Kitware

BoneTextureExtension

User Tutorial

Jean-Baptiste Vimort, et al.

UNIVERSITY OF MICHIGAN

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

CONTENTS

- Overview
  - Introduction
  - Installation
  - The modules in 3DSlicer
- The interactive modules
  - Bone Texture
  - Bone Texture Serializer
- The Computational modules
  - Compute GLCM Features
  - Compute GLCM Feature Maps
  - Compute GLRLM Features
  - Compute GLRLM Feature Maps
  - Separate Vector Image
- Parameters
  - Experimental conditions
  - Default parameters
  - Mask
  - Number of bins
  - Voxel intensity Range
  - Distance Range
  - Neighborhood radius
- Resources, and Contact
- Acknowledgements

Kitware

Overview

- Introduction
- Installation
- The modules in Slicer

Introduction

The goal of this 3DSlicer extension is to allow textural analysis of 3D volumes.

For that two well-known texture analysis methods are used:

- the study of Grey Level Co-occurrence Matrix (GLCM)
- the study of Grey Level Run Length Matrix (GLRLM)

Two different types of outputs can be computed:

- ***textural features****:* this type of 1D feature set will be computed over the whole volume and thereby characterize the texture of the entire 3D volume
- ***textural maps****:* for each voxel of the input volume, a set of textural features will be computed for the neighborhood of this voxel. The output will be a 3D texture map where each voxel locally describe the texture of the input volume

Introduction

BoneTextureExtension is composed of 7 modules:

- *Two main modules:* These modules allow the user to run all the algorithms from the same modules. Users can easily modify the parameters, visualize the results, and recursively run the algorithms on an input set.

- *Five secondary modules:* Each of these modules runs a single algorithm. Used by the two main modules, they are useful to have better computation time. If needed these modules can also be used outside of Slicer using command line.

Introduction

The two main modules of BoneTextureExtension are:

- *BoneTexture:*
  - four textural analysis algorithms available
  - runs on a single case
  - immediate visualization of the results in Slicer
  - easy to modify the input parameters and re-run

- *BoneTextureSerializer:*
  - four textural analysis algorithms available
  - can run on several cases
  - saves all results (no immediate visualization in Slicer)
  - easy specification of the input results

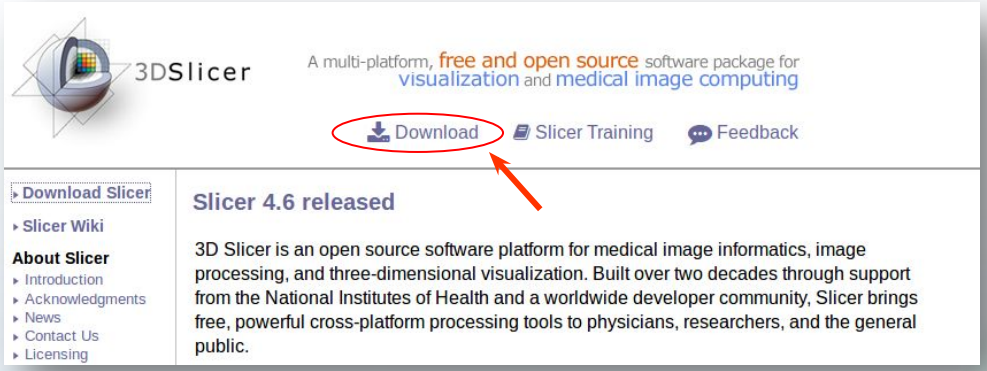
Introduction

The five secondary modules are:

- *ComputeGLCMFeatures:* Computes the textural features over the whole image using the Grey Level Co-occurrence Matrix
- *ComputeGLCMFeatureMaps:* Computes the textural feature maps using the Grey Level Co-occurrence Matrix
- *ComputeGLRLMFeatures:* Computes the textural features over the whole image using the Grey Level Run Length Matrix
- *ComputeGLRLMFeatureMaps:* Computes the textural feature maps using the Grey Level Run Length Matrix
- *SeparateVectorImage:* Separates the feature maps contained in a single ND 3D file (like in a dwi volume) into several separated modules

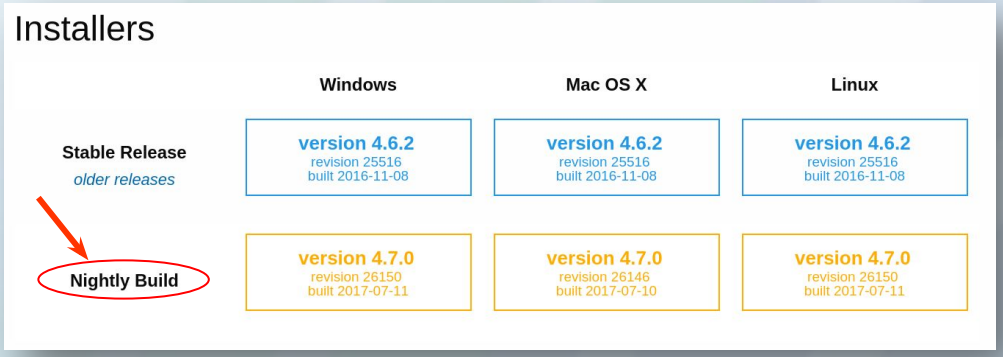
Installation

BoneTextureExtension is a plugin extension only available on the nightly version of Slicer as of 07/10/2017, if you don't already have a nightly version of Slicer on your machine, you should download one on the 3D Slicer website and install it:

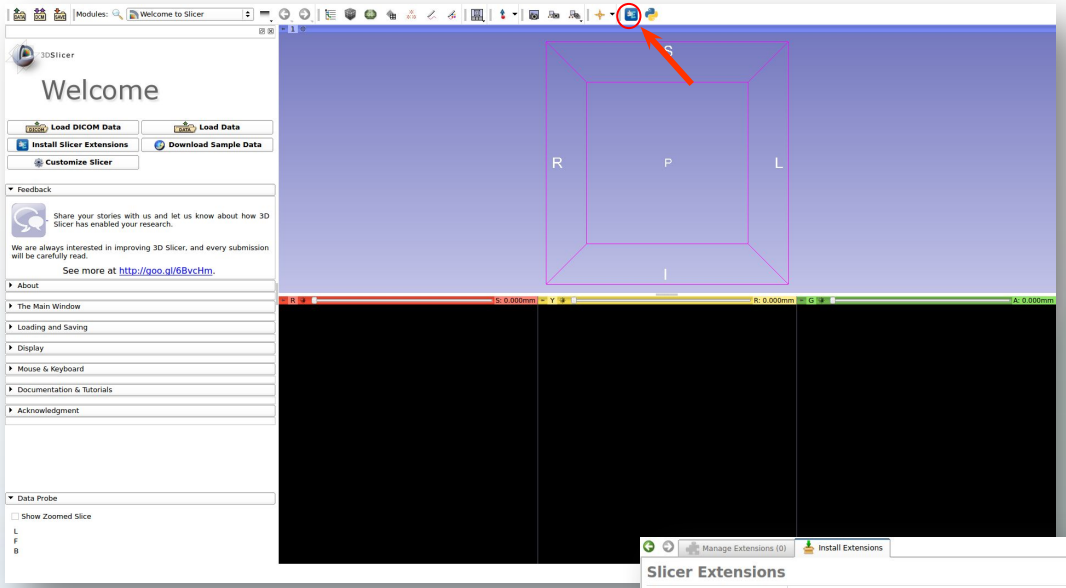


- Go to the Slicer download page

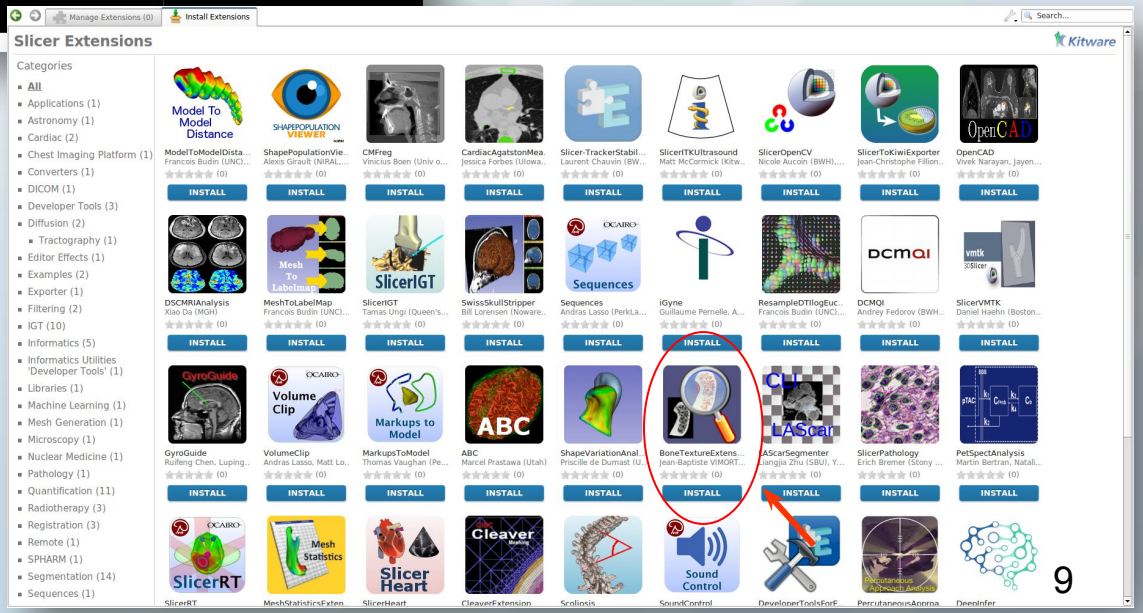
- Download the nightly version of 3DSlicer corresponding to your operating system and install it



Installation



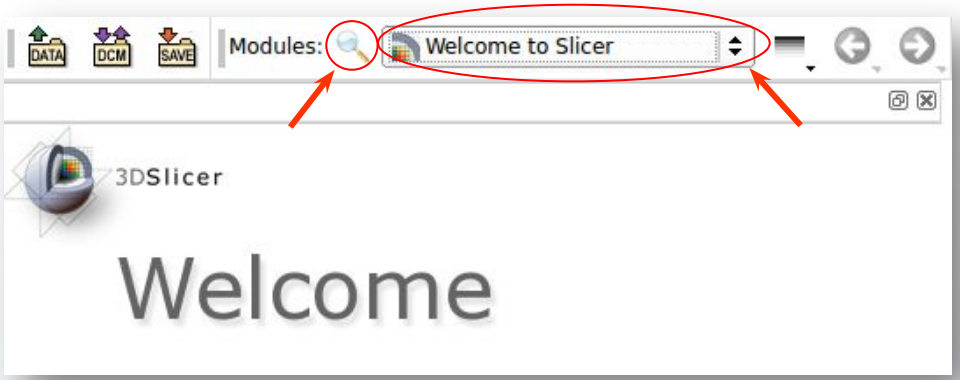
- Once you launch 3DSlicer, open the extension manager



- In the Extension Manager, look for BoneTextureExtension and install it by clicking on the “install” button under it, then restart Slicer

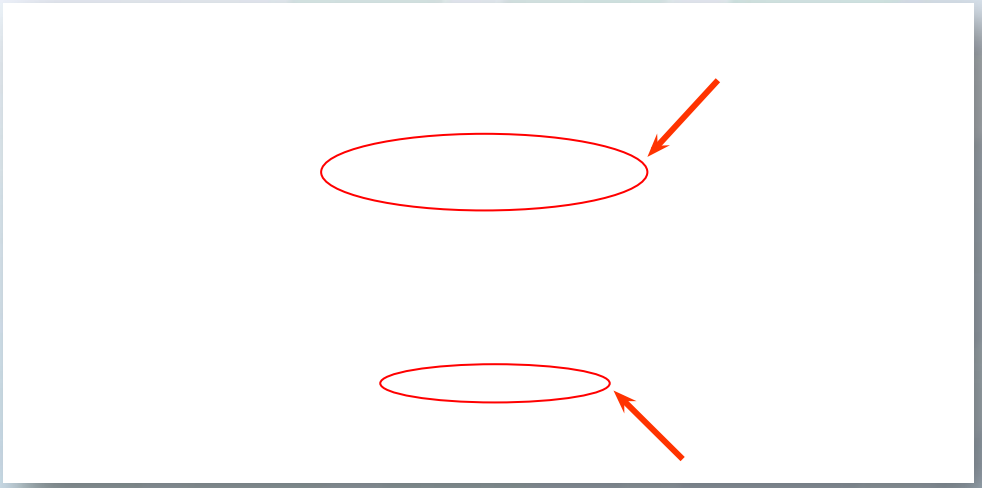
The modules in Slicer

In order to find the modules in Slicer you can either:



- Click on the search button and look for one of the modules by writing its name OR
- Use the drop-down menu to list all the installed modules

