STL データ処理用のプログラムについて

地質調査総合センター・中野 司
tsukasa.nakano@aist.go.jp

16 August 2013

UNIX の端末環境（Cygwin を含む）と Windows の DOS 窓の両方で実行可能な自作の STL データ処理用のプログラム群を紹介する。これらを利用すれば STL データのチェックやファイル形式の変換、 STL データが表している３次元物体像の陰影つき鳥瞰図の描画（カラーの鳥瞰図やそれらのアニメーションも作成可能）、および、指定した３次元画像中の物体像を表す白黒やカラーの STL データの作成などの処理を行うことができる。また、これらのプログラム群と一緒に使えば有用なフリーの STL データ処理用ソフトウェアのいくつかも紹介する。

STL データとは何か

STL（STereo Lithography）データとは計算幾何学や CAD（Computer-Aided Design）などの分野で３次元物体像の形状を表現する場合に広く用いられている STL 形式（STL format）のファイルに格納されているものと同等なデータのことである。このファイル形式は３次元物体像の形状を取り扱うソフトウェアに必須のものであり、特に製品開発の過程において必要不可欠な実体模型の造形（rapid prototyping；RP）の処理では事実上の標準として用いられている。

STL 形式の正式の仕様書の有無は不明だが、インターネット上には例えば下記のホームページのようなそれに関する解説が数多く出回っている。

http://en.wikipedia.org/wiki/STL_%28file_format%29
http://www.ennex.com/~fabbers/StL.asp

これらに記されているように、 STL 形式は三角形の面（facet と呼ばれている）から構成される多面体で近似した３次元物体像の形状を表現できる。具体的には、多面体近似した３次元物体像の表面を覆い尽くす三角形のそれぞれを特定する以下の 12 個の単精度浮動小数点数の値（通常用いる xyz 座標系における座標値など）が STL データの主な構成要素である。

面の（物体像の内から外に向かう）法線ベクトルの成分値 (nx, ny, nz)
頂点の座標値 (x1, y1, z1)、(x2, y2, z2) および (x3, y3, z3)

後で紹介する color STL 形式では三角形それぞれの色の情報も記述できるが、それは拡張された構成要素であり、 STL データにとって本質的なものではない。すなわち、 STL 形式は多面体近似した３次元物体像の形状の表現に必要最小限の情報を記述できるだけである

最近の CAD 用などのソフトウェアが採用している汎用なファイル形式（それらの多くはソフトウェアに固有なもので標準化は未決のようであるが）とは異なり、STL 形式は物体像を被覆している三角形の相互関係（複数の三角形が共有している頂点や辺に関する情報）を記述できない。GIS（Geographical Infromation System）などで処理される２次元の地図データに即して言うと、STL データは領域の輪郭を示している折れ線に相当する三角形それぞれを個別に記述しているに過ぎない。また、本来の STL 形式ではファイルごとに三角形に完全に覆われた単連結の１個の物体像の記述しか許していない。言い換えると、このような３次元物体像の連結性や三角形による被覆の完全性などの STL データのトポロジカルな整合性をチェックして修正・編集することは非常に困難（原理的には不可能）である。

その反面、 STL データは他の形式のものに比べると作成が容易なので、指定したパラメータをもとに新たな３次元物体像の形状を計算プログラムで記述する場合などに適している（特に後で紹介する ASCII STL 形式のファイルでは物体像を覆い尽くす三角形の総数の記述すら不要）。

また、（データの整合性がさほど深刻な問題にならない）３次元物体像の鳥瞰図の描画では三角形を単位とした物体像の表面の塗りつぶし（surface rendering）が一般的なので、 STL データを介すことによって無駄のない高速な処理を期待できる。さらに、後で紹介するようにして３次元画像から取り出した（トポロジカルな整合性に問題がない）STL データに対して Gauss の発散定理を応用して物体像の領域における体積積分（モーメント）の値を計算すれば、３次元物体像の形状を特徴づけるパラメータ（物体像を近似する楕円体の軸方向や軸半径など）を元の画像から直接計算するよりもずっと効率的に得ることができる。

ここでは、このような STL データの処理用に中野が開発しているプログラム群のうちの以下のものを紹介する（蛇足ながら、これらのうちで３次元画像に関わるものと楕円体近似に関連したプログラム群で用いているアルゴリズムは、論文としては未公表だが、他に例を見ない効率的なものではないかと思っている）。

STL データのチェックのためのプログラム群
STL データの加工やファイル形式の変換のためのプログラム群
STL データが表している物体像の鳥瞰図の描画に関連したプログラム群
３次元画像中の物体像の STL データの抽出とその再画像化のためのプログラム群
STL データが表す物体像の楕円体近似に関連したプログラム群

また、これらのプログラム群と一緒に使えば有用だと思われるこれらと相補的なフリーの STL データ処理用のソフトウェアについてもいくつかを紹介する。

STL 形式のファイルの構成

STL 形式のファイルの構成には ASCII、binary、そして、color STL 形式の３種類のものがある。ASCII STL 形式は STL データのファイルとして基本的なもので、これだけが STL 形式だと主張するソフトウェアもある。また、binary STL 形式は標準として広く認められているが、それから拡張された color STL 形式は事実上の標準とは言え非標準である。 color STL 形式では binary STL 形式で詳細が未定義だったそれぞれの三角形の属性（flag；後述）の値をその色の情報の保持に用いている。これらのファイルのデータ構造はまったく同じなので、 color STL 形式を知らないソフトウェアでもそのファイルを（三角形それぞれの flag の値を解釈せずに無視して） binary STL 形式のものとして処理できるはずである。後で紹介するフリーのソフトウェア VisCAM View と admesh および中野が書いたプログラム群はこれらすべての STL 形式のファイルを処理できる。このような３種類の STL 形式のファイルの詳細は以下の通りである。

ASCII STL 形式のファイル

ASCII テキストで STL データが書き込まれているファイル。 STL データの主な構成要素である前記の 12 個の単精度浮動小数点数の値（nx ～ z3）が以下のようにキーワード（小文字で表記した solid など）を交えてプログラミング言語風に記述されている。

```
solid  NAME
 ―――――――――――――――――――――――――――――――――――― 繰り返しはここから
 facet  normal  nx  ny  nz
  outer  loop
    vertex  x1  y1  z1
    vertex  x2  y2  z2
    vertex  x3  y3  z3
  endloop
 endfacet
 ―――――――――――――――――――――――――――――――――――― 繰り返しはここまで
 ...（上記の facet ～ endfacet の記述を三角形の個数分繰り返す）
endsolid  NAME
```

上記の NAME は３次元物体像の名前を表す任意長の ASCII 文字列で、記述しなくてもかまわない。中野が書いたものを含む多くのソフトウェアは solid や endsolid から行末までの文字を読み飛ばすようになっている。これら以外の ASCII STL 形式のファイル上の記述の仕方は基本的にはフリーフォーマットで、キーワード（大文字表記でもよい）や数値（C 言語や FORTRAN で単精度浮動小数点数の表現に許されている文字列）を空白、タブもしくは改行コードで区切って記述すればよい。ただし、大文字表記のキーワードやタブコードの使用を禁じていたり、上記のような構成の行（本来はインデントの有無やその段下げ量は任意）しか受け付けないソフトウェアもある。後で紹介する VisCAM View、admesh および中野のプログラム群はこのような制限なしのフリーフォーマットの ASCII STL 形式のファイルを処理できる。

binary および color STL 形式のファイル

いずれも STL データに関する任意のコメントの記述と STL データを構成する三角形の総数に続いて、三角形それぞれを特定する 12 個の単精度浮動小数点数と（STL データにとって本質的なものではない）三角形ごとの属性の値（ここでは flag と呼ぶ）が書き込まれているバイナリファイル。これらのデータはファイル上で以下の順に並んでいる。

byte 0 〜 79	：コメントの記述
byte 80 〜 83	：三角形の総数（N）
――――――――――――――――――――――――――――――――	繰り返しはここから
byte 84 + n × 50 + 0 〜 3	：n 番目の三角形の nx の値
byte 84 + n × 50 + 4 〜 7	：n 番目の三角形の ny の値
byte 84 + n × 50 + 8 〜 11	：n 番目の三角形の nz の値
byte 84 + n × 50 + 12 〜 15	：n 番目の三角形の x1 の値
byte 84 + n × 50 + 16 〜 19	：n 番目の三角形の y1 の値
byte 84 + n × 50 + 20 〜 23	：n 番目の三角形の z1 の値
byte 84 + n × 50 + 24 〜 27	：n 番目の三角形の x2 の値
byte 84 + n × 50 + 28 〜 31	：n 番目の三角形の y2 の値
byte 84 + n × 50 + 32 〜 35	：n 番目の三角形の z2 の値
byte 84 + n × 50 + 36 〜 39	：n 番目の三角形の x3 の値
byte 84 + n × 50 + 40 〜 43	：n 番目の三角形の y3 の値
byte 84 + n × 50 + 44 〜 47	：n 番目の三角形の z3 の値
byte 84 + n × 50 + 48 〜 49	：n 番目の三角形の flag の値
――――――――――――――――――――――――――――――――	繰り返しはここまで

...（n = 0 〜 N − 1 について上記の 50 bytes 長の記述を繰り返す）

ファイルの先頭にあるコメントの記述では ASCII 文字だけが許されており、合計で 80 bytes 長になっていなければならない（それよりも短い場合は ASCII コードの ’\0’ で残りを補填すればよい）。これに続くバイナリデータはいずれも Intel/DEC 形式（little-endian）の byte の並び（byte sequence）でファイルに書き込まれており、三角形の総数（N）は 4 bytes 長の符号なし整数、単精度浮動小数点数（nx 〜 z3）はいずれも 4 bytes 長の IEEE 754 floating point standard でコード化された値である。

三角形ごとの記述の最後の 2 bytes 長の符号なし整数の flag の値は binary STL 形式では未定義である（color STL 形式の値に合わせて 0 を書き込むとよい）。 color STL 形式ではそれは以下の bit 構成で三角形それぞれの色の情報を保持している。

bit 15	：以下の色の情報の有無に応じて 1 もしくは 0 とする。
bit 10 〜 14	：三角形の色の R（赤）成分の強度（0 〜 31）
bit 5 〜 9	：三角形の色の B（緑）成分の強度（0 〜 31）
bit 0 〜 4	：三角形の色の G（青）成分の強度（0 〜 31）

