User's manual for the downscaling code

October 22, 2016

1 General information

The downscaling code allows to enhance X-ray computed tomography (XCT) images of heterogeneous porous sample and extract pore space geometry without performing a preliminary noise filtering. The code is accurate when the porous sample is chemically effectively homogeneous across the solid phase and the void phase and the uncertainties of the imaging experiment originate only from the photon shot-noise and the unresolved micro-porosity or from the partial volume effect. Due to the noise, the intensity value of the output data varies even though the sample is chemically homogeneous. The noise can be described by a Poisson probability density function (PDF), which can be approximated by a Gaussian distribution. We assume that the signal-to-noise ratio is high enough to allow identification of the solid and void phases as maximums on the data intensity histogram. The geometrical features of the unresolved micro-porosity are uncertain, but information about the porosity is still preserved in the image intensity. We define the unresolved micro-porosity as a porous phase (see Fig. 1). The main idea of the downscaling code is to map the low-resolution gray-scale image into a high-resolution binary image, while preserving the maximum information regarding the unresolved porosity, then extract the multi-scale pore distribution by segmentation of the void space of the binary image on interconnected void regions bounded by the solid phase (see Fig. 2).

The code and the the XCT data example are available as Amazon AWS EC2 public image (AMI) or at the GitHub
https://github.com/svyatoslavkorneev/downscaling. Amazon AMI is a fully pre-configured environment which allows to deploy Amazon EC2 computing instance and run the code withing a minute. The AWS image also includes operating system Ubuntu 14.04.5 LTS, all required Python libraries and Anaconda. After the computing instance is initialized, the downscaling code is ready to run. You can also download the source code and the XCT data to your local desktop.

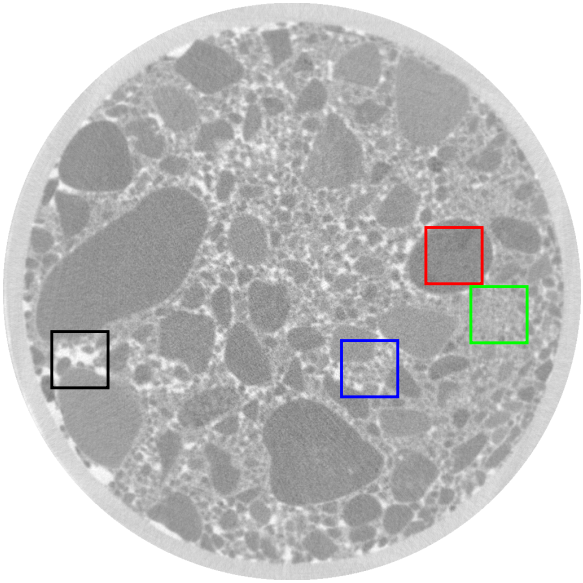

Figure 1: Example of the XCT image, where red square shows the solid phase, black the void phase, green the porous phase and blue mixture of all phases.

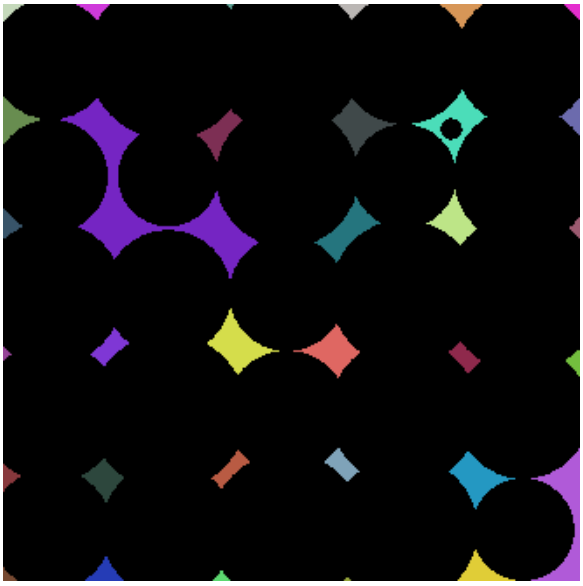

Figure 2: The schematic representation of the segmentation of the void space on bounded interconnected regions, where the black corresponds to the solid phase and the color to the different regions or different pores.

