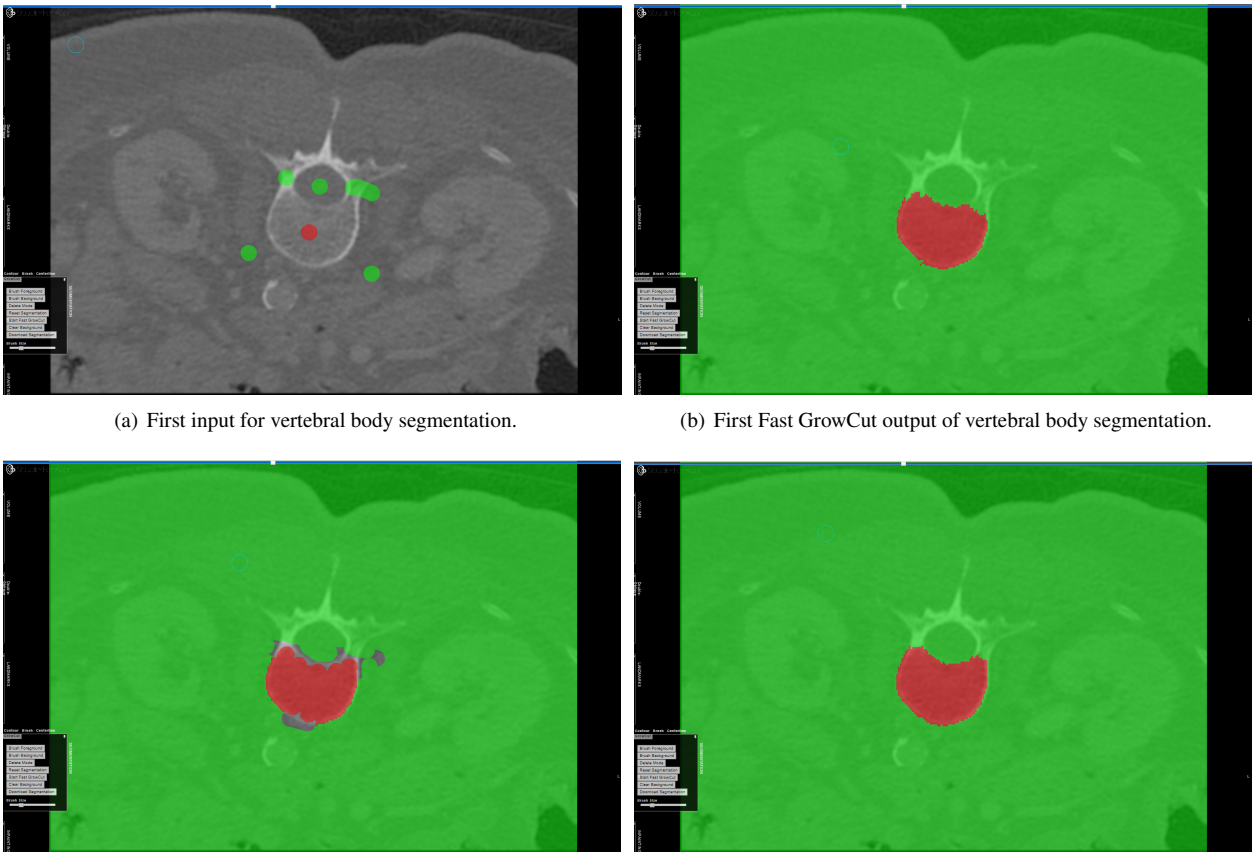


(b) Fast GrowCut output of tumor segmentation.

Figure 5: Segmentation of a brain tumor of an MRI head scan with Fast GrowCut in StudierFenster.

Figures 5 and 6 show that the algorithm successfully segmented the colored regions, based on the user input. However, it is visible that the algorithm did not always find the correct borderline between object and background. This was especially the case when the two had similar colors, or when the border was more of a transition, rather than a line. This can be seen in Sub-figure 6(b), where the red color is spilling outside the region of the vertebral body. In Figure 6 it is visible that the segmentation output is more satisfactory, after iterative refinement by the user. Running the algorithm again is significantly faster than running it for the first time, as the algorithm only iterates through pixels that are not colored.

Future research could examine if allowing user input during the segmentation would accelerate the process of refining the segmentation. This could lead to more sat-



(a) First input for vertebral body segmentation.

(b) First Fast GrowCut output of vertebral body segmentation.

(c) Second input for vertebral body segmentation.

(d) Second Fast GrowCut output of vertebral body segmentation.

Figure 6: Vertebral body segmentation of a CT chest scan with Fast GrowCut in StudierFenster, executing the Fast GrowCut twice.

isfying results after running the Fast GrowCut only once. Allowing user input during the execution of the algorithm would enable users to correct wrongly placed seeds early on.

Furthermore, the Fast GrowCut in StudierFenster could be extended to be used for three-dimensional segmentations, as it is currently only used for two-dimensional segmentations. The implemented Python script of the Fast GrowCut algorithm already allows n-dimensional segmentation [18], however, the process of the input mask creation would need to be extended.