このように color STL 形式は三角形それぞれに対して成分ごとに 32 階調の強度（合計すると 32768 種類）の色の情報を保持できる。ただし、中野が書いたプログラム群のいくつかはこれらが 256 階調（16777216 色）であるかのように取り扱う。

なお、中野のプログラム群の一部では ASCII STL 形式のテキストデータはもとより binary もしくは color STL 形式のバイナリデータも標準入出力を通して読み書きできるようにしてあるが、 Windows の DOS 窓でそれが正常に行われるかどうかは不明である（DOS 窓ではバイナリデータのパイプラインもしくはファイルリダイレクション処理が正常に行われない可能性がある）。

プログラムのインストール

ここでは中野が書いたプログラム群のインストールの方法だけを説明する。それ以外のフリーのソフトウェア（VisCAM View、plyview および admesh）のインストール法などについては後に記すこれらそれぞれのホームページなどを参照していただきたい。

中野が書いたプログラム群（および Anthony D. Martin が書いた admesh）のインストールに必要なファイルは下記の TAR 形式書庫ファイル stl.tar にまとめてアップロードしてある。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl.tar
https://www.gsj.jp/researches/openfile/openfile2006/openfile0448.html

これを適当なディレクトリにダウンロードして書庫ファイル処理ソフトウェアで展開する。UNIX（Cygwin）環境の場合は以下のようにキー入力すれば stl.tar を展開できる。

```
tar  xvf  stl.tar
```

stl.tar から展開されたディレクトリ src/ にあるプログラムのソースファイルはすべて、標準ライブラリ関数だけを使った C 言語で書かれている。 UNIX 環境の場合は src/ に移動した後に以下を入力すればプログラムすべてのインストールが自動で行われる。

```
make  install
```

これにより gcc（GNU-C コンパイラ）を用いたソースファイルのコンパイルが行われ、その結果得られた実行ファイルすべてがディレクトリ bin/ にコピーされるはずである。ただし、コンパイルの設定などの変更のために src/ にあるファイル Makefile の修正が必要になるかもしれない。

Windows の DOS 窓（コマンドプロンプト）用のプログラムのインストール法は使用する C 言語の処理系ごとに大きく異なるので、 Makefile の記述を参考に各自で対応していただきたい。ここでは src/ にある Makefile をそのまま使って MinGW（Minimalist GNU for Windows）の gcc でコンパイルした実行ファイルをディレクトリ exe/ に格納しておいた。

なお、exe/ に入っている DOS 窓用の実行ファイルは Cygwin の上でも起動可能である。 Cygwin の gcc でコンパイルした実行ファイルはデータファイルの入出力がかなり遅いので、 MinGW の gcc で得た DOS 窓用のものを使った方がよいかもしれない。

個々のプログラムについて

ここで紹介するプログラムのうち VisCAM View と plyview は Windows 専用であり、 GUI（Graphical User Interface）で操作できる。これに対して、admesh は UNIX 用（Windows の DOS 窓でも動作可すると思われるが確認していない）、また、中野の作品は UNIX 環境と DOS 窓の両方で実行可能だが、端末（コンソール）からの起動を前提とした単純なユーザーインターフェイスしか持たない。

これらのプログラムの概略を以下の一覧表に示した。また、これらそれぞれの機能や起動法などのより詳しい説明はその後に記した通りである。

STL データ処理用のプログラムの概略

フリーの STL データ処理用のソフトウェア	
VisCAM View	Marcam engineering 社が配布していた RP データ処理用フリーソフトウェア。
plyview	Cyberware 社が配布している PLY 形式データ表示用フリーソフトウェア。
admesh	Anthony D. Martin が書いた STL データのチェックなどのためのプログラム。
STL データのチェックのためのプログラム群	
stl_dmp	STL データの三角形の頂点の座標値を ASCII 表示する。
stl_dmp_C	同上。ただし、STL データの三角形それぞれの色情報も表示する。
stl_nmm	STL データが表す物体像の座標値の値域を調べる。
stl_chk	STL データの整合性を Gauss の発散定理を用いて調べる。
stl_nav	STL データが表す物体像の表面積や体積を調べる。
stl_fc	異なる色情報を持つ STL データの三角形それぞれの個数と総面積を調べる。
stl_ac	指定した色情報を持つ STL データの三角形の面積の出現頻度分布を調べる。
STL データの加工やファイル形式の変換のためのプログラム群	
stl_resize	STL データが表す物体像のスケールを変える。
stl_shift	STL データが表す物体像を平行移動する。
stl_affine	STL データを構成する座標値の affine 変換を行う。
stl_a2b	ASCII STL 形式のデータを binary STL 形式に変換する。
stl_b2a	binary STL 形式のデータを ASCII STL 形式に変換する。
stl_color	STL 形式のデータの色情報を一括処理する。
stl_merge	複数の STL データファイルをひとつにまとめる。
stl_ply	STL 形式のデータを plyview 用の PLY 形式に変換する。
stl_ply_big	同上。ただし、Intel 系 CPU 用のプログラム。
stl_ply_zcp	STL 形式のデータを PLY/ZCP 形式に変換する。
stl_zcp	同上。ただし、物体像表面を覆う三角形の頂点のデータを統合する。
zcp_stl	PLY/ZCP 形式のデータを color STL 形式に変換する。
STL データが表している物体像の鳥瞰図の描画に関連したプログラム群	
stl_bev	STL データが表す物体像のグレースケールの鳥瞰図画像を作成する。
stl_bev_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_bev_C	STL データが表す物体像のカラーの鳥瞰図画像を作成する。
stl_bev_C_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_bev_wf	物体表面を線画で表した白黒の鳥瞰図画像を作成する。
stl_bev_wf_nf	同上。ただし、図上に座標軸と平行な枠を描かない。
stl_bev_GIF	物体像のグレースケールの鳥瞰アニメーションを作成する。
stl_bev_GIF_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_bev_C_GIF	物体像のカラーの鳥瞰アニメーションを作成する。
stl_bev_C_GIF_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_ar	軸回転を加えた物体像のグレースケールの鳥瞰図画像を作成する。
stl_ar_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_ar_C	軸回転を加えた物体像のカラーの鳥瞰図画像を作成する。
stl_ar_C_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
stl_ar_wf	軸回転を加えた物体像を線画で表した白黒の鳥瞰図画像を作成する。
stl_ar_wf_nf	同上。ただし、図上に座標軸と平行な枠を描かない。
stl_ar_GIF	軸回転を加えた物体像のグレースケールの鳥瞰アニメーションを作成する。
stl_ar_GIF_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。

stl_ar_C_GIF	軸回転を加えた物体像のカラーの鳥瞰アニメーションを作成する。
stl_ar_C_GIF_SS	同上。ただし、物体像表面の局所的なスムーシングを行う。
t2g_movie	複数の TIFF 画像をまとめたアニメーション用 GIF 画像を作成する。
t2g_movie_CM	同上。ただし、フルカラー形式の TIFF 画像を取り扱えない。
t2g_movie_gray	同上。ただし、グレースケールの TIFF 画像しか取り扱えない。
gif_movie	同上。ただし、複数の GIF 画像をまとめる。
g2t_split	アニメーション用 GIF 画像上の複数の画像を個別の TIFF 画像に分離する。
g2t_split_RGB	同上。ただし、分離する TIFF 画像を RGB フルカラー形式にする。
gif_split	同上。ただし、個別の GIF 画像に分離する。
gif_area	GIF 画像上の前景の領域のひろがりを調べる。
gif_area_nbg	GIF 画像上の背景ではない領域のひろがりを調べる。
gif_trim	GIF 画像のトリミングを行う。
３次元画像中の物体像の STL データ抽出とその再画像化のためのプログラム群	
si_stl_A	３次元画像から物体像を表す ASCII STL 形式のデータを作成する。
si_stl_B	３次元画像から物体像を表す binary STL 形式のデータを作成する。
si_stl_C	３次元画像から物体像を表す color STL 形式のデータを作成する。
stl_si	STL データが表す物体像の内部を塗りつぶした３次元２値画像を作成する。
STL データが表す物体像の楕円体近似に関連したプログラム群	
stl_of	STL データが表す物体像を近似する３軸不等楕円体のパラメータを決める。
stl_of_oblate	同上。ただし、楕円体の短、中、長軸の順でパラメータを決める。
of_stl_ih	正 20 面体を元にして３軸不等楕円体を表す STL データを作成する。
of_stl_oh	正８面体を元にして３軸不等楕円体を表す STL データを作成する。
of_stl_th	正４面体を元にして３軸不等楕円体を表す STL データを作成する。

フリーの STL データ処理用のソフトウェア

VisCAM View

起動法

最初に Windows でプログラム名を指定して起動し、プログラム内の Options メニューの File Options で STL や PLY/ZCP などの読み込むべきデータファイルの形式を登録しておけば、その後は Explorer でデータファイル名をクリックするなどして起動できる。

概略説明

Marcam engineering 社（ドイツ）が配布していた RP（rapid prototyping；３次元造形）データ処理用のフリーソフトウェア（売り物だった VisCAM RP の機能を落としたプログラム）。同社による開発は終了したが、現在でも最終版の ver. 5.2（Build 8600）をインターネット上の複数の場所（自分で探して下さい）からダウンロードできる。 STL（color STL も可）や PLY/ZCP などの各種形式の RP データの鳥瞰図表示（+ α）に最適なプログラムである。

plyview

起動法

Windows の Explorer で PLY 形式のファイル名をクリック、もしくは、DOS-prompt で

```
plyview {−v} PLY_file
plyview −h
```