Listing 1 Commands to establish the SFTP connection with Amazon AWS EC2 instance and download the data.

```
# To establish SFTP connection, use Linux command sftp
sftp -i [ssh_key] ubuntu@[instance public IP]

# To download data, use SFTP command get
get [file name]
```

Listing 2 Compilation of the parallel Cython library.

```
#Translate to C code
cython downscale.pyx

#Compile the C code with OpenMP
#and output the compiled shared library
#which can be imported from the Python
gcc -fPIC -fopenmp -shared -o downscale.so downscale.c \
-I/home/ubuntu/anaconda2/include/python2.7/
```

2 Starting Amazon AWS instance

Amazon AWS account is required to launch the computing instance. If you don't have the account, the registration can be done by the web link https://aws.amazon.com. The registration is free. You can find complete user's manual for using Amazon AWS EC2 at the website https://aws.amazon.com/documentation/ec2/. After the login, the computing instance can be launched through EC2 control panel. The instance has to be started inside **US East (N. Virginia)** region. To start the instance push the button *Launch Instance* and choose the section **Community AMIs** at the first step. Type **downscaling_code_Korneev_Battiato** in the search field. At the next step, choose the hardware configuration. The minimal requirements for the code and the given XCT data is the instance **m4.xlarge**. Please note, the **m4.xlarge** is not free of charge. The price list can be found by the https://aws.amazon.com/ec2/pricing/. You can always choose free of charge **t2.micro** and download the source code and the XCT data to your local desktop through SFTP protocol (see Listing 1), then compile the parallel Cython library **downscale.pyx** (see Listing 2). After the instance is initialized, it will be available by the public IP address.

If you decided to proceed with Amazon instance, you have to establish SSH connection with the remote Linux server (see 3). MS Window users can use free SSH client PuTTY to connect to the remote Linux machine.

Listing 3 Command to establish the SSH connection with Amazon AWS EC2 instance.

```
# To establish SSH connection, use Linux command ssh
ssh -i [ssh_key] ubuntu@[instance public IP]
```

Listing 4 Structure of the project.

```
/home/ubuntu/Process/

+-- downscale.c
+-- downscale.pyx
+-- downscale.so
+-- process_downscale.py
|
+-- Images
|   +-- *.tif
|
+-- Mask
|   +-- *.tif
|process_downscale.py
+-- x1x2
|   +-- *.txt
|
+-- Regions
|   +-- *.txt
|
\-- Downscale
    +-- *.tif
```

3 Running the code on Amazon instance

The project is located in the folder **Process** of the home folder. It includes source files inside the root folder, input XCT images in the folder **Images**, mask for XCT images in the folder **Mask**, input values of the intensity of the void and solid phases in the folder **x1x2**, output pore space measurements in the folder **Regions** and output downscaled images in the folder **Downscale** (see Listing 4). The code is executed by the command *python process_ downscale.py*. Please note, after the instance initialization the first execution may take 20 min long due to the IO bottleneck of the cloud network drive. After the first run, IO functions operate in the optimal regime and it takes only 2 minutes on **c4.8xlarge** instance for an image with dimensions 1024×1024 pixels.

The main Python program **process_downscale.py** reads TIFF images from the folder **Images**, inverts and scales the color, so that the maximal inten-