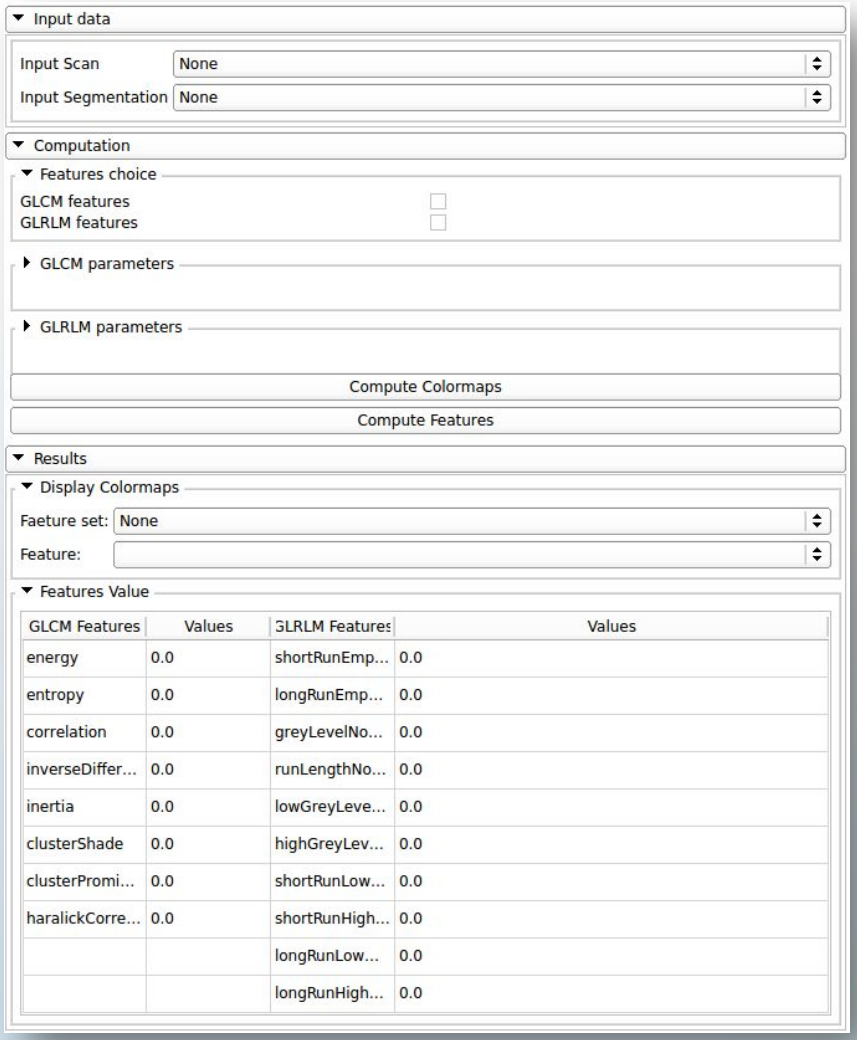
- In the drop-down menu, the two main modules are located in the “quantification” tab, and the secondary modules are located in a sub-tab called “Texture Feature” in the “quantification” tab



The interactive modules
(scripted modules)

- Bone Texture
- Bone Texture Serializer

Bone Texture



The BoneTexture module is composed of three main parts:

- *The input data tab:* used to specify the input volume and the input segmentation (if it exists)
- *The computation tab:* used to specify the type of results wanted, as well as the parameters for each algorithm
- *The results tab:* used to observe the results

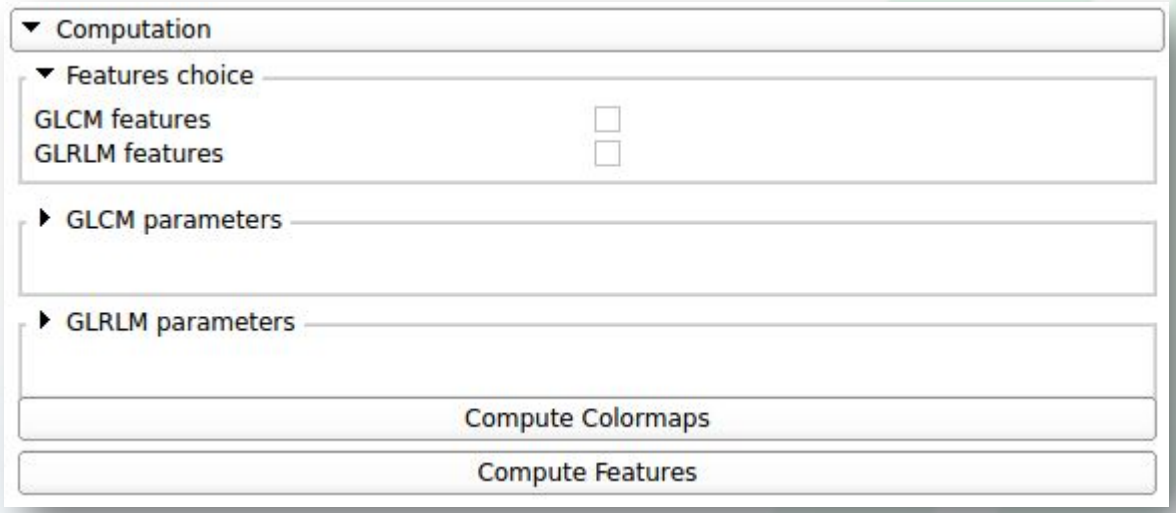
Bone Texture

The input section allows the user to specify:

- *An input scan:* it's interesting to notice that the computation time will improve if the useful data is taking as much space as possible in the input scan. Tip: Consider a preliminary cropping step to erase the useless data on the boundaries, and decrease computation time.

- *An input mask:* the mask is optional, if given it will reduce the computation time of the algorithms.

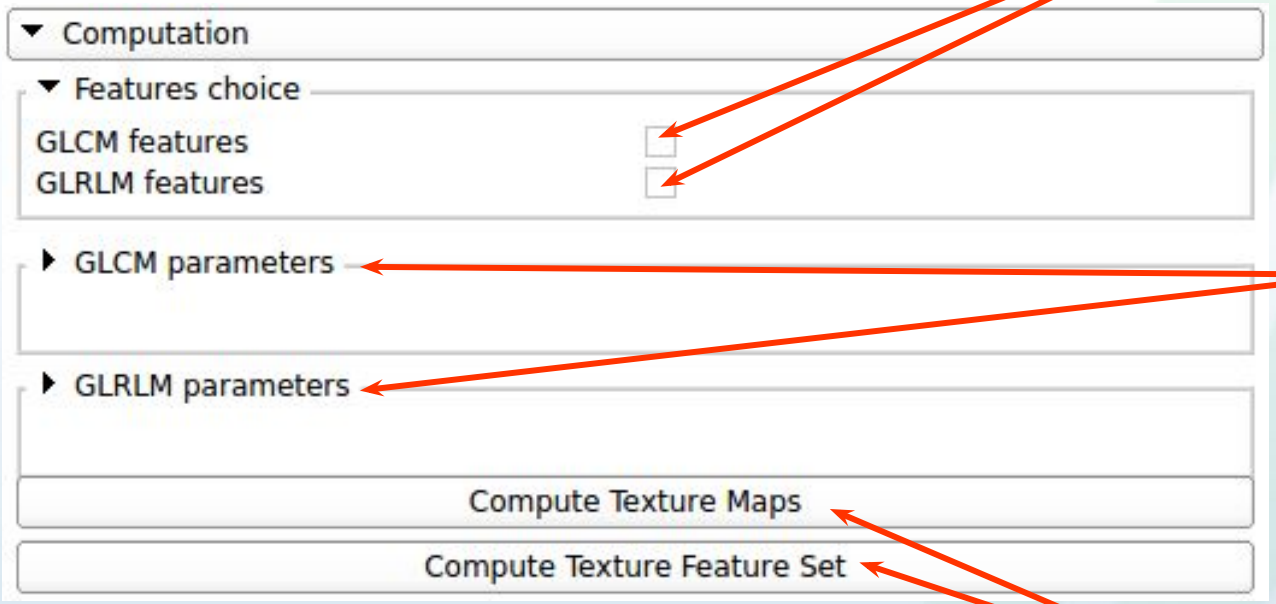
Bone Texture



This section allows the user to choose:

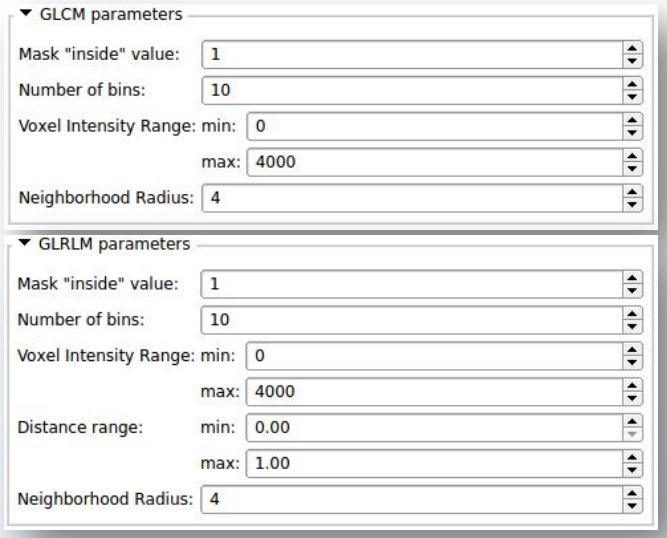
- The type of features wanted: Run length features and/or co-occurrence features
- All the parameters necessary to run each algorithm
- The type of outputs wanted: texture features and/or texture maps

Bone Texture



- Check the type of feature desired
- Specify the input parameters for each algorithm
- Click on the button to obtain the corresponding features

Bone Texture

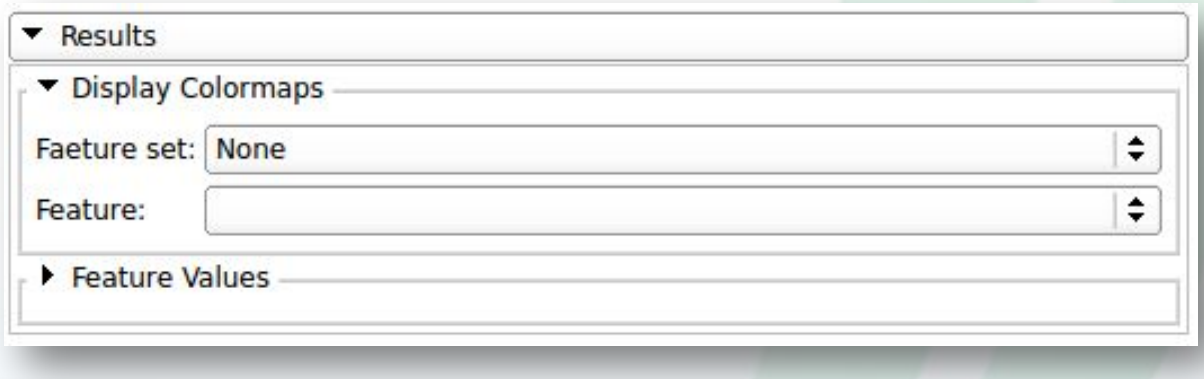


There is more description about the influence of each parameter later in the presentation

The GLCM and GLRLM sub-tabs allow to choose one of algorithms: (more description about the influence of each parameter later in the presentation)

- *Inside Mask Value:* The pixel value that defines the ”inside” of the mask
- *Number of Intensity bins:* The number of intensity bins in the Grey Level Matrices
- *Pixel Intensity Min:* Minimum of the pixel intensity range over which the features will be calculated
- *Pixel Intensity Max:* Maximum of the pixel intensity range over which the features will be calculated
- *Distance Min:* Minimum of the distance (in mm) range over which the features will be calculated
- *Distance Max:* Maximum of the distance range (in mm) over which the features will be calculated

Bone Texture



The results section allows the user to observe:

- *The texture features:* they will be displayed using a table.
- *The texture maps:* the user will be able to choose the dwi file (file generated by the algorithms) containing the texture maps, and the feature to display in the visualization window of Slicer.

Bone Texture

Texture Features Output

GLCM Features	Values	GLRLM Features	Values
energy	0.068...	shortRunEmphasis	1.0
entropy	4.322...	longRunEmphasis	1.0
correlation	0.203...	greyLevelNonuniformity	14362.6
inverseDifferenceMoment	0.823...	runLengthNonuniformity	105010.0
inertia	0.381...	lowGreyLevelRunEmphasis	0.068747
clusterShade	39.29...	highGreyLevelRunEmphasis	33.1306
clusterProminence	840.5...	shortRunLowGreyLevelEmphasis	0.068747
haralickCorrelation	6464....	shortRunHighGreyLevelEmphasis	33.1306
		longRunLowGreyLevelEmphasis	0.068747
		longRunHighGreyLevelEmphasis	33.1306

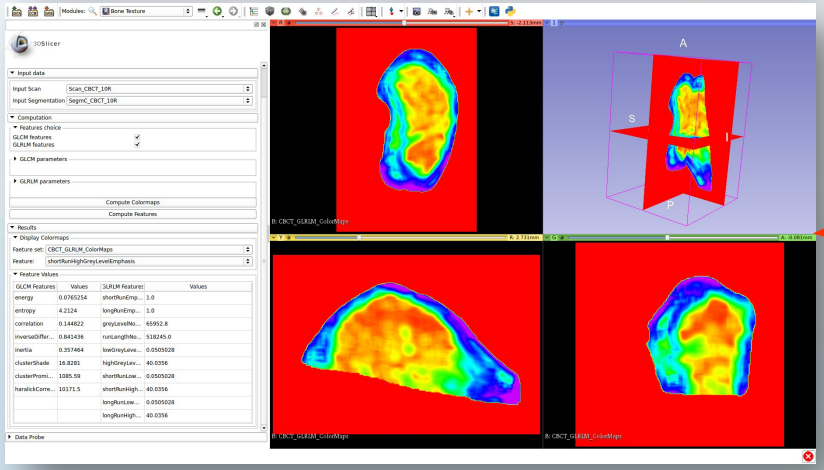
Each texture features computed over the whole input volume will be displayed in this table. Only the selected features will be computed and displayed.

