The 3D image processing tools

SLICE

JASRI/SPring-8

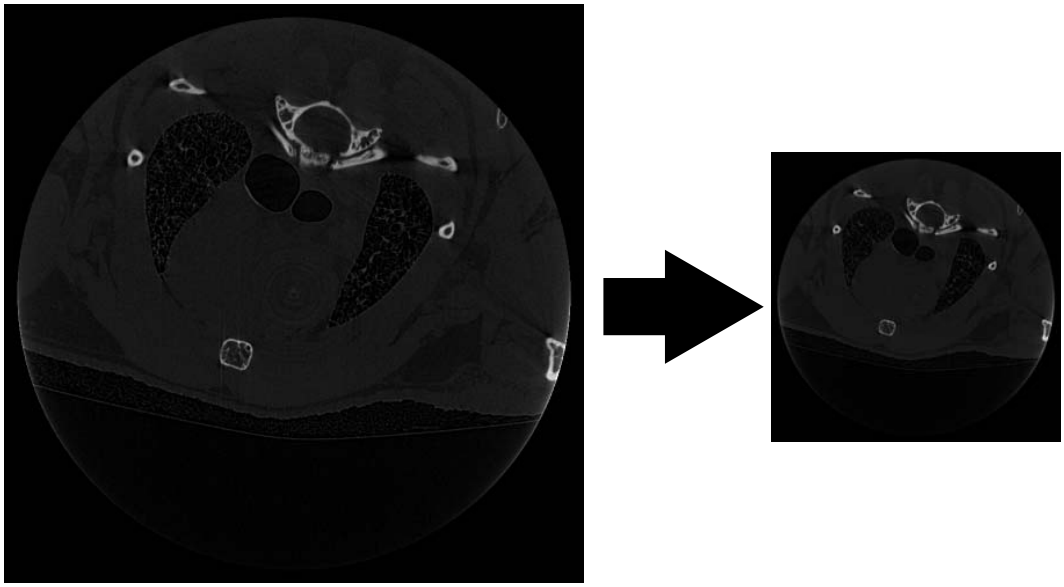


Figure 0.1: スライス画像の拡大、縮小（§4.1）

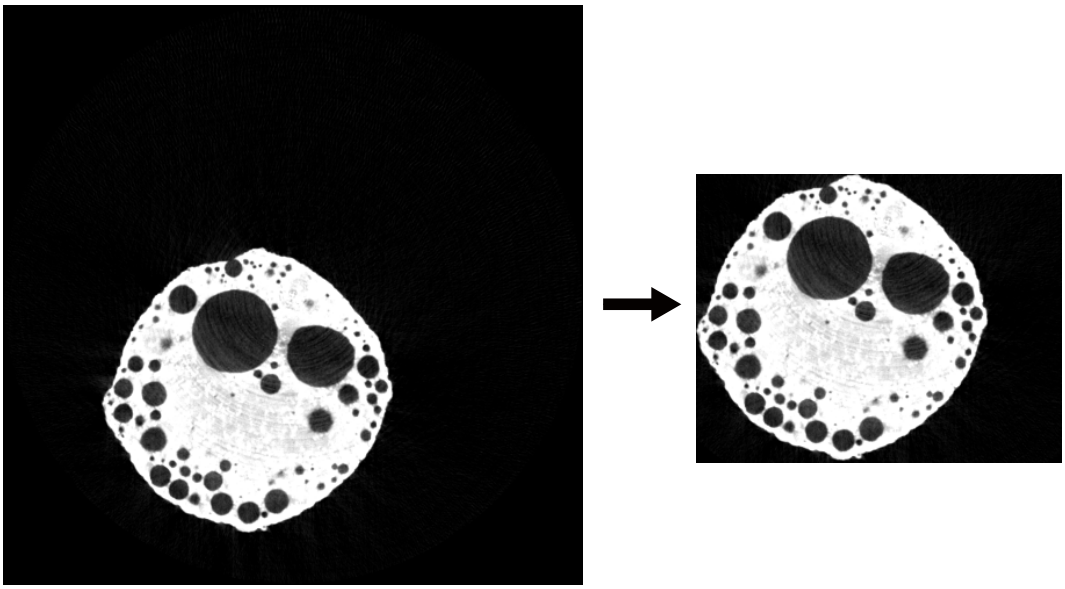


Figure 0.2: スライス画像のトリミング（§4.2）

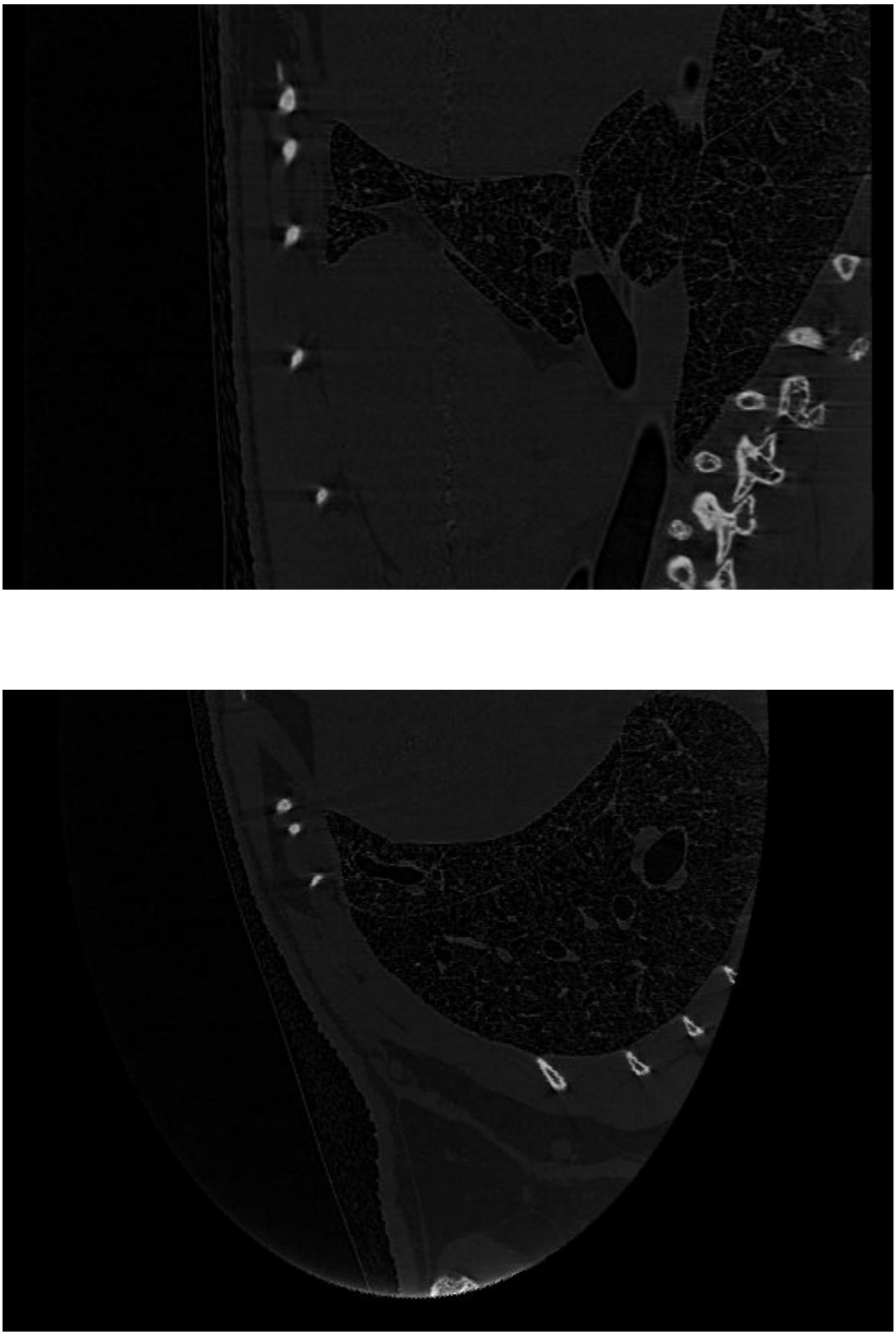


Figure 0.3: スライス画像の reslicing（§5）

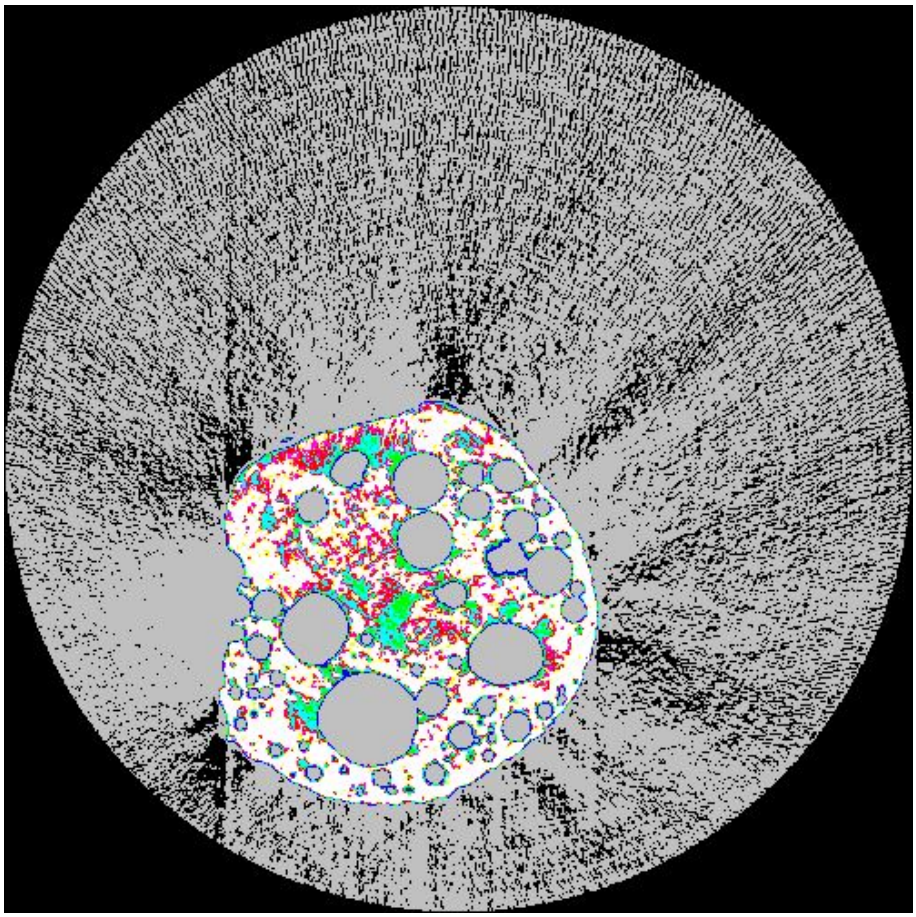


Figure 0.4: スライス画像のカラー化（§5）

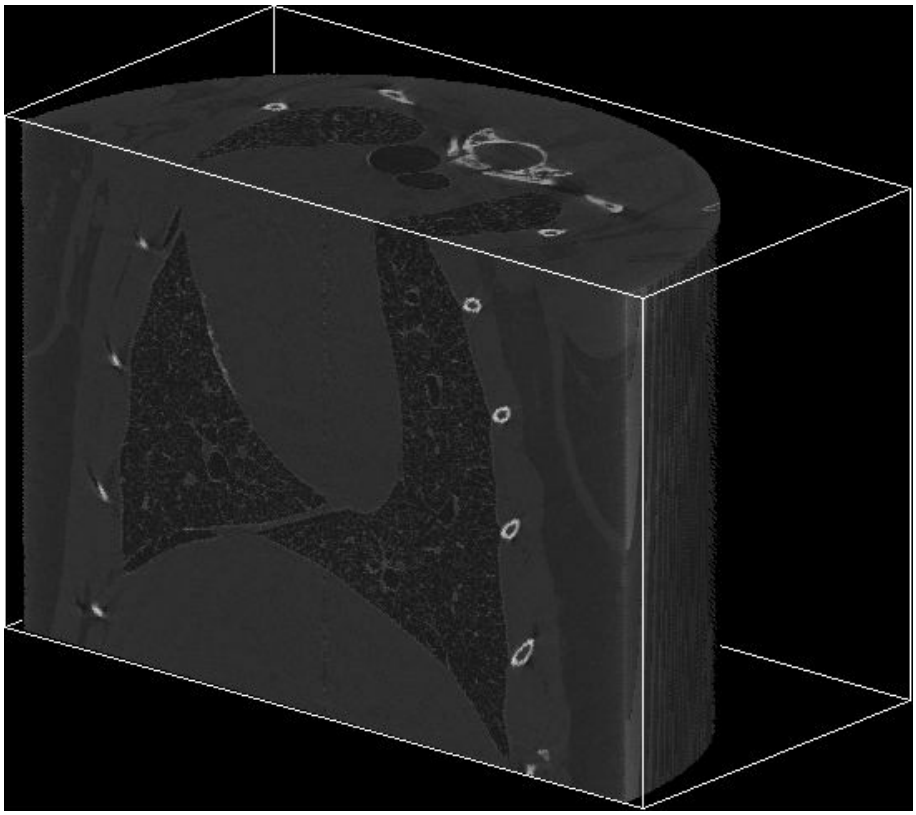


Figure 0.5: ボリュームレンダリング（§6.1）

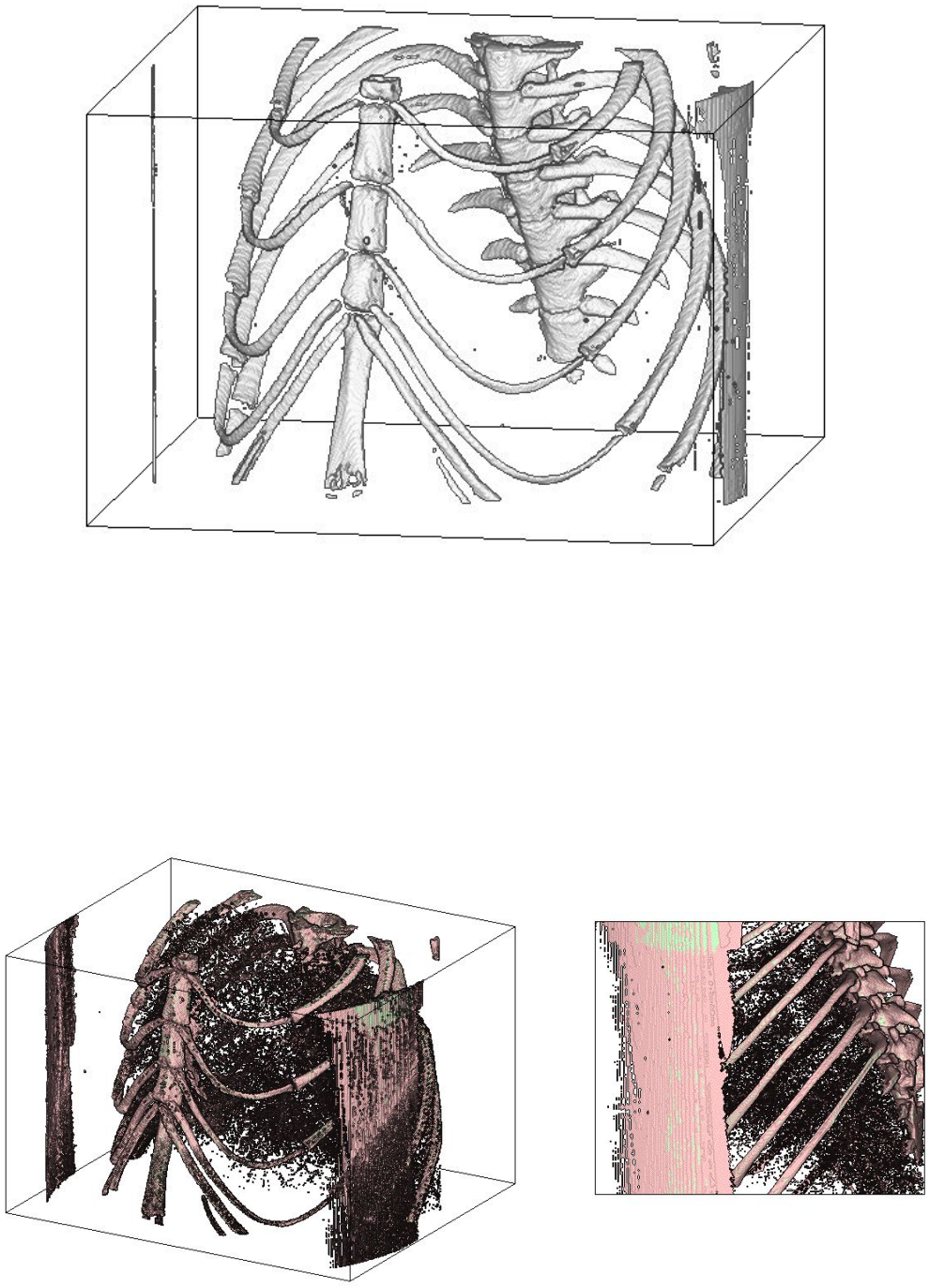


Figure 0.6: ポリゴンを用いたボリュームレンダリング（§8.1）

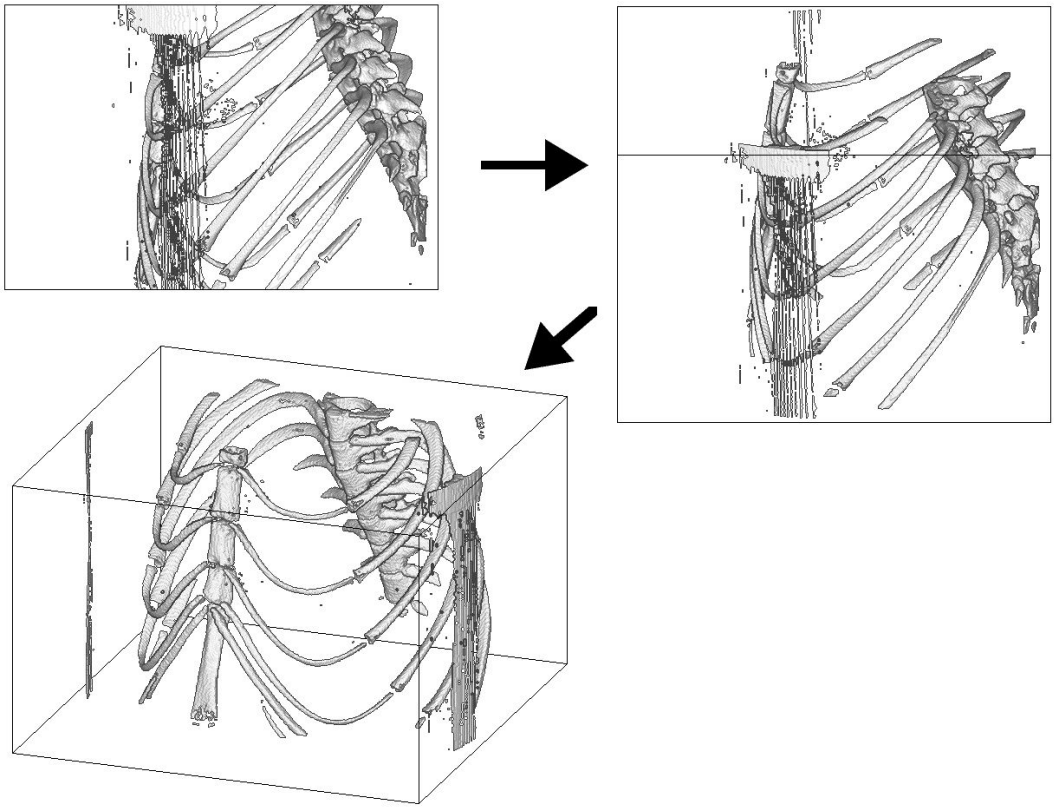


