The 3D image processing tools

SLICE

JASRI/SPring-8

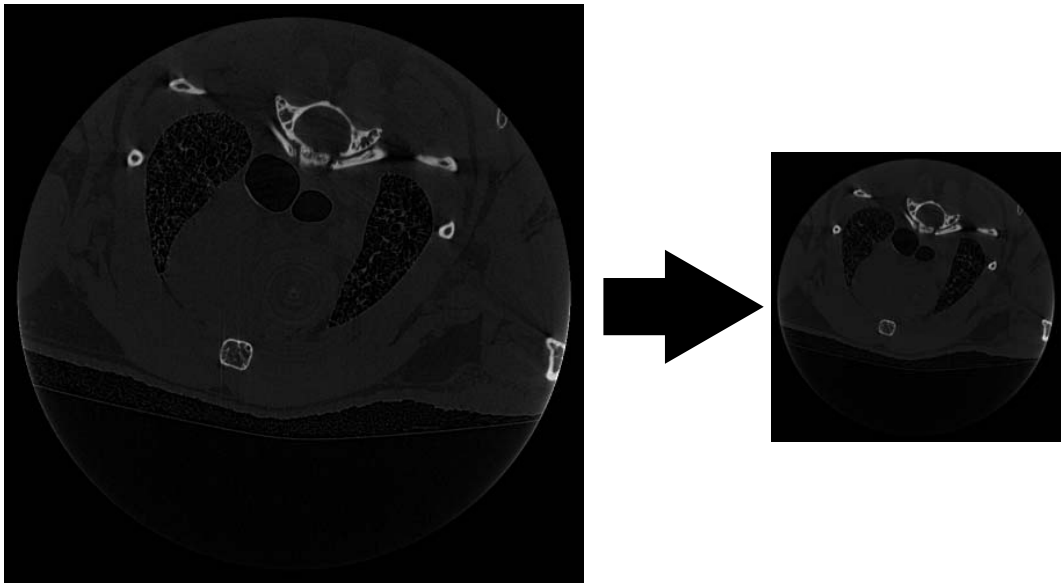


Figure 0.1: zoom（§4.1）

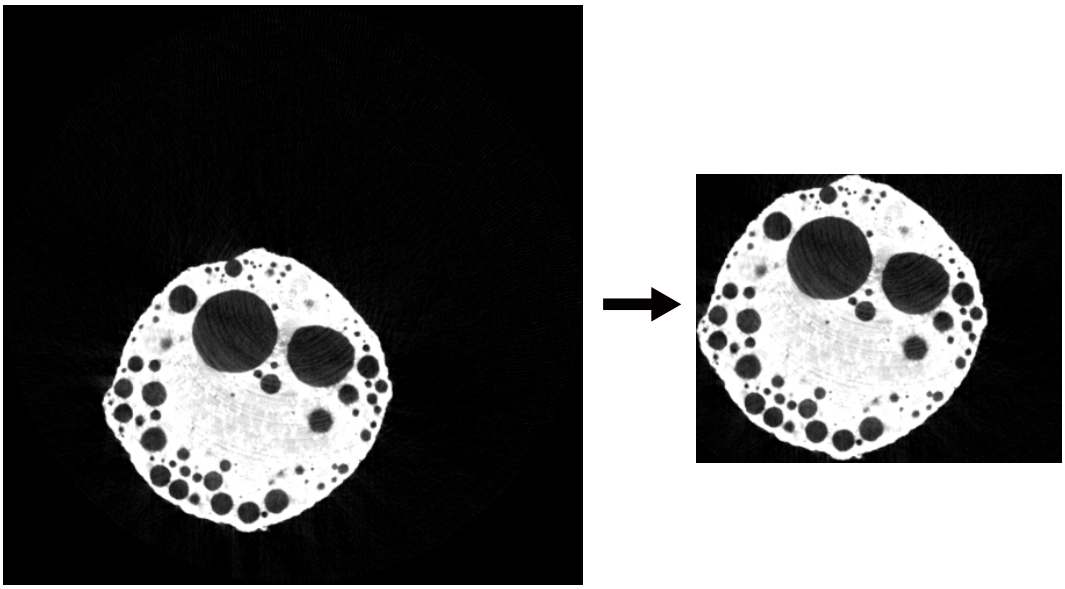


Figure 0.2: triming（§4.2）

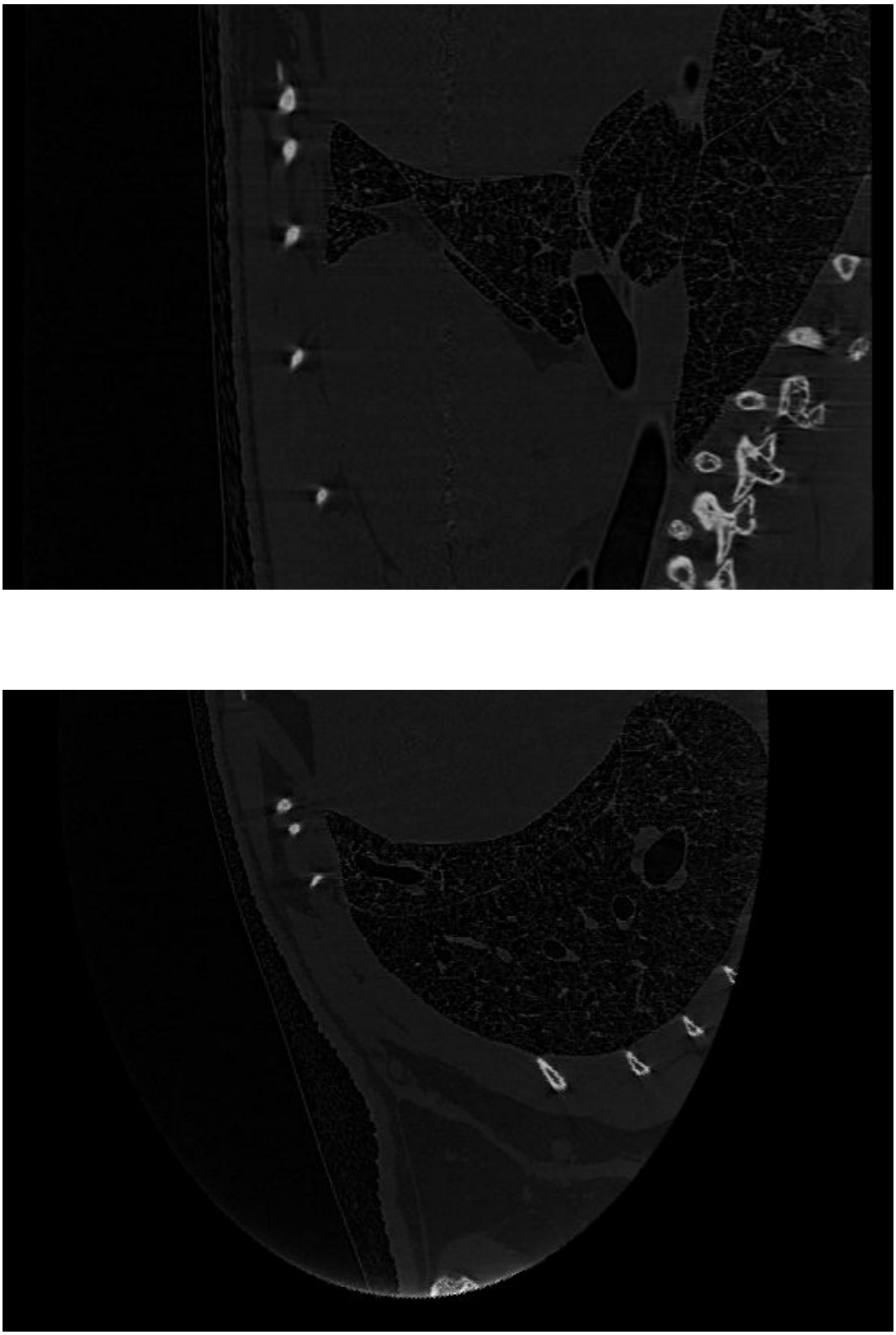


Figure 0.3: reslicing (§5)

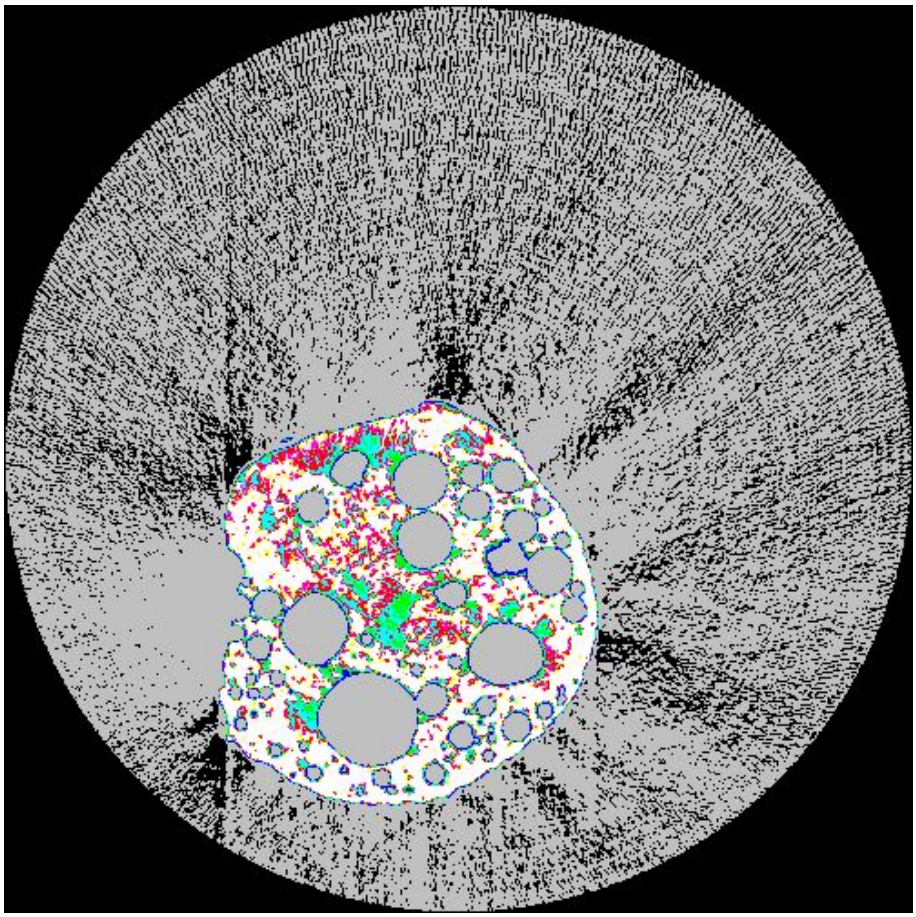


Figure 0.4: colorize the slice images (§5)