Bone Texture

Feature Maps Output

- Select feature set (saved as dwi image) containing the different feature maps

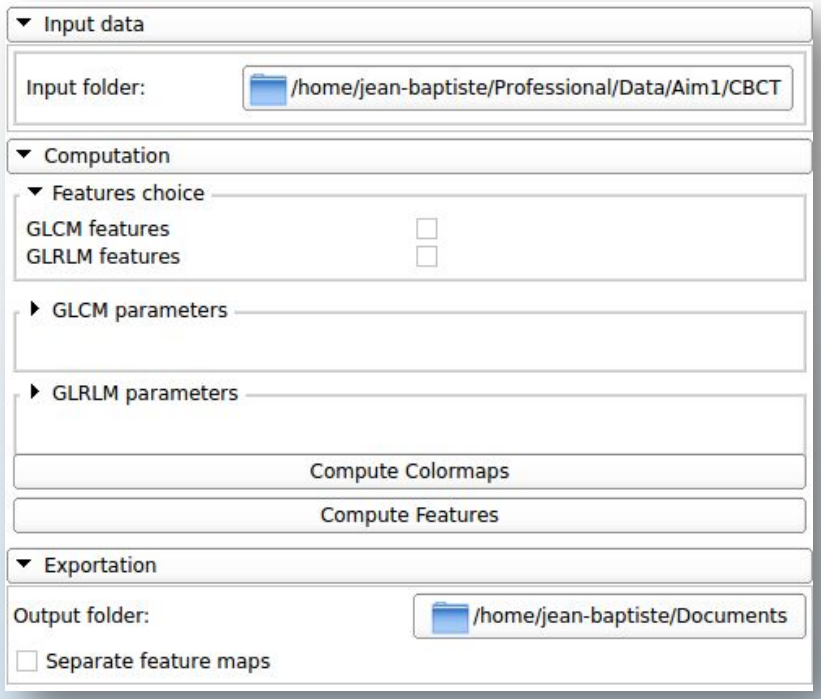
- Choose in this feature set, the feature that you want to display



- The feature map will be displayed in the visualization window of Slicer

Kitware

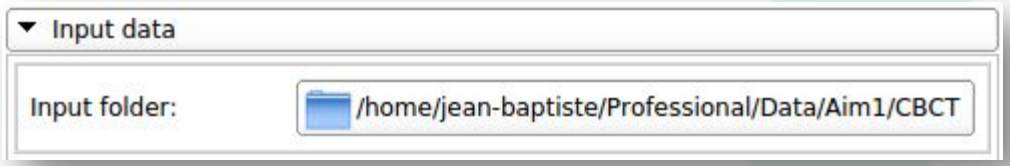
Bone Texture Serializer



The BoneTextureSerializer module is composed of three main parts:

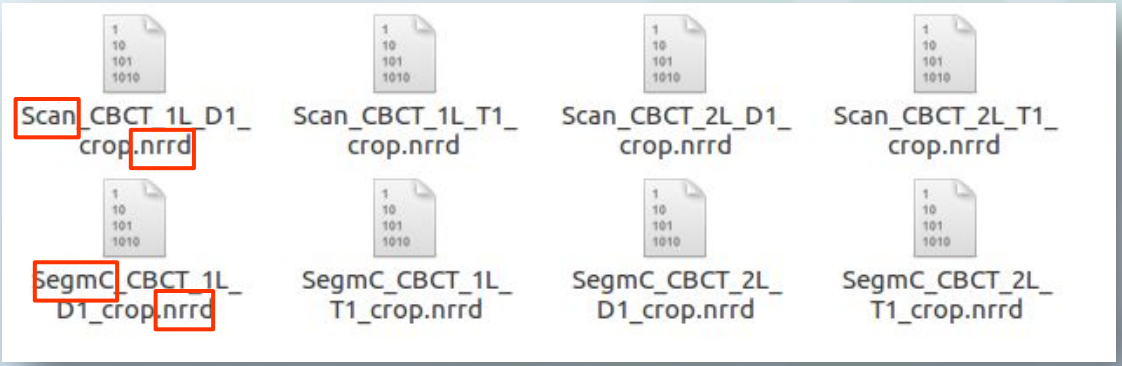
- *The input data tab:* used to specify the folder containing the input volumes and the input segmentations (if they exist)
- *The computation tab:* used to specify the type of results wanted as well as the parameters for each algorithm
- *The exportation tab:* used to specify the output folder where the results will be saved

Bone Texture Serializer



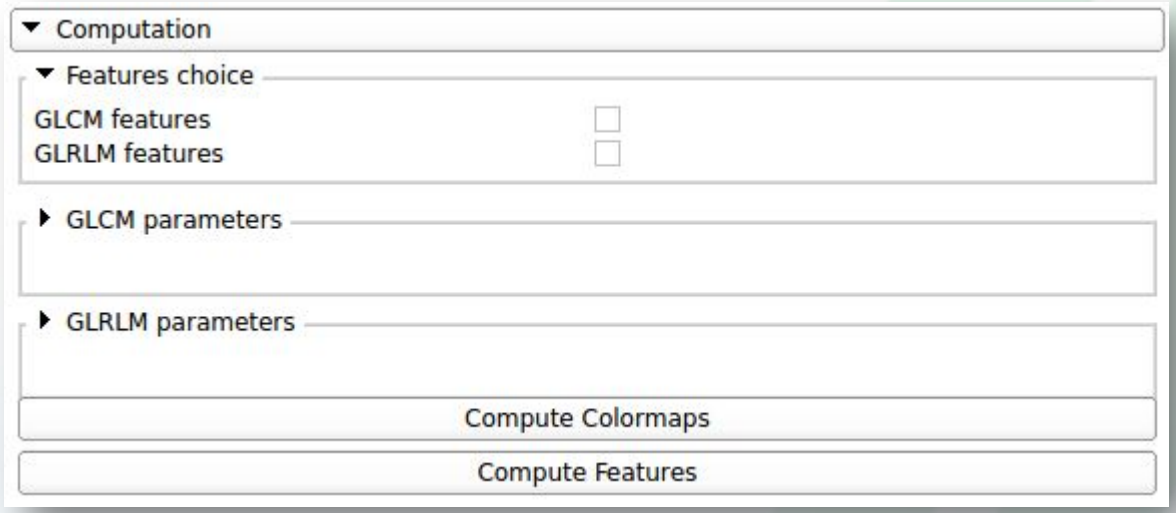
The input section allows the user to specify:

- An input folder that should contains all the input volumes and segmentations; (for a better computation time, the inputs should be cropped as suggested in the BoneTexture input slide)
- The input scan should be named ScanXXX.nrrd, where XXX is the ID of the case
- If it exists, the input mask should be named SegmCXXX.nrrd, where XXX is the ID corresponding to the the input scan



Kitware

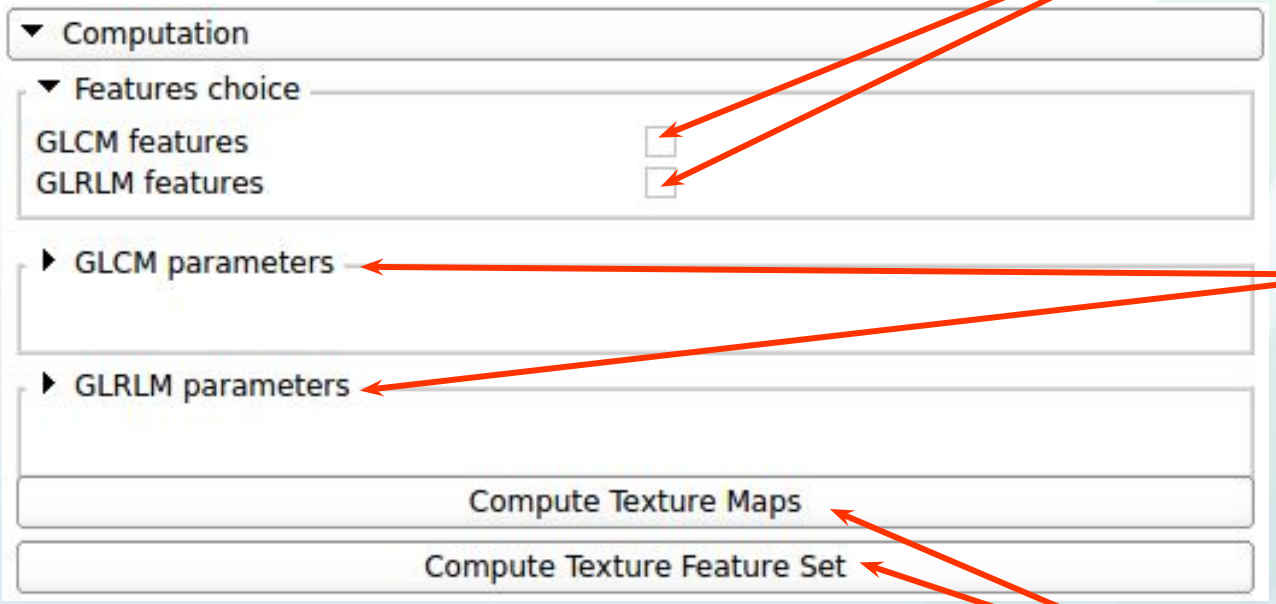
Bone Texture Serializer



This section allows the user to choose:

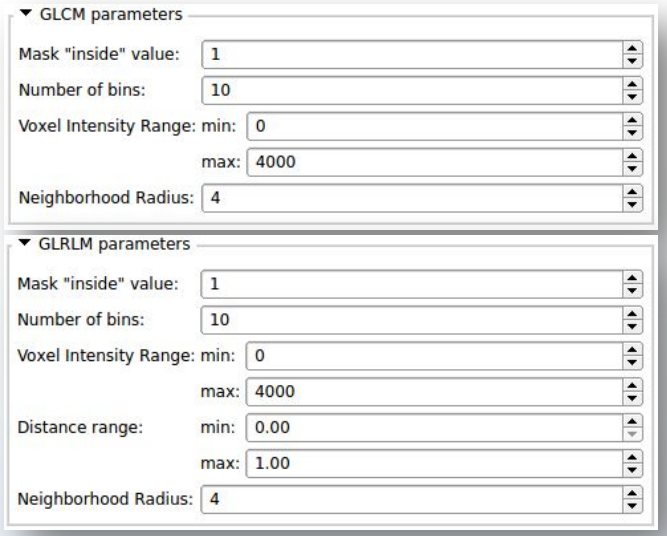
- The type of features wanted: Run length features and/or co-occurrence features
- All the parameters necessary to run each algorithms
- The type of outputs wanted: texture features and/or texture maps

Bone Texture Serializer



- Check the type of feature desired
- Specify the input parameters for each algorithm
- Click on the button to obtain the correspondent features

Bone Texture Serializer

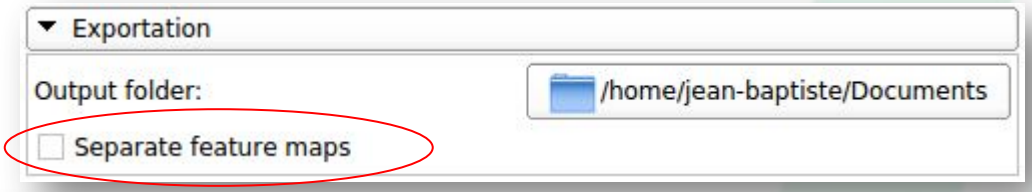


There is more description about the influence of each parameter later in the presentation

The GLCM and GLRLM sub-tabs allow to choose for each type of algorithm: (more description about the influence of each parameter later in the presentation)

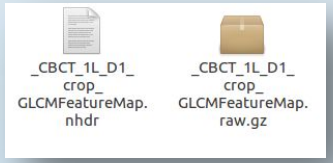
- *Inside Mask Value:* The pixel value that defines the ”inside” of the mask
- *Number of Intensity bins:* The number of intensity bins in the Grey Level Matrices
- *Pixel Intensity Min:* Minimum of the pixel intensity range over which the features will be calculated
- *Pixel Intensity Max:* Maximum of the pixel intensity range over which the features will be calculated
- *Distance Min:* Minimum of the distance range (in mm) over which the features will be calculated
- *Distance Max:* Maximum of the distance range (in mm) over which the features will be calculated

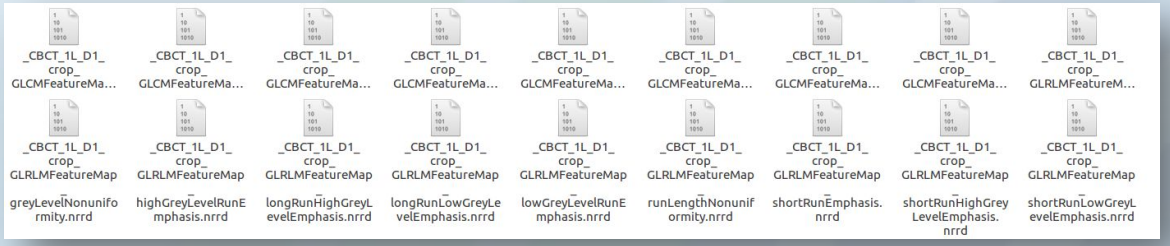
Bone Texture Serializer



The exportation section allows the user to specify the output folder where all the results will be saved:

- The texture features for the whole image will be saved in a csv file
- The texture maps can either be saved:
  - in a single file (a diffusion weighted image containing all the feature maps of a case)
  - in separate files (each feature map will be saved as a different volume)