Figure 0.7: 立体画像の動画作成（§8.3）

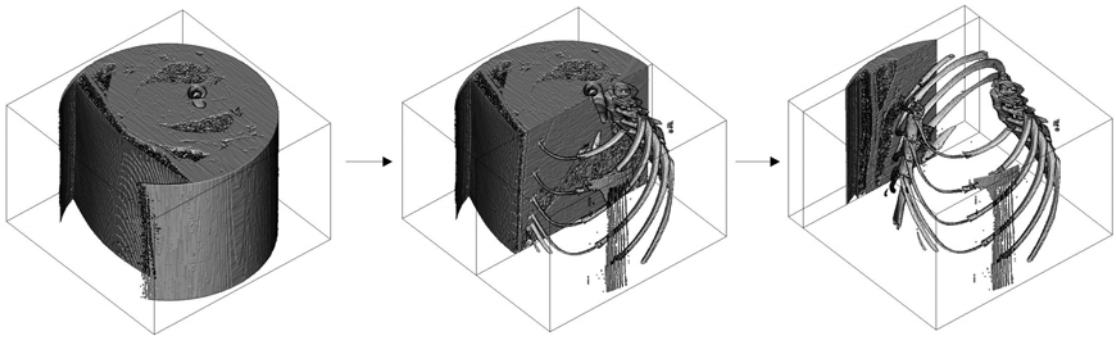


Figure 0.8: 二つのポリゴン画像をある断面で接合した画像、遷移動画（§8.5）

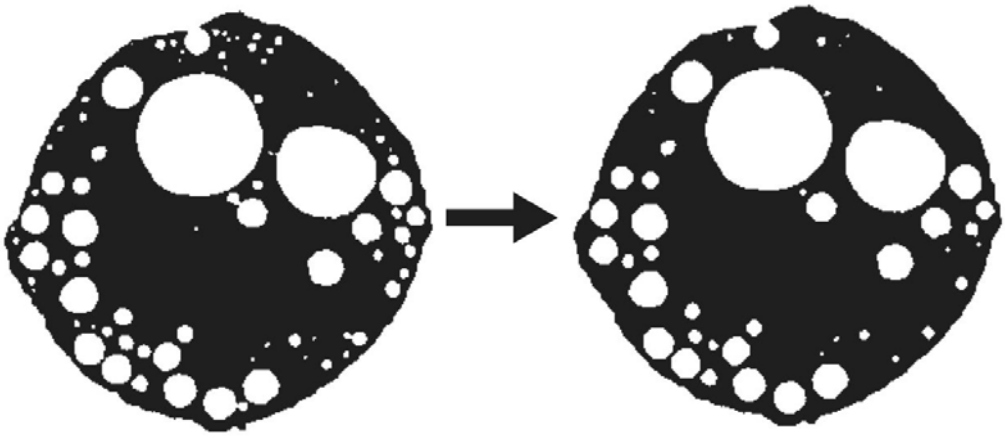


Figure 0.9: 二値化データの穴埋め（§9.1）

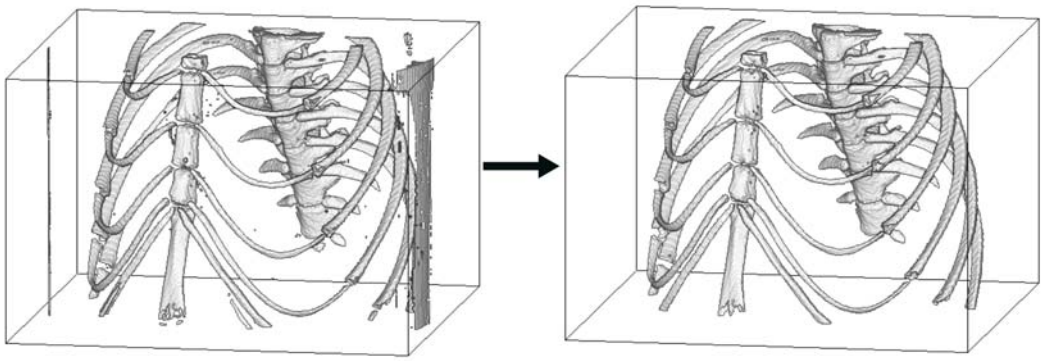


Figure 0.10: 二値化データの不要な部分の除去（§9.2）

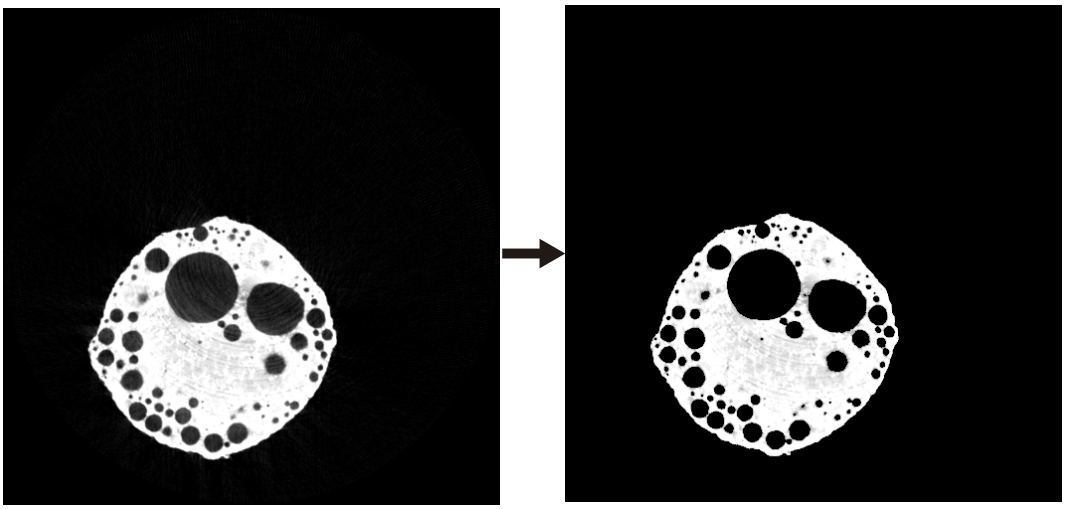


Figure 0.11: 二値化データを用いて元データのノイズ除去、部分抽出（§9.3）

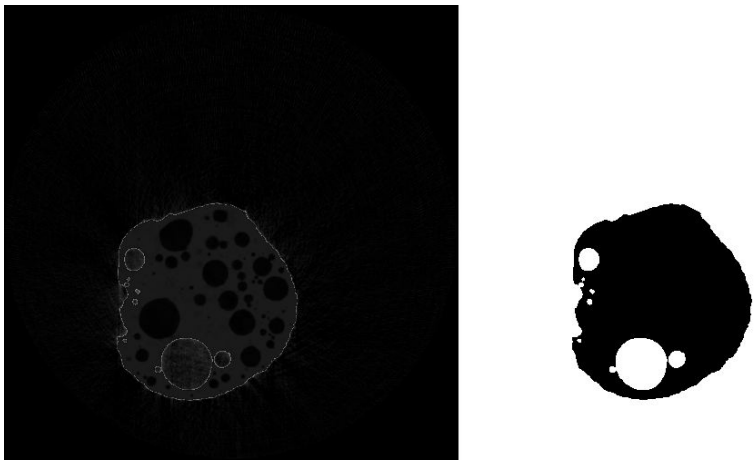


Figure 0.12: 透過重ね合わせ（§9.4）

Figure 0.13: 三軸不等楕円体近似した気泡のポリゴンデータ （§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 Slice とは