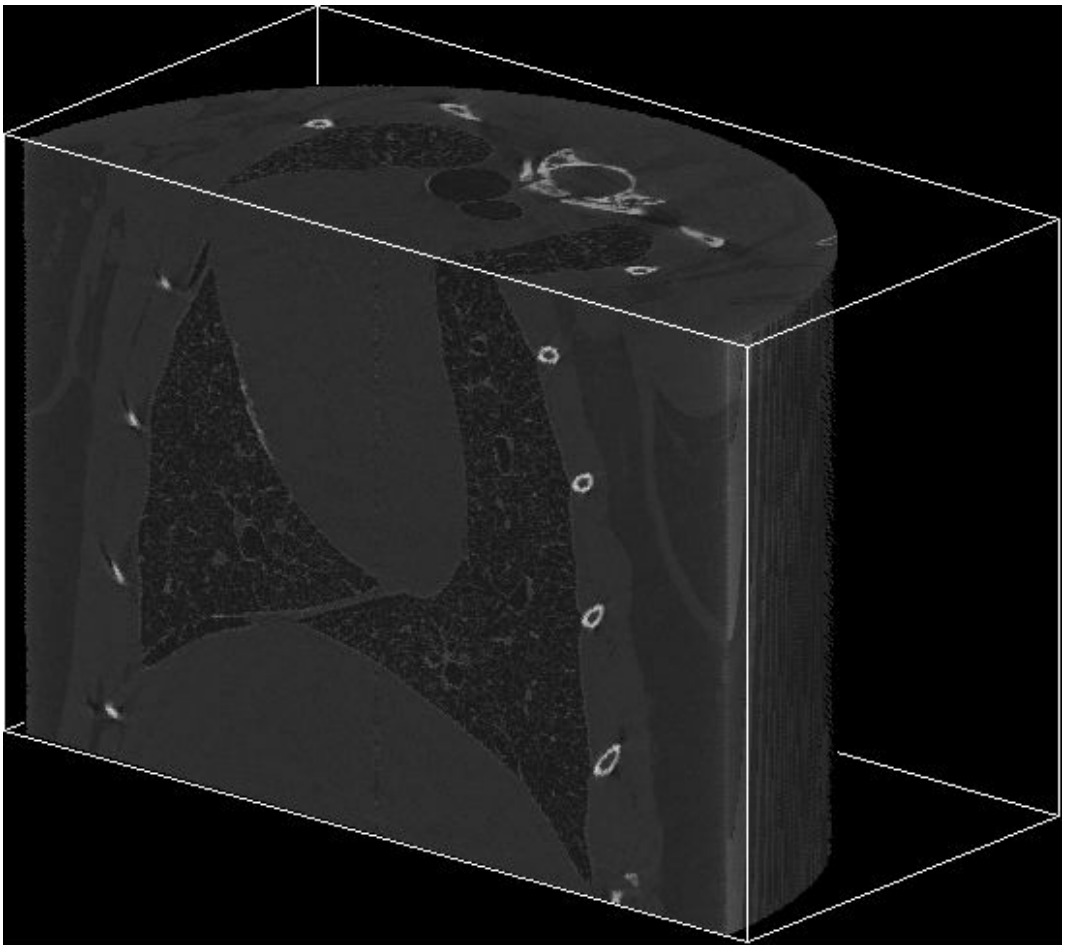


Figure 0.5: volume rendering（§6.1）

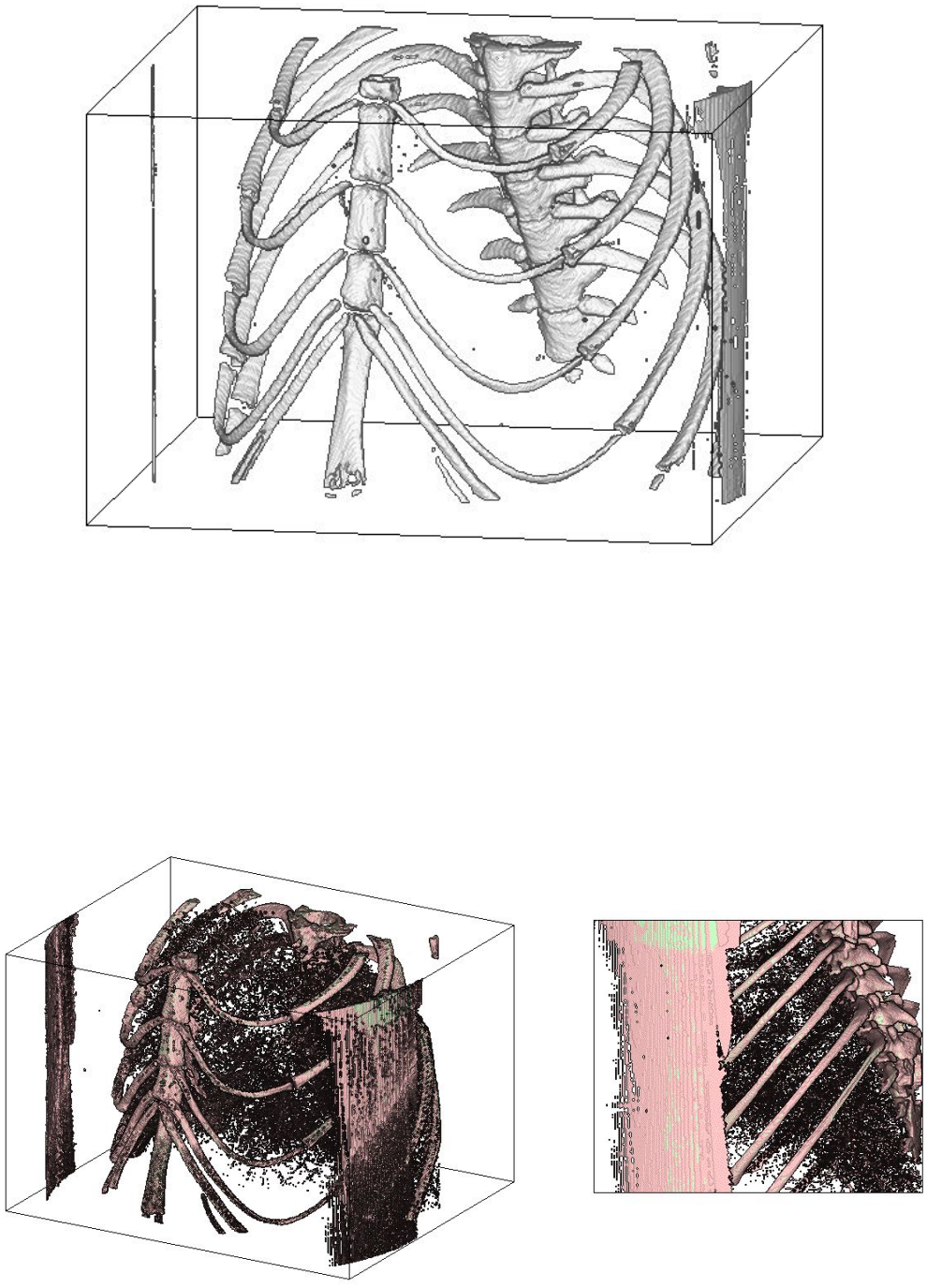


Figure 0.6: Volume rendering with polygon（§8.1）

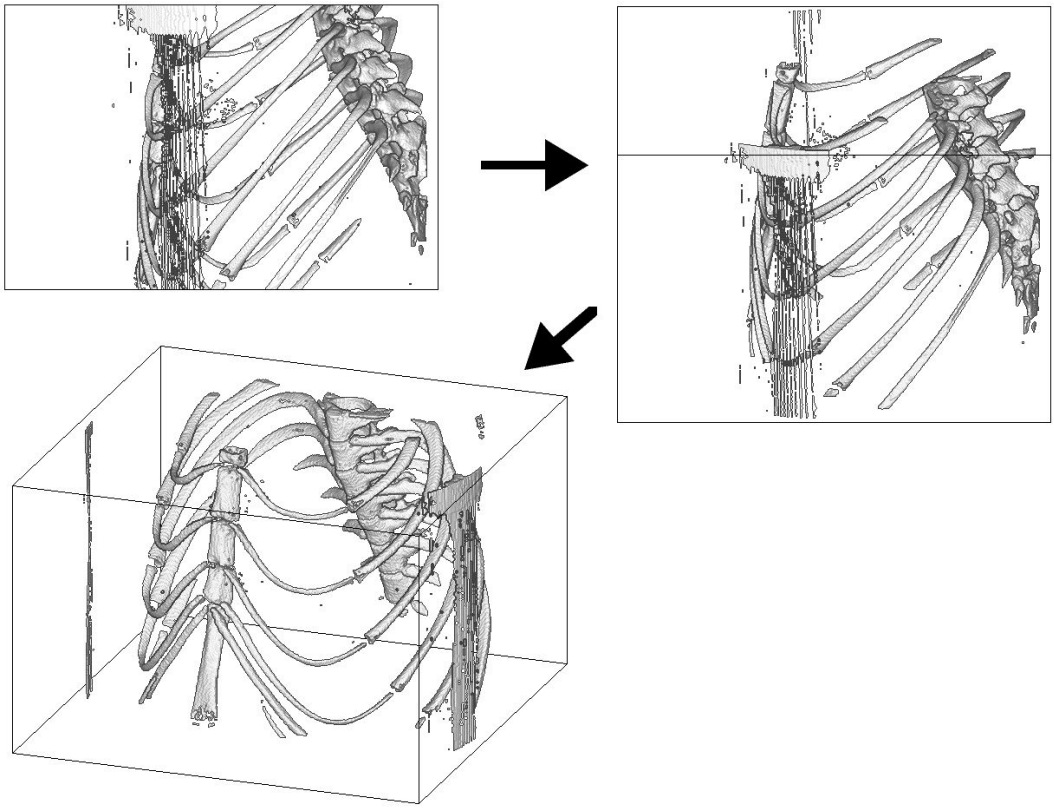


Figure 0.7: Three dimensional movie (§8.4)

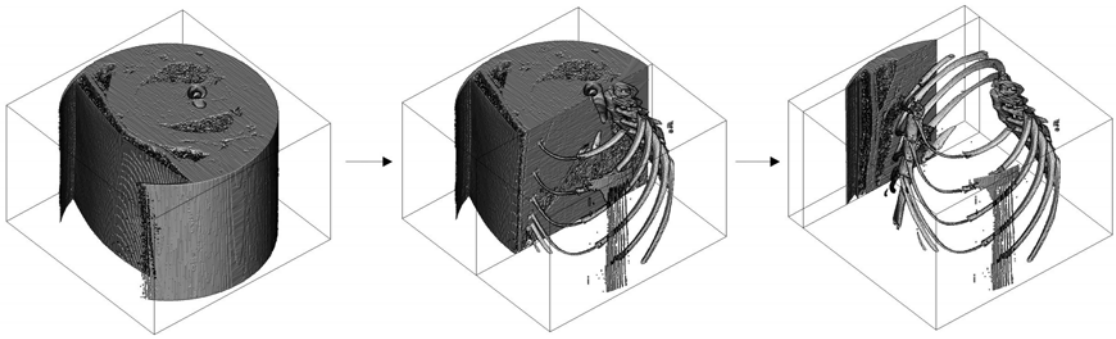


Figure 0.8: incorporating two polygon data and transition movie (§8.6)

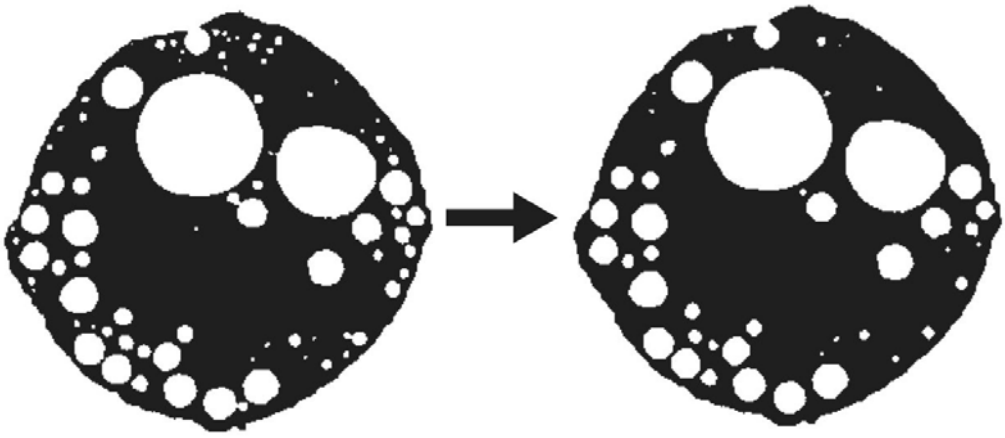


Figure 0.9: binarizing and hole labeling (§9.1)

