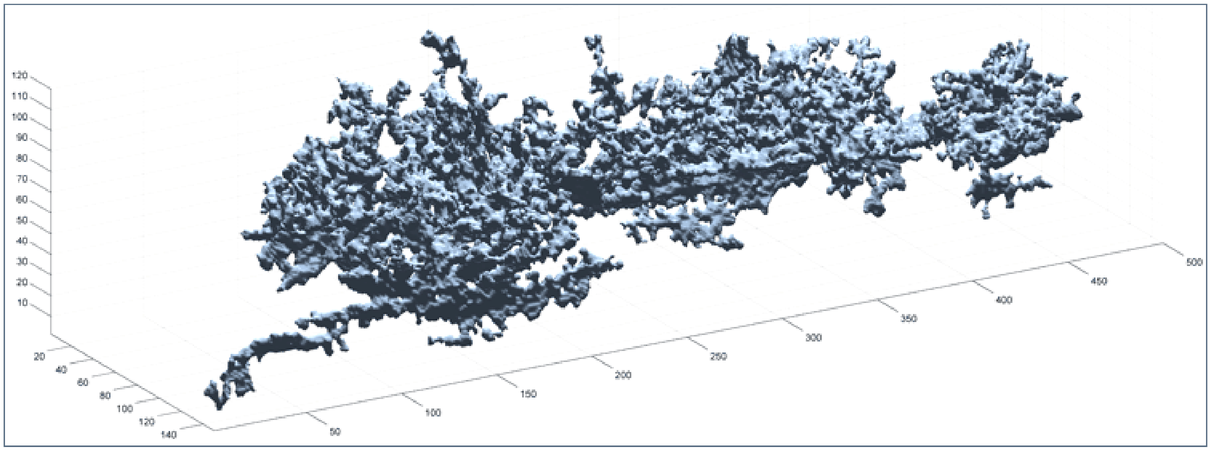
µCT Data Processing Toolbox

Research of contribution of pores and voids inside carbon/carbon composites to the effective properties of the material is a significant part of the project. For this research real pore shapes obtained by the means of X-ray computed microtomography (µCT) were considered. µCT technique results in large amounts of data about microstructure of the analyzed specimen. For the purpose of processing and extracting quntitative information characterizing individual pores in the data, "µCT Data Processing Toolbox" was developed for the "Carbon-Carbon Composites Project". The toolbox consists of several Matlab functions that import µCT data into Matlab environment, find and label all pores inside, extract individual pores and compute their geometric parameters, and finally export individual pores to STL format in the form of 3D triangular surface meshes for Finite Element Analysis. For details on the procedure see description below, comments section in Matlab codes and the paper Drach et al. 2012 "Characterization and statistical modeling of irregular porosity in carbon/carbon composites based on X-ray microtomography data".



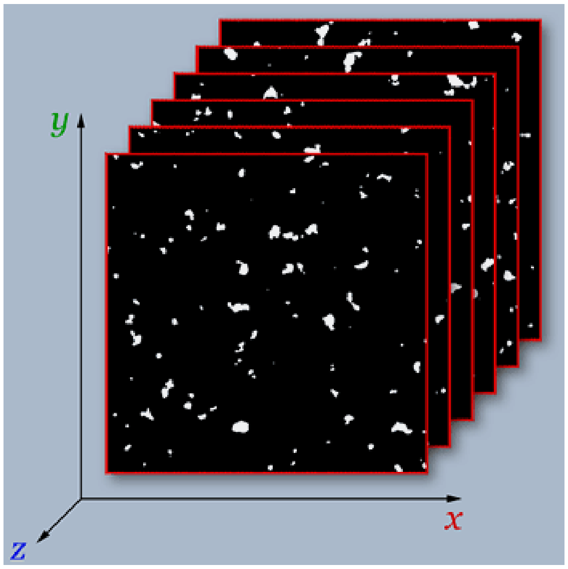
A pore extracted from microtomography data of a C/C composite sample (669 304 faces)
using µCT toolbox described below

µCT Processing Procedure:

1. First microtomography data in 8bit VTK format is imported into Matlab environment. For this purpose function **readVTK.m** (Erik Vidholm, 2006) is utilized.

VTK file is basically a stack of black and white images representing sections ("slices") of the analyzed specimen.