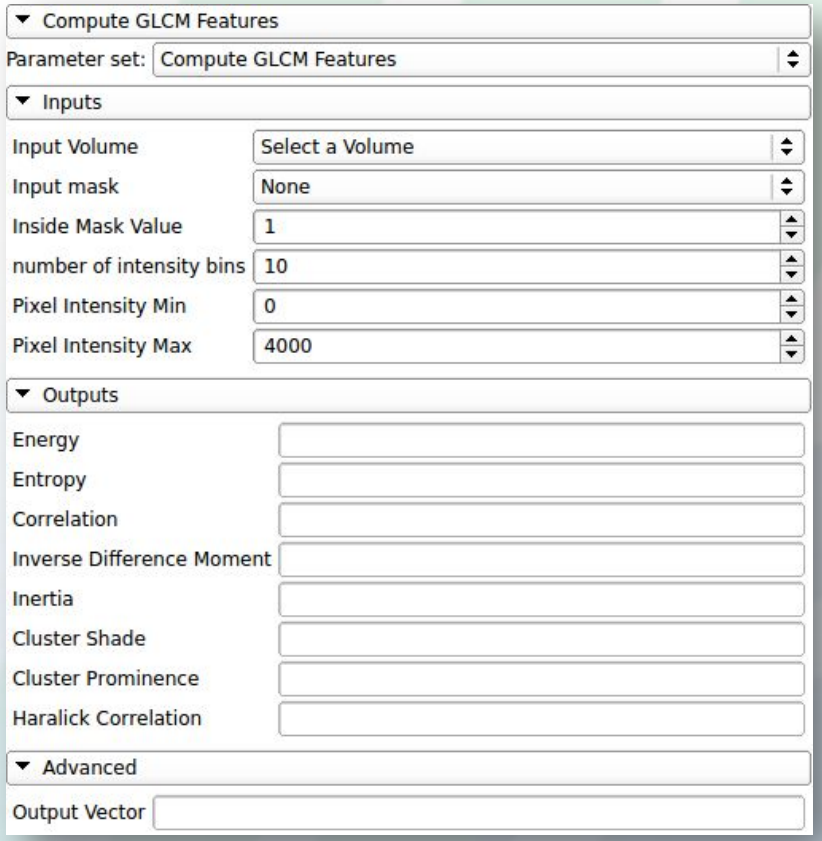
Kitware

The computational modules
(CLI modules)

- Compute GLCM Features
- Compute GLCM Feature Maps
- Compute GLRLM Features
- Compute GLRLM Feature Maps
- Separate Vector Image

Compute GLCM Features

This module will compute the textural features over the whole scan using the Grey Level Co-occurrence Matrix:



Inputs:

- Input volume [index: 0] : Input Volume
- Input mask [-s --inputMask] (default: None) : A mask defining the region over which texture features will be calculated
- Inside Mask Value [-i --inputMask] (default: 1) : The pixel value that defines the ”inside” of the mask
- Number of Intensity bins [-b --binNumber] (default: 10) : The number of intensity bins
- Pixel Intensity Min [-p --pixelIntensityMin] (default: 0) : Minimum of the pixel intensity range over which the features will be calculated
- Pixel Intensity Max [-p --pixelIntensityMax] (default: 4000) : Maximum of the pixel intensity range over which the features will be calculated

Compute GLCM Features

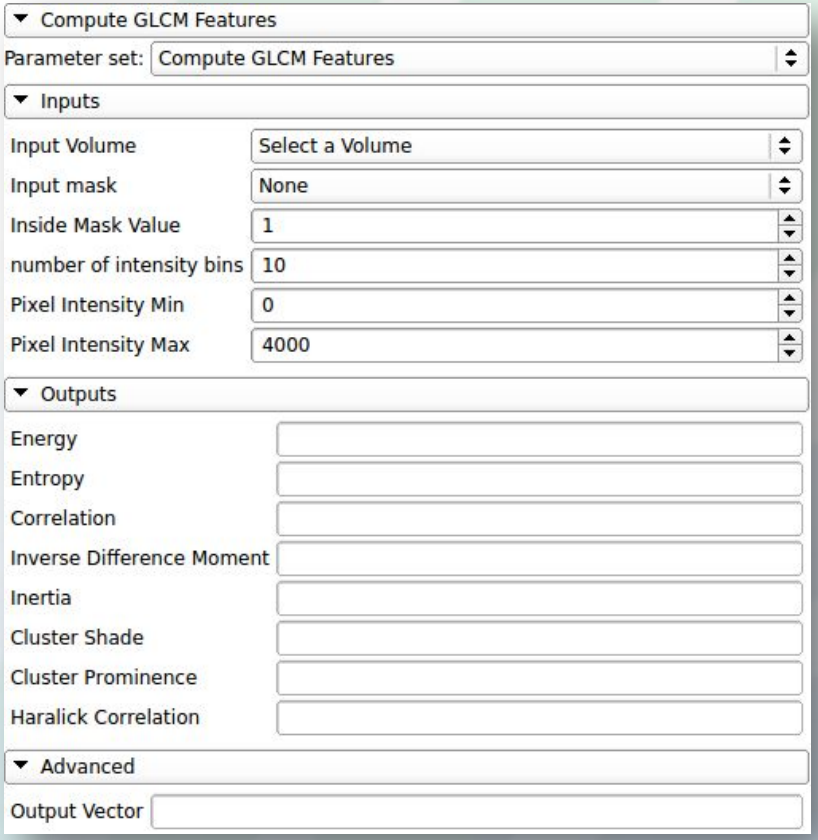
This module will compute the textural features over the whole scan using the Grey Level Co-occurrence Matrix:

Outputs:

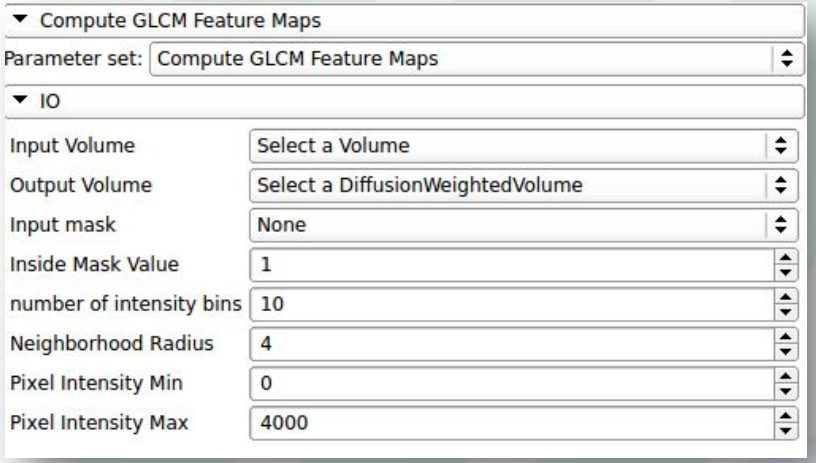
- Energy [output]
- Entropy [output]
- Correlation [output]
- Inverse difference moment [output]
- Inertia [output]
- Cluster shade [output]
- Cluster prominence [output]
- Haralick correlation [output]

Advanced:

- Output Vector [output] : Output vector containing all the feature value stored in the same order as above



Compute GLCM Feature Maps



This module will compute the textural features maps using the Grey Level Co-occurrence Matrix:

Inputs/Outputs:

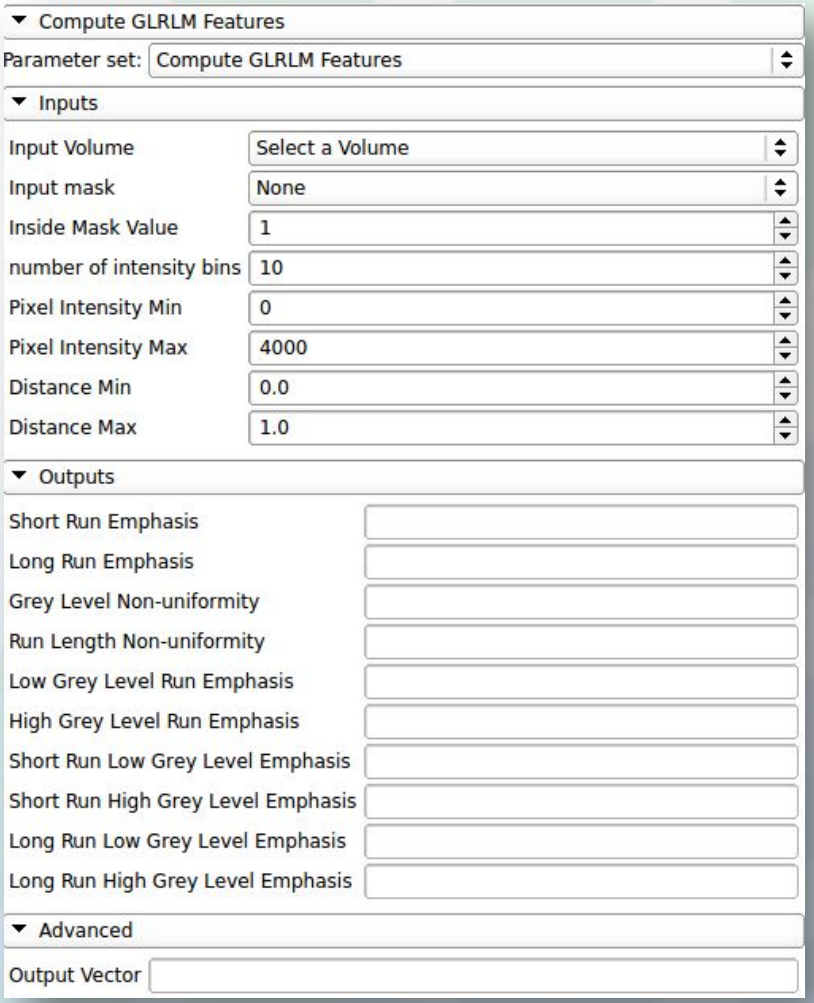
- Input volume [index: 0] : Input Volume
- Output volume [index: 1] : Output diffusion-weighted volume where the 8 feature maps will be stored
- Input mask [-s --inputMask] (default: None) : A mask defining the region over which texture features will be calculated
- Neighborhood radius [-n --neighborhoodRadius] (default: 4) : The size of the neighborhood radius
- Inside Mask Value [-i --inputMask] (default: 1) : The pixel value that defines the ”inside” of the mask
- Number of Intensity bins [-b --binNumber] (default: 10) : The number of intensity bins
- Pixel Intensity Min [-p --pixelIntensityMin] (default: 0) : Minimum of the pixel intensity range over which the features will be calculated
- Pixel Intensity Max [-P --pixelIntensityMax] (default: 4000) : Maximum of the pixel intensity range over which the features will be calculated

Compute GLRLM Features

This module will compute the textural features over the whole scan using the Grey Level Run Length Matrix:

Inputs(1/2):

- Input volume [index: 0] : Input Volume
- Input mask [-s --inputMask] (default: None) : A mask defining the region over which texture features will be calculated
- Inside Mask Value [-i --inputMask] (default: 1) : The pixel value that defines the ”inside” of the mask
- Number of Intensity bins [-b --binNumber] (default: 10) : The number of intensity bins

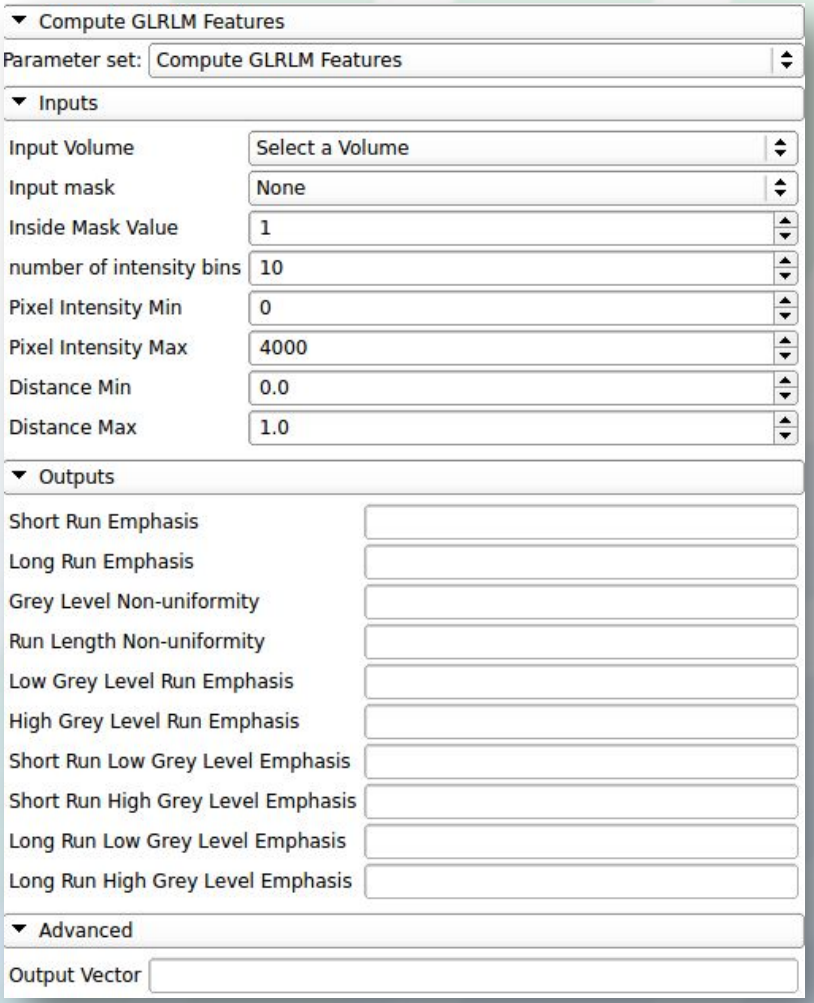


Compute GLRLM Features

This module will compute the textural features over the whole scan using the Grey Level Run Length Matrix:

Inputs(2/2):

- Pixel Intensity Min [-p --pixelIntensityMin] (default: 0) : Minnimum of the pixel intensity range over which the features will be calculated
- Pixel Intensity Min [-p --pixelIntensityMin] (default: 0) : Minimum of the pixel intensity range over which the features will be calculated
- Distance Min [-d --distanceMin] (default: 0.0) : Minimum of the distance range (in mm) over which the features will be calculated
- Distance Max [-D --distanceMax] (1.0) : Maximum of the distance range (in mm) over which the features will be calculated