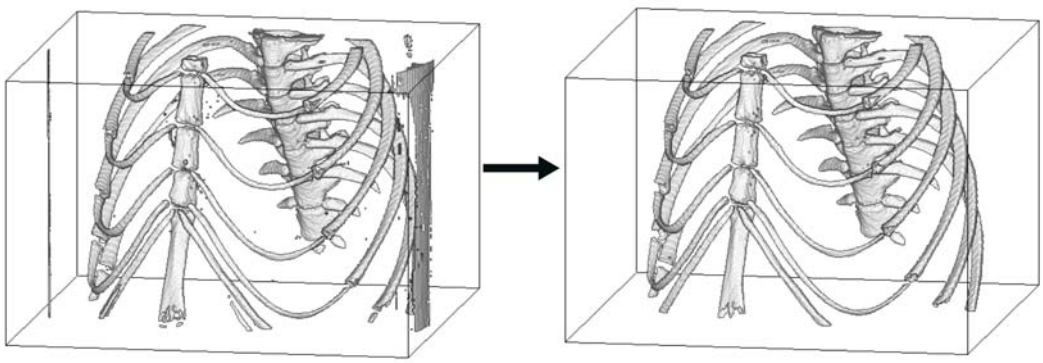


Figure 0.10: binarizing and noise subtraction (§9.2)

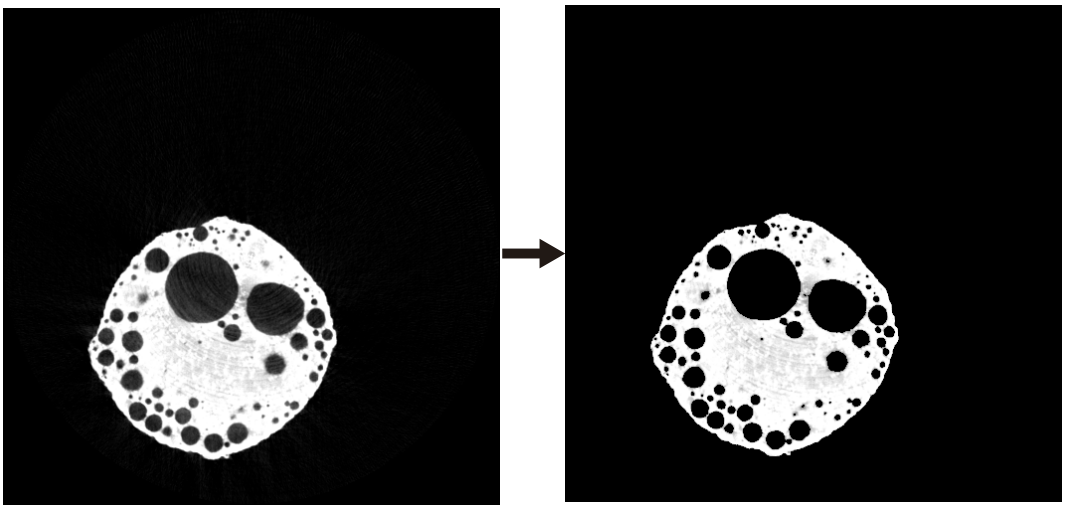


Figure 0.11: binarizing and masking (§9.3)

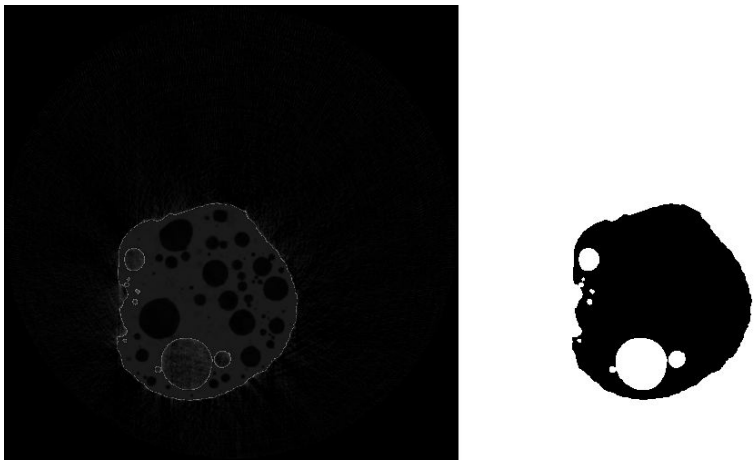


Figure 0.12: transparent overlay (§9.4)

Figure 0.13: ellipsoidal approximation (§10.2)

made by

Tsukasa Nakano (GSJ/AIST, JASRI/SPring-8),

Akira Tsuchiyama (Osaka Univ., JASRI/SPring-8),

Kentaro Uesugi (JASRI/SPring-8),

Masayuki Uesugi (Osaka Univ., JASRI/SPring-8)

and

Kunio Shinohara (Waseda Univ., JASRI/SPring-8)

Contents

1 Introduction

1.1 About "Slice"

Slice is a series of softwares for three dimensional image analysis of CTdata that works on command line. Users can manipulate the image processing and quantitative analysis of three dimensional data by Slice series, and also create new images for presentations.

The advantage of Slice is image processing of three dimensional data composed of two dimensional images, in at once. The image processings such as erosion and dilcation are operated on three dimesional structure, not on the 2D images. And also, Slice is availabe on UNIX, Windows and Macintosh platform, and open source with written by primitive C-language.

This book is a manual for the image processing of CT data by Slice series softwares.

1.2 About this book

The Slice manual is composed of two books. This book is the cookbook for the several image processing using test samples with sample scripts. However, the cookbook does not covers whole programs of Slice series and its usage. If further infomation of Slice series programs is required, please refer Program reference book, which is reference of programs included in Slice series.

1.3 Bibliographic Reference

Please use following reference sample for using images created by Slice on public.

Tsukasa Nakano, Akira Tsuchiyama, Kentaro Uesugi, Masayuki Uesugi and Kunio Shinohara (2006) Slice -Softwares for basic 3-D analysis-, Slice Home Page (web), http://www-bl20.spring8.or.jp/slice/, Japan Synchrotron Radiation Research Institute (JASRI).

1.4 Important

Poor English in the text is made by MU. If you have any suggestions for the correction, please access http://www-bl20.spring8.or.jp/slice/

2 Getting Started

2.1 Download files

Slice works under unix environment. If you want to use on the Windows platform, you can use Cygwin environment. The install files and documents for the installation is available at
http://www.cygwin.com/

Files of slice program is available at
http://www-bl20.spring8.or.jp/slice/

Body of slice(XXXXXX is date)
http://www-bl20.spring8.or.jp/slice/file/sliceXXXXXX.tar.gz (2.2MB)

Grain identification
http://www-bl20.spring8.or.jp/slice/file/gi.tar (1MB)

3D image mosaic
http://www-bl20.spring8.or.jp/slice/file/rmsd.tar (1.6MB)

inclination correction
http://www-bl20.spring8.or.jp/slice/file/irac.tar (730MB)

STLfile operation（NIST）
http://www.gsj.jp/GDB/openfile/files/no0448/0448index.html (5MB)

sample data for training
http://www-bl20.spring8.or.jp/slice/file/mouse.lzh (35MB)
http://www-bl20.spring8.or.jp/slice/file/bubble.lzh (28MB)

The “body of slice” is requied for all image processing shown in this book, and stl.tar.gz is partially required for advanced volume rendering processes.

2.2 Compile and install

Open the terminal window, and go to the directory where the downloaded files are extracted. Change directory to the slice (or other) directory where created by extracting the archive, and type ”make install (enter)”.

Please note that executable files are installed on the bin directory in slice directory. If you want to install other directories, you must edit Makefile in slice directory or move the executables manually.

After the installation, environment variable $PATH should be changed or added. In UNIX environment, make following changes

.bash_profile :

PATH=$PATH:absolute path of program directory

.cshrc :
set path = ($path absolute path of program directory)

How to install the files is also introduced in “install manual” uploaded at same page of slice. Please refer it for the installation.

2.3 How to execute the sample script

Slice series programs basically treat the data in directory unit. So the working directory should be upper directory of the data directory where 3D data files are exist. For example, if slice image files are saved at /data/BYTE/ directory, then working directory should be /data/.

Sample scripts in this cookbook can be run from command line, and also run from batch files. For example, you can run slice.No by either typing
slice.No BYTE no -r (enter)
or preparing text file named no.txt with written ”slice.No BYTE no -r” and typing
csh no.txt (enter)
The name of text file is arbitrary.

2.4 Other softwares

Following programs are available for displaying images saved by Slice programs or original slice images.

Windows

- irfan view (Freeware: http://www8.plala.or.jp/kusutaku/iview/)
- ImageJ (Freeware: http://rsb.info.nih.gov/ij/)

MacOSX

- Preview (OS)
- ImageJ (Freeware: http://rsb.info.nih.gov/ij/)
- Graphic Converter (shareware: http://www.bridge1.com/ grahpicconverter.html)

UNIX

- ImageJ (Freeware: http://rsb.info.nih.gov/ij/)
- Image magick
- XV (http://www-bl20.spring8.or.jp/slice/file/xv-hipic.tar.gz 2.6MB)

3 Fundamentals

3.1 About the pixel value

X-ray CT is a imaging method that use the distribution of transmitted X-ray flux through the samples to observe the internal structure. The contrast of the CT slice images shows the linear absorption coefficient (LAC value, cm^{-1}) of the local material converted into gray value of the picutre, called pixel value (PV). The relation depends on the X-ray energy, experimental devices, and chemical composition of the matetrial.

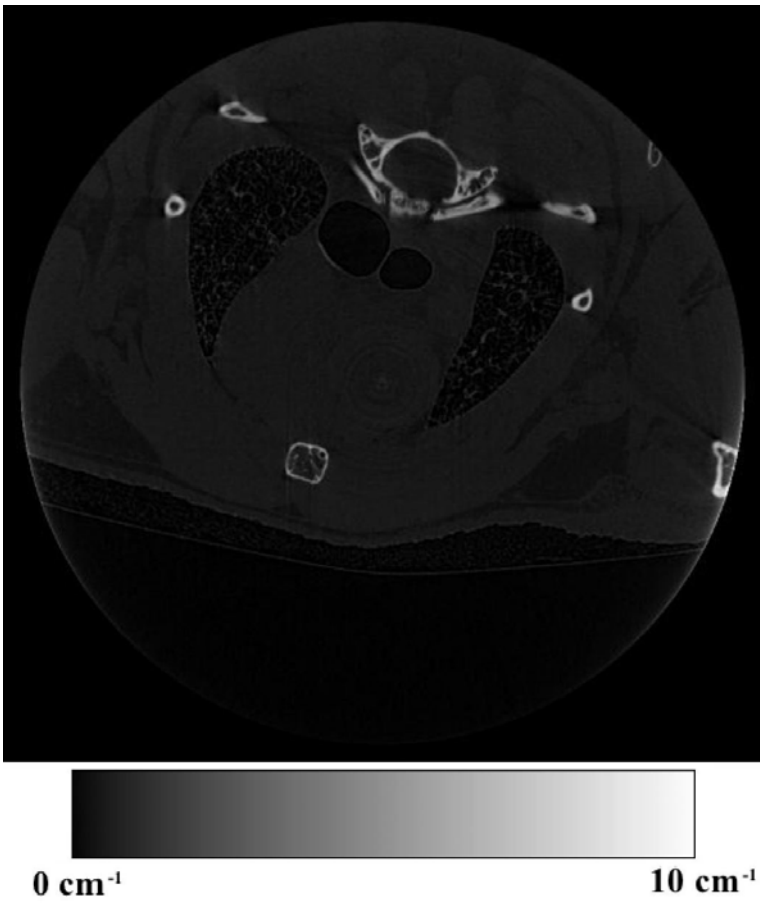


Figure 3.1: relation of the gray scale of the image and LAC value

If the contrast of each slice image is different, it indicates that the relation between the pixle value and LAC value is not uniform, and should be corrected. Futher information for the correction and LAC value is available in the text of beamline introduction.