Listing 5 Example of the output of the program process_downscale.py, where delta is the half of the width of the sub-pixel pore at the void phase, porosity is the image porosity, residual is the accuracy of the root finding, number of extracted regions is number of pores of the downscaled image and average region area is the average area of extracted pores in pixels.

```
Image: C6197_0009, delta: 0.755033344803 \
porosity: 0.310749161206, \
residual -9.7293157908e-07, \
number of extracted regions: 450582
C6197_0009 measure regions area...
Average region area is 0.505825171667
C6197_0009 done.
```

sity is 1 and the average intensity of the void phase is greater than the average intensity of the solid phase (see Fig. 1). In the TIFF file with the mask, the area of the interest is masked by pure white and the area outside by pure black (see Fig. 3). The Python program outputs the extracted pore area measurements to the text file inside the folder **Regions**, where the area is measured in pixels. The downscaled TIFF images (see Fig. 4) are written to the folder **Downscale**, where the black pixels correspond to the solid phase and white to the void. Porosity for every image and other useful information are written to the terminal (see Listing 5). The output data can be downloaded to your local desktop through SFTP protocol (see Listing 1). After the code is done, the Amazon EC2 instance can be terminated and you will be only charged for the instance uptime.

The function to perform downscaling is
imagedown, regprop, fi, logstr = downscale(image, mask, scale, x1, x2), where input variable **image** is the 2D array of float numbers, where each element stores pixel intensity and average intensity value for the solid phase has to be less than for the void phase. Input variable **mask** is the 2D array of integers, where value 1 masks the area of the interest and value 0 outside. The value of the integer variable **scale** defines the resolution of the downscaled image with respect to the resolution of the original image. Values of the float variables **x1** and **x2** define the intensity of the void and solid phases, respectively. The function outputs 2D integer array, where each element stores the value of the pixel intensity. Value 1 corresponds to the void phase and value 0 to the solid phase. The function also output the *skimage* object with the list of properties of labeled regions, where each region correspond to a single pore. It also outputs the value of porosity and the string variable with the useful technical information. After the downscaling, the extracted regions properties can be measured by using library *skimage*.

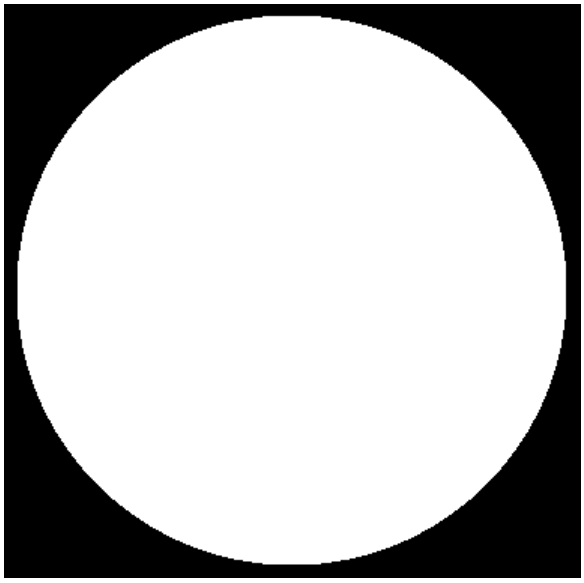

Figure 3: Example of the core mask.

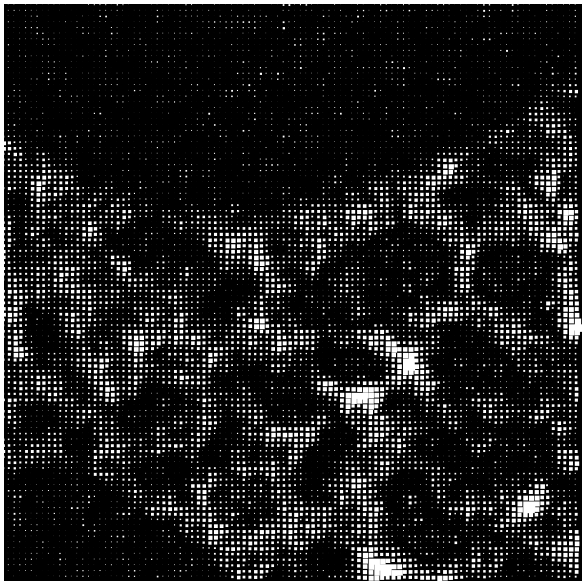

Figure 4: Example of the downscaled binary image, where black corresponds to the solid phase.