のように入力して起動する。

概略説明

Cyberware 社（USA）が配布しているフリーのソフトウェアパッケージに含まれる PLY 形式データの表示用プログラム（このパッケージには複数のデータ形式変換用プログラムなども含まれている）。下記のアドレスからダウンロードできる。

http://www.cyberware.com/products/software/plyview.html

plyview は多量の三角形で表された物体像の鳥瞰図を表示できるが、鳥瞰図の視点の変更などをキー入力で指定するため使いにくい。その詳細は下記の説明書などをご覧下さい。

http://www.cyberware.com/products/software/plyviewQS.html

admesh

起動法

admesh {options} STL_file

admesh −−help

概略説明

Anthony D. Martin が作成した STL データの整合性（三角形による物体像の被覆の完全性）のチェックとその自動修復（ただし、後述するプログラム si_stl_[A,B,C] が出力したデータに対しては整合性に問題がなくても間違って修復されることがあるので注意が必要）、座標値のスケール変換・回転・平行移動や各種形式のファイル出力などの多様な機能を持つプログラム。１番目の起動法で指定できるこのような機能については２番目の起動法で表示される説明などをご覧下さい。また、下記の admesh のホームページもご覧下さい。

http://www.varlog.com/admesh-htm

STL データのチェックのためのプログラム群

stl_dmp および stl_dmp_C

起動法

stl_dmp STL_files

stl_dmp_C STL_files

機能と使用法

どちらのプログラムも指定されたファイル（複数も可）をスキャンし、それぞれのファイル名を表示した後にその中の STL データを構成するすべての三角形ごとに、

頂点０の座標値（x、y、z 成分の順）
頂点１の座標値（x、y、z 成分の順）
頂点２の座標値（x、y、z 成分の順）

の９個の数値（ASCII 文字）を空白区切りの１行で標準出力に書き出す（ただし、これらの三角形の頂点の番号は STL データが表している物体像の外部から見たときに反時計まわりの順になっている）。 stl_dmp_C ではそれに続けて、色情報を持つ三角形に対して

色の R、G、B 成分の強度の値（０〜 255 の値域の３個の整数値）

を空白コード区切りで出力する（色情報なしの三角形に対してはこの出力はない）。なお、ファイル名として "−" を指定すると標準入力からデータの読み込みが行われる。

stl_nmm

起動法

stl_nmm STL_files

機能と使用法

指定されたファイル（複数も可）をスキャンし、

ファイル名
それに格納されている STL データを構成する三角形の総数
それらの頂点の座標値の最小値（x、y、z 成分それぞれの値）
それらの頂点の座標値の最大値（x、y、z 成分それぞれの値）

の８項目のデータをタブコード区切りの ASCII 文字の１行にまとめて標準出力に書き出す。ファイル名に "−" を指定すると標準入力から読み込みが行われる。

stl_chk

起動法

stl_chk STL_files

機能と使用法

指定されたファイル（複数も可；ファイル名に "−" を指定すると標準入力からデータを読み込む）をスキャンし、その STL データの整合性を Gauss の発散定理に基づいてチェックするためのデータを出力する。すなわち、このプログラムがそれぞれのファイルに対してタブコード区切りの１行にまとめて標準出力に書き出す

ファイル名
それに格納されている STL データを構成する三角形の総数
三角形の面積要素ベクトルの x 成分の総和
三角形の面積要素ベクトルの y 成分の総和
三角形の面積要素ベクトルの z 成分の総和
三角形の面積要素ベクトルの絶対値の総和（STL データが表す物体像の表面積）

の６項目の ASCII 文字のデータのうちの面積要素ベクトルの成分の総和の３個の値は三角形が閉じた領域（STL データで表現したい物体像）の表面を過不足なく覆い尽くしている場合にすべて０になる。

Gauss の発散定理について

３次元領域 V におけるベクトル関数 $\vec{f} \equiv (f_x, f_y, f_z)$ の発散の体積積分は Gauss の発散定理により V を取り囲む曲面 S 上での面積分に変換できる。すなわち、

$$\int_S \vec{f} \cdot d\vec{S} = \int_V \nabla \cdot \vec{f}\, dV$$

ここで $d\vec{S} \equiv (dS_x, dS_y, dS_z)$ は S 上の面積要素ベクトル、 dV は V 内の体積要素、そして、$\nabla \cdot \vec{f} \equiv \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y} + \frac{\partial f_z}{\partial z}$ は $\vec{f}$ の発散である。この関係式は任意のベクトル関数に対して成り立つので、 $\vec{f}$ として座標値に依存しない定数の成分を持つベクトルを与えることにより

$$\int_S dS_x = 0、\ \int_S dS_y = 0 \ \text{かつ}\ \int_S dS_z = 0$$

が得られる。すなわち、領域 V が曲面 S に完全に囲まれているなら S の面積要素の各成分の積分（総和）はすべて０になる。 STL データ中の三角形は（厳密に）面積分の面積要素と見なせるので、これらの関係式を用いて STL データ中の三角形による物体像の被覆の状況をチェックできる。

なお、ついでながら、例えば $\vec{f} = \frac{1}{3}(x, y, z)$ とすると

$$\frac{1}{3}\int_S (x\, dS_x + y\, dS_y + z\, dS_z) = \int_V dV = \text{３次元領域 } V \text{ の体積}$$

となる。つまり、表面を覆い尽くす三角形の情報だけから物体像の体積を計算できる。後述するプログラム stl_nav ではこの式を用いて STL データから物体像の体積を計算している。さらに、後述するプログラム stl_of などでは Gauss の発散定理から得られるもう少し複雑な関係式を利用して物体像の体積積分（モーメントの値）を計算している。

stl_nav

起動法

stl_nav STL_files

機能と使用法

指定されたファイル（複数も可；ファイル名に "−" を指定すると標準入力からデータを読み込む）をスキャンし、それぞれの

ファイル名
それに格納されている STL データを構成する三角形の総数
STL データが表す物体像の表面積（三角形の面積要素ベクトルの絶対値の総和）
STL データが表す物体像の体積

の４項目のデータをタブコード区切りの ASCII 文字の１行にまとめて標準出力に書き出す。なお、このプログラムは stl_chk のところで説明した Gauss の発散定理を用いて STL データが表す物体像の体積を計算している。

stl_fc

起動法

stl_fc STL_file

機能と使用法

指定されたファイル（"−" の指定で標準入力からデータを読む）をスキャンし、異なった色情報を持つ三角形ごとの

三角形の色情報をあらわす 16 bits 長の整数値（flag）
色の R、G、B 成分の強度の値（０～ 255 の値域の３個の整数値）
その色情報を持つ三角形の個数
その色情報を持つ三角形の面積の総和

を調べてタブコード区切りの行として標準出力に書き出す。ただし、色情報を持たない三角形の flag の値は０であり、その場合には色成分の強度の値を出力しない。また、三角形の面積の総和は STL データの座標値の単位で測った値である。

stl_ac

起動法

stl_ac STL_file bin_size
stl_ac STL_file flag bin_size
stl_ac STL_file R G B bin_size

機能と使用法

指定されたファイル（"−" の指定で標準入力からデータを読む）をスキャンし、STL データを構成する三角形の面積の出現頻度分布（ヒストグラム）を調べる。その際、ヒストグラム

の柱（ビン）の幅はパラメータ bin_size で指定した値（STL データの座標値で測った面積と同じ単位の値）となる。１番目の方法で起動した場合は STL データのすべての三角形を対象に面積のヒストグラムを調べる。２番目の方法ではパラメータ flag の値に相当する色情報を持つ三角形だけを、また、３番目の起動法では０～ 255 の値域の整数値のパラメータ R、G、B で指定された R、G、B 成分の強度を持つ色をした三角形だけを調べる。このようなヒストグラムのうちで三角形が存在する（高さが０でない）ビンについてのみ、

ビン番号（０以上の整数値）
ビンの下端の面積の値
ビンの上端の面積の値
そのビンに落ちる面積を持つ三角形の個数

の４つの値がタブコードで区切られた行としてビン番号の順で標準出力に書き出される。

STL データの加工やファイル形式の変換のためのプログラム群

stl_resize

起動法

stl_resize org_STL ratio new_STL
stl_resize org_STL Rx Ry Rz new_STL

機能と使用法

指定されたファイル org_STL（"−" を指定すると標準入力からデータを読む）に格納されている STL データのすべての三角形の頂点の座標値の単純なスケール変換を行う。１番目の方法で起動すると座標値の３成分に同じ倍率 ratio を乗じる。また、２番目の起動法では x、y、z 成分に異なる倍率 Rx、Ry、Rz を乗じる。これらの倍率は負の値でもよい（例えば Rx だけを負の値にすると結果の STL データは元のものの yz 平面に関する鏡像になる）。変換の結果は元のファイルの三角形の色情報とともに binary もしくは color STL 形式でファイル new_STL に格納される。

stl_shift

起動法

stl_shift org_STL Sx Sy Sz new_STL

機能と使用法

ファイル org_STL（"−" を指定すると標準入力からデータを読む）の STL データのすべての三角形の頂点の座標値の x、y、z 成分それぞれに指定した数値 Sx、Sy、Sz を加える（STL データが表す物体像を３次元空間で平行移動する）。変換の結果は元のファイルの三角形の色情報とともに binary もしくは color STL 形式でファイル new_STL に格納される。

stl_affine

起動法

stl_affine org_STL h0 hx hy hz v0 vx vy vz d0 dx dy dz new_STL

機能と使用法

ファイル org_STL（"−" を指定すると標準入力からデータを読み込む）の中の STL データを構成する三角形の頂点の座標値 (x, y, z) に対して、起動パラメータとして指定した任意の浮動小数点の係数値を用いた以下の変換（affine 変換）を行う。

$$\begin{pmatrix} h \\ v \\ d \end{pmatrix} = \begin{pmatrix} \mathrm{h0} \\ \mathrm{v0} \\ \mathrm{d0} \end{pmatrix} + \begin{pmatrix} \mathrm{hx}, & \mathrm{hy}, & \mathrm{hz} \\ \mathrm{vx}, & \mathrm{vy}, & \mathrm{vz} \\ \mathrm{dx}, & \mathrm{dy}, & \mathrm{dz} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

この変換で得られた新しい座標値 (*h*, *v*, *d*) からなる STL データを元の三角形の色の情報とともに binary もしくは color STL 形式でファイル new_STL に書き込む。