3.2 Coordinate

The reconstructed 3D image data of the CT has a series of file name such as 0000.tif, 0001.tif. The origin of the coordinate for the CT data is left-upper of the image that has smallest number (see Figure). The Slice proograms and this manual are based on this coordinate, but some of the programs use different coodinate (see §8.5).

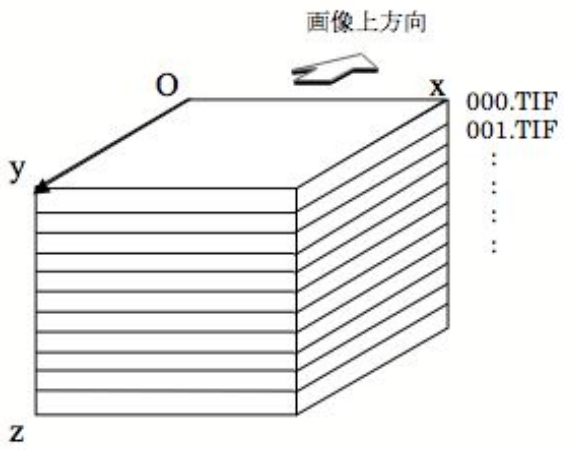


Figure 3.2: coordinate

3.3 Sample data

The sample data is available from the Web site of the Slice. mouse.lzh is a CT image data of a mouse that composed of 000.TIF ∼ 327.TIF、500x500 pixel images.

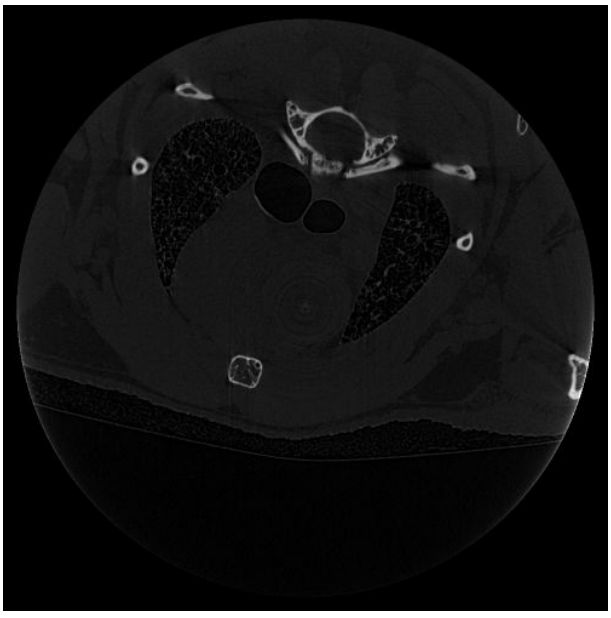


Figure 3.3: Example of the sample data : 000.TIF

The bubble.lzh is the heated sphere of a rock which composed of 000.TIF ∼ 297.TIF、500x500 pixel images.

The sample scripts in this cookbook use these examples. For the case of the analysis of the original data, the filename and directory name should be changed to the names appropreate for the data.

4 Resizing and triming

Some program in Slice requires amount of available memory. So if the memory is not enough to the operation, such programs would appear errors and abort. Such cases can be avoided by reducing the image size. sliceXXXXX.tar.gz should be installed for the image processings of this section.

4.1 Resizing (sliceBIC)

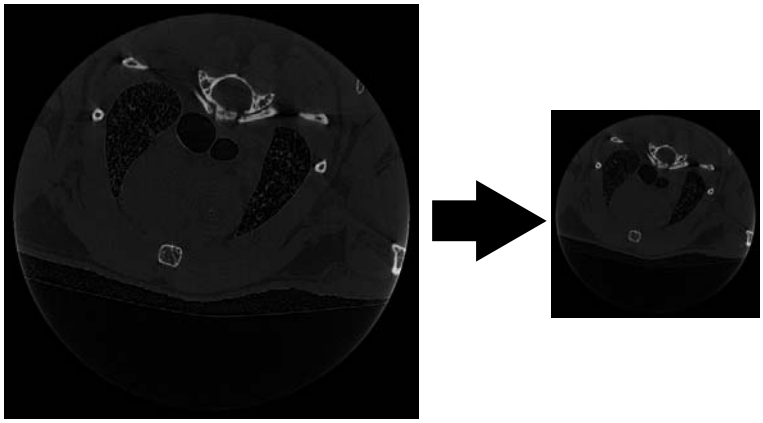


Figure 4.1: Reducing slice image

sample script

```
mkdir bic
sliceBIC BYTE - 2 2 2 bic
```

1st line: make save directory

2nd line: reduce slice image by sliceBIC. The 2 pixels for x, y ,z direction each are averaged and reduced into 1 pixel, and stored in new directory, bic. The image size for each direction are reduced into 1/2 and total size of the data become 1/8 of original data.

saved images by sample script: bic/000.tif∼163.tif

4.2 Triming (sliceIT)

Trim off the unnecessary region of 3D data, and reduce the data size.

sample script

```
sliceOSP3 BYTE - 140 0
mkdir trim
sliceIT BYTE - 85 209 32 399 456 282 trim
```

1st line: detect cubic area which completely cotain the object by sliceOSP3 (material have pixel value >140)

2nd line: make save directory

3rd line: get cubic data by sliceIT and save to the trim directory

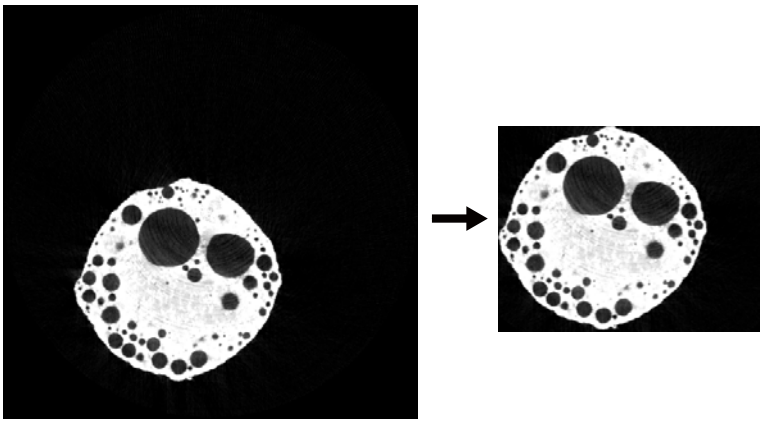


Figure 4.2: Triming

supplement：
Text displayed by sliceOSP3

6545861 85 209 32 399 456 282

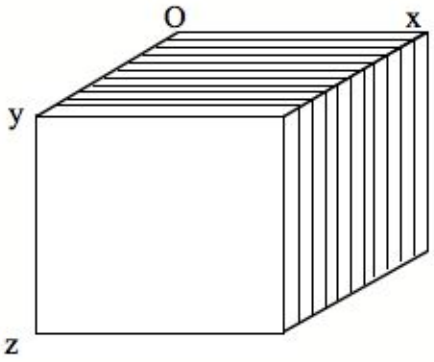
sliceIT requires two sets of (x,y,z) coordinate data of the cubic area that are the nearest vertex of the origin and the diagonal vertex. sliceOSP3 can detect the cubic area which contain the object which composed of pixels which have higher pixel value than the value given in the command line. See §7

saved images by sample script: trim/000.tif∼250.tif

5 Reslicing

Make the 3D image date with variable cutting surfaces that align X, Y, Z plane and arbitrary surfaces. In this section, we use mouse.lzh sample data. sliceXXXXX.tar.gz should be installed for the image processings of this section.

5.1 Reslicing on x-z plane (slice_NSX)



Sample script
`mkdir nsx`
`slice_NSX BYTE - 1 1 1 nsx`

1st line: make save directory
2st line: reslice slice images by slice_NSX

Figure 5.1:

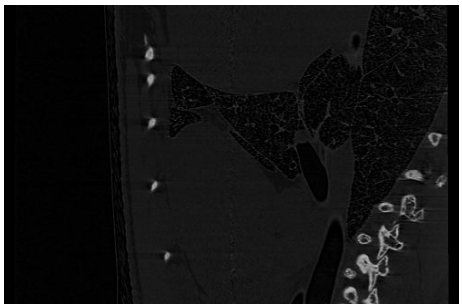
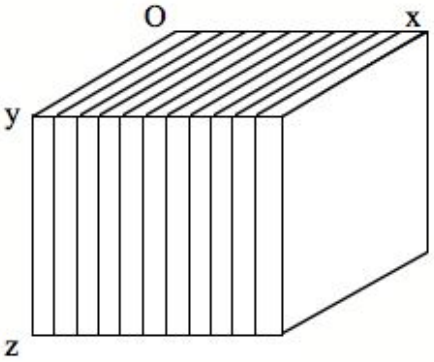


Figure 5.2: Example 250.tif

saved images by sample script: nsx/000.tif∼499.tif

5.2 Reslicing on y-z plane(slice_NSY)



Sample script
`mkdir nsy`
`slice_NSY BYTE - 1 1 1 nsy`

1st line: make save directory
2st line: reslice slice images by slice_NSY

Figure 5.3:

saved images by sample script: nsy/000.tif∼499.tif

5.3 Make slice image by arbitrary surface (sliceNSG)

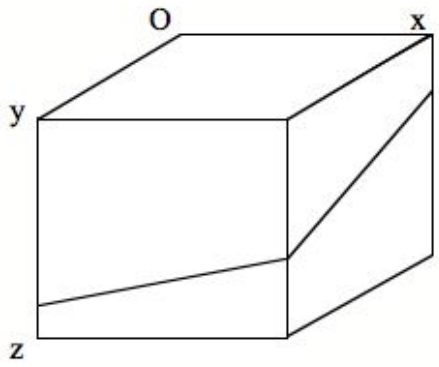


Figure 5.4:

Sample script
`echo 250 250 300 20 40 0 cs.tif | sliceNSG BYTE - 1 1 1 0`

1st line: input the coordinates of the point (250, 250, 300) which included by the new slice surface, to the sliceNSG command using "echo" command. The direction of eye (the direction perpendicular to the new slice surface) lon and lat = (20,40) and rotation in the slice image =0. The scale of all direction is 1, and if the new image data contains the outside area of the original data, the pixel value = 0 is given to the pixel.

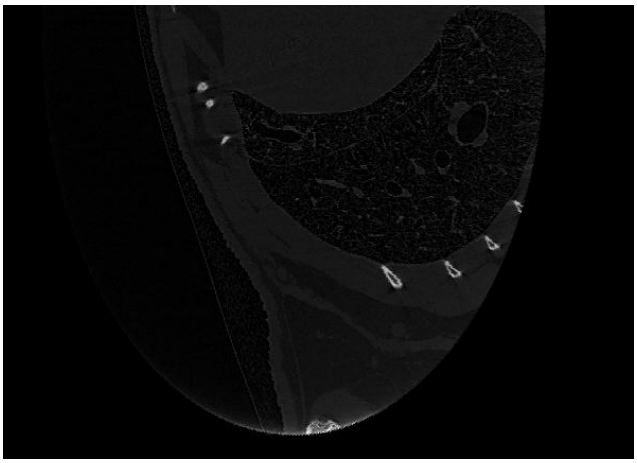


Figure 5.5: Example, cs.tif

saved images by sample script: cs.tif

5.4 Make slice images by arbitrary surface(sliceIR)

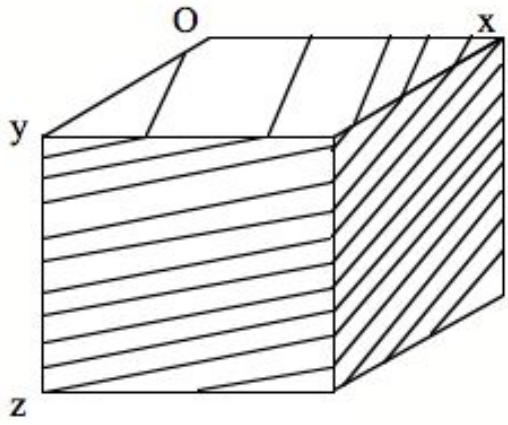


Figure 5.6: make slice images by arbitrary surface

Sample script
`mkdir ir` `sliceIR BYTE - 60 -20 0 0 ir`

1st line: make save directory

2nd line: make new slice images by sliceIR, with rotation of lon and lat = 60,-20 and rotation in the surface of new slice image is 0.

saved images by sample script: ir/000.tif∼752.tif

supplement:
This program store the image data into memory. So if there are smaller memory available for the program, program may be aborted with errors. Such case can be avoided by the size reduction of the image data (see previous section) . This program took some amount of time.

