

Network extraction code – pnextract

pnextract extracts a conventional pore network from a microCT image. The algorithm is a rewrite of the Dong and Blunt (2009) code. There are major differences though. First, the pore and throat detection algorithm is revised; see Stages 1 and 2 described in Raeini et al. (2017) <https://doi.org/10.1103/PhysRevE.96.013312>. Raeini et al. (2017) is an extension of this code. The shape of pores in this code are deduced from shape factors, the shape-factor equation is changed compared to the old definition, see Bultreys et al (2018, currently under-review).

1 Input file

The input file for the network extraction code is a mhd header file compatible with Paraview and Fiji (ImageJ with plugins) with additional optional keywords specific to network extraction algorithm. See the file Image.mhd for a sample input.

Additionally, if the input to the code has a suffix of .raw, or .raw.gz, the code will try to interpret the file name for image and voxel size, if it is formatted like Image_1000x1000x1000_3p5um.raw.gz: the first three numbers are interpreted as image size (1000^3) and the forth as the voxel length ($3.5\mu m$). In this case, the void space voxels are assumed to have been set to 0.

1.1 Input file keywords:

1. The order of the first 6 keywords should not be changed for compatibility with third-party software (ImageJ and Paraview)
2. Use “#” or “//” for adding comments
3. All keyword and its data should be given in a single line

Important keywords:

1. `ObjectType = Image` – ignored by ibvoxel and gnextract
2. `NDims = 3` – ignored by ibvoxel and gnextract
3. `ElementType = MET_UCHAR` – set image data type (`MET_UCHAR`, `MET_USHORT`, applicable to .raw.gz, .raw and .dat/.txt files only).
4. keyword: `DimSize` – used to assign the dimensions of the image: N_x , N_y and N_z
5. keyword: `ElementSize` – used for assigning voxel size: δx , δy and δz should be equal
6. keyword: `ElementDataFile` – specifies the name of the image data file, with suffix .raw, .dat (ascii), .raw.gz, .tif or .am (Avizo format).

```

ObjectType = Image
NDims =      3
ElementType = MET_UCHAR

DimSize =    400   400   400
ElementSize = 5.345 5.345 5.345
Offset =      0      0      0

ElementDataFile = Berea.raw

```

Fig. 1: Sample input header file

Additional image processing commands, (see libvoxel/voxelImageProcess documentation) can be given immediately after the above keywords, or as arguments to a "VxlPro" keyword enclosed by brackets.

```

VxlPro: {
  crop: 0 0 0 200 200 200;
  circleOut: z 500 500 450  //< Exclude outside of a cylinder centred
    at x=500,y=500 and of radius 450 voxels.
  redirect: z //< flip x and z directions.
}

```

Medial-surface settings (optional)

The medialSurfaceSettings is an optional technical keyword which can be used for sensitivity analysis, for instance.

```
medialSurfaceSettings 0.1 0.9 0.7 0.5 1.5 1.21 7 0.25 1.6;
```

where the keyword arguments are clipROutx clipROutyz midRFrac RMedSurfNoise lenNf vmvRadRelNf nRSmoothing RCorsf RCors, respectively.

The pnextract code produces few lines showing the settings being used, like:

```

medialSurfaceSettings: 0.05 0.98 0.7 2.75 0.6 1.1 3 0.15 1.75
medialSurfaceSettings:
  clipROutx      : 0.05
  clipROutyz     : 0.98
  midRFrac       : 0.7
  RMedSurfNoise  : 2.75
  lenNf          : 0.6
  vmvRadRelNf    : 1.1
  nRSmoothing    : 3
  RCorsf         : 0.15

```

```
RCors : 1.75
```

The first line is the keyword and its parameters and the rest are short names for each of the parameters and their values. In case you want to do a quick evaluation, you can copy the first line into the pnextract input, the .mhd file, and change the parameters and re-run the code. Here is a short explanation for these parameters:

`clipROutx` is used to limit the size of maximal-spheres extending outside the rock image in the x direction.

`clipROutyz` is used to limit the size of maximal-spheres extending outside the rock image in the y and z directions.

`midRFrac` is the relative size of the distance-map of the voxel between two maximal-spheres, for the spheres to be considered part of the same pore.

`RMedSurfNoise` is a measure of noise amplitude. Decreasing this will likely increase the number of pores, but it also affects the number of corners per throat.

`lenNf` is a relative distance for merging adjacent pores which are too close to each other.

`vmvRadRelNf` is the relative size of the throat between the two pore considered for merging, the contraction should be less than this to merge the nearby pores (that are less than `lenNf` apart), otherwise the pore will not be merged. Decreasing these two will increase the number of pores.

`nRSmoothing` applies a small amount of Gaussian-like smoothing on the computed distance map, which in turn affect the rest of the computations. Decreasing this will probably increases the number of pores.

`RCorsf` controls the distance between the maximal spheres. This is a sensitive parameter, changing it may need changing other parameters to get good results.

`RCors` controls the minimum distance between (small) maximal-spheres. This is a sensitive parameter, changing it may need changing other parameters to get good results.