Compute GLRLM Features

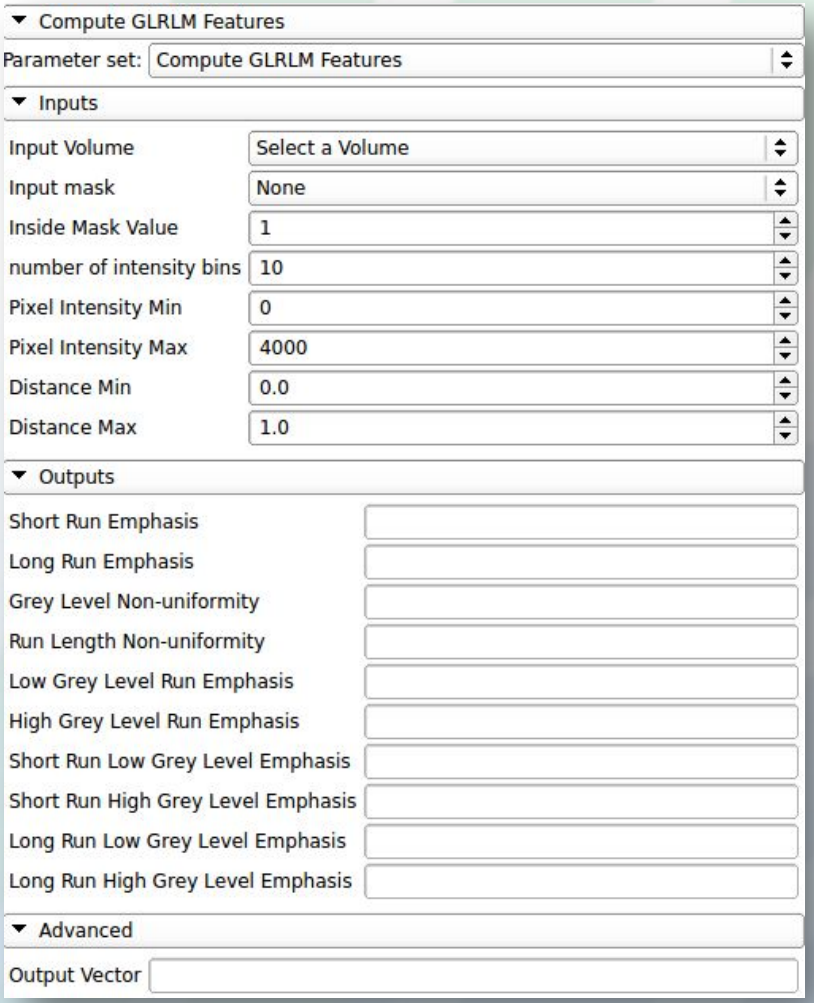
This module will compute the textural features over the whole scan using the Grey Level Run Length Matrix:

Outputs:

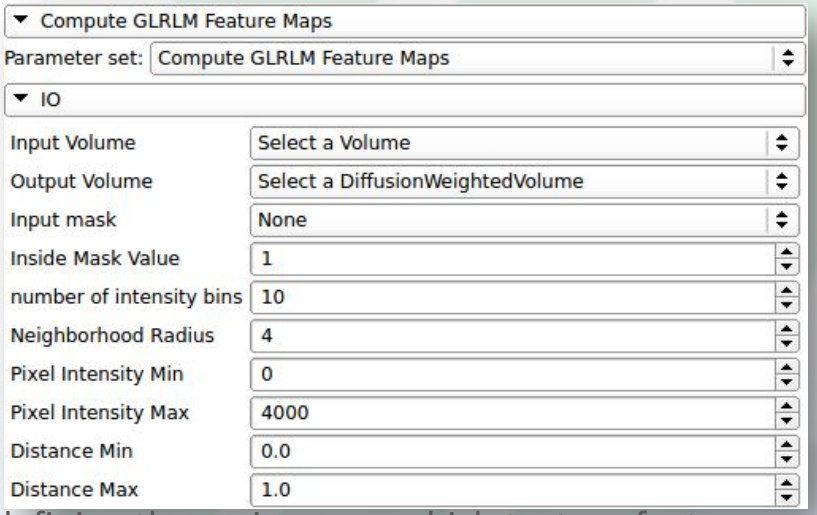
- Short Run Emphasis [output] :
- Long Run Emphasis [output] :
- Grey Level Non-uniformity [output] :
- Run Length Non-uniformity [output] :
- Low Grey Level Run Emphasis [output] :
- High Grey Level Run Emphasis [output] :
- Short Run Low Grey Level Emphasis [output] :
- Short Run High Grey Level Emphasis [output] :
- Long Run Low Grey Level Emphasis [output] :
- Long Run High Grey Level Emphasis [output] :

Advanced:

- Output Vector [output] : Output vector containing all the feature value stored in the same order as above



Compute GLRLM Feature Maps



This module will compute the textural features maps using the Grey Level Run Length Matrix:

Inputs/outputs:

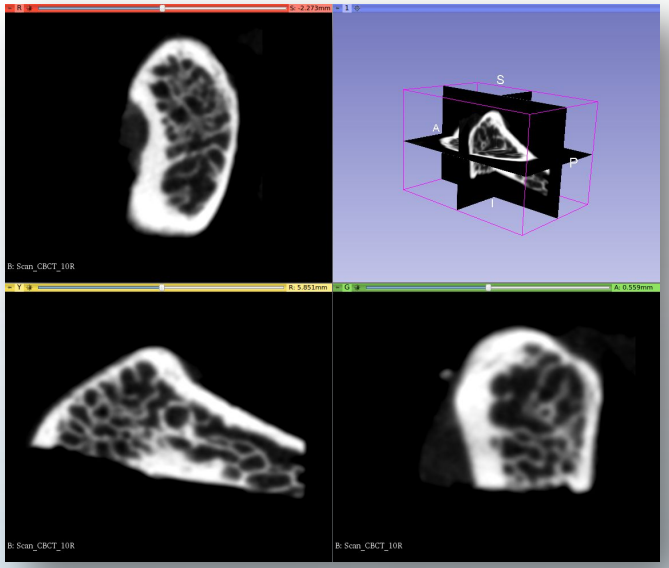
- Input volume [index: 0] : Input Volume
- Output volume [index: 1] : Output diffusion-weighted volume where the 8 feature maps will be stored
- Input mask [-s --inputMask] (default: None) : A mask defining the region over which texture features will be calculated
- Inside Mask Value [-i --inputMask] (default: 1) : The pixel value that defines the ”inside” of the mask
- Number of Intensity bins [-b --binNumber] (default: 10) : The number of intensity bins
- Neighborhood radius [-n --neighborhoodRadius] (default: 4) : The size of the neighborhood radius
- Pixel Intensity Min [-p --pixelIntensityMin] (default: 0) : Minimum of the pixel intensity range over which the features will be calculated
- Pixel Intensity Max [-P --pixelIntensityMax] (default: 4000) : Maximum of the pixel intensity range over which the features will be calculated
- Distance Min [-d --distanceMin] (default: 0.0) : Minimum of the distance range (in mm) over which the features will be calculated
- Distance Max [-D --distanceMax] (default: 1.0) : Maximum of the distance range (in mm) over which the features will be calculated

Parameters

- Experimental conditions
- Default parameters
- Mask
- Number of bins
- Voxel intensity Range
- Distance Range
- Neighborhood radius

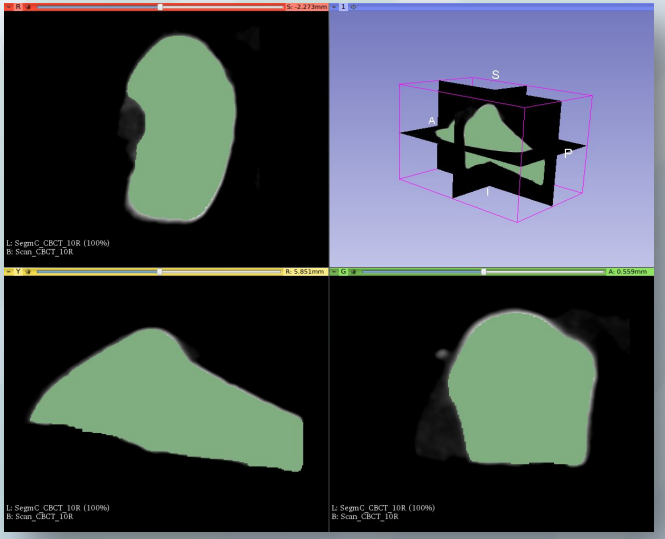
Experimental Conditions

In order to observe the influence of each parameter on the output textural maps and the computation time, BoneTexture module was run on the same input with different parameter sets.



Our study input case is a Cone Beam Computed Tomography (CBCT) of a condyle. The input volume size is 197x287x193 (10.912.024 voxels) with an isotropic spacing of 0.08mm and an intensity range between -777 and 4276.

We also used a segmentation of the condyle (mask), this segmentation is composed of 1.622.440 voxels (15% of the scan).



Experimental Conditions

The machine used for the experiments had the following configuration:

The version of 3DSlicer used was the nightly build of July 4th 2017 (built in release mode).

Default parameters

The default parameter set that is used as a reference is:

- *Inside Mask Value:* 1
- *Number of Intensity bins:* 10
- *Pixel Intensity Min:* 0
- *Pixel Intensity Max:* 4500
- *Distance Min:* 0.0
- *Distance Max:* 1.25
- *Neighborhood:* 4

As a reference, with the default parameters, the GLCM algorithm takes about 1m 11s to run, and the GLRLM took 2m 49s.

Warning: The discussions that will be made in this section might be specific to the study case and the type of features that were computed.

They are mainly here to illustrate the importance of choosing the parameters correctly, and try to give a better understanding of the influence of each one of those parameters.

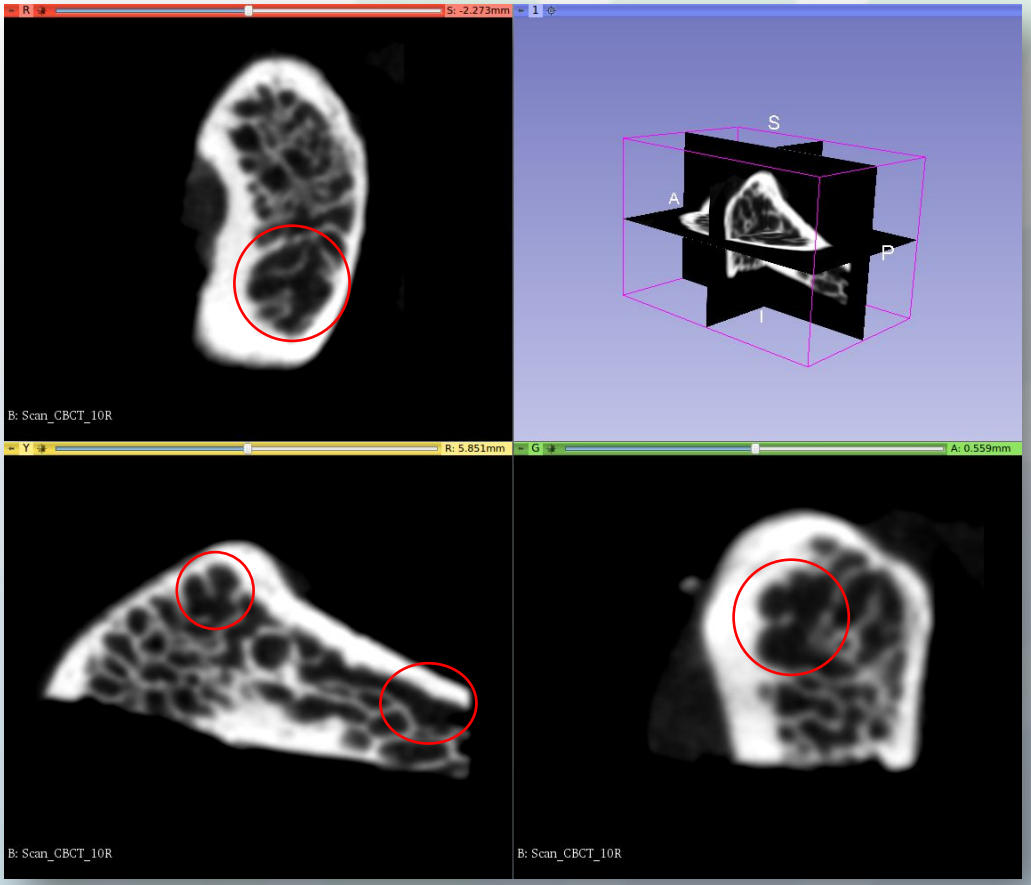
The best parameter set specific to your study case might be different.

Default parameters

Detection of early stage of Osteoarthritis (OA) in Temporomandibular Joints (TMJ):

For our particular case study, we are trying to detect early stage of TMJOA, which is characterized by a lack of trabecula in the bone texture (i.e. holes in the bone texture).

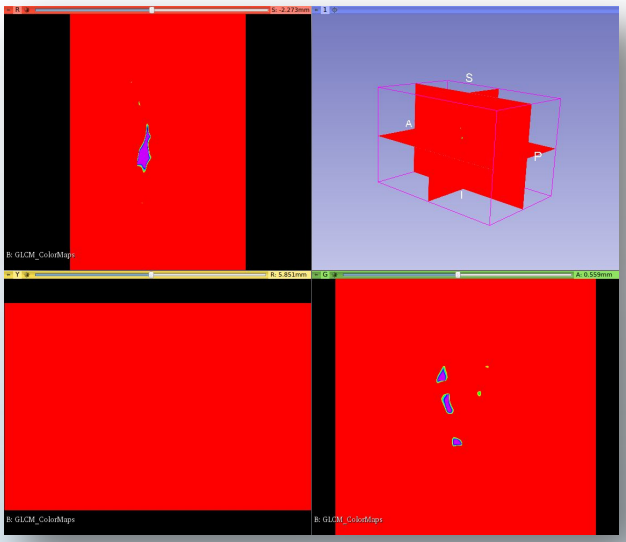
In our input data we can identify four different places where the bone seems to be diseased.



