# Segmentation of 3D Dental model

# Motivation:

Segmentation is an essential part of most of the clinical and research related projects for determining and separating your region of interest in many applications like 3D printing, surgery planning, visualization and machine learning training. Currently manual or semi automates segmentations are mostly being employed but fully automated segmentation method is not being employed yet as a comprehensive technique but are continually developed, however it is unrealistic to assume that a perfectly accurate and comprehensive method will arise in the near future. To this end, the results of the automated segmentations need to be corrected, as well as new segmentation algorithms developed. Correction of faulty segmentation necessitates manual editing. Training data for state-of-the art segmentation methods (*i.e.* multi-atlas and deep learning-based algorithms) also need to be created manually or semi-automatically.

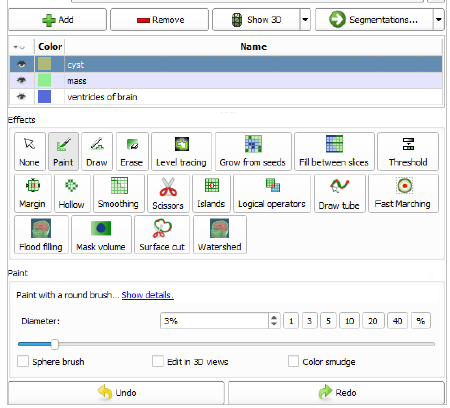
Labelmap volume – *i.e.* a 3D image containing multiple structures (labels) indicated by its different voxel values – are typically used in segmentation tools, such as 3D Dental’s legacy Editor module, or ITK-SNAP (Yushkevich et al., 2006). While it is tempting to choose labelmap volume as the edited representation due to its simplicity and memory efficiency, the fact that it does not support overlapping labels makes it impossible to use it as a generic storage format. This is also the main reason why the DICOM Segmentation IOD standard does not rely on the labelmap volume concept. Usage of individual volumes to store segments allows having overlapping segments and storing masks or intermediate processing results in the same segmentation object. Segments may contain each other (*e.g.* body > teeth > planning target volume > clinical target volume > gross target volume), and margins can be added without the risk of losing data in other structures.

While labelmap volumes are simple and can be stored and processed efficiently, segmentation objects have comparable performance in most cases. The storage cost of the segmentation object can be lower than that of the labelmap volume containing the same structures if the structures are relatively small and are not or only somewhat overlapping and can be higher when many segments extend over large areas in the volume. However, as higher storage cost mostly occurs in the case of overlaps, a fair comparison cannot be made due to the inability of labelmap volumes to represent overlapping structures.

Existing freely available image segmentation applications such as the abovementioned ITK-SNAP and 3D Dental’s legacy Editor, as well as the segmentation tools in other platforms. Considering this, PolySeg seemed a promising basis to a redesigned manual and semi-automated segmentation application to better support the typically overlapping RT structure sets. PolySeg’s design not only enables handling overlapping structures, but its automation features facilitate streamlining workflows in terms of visualization, data management, transformation, and export.

# Segment Editor overview

*Segment Editor* within 3D Dental intending it to be one of the most versatile, flexible, and user-friendly freely available medical image segmentation tools. Segment Editor is a reusable widget that can be added into any 3D Dental module or application and customized, but it also has its own user module with identical name for easy access to the widget with the default settings.



Editing can start from an empty or existing segmentation, and after specifying a *master volume*, which is typically the anatomical image to segment, and which lends its geometry to the new segment labelmaps, the editor *effects* can be used to create and edit the delineation. The master representation needs to be binary labelmap in order to allow editing. If a different master is found, a dialog asks the user whether they approve converting and setting the master to binary labelmap. Full support for fractional labelmaps will follow soon as future work. Using PolySeg as the basis for Segment Editor implicitly enabled major unique features that make it stand out from its competition:

**Overlapping structures**

## As mentioned above, one of the main advantages of relying on PolySeg in an image segmentation application is the ability to manage overlapping structures implicitly. When using traditional labelmaps, this is only possible through workarounds and complex application logic, unavoidably leading to errors and a codebase that is getting more difficult to maintain with each additional feature. Before designing and implementing PolySeg, the SlicerRT toolkit used a module called *Contours* to manage the different representations. Handling the diverse representations was one of the first obvious issues that arose when we were facing the complexities of dealing with RT datasets. This module tried to keep different MRML node types in sync, representing the same structures. One of them, in the absence of a way to efficiently store individual structures in binary labelmap representation, was a classic multi-label labelmap volume. This single node needed to be kept in sync with multiple model nodes containing individual surface meshes and point data for the closed surface and planar contour representations, respectively. Besides the inability to store the overlapping structure information, this synchronization task became unfeasible after covering more and more RT workflows. In contrast, PolySeg now makes efficient management of overlapping structures a trivial task.

