Porosity Analysis

Segmentation, Connected Components, and Visualization

Vedrana Andersen Dahl

April 10, 2019

Overview

The plan for the porosity analysis is

1. Volumetric segmentation and related measures
2. Connected component analysis and related measures
3. Local thicknes and related measures - covered in another note

1 Volumetric segmentation

The first step of porosity analysis is segmentation. In the simplest (and most common) case of having only two labels, material and void, segmentation corresponds to binarization of the volume. All subsequent porosity analysis operates a binarized data, so this step has a strong influence on the result.

Functions for some commonly performed tasks have been implemented for the course. Those include `inspect_tifvol` for visual inspection without loading the volume, `load_tifvol` for loading the volume, `inspect_matvol` for visually inspecting a loaded volume, `inspect_volume_binarization` for visually inspecting binarization on top of volume data.

Inspecting the data is a first step of any data analysis, and should be done before actual segmentation.

- Load the volumetric data and a few slices. Useful MATLAB functions: `imread` for reading tif files, `fopen`, `fread` and `fclose` for reading raw files, `imagesc` for inspection, combined with `colormap` and `axis`. Try also the provided functions for inspecting and loading the data.
- Note basic available information about the volume: the dimensions, file format (usualy tif), numerical type (usually uint8, uint16 or double), intensity range. Hint: take a look at the workspace. Useful functions: `size`, `min`, `max`.
- Optionally crop the image, such that you can conduct initial experiments faster, on a smaller part of the image. Crop by indexing.
- Optionally resize the image. Useful function `imresize3`.
- Make images of three orthogonal cross sections trough the center of the volume. Hint: use indexing, alternatively use `slice`.
- Produce a histogram of the data. Useful functions: `hist`, `histogram`.
- Save the screenshots of MATLAB figures using function `saveas`. Save images using `imwrite`. Note that `imwrite` changes behaviour depending on the numberical type.

Visualizing data in 3D often produces comprehensive view of the imaged matter. While 3D visualization is possible in MATLAB specialized software that allows better interactivity is sometimes a faster way of producing visually pleasing images. Try producing a 3D visualization of the data by using ParaView, choose Volume representation and use color map editor to adjust transparency of the mapping function, see Fig. 1. Alternatively, use ImageJ plugin Volume Viewer.

ParaView crash-course for visualizing 3D gray-scale data.

- Open the data. For the data to load, you may need to click apply in the Properties (if not visible, check its visibility in View menu).
- In the Pipeline Browser (if not visible, check its visibility in View menu) visualize the data by clicking the small eye icon next to the filename.
- Choose Representation as Volume (default is Outline).
- In Color Map Editor (if not visible, check its visibility in View menu) adjust the Mapping Data by dragging the mapping curve which defines the transparency associated with each gray value.
- You may want to change the colormap which defines a color associated with each gray value. The selection of useful colormaps is available by clicking the folder+heart icon left to the Mapping Data, and even more colormaps is shown if you open additional options by clicking a small gear icon.

Thresholding is the simplest way of binarizing (segmenting) the data.

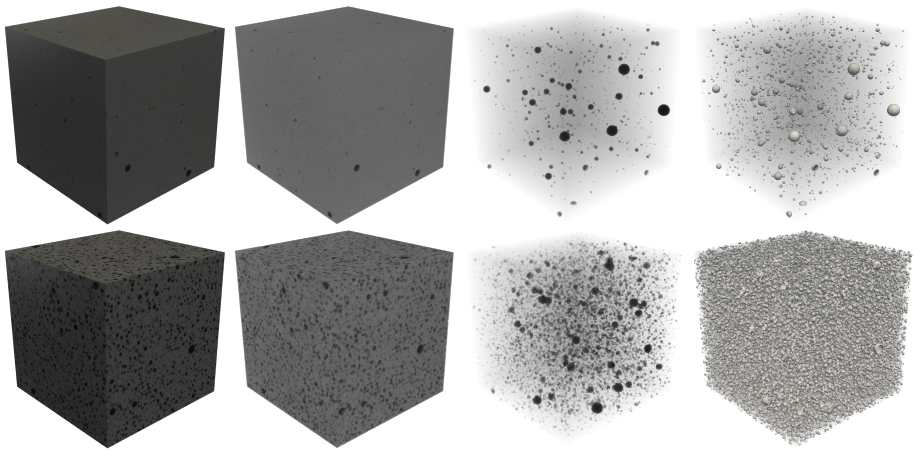

Figure 1: Some of 3D visualization options provided by ParaView: surface, solid volume, volume with transparency, contour.

- Pick a suitable threshold and create a binary volume by using relational expression *greater than*.
- Make images showing the binarization result on three orthogonal cross sections trough the center of the volume. Try also provided function for inspecting binarization.
- Compute mean intensity of the material phase, and the mean intensity of the void phase. Useful functions: `mean`. Remember also that `~` inverts the binary values.
- Compute volume fraction of the material. Useful functions: `sum`, `numel`.
- Compute the area of the interface between the material and the void.

Filtering is commonly used in connection with thresholding, to remove the noise from the data. The filtering is usually applied to the volume prior to thresholding, but some types of filtering can also be applied after thresholding.

- Try filtering the volume data prior to thresholding, use a Gaussian or a median filter. Useful functions: `imgaussfilt` for 3D Gaussian filtering or `imfilter` combined with `fspecial('gaussian',...)` which can be a more flexible approach, `medfilt3` for 3D median filtering.
- Try filtering the binary outcome of thresholding using a median filter, or a morphological filter. Useful functions: already used `medfilt3`, for morpholigical filter `imclose`, `imopen` in combination with `strel('sphere',...)`.
- Produce histograms showing distribution of intensities in the two classes.