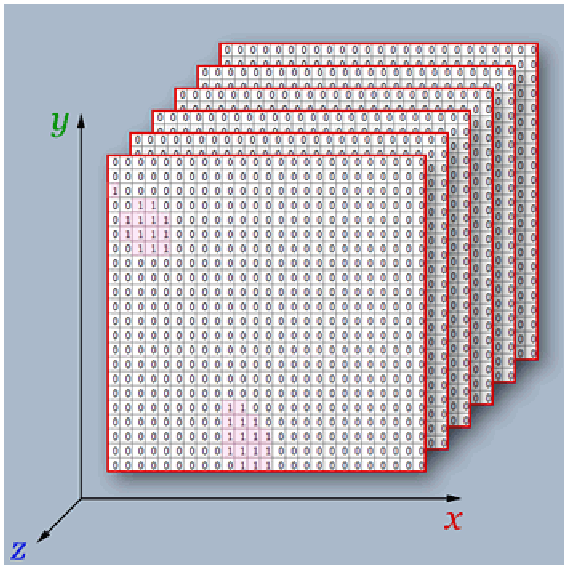
Image on the right shows sveral "slices" of a computer generated VTK file **sample_pores.vtk** (available for download with the toolbox for demonstration purposes).



2. µCT data is stored in Matlab as a 3D matrix: column and row numbers (first and second dimensions) represent coordinates X and Y of the image correspondingly, third dimension represents the "slice" number.

During import of the VTK file, the "slices" are converted to binary images (ones represent pores, zeros - everything else).

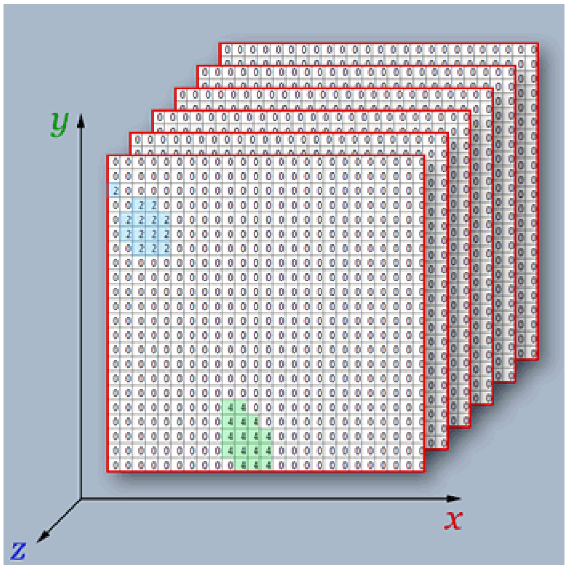
Image on the right shows the 3D matrix (obtained after importing the file **sample_pores.vtk**) as a series of 2D matrices representing individual "slices" each.



3. Using functions from the Matlab Image Processing Toolbox every object (pore) in the 3D matrix is identified and labeled.

In this case an object is a collection of connected white voxels (or ones in the 3D matrix). [*A voxel is a unit of a 3D image, analogous to a pixel in 2D image*]

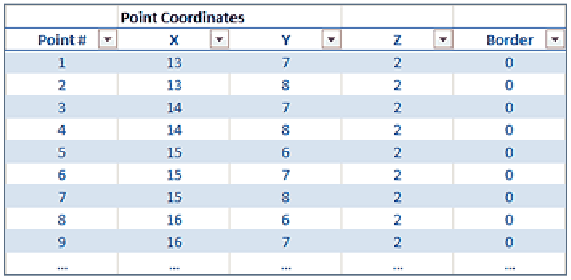
Image on the right shows the same 3D matrix as shown in **Step 2**. with different objects shown in different colors.



4. Next individual objects are analyzed. For illustrative purposes, object **#937** (file **sample_pores.vtk**) was selected.

First, the coordinates of all voxels comprising the object are determined using function **find_cond.m**.

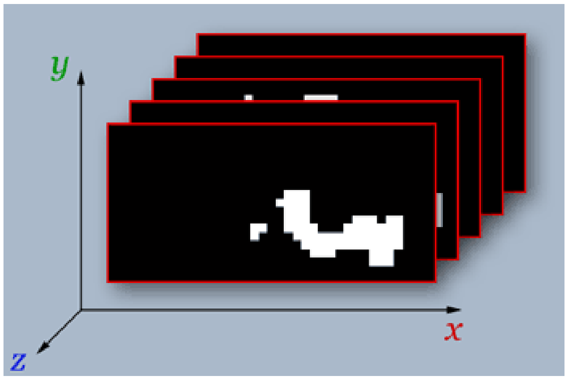
In the procedure, all objects touching the boundaries of the analyzed VTK file are excluded from analysis. Therefore every voxel's coordinate is compared to the VTK file dimensions.



	Point Coordinates			
Point #	X	Y	Z	Border
1	13	7	2	0
2	13	8	2	0
3	14	7	2	0
4	14	8	2	0
5	15	6	2	0
6	15	7	2	0
7	15	8	2	0
8	16	6	2	0
9	16	7	2	0
...	...	...	...	...

5. A new matrix the size of the object is created and the object is copied to the matrix. This is done for faster processing of individual objects.