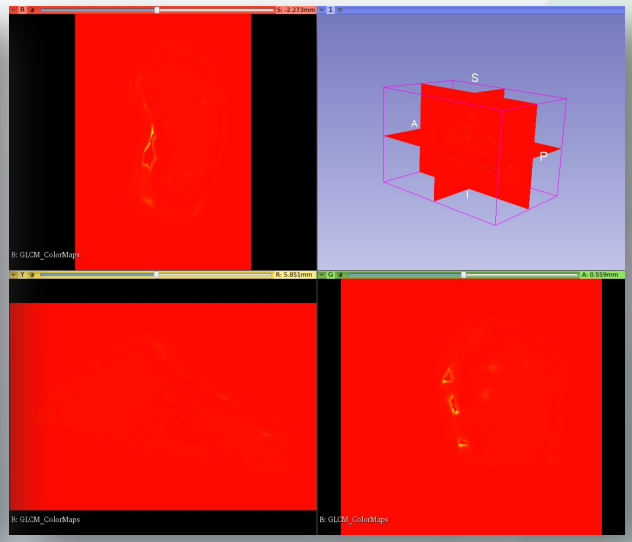
Default parameters

After running both algorithms (GLCM and GLRLM) with the default parameter set:

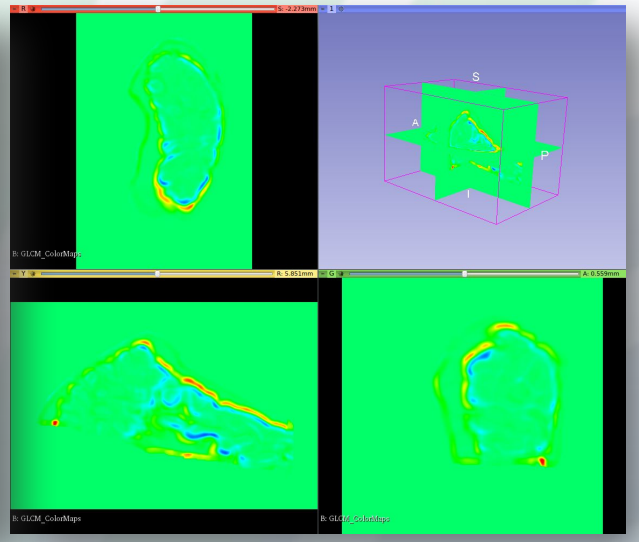
Some texture maps does not give any information



inverse difference moment



correlation



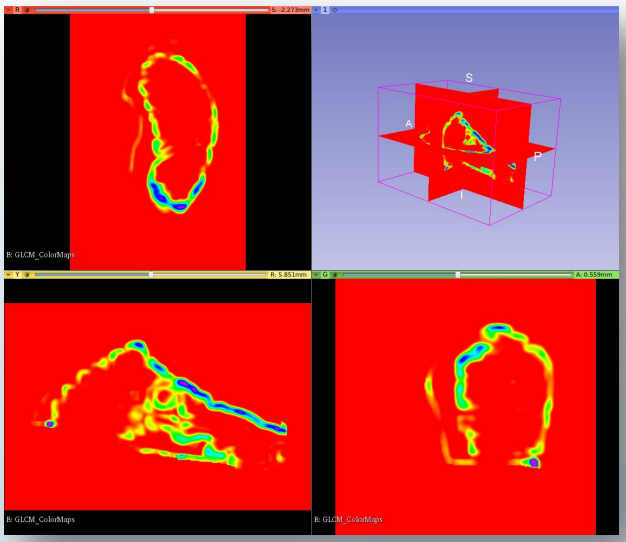
cluster shade

Kitware

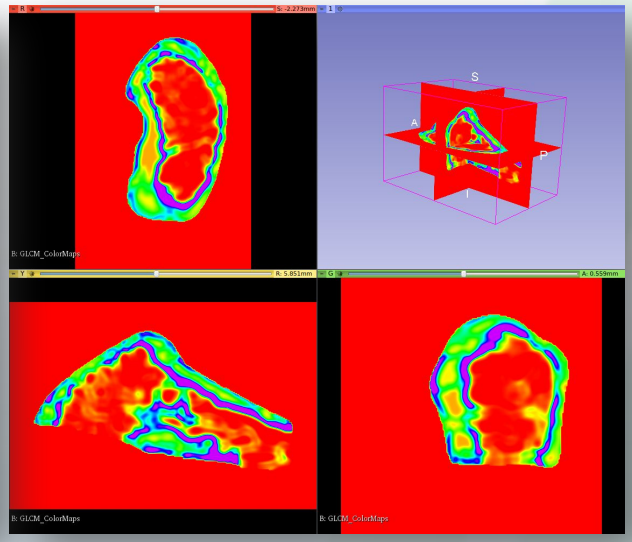
Default parameters

After running both algorithms (GLCM and GLRLM) with the default parameter set:

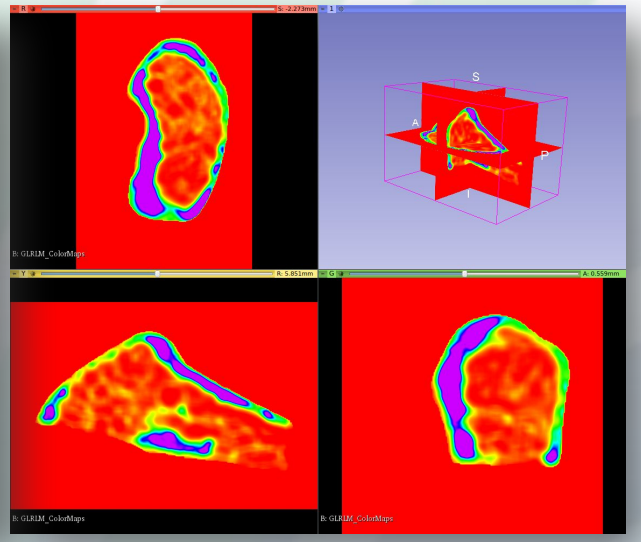
Some texture maps give information, but they are not useful in our case



cluster prominence



Haralick correlation



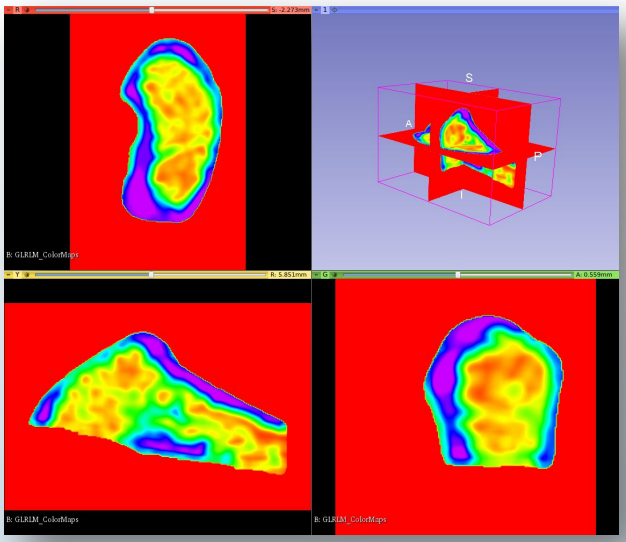
Long run high gray level emphasis

Kitware

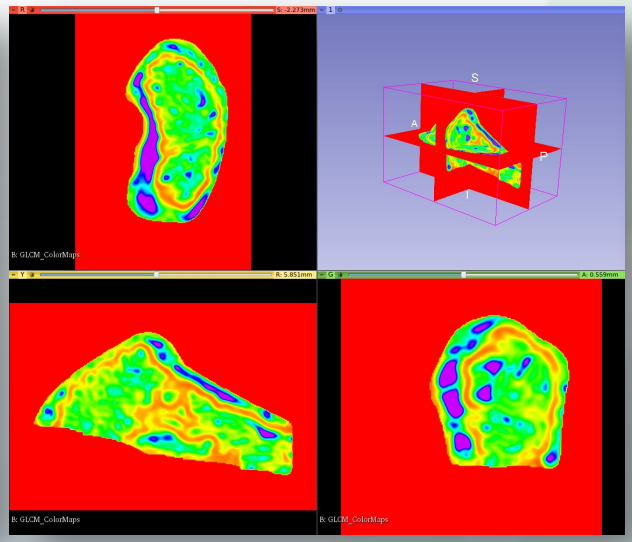
Default parameters

After running both algorithms (GLCM and GLRLM) with the default parameter set:

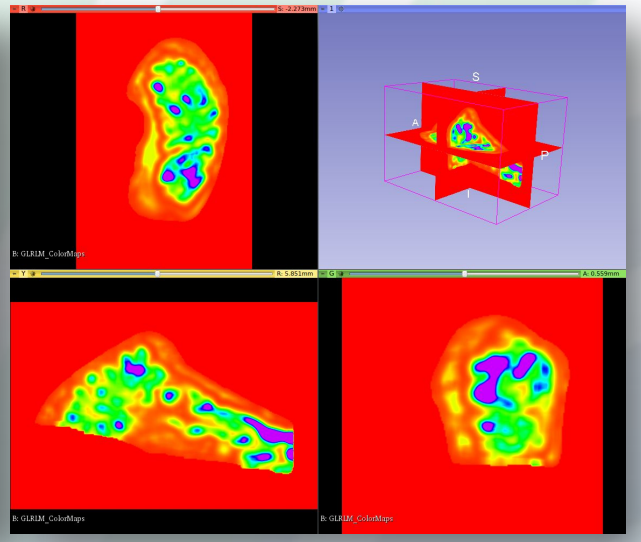
And some give us information we could exploit



high gray level run emphasis



energy



long run low gray level emphasis

Mask

For this experiment we used the default parameter set, but *WITHOUT* any mask:

The computation time is significantly higher: 3m 18s for the GLCM algorithm, 6m 17s for the GLRLM algorithm

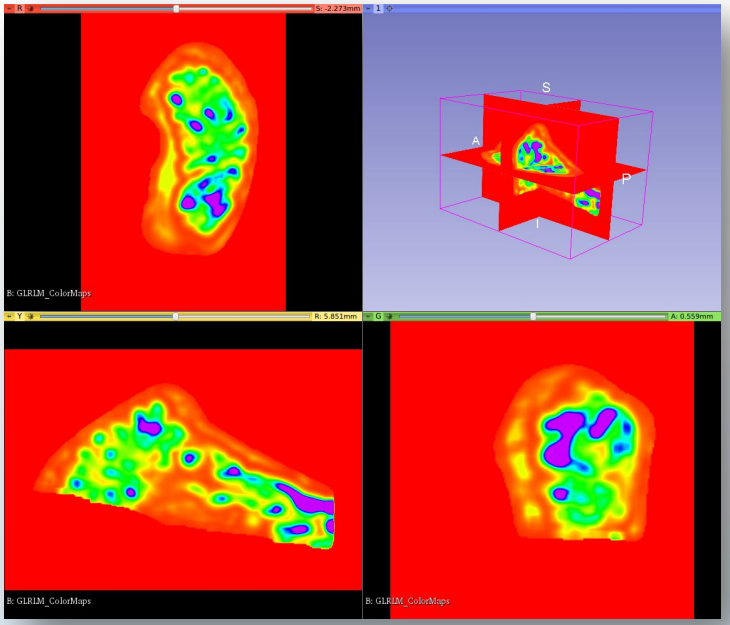
For the voxels that were ''inside the mask'' (voxels of interest) the output values were exactly the same as when the mask was given.

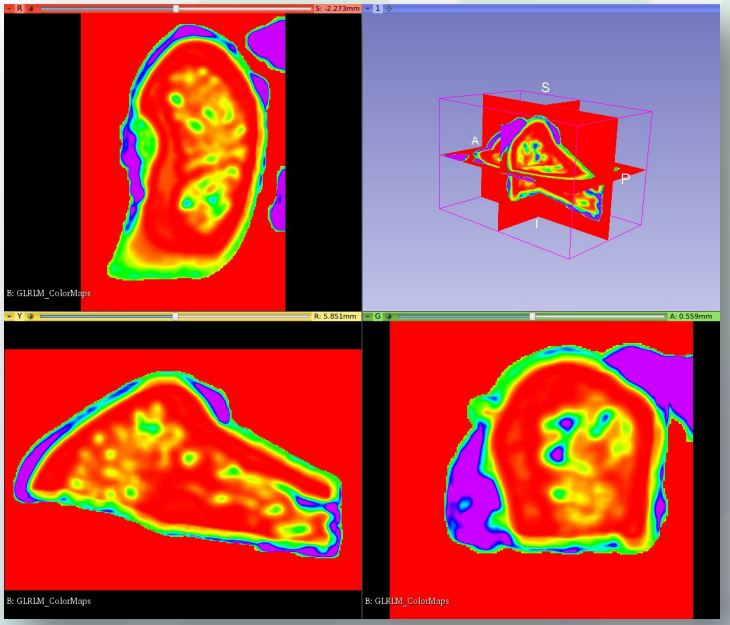
For the voxels that were on the boundaries of the mask the values are slightly different

For the voxels that were outside of the mask, new values were computed

Mask

If we compare the Long Run Low Grey Level Emphasis feature maps with (left) and without (right) a mask:





It looks like the feature maps computed without a mask is not giving the same information even for the voxels that were inside the mask. But, they contain the same information: not using the mask on the right creates a different feature value range, which changes the color mapping.