Slightly more advanced thresholding approaches involve computing the threshold level, useful functions are `otsuthresh`, `graythresh`. Adaptive thresholding com-

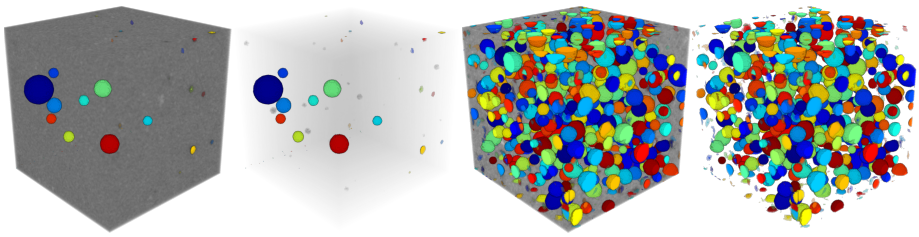

Figure 2: Visualizing connected components for two cement samples.

putes threshold level locally for each pixel, useful function is `adaptthresh`. A function `imbinarize` combines functionality for some common approaches to thresholding, including adaptive thresholding.

Quantitative measures from segmentation can be computed for each of segmentation regions. As for a binary case, the measures include: volume (and volume fraction), interface area, mean and standard deviation of intensities.

2 Connected component analysis

For connected component analysis, one of the segmentation phases is divided in connected components, which are then analyzed individually, yielding a distribution of measures. This will often reveal shortcomings in the obtained segmentation, for example when objects which should be disconnected appear connected. Consequently, re-visiting and improving on segmentation is often required after initial connected component analysis.

Connected components analysis in MATLAB is provided by the function `bwconncomp`.. The use of this built-in MATLAB function is complicated by the fact that it returns a structure which is not easy to interpret. This structure may be passed to other built-in MATLAB functions which will return a more informative outputs. It is a good idea to browse examples provided by help and documentation, to see the intended use of `bwconncomp`. One of the related functions is `labelmatrix` which creates a label volume from the structure returned by `bwconncomp`. A label volume is a volumetric image where a voxel intensity is an index to a connected component the voxel belongs to. The ordering of indices is given by the geometry, so it often makes sense to use random colors.

Watershed can be used to further separate a binary region, and obtain a more desirable outcome when using connected components analysis. The shape of the regions determines the outcome of watershed, and undesired fragmentation of may

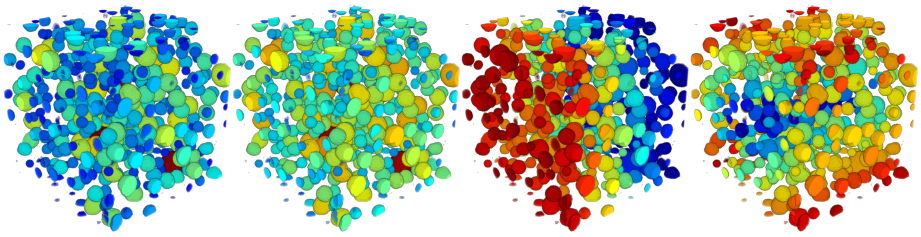

Figure 3: Visualizing quantitative measures obtained from connected component analysis. From left to right: volume, area, distance from a plane, distance from a line trough volume center.

occur. Therefor, method often requires fine-tuning and alterations. Useful MATLAB functions: `bwdist` and `watershed`.

Visualizing connected components in 3D can be very informative. Results in Fig. 2 are made using ParaView. Using external software for visualizing the outcome of an analysis done in MATLAB requires saving the results, for example by writing image files. As conventions on handling volumetric data vary, this may require adjustments. For example, x and y axes may need to be swapped, spacing between voxels may be inconsistent, and transparency channel may be inverted. To ease these issues, you are provided with a few functions for saving volumetric data: `save_grays` saves volume as collection of `.tif` or `.png` files, `save_labels2pngs` saves output of connected component analysis as a collection of rgba `.png` files. Volumes saved this way may be loaded in ParaView and ImageJ. Furthermore, you are provided with functions for saving volumes as `.vtk` files, which are readable by ParaView.

Quantitative measures from connected component analysis are computed per each connected component, and may be collected in distributions. Measures include: volume, interface area, centroid position. The built-in function `regionprops` provides some of those measures, but the function was originally made for 2D analysis and has a limited 3D support. For this reason, the volume of 3D connected components obtained from `regionprops` as `'Area'` which refers to a voxel count of component. To compute interface are of connected components, binary images need to be processed. Having collected quantitative measures, and depending on problem at hand, you can investigate various relationships: for example, how does the size of the bubbles change with the distance from the horizontal plane? Quantitative measures can also be visualized in 3D, see Fig. 3.

Paraview crash-course for visualizing connected components colored by quantitative measures.

- Open the data. For the data to load, you may need to click apply in the Properties (if not visible, check its visibility in View menu).
- In the Pipeline Browser (if not visible, check its visibility in View menu) visualize the data by clicking the small eye icon next to the filename.
- Choose Representation as Volume (default is Outline).
- Under Properties open additional options by clicking a small gear icon and un-check Map Scalars. This is a crucial step for making ParaView show the RGB color saved in the file, and not mapping it to one of the built-in colormaps. (In my version of ParaView, the colorbar shown will still, wrongly, show the mapped colors. I would therefor turn colorbar off by clicking a rainbow icon in Color Map Editor.)
- If needed, in Color Map Editor (if not visible, check its visibility in View menu) adjust transparency of the Mapping Data by dragging the mapping curve – good result is obtained with linear growing curve, which is usually the default and may be achieved by restoring the defaults by clicking a small recycle icon.