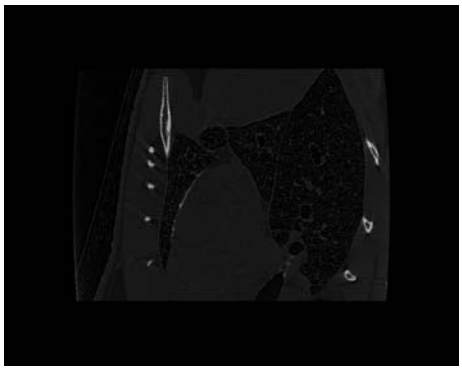


Figure 5.7: Example

5.5 Make gif movie of slice images(gif_movie)

Sample script

```
mkdir  gifs
slice.T2G  BYTE  gifs
gif_movie gifs - 0 65536 slice.gif
```

1st line: make save directory

2nd line: convert slice images from tif to gif images by slice.T2G

3rd line: make gif movie file named as slice.gif which has no deray and infinit loop number.

saved images by sample script: slice.gif

5.6 Colorize of slice images(slice.CMA)

Put colors to the slice images. In this subsection, bubble.lzh was used.

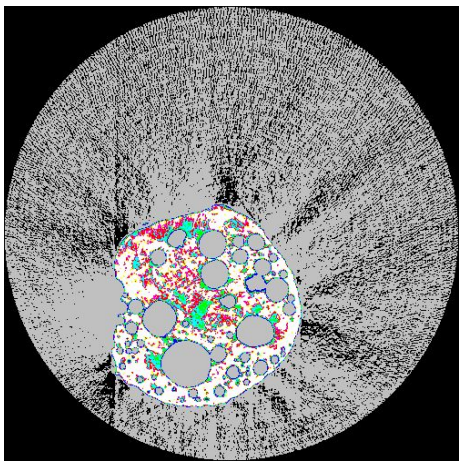


Figure 5.8: Example 200.tif

Sample script

```
mkdir  cma
slice.CMA BYTE cluster.cm cma
```

1 行目: make save directory

2 行目: put colors to the slice images using color table file, cluster.cm, and save them to cma directory

saved images by sample script: cma/000.tif∼297.tif

supplement:
Samples of the color table for slice.CMA are saved in slice/slice/etc in the sliceXXXXXX.tar.gz archive. These color table should be copied to the working directory, or, the directory should be specified in the command line.

6 Volume rendering

Perform volume rendering from the 3D image data. sliceXXXXX.tar.gz should be installed for the image processings of this section.

6.1 Volume rendering(sliceIT, slice.NO, sliceBEVM.DS, bevm.WD, bevmGS, tiffmask)

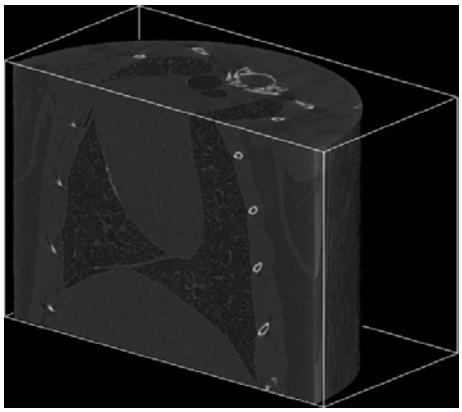


Figure 6.1: Example

Sample script

```
mkdir it
sliceIT BYTE - 0 0 0 499 250 326 it
mkdir no
slice.No it no -r > no.sh
csh no.sh
echo 30 20 40 10 mouse-bevm | sliceBEVM.DS no - 1 1 1
bevm.WD mouse-bevm.d.tif mouse-bevm.w.tif
bevmGS mouse-bevm.s.tif mouse-bevm.l.tif 1 25 0 mouse.ls.tif
tiffmask mouse-bevm.e.tif 1 mouse.ls.tif -0 mouse.lse.tif
tiffmask mouse-bevm.w.tif 1 mouse.lse.tif -255 mouse.tif
```

1st line: make save directory

2nd line: trim the slice images by sliceIT

3rd line: make save directory,“no”

4th line: make script file to sort files by slice.No

5th line: sort files to reverse order and save “no” directory

6th line: input the lon and lat of viewpoint = (30,20) and light source = (40 10), and filename “mouse-bevm” to sliceBEVM.DS using ”echo” command

7th line: get flame of 3D ragion from depth image (mouse-bevm.d.tif)

8th line: render 3D image with shade from label image (mouse-bevm.l.tif) and shade image (mouse-bevm.s.tif)

9th line: put edge enhance effect from edge image (mouse-bevm.e.tif)

10th line: put flame (mouse-bevm.w.tif) on the 3D image

supplement :
Because the coordinate of the 3D data is different from general coordinate, the bird eye's view image created by sliceBEVM.DS would be rotated. See §8.5. It is also important to sort the image into reverse direction.

saved images by sample script:
it/000.tif∼599.tif
mouse-bevm.s.tif, mouse-bevm.e.tif, mouse-bevm.d.tif, mouse-bevm.l.tif , mouse-bevm.w.tif, mouse-bevm.ls.tif, mouse-bevm.lse.tif, mouse.tif

7 Binarizing

7.1 Binarizing

The contrast of the CT image basically represent the difference of local material. By making threshould of the pixel value (i.e.LAC value), we can divide the object of the interest and other object in the CT image data. This procedure is called as "binarizing". In this procedure, generally the pixel value of the object is set to 1 and that of other object is set to 0.

Table 7.1: difference of gray scale data and binarized data

gray scale data	binarized data
Observation of whole internal structure of the sample	Observation of specific objects
Not suitable for quantitative analysis	suitable for quantitative analysis

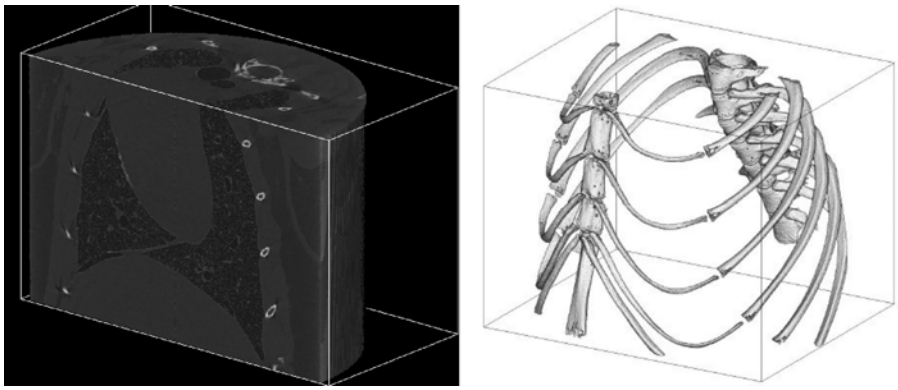


Figure 7.1: Examples of bird eye's view image of a mouse by gray scale data (left) and binarized data (right)

For the case of gray scale data, the gradation in the image is based on the LAC value. On the other hand, for the case of binarized data, the contrast of the image shows the shape of the surface of the object. Thus, the structure of a specific material inside of the sample can be well observed by binarized data rather than gray scale data.

7.2 Preparation of binarizing: threshould

The pixel value of the CT image shows the LAC value and can be used for distinguishing the internal material. In this subsection, a simple procedure of the binarizing is introduced. The thresould value is the boundary of the pixel value between the body of interest and other portion of the CT data. How to obtain the threshould value is different according to the sample structure. If the object of the interest has largely different pixel value (i.e. LAC value) from the surrounding material, finding the threshould value is easy. Figure 7.2 show the histgram of the sample data of mouse. In the histgram, a highest peak near the 0 is a air (the portion without objects). The next peak near the 30 shows the body of the mouse. The threshould value of the mouse body can be the valley between the peaks. In the general cases, finding the threshould value is empirical and is complicated. In addition, several procedures, such as noise reduction or selection of the objects, are required before the quantitative analysis.

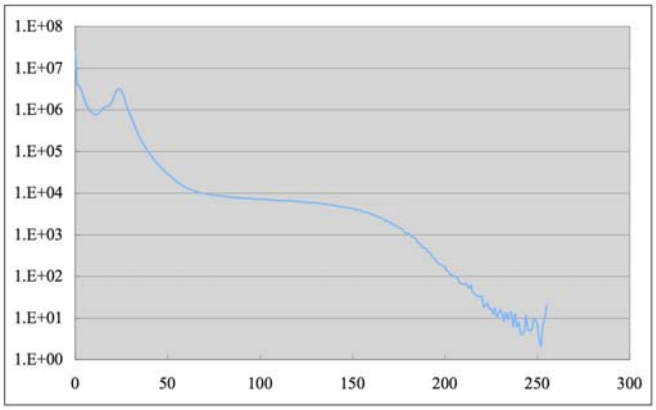


Figure 7.2: histgram of the sample CT data, mouse.lzh

7.3 Example

Here we show a some representative method for the binarizing (or value multiplexing) of CT data. There are several possible ways as well as the method introduced in this section. A number of programs in the Slice series, such as sliceBEVM.DS, sliceDE, sliceED can binarize the images with their specific procedure. Further information for those programs is available in the program reference.

7.3.1 Example 1:Binarizing based on the pixel value only (slicePVR).

Following sample script show the binarizing procedure by slicePVR program. By this script, slicePVR saves CT data to pvr directory the images that pixel value 0-127 to be replaced to 0 and 128-255 to be replaced to 1.

Sample script

```
mkdir pvr
(echo 0 127 0 ; echo 128 255 1) | slicePVR BYTE - pvr > pvr.txt
```

1st line: make save directory

2nd line: binarize the CT data with -127 and 128- using slicePVR

This is the simplest way of the binarizing.

7.3.2 Example 2 : binarizing or value multiplexing based on the pixel value and structure of the object (sliceMCL, sliceMHL)

Next we show the cluster labeling and hole labeling which also can binarize or value multiplex the CT data.

Cluster labeling

In this method, the pixels which have a given value range adjacented are recongnized as a part of a cluster, and labeled the pixel value from 1 in decreasing order of the number of the pixels belonging the cluster. The pixel value of the portion outside the object is set to 0. This procedure is called cluster labeling, and is used for the indentifying the object inside the CT data.

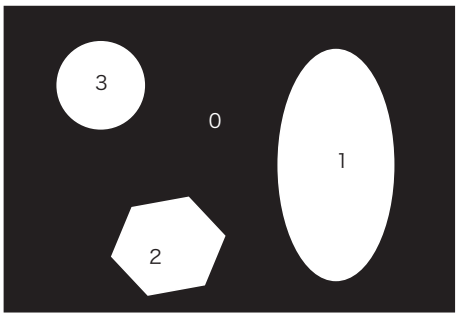


Figure 7.3: A conceptual diagram for the cluster labeling : The numbers in the diagram show the pixel value.