## **Real-time 3D visualization**

Leveraging the automated conversion mechanism, Segment Editor can visualize the edited segmentation in 3D without any repeated user interaction. 3D surface visualization is useful, because it allows seeing the out-of-plane topology and general shape of the edited segments while performing the segmentation in a distraction-free manner. When the user checks the *Show 3D* button, the closed surface representation is generated from the current state of the edited binary labelmap using the default conversion parameters. Once a closed surface representation is created, the automatic conversion mechanism makes sure that every time the master binary labelmap representation is changed (*i.e.* editing happens), the surface is also updated.

The current implementation enables real-time 3D visualization experience using the default parameters, but only if the number and extent of segments do not reach a certain point. Once, however, the conversion task becomes computationally more and more intensive, one conversion cycle may become uncomfortably long. The ultimate solution will be parallelization, allowing conversion to be executed in the background. It is still possible with the current implementation, however, to speed up conversions by a factor of ten or more, by disabling the surface smoothing post-processing conversion step: The Show 3D button’s advanced options include a slider with access to the smoothing factor, which, when set to 0, no smoothing is performed. Hence, near real-time experience can be maintained even for large segmentations.

Another considerable advantage of the synchronized automatic creation of the closed surface representation is the ability to use certain editing tools in the 3D display. Since the 2D views only allow seeing one slice of the image and the segments, they do not make it possible to see the segmentation. Sometimes, however, it is very convenient to use the 3D view to make changes to the segments, *e.g.* removing parts or cutting them into pieces.

## **Direct file export**

An important use case of image segmentation is 3D printing, and after the first release of Segment Editor it quickly became clear that a considerable part of the 3D Slicer community using the new segmentation application uses it for 3D printing. The automatic conversion mechanism and the storage classes in the application layer made it trivial to add direct export feature for the typical input format of 3D printers (*i.e.* STL) in Segment Editor. With the users having direct access to this feature in the sub-menu of the “*Segmentations...*” button (which otherwise takes the user to the Segmentations module for advanced management of their segmentation node) they have a shortcut for a frequent operation in their workflow.

**Control over geometry**

One of the issues that frequently arises in a variety of use cases is the ability to control the image geometry (also known as the grid) of the edited binary labelmap representation. Reasons to do this include the need to a) have isotropic (*i.e.* cube-shaped) voxels, b) have a segment resolution greater than the reference image to capture details otherwise impossible, or c) manually specify image properties when there is no reference/master image available (*e.g.* when needing to edit a surface model with a certain resolution).

A reusable UI widget has been created for this purpose, which is available from Segment Editor. It allows selecting a MRML node as the source geometry. If the selected node is a volume, then its geometry will serve as starting point, and oversampling factor can be specified or isotropic spacing requested. Otherwise spacing can be manually entered. In the latter case axis directions will coincide with the main anatomical axes, origin will be set according to the source node’s bounding box, and the effective extent will make sure that only the necessary memory is allocated.

## Advanced masking options

The versatile functionality of the vtkOrientedImageDataResample class (see also subenables the user to utilize advanced masking options to have exact control over which parts of the segments can change. The masking panel appears below the effect properties panel and the undo/redo buttons. It offers three settings:

* *Editable area*: The region where edits are allowed. It has various settings: everywhere (default), inside all segments (useful to split a segment), inside all visible segments, outside all segments (useful to preserve existing segments when overwrite is enabled), outside all visible segments. It also offers a list of the segments, from which choosing one will allow editing only inside the selected segment.
* *Editable intensity range*: If checked, then a double-ended slider appears, which can be used to specify voxel values used in the master volume. The region inside the voxels of the master volume with values falling within the selected range are the ones that can be edited. Off by default.
* *Overwrite other segments*: Its options are all segments, visible segments, and none. The default selection is “all segments”, which basically specifies non-overlapping segments. The default was selected to make its operation more familiar for users used to traditional segmentation tools.