Slice シリーズは中野司氏によって作成された、コマンドラインから実行するよう設計されたプログラムで、これを用いて X 線 CT 画像から 3 次元立体図の作成、データ解析、加工などができる。
吸収 X 線 CT 装置は、物体を通過した X 線が、通過の際に物体に吸収された量を透過光と透過前の光の比をとることで調べ、その値を元に物体の内部構造を調べる装置である。サンプルを回転させることで得られた複数の透過光の画像は、サンプルの回転軸に対して垂直な 1 ピクセルの厚さの面を 1 枚の画像（スライス）として、透過光画像の縦のピクセルの枚数分の画像（スライス）からなる 3 次元データに再構成される。このとき各スライス画像は縦、横とも透過光の画像の横のピクセル数と同じ数のピクセル数となる。

Slice プログラムの大きな特徴は 2 次元画像群で構成される 3 次元画像データを一度に処理できる点にある。本プログラムを使用する際は、3 次元試料のスライス画像である各画像ファイルをいちいち個別に処理する必要は無く、また erosion, dilation やクラスタラベリングといった処理は、2 次元の各スライス画像上で行うのではなく、物体の 3 次元構造を反映した処理を行う。

1. コマンドライン上で動くプログラムであるため、バッチ処理を容易に行うことができる。
2. UNIX, Windows, Macintosh の各 OS 上で動作する。

といった利点がある。本書は X 線 CT 装置によって撮影された画像データを、この Slice シリーズを用いて利用するための参考文献である。

1.2 本書について

本書は大きく 2 つのパートからなっており、まずこの Slice を利用して作成できる事例を作成された画像やモデル像を使って紹介し、これを作成するためのコマンドの羅列（スクリプト）をサンプルとして紹介する、クックブック、それから各プログラムに関して必要な情報をすべて記載したプログラムリファレンスである。クックブックのパートではわかりやすく説明することを第一としており、詳細なプログラムの説明、使い方や仕様については極力省略している。この部分でプログラムの使い方を大体把握した後、さらに情報が必要な場合はプログラムリファレンスのパートを参照して欲しい。

また、クックブックのパートではサンプルスクリプトを通して Slice シリーズのプログラムの使用法を説明しているが、これらのサンプルスクリプトは Slice シリーズに含まれるすべてのプログラムを網羅してはいない。プログラムリファレンスの中で似たような作業ができる物を関連プログラムとして記載しているので、サンプルスクリプトで説明されている以外のプログラムについては、これを元にそれぞれのリファレンスを参照して欲しい。

1.3 引用の方法について

本プログラムを用いて作画し、その旨を明示する場合、以下の例を用いると良い。

＊引用例

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).

中野司, 土 ‘山明, 上杉健太郎, 上椙真之, 篠原邦夫（2006） Slice -Softwares for basic 3-D analysis-, Slice Home Page (web), http://www-bl20.spring8.or.jp/slice/， 財団法人　高輝度光科学研究センター.

2 Getting Started

2.1 プログラムのインストール

Slice は Winsows のコマンドプロンプトや、UNIX (Liunx, FreeBSD, MacOSX 等) のようなプログラムをコマンドラインで実行する環境で使用することができる。ただし、Windows で本書のサンプルスクリプトを用いる場合 Cygwin と呼ばれるツールをインストールする必要がある場合がある。Cygwin のダウンロード、インストールの方法等については以下の URL を参照して欲しい。
http://sohda.net/cygwin/

○プログラムダウンロード・解凍

Slice のインストール方法を説明する。まず下記の URL からプログラムをダウンロードする。**サンプルデータはサイズが大きいので注意**。なお、本書では slice.tar.gz と stl.tar.gz の二つを用いており、本書の範囲内の作業を行うのであれば他のプログラムは基本的に不要である。本書で取り扱ったサンプルデータは同様にダウロードできるようにしている

本体 (XXXXXX は日付が入る)
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 画像モザイク
http://www-bl20.spring8.or.jp/slice/file/rmsd.tar (1.6MB)

傾き補正
http://www-bl20.spring8.or.jp/slice/file/irac.tar (730MB)

STL ファイル作成・操作プログラム（産総研）
http://www.gsj.jp/GDB/openfile/files/no0448/0448index.html (5MB)

練習のためのサンプルデータ
http://www-bl20.spring8.or.jp/slice/file/mouse.lzh (35MB)
http://www-bl20.spring8.or.jp/slice/file/bubble.lzh (28MB)

これらは圧縮されたファイルであり、これを適当な場所で展開するとディレクトリが作成される。

○ コンパイル・インストール（詳細はインストールガイドを参照）

Slice のインストール

Windows なら cygwin のコンソール画面で、UNIX, MacOSX ではターミナルの画面で解凍することによって出来た slice ディレクトリに移動し、

make install（リターン）

と入力することで bin に実行ファイルが作成される。**/usr/bin などのディレクトリではなく、sliceXXXXXX.tar.gz の解凍先のディレクトリ (slice/bin) にインストールされることに注意。このまま解凍した際に作成された**

slice フォルダを削除した場合、プログラムも一緒に削除される。これを回避するためには、各自で適切なディレクトリに移動させたあと、PATH を設定する必要がある。

以下、必要に応じて同様に stl, gi, rmsd, irac, 等も必要に応じてインストールする。

○ PATH の設定

Windows の場合：スタートメニュー→プログラム→アクセサリ→システムツール→システム情報→ツール→システム設定ユーティリティ→環境→ PATH を選択。上記のコンパイル、インストールでプログラムが作成された bin (or exe) フォルダの絶対パスを加える。

UNIX, MacOSX の場合：各自のホーム（ディレクトリ）下の .bash_profile、 .cshrc 等の設定ファイルにパスを設定する。(MacOS では. で始まるファイルは不可視であるため、mi などのそれが見えるソフトか、ユーティリティ：ターミナルから vi を用いて編集する。)

.bash_profile : 以下の行を編集する（：を忘れずに)。
PATH=$PATH:bin の絶対パス

.cshrc : 以下の行を加える。
set path = ($path bin の絶対パス)

2.2 サンプルスクリプトの実行方法

本文中にはサンプル画像を作成したスクリプトを掲載している。この実行方法について説明する。実際の CT データも同様にして解析することができる。

手順１：cygwin やコマンドプロンプトなどの端末画面上で、データファイルが保存してあるディレクトリの一つ上のディレクトリに移動する。例えばデータがハードディスク内部の/data/BYTE/に保存してある場合は/data/に移動する。移動は cd /data/（リターン）で移動できる。**今後の説明、サンプルスクリプトなどはこの、データファイルのおかれたディレクトリの一つ上のディレクトリで作業をすることを前提としていることに注意**。移動後、サンプルスクリプトの内容を一行ずつ打ち込む

手順２：このように Slice がインストールされた cygwin や UNIX 端末上でコマンドを逐次入力することで実行可能であるが、バッチファイルに書き込み、一気に走らせることも可能である。ただしその際、場合によっては GB 単位のデータが一気に大量に作成されるので、HDD の残り容量には十分に気をつける必要がある。

バッチファイルの作成、実行方法は各 OS によって異なる。

○ Windows の場合

ノートパッドなどのテキストエディタでスクリプトをファイルに書き込み、ファイル名に拡張子.bat をつけて保存する。これをコマンドプロンプト、もしくは Cygwin で実行する。**ただしこれらの端末画面上で実行する場合はファイルの保存、実行を上記ディレクトリにおいて行うこと。**

○ UNIX, MacOSX の場合

vi やテキストエディットを用いてスクリプトをファイルに書き込み、適当な名前を付けて上記ディレクトリに保存する。ターミナルの画面上で目的のディレクトリに移動し、csh ファイル名（リターン）と入力することで実行できる。

2.3 画像表示ソフト

各 OS に対応している 2006 年現在の主な画像表示ソフトは以下のような物があげられる。

Windows

- irfan view (フリーウェア: http://www8.plala.or.jp/kusutaku/iview/)
- ImageJ (フリーウェア: http://rsb.info.nih.gov/ij/)

MacOSX

・ Preview (OS 付属)

・ ImageJ (フリーウェア: http://rsb.info.nih.gov/ij/)

・ Graphic Converter (シェアウェア: http://www.bridge1.com/ grahpicconverter.html)

UNIX

・ ImageJ (フリーウェア: http://rsb.info.nih.gov/ij/)

・ Image magick

・ XV (以下の URL に HiPic で作成された.img ファイルを表示できる XV：中野司氏作、がアップロードされている; http://www-bl20.spring8.or.jp/slice/file/xv-hipic.tar.gz 2.6MB)

3 Fundamentals

3.1 画素値について

吸収 X 線 CT 装置は、物体を通過した X 線が、通過の際に物体に吸収された量を透過光と透過前の光の比をとることで調べ、その値を元に物体の内部構造を調べる装置である。CT 撮影で得られたスライス画像のそれぞれのピクセルのグレー値、“画素値”をこの吸収係数になおした値は、その地点における物質の X 線の線吸収係数の値 (Linear Absorption Coefficient : LAC 値などとよばれる。単位は cm^{-1}）と同一である (図参照)。

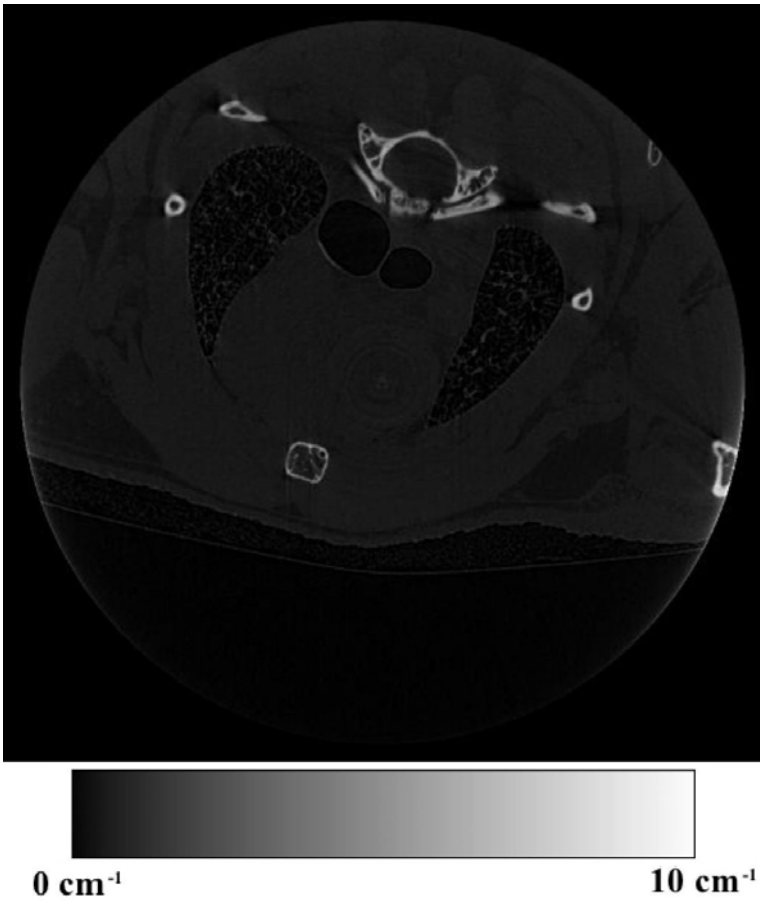


Figure 3.1: 画像のグレースケールと LAC 値の例

SPring-8 で取得された CT データの場合、再構成処理の際に tif_h2o(tif_h2h) 等でこの処理が行われていれば、各スライス画像のコントラストは与えられた LAC 値の範囲を 256 階調グレー (8bit: tif_h2o を用いた場合)、または 65536 階調グレー (16bit: tif_h2h を用いた場合) で表している。各スライス画像でコントラストがバラバラになっている等、この処理がなされていないと考えられる場合は場合は再構成マニュアルに従って撮影された画像の再処理をしなければならない。詳細についてはビームラインに設置されている画像再構成マニュアルを参照すること。それ以外のシステムで撮影された CT データを再構成した場合は、それぞれの装置に付随のマニュアルなどを参照しこの処理を行っておく必要がある。

3.2 CT データについて

再構成された CT の三次元画像データは、基本的に 000.tif（もしくは 0000.tif, ro000.tif, ro0000.tif など、一連の連番の名前を持つ画像群の中でもっとも番号の若い画像）の左上を原点（O）とした三次元座標を用い

て表される (図参照)。本書で紹介する Slice シリーズのソフトのうち、slice と名前の付いているプログラムでは、これらの画像群はいちいちファイル名を指定しなくとも、一つのディレクトリにあるファイル全てを一連の 3 次元データと見なし、扱うことが出来る。本書でもこの座標系に乗っ取って解説を行う。なお、プログラムによって座標変換が行われる場合があるが、そのようなプログラムについては §??の章末を参照。

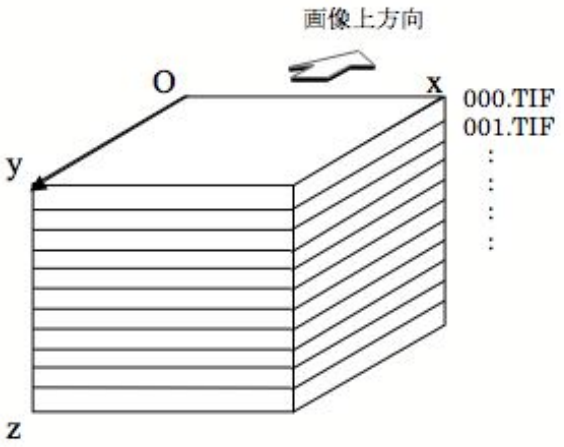


Figure 3.2: 座標系

3.3 サンプルデータ

本書で使用した二つのデータは練習用データとして Slice のオフィシャルサイトにダウロードできるように用意されている。mouse.lzh は ICR マウス、オスの胴体の CT 像で次のような撮影条件になっている。Energy 25keV, Pixel Size 16μm (reduced to 32μm by sliceBIC)。データは LAC から pixel value への変換は 0-10cm^{-1} to 0-255(8 bit) とし、000.TIF $\sim$ 327.TIF、500x500 pixel というフォーマットになっている。