Sample script

```
mkdir mcl
sliceMCL  BYTE - 65 80 mcl > mcllog.txt
```

1st line: make save directory

2nd line: adopt the cluster labeling to the objects that composed of pixels with the value 65-80.

Hole labeling

In this method, the objects are recognized as same manner of the cluster labering. The difference is that the labeling is adopted to the holes of the objects, not objects it self. In this case, all objects has same pixel value, 1, and holes have pixel value decreasing order of the number of the pixels belonging the hole from 2. The example image is created using bubble.lzh and color table cm.cm in slice/slice/etc/ directory in the sliceXXXXXX.tar.gz.

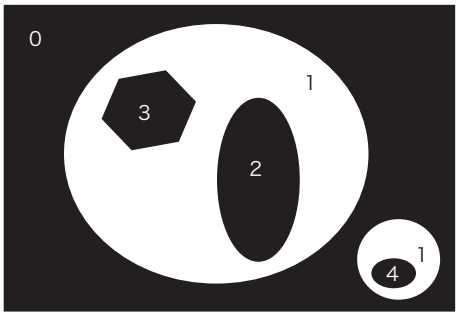


Figure 7.4: A conceptual diagram for the hole labeling : The numbers in the diagram show the pixel value.

Sample script

```
mkdir mhl
sliceMHL  BYTE - 100 255 mhl > mhllog.txt
mkdir pvr
echo 256 65535 0 | slicePVR mhl - pvr > /dev/null
mkdir cma
slice.CMA pvr cm.cm cma
```

1st line: make save directory
2nd line: adopt hole labeling to the holes in the objects that have pixels with the value=65-80
3rd line: make save directory
4th line: give 0 to the pixels smaller than 256 (holes with pixel value 256-65535) by slicePVR
5th line: make save directory
6th line: put colors to the images by slice.CMA introduced in §5.

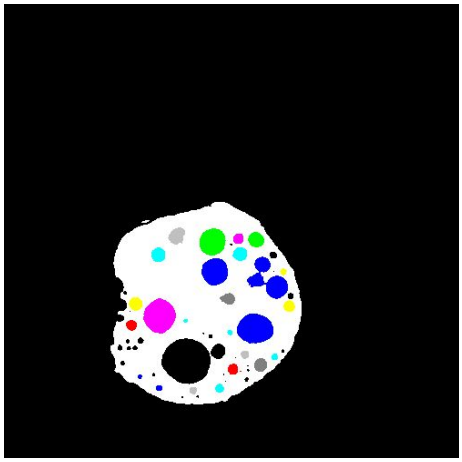


Figure 7.5: Example 200.tif

8 Volume rendering with Polygon

The volume rendering with polygon model is convenient for the analysis of binarized image data. In this section, we show the polygon modeling of 3D data with stl format. sliceXXXXX.tar.gz and stl.tar.gz should be installed for the image processings of this section.

8.1 Make birdeye's view image with polygon

make bird eye's view image with polygon. There are two ways: directory render from slice image data and create and use stl format file. If you want to make stl files, you should use stl.tar.gz.

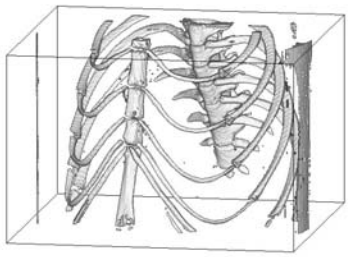


Figure 8.1: Example

8.1.1 Direct rendering from slice data (si_s_bev, si_m_bev)

Gray scale image(si_s_bev)

Sample script
`echo 260 190 | si_s_bev BYTE - 90-255 1 1 1 1 64 255 0 mouse.gif`

1st line: make gray scale image of the object which have pixel value 90<PV<255 with lon and lat of viewpoint = (260, 190)

Color image (si_m_bev)

Sample script
`si_m_bev BYTE - color.tbl 1 1 1 1 64 255 255 255 0 0 0 mouse_C.gif (Enter)` `260 190 (Enter, Control-d)`

1st line: make color image by si_m_bev using color table

2nd line: lon and lat of viewpoint = (260, 190). End of the input

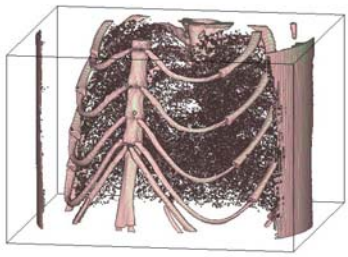


Figure 8.2: Example

color.tbl：
prepare a text file with text editor with following lines

0-50
51-89 255 200 200
90-255 200 255 200

1st line: pixels with PV=0-50 are not body
2nd line: set color of pixels with PV=51-89 as RGB=(255 200 200)
3rd line: set color of pixels with PV=90-255 as RGB=(200 255 200)

saved images by sample script: mouse.gif, mouse_C.gif

8.2 Make stl polygon data file(si_stl_B, si_stl_C：stl.tar.gz is required)

Make stl-format file using stl.tar.gz. The data is available other programs which can read the stl file.

Make STL- file (si_stl_B)

Sample script

```
si_stl_B BYTE - 90 255 mouse.stl
500     500     328
388037  13      13      0       487     375     327
659390  774456
```

1st line: make polygon datafile of the object that composed of pixels with value 90<PV<255 by si_stl_B
2nd line: pixel number of x, y, z direction of the data (output text)
3rd line: pixel number of the object and its region (output text)
4th line: pixel number of the object surface and number of triagles in the polygon surface (output text)

make color stl file (si_stl_C)

Sample script

```
si_stl_C BYTE - mouse_C.stl (Enter)
500     500     328
0 50 (Enter)
51 89 255 200 200 (Enter)
90 255 200 255 200 (Enter、Control-d)
921156  12      12      0       488     407     327
1233694 1516171
```

1st line: make polygon datafile mouse_C.stl by si_stl_C

2nd line: pixels that have pixel value=0-50 are not body

3rd line: set color of pixels with value=51-89 as RGB=(255 200 200)

4th line: set color of pixels with value=90-255 as RGB=(200 255 200)

5th line: pixel number of the object and its region (output text)

6th line: pixel number of the object surface and number of triagles in the polygon surface (output text)

supplement：
For the initial setting, the stl series program could not read 16 bit image files. In order to read 16 bit data file, re-compile the stl program (see stl program document in stl.tar.gz).

saved images by sample script: mouse.stl, mouse_C.stl

8.3 Make polygon bird eye's view image from stl files(stl_bev_SS, stl_bev_C_SS)

make gray scale image(stl_bev_SS)

Sample script

```
echo 260 190 260_190.tif | stl_bev_SS mouse.stl 1 1 16 255 0
```

1st line: make bird eye's view image using stl_bev_SS from mouse.stl from the viewpoint with lon and lat= (260,190)

make color image (stl_bev_C_SS)

Sample script

```
stl_bev_C_SS mouse_C.stl 1 1 16 255 255 255 0 0 0 0 0 0 (Enter)
0 0 X0_0_C.tif (Enter)
260 190 260_190_C.tif (Enter、Control-d)
```

1st line: stl_bev_C_SS from mouse_C.stl

2nd line: make image from the viewpoint with lon and lat = (0, 0)
3rd line: make image from the viewpoint with lon and lat = (260, 190)

supplement :
For the initial setting, the stl series program could not read 16 bit image files. In order to read 16 bit data file, re-compile the stl program (see stl program document in stl.tar.gz).
Because the coordinate of the 3D data is different from general coordinate, the bird eye's view image would be rotated. See §8.5.

saved images by sample script: 260_190.tif, X0_0_C.tif, 260_190_C.tif

8.4 Make polygon movie

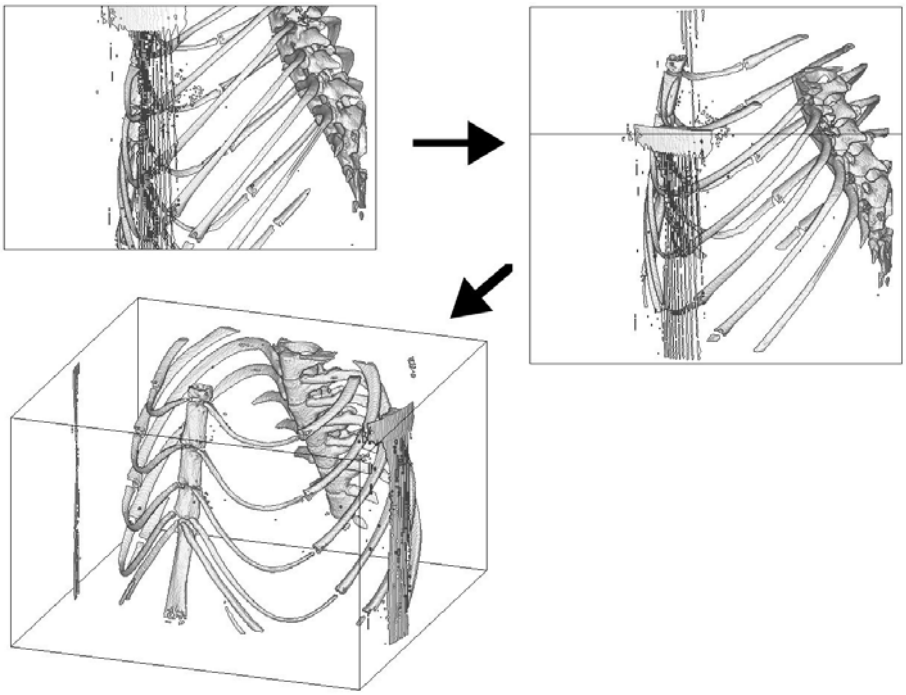


Figure 8.3: Example

8.4.1 Render from slice images (si_s_bev, si_m_bev)

Make gray scale movie (si_s_bev)

Sample script

```
si_s_bev BYTE - 100-255 1 1 1 1 64 255 0 mouse.gif(Enter)
180 180 0 10 2(Enter)
180 200 10 0 8(Enter, Control-d)
```

1st line: make polygon image of the object that composed of pixels twith the value 100<PV<255 by si_s_bev

2nd line: make movie from the viewpoint with lon and lat = (180, 180) and 2 times moves by (0, 10) degree

3rd line: add movie from the viewpoint with lon and lat = (180, 200) and 8 times moves by (10, 0) degree

Make color movie file(si_m_bev)

Sample script

```
si_m_bev BYTE - color.tbl 1 1 1 1 64 255 255 255 0 0 0 mouse_C.gif (Enter)
180 180 0 10 2(Enter)
180 200 10 0 8(Enter, Control-d)
```

1st line: make color polygon movie by si_m_bev

2nd line: make movie from the viewpoint with lon and lat = (180, 180) and 2 times moves by (0, 10) degree

3rd line: add movie from the viewpoint with lon and lat = (180, 200) and 8 times moves by (10, 0) degree

color.tbl：
prepare a text file with text editor with following lines

0-50
51-89 255 200 200
90-255 200 255 200

1st line: pixels that have pixel value=0-50 are not body

2nd line: set color of pixels that have pixel value=51-89 as RGB=(255 200 200)

3rd line: set color of pixels that have pixel value=90-255 as RGB=(200 255 200)

saved images by sample script: mouse.gif, mouse_C.gif

8.4.2 Make movie from STL file (stl_bev_GIF_SS, stl_bev_GIF_C_SS：stl.tar.gz が必要)

Make gray scale movie (stl_bev_GIF_SS)

Sample script

```
( echo 180 180 0 10 2 ; echo 180 200 10 0 8 ) | stl_bev_GIF_SS mouse.stl 1 1 16 255
0 mouse_move.gif
```

1st line: make polygon image by stl_bev_GIF_C_SS from mouse.stl from the viewpoint with lon and lat = (180, 180) and 2 times moves by (0, 10) degree, next 8 times moves by (10, 0) degree from (180, 200).