以下のパラメータ値を指定すれば、stl_affine を用いて前述の stl_resize と stl_shift のそれぞれのモノと等価な処理を行うことができる。

stl_resize： hx = Rx、vy = Ry、dz = Rz かつ h0 = hy = hz = v0 = vx = vz = d0 = dx = dy = 0
stl_shift： h0 = Sx、v0 = Sy、d0 = Sz、hx = vy = dz = 1 かつ hy = hz = vx = vz = dx = dy = 0

stl_a2b

起動法

stl_a2b ASCII_STL binary_STL

機能と使用法

ファイル ASCII_STL（"−" を指定すると標準入力からデータを読む）の中の（ASCII STL 形式の）STL データを binary STL 形式のファイル binary_STL にコピーする。

stl_b2a

起動法

stl_b2a binary_STL {format}

機能と使用法

ファイル binary_STL（"−" を指定すると標準入力からデータを読む）の中の（binary STL 形式の）STL データを ASCII STL 形式に変換して標準出力に書き出す。パラメータ format として C 言語の標準ライブラリ関数 printf() などで単精度浮動小数点数（float）の出力形式の指定に用いる文字列を与えれば、出力する ASCII STL データの座標値の表示桁数などを指定できる（例えば、後述の si_stl_[A,B,C] で作成したもののように、 STL データ中の座標値すべてが整数値の場合には format として "%g" を指定すればよい）。この指定を省略すると format として "%f" が用いられる。

stl_color

起動法

stl_color org_STL {R G B} new_STL

機能と使用法

このプログラムはファイル org_STL（"−" を指定すると標準入力からデータを読む）の中のすべての三角形の色情報をまとめて操作する。パラメータ R、G、B として色の R、G、B 成分の強度をそれぞれ０〜 255 の整数値で指定すればそれに応じた色情報が、また、これらの指定を省略すれば元のものから色情報を取り除いた STL データが color もしくは binary STL 形式でファイル newSTL に書き込まれる。

stl_merge

起動法

stl_merge STL_files STL_file

機能と使用法

指定された複数のファイル STL_files（これらそれぞれのファイル名として "−" を指定するとそのデータを標準入力から読み込む）の STL データをまとめ、 binary もしくは color STL 形式でファイル STL_file に書き込む。

stl_merge が作る STL データには複数の物体像が含まれることになるため、これは厳密には STL 形式として正しいものではない。中野が書いたプログラムでは問題は生じないが、それ

以外のソフトウェアでこれを正常に取り扱えないものがあるかもしれない。

stl_ply および stl_ply_big

起動法

stl_ply STL_file {R G B} PLY_file

stl_ply_big STL_file {R G B} PLY_file

機能と使用法

ファイル STL_file 中の STL 形式のデータを PLY 形式（後述；正確には Cyberware 社のプログラム plyview が受け付ける PLY 形式）のものに変換してファイル PLY_file に書き込む。ただし、STL_file もしくは PLY_file としてファイル名の代わりに "−" を指定すると標準入力もしくは標準出力に対してデータの読み書きが行われる。色情報を持たない三角形にはパラメータ R、G、B として０～ 255 の整数値で指定した R、G、B 成分の強度を持つ色が割り当てられる。また、これらの指定を省略すると白色（R = G = B = 255）に相当する色情報がその三角形を表現する PLY データに埋め込まれる。

stl_ply は用いている計算機の CPU に自然な byte の並び（byte sequence）で PLY 形式データ中のバイナリデータを書き出す。このデータは PLY 形式の仕様上の問題はないが、 Intel 系の CPU 上で stl_ply を走らせて得た場合にはプログラム plyview に受け付けてもらえない。この点を改善したプログラムが stl_ply_big である（Intel 系の CPU で走らせる場合はこちらをお使い下さい）。

PLY 形式について

PLY 形式（PLY polygon file format）は Stanford 大学で設計された３次元物体像を多面体で表現するためのデータ形式で、下記のホームページなどに説明がある。

http://www.cc.gatech.edu/projects/large_models/ply.html

http://www.cc.gatech.edu/projects/large_models/files/ply.tar.gz

上記の gzip 圧縮された TAR 形式書庫ファイル ply.tar.gz の中に Greg Turk によって書かれた PLY 形式の正式の仕様書らしきテキストファイル PLY_FILES.txt が格納されている。これによると PLY 形式の仕様はバイナリデータの byte の並びの機種依存性やユーザによるデータ構造の拡張などを許すかなり柔軟なものなので、それらすべてに対応したソフトウェアの実現は非常に困難である（例えば Cyberware 社が配布している plyview などのフリーのプログラム群はサブセットの PLY 形式ファイルしか取り扱えない）。 PLY 形式に関するこれ以上詳しい説明は上述のテキストファイル PLY_FILES.txt をご覧ください。

stl_ply_zcp および stl_zcp

起動法

stl_ply_zcp STL_file {R G B} ZCP_file

stl_zcp STL_file {R G B} ZCP_file

機能と使用法

ファイル STL_file の中の STL データを後述する PLY/ZCP 形式のデータに変換してファイル ZCP_file に書き込む。ただし、STL_file や ZCP_file として "−" を指定すると標準入力や標準出力に対してデータの読み書きが行われる。 STL データの三角形で色情報がないものに対してはいずれも０～ 255 の整数値のパラメータ R、G、B で３成分の強度を指定した色が、また、これらの指定を省略すると白色（R = G = B = 255）が割り当てられる。

stl_ply_zcp はデータをそのまま変換するのに対して stl_zcp は STL データの複数の三角形が共有している頂点の統合を行う。このため stl_zcp が出力するデータファイルのサイズは半減するが、それには stl_ply_zcp よりも長い処理時間を要する。

PLY/ZCP 形式について

PLY/ZCP 形式はホームページ

http://www.zcorp.com

に紹介されている Z corporation 社（USA）製の３次元プリンタ（３次元物体像の実体模型造形装置）用に設計された前述の PLY 形式のサブセットのデータ形式である。表現可能な色数が多いことを除いてこの形式のデータは color STL 形式のものと互換であり、また、前述のようにプログラム stl_zcp を使えばサイズが半減するので、 color STL 形式のものをこの形式のデータファイルに変換しておくと有用である。

PLY/ZCP 形式の正式の仕様書の有無は不明で解説書の類も皆無に近かった。中野は前述の PLY 形式の仕様書と物体像の形状の PLY/ZCP 形式のデータを出力する Java プログラム

http://protein3dprint.sourceforge.net

から PLY/ZCP 形式の仕様を下記の通りだと推定し、それを TACC/AIST に設置されていた Z corporation 社の３次元プリンタ Z406 を使ったカラーの実体模型の造形によって確認した。

```
ply
format binary_little_endian 1.0
element vertex N_v          ┐
property float x            │ N_v 個の多面体の頂点それぞれに対する
property float y            │ x、y、z 座標値のデータの型の宣言
property float z            ┘
element face N_f            ┐
property uchar red          │
property uchar green        │ N_f 個の多面体の面（三角形）それぞれに対する
property uchar blue         │ 面の表示色の red、green、blue 成分の強度値、
property list uchar int vertex_indicies │ 頂点の個数（三角形なので常に３）、および、
                            │ 頂点の番号のデータの型の宣言
end_header                  ┘ ここまで ASCII 文字のヘッダ
---------------------------   ここからバイナリデータ
多面体の頂点の x 座標値（float）  ┐
多面体の頂点の y 座標値（float）  │ この３個一組の値の記述を N_v 回繰り返す
多面体の頂点の z 座標値（float）  ┘
三角形の色の R 成分の強度（uchar）  ┐
三角形の色の G 成分の強度（uchar）  │
三角形の色の B 成分の強度（uchar）  │
3（三角形の頂点の個数；uchar）      │ この７個一組の値の記述を N_f 回繰り返す
三角形の１番目の頂点の番号（int）   │
三角形の２番目の頂点の番号（int）   │
三角形の３番目の頂点の番号（int）   ┘
```

PLY/ZCP 形式のファイルは多面体の形状を表現するデータの型などの宣言が行われているヘッダ部とそれに従ったデータが格納されているデータ部から構成される。 ASCII 文字からなるヘッダ部の行の構成やキーワード（ply など）の並びは（自然数の値 N_v および N_f を除いて）上に記した通りになっている必要がある。また、データ部のバイナリデータはヘッダ部の format の行の宣言に従った little endian（Intel/DEC 形式）の byte の並びになっており、これらの型を表す float は 4 bytes 長の IEEE 754 floating point standard で表した単精度浮動小数点数、 uchar と int は 1 もしくは 4 bytes 長の符号なし整数である。なお、データ部の後半で用いている三角形の頂点の番号は、データ部の前半で座標値を記述した順に多面体の頂点につけた番号 0 〜 $N_v - 1$ のことである。

zcp_stl

起動法

zcp_stl ZCP_file STL_file

機能と使用法

PLY/ZCP 形式のファイル ZCP_file の中のデータを color STL 形式に変換してファイル STL_file にコピーする。ファイル ZCP_file もしくは STL_file として "−" を指定すると標準入力もしくは標準出力に対してデータの読み書きが行われる。

STL データが表している物体像の鳥瞰図の描画に関連したプログラム群

stl_bev、stl_bev_SS、stl_bev_C および stl_bev_C_SS

起動法

stl_bev STL_files scale gamma bias bg fg

stl_bev_SS STL_files scale gamma bias bg fg

stl_bev_C STL_files scale gamma bias bgR bgG bgB fgR fgG fgB scR scG scB

stl_bev_C_SS STL_files scale gamma bias bgR bgG bgB fgR fgG fgB scR scG scB

標準入力から読み込むパラメータ指定行

lon lat TIFF_file

lon_base lat_base lon_step lat_step total format

機能と使用法

これらのプログラムはいずれも指定したファイル STL_files（複数も可）の中の STL データが表す物体像を任意の視線方向から眺めた正射図法（orthographic projection；後述）の鳥瞰図（bird's eye view map）の画像（正方形の画像；複数も可）を作成して TIFF 形式画像ファイルに格納する。ただし、プログラム名に "_SS" が付いているものは物体像の表面に局所的なスムーシングを施した鳥瞰図の画像を作成する（SS は surface smoothing の略のつもり；この処理は STL データを用いた物体像の表面の描画の後に画像処理的に行われ、少数の三角形で構成される像の場合には有効ではない）。また、"_C" 付きのプログラムは（color STL 形式データの色情報に基づいた）カラーの鳥瞰図を、そうでないものは（入力データが color STL 形式であってもその色情報を無視して）グレースケールの鳥瞰図を描画する。