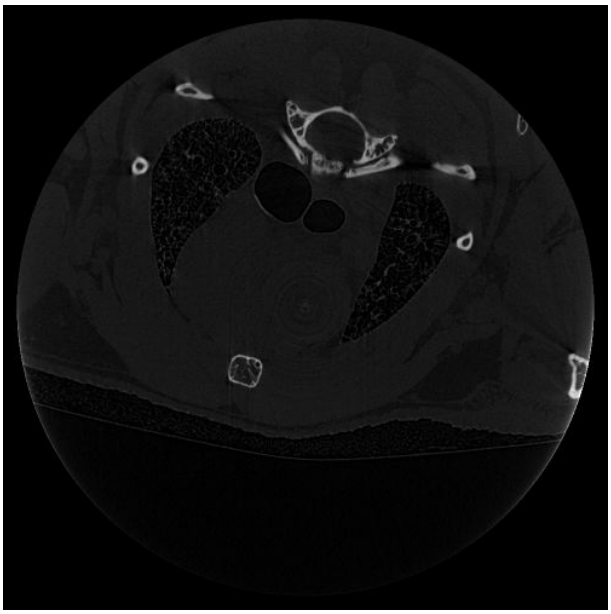


Figure 3.3: サンプルデータ例：000.TIF

bubble.lzh は加熱した球形の岩石の CT 画像で 000.TIF $\sim$ 297.TIF、500x500 pixel というフォーマットになっている。lha 等でサンプルデータを解凍すると、BYTE ディレクトリが作成されその内部に TIFF 形式

の画像ファイルが作成される。
本書のサンプルスクリプトはこのサンプルデータを展開したディレクトリでプログラムを実行していることを仮定している。データがない状態で練習をする際などに利用すると良い。なお、実際に Slice を用いて撮影されたデータを使用してサンプルスクリプトを実行場合は、ファイル名、ディレクトリ名などを適宜変更する必要がある。

4 三次元像の拡大、縮小、トリミング、切り出し

以降の章で説明するプログラムでメモリエラーが出る場合、本節の作業を通して画像データの容量を小さくすることで、このエラーを回避できる場合がある。なお、本章の作業を行うためには slice.tar.gz をインストールしておく必要がある。

4.1 スライス画像の拡大、縮小を行う (sliceBIC)

3 次元画像データを縮小する。これによりデータ全体のサイズ、容量を小さくすることが出来る。これらの作業では mouse.lzh のサンプルデータを用いている

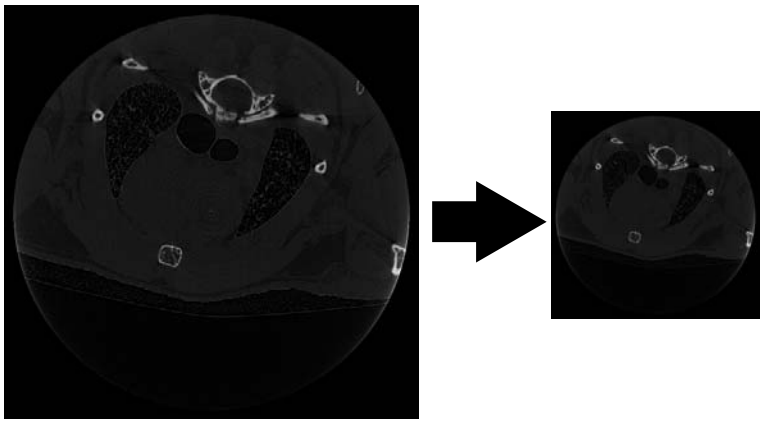


Figure 4.1: スライス画像の縮小

サンプルスクリプト

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

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceBIC を使って x, y, z 方向に 2 画素の画素値を平均して 1 画素にした画像を作成し、新しいディレクトリに保存する。画像はそれぞれの方向に 1/2 になり、全体のサイズ（容量）は 1/8 になる。

サンプルスクリプトによって作成されるファイル：bic/000.tif~163.tif

4.2 スライス画像のトリミングを行う (sliceIT)

画像データから不要な部分を削除する。これによって画像のデータサイズやその後の作業時間を少なく抑えることができる。この例では新しいサンプルデータ（bubble.lzh）を使用していることに注意

サンプルスクリプト

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

1 行目: 画素値 140 以上の部分を物体と見なし、それがすっぽり収まる長方形の領域の座標を表示する

2 行目: データ保存用のディレクトリを作成する。

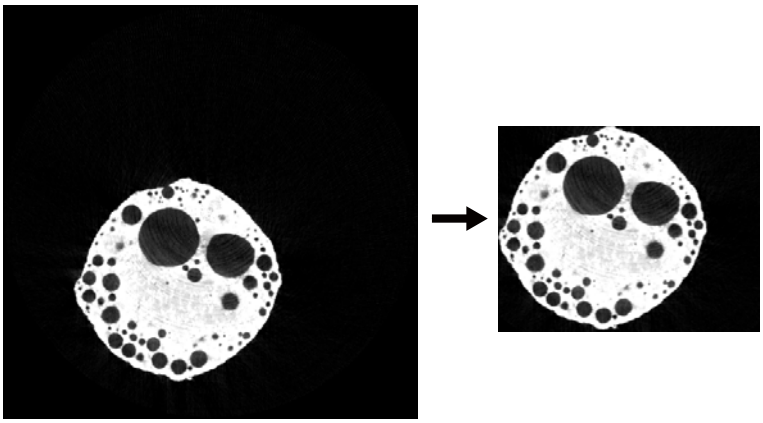


Figure 4.2: スライス画像のトリミング

3 行目: sliceIT を使って指定された領域で画像を切り取ったデータを trim ディレクトリに保存する。

補足説明：
sliceOSP3 実行時の表示

6545861 85 209 32 399 456 282

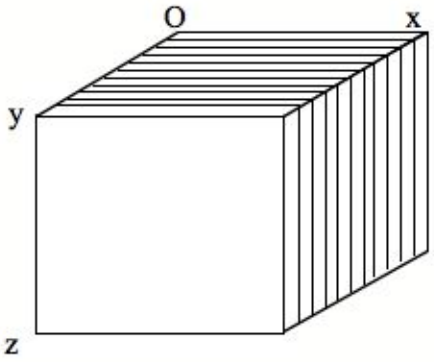
sliceIT に与えられている 6 つのパラメータは OSP3 で表示される切り出す立方体の領域の、原点に一番近い点の座標ともっとも離れた点の座標である。ここでは物体と見なす部分の画素値を手動で与えているが、この作業については §7 を参照。

サンプルスクリプトによって作成されるファイル：trim/000.tif∼250.tif

5　三次元像の reslicing、表示

３次元画像データを用い、ＸＹＺ空間中の様々な平面でカットした断面図を作成する。なお、本章の作業を行うためには slice.tar.gz をインストールしておく必要がある。また、本章のプログラムを実行する際、大きなメモリを必要とする物がある。これらのプログラムを実行する際にエラーが出た場合、前章に戻り、サイズを変更することで本章の作業を実行できるようになる可能性がある。これらの作業では mouse.lzh のサンプルデータを用いている

5.1　x-z 平面での断層画像群を作る (slice_NSX)



サンプルスクリプト
`mkdir nsx`
`slice_NSX BYTE - 1 1 1 nsx`

１行目：作成した断面図を保存する先を作成
２行目：slice_NSX を使って断面図を作る（スケールはすべて 1）

Figure 5.1:

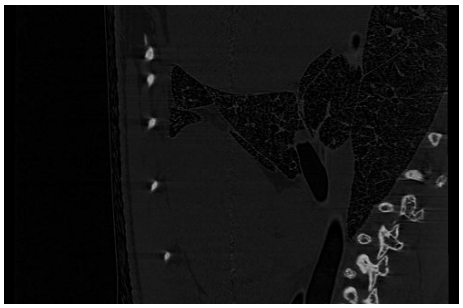
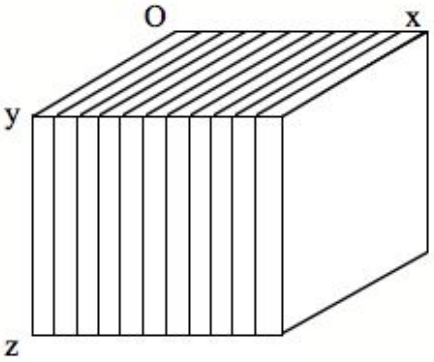


Figure 5.2: 作図例、250.tif

サンプルスクリプトによって作成されるファイル：nsx/000.tif∼499.tif (作図例)

5.2　y-z 平面での断層画像群を作る (slice_NSY)



サンプルスクリプト
`mkdir nsy`
`slice_NSY BYTE - 1 1 1 nsy`

1 行目：作成した断面図を保存する先を作成
2 行目：slice_NSY を使って断面図を作る（スケールはすべて１）。

Figure 5.3:

サンプルスクリプトによって作成されるファイル：nsy/000.tif∼499.tif

5.3 任意の面での断層画像を作る (sliceNSG)

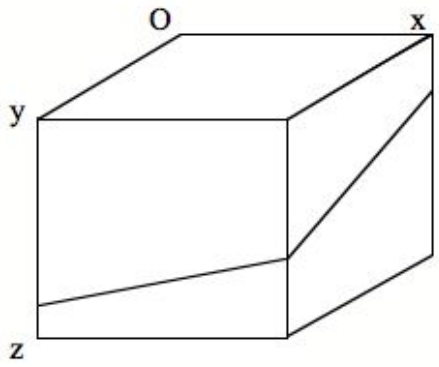


Figure 5.4:

サンプルスクリプト
`echo 250 250 300 20 40 0 cs.tif | sliceNSG BYTE - 1 1 1 0`

1 行目: echo を使って、断面が通るある 1 点のｘｙｚ座標 (250, 250, 300) とその面の視線方向 (断面の法線方向) の経度と緯度 (20 °, 40 °)、断面内での回転 (0 °)、出力ファイル名（cs.tif）を sliceNSG に入力する。sliceNSG で用いられるスケールはすべて 1 で、断層画像に 3 次元画像の外部の領域が含まれる場合、そこに画素値 0 を与える。

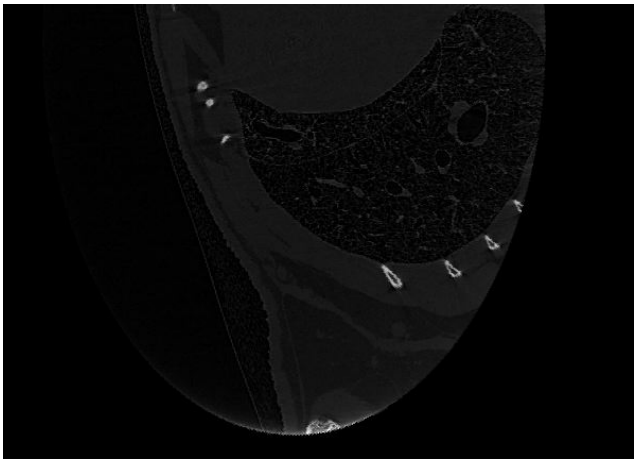


Figure 5.5: 作図例、cs.tif

サンプルスクリプトによって作成されるファイル：cs.tif (作図例)

5.4 任意の面での断層画像群を作る (sliceIR)

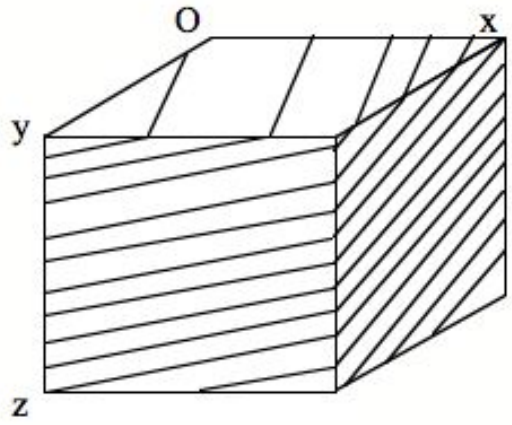


Figure 5.6: 任意の面での断層画像群

サンプルスクリプト
`mkdir ir` `sliceIR BYTE - 60 -20 0 0 ir`

1 行目: 作成した断面図を保存する先を作成

2 行目: 経度と緯度の移動量を 60, -20 とし、面内の回転を０，元の画像の外側が新しい画像に含まれるときの画素値を 0 として、sliceIR を使って断面図を作る（スケールはすべて１で、指定を省略している）。なお、元のデータの x, y, z 座標と新しいデータの x, y, z の関係についてはプログラムリファレンスを必ず参照すること。

補足説明：
このプログラムでは全てのデータを一度メモリに読み込むため、メモリが足りない場合にはエラーが表示され、処理が中断することがある。その場合は画像サイズをちいさくする（前章参照）など、対応をする必要がある。また、このプログラムの実行には多少時間がかかることがある。