These masking options allow fine control over the editing process and can potentially speed up segmentation considerably. Using them in an efficient way, however, requires a certain level of experience with Segment Editor, so generally they are recommended to more advanced users.

## **Standard terminologies**

3D Slicer offers a module called *Terminologies* (also created by the author as part of one of the grants that made this work possible) that supports using standard clinical terminology dictionaries such as SNOMED Clinical Terms (Stearns et al., 2001). These terminologies are carefully assembled lists of multi-level entries that can be assigned to data objects; in our case the segments. This means that instead of relying on arbitrary names, standard codes can be assigned to segments (*e.g.* type: Morphologically altered structure / Necrosis, region Cerebral cortex / Left), which allows automatic look-up and grouping of structures in a database. This is highly useful when building large training datasets for machine learning based methods or sharing data between institutions, because oftentimes the structure naming conventions are different, and the operators make mistakes in naming. Using standard terminologies circumvents the need to rely on manual curation or heuristics.

When double-clicking on the color swatch in the row of a segment, the terminology selector dialog pops up instead of a conventional color selector, to make use of this application-level feature. The default dictionaries available are the full SNOMED dictionary and a subset of the dictionary that is based on 3D Slicer’s “generic anatomy color table”, a carefully curated limited list of organs and anatomic structures. The dictionaries are stored in JSON files. It is also possible to provide custom

dictionaries, which is useful to create a shortlist for site-specific segmentation and can be done by simply cloning and trimming one of the default dictionaries. The terminology entries are stored persistently as segment metadata.

**Unique editing tools**

The reworked architecture of Segment Editor enabled a general speed-up and richer features for traditionally existing editing tools (effects), as well as creating novel ones. A non-exhaustive list of noteworthy effects follow.

*Paint*: the classic editing brush received improvements making it more suited for the flexible editing and data management capabilities of the editor. It can paint in 3D using the “Sphere brush” option, even in the 3D views. The paint effect can also handle non-axis-aligned (oblique) slices, when the 2D view in which editing happens is not aligned with the main coordinate axes.

* *Grow from seeds* (a.k.a. GrowCut): a semi-automated segmentation tool based on the work of Zhu et al. (2014), which allows segmenting structures in any anatomic site provided there is contrast between them. The user needs to create segments for each structure and one for the background, and initialize each of them with a few painted areas, favorably by the more difficult edges. The initial segmentation is calculated within seconds, after which the user can make additions and deletions in the areas where the algorithm erred. A few iterations are typically enough to make an accurate segmentation of even large and complex structures.
* *Fill between slices*: Another highly useful semi-automated editing effect. It relies on a sparsely painted series of contours, which it completes using morphological contour interpolation (Zukic et al., 2016) to form a closed shape. The input contours can be of any orientation. This effect, similarly to Grow from seeds, is also capable of presenting a preview in both 2D and 3D, and automatically updating it when the inputs change.
* *Scissors*: a versatile new effect that can be used in both 2D and 3D to cut slabs of space. Its options allow erase inside or outside and fill inside and outside of the marked region. Marking can be done using a free-form lasso-like tool or using a circle or rectangle. The direction and range of the action can also be specified. The projection of the camera in the 3D view is considered when cutting. The effect is very useful to remove unnecessary parts of other automated effects such as *Threshold* (which marks voxels falling in a specified intensity range), or creating new objects, such as supporting columns of printed phantoms.
* *Surface cut*: this effect can create closed surfaces from a point cloud in space. The user can place points in either 2D or 3D views, which are used as input to generate the surface. The surface then can be used to cut the space similarly to the Scissors effect: set, erase inside/outside, or fill inside/outside. The points are serialized into segment metadata, so the input can be reconstructed even if the MRML scene is loaded from file.
* *Draw tube*: a tool similar to Surface cut in that it uses a recoverable list of input points to generate the segment. Instead of a convex surface, this effect creates a tube. Options include radius and interpolation method, such as linear, moving polynomial (default), etc.
* *Logical operators*: A simple, but essential effect for combining and modifying segments. Its operating modes are copy, add, subtract, intersect, invert, clear, and fill. It can be set to use or ignore the masking options.
* *Joint smoothing*: This feature is one of the modes of the *Smoothing* effect besides median, opening, closing, and Gaussian. The typically used smoothing options tend to shrink the structure, even the state-of-the art methods, including the one used in the binary labelmap to closed surface conversion algorithm (see subsection 3.4.1). This mode is unique, because it preserves watertight interface between the segments. Keeping the structures touching each other can be useful for removing noise or segmentation errors while preserving topology, as well as supporting legacy features such as traditional multi-label labelmap volumes.