起動時に指定したパラメータ scale が正の数値の場合、鳥瞰図画像の画素数は STL データの鳥瞰図投影後の座標値に scale を乗じたもの（スケーリングされた座標値）に見合った値となる（例えば STL データ中の座標値が幅１の値域に分布するなら、鳥瞰図画像の一辺は概ね scale 程度の画素数になる；どのような視線方向から眺めた場合でも物体像全体がおさまるように鳥瞰図画像の一辺の画素数が自動的に決められる）。これに対して、scale に負の整数値を指定するとその絶対値が鳥瞰図画像の一辺の画素数となる。

パラメータ gamma（非０の任意の数値）と bias（０～ 255 の数値）は物体像表面の陰影の程度を決める際の γファクタとバイアスの値で、その詳細は後で説明する。通常は gamma に１～２、また、bias に 16 ～ 64 程度の値を指定すればよい（gamma を大きな値にすると陰影の差が広がり、 bias が０なら影の部分は真っ黒に、255 なら影がなくなる）。

グレースケールの鳥瞰図用プログラムのパラメータ bg と fg はそれぞれ物体像以外の背景の部分と図上に描く座標軸に平行な枠の輝度（画素値）を指定する０～ 255 の整数値である。ただし、fg には負の値の指定も許されており、その場合には枠を描かない。また、カラーの鳥瞰図用プログラムの bg[R,G,B] と fg[R,G,B] も同様で、背景と枠に塗る色の３成分の強度をそれぞれ０～ 255 の整数値で指定する（この場合も fg[R,G,B] の値のいずれかが負であるなら枠を描かない）。さらに、カラーの鳥瞰図の場合には色情報を持っていない三角形に塗るべき色の３成分の強度（０～ 255 の整数値）をパラメータ sc[R,G,B] でそれぞれ指定する

必要がある（sc は surface color の略のつもり）。

以上のパラメータを与えて起動するといずれのプログラムも鳥瞰図の視線方向とその画像を格納するファイル名の入力待ちになる。これらは前記の２種類の記述法のいずれかで標準入力から１行づつ指定する（２種類の記述法を混ぜて複数行指定してもよい）。

標準入力からのパラメータ指定行の１番目の記述法は鳥瞰図の視線方向（経度 lon と緯度 lat ；いずれも度単位の任意の数値）とその画像を格納するファイル名（TIFF_file）を逐一指定する場合に用いる。ただし、経度 lon と緯度 lat の定義は地球に対するものと同じで、 x 軸方向を指定する場合は lon = lat = 0、 y 軸方向なら lon = 90 かつ lat = 0、また、 z 軸方向なら lat = 90 となる（z 軸方向の経度 lon は任意だが、lon = −90 とすると元の STL データの x および y 軸方向のそれぞれが画面表示した鳥瞰図画像の右および上方向になる；詳しくは後述する鳥瞰図で用いている投影法 = 正射図法の座標変換の式を見よ）。

パラメータ指定行の２番目の記述法は視線方向を連続的に変えた一連の鳥瞰図画像（自然数のパラメータ total で指定された枚数の鳥瞰図画像）を描画する場合に用いる。この場合のそれぞれの鳥瞰図の視線方向の経度 lon と緯度 lat は度単位の任意の数値のパラメータ lon_base、lat_base、lon_step、lat_step を用いて、

$$\left.\begin{aligned} \mathrm{lon} &= \mathrm{lon\_base} + \mathrm{lon\_step} \times \mathrm{index} \\ \mathrm{lat} &= \mathrm{lat\_base} + \mathrm{lat\_step} \times \mathrm{index} \end{aligned}\right\} \text{ ただし、 } \mathrm{index} = 0 \sim \mathrm{total} - 1$$

となる。そして、それぞれの鳥瞰図を格納する TIFF 画像のファイル名はパラメータ format で指定する。これは C 言語の標準ライブラリ関数 sprintf() などで 0 〜 total - 1 の個々の整数値を含む文字列（ファイル名）が得られるような format 指定の文字列で、例えば total = 100 として描画した 100 枚の鳥瞰図の画像ファイルすべてをディレクトリ "DIR"（あらかじめ作成しておく必要がある）の下に格納する場合は "DIR/%02d.tif" のように指定すればよい（"%02d" は整数値 0 〜 99 を２桁の文字列 "00" 〜 "99" に変換するための関数 sprintf() などの format 指定； TIFF 形式画像のファイル名なので拡張子 ".tif" を付けた）。

以上のようにして描画した鳥瞰図をおさめた TIFF 形式画像ファイルの画素データは LZW 圧縮されており、グレースケール画像では 8 bits 長である。また、カラー画像の場合には色の R、G、B 成分ごとに 8 bits 長の画素データを持つ RGB フルカラー形式になっている。

正射図法について

ここで紹介している鳥瞰図描画用のプログラムはすべて以下の座標変換式を用いた正射図法を採用している。

$$\begin{pmatrix} h \\ v \\ d \end{pmatrix} = \begin{pmatrix} -\sin(\mathrm{lon}), & \cos(\mathrm{lon}), & 0 \\ -\sin(\mathrm{lat})\cos(\mathrm{lon}), & -\sin(\mathrm{lat})\sin(\mathrm{lon}), & \cos(\mathrm{lat}) \\ -\cos(\mathrm{lat})\cos(\mathrm{lon}), & \cos(\mathrm{lat})\sin(\mathrm{lon}), & \sin(\mathrm{lat}) \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

ここで (x, y, z) は物体像上の点の座標値、 (h, v) は鳥瞰図上の点の座標値、また、d は鳥瞰図の投影面（平面）から測った物体像上の点の高さ（もしくは深さ）である。鳥瞰図描画にしばしば用いられる一点透視図法とは異なり、この図法では投影した物体像の歪み（遠近感）は生じない。そのため、３次元物体像に付随したある長さは鳥瞰図においてプログラムの起動パラメータ scale だけで決まる長さに変換される。

なお、ついでながら、鳥瞰図描画の際の隠面・隠線処理はいわゆる Z バッファアルゴリズムによって行っている（中野 司、情報地質、3、65 − 82、1992）。

物体表面の陰影付けについて

ここで紹介している鳥瞰図描画用のプログラムは物体像表面を覆う三角形ごとにその陰影の程度 $\delta = 0 \sim 1$ を以下のようにして決めている。

$$\delta = \begin{cases} \text{未定義} & \alpha < 0\text{：三角形が視線方向から見えない場合} \\ \beta + (1-\beta) \times \alpha^{1/\gamma} & \alpha > 0\text{：三角形が視線方向に面している場合} \end{cases}$$

ただし、α は三角形の法線方向と視線の方向のなす角の余弦値（cosine）、$\beta = 0$ 〜 1 は影の明るさを調節するためのバイアス値、γ は画像表示装置の輝度特性を補正するための γ ファクタである。グレースケールの鳥瞰図では $\delta \times 255$ に近い整数値を三角形の内部の点に対応する画素の輝度の値としている（そのための $\beta \times 255$ の値をパラメータ bias として指定する）。また、カラーの鳥瞰図では三角形ごとにその色の強度の 3 つの成分値に単一の δ を乗じた結果の成分値の色で塗っている（指定すべきパラメータの個数が増えるので、色の成分ごとに別個の β と γ から δ を計算するようにはしなかった）。

stl_bev_wf および stl_bev_wf_nf

起動法

stl_bev_wf STL_files scale

stl_bev_wf_nf STL_files scale

標準入力から読み込むパラメータ指定行

lon lat TIFF_file

lon_base lat_base lon_step lat_step total format

機能と使用法

これらのプログラムは指定したファイル STL_files（複数も可）中の STL データが表す物体像を任意の視線方向から眺めた正射図法の線画（wire frame；線で描かれるのは STL データを構成する三角形の辺）の鳥瞰図を描き、その画像（複数も可）を TIFF 形式画像ファイルに格納する。ただし、stl_bev_wf は物体像の線画に加えて座標軸に平行な枠を描き、stl_bev_wf_nf はそれを描かない。

これらのプログラムの使い方は概ね stl_bev などと同様である。すなわち、起動パラメータ scale で鳥瞰図画像（正方形）の一辺の画素数を指定し（scale の値の正負によりその画素数の決定法が異なる）、また、鳥瞰図の視線方向とその画像を格納する TIFF 形式画像ファイルの名前の指定（上記のように 2 種類の記述法があり、複数回の指定も可能）は起動後に標準入力から読み込まれる（詳細は stl_bev などの説明をご覧下さい）。

これらのプログラムは LZW 圧縮された 1 bit の画素値を持つ黒白画像を作成する。この画像上では描画した線を構成する画素の値は 1 で、画面表示するとそれが黒に表示されるような属性を持つ TIFF 形式画像ファイルとなっている。なお、stl_bev_wf などで描いた鳥瞰図の画像は stl_bev などでそれと同じ画像の画素数・視線方向を指定して描いたグレースケールもしくはカラーの鳥瞰図画像と重ね合わせることができる。

stl_bev_GIF、stl_bev_GIF_SS、stl_bev_C_GIF および stl_bev_C_GIF_SS

起動法

stl_bev_GIF STL_files scale gamma bias bg fg GIF_file

stl_bev_GIF_SS STL_files scale gamma bias bg fg GIF_file

stl_bev_C_GIF STL_files scale gamma bias （改行しない）

bgR bgG bgB fgR fgG fgB scR scG scB GIF_file

stl_bev_C_GIF_SS STL_files scale gamma bias （改行しない）

bgR bgG bgB fgR fgG fgB scR scG scB GIF_file

標準入力から読み込むパラメータ指定行

lon lat

lon_base lat_base lon_step lat_step total

機能と使用法

これらのプログラムはいずれも指定されたファイル STL_files（複数も可）中の STL データが表す物体像を任意の視線方向から眺めた正射図法の鳥瞰図を描き、それらすべての画像を GIF 形式画像ファイル GIF_file （"−" の指定で標準出力に書き込む）にまとめて格納する。この画像ファイルは Netscape Navigator などを使えばアニメーション表示可能である。先に説明したプログラム stl_bev_SS や stl_bev_C の場合と同様で、プログラム名に "_SS" が付いているものは物体像の表面に局所的なスムーシングを施し、また、"_C" 付きのプログラムはカラーの鳥瞰図（それ以外はグレースケールの鳥瞰図）を作成する。