Image on the right shows "slices" (visualization of the 3D matrix containing the object) of the object **#937** (file **sample_pores.vtk**).



6. 3D shape is extracted from the 3D matrix with the considered object using **isosurface** bult-in Matlab function.

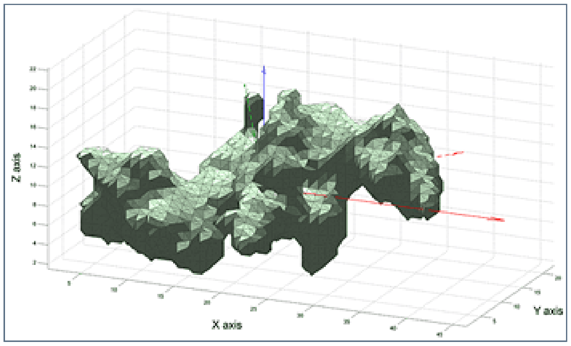
7. Once the 3D shape is extracted the function **isosurf_geom_params.m** is called to determine geometric parameters of the object. The following parameters are calculated: principal moments of inertia (I1, I2, I3), semi-axes of the approximating ellipsoid (a, b, c), directions of the principal axes (V11, V21, V31, V12, V22, V32, V13, V23, V33), object volume (Vol) and surface area (Area), number of the triangular elements (Faces) comprising the shape, linear dimensions of the object (X_i, Y_i, Z_i).

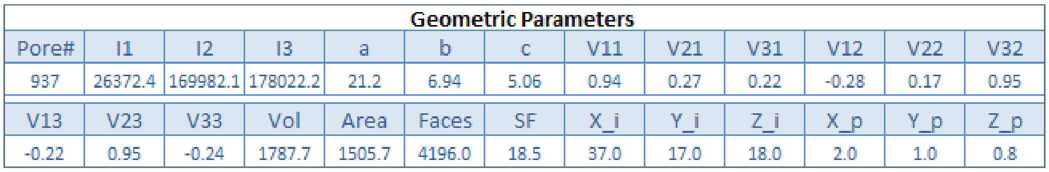
8. The object is then rotated so that its principal axes are aligned along global coordinate axes and scaled so that the largest linear dimension is equal to 2 (scaling factor is stored for further calculations). Scaling is performed for easier FEA analysis of the individual pores (objects). Transformations are performed by the function **isosurf_transform.m**.

9. 3D shapes of the individual objects are then exported as STL files using **stlwrite.m** (Sven Holcombe, 2012).

Image on the right shows the object **#937** (file **sample_pores.vtk**) with global (solid lines) and principal (dashed lines) axes.

Table below shows the summary of geometric parameters of the object **#937** (file **sample_pores.vtk**).





Geometric Parameters

Pore#	I1	I2	I3	a	b	c	V11	V21	V31	V12	V22	V32
937	26372.4	169982.1	178022.2	21.2	6.94	5.06	0.94	0.27	0.22	-0.28	0.17	0.95

V13	V23	V33	Vol	Area	Faces	SF	X_i	Y_i	Z_i	X_p	Y_p	Z_p
-0.22	0.95	-0.24	1787.7	1505.7	4196.0	18.5	37.0	17.0	18.0	2.0	1.0	0.8

Below is a download link to the "µCT Data Processing Toolbox". Image Processing Toolbox v7.1(or newer) is required. The codes were developed for and tested in Matlab 7.11.0 (R2010b).

Download Here

Description of Contents:

- **Folder 'png'**. Snapshots of the extracted objects will be saved here;
- **Folder 'stl'**. STL files of the extracted objects will be saved here;
- **Folder 'vtk'**. VTK files are read (by the main code) from this folder;
- **File 'vtk/sample_pores.vtk'**. Generated sample 8-bit VTK file: (256x256 pixels, 256 "slices");
- **File 'filename_app_zeros.m'**. Matlab function, appends proper number of zeros to the names of the extracted objects' files (for consistent numbering);
- **File 'find_cond.m'**. Matlab function, converts 3D matrix element numbers to coordinates;
- **File 'isosurf_geom_params.m'**. Matlab function, computes geometrical parameters of input isosurface object;
- **File 'isosurf_transform.m'**. Matlab function, transforms (rotation, scaling, moving) isosurface object;
- **File 'process_vtk_stl.m'**. Main Matlab code;
- **File 'readvtk.m'**. Matlab function, imports VTK file into Matlab Workspace (Erik Vidholm, 2006). Also available on Centre for Image Analysis (Swedish University of Agricultural Sciences, Uppsala University) website;
- **File 'stlwrite.m'**. Matlab function, writes STL file from patch or surface data (by Sven Holcombe, 08-20-2012). Also available on Matlab File Exchange portal;
- **File 'stlwrite_license.txt'**. License file for the STLWRITE function.