# Implementation in 3D Dental:

1. First step in implementations is to load a CT dataset and convert the data set into 3D Model using volume rendering.

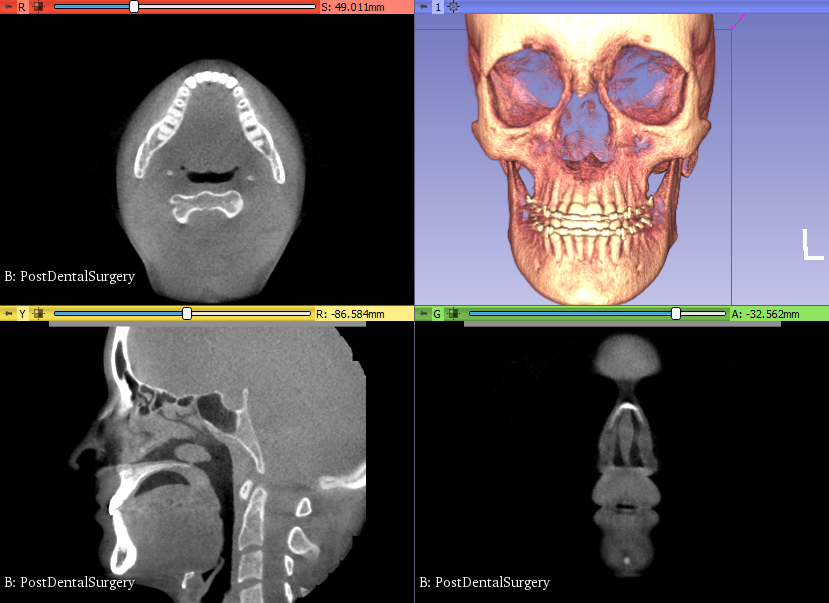


Figure 1: Converting CT model into 3D Model

1. Second Step is to crop the required area for example required dental area.

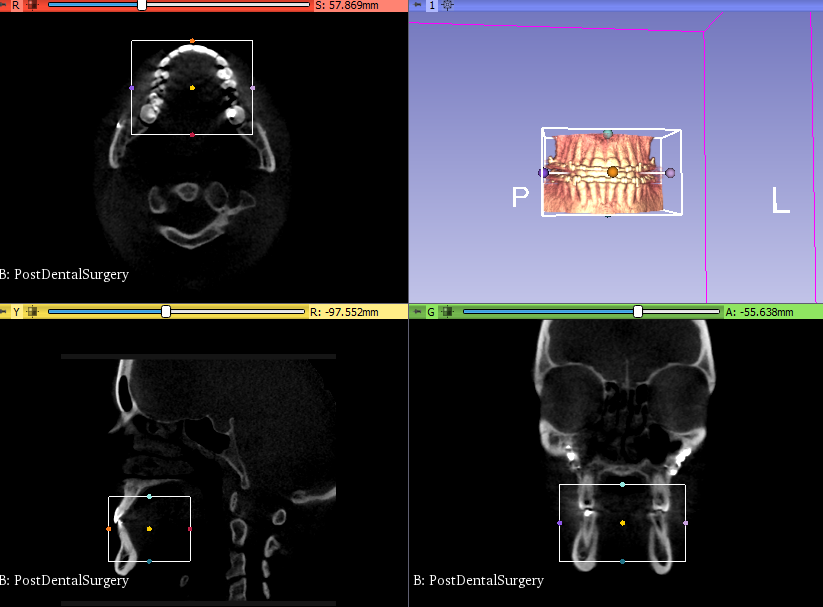


Figure 2: After selecting the required area for cropping

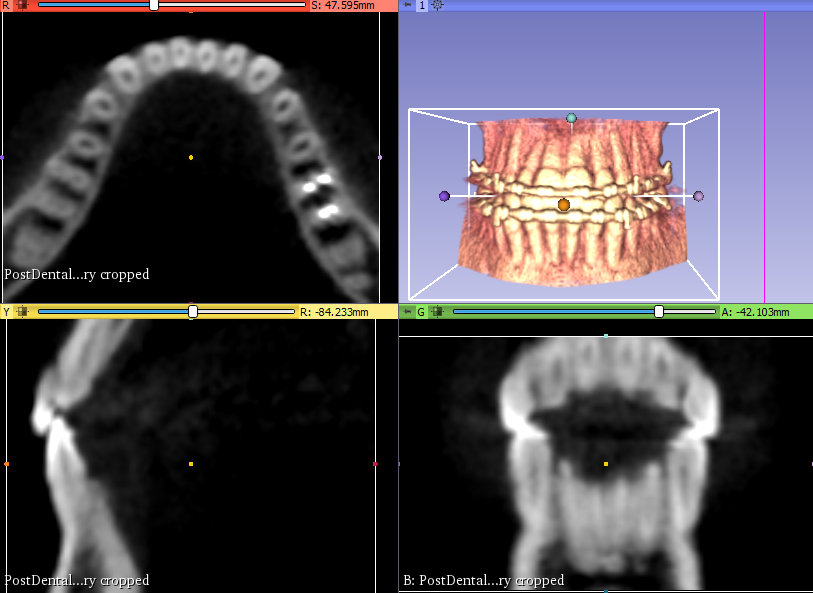


Figure 3: After applying cropping on the required area