サンプルスクリプトによって作成されるファイル：ir/000.tif∼752.tif

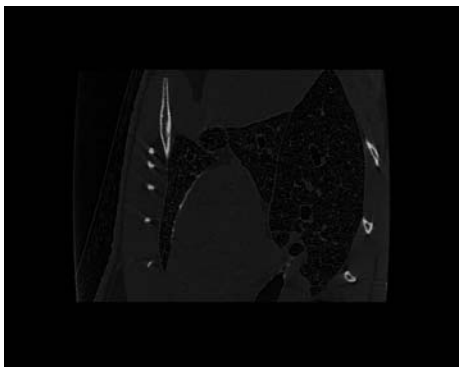


Figure 5.7: 作図例

5.5 3 次元スライス画像の連続表示動画 (gif_movie)

プレゼンテーションなどによく使われる、スライス画像を簡単に閲覧できる動画ファイルを作成する。

サンプルスクリプト

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

1 行目: 各スライス画像の表示間隔がなしで、ループの回数が無限のスライス連続表示動画 gif ファイル slice.gif を作成する。

サンプルスクリプトによって作成されるファイル：slice.gif

5.6 3 次元スライス画像の着色 (slice.CMA)

画像データにカラーを付ける。この例では新しいサンプルデータ（bubble.lzh）を使用していることに注意

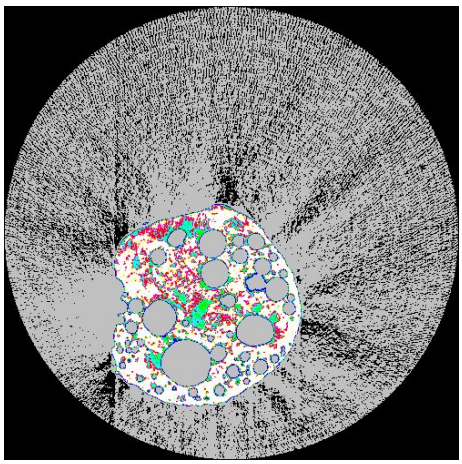


Figure 5.8: 作図例 200.tif

サンプルスクリプト

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

1 行目: 保存用ディレクトリ作成

2 行目: カラーテーブルファイル cluster.cm の中身に従って着色し、cma ディレクトリに保存する

サンプルスクリプトによって作成されるファイル：cma/000.tif~297.tif

補足説明：
カラーテーブルのフォーマットについては reference manual 参照。sliceXXXX.tar.gz（XXXXX は日付）を展開した際に作成される slice/slice/etc ディレクトリに、*.cm という名前のカラーテーブルのサンプルが入っている（入ってない場合は最新版の slice をダウンロードする）。これを作業中のディレクトリにコピーして使用するか、そのカラーテーブルのパス (カラーテーブルのあるディレクトリ) をコマンドで指定する (例: slice.CMA BYTE slice/slice/etc/cluster.cm cma)。

6 ボリュームレンダリング

3次元画像データを用い、ボリュームレンダリングをおこなう。なお、本章の作業を行うためには slice.tar.gz をインストールしておく必要がある。

6.1 ボリュームレンダリング (sliceIT, slice.NO, sliceBEVM.DS, bevm.WD, bevmGS, tiffmask)

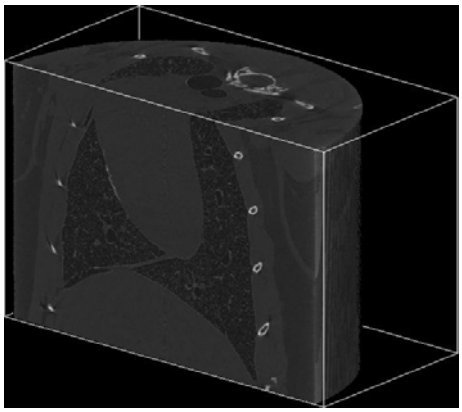


Figure 6.1: 作図例

サンプルスクリプト

```
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
```

1 行目: データ保存用のディレクトリを作成する

2 行目: sliceIT を使って 3 次元画像の必要な部分を切り出す

3 行目: 逆順に並べたファイル（シンボリックリンク）を保存するディレクトリ “no”を作成する

4 行目: slice.No を用いてファイルを逆順に並べるためのスクリプトを作成する

5 行目: スクリプトを実行してファイルを逆順に並べる

6 行目: echo をつかって視点の経度、緯度が (30,20)、光源の経度、緯度が（40 10）の画像を mouse-bevm という単語を含んだ名前で保存するように sliceBEVM.DS に入力し、pvr ディレクトリのデータを使ってこれらのデータがスケールが 1（三つの 1)、となるように 3 次元立体モデルの鳥観図を作成する

7 行目: depth 画像（mouse-bevm.d.tif）から元の 3 次元領域を示す枠線の画像を作成する

8 行目: label 画像（mouse-bevm.l.tif）と shade 画像（mouse-bevm.s.tif）から陰影処理のついた鳥瞰図を作成する

9 行目: edge 画像（mouse-bevm.e.tif）を上記の画像に重ね合わせ、輪郭線を描いた鳥瞰図を作成する

10 行目: 枠線画像（mouse-bevm.w.tif）を上記の画像に重ね合わせる

補足説明：

sliceBEVM.DS で作成される画像は座標系が違うため、回転していることがある。これは sliceBEVM.DS の引数で与える視線方向の角度に依存する（プログラムリファレンス参照)。

サンプルスクリプトによって作成されるファイル：

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 (作図例)

7　二値化について

7.1　二値化とは

二値化とは、画像データにおいて「観察対象となる物体がある」部分と「空気、水、脂肪など、その他の必要ない部分」に分ける作業をさす。この作業を行うことにより、**物体の外形抽出、及び画像処理によるその外形の補完、観察対象のサイズや形状の定量的な議論等が容易になる**。この作業によって作成された画像データの中身は、たとえば物体のないところを 0，物体のあるところを 1 とする 1 bit 白黒画像となる。

Table 7.1: グレースケールデータと二値化データの比較

グレースケールデータ	二値化データ
物体の全体的な内部構造の観察	物体の特定の組織を抽出しての観察
定量的な解析には不向き	体積や形状の定量的な解析が可能

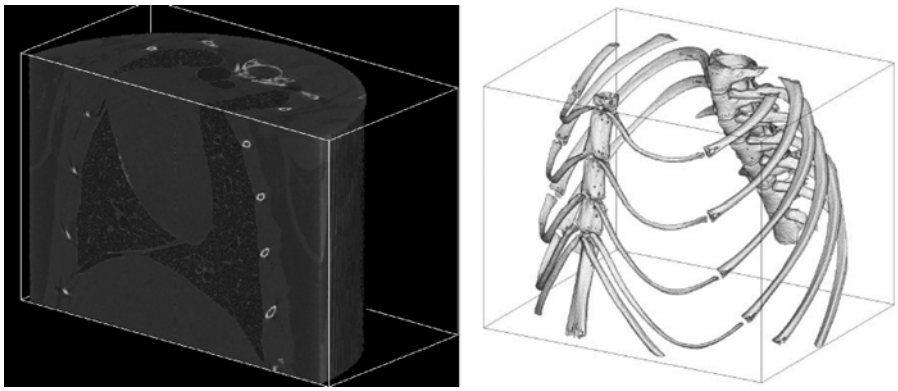


Figure 7.1: マウスの胸部のグレースケールデータ（左）と二値化データ（右）による鳥観図例

いずれの場合も鳥瞰図、3 次元断面図などを作成することはできるが、それぞれ解析の方法によって向き、不向きがあり、場合によって使い分ける必要がある。たとえば、二値化して鳥瞰図を作成した場合、左の画像でみられるグレーのグラデーションで見えている物質の違い（X 線吸収係数の違い）による細かい構造はみることができなくなり、右図のように**グラデーションは単に表面の凹凸を示すだけになる**。逆に、グレースケールデータだと、右の画像のような、必要な部分だけを抜き出したような鳥観図を作ることは難しく、全体の外形と大まかな断面図をとらえることが出来るにとどまる。右の図のような鳥観図を作成するためには、マウスの骨とそれ以外の皮膚から内蔵に至る組織を区別するというような手順を経る必要がある。

また、二値化して作られた物体のデータを元にフィルタをかけて元のグレースケールデータから必要な部分だけを抜き出して表示する、あるいは閾値を決めて不必要な部分をデータから消すなど、様々な応用のしかたがある。

7.2　二値化の準備：閾値

§3.1 で述べたとおり、CT 画像の画素値はその地点における物質の X 線の線吸収係数の値を表しており、この画素値を用いることで自分の関心部分とそうでない部分を切り分けることができる。ただし、関心部分が吸収係数が似通った物質に覆われているような場合、この過程は非常に煩雑であり、目的の構造を取り出すような二値化が不可能な場合もある。以下では二値化が容易なサンプルデータを例に用いてこの過程を説明する。

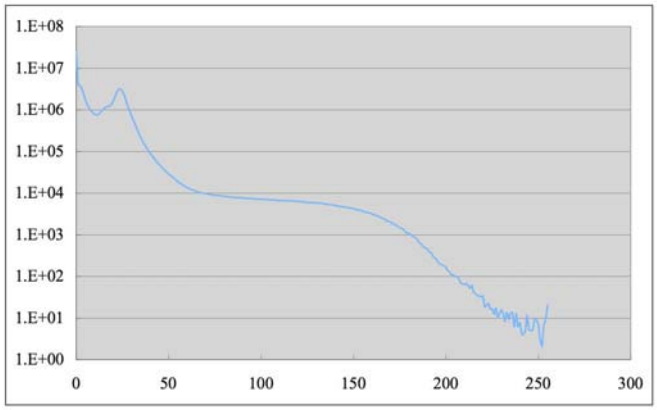


Figure 7.2: mouse サンプルデータのヒストグラム

まず、閾値と呼ばれる、関心物体とそうでない部分の境目となる画素値を決定する。Fig 7.4 は mouse の CT データのヒストグラムである。0 付近の大きなピークは空気（何もない部分）によるものである。空気とそれ以外の部分、のようにデータの内容が単純化できる場合、閾値はこの空気と物体のピークの半分の画素値（もしくはそれよりもやや大きい値）を用いることで閾値としては十分である場合が多い。また、VGStadioMAX など、市販のボリュームレンダリングソフトウェアを用いることで大まかにこの閾値を探すこともできる。

このようにして閾値を求め、二値化した場合でも、一度の作業で自分の求めるデータを得られることは多くない。定量的な解析をする際は多くの場合、この作業の後さらに不必要な部分を排除するなどの作業が必要である。

7.3 具体例

二値化（多値化）の代表的な方法を幾つか説明する。本章では１．画素値のみを基準とする場合、と２．画素値と物体の構造を元にする場合の 2 種類の場合を紹介する。なお、これらの他に sliceBEVM.DS や sliceDE, sliceED, stl など、画像処理プログラムの中には多くの物が二値化、あるいは多値化の機能を備えている。詳しくは以降の章、及びプログラムリファレンスを参照。

7.3.1 例 1:画素値のみを基準として二値化する (slicePVR)

次の例は slicePVR の標準入力に echo を使って、画素値 0-127 の画素の値を 0 に、画素値 128-255 の画素の値を 1 に置き換えるよう指定し、新しいディレクトリ pvr に保存する。その際の結果を pvr.txt に保存する。

サンプルスクリプト

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

1 行目: データ保存用のディレクトリを作成する。
2 行目: slicePVR を使って画素値-127 と 128-で画像を二値化する。

もっとも簡単な二値化の方法。

7.3.2 例 2:画素値と物体の構造を元に多値化する (sliceMCL, sliceMHL)

次にクラスタラベリング, ホールラベリングという手法を用いて物体の構造を元に 3 次元データを多値化、複数の部分に区別して分ける、場合を紹介する。

クラスタラベリング
ある画素値の範囲を持つ画素のうち、隣り合っている物を全て一つの固まりと見なし、背景に 0, 個々の固まりに同一の画素値をクラスタのサイズの大きい順に 1 から与えることにより、物体の構造をより定量的に調べることが出来るようになる。この作業を行うのがクラスタラベリングである。

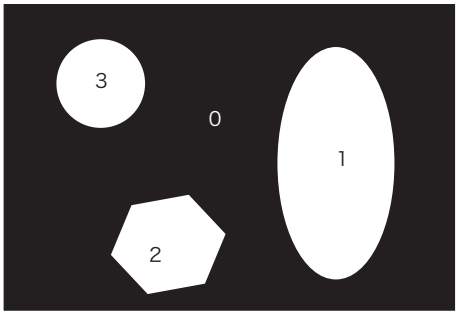


Figure 7.3: クラスタラベリングの概念図：数字は画素に与えられた画素値を表す

サンプルスクリプト

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

1 行目: データ保存用のディレクトリを作成する。
2 行目: sliceMCL を使って画素値 65-80 の物を物体としてクラスタにラベリングを行う。

ホールラベリング
ホールラベリングとは、ある画素値の範囲を持つ画素を物体と見なし、この物体像に開いた穴（外部とつながっていない物体像以外の部分）に対してクラスタラベリングを行うことをさす. このとき、物体像でない背景の画素には 0、物体像に 1、それ以外には画素が属する hole のサイズ（hole に属する画素数）の順に 2 から付けられた値を画素値とする. 作図例は bubble.lzh のデータを用いており、カラーテーブルは sliceXXXXX.tar.gz の中の slice/slice/etc の中の cm.cm を用いている（XXXXX は日付）。