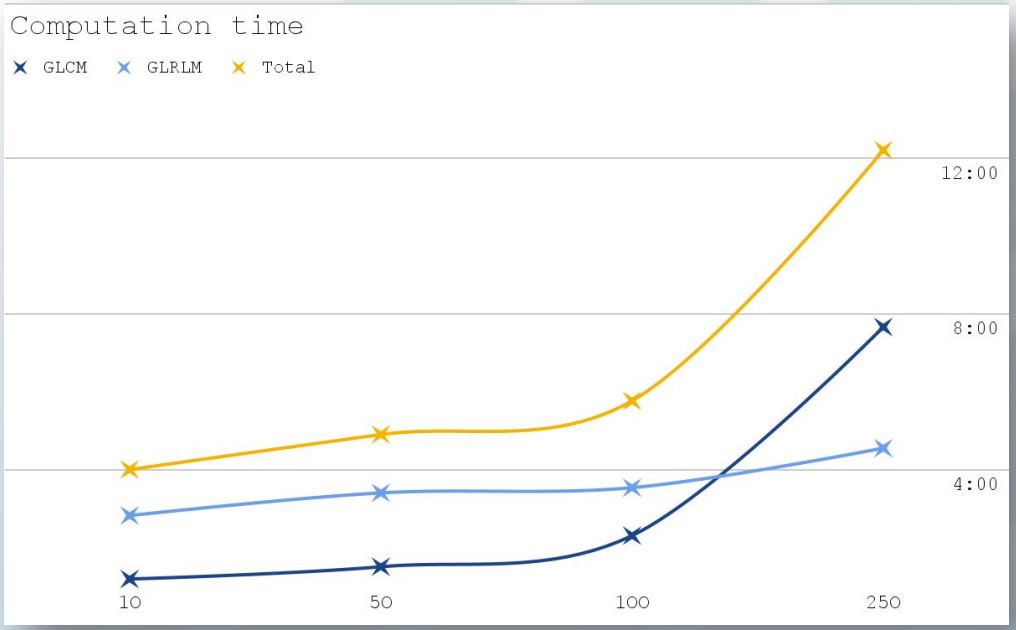
Number of bins

In addition to the default value of 10 bins, we also used 50, 100 and 250 bins:

The computation time increases with the number of bins as illustrated below:

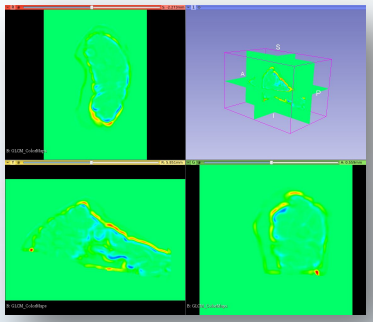
	10 bins	50 bins	100 bins	250 bins
GLCM	1:11	1:30	2:18	7:40
GLRLM	2:49	3:24	3:32	4:33
Total	4:00	4:54	5:46	12:13

The computation time unit is min:sec

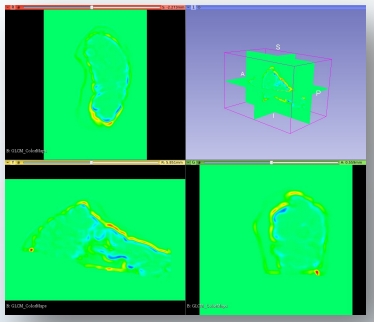


Number of bins

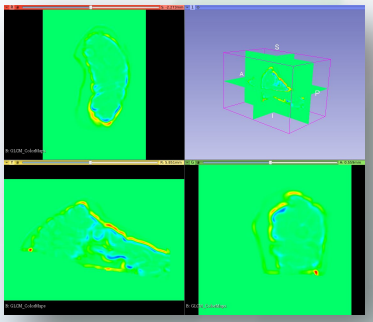
Some texture feature maps don't seem to be affected by the number of bins used: like cluster prominence, cluster shade (se fig below), correlation



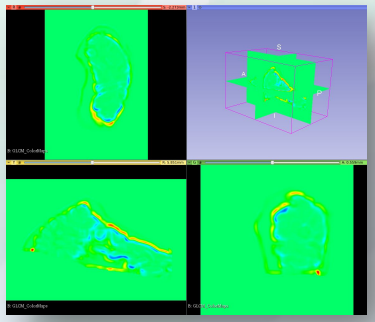
10 bins



50 bins



100 bins

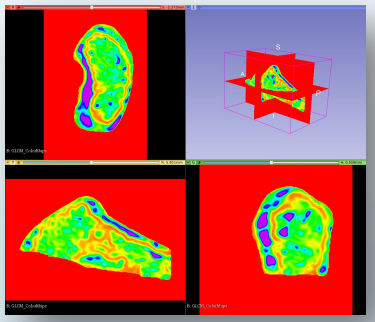


250 bins

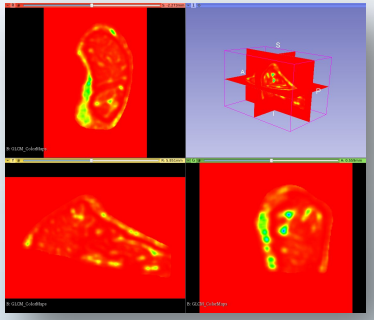
Number of bins

Some other texture feature maps give us different information when the number of bins increases:

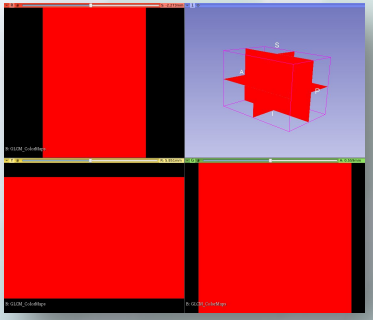
The energy for example seems to give us no information for 50 or 100 bins and a totally different one for 250 bins



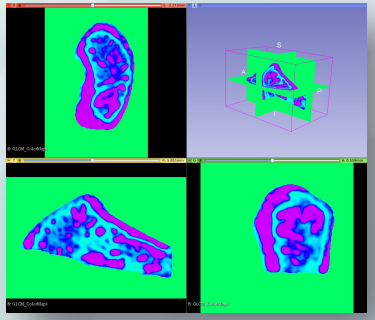
10 bins



50 bins



100 bins

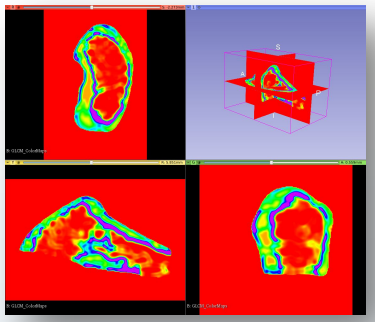


250 bins

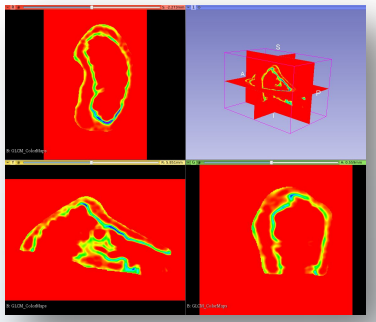
Kitware

Number of bins

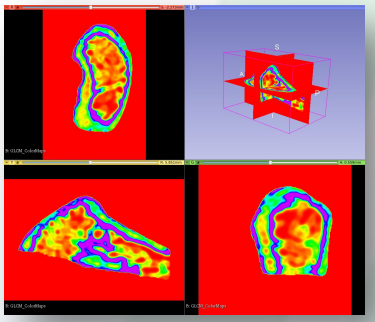
Same with The Harralick Correlation, it gives us more information for 100 and 250 bins



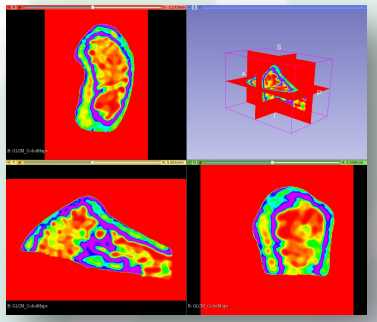
10 bins



50 bins

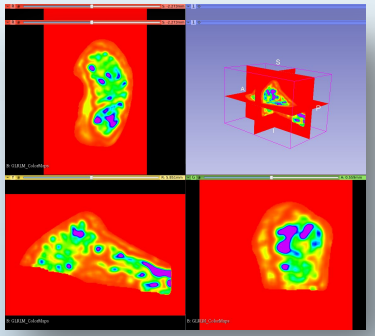


100 bins

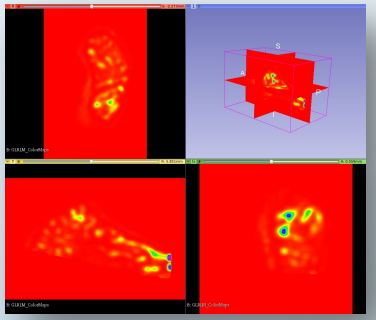


250 bins

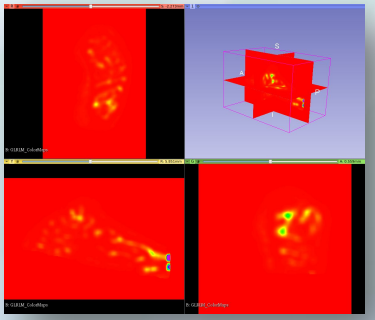
On the other side, some feature maps like the Long Run Low Grey Level Emphasis give us less information when the number of bins increases



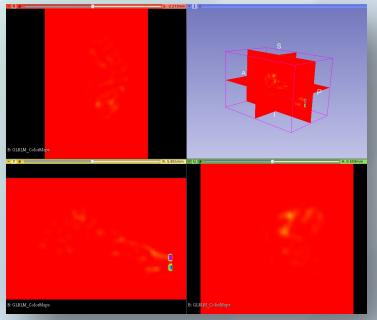
10 bins



50 bins



100 bins



250 bins

Voxel Intensity Range

In addition to the default 0-4500 intensity range (for the use case) we also used 0-2000, 0-8000 and 1500-4500 value ranges:

The computation time doesn't seem to be related to the voxel value range size in a linear way. The computation time will mainly depend on how many voxel are contained in this range and how they are distributed in the different bins.

	0 - 2000	1500 - 4500	0 - 4500	0 - 8000
GLCM	0:42	0:49	1:11	1:09
GLRLM	1:37	2:00	2:49	2:28
Total	2:19	2:49	4:00	3:38

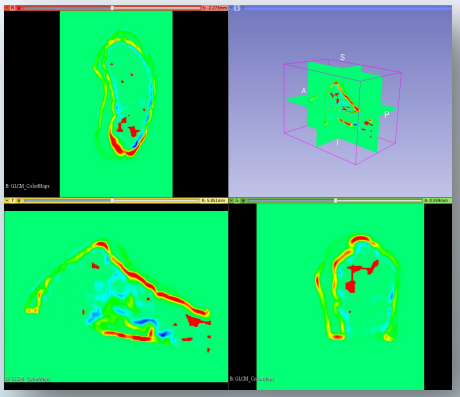
The computation time unit is min:sec

Voxel Intensity Range

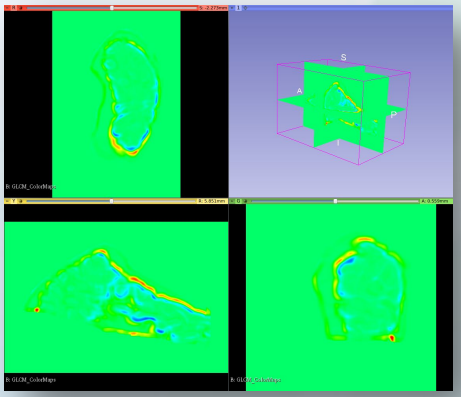
Choosing correctly the value range is very important as all the voxels out of this range will not be taken into account.

In our study case the input image has a value range of -777 to 4276, but we chose a range of 0-4500 to represent the intensity range of the foreground voxels as marked by the mask image. The background in mask image is ignored during computation.

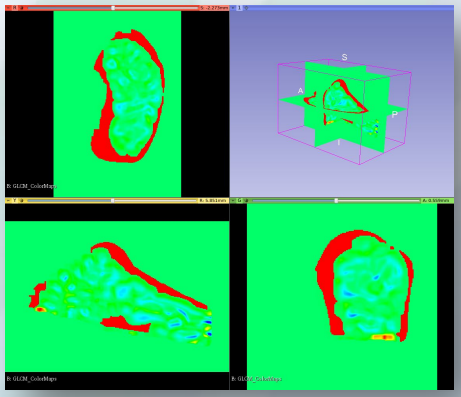
Choosing range like 0 - 2000, or 1500 - 4500 will exclude a part of the useful data, and will lead to ''holes'' in the textures maps. They are particularly visible in the Cluster Shade feature maps (red in the figure below).



1500-4500



0-4500

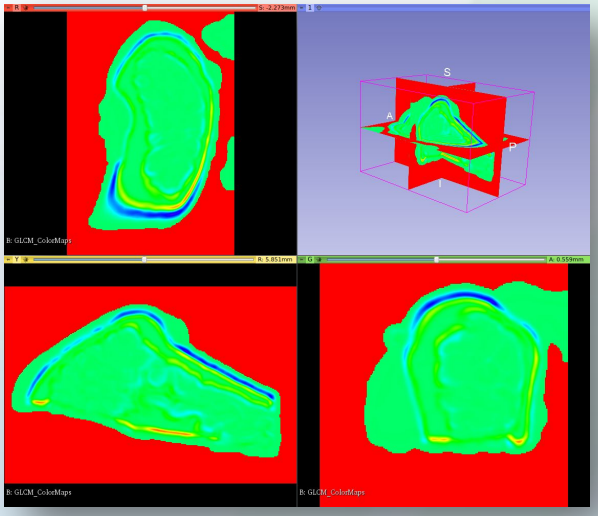


0-2000

Cluster Shade

Voxel Intensity Range

However, excluding some voxels can be useful, particularly if they represent noise in the image. That's actually what happened in the computation without mask, the noisy part of the image hasn't been computed (red part) which improved the computation time.