鳥瞰図の画像を格納するファイル名 GIF_file 以外の起動時に指定すべきパラメータの意味は stl_bev などのものとまったく同じである（出力画像のファイル形式の違いを除けば、こちらのプログラムはそれらの名前から "_GIF" を取り除いた先に説明したプログラムと同じ処理を行う）。また、起動後に標準入力から読み込むパラメータ指定行に関しても同様で、stl_bev_GIF などに対しては鳥瞰図の視線方向だけを上記の 2 種類の記述法のいずれかで指定する。とにかく、stl_bev_GIF などを使うためには stl_bev などの説明を御一読下さい。

stl_bev_GIF などで作成した鳥瞰図の画像は正確には GIF87a 形式でファイルに格納される。すなわち、アニメーション表示の繰り返しの回数や画像ごとの表示遅延時間のデータはファイルに書き込まれていない。なお、カラー画像を出力する場合には GIF 形式の制限のために画像上の色の数を 256 以下に削減している。これは画像ごとに画素値の平均情報量をできる限り減らさない手法（中野 司、情報地質、5、187－210、1997）で行っているので、その影響は目視では識別できないハズである。

stl_ar、stl_ar_SS、stl_ar_C および stl_ar_C_SS

起動法

stl_ar STL_files xO yO zO lon lat scale gamma bias bg fg

stl_ar_SS STL_files xO yO zO lon lat scale gamma bias bg fg

stl_ar_C STL_files xO yO zO lon lat scale gamma bias （改行しない）
bgR bgG bgB fgR fgG fgB scR scG scB

stl_ar_C_SS STL_files xO yO zO lon lat scale gamma bias （改行しない）
bgR bgG bgB fgR fgG fgB scR scG scB

標準入力から読み込むパラメータ指定行

λ ϕ θ TIFF_file

λ ϕ θ_base θ_step total format

機能と使用法

stl_bev などが行う処理内容と同様でこれらのプログラムはいずれも指定したファイル STL_files（複数も可）中の STL データが表す物体像を指定した視線方向から眺めた場合の正射図法の鳥瞰図を描き、その画像（複数も可）を TIFF 形式の画像ファイルに格納する。プログラム名中の文字列 "ar" と "bev" を入れ替えたものがそれぞれ対応おり、プログラム名に "_SS" が付いているものは物体像の表面に局所的なスムーシングを施し、"_C" 付きのものはカラーの鳥瞰図（それ以外はグレースケールの鳥瞰図）を作成することも同様である。しかしながら、stl_bev などとは異なり、プログラム stl_ar などは指定したある視線方向から見た、指定した軸の回りに任意の角度の回転（axial rotation；これを表す座標変換の式については後述）を加えた物体像の鳥瞰図を描く。

起動時に指定すべきパラメータのうち gamma 以降のものの意味は stl_bev などのものとまったく同じなのでそちらの説明をご覧下さい。また、（stl_bev などでは正負いずれの値も許されていた）パラメータ scale は stl_ar などでは正の数値の指定だけが有効で、それは鳥瞰図画像のスケーリングファクタになる（STL データの座標値の単位で 1 だけ離れた鳥瞰図上の

２点は画像上で画素幅の scale 倍だけ離れた２画素にそれぞれ投影される）。パラメータ lon と lat は（stl_bev などでは標準入力から指定していた）鳥瞰図の視線方向の度単位の経度と緯度の数値で、これらの定義は stl_bev などのものとまったく同じである。パラメータ xO、yO および zO は物体像の回転軸が通る点の座標値の３成分である。この点は STL データが表す物体像の内部（その座標値の範囲は前述のプログラム stl_nmm で調べることができる）にあることが望ましい（そうしないと鳥瞰図画像の大半が物体像のない背景になる）。

stl_ar などは物体像の回転軸の方向、その回りの回転角および軸回転を加えた物体像の鳥瞰図を格納するファイル名の指定を標準入力から行単位で読む。その記述法には上記の２種類のものがあり、それらを順不同で複数行づつ混ぜた指定が可能である。いずれの記述法でもパラメータ λ と ϕ として与えた度単位の数値で回転軸の方向を示す経度と緯度（これらの定義は視線方向の場合と同じである）を指定する。１番目の記述法で指定した場合、 stl_bev などはこの軸の回りをその方向から見て反時計回りに角度 θ（単位は度）だけ回転させた物体像の鳥瞰図を描き、その画像を TIFF 形式画像ファイル TIFF_file に格納する。また、２番目の記述法は軸の回りの回転角を連続的に変えた一連の鳥瞰図（自然数のパラメータ total で指定した枚数の鳥瞰図）を描く際に用いる。すなわち、この場合の回転角 θ は度単位の数値のパラメータ θ_base と θ_step を用いて、

$\theta = \theta\_\mathrm{base} + \theta\_\mathrm{step} \times (0 \sim \mathrm{total} - 1)$

となる。そして、それぞれの鳥瞰図はパラメータ format として与えた文字列（C 言語の標準ライブラリ関数 sprintf() 用の format 指定用の文字列）によって決まる 0 ～ total − 1 の個々の整数値を含んだ名前の TIFF 形式の画像ファイルに格納される。このような一連の鳥瞰図画像のファイル名の指定法は stl_bev などのものと同じであり、また、それらの TIFF 形式画像ファイルの特徴（画素値の bits など）に関しても同様なので、その詳細については stl_bev などのために書かれた説明や例をご覧下さい。

なお、stl_bev などのものとは異なり、 stl_ar などが作成した鳥瞰図画像は正方形になるとは限らない。また、パラメータ scale と視線方向に同じ値を指定した場合には stl_ar などが作成する鳥瞰図は stl_bev などのものよりも一回り大きな画像（画素数が多い画像）になる。これは、任意の軸回転に備えて stl_ar などが画像の余白（背景の部分）を広い目に確保するためである。

軸回転の座標変換の式について

ここで紹介しているプログラムは物体像の軸回転に以下の座標変換の式を用いている。

$$\Lambda\Phi = \begin{pmatrix} -\sin\lambda, & -\cos\lambda\sin\phi, & \cos\lambda\cos\phi \\ \cos\lambda, & -\cos\lambda\sin\phi, & \sin\lambda\cos\phi \\ 0, & \cos\phi, & \sin\phi \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \Lambda\Phi \begin{pmatrix} \cos\theta, & -\sin\theta, & 0 \\ \sin\theta, & \cos\theta, & 0 \\ 0, & 0, & 1 \end{pmatrix} (\Lambda\Phi)^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

ただし、(x, y, z) と (x', y', z') はそれぞれ軸回転前後の座標値である。

stl_ar_wf および stl_ar_wf_nf

起動法

stl_ar_wf STL_files xO yO zO lon lat scale

stl_ar_wf_nf STL_files xO yO zO lon lat scale

標準入力から読み込むパラメータ指定行

λ ϕ θ TIFF_file

λ ϕ θ_base θ_step total format

機能と使用法

これらのプログラムは指定したファイル STL_files（複数も可）の中の STL データが表す物体像に任意の軸回転を加え、回転後の像に対する指定した視線方向から眺めた正射図法の線画（STL データ中の三角形の辺を示す線画）の鳥瞰図画像（複数も可）を作成し、TIFF 形式画像ファイルに格納する。ただし、stl_ar_wf は物体像の線画に加えて座標軸と平行な枠を描き、 stl_ar_wf_nf はそれを描かない。

これらのプログラムの使い方は概ね stl_ar などと同様である。すなわち、起動パラメータ xO、yO、zO で回転軸が通る点の座標値、 lon と lat で鳥瞰図の視線方向、そして、scale で鳥瞰図画像のスケールを指定する。また、起動後に上記の２種類の記述法で標準入力から軸回転の軸の方向、その回りの回転角および回転後の物体像の鳥瞰図を格納する TIFF 画像ファイルの名前を指定する（これらについての詳細は stl_ar などの説明をご覧下さい）。

前述の stl_bev_wf などで作成されるものと同様で stl_ar_wf と stl_ar_wf_nf が作成する鳥瞰図は LZW 圧縮された白黒画像である。また、stl_ar_wf などと stl_ar などで同じパラメータ（視線方向、scale および軸回転の角度）を指定して描いた２枚の鳥瞰図画像は完全に重ね合わせることができる。

stl_ar_GIF、stl_ar_GIF_SS、stl_ar_C_GIF および stl_ar_C_GIF_SS

起動法

stl_ar_GIF STL_files xO yO zO lon lat scale gamma bias bg fg GIF_file
stl_ar_GIF_SS STL_files xO yO zO lon lat scale gamma bias bg fg GIF_file
stl_ar_C_GIF STL_files xO yO zO lon lat scale gamma bias （改行しない）
bgR bgG bgB fgR fgG fgB scR scG scB GIF_file
stl_ar_C_GIF_SS STL_files xO yO zO lon lat scale gamma bias （改行しない）
bgR bgG bgB fgR fgG fgB scR scG scB GIF_file

標準入力から読み込むパラメータ指定行

λ ϕ θ
λ ϕ θ_base θ_step total

機能と使用法

これらのプログラムはいずれも指定したファイル STL_files（複数も可）の中の STL データが表す物体像に一連の軸回転を加え、回転後の像の指定した視線方向から眺めた正射図法の鳥瞰図画像を作成して GIF 形式のファイル GIF_file にまとめて格納する（"−" を指定すると標準出力に書き込む）。この画像ファイルは Internet Explorer などを使えばアニメーション表示可能である。前述のプログラム stl_ar_SS や stl_ar_C の場合と同様で、プログラム名に "_SS" が付いているものは物体像の表面に局所的なスムーシングを施し、また、"_C" 付きのものはカラーの鳥瞰図（それ以外はグレースケールの鳥瞰図）を作成する。