Make color movie (stl_bev_GIF_C_SS)

Sample script

```
stl_bev_GIF_C_SS mouse_C.stl 1 1 16 255 255 255 0 0 0 0 0 0  mouse_move_C.gif(Enter)
180 180 0 10 2(Enter)
180 200 10 0 8(Enter, Control-d)
```

1st line: make polygon image by stl_bev_GIF_C_SS from mouse_C.stl

2nd line: make movie from the viewpoint with lon and lat = (180, 180) and 2 times moves by (0, 10) degree

3rd line: add movie from the viewpoint with lon and lat = (180, 200) and 8 times moves by (10, 0) degree

supplement :
Because the coordinate of the 3D data is different from general coordinate, the bird eye's view image would be rotated. See §8.5.

saved images by sample script: mouse_move.gif, mouse_move_C.gif

8.5 Angle

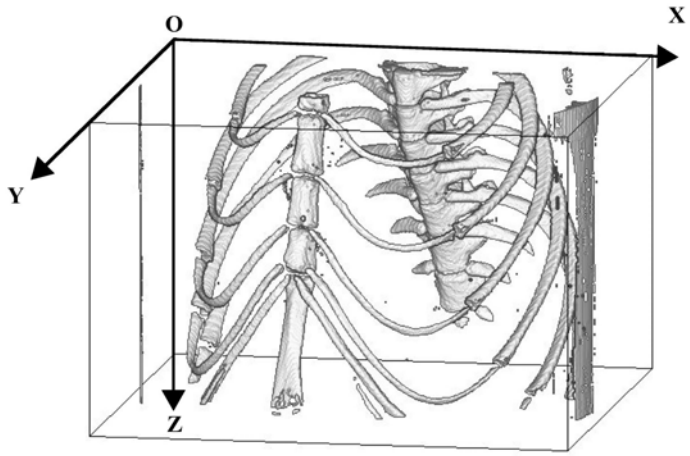
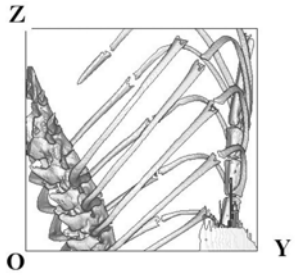
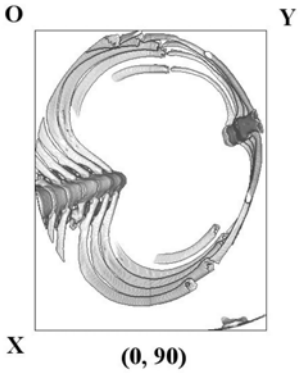


Figure 8.4: coordinate

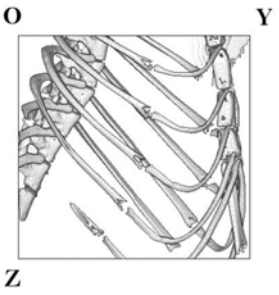
When a polygon data (Fig. 8.4) is rotated a certain degree (lon, lat) , following results are obtained.



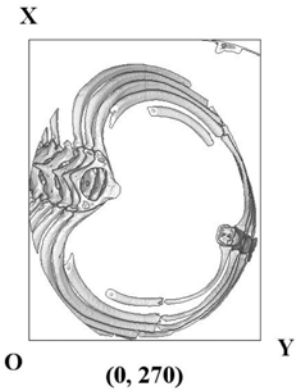
(0, 0)



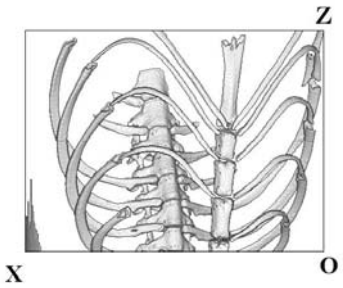
(0, 90)



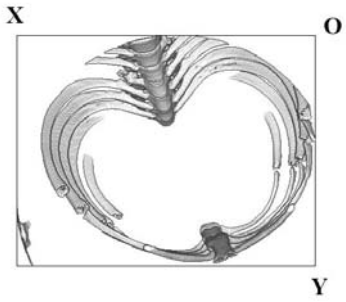
(0, 180)



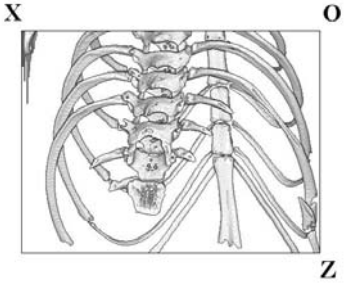
(0, 270)



(90, 0)

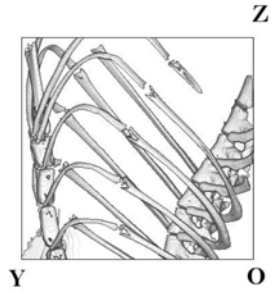


(90, 90)

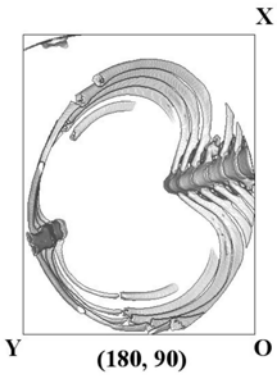


(90, 180)

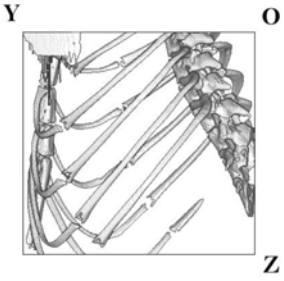
(90, 270)



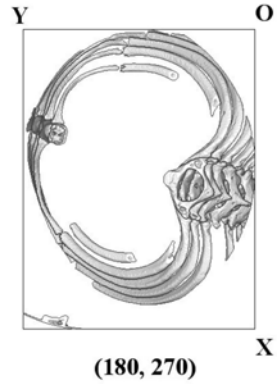
(180, 0)



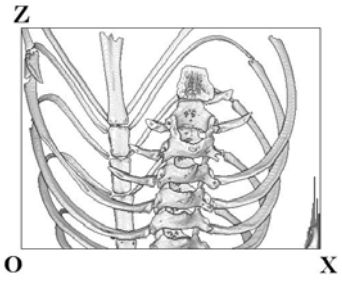
(180, 90)



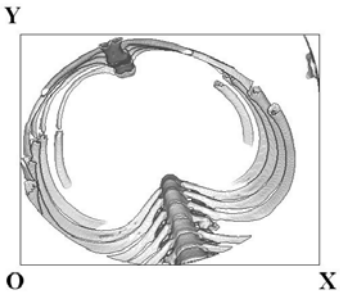
(180, 180)



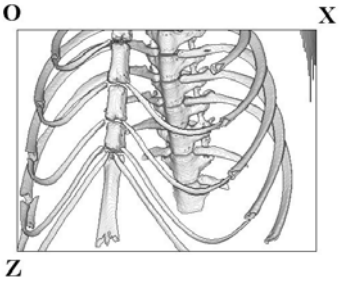
(180, 270)



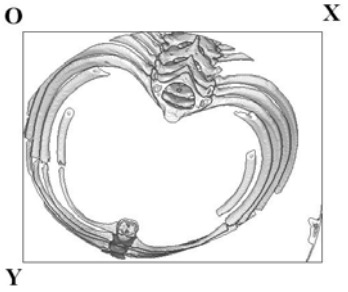
(270, 0)



(270, 90)



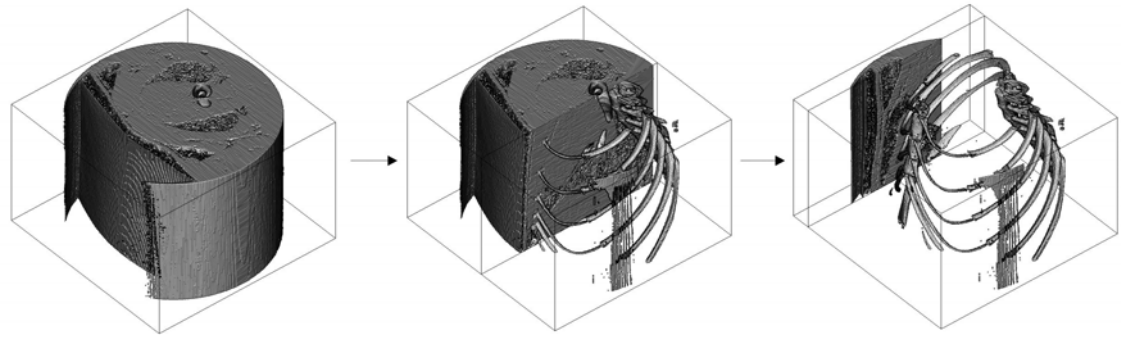
(270, 180)



(270, 270)

8.6 Incorporating polygon data

Incorporate two polygon images with a certain surface, and make a transition movie



sample script

```
echo 500 220 220 -50 0 0 10 | si_x_bev BYTE - color.tbl  (do not break)
BYTE - color2.tbl 1 64 255 255 255 0 0 0 move.gif
```

1st line: Make bird eye's view image of polygon by si_x_bev from data in BYTE directory. This program makes two polygon data from color.tbl and color2.tbl and incorpolate them at y-z plane which contains x=500 point, and output bird eye's view image from the viewpoint with lon and lat =(220, 220) . The y-z plane is moved from 500 by -50 pixels10 times, and the images are added to transition movie in gif format.

color table

prepare text files with text editor with following lines

color.tbl :
0-19
20-255 255

color2.tbl :
0-89
90-255 255

supplement :
In this example, we use same 3D data in BYTE, but different data can be used in the script. It also possible to make color polygon data by si_x_bev and incopolate them (by editing color table file). If the second 3D data (BYTE - color2.tbl) is (- - /dev/null), cross section image of the bird eye's view image could be obtained.

saved images by sample script: move.gif

9 Image processing of binarized image

The quantitative analysis of 3D data is possible by using binarized image. In this section, we show the examples of image processings which would be required before the quantitative analysis.
In this section, we also use another sample data (bubble.lzh).

9.1 Hole filling (sliceMHL, sliceDE, slicePVR)

The image processing to fill the holes in object is sometime required in order to estimate the volume of a object in CT data.