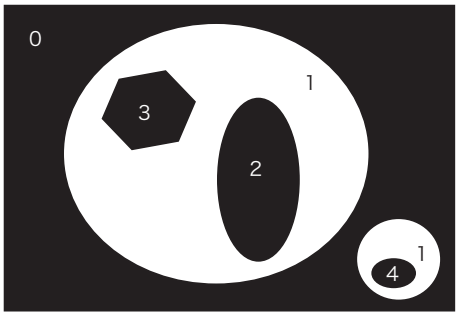


Figure 7.4: ホールラベリングの概念図：数字は画素に与えられた画素値を表す

サンプルスクリプト

```
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
```

1 行目: データ保存用のディレクトリを作成する
2 行目: sliceMHL を使って画素値 100-255 の物を物体としてその物体に開いた穴にラベリングを行う。
3 行目: データ保存用ディレクトリを作成する。
4 行目: slicePVR で小さな穴 (256-65535 番目の穴) の画素値を 0 にする
5 行目: データ保存用ディレクトリを作成する。
6 行目: §5 で紹介した slice.CMA を用いて着色する。

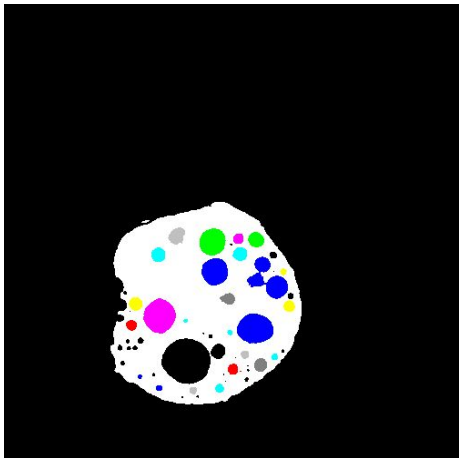


Figure 7.5: 作図例 200.tif

以降の章では様々な二値化の方法と、それを元にポリゴンデータを作成し、立体表示を行う方法など、slice シリーズにおける二値化データの利用方法について説明する。

8 Polygonを用いたボリュームレンダリング

§7 のような手法で二値化されたデータを表示するにはポリゴンを用いた表示法が便利である。本章では STL 形式のポリゴンデータを用いた鳥観図の作成方法を説明する。なお、本章の一部の作業を行うためには stl.tar.gz をインストールしておく必要がある。

8.1 ポリゴンによる鳥観図を作成する

ポリゴンを用いた 3 次元鳥観図を作成する。スライスデータから直接作成する方法と、STL ファイルを使う、二通りの方法がある。

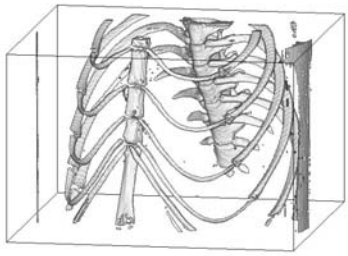


Figure 8.1: 作図例

8.1.1 スライスデータから直接作成する (si_s_bev, si_m_bev)

まず、スライス画像データから直接ポリゴン表示による三次元鳥観図を作成する。

グレースケールの鳥観図を作成 (si_s_bev)

サンプルスクリプト
`echo 260 190 | si_s_bev BYTE - 90-255 1 1 1 1 64 255 0 mouse.gif`

1 行目: si_s_bev を使って BYTE ディレクトリのデータから、(260, 190) の角度からみたポリゴンによる鳥観図を作成する。この際 90 から 255 までの画素値を持つ画素を物体としている。

カラーの鳥観図を作成 (si_m_bev)

サンプルスクリプト
`si_m_bev BYTE - color.tbl 1 1 1 1 64 255 255 255 0 0 0 mouse_C.gif` (リターン) `260 190` (リターン、続いて `Control` を押しながら `d`)

1 行目: si_m_bev を使って BYTE ディレクトリのデータから、カラーテーブルを使ってポリゴンによる鳥観図を作成する。

2 行目: (260, 190) の角度からみた鳥観図を作成することを指示する入力、入力を終了。

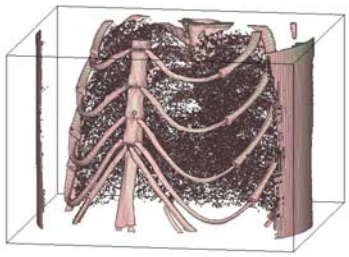


Figure 8.2: 作図例

color.tbl：
テキストエディタなどで以下の内容を記述したファイルを用意する。

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

説明

1 行目: 0 から 50 までの画素値を持つ画素は物体と見なさない
2 行目: 51-89 までの画素値を持つ画素を RGB=(255 200 200) にする
3 行目: 90-255 までの画素値を持つ画素を RGB=(200 255 200) にする

サンプルスクリプトによって作成されるファイル：mouse.gif, mouse_C.gif (作図例)

8.2 STL 形式のポリゴンデータファイルを作る (si_stl_B, si_stl_C：stl.tar.gz が必要)

一般的にポリゴンデータのファイルフォーマットとして用いられている STL 形式のデータファイルを作成する。作成されたデータは VGStadioMAX 等の他のソフトでも使用することが出来る。

グレースケールの STL ファイル作成例 (si_stl_B)

サンプルスクリプト

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

1 行目: si_stl_B を使って BYTE ディレクトリのデータから mouse.stl という stl 形式の三次元データを作成する。この際 90 から 255 までの画素値を持つ画素を物体としている。

2 行目: 3 次元画像の x、y、z 方向の画素数（自動的に表示される）

3 行目: 3 次元画像上で物体像と見なした画素の総数 、物体のある範囲（自動的に表示される）

4 行目: 物体像の表面積 (表面に面する画素の面の総数)、物体像の表面を分割した三角形の総数（自動的に表示される）

カラーの STL ファイル作成例 (si_stl_C)

サンプルスクリプト

```
si_stl_C BYTE - mouse_C.stl（リターン）
500     500     328
0 50（リターン）
51 89 255 200 200（リターン）
90 255 200 255 200（リターン、続いて Control を押しながら d）
921156  12      12      0       488     407     327
1233694 1516171
```

1 行目: si_stl_B を使って BYTE ディレクトリのデータから mouse_C.stl という stl 形式の三次元データを作成する。

2 行目: 0 から 50 までの画素値を持つ画素は物体と見なさないことを指示する入力

3 行目: 51-89 までの画素値を持つ画素を RGB=(255 200 200) にすることを指示する入力

4 行目: 90-255 までの画素値を持つ画素を RGB=(200 255 200) にすることを指示する入力、入力を終了

5 行目: 3 次元画像上で物体像と見なした画素の総数 、物体のある範囲（自動的に表示される）

6 行目: 物体像の表面積 (表面に面する画素の面の総数)、物体像の表面を分割した三角形の総数（自動的に表示される）

補足説明：

si_stl_B,C はデフォルトでは 8 ビットの画像しか扱えない。これをサンプルプログラムのように 16 ビットの画像で動かそうとするとエラーが出る。この場合は stl コンポーネントをコンパイルし直す必要がある（stl.tar.gz 内のプログラムリファレンス参照)。

このスクリプトはコマンドラインで実行することを前提としている。バッチ処理スクリプトなどに組み込む場合はカラー情報の入力や視線方向の入力などはヒアドキュメントなどを用いてコマンドに標準入力から入力する必要がある（詳細はプログラムリファレンス参照)。

サンプルスクリプトによって作成されるファイル：mouse.stl, mouse_C.stl

8.2.1 STL データからポリゴン鳥観図を作成する (stl_bev_SS, stl_bev_C_SS)

スライス画像データから直接作成する場合と同様の鳥観図を、§8.2 で作成した STL ファイルを用いて作成することも出来る。

グレースケールの鳥観図を作成 (stl_bev_SS)

サンプルスクリプト

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

1 行目: stl_bev_SS に、mouse.stl を使って (260,190) からみた 3 次元の鳥観図を作成するよう入力

カラーの鳥観図を作成 (stl_bev_C_SS)

サンプルスクリプト

```
stl_bev_C_SS mouse_C.stl 1 1 16 255 255 255 0 0 0 0 0 0(リターン)
0 0 X0_0_C.tif(リターン)
260 190 260_190_C.tif（リターン、続いて Control を押しながら d）
```

1 行目: stl_bev_C_SS に、mouse_C.stl を使って 3 次元の鳥観図を作成するよう入力
2 行目: (0, 0) の角度からみた鳥観図を作成することを指示する入力
3 行目: (260, 190) の角度からみた鳥観図を作成することを指示する入力、入力を終了。

補足説明：
stl_bev_C_SS で作成される画像は座標系が違うため、回転していることがある。これは stl_bev_C_SS の引数で与える視線方向の角度に依存する（プログラムリファレンス参照)。サンプルスクリプトの例では作成されたそれぞれの画像を 180 °回転させている。

サンプルスクリプトによって作成されるファイル： 260_190.tif, X0_0_C.tif, 260_190_C.tif

8.3 立体画像の動画を作る

3 次元空間で視線方向を移動させた、ポリゴン表示による動画ファイルを作成する。ファイルフォーマットは動画 gif 形式を用いている。
鳥観図と同様に、STL ファイルを使う方法と、スライスデータから直接作成する、二通りの方法がある。

8.3.1 スライスデータから直接作成する (si_s_bev, si_m_bev)

グレースケールの立体動画を作成 (si_s_bev)

サンプルスクリプト

```
si_s_bev BYTE - 100-255 1 1 1 1 64 255 0 mouse.gif(リターン)
180 180 0 10 2(リターン)
180 200 10 0 8(リターン、続いて Control を押しながら d)
```

1 行目: si_s_bev を使って BYTE ディレクトリのデータからポリゴンによる鳥観図を作成する。この際 100 から 255 までの画素値を持つ画素を物体としている。
2 行目: 経度、緯度が (180, 180) から (0, 10) 刻みで 2 回移動させる
3 行目: 2 行目の指示に引き続き、経度、緯度が (180, 200) から (10, 0) 刻みで 8 回移動させる。入力終了

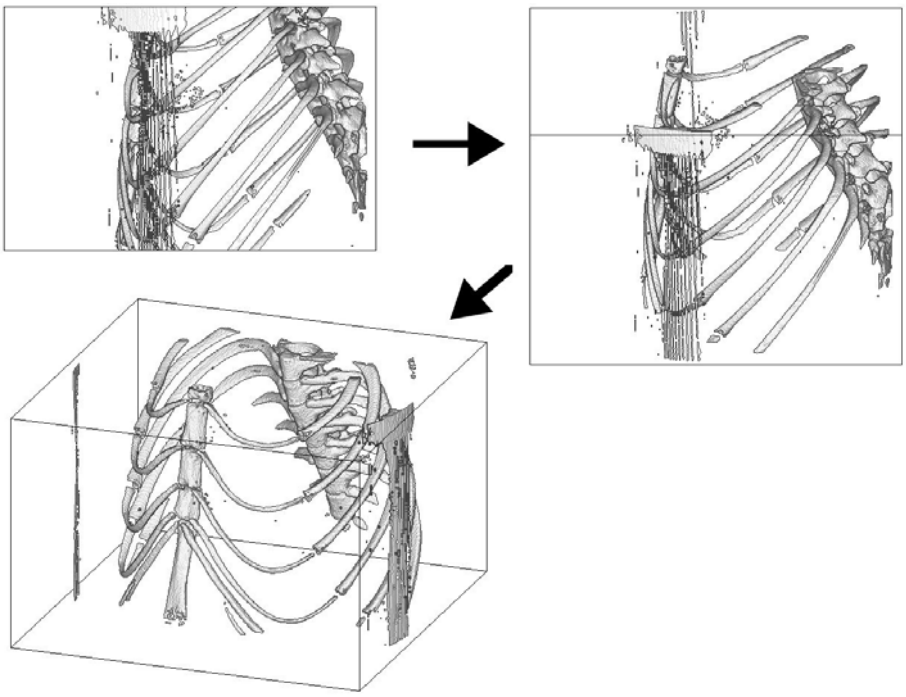


Figure 8.3: 動画作図例

カラーの立体動画を作成 (si_m_bev)

サンプルスクリプト

`si_m_bev BYTE - color.tbl 1 1 1 1 64 255 255 255 0 0 0 mouse_C.gif（リターン）`
`180 180 0 10 2(リターン)`
`180 200 10 0 8(リターン、続いて Control を押しながら d)`

1 行目: si_m_bev を使って BYTE ディレクトリのデータから、ポリゴンによる鳥観図を作成する。
2 行目: 経度、緯度が (180, 180) から (0, 10) 刻みで 2 回移動させる
3 行目: 2 行目の指示に引き続き、経度、緯度が (180, 200) から (10, 0) 刻みで 8 回移動させる。入力終了

color.tbl：
テキストエディタなどで以下の内容を記述したファイルを用意する。

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

説明

1 行目: 0 から 50 までの画素値を持つ画素は物体と見なさない
2 行目: 51-89 までの画素値を持つ画素を RGB=(255 200 200) にする
3 行目: 90-255 までの画素値を持つ画素を RGB=(200 255 200) にする

サンプルスクリプトによって作成されるファイル：mouse.gif, mouse_C.gif (作図例)

8.3.2 STL データから作成する (stl_bev_GIF_SS, stl_bev_GIF_C_SS：stl.tar.gz が必要)

グレースケールの立体動画を作成 (stl_bev_GIF_SS)

サンプルスクリプト

```
( 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
```

1 行目: stl_bev_GIF_SS に、mouse.stl を使って、経度、緯度が (180, 180) から (0, 10) 刻みで 2 回移動させ、次に (180, 200) から (10, 0) 刻みで 8 回移動させる立体動画を作成するよう入力

カラーの立体動画を作成 (stl_bev_GIF_C_SS)

サンプルスクリプト

```
stl_bev_GIF_C_SS mouse_C.stl 1 1 16 255 255 255 0 0 0 0 0 0  mouse_move_C.gif(リター
ン)
180 180 0 10 2(リターン)
180 200 10 0 8(リターン、続いて Control を押しながら d)
```