鳥瞰図の画像を格納するファイル名 GIF_file 以外の起動時に指定すべきパラメータの意味は stl_ar などのものとまったく同じである（出力画像のファイル形式の違いを除けば、こちらのプログラムは名前から "_GIF" を取り除いたプログラムと同じ処理を行う）。また、起動後に標準入力から読み込むパラメータ指定行に関しても同様で、 stl_ar_GIF などに対しては回転軸の方向とその回りの回転角だけを上記の２種類の記述法のいずれかで指定する。とにかく、stl_ar_GIF などを使うためには stl_ar などの説明を御一読下さい。

stl_ar_GIF などが作成した鳥瞰図画像の詳細（GIF 形式の正式名称やカラー画像における色数について）は stl_bev_GIF などのものと同様なので、そちらの説明も御一読下さい。

t2g_movie、t2g_movie_CM および t2g_movie_gray

起動法

t2g_movie directory name_file GIF_file

t2g_movie_CM directory name_file GIF_file

t2g_movie_gray directory name_file GIF_file

機能と使用法

これらのプログラムはいずれもディレクトリ directory の下に格納されている name_file で指定された複数の TIFF 形式の画像を GIF 形式の画像ファイル GIF_file （"−" を指定すると標準出力）にまとめて格納する。これらを使えば、 stl_bev などで作成した TIFF 形式画像を name_file で指定した順番（シナリオ）で次々と表示するアニメーションを作成できる。

t2g_movie_gray は白黒もしくはグレースケール形式の TIFF 画像だけを処理できる。また、8 bits 以下の画素値のものに限られるが、 t2g_movie_CM はカラーマップ形式の TIFF 画像も処理できる。 t2g_movie ではこれらに加えてフルカラー形式の TIFF 画像を取り扱うことができるが、その際に行われる stl_bev_GIF などで使われているものと同じ手法による表示色の減色処理（GIF 形式では画像ごとに 256 種類以下の表示色しか保持できない）にはやや長い処理時間を要する。

複数の画像のファイル名の指定について

上記のプログラムや後述する gif_movie および si_stl_[A,B,C] の起動パラメータ name_file は指定したディレクトリの下にある処理したい画像ファイルの名前のリストを書き込んだテキストファイルの名前である。このリストにはアニメーション表示の順や３次元画像の z 座標値に相当するスライス番号の順で（２次元）画像のファイル名（ディレクトリ名は不要）を空白、タブもしくは改行コード区切りで書き並べればよい。

指定したディレクトリの下に処理したい画像ファイルだけがあり、かつ、それらの名前の辞書式（英数字）順が処理の順（もしくはその逆順）に一致するなら、 name_file として "−"（もしくは "−−r"）を指定すればよい。また、"−−n"（"−−nr" か "−−rn" のいずれか）を指定してやるとファイル名に含まれる数値（浮動小数点数でもよい）の順（その逆順）に処理を行う。そして、画像ファイル名の先頭に "−" を付加した文字列（画像ファイル名を "image_file" として、"−image_file"）を指定すれば、その１枚の画像だけを処理できる。

gif_movie

起動法

gif_movie directory name_file delay_time repeat_count GIF_file

機能と使用法

ディレクトリ directory の下に格納されている name_file で指定された複数の GIF 形式の画像をまとめたアニメーション表示可能な GIF ファイル GIF_file を作成する。起動パラメータ delay_time はそれぞれの画像表示の間の μsec. 単位の遅延時間で、０～ 65535 の整数値を指定する。また、０～ 65536 の範囲の整数値 repeat_count はアニメーションの繰り返しの回数を表し、通常は０（繰り返しの回数は再生ソフトウェアにまかせる）もしくは 65536（繰り返しを無限に行う）を指定すればよい。なお、これら２つのパラメータのいずれかが０でなければ GIF_file は GIF89a 形式となる。

例えば MS-PowerPoint のように前述の t2g_movie や stl_bev_GIF などで作成した GIF87a 形式のファイルのアニメーションを再生できないソフトウェアがある。それに対処するには以下のようにして GIF87a 形式のファイル GIF87a_file （カレントディレクトリ "." にあるものとする）を GIF89a 形式のもの（GIF89a_file）に変換すればよい。

gif_movie . −GIF87a_file 0 65536 GIF89a_file

g2t_split、g2t_split_RGB および gif_split

起動法

g2t_split GIF_file format
g2t_split_RGB GIF_file format
gif_split GIF_file format

機能と使用法

これらのプログラムはいずれも「スチール画像」の取得やアニメーションの編集などのために GIF 形式画像ファイル GIF_file に含まれている複数の画像を分離し、それらを g2t_split と g2t_split_RGB は TIFF 形式の、また、gif_split は GIF 形式の画像ファイルにそれぞれ格納する。 GIF_file から分離されたそれぞれの画像は起動パラメータ format で指定した名前のファイルに格納される。これは C 言語の標準関数 sprintf() などの format 指定用の文字列で、例えばアニメーションを構成する 100 枚の画像のそれぞれを gif_split を使って GIF 形式の画像に変換し、それらをディレクトリ "DIR"（あらかじめ作成しておく必要がある）の下にファイル名 "00.gif" 〜 "99.gif" で格納したい場合には "DIR/%02d.gif" と指定すればよい。

g2t_split はアニメーション用 GIF 画像のそれぞれをそのままカラーマップ形式の TIFF 画像のファイルに変換する。これに対して、 g2t_split_RGB はそれぞれの画素で色の R、G、B 成分が同じ値になっている場合は 8 bits の画素値を持つグレースケール形式の TIFF 画像に、また、そうでない場合は 8 bits の R、G、B 成分の容量を持つ画素からなる RGB フルカラー形式の TIFF 画像に変換する。

gif_area、gif_area_nbg および gif_trim

起動法

gif_area GIF_file R1 G1 B1 R2 G2 B2
gif_area_nbg GIF_file R1 G1 B1 R2 G2 B2
gif_trim org_GIF x1 y1 x2 y2 R G B new_GIF

機能と使用法

これらはアニメーション用の GIF 画像に含まれている２次元画像のすべてにおいて不要な、表示したい２次元像（前景）の周囲にある余白（背景）の部分を取り除くためのプログラムである。 gif_area と gif_area_nbg はそれぞれ、すべての２次元画像上の前景もしくは背景でない部分をカバーする最小の長方形領域を指定された画素の色情報をもとにして探し出す。そして、 gif_trim はこのようにして決めた長方形領域を切り出した、元のものと同じ枚数・順番の２次元画像を含む GIF 画像を作成する。

gif_area は GIF 形式画像ファイル GIF_file 上の色の R、G、B 成分の値が起動パラメータで指定された R1 〜 R2、G1 〜 G2、B1 〜 B2 の範囲にある画素を前景と見なして、それらの x および y 座標の値域を調べる。これに対して、 gif_area_nbg は指定された範囲の色を持つ画素を背景と見なして、それら以外の画素の座標の値域を調べる。ただし、これらの色の成分値の範囲を表すパラメータはすべて整数で、 $0 \le R1 \le R2 \le 255$、 $0 \le G1 \le G2 \le 255$ かつ $0 \le B1 \le B2 \le 255$ となっている必要がある。最終的に得られた GIF_file 上の２次元画像の前景もしくは背景でない部分すべてをカバーする x および y 座標値の値域をそれぞれ x1 〜 x2 と y1 〜 y2 とすると、 gif_area と gif_area_nbg はいずれも x1、y1、x2、y2 の値をこの順でタブコードで区切った１行にまとめて標準出力に書き出す。

gif_trim はファイル org_GIF 上のすべての２次元画像から x と y 座標値がそれぞれ x1 〜 x2 と y1 〜 y2 の範囲の長方形領域を切り出す。これらの座標値の範囲を示すパラメータはいずれも整数値で、 gif_area や gif_area_nbg で調べた値そのものを指定すればよい。 UNIX 環境なら以下のように入力すれば gif_area や gif_area_ngb で調べた値をそのまま gif_trim に引き渡すことができる（ただし、以下に記した "‘" は Windows 用のキーボードなら "@" と同じ

キーによって入力できる reverse quote である）。

gif_trim org_GIF ‘gif_area GIF_file R1 G1 B1 R2 G2 B2‘ R G B new_GIF
gif_trim org_GIF ‘gif_area_nbg GIF_file R1 G1 B1 R2 G2 B2‘ R G B new_GIF

また、これを行わない場合は、gif_trim に $0 \le x1 \le x2 \le Fx$ かつ $0 \le y1 \le y2 \le Fy$ を満たす座標値の範囲を指定しなければならない。ここで、Fx と Fy はそれぞれ org_GIF の２次元画像の x もしくは y 方向の画素数の最大値である（GIF 画像の２次元画像はすべて同じ画素数とは限らない）。 org_GIF の様々な画素数の２次元画像のいずれかが指定された長方形領域よりも小さい画像なら、それを切り出した画像上には値の存在しない画素が含まれる。パラメータ R、G、B（いずれも 0 〜 255 の整数値）はそのような画素に入れる色の３成分の値である。このようにして切り出した元のものと同じ枚数の２次元画像は元のものと同じ順番で GIF87a 形式のファイル new_GIF に書き込まれる。

３次元画像中の物体像の STL データ抽出とその再画像化のためのプログラム群

si_stl_A および si_stl_B

起動法

si_stl_A directory name_file PV1 PV2
si_stl_B directory name_file PV1 PV2 {R G B} STL_file

機能と使用法

これらのプログラムはいずれもディレクトリ directory の下にある name_file で指定された TIFF 形式画像ファイル上のスライス画像（slice images）から構成される３次元画像を読み込む。ただし、これらの画像ファイルは画素値の容量（bits per sample；BPS）が 16 bits 以下の白黒、グレースケールもしくはカラーマップ形式のものでなければならない（後の処理ではこれらの表示色の情報は無視され、画素値のデータだけが使われる）。読み込んだ３次元画像上で値域が PV1 〜 PV2 （PV1 と PV2 はともに整数値で $0 \le PV1 \le PV2 \le 65535$）の値を持つ画素を物体像と見なして、像の表面をできるだけ少ない個数の三角形で分割した STL データを作成する。