9.1.1 sliceDE

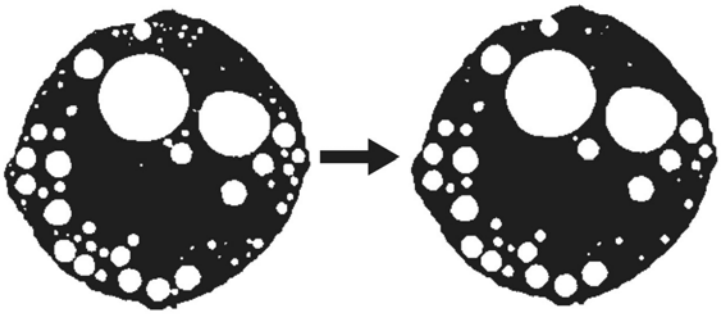


Figure 9.1: Example 152.tif

Sample script

```
mkdir de
sliceDE BYTE - 140 255 3 de
```

1st line: make save directory

2nd line: adopt dilation and erosion in three pixels to the objects that have pixel value=140-255 by sliceDE, and save images in de

supplement：
In order to fill the holes, dilation (increase the pixels of object to outside) and erosion (reduce the pixels of the object to inside) effect is adopted on the 3D data. By this processing, holes ad gaps which have radius smaller than 6 pixels are filled. Through this process, the pixel value of the object is set to 1.
By using sliceED, the small object can be erased.

saved images by sample script:
de/000.tif∼297.tif

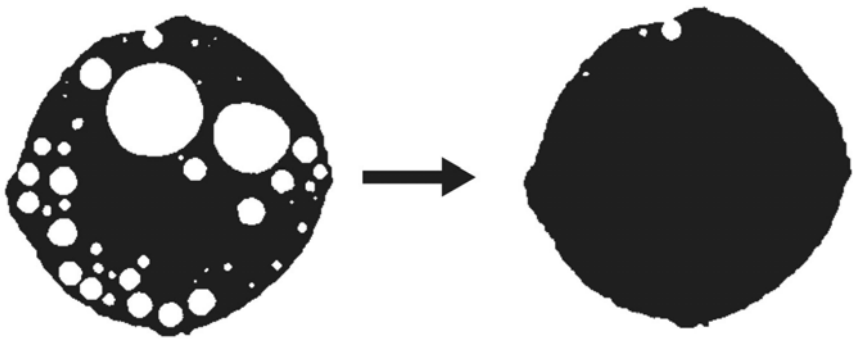


Figure 9.2: Example 152.TIF

9.1.2 sliceMHL, slicePVR

Sample script

```
mkdir mhl
sliceMHL BYTE - 140 255 mhl
mkdir mhlpvr
echo 2 65535 1 | slicePVR mhl - mhlpvr
```

1st line: make save directory

2nd line: adopt the hole labeling to the holes in the object that are composed of pixels with the value=140-255 by sliceMHL

3rd line: make save directory

4th line: replace pixel value 2-65535 to 1 by slicePVR

supplement：
sliceMHL labels values larger than 1 to the pixels that are in the hole of the object. So by replacing the pixel values 2 to 65535 to 1 (the pixel value of the object) after the sliceMHL by slicePVR, we can fill the hole in the objects. However, if a hole connects to the outside of the object, the hole could not be identified. In this processing, the deformation of the hole and object does not appears in the images, which appears for the case of sliceDE or ED. Through the process, the pixel value of the object is replaced to 1.

saved images by sample script:
mhl/000.tif∼297.tif
mhlpvr/000.tif∼297.tif

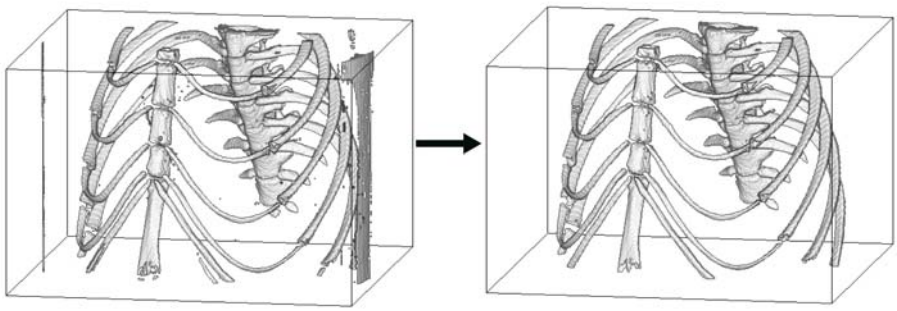


Figure 9.3: Example

9.2 Noise reduction (slicePVR)

Sample script

```
mkdir mcl
sliceMCL BYTE - 90 255 mcl > /dev/null
mkdir mclpvr
(echo 22 22 0 ; echo 44 65535 0 ; echo 2 43 1) | slicePVR mcl - mclpvr > /dev/null
echo 260 190 | si_s_bev BYTE - 1 1 1 1 1 64 255 0 mouse.gif
```

1st line: make save directory

2st line: adopt claster labeling to the object that are composed of pixels with the value=90-255 by sliceMCL

3st line: make save directory

4st line: replace pixel value 22 and 44-65535 to 0, and other pixels to 1 by slicePVR

5st line: make bird eye's view image with lon and lat of viewpoint = (30,20) by si_s_bev

supplement :
sliceMCL labels values larger than 0 to the pixels that composed of the objects. The largest cluster has pixel value = 1 and small objects have pixel value = 65535. Small objects generally appear as noise in the 3D image. In this case, by replacing 22 (which is large noise near the corner) and 44-65535 to 0, we can erase the noises in the data.

saved images by sample script:
mcl/000.tif∼327.tif
mclpvr/000.tif∼327.tif
mouse.gif

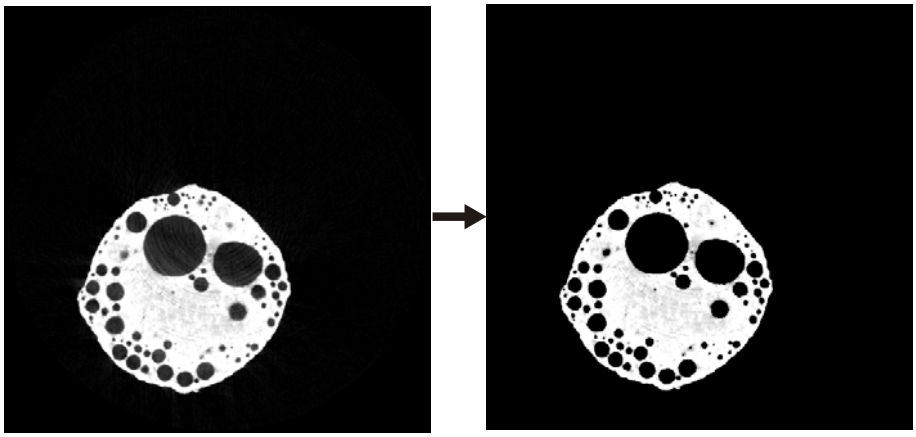


Figure 9.4: Example 152.tif

9.3 Mask(slicePVM)

Sample script

```
mkdir mhl
sliceMHL BYTE - 140 255 mhl > /dev/null
mkdir pvr
echo 2 65535 0 | slicePVR mhl - pvr > /dev/null
mkdir pvm
slicePVM pvr - 0 BYTE pvm
```

1st line: make save directory

2st line: adopt hole labeling to the holes in the object that have PV=140-255 by sliceMHL

3st line: make save directory

4st line: replace pixel value 2-65535 to 1 by slicePVR

5st line: make save directory

6st line: mask images in the BYTE directory using binarized slice images in the pvr directory, and save in the pvm directory

supplement :
In the sample script, the area of the object appears in the binarized image data (pvr) are selected from the original data (BYTE) and saved in the new directory (mhlpvr). Then the noise appears outside of the object (or inside the holes) in the original data can be erased in the new image data.

saved images by sample script:
pvm/000.tif∼297.tif
pvr/000.tif∼297.tif
mhl/000.tif∼297.tif

9.4 Transparent ovarlay(si_cim)

The result of hole filling processing (§9.1) is confirmed by transparent overlaying on the original data.

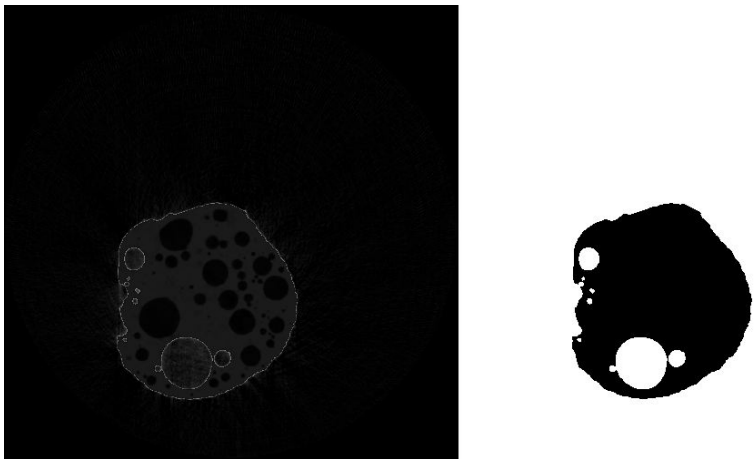


Figure 9.5: Exsample

Sample script
`mkdir cim` `si_cim BYTE - mhlpvr - 0.1 cim`

1st line: make save directory

2st line: using si_cim, image data in the mhlpvr directory is overlayed on the original data in the BYTE directory for factor 0.1, and the new image data is saved in the cim directory

supplement：
Right hand side of the sample image is hole filled image and left hand side of the sample image is the image overlayed on the original data in the BYTE directory. The holes did not filled by sliceMHL show white embedding lines around the hole because of original data. The si_cim can read and save color images.

saved images by sample script:
cim/000.tif∼297.tif

10 Quantitative analysis of binarized images

In this section, the examples of quantitative analysis of the CT data are shown in some aspects. We use bubble.lzh for the sample data in the sample script.

10.1 Evaluation of porosity (sliceMHL, slicePVR)

Sample script

```
sliceMHL BYTE - 140 255 mhl > /dev/null
(echo 2 65535 255) | slicePVR mhl -  pvr > /dev/null
echo '' | slicePVR pvr -  > pvr.txt
```

1st line: adopt hole labeling to the objects that are composed of pixels with value=140-255 by sliceMHL

2nd line: replace pixel value 2-65535 to 1 by slicePVR

3rd line: make histogram data by slicePVR and save it as pvr.txt

supplement :
pvr.txt
0 0 67954139
1 1 4515889
255 255 2029972

The object has pixel value=1 and the holes have pixel value=255. The whole volume of the object is (number of pixel value=1) + (number of pixel value=255),and porosity is (whole volume)/(number of pixel value=255). If a number of holes connected to the outside of the objects, their volume could not be included. In such case, another processing before making histogram, such as sliceDE, may be requried.

saved data by sample script: pvr.txt

10.2 Ellipsoidal approximation of binarized object (sliceOF, of_stl_ih)

Sample script

```
mkdir mhl
sliceMHL BYTE - 140 255 mhl > /dev/null
mkdir pvr
(echo 3 65535 0 ; echo 1 1 0) | slicePVR mhl - bb_pvr > /dev/null
sliceOF bb_pvr - 2 2 2 > of.txt
```

1st line: make save directory

2nd line: adopt hole labeling to the holes in the object that have pixel value=140-255 by sliceMHL

3rd line: make save directory

4th line: replace pixel value 3-65535 and 1 to 0 by slicePVR

5th line: adopt ellipsoidal approximation and save the date in of.txt. The scale of each axis is 2.

supplement :
of.txt
0 74129789 249.731 249.219 148.586 499.462 498.437 297.172 35.144 0.00616071 90.0343 385.557 646.646 647.054 675747719.388186
2 370211 203.27 305.823 131.288 406.539 611.646 262.576 -100.378 33.7734 44.0446 70.8966 75.1825 167.559 3741083.083801

1st line in the of.txt is the information of background and following lines show the information of the objects. The data format is number of cluster, number of pixel, x,y,z coordinate of the gravity center of the object (pixel), x,y,z coordinate of the gravity center of the object (scaled), angle of each axis of the approximated ellipsoid λ,ϕ,θ (degree), the length of each axis (scaled), the volume of the approximated ellipsoid.

The data of the approximated ellipsoids obtained by sliceOF can be converted into the stl format polygon data, and bird eye's view.

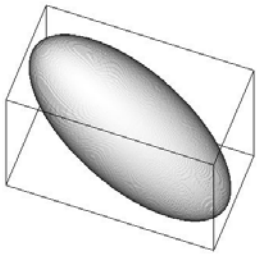


Figure 10.1: ellipsoidal approximation image of a bubble

Sample script

```
tail -n+2 of.txt | cut -f3-5,9-14 | of_stl_ih 7 of.stl
echo 20 60 20_60.tif | stl_bev_SS of.stl 1 1 16 255 0
```

1st line: make stl file named of.stl from the of.txt using cut and tail command in UNIX by of_stl_ih
2nd line: make bird eye's view image from the of.stl file by stl_bev_SS

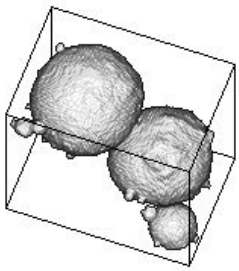


Figure 10.2: shape of the original bubble (Try to reproduce this image as the practice !)

saved images by sample script: of.txt, of.stl, 20-60.tif