1 行目: stl_bev_C_SS に、mouse_C.stl を使って 3 次元の鳥観図を作成するよう入力

2 行目: 経度、緯度が (180, 180) から (0, 10) 刻みで 2 回移動させる

3 行目: 2 行目の指示に引き続き、経度、緯度が (180, 200) から (10, 0) 刻みで 8 回移動させる。入力終了

補足説明：
stl_bev_GIF_C_SS で作成される画像は座標系が違うため、回転していることがある。これは stl_bev_GIF_C_SS の引数で与える視線方向の角度に依存する（プログラムリファレンス参照)。サンプルスクリプトの例では作成されたそれぞれの画像を 180 °回転させている

サンプルスクリプトによって作成されるファイル： mouse_move.gif, mouse_move_C.gif

8.4 注：角度について

Fig8.4 のような物体の 3 次元画像データから様々な (経度、緯度) の値を指定して ポリゴン画像を作成したとき、画像の向きがどのようになるかを示す。

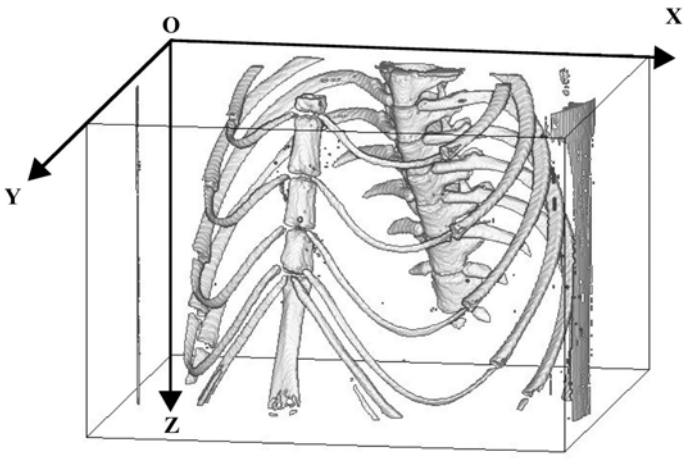
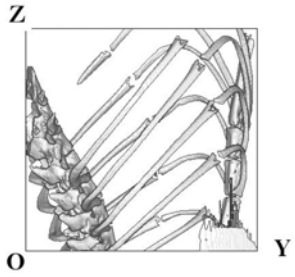
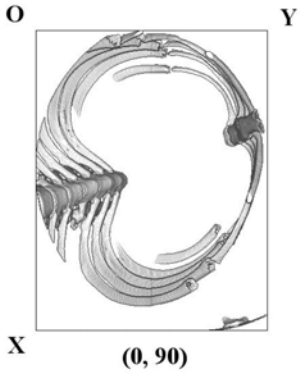


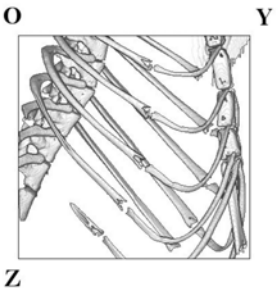
Figure 8.4: 元データの座標



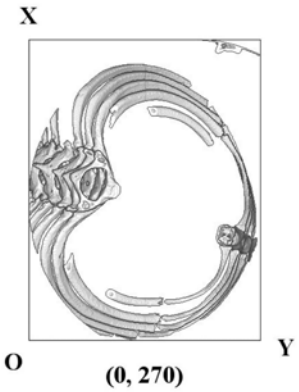
(0, 0)



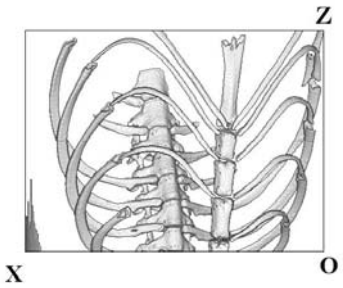
(0, 90)



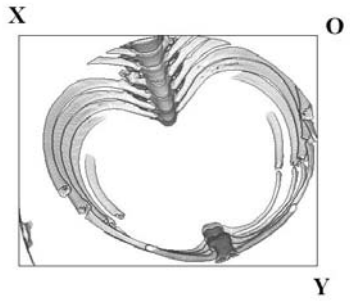
(0, 180)



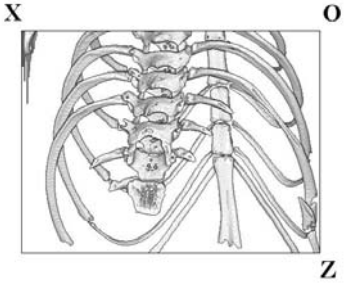
(0, 270)



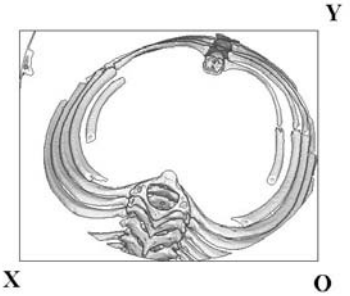
(90, 0)



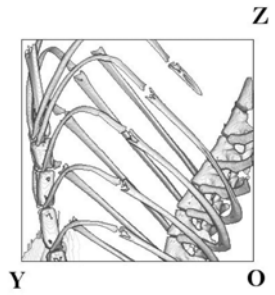
(90, 90)



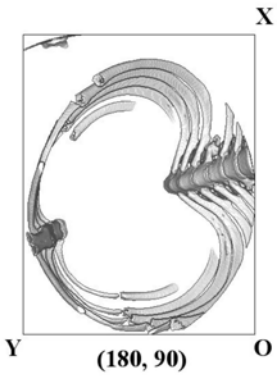
(90, 180)



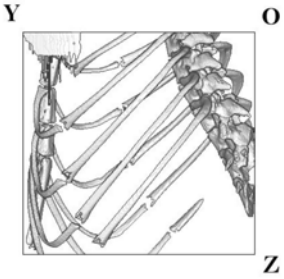
(90, 270)



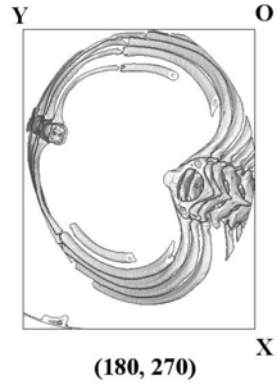
(180, 0)



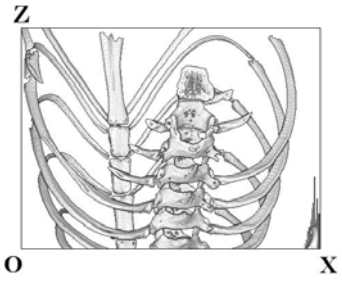
(180, 90)



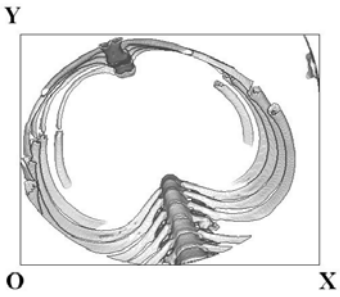
(180, 180)



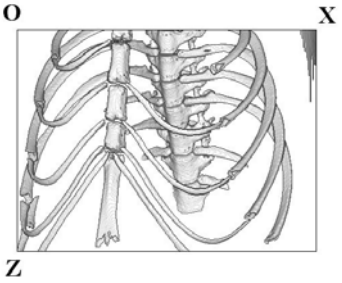
(180, 270)



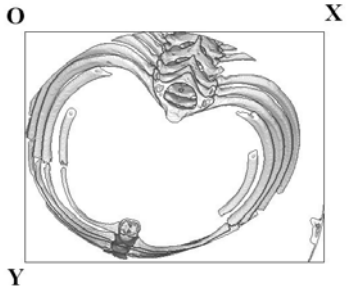
(270, 0)



(270, 90)



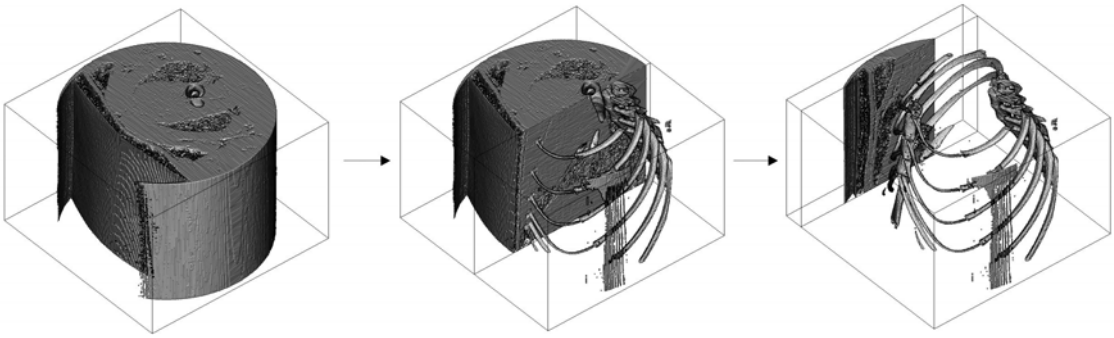
(270, 180)



(270, 270)

8.5 ポリゴン画像データの重ね合わせ

二つのポリゴン画像をある断面で接合した画像、遷移動画を作成する。



サンプルスクリプト

```
echo 500 220 220 -50 0 0 10 | si_x_bev BYTE - color.tbl （改行しない）
BYTE - color2.tbl 1 64 255 255 255 0 0 0 move.gif
```

1 行目: si_x_bev を使って BYTE ディレクトリのデータからポリゴンによる鳥観図を作成する。x=500 を通る y-z 平面で接合した color.tbl color2.tbl で作成した二つの鳥観図を、経度緯度が (220, 220) の方向からみた図を作成する。また、接合面を x=-50（緯度経度は 0,0 なので動かさない）ずつ 10 回ずらして、動画を作成する。

カラーテーブル

テキストエディタなどで以下の内容を記述したファイルを用意する。

color.tbl：
0-19
20-255 255

color2.tbl：
0-89
90-255 255

補足説明：
この例ではおなじ BYTE ディレクトリのデータを用いているが、違うディレクトリの違うデータを接合することも出来る。また、同じプログラムでカラーのポリゴンデータを作成し、重ね合わせることも可能（その場合、カラーテーブルを書き換えればよい)。また、二つめのポリゴンデータのフォーマット指定（BYTE - color2.tbl）を (- - /dev/null) などとすると、重ね合わせるポリゴンが無いため、断面の鳥観図を作成することが出来る。y,z 平面で接合したい場合はそれぞれ si_y_bev,si_z_bev を用いる。

サンプルスクリプトによって作成されるファイル： move.gif

9　二値化データの画像処理

二値化したデータを用いれば、必要な部分を抜き出したり定量解析を行ったりと様々な処理が可能になる。この章では定量解析をする際に必要となるような画像処理例を紹介する。なお、9.1 章ではサンプルデータに加熱によって内部に泡（空洞）を含んでいる岩石のサンプルデータ（bubble.lzh）も用いている。

9.1　穴埋めをする (sliceMHL, sliceDE, slicePVR)

物体に穴が開いた CT データを処理することがある。このような穴は 単に物体の外形だけを 3 次元データ化したいときに、その出力ファイルのサイズが増加するなど、デメリットがあるため、これを消す方法を幾つか紹介する。

9.1.1　sliceDE を用いる

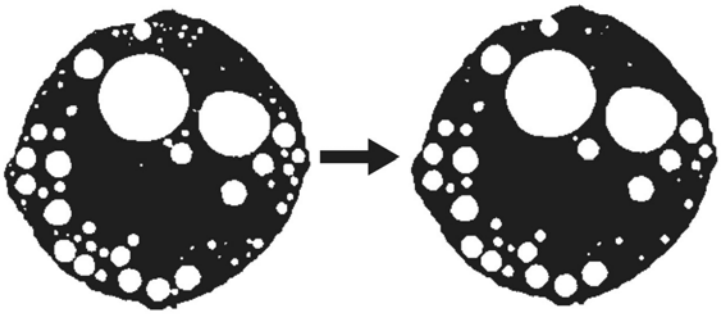


Figure 9.1: 作図例 152.tif

サンプルスクリプト

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

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceDE を使って画素値が 140 から 255 までの pixel で構成される部分を物体と見なし、3 画素分だけ dilation, erosion を行う。結果を de に保存する。

補足説明：
sliceDE は dilation（画像の物体部分を指定したピクセルだけ外側に増やす）その後同じピクセル数だけ erosion（画像の物体部分を指定したピクセルだけ内側に削る）を行い、この場合は 6 ピクセルより小さな穴などを埋めることができるプログラムである（詳細はプログラムリファレンスを参照)。画像の処理前（図 9.1 左）と処理後（図 9.1 右）を比べると小さな穴の数が減っていることが分かる。しかし、このプログラムを使うとホールの形が変わってしまったりしている。なお、この作業行うと、物体が存在しているとされたすべての画素の画素値は 1 となる。

sliceDE を使うことで小さな空隙や隙間を埋めることができるが、sliceED を使えば逆に画像上の小さな点（前節で示したノイズなど）を消すことができる。

サンプルスクリプトによって作成されるファイル：
de/000.tif∼297.tif

9.1.2 sliceMHL, slicePVR を用いる

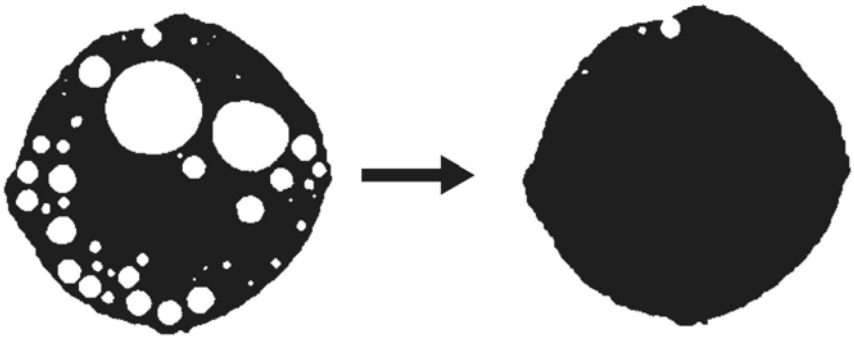


Figure 9.2: 作図例 152.TIF

サンプルスクリプト

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

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceMHL を使って画素値が 140 から 255 までの pixel で構成される物体にあいたホール（集団）に対して、1 を物体として 2 から順番にラベリングしていく。（§7 参照）