There are open questions, regarding the usage of GrowCut in StudierFenster to segment patient data. The current workflow requires the image data to be stored in the file system on the server side. As this might be undesirable when handling confidential data alternatives should be explored in the future.

Finally, conducting a user study would help to identify flaws in the proposed workflow and could be of assistance in improving the current graphical user interface.

References

[1] Alireza Norouzi, Mohd Shafry Mohd Rahim, Ayman Altameem, Tanzila Saba, Abdolvahab Ehsani Rad, Amjad Rehman, and Mueen Uddin. Medical Image Segmentation Methods, Algorithms, and Applications. *IETE Technical Review*, 31(3):199–213, 2014.

[2] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[3] Jan Egger, Christina Gsaxner, Antonio Pepe, Kelsey L Pomykala, Frederic Jonske, Manuel Kurz, Jianning Li, and Jens Kleesiek. Medical deep learning–a systematic meta-review. *Computer methods and programs in biomedicine*, page 106874, 2022.

[4] V. Vezhnevets and V. Konouchine. "GrowCut" - Interactive Multi-Label ND Image Segmentation By Cellular Automata. *Graphicon*, 1(4):150 – 156, 2005.

[5] L. Zhu, I. Kolesov, Y. Gao, R. Kikinis, and A. Tannenbaum. An Effective Interactive Medical Image Segmentation Method using Fast GrowCut. In *Int Conf Med Image Comput Comput Assist Interv. Workshop on Interactive Methods.*, volume 17, 2014.

[6] H. Veeraraghavan and J. Miller. Modules:GrowCutSegmentation-Documentation-3.6 - Slicer Wiki. Technical report, 2010. `https://www.slicer.org/wiki/Modules:GrowCutSegmentation-Documentation-3.6`, accessed: 2022-10-23.

[7] J. Egger, D. Wild, M. Weber, C. Bedoya, F. Karner, A. Prutsch, M. Schmied, C. Dionysio, D. Krobath, J. Yuan, C. Gsaxner, J. Li, and A. Pepe. Studierfenster: an Open Science Cloud-Based Medical Imaging Analysis Platform. *Journal of Digital Imaging*, 35, 01 2022.

[8] M. Weber, D. Wild, J. Wallner, and J. Egger. A Client/Server based Online Environment for the Calculation of Medical Segmentation Scores. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3463–3467. IEEE, 2019.

[9] Alexander Prutsch, Antonio Pepe, and Jan Egger. Design and development of a web-based tool for inpainting of dissected aortae in angiography images. In *24th Central European Seminar on Computer Graphics: CESCG 2020*, 2020.

[10] Christina Dionysio, Daniel Wild, Antonio Pepe, Christina Gsaxner, Jianning Li, Luis Alvarez, and Jan Egger. A cloud-based centerline algorithm for studierfenster. In *Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications*, volume 11601, pages 201–206. SPIE, 2021.

[11] Jianning Li, Antonio Pepe, Christina Schwarz-Gsaxner, and Jan Egger. An online platform for automatic skull defect restoration and cranial implant design. In *SPIE Medical Imaging Conference 2021*, page 115981Q, 2021.

[12] Florian Karner, Christina Gsaxner, Antonio Pepe, Jianning Li, Philipp Fleck, Clemens Arth, Jürgen Wallner, and Jan Egger. Single-shot deep volumetric regression for mobile medical augmented reality. In *Clinical Image-Based Procedures, 9th International Workshop, CLIP 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 9*, pages 64–74. Springer, 2020.

[13] J. Egger, C. Nimsky, and X. Chen. Vertebral body segmentation with GrowCut: Initial experience, workflow and practical application. *CoRR*, abs/1711.04592, 2017.

[14] Jan Egger, Tina Kapur, Andriy Fedorov, Steve Pieper, James V Miller, Harini Veeraraghavan, Bernd Freisleben, Alexandra J Golby, Christopher Nimsky, and Ron Kikinis. Gbm volumetry using the 3d slicer medical image computing platform. *Scientific reports*, 3(1):1–7, 2013.

[15] L. Zhu. Documentation/4.8/Modules/FastGrowCut - Slicer Wiki. Technical report, 2017. `https://www.slicer.org/wiki/Documentation/4.8/Modules/FastGrowCut`, accessed: 2022-10-23.

[16] C. Pinter, A. Lasso, K. Sunderland, S. Pieper, W. Plesniak, R. Kikinis, and J. Miller. Segment editor - 3D Slicer documentation. Technical report, 2022. `https://slicer.readthedocs.io/en/latest/user_guide/modules/segmenteditor.html`, accessed: 2022-10-23.

[17] D. Wild, M. Weber, and J. Egger. A Client/Server Based Online Environment for Manual Segmentation of Medical Images. In *The 23rd Central European Seminar on Computer Graphics (CESCG)*, pages 1–8, 2019.

[18] R. Shen. Fast Growcut alogorithm with shortest path. Technical report, 2019. `https://github.com/Sherry-SR/fastgc_python`, accessed: 2022-12-27.

[19] Mozilla and individual contributors. ImageData - Web APIs — MDN. Technical report, 2022. `https://developer.mozilla.org/en-US/docs/Web/API/ImageData`, accessed: 2022-10-19.