2 Visualization and optional outputs data

Additional data generated during network extraction can be saved for visualization or further analysis by providing the following set of keywords starting with “write_”. The “write_all true” keyword is a short-cut for (and has a higher priority over) these keywords:

```
write_radius:      true;  //!< writes distance maps

//! Write pore labels in a _VElems(.raw...) image file
write_elements:    true;

//! Write pore maximal balls mapped to image (.raw...) format
write_poreMaxBalls: true;

//! Write throat maximal balls mapped to image format
write_throatMaxBalls: true;

//! Write _throats mapped to image (.raw...) format
write_throats:     true;

//! Write medial-surface branches, in .vtu format for Paraview
write_hierarchy:   true;

//! Write medial-surface branches (partially joined to form surfaces) in
    .vtu format (deactivated in pnextract)
write_medialSurface: true;

//! Write pore-throat lines along their medial axes in .vtu format
write_throatHierarchy: true;

//write_cnm: true;  //!< used in GNM to also write the classical network

//write_fullThroats: true;  //!< writes full throat images, GNM only

//write_statistics: true;  //!< not implemented, use pnflow code instead

//! Write pore and throats for 3D visualization,
    outputs: *_pores.vtu, *_throats.vtu, *_throatsBalls.vtu
write_vtkNetwork:  true;

//! Angular resolution for cylinders and spheres in write_vtkNetwork
vtk_resolution:   8

//! Scale radial extrusion of throats in write_vtkNetwork
vtk_scaleRthroat: 1.0

//! Scale radial extrusion of pores in write_vtkNetwork
vtk_scaleRpore:   1.0
```

3 The Xmf network format (pnextract version 2021+)

If you are using the public domain version of pnflow please ignore this section.

In the pnextract versions 2020+, the classical network data are written in Xmdf format, essentially in the form of a single ASCII (text) xml file which can be opened in any light-weight text editor, or visualized using VTK/Paraview. The file suffix is Net.xmf. This format similar to the format used in gnflow and pnflow versions 2020+ quasi-static (capillary-dominated) two-phase flow codes.

To use the Xmf format the pnflow code, you should assign it to the `networkFile:` `PATH/TO/ROCK/ImageNet.xmf` in the pnflow input file. The keyword `NETWORK` is reserved for reading network files in old (Oren/Statoil) format described in the next section.

4 The Structure of Network Data Files Statoil format

This Section is taken from PhD thesis of Taha Sochi (2007), Appendix I.

The network data are stored in four ASCII files. The format of these files is that of Statoil. The physical data are given in SI unit system.

4.1 Throat Data

The data for the throats are read from the link files. The structure of the link files is as follows:

***_link1.dat file**

The first line of the file contains a single entry that is the total number of throats, say N , followed by N data lines. Each of these lines contains six data entries in the following order:

1. Throat index
2. Pore 1 index
3. Pore 2 index
4. Throat radius
5. Throat shape factor
6. Throat total length (pore center to pore center)

```

364292
  1   -1  32923  1.851E-005  1.737E-002  8.236E-005
  2   -1  11893  8.402E-006  4.457E-002  1.479E-004
  3   -1   187  2.571E-005  2.476E-002  1.116E-004
  4   -1  50384  1.134E-005  1.490E-002  6.083E-005
  ...

```

Fig. 2: Example of *_link1.dat file

***_link2.dat file**

For a network with N throats, the file contains N data lines. Each line has eight data entries in the following order:

1. Throat index
2. Pore 1 index
3. Pore 2 index
4. Length of pore 1
5. Length of pore 2
6. Length of throat
7. Throat volume
8. Throat clay volume

```

22714  10452  10533  0.178E-04  0.120E-03  0.239E-04  0.218E-13  0.137E-14
22715  10452  10612  0.121E-04  0.747E-04  0.100E-04  0.266E-13  0.355E-14
22716  10453  10534  0.100E-04  0.270E-04  0.139E-04  0.543E-13  0.863E-14
...

```

Fig. 3: Example of *_link2.dat file

4.2 Pore Data

The data for the pores are read from the node files. The structure of the node files is as follows:

***_node1.dat file**

The first line of the file contains four entries: the total number of pores, the length (x-direction), width (y-direction) and height (z-direction) of the network. For a network with M pores, the first line is followed by M data lines each containing the following data entries:

1. Pore index

2. Pore x-coordinate
3. Pore y-coordinate
4. Pore z-coordinate
5. Pore connection number
6. For a pore with a connection number i there are $2(i + 1)$ entries as follows:
 - (a) The first i entries are the connecting pores indices
 - (b) The $(i + 1)$ st entry is the pore inlet status (0 for false and 1 for true)
 - (c) The $(i + 2)$ nd entry is the pore outlet status (0 for false and 1 for true)
 - (d) The last i entries are the connecting throats indices

Note: the inlet/outlet pores are those pores which are connected to a throat whose other pore is the inlet/outlet reservoir, i.e. the other pore has an index of -1/0. So if the $(i + 1)$ st entry is 1, one of the connecting pores indices is -1, and if the $(i + 2)$ nd entry is 1, one of the connecting pores indices is 0.

```
12349  0.3000E-02 0.300E-02 0.300E-02
1      0.350E-03 0.000E+00 0.700E-04 3 796 674 2 0 0 522 523 524
2      0.450E-03 0.500E-04 0.000E+00 3 359 31 1 0 0 525 526 524
3      0.880E-03 0.100E-04 0.000E+00 1 392          0 0 527
...
```

Fig. 4: Example of *_node1.dat file

***_node2.dat file**

For a network with M pores, the file contains M data lines. Each line has five data entries in the following order:

1. Pore index
2. Pore volume
3. Pore radius
4. Pore shape factor
5. Pore clay volume

```
50  0.3733E-13 0.1957E-04 0.3369E-01 0.7846E-16
51  0.1555E-14 0.8215E-05 0.3262E-01 0.4717E-16
52  0.1711E-13 0.1224E-04 0.3298E-01 0.1485E-15
...
```

Fig. 5: Example of *_node2.dat file

Contact:

For any queries please email:

Ali Q. Raeini : a.q.raeini@imperial.ac.uk

References:

Publications: <https://www.imperial.ac.uk/earth-science/research/research-groups/pore-scale-modelling/publications/>

PhD theses: <https://www.imperial.ac.uk/earth-science/research/research-groups/pore-scale-modelling/phd-theses/>