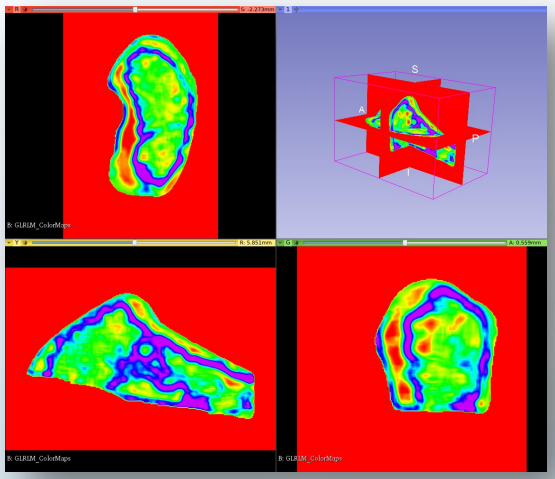


0-4500 without mask
Cluster Shade

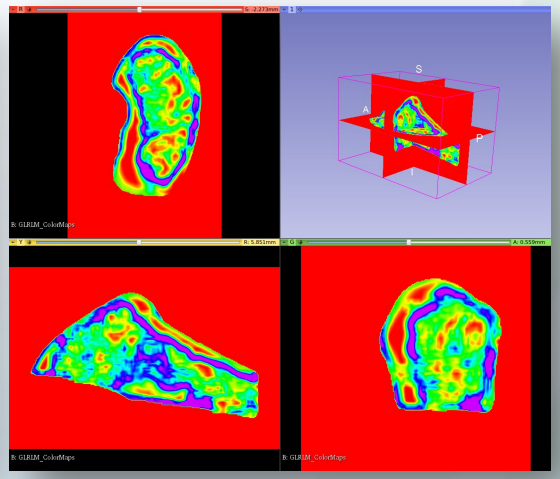
Voxel Intensity Range

Increasing the voxel intensity range, even without noise in the image, will modify the output texture maps: in fact it will modify the repartition of the bins and the construction of the matrices.

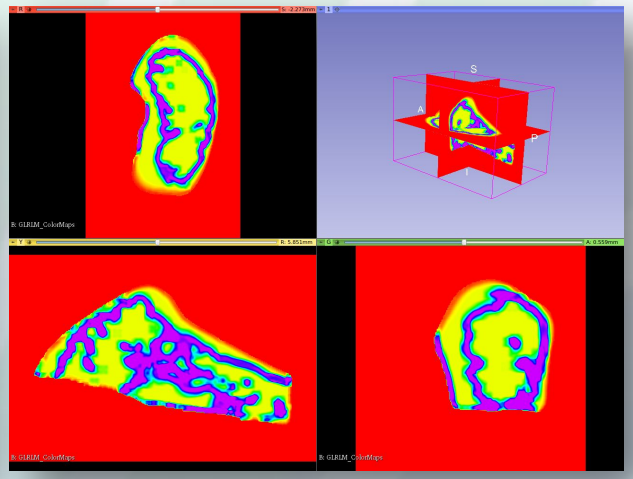
In our case, the modification is not that significant with the 0 - 8000 range, but the feature maps are not exploitable anymore with a 0 - 20000 range



0-4500



0-8000



0-20000

Run Length Non Uniformity

Kitware

Distance Range

In addition to the default 0.0-1.25 mm distance range we also used 0.0-0.5 mm, 0.0-2.0 and 0.5-1.25 mm ranges:

The computation time seems to be totally independent from the distance range

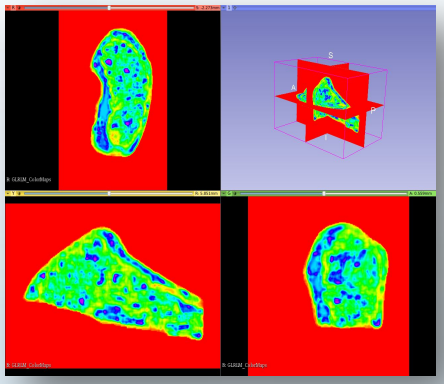
	0.5 - 1.25 mm	0.0 - 0.5 mm	0.0 - 1.25 mm	0.0 - 2.0 mm
GLRLM	2:38	2:35	2:49	2:36

The computation time unit is min:sec

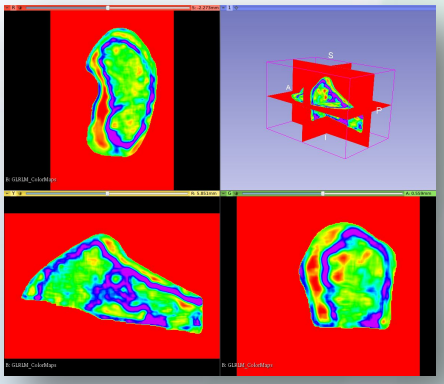
Distance Range

In our case study we chose to take into account the run of all lengths, the choice of 1.25 mm is then motivated by the fact that we have a neighborhood radius of 4 voxels (a cube of 9x9x9) and a spacing of 0.08mm in each direction: the longest run possible will be $\sqrt{(3\times(0.08\times9)^3)}\approx1.05$mm

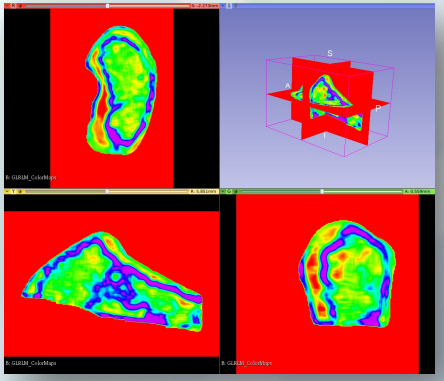
Therefore, excluding runs that are too short or too long might be a good idea depending on what you want to find in your image.



0.25-1.25 mm



0-1.25 mm



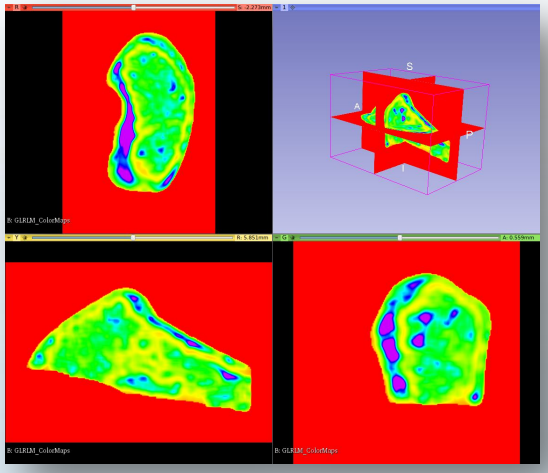
0-0.5 mm

Run Length Non Uniformity

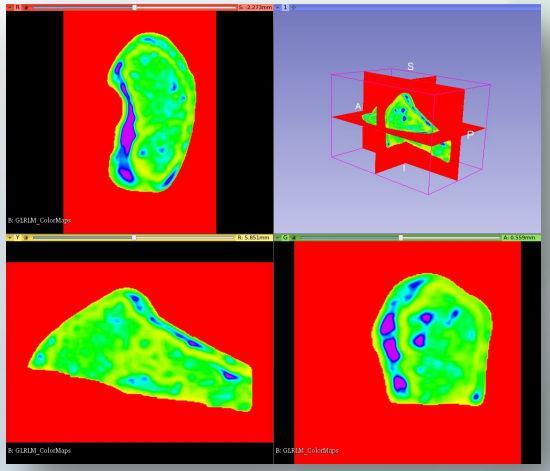
Distance Range

Similarly to the Voxel Intensity range, increasing the distance range, will modify the output texture maps: in fact it will modify the repartition of the bins and the construction of the matrices.

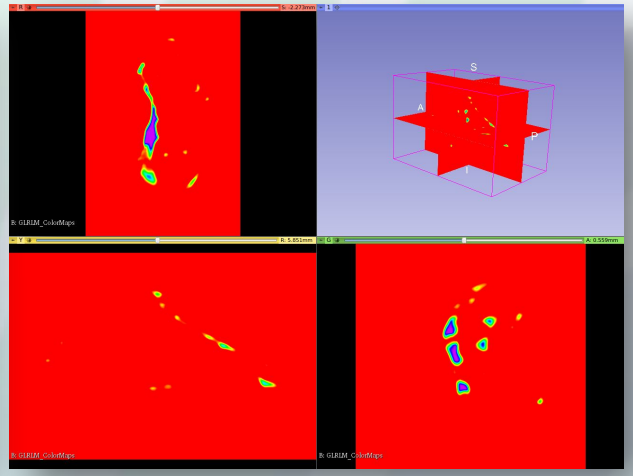
In our case, the modification is not that significant with the 0.0-2.0 mm range, but the features map are not exploitable anymore with a 0.0-5.0 mm range



0-1.25 mm



0-2.0 mm



0-5.0 mm

Long Run Emphasis

Kitware

Neighborhood Radius

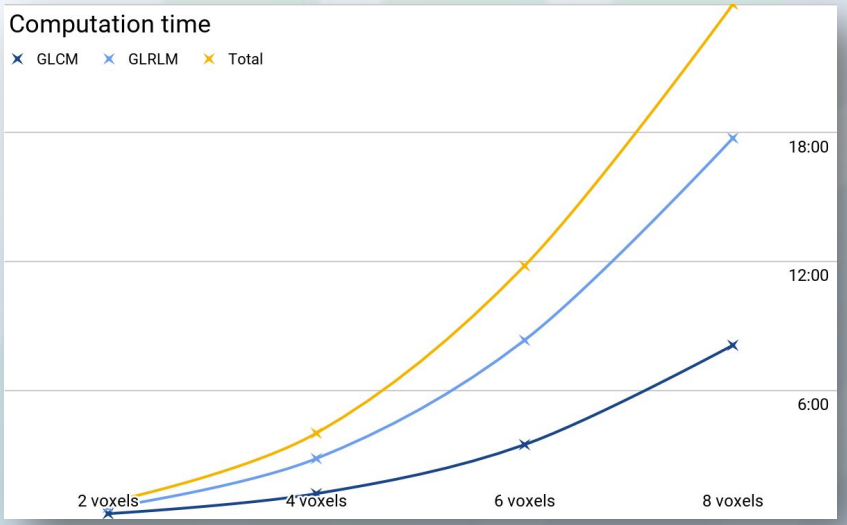
In addition to the default 4 voxel neighborhood radius we also used 2, 6 and 8 voxel radii:

In our case the distance range has been adapted to the size of the neighborhood in order to take all the runs into account.

The computation time increase with the neighborhood radius (as we have seen before that the distance range does not affect the computation time)

	2 voxels	4 voxels	6 voxels	8 voxels
GLCM	0:15	1:11	3:28	8:06
GLRLM	0:32	2:49	8:20	17:45
Total	0:47	4:00	11:48	25:53

The computation time unit is min:sec

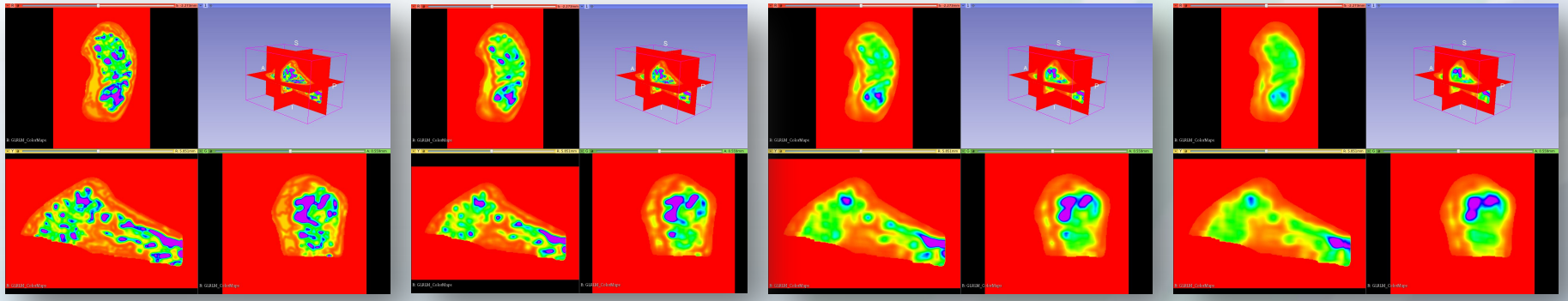


Kitware

Neighborhood Radius

The size of the neighborhood will have a lot of impact on the output texture maps

This parameter should be defined depending on the user's goal: the type of features that need to be observed and particularly their size.



2 4 6 8

Long Run High Grey Level Emphasis

Resources - Contact

Github repositories:

- 3DSlicer
- itkTextureFeatures
- BonTextureExtension

Slicer Wiki:

- BonTextureExtension

Slicer Forum:

- Discourse

Insight Journal Article:

- itkTextureFeatures Insight Journal Article

Data:

- Input/Output data used in this presentation

For other remarks or questions please email:

- jb.vimort@kitware.com

Kitware

Acknowledgement

This work was supported by the National Institute of Health (NIH) National Institute for Dental and Craniofacial Research (NIDCR) grant R21DE025306 (Textural Biomarkers of Arthritis for the Subchondral Bone in the Temporomandibular Joint) and NIDCR grant R01DE024450 (Quantification of 3D bony Changes in Temporomandibular Joint Osteoarthritis)

We would like to thank Dr. Larry Wolford from the Baylor University Medical Center for kindly providing the bone specimens from which we obtained the scans used in the paper. We would like to thank Drs. Lucia Cevidanes, Erika Benavides and Antonio Ruellas at the University of Michigan School of Dentistry as well, for generating the CBCT scans that were processed with the filters presented in the paper.

We are also grateful for the support received from the ITK community.

Kitware