1. After cropping the area, we can apply segmentation using segment editor toolbox. Here we can use different tools to fine tune our required area of interest by increasing or decreasing the density and selecting and deselecting required parts for segmentation. Segmentation results using GPU Ray Casting and VTK Ray casting are shown in below figures.

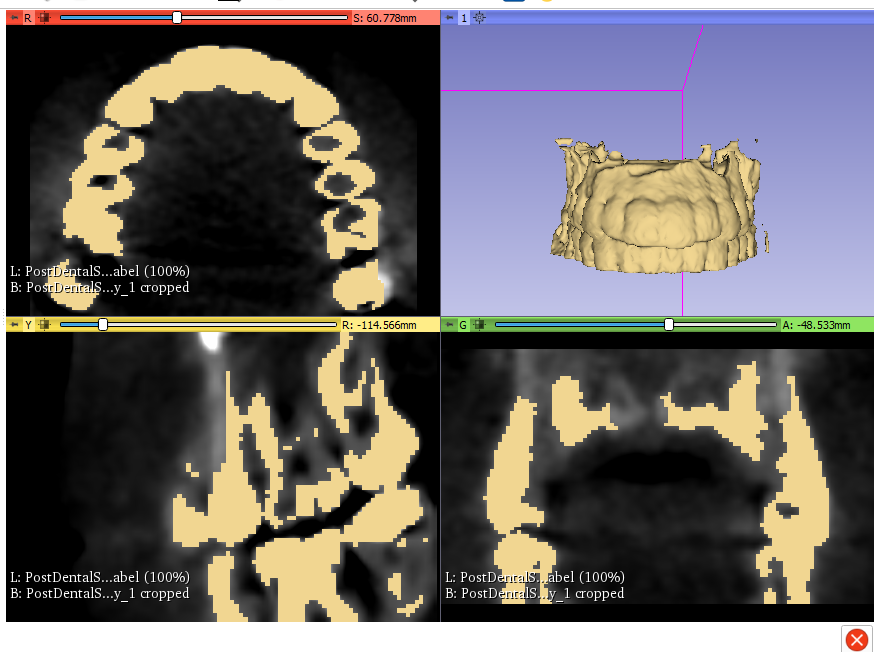


Figure 4: Segmentation Results using GPU Ray Casting

A screenshot of a cell phone

Description automatically generated

Figure 5: Segmentation results using VTK Ray casting