3 行目: データ保存用のディレクトリを作成する。

4 行目: echo で画素値 2 から 65535 を持つピクセルに画素値 1 を与えるよう slicePVR に入力し、その結果を mhlpvr に保存する。

補足説明：
sliceMHL は物体と見なした部分にあいた穴に 2 以上の画素値を与えることで穴を識別することが出来る。さらに slicePVR を用いてその穴にも画素値 1 を与えることで穴を埋めている。ここでいう「穴」とは 3 次元的に外部とつながっていない閉じられた空間を指し、たとえ空洞がビーズの内部に入り込んでいても、その空洞が 3 次元領域で外部とつながっている場合は穴として認識しない。なお、3 次元データの外側は何もない部分としてカウントされているので、穴の開口が画像データの外側に接している場合も穴であると認識は出来ない。このプログラムでは、sliceDE を使ったときのような穴の変形はみられないが、上述のように外部とつながっている空間を埋めることはできない。これを埋めたい場合は sliceDE を併せて使うと良い。また、この作業を通しても物体が存在しているとされたすべての画素の画素値は 1 となる。

サンプルスクリプトによって作成されるファイル：
mhl/000.tif∼297.tif
mhlpvr/000.tif∼297.tif

9.2 余分な部分を削除する（slicePVR）

スライス画像からポリゴン鳥観図を作成する場合、必要な部分以外を削除したい場合がある。これを slicePVR を用いて行う。この作業は鳥観図を作成するだけなら slicePVR を使わなくとも si_s_bev 単体で行うことが出来るが、この場合は画像データには処理が反映されない。

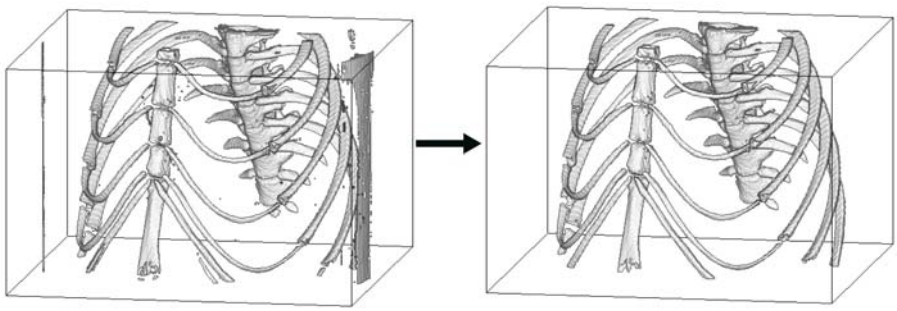


Figure 9.3: 作図例

サンプルスクリプト

```
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
```

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceMCL を使って画素値が 90 から 255 までの pixel で構成されるクラスタ（集団）に対して、1 から順番にラベリングしていく。(§7 参照)

3 行目: データ保存用のディレクトリを作成する。

4 行目: echo を使って slicePVR に画素値が 22, また 44 から 65535 までの pixel で構成されるクラスタ（集団）に画素値 0 を与える。又その他の画素を全部 1 にする

5 行目: si_s_bev をつかって (260, 90) から見たポリゴン鳥観図を作成する

補足説明：
sliceMCL でラベリングを行い、調べると画素値 22 の物が画像端部分のノイズに相当することが分かり、また画素値 44 以下のクラスタは小さなゴミであることが分かる。この必要ない部分の画素をすべて 0 に置き換えることで、画面内の不必要な物体を消去している。

サンプルスクリプトによって作成されるファイル：
mcl/000.tif∼327.tif
mclpvr/000.tif∼327.tif
mouse.gif

9.3 マスク処理をする (slicePVM)

二値化した画像を用いて、元画像のマスクを行うことで、元画像から必要な部分だけを抜き出すことも出来る。

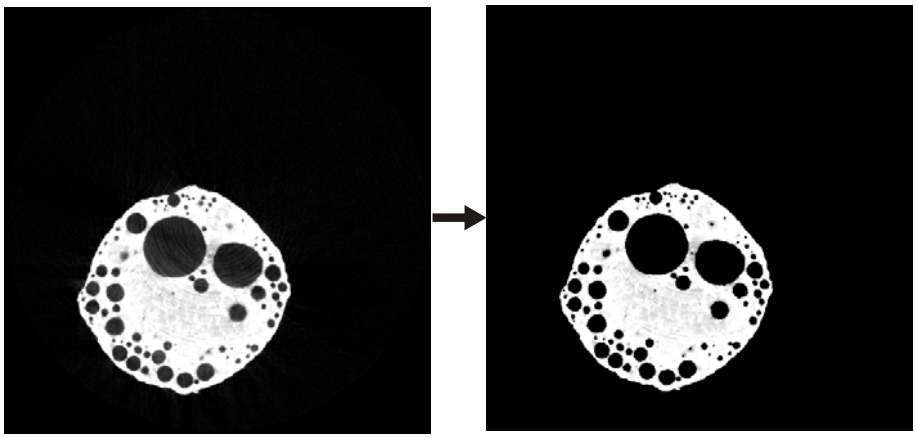


Figure 9.4: 作図例 152.tif

サンプルスクリプト

```
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
```

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceMHL を使って画素値が 140 から 255 までの pixel で構成される物体にあいたホール（集団）に対して、1 を物体として 2 から順番にラベリングしていく。(§7 参照)

3 行目: データ保存用のディレクトリを作成する。

4 行目: echo を使って slicePVR に画素値が 2 から 65535 までのホールに画素値 0 を与える。

5 行目: データ保存用のディレクトリを作成する。

6 行目: slicePVM を使って BYTE ディレクトリの中の画像を pvr 内の二値化されたデータでマスクする。作成されたデータを pvm に保存する。

補足説明：
マスク処理とはある画像データを元に元画像の一部を切り取る作業のことである（詳細はプログラムリファレンスを参照)。サンプルスクリプトの例では、元データ（BYTE）から、二値化されたデータ（pvr）の物体に当たる部分だけを抜き出した画像を作成することにより、大きな穴の内部などに見られる余分なノイズを大幅に消去することができている。

サンプルスクリプトによって作成されるファイル：
pvm/000.tif∼297.tif
pvr/000.tif∼297.tif
mhl/000.tif∼297.tif

9.4 三次元画像データの透過重ね合わせ (si_cim)

前節で穴埋めをした画像の結果比較を、画像を透過しながら重ね合わせることで確かめる。

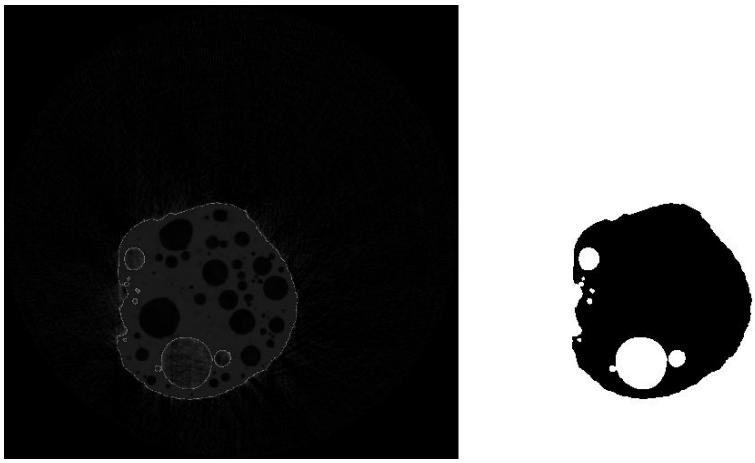


Figure 9.5: 作図例

サンプルスクリプト
`mkdir cim` `si_cim BYTE - mhlpvr - 0.1 cim`

1 行目: データ保存用のディレクトリを作成する。

2 行目: si_cim を使って BYTE ディレクトリのデータの上に mhlpvr の穴埋めしたデータを 0.1 倍にして重ね合わせた画像を cim ディレクトリに保存する。

補足説明：
作図例右は穴埋めをした画像データ。BYTE ディレクトリの画像に重ね合わせたのが左の図。穴埋めの出来ていない画像には BYTE 画像の中身がうつっているため、白い縁取りが見えている。画像はスライス画像ではない stl 等で作成された tif のカラー画像も重ね合わせることが出来る

サンプルスクリプトによって作成されるファイル：
cim/000.tif∼297.tif

10　二値化されたデータを用いた定量解析例

この章では二値化されたデータを用い、CT データを定量解析する例を挙げる。なお、この章ではサンプルデータに加熱によって内部に泡（空洞）を含んでいる岩石のサンプルデータ（bubble.lzh）を用いている。

10.1　ポロシティの評価 (sliceMHL, slicePVR)

内部に外部と接続していない気泡がたくさん存在するデータを用い、ポロシティ（空隙率）の評価をする

サンプルスクリプト

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

1 行目: sliceMHL を使って画素値が 140 から 255 までの pixel で構成される物体内部のホール（外部とつながっていない空洞）に対して、2 から順番にラベリングしていく。このとき物体外部の空間に画素値 0, 物体部分に画素値 1 が与えられる（§7 参照）

2 行目: slicePVR を用いてすべての穴の画素値を全て 255 にする。

3 行目: slicePVR を用いてヒストグラムデータを作成し、その結果を pvr.txt に書き出す

補足説明：
pvr.txt の中身
0 0 67954139
1 1 4515889
255 255 2029972

画素値 1 は物体を表しており、255 は内部の空間を表している。画素値 1 と 255 の画素数を足した物が全体積であり、それに対する空隙率は画素値 255 の画素数を全体積で割ってやることで得ることが出来る。ただし、これは物体に外部とつながった大きな空隙が存在しないことを前提とした解析手法であり、そういった空隙が存在する場合はその空隙の外部とのつながりを sliceDE 等でふさぐ必要がある。

サンプルスクリプトによって作成されるファイル： pvr.txt

10.2　二値化されたデータの三軸不等楕円体近似 (sliceOF, of_stl_ih)

試料内部の泡の形状を二値化データを元に三軸不等楕円体近似し、その結果をテキストデータに出力する。

サンプルスクリプト

```
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
```

1 行目: データ保存用のディレクトリを作成する。

2 行目: sliceMHL を使って画素値が 140 から 255 までの pixel で構成される物体内部のホール（外部とつながっていない空洞）に対して、2 から順番にラベリングしていく。（§10.1 参照）

3 行目: データ保存用のディレクトリを作成する。

4 行目: slicePVR を用いて一番大きな穴（画素値が 2 のクラスタ）以外の穴の画素値を全て 0 にする

5 行目: sliceOF を用いて楕円体近似を行い（スケールは各辺 2 倍)、その結果を of.txt に保存する

補足説明：
of.txt の中身
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

of.txt には最初の一行目に画素値 0（背景）の近似データ、その後に気泡のデータが出力される。データフォーマットは, クラスタ番号, クラスタに属する画素の個数, クラスタの重心の X、Y、Z 座標値（画像上の座標表現）, クラスタの重心の X、Y、Z 座標値（実際の長さの座標系）, 楕円体の 3 軸の方向を示す回転角 λ、ϕ、θ（単位は度）, 楕円体の 3 軸の長さ a、b、c（実際の長さの単位）, 軸長から計算された楕円体の体積（a×b×c×π×4/3）となる。スケールを 1 にすると実際の長さで示されたデータはこの表示の半分になる。

次に上記のスクリプトで出力された楕円体のテキストデータ (of.txt) を元に stl 形式のポリゴンファイル、及び鳥観図を作成する

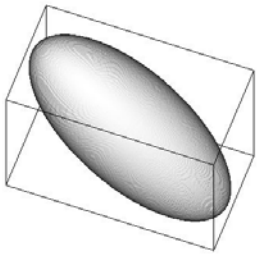


Figure 10.1: 三軸不等楕円体近似した気泡のポリゴンデータ

サンプルスクリプト

```
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
```

1 行目: unix の tail, cut コマンドを用いて必要なデータを of.txt から抜き出して of_stl_ih に入力し、of.stl という名前の stl 形式のポリゴンデータファイルを作成する

2 行目: stl ファイルから鳥観図を作成する

このデータでは幾つかの気泡が合体した物を楕円体近似しているため、非常に長細い（長軸が長い）形の楕円体に近似されている。

サンプルスクリプトによって作成されるファイル： of.txt, of.stl, 20-60.tif

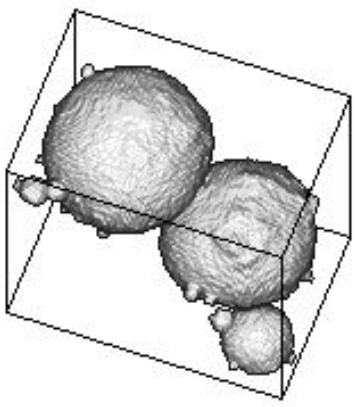


Figure 10.2: 元の気泡の形 (練習問題としてこの画像を作ってみてください)