こうして抽出した３次元物体像の形状の STL データを si_stl_A は ASCII STL 形式で標準出力に書き出す。これに対して、si_stl_B は binary STL 形式でファイル STL_file にデータを格納する。ただし、起動パラメータ R、G、B として色の３成分の強度（それぞれ 0 〜 255 の整数値）を指定すれば、 si_stl_B はその色情報をすべての三角形に持たせた color STL 形式のデータを STL_file に格納する。このような一連の処理の最中に si_stl_[A,B] は処理結果の概要を示す以下の 12 個の整数値をタブコードで区切った３行にまとめて標準エラー出力に書き出す。

３次元画像の x、y、z 方向の画素数
３次元画像上で物体像と見なした画素の総数
物体像の画素の座標値の最小値（x、y、z 成分それぞれの値）
物体像の画素の座標値の最大値（x、y、z 成分それぞれの値）
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

なお、si_stl_[A,B] は物体像の画素に印をつけた３次元２値画像をメモリに保持するので、実行に際してはその総画素数の８分の１に相当する bytes のメモリが最小限必要である。

si_stl_C

起動法

si_stl_C directory name_file STL_file

標準入力から読み込むパラメータ指定行

PV

PV1 PV2

PV R G B

PV1 PV2 R G B

PV1 PV2 R1 G1 B1 R2 G2 B2

機能と使用法

このプログラムは概ね前述した si_stl_B （および si_stl_A）のカラー版の機能を持つ。すなわち、si_stl_C は指定したディレクトリ directory の下の name_file で指定した TIFF 形式スライス画像ファイルを読み込み、標準入力から指定した画素値の値域と色情報などの対応関係に基づいて３次元画像上の物体像を識別してその表面を色情報ごとの三角形に分割し、その結果のデータを color STL 形式でファイル STL_file に書き込む。２値化した画素値だけを保持する si_stl_[A,B] とは異なり、 si_stl_C は読み込んだ３次元画像の画素値のデータをそのままメモリ上に保持する。この制約条件のもとでできるだけ画素数が多い３次元画像を処理できるように、 si_stl_C が読み込み可能な TIFF 形式スライス画像の画素値の容量（BPS）を 8 bits 以下にしてある（si_stl_B では 16 bits 以下）。ただし、この BPS の制限は si_stl_C のコンパイル用スクリプト Makefile の書き換え（ソースファイルの書き換えは不要）により容易に変更可能である（以下の説明ではこの BPS の制限に従った場合の画素値の上限値の 255 を文中に埋め込んであります）。

標準入力からの指定には上記の５種類の記述法があり、種類の違うものを混ぜて複数回づつ指定してもよい。これらに含まれるパラメータのうち PV、PV1 および PV2 はいずれも画像上の画素値を示す 0 〜 255 の整数値で PV1 ≤ PV2 の制限がある。また、R、G、B、R1、G1、B1、R2、G2 および B2 は STL データ中の三角形の色情報を決める際に使う色の R、G、B 成分の強度である。これらはすべて 0 〜 255 の整数値で、画素値の場合とは異なり例えば R1 > R2 なども許される。しかし、color STL 形式の制限のため、三角形の色情報としては各成分とも値の 8 bits 中の上位 5 bits だけが有意である。 STL データの鳥瞰図などの表示色の減少と言う意味ではこれには実際上の問題はない（目視では認識不能）と思われるが、si_stl_C がこれらの 3 × 5 = 15 bits の色情報に基づいて物体像表面を別個の三角形に分割することには若干の注意が必要である。例えば成分値が１だけ異なる２色を指定したときそれらの色を割り当てられた画素が表す物体像の表面は大抵の場合は同じ三角形に分割されるが、成分値の大きい方が８で割り切れるならそれらは別の三角形に分割されて出力される STL データのサイズを大幅に増加させることになる。

標準入力からの１番目と２番目の記述法は３次元画像上で物体像ではない画素の値域を指定する場合に用いる（これらが物体像ではない画素の値の指定であることに注意せよ）。すなわち、１番目の場合は PV、２番目では PV1 〜 PV2 の値を持つ画素を後の処理（３次元画像上の物体像表面の三角形分割）において物体像ではないと見なす。

これら以外の標準入力からの記述法は画素値ごとの色情報を指定する場合に用いる。３番目と４番目の記述法はそれぞれ PV もしくは PV1 〜 PV2 の値を持つ画素に３成分の強度値が R、G、B の色情報を指定する。また、５番目のものは画素値 PV = PV1 〜 PV2 に対して以下の式で得られる強度値 R、G、B（正確にはこれらを整数化した値）を割り当てる。

$$\left.\begin{aligned} R &= R1 + (R2 - R1) \times (PV - PV1) / (PV2 - PV1) \\ G &= G1 + (G2 - G1) \times (PV - PV1) / (PV2 - PV1) \\ B &= B1 + (B2 - B1) \times (PV - PV1) / (PV2 - PV1) \end{aligned}\right\} \text{ただし、} 0 / 0 \equiv 0 \text{ とする}$$

このような標準入力からの記述すべてが終了した後に指定した色情報に基づいた物体像表面の三角形分割が始まる。ただし、重複した画素値の値域の指定があった場合は後で指定したものが優先される。また、すべての指定の前に上記の記述法で

```
0                       ：値０の画素を物体像ではないとする
1  255  255  255 255    ：値０以外の画素は白色の物体像を表すとする
```

があらかじめ指定されているので、標準入力からの記述なし（空入力）でも si_stl_C はとりあえずの STL データ（画素値０以外を物体像とした２値画像に対するもの）を作成する。なお、si_stl_C が出力したこのような２値画像の STL データ上の分割三角形は si_stl_[A,B] のものと同じではない（物体像表面を完全に覆うと言う意味では同じであるが）。この違いは si_stl_C が採用している多値画像上の物体像表面の三角形分割に関するウルトラ Q 難易度のアルゴリズム（言われると気づく非常に簡単な仕組みで複雑な処理を単純にするアルゴリズム）に起因したもので、実用上は問題ないはずである。

si_stl_C は一連の処理中にその概要を示す 12 個の整数値を標準エラー出力に書き出す。その内容は si_stl_[A,B] のものとまったく同じなので、ここでは説明を省略する。

stl_si

起動法

```
stl_si  STL_files  xO  yO  zO  Sx  Sy  Sz  Nx  Ny  Nz  directory
```

機能と使用法

指定したファイル STL_files（複数も可； "−" を指定すると標準入力からデータを読み込む）の中の STL データが表す物体像の内部を塗りつぶした３次元２値画像（物体像内部の点の画素値が１）を作成し、ディレクトリ directory（あらかじめ作成しておく必要がある）の下にスライス番号（３次元画像の z 座標値）を名前の一部とする TIFF 形式スライス画像ファイルとして格納する。パラメータ xO、yO、zO は画像化する領域の座標値（STL データの三角形の頂点の座標値に対応する値）の３成分それぞれの最小値、 Sx、Sy、Sz はそれぞれ画素の x、y、z 方向の幅（辺長）、そして Nx、Ny、Nz は作成する画像の x、y、z 方向の画素数である。すなわち、STL データが表す物体像のうちで座標値の値域が

$$x = xO \sim xO + Sx \times Nx$$
$$y = yO \sim yO + Sy \times Ny$$
$$z = zO \sim zO + Sz \times Nz$$

の直方体内部だけが画像化される。物体像の全体を画像化する場合には先に説明したプログラム stl_nmm で得られる座標値の値域などを利用してこれらのパラメータの値を決定すればよい。また、si_stl_[A,B,C] などで３次元画像から作成した STL データを再度画像化する場合で xO = yO = zO = 0、Sx = Sy = Sz = 1、かつ、Nx、Ny、Nz として元の画像の値と同じ画素数を指定すれば、元の３次元画像で物体像と見なした画素の値を１（それ以外を０）としたものと同じ３次元２値画像が得られる。なお、このプログラムが作成したそれぞれの TIFF 形式スライス画像ファイルでは画素値が LZW 圧縮されており、画面表示したときに画素値０が白で画素値１が黒となるような表示属性が付けられている。

読み込んだ STL データのトポロジカルな整合性に問題がある場合、その物体像の内部を stl_si で完全に塗りつぶすことは原理的に不可能である。 stl_si は x 軸に平行な画素列ごとにそれと交差する STL データの三角形の向き（法線ベクトルの x 成分の符号）を調べて物体像の内部と外部を判断しているが、このような三角形との交点のシーケンスが「異常」な画素列に対しては何もしない（その画素列全体を塗り残す）。このエラーが発生した場合、stl_si は塗り残した画素列ごとにその座標値などのデータをタブコード区切りで並べた以下の３行を標準出力（標準エラー出力ではない）に書き出す。

その画素列の z 座標値と y 座標値（いずれも整数値）
法線ベクトルの x 成分が負の三角形と画素列の交点の x 座標値（浮動小数点数）
法線ベクトルの x 成分が正の三角形と画素列の交点の x 座標値（浮動小数点数）

ただし、後２行の複数個の x 座標値はいずれも昇順に並んでいる。これらのデータを使って問題があった画素列をもっともらしく塗るコードを今後開発する。

STL データが表す物体像の楕円体近似に関連したプログラム群

stl_of および stl_of_oblate

起動法

stl_of STL_file {xO yO zO}
stl_of_oblate STL_file {xO yO zO}

機能と使用法

これらのプログラムは指定したファイル STL_file （"−" を指定すると標準入力からデータを読む）の中の STL データが表す物体像全体（複数の像があってもそれらを単一のものとして取り扱う）の形状を近似した３軸不等楕円体のパラメータを計算する。ただし、起動時に xO、yO、zO として３次元空間上の任意の点の座標値の３成分の値が指定された場合はその点を、また、それらの指定が省略された場合には物体像の体積の重心を近似楕円体の中心とする。なお、この楕円体の３軸は相互に直交しており、 stl_of ではそれらのうちの長軸を最初に、また、stl_of_oblate では短軸、中軸、長軸の順でそれらを示すパラメータを決める。

stl_of と stl_of_oblate では Ikeda *et al.* （Mineral. Mag.、64、945 − 959、2000）に記されているものとほぼ同じ手法で STL データが表す物体像の楕円体近似を行っている。ただし、Ikeda *et al.* の手法には明らかな誤りがあったのでそれを正したものを用い、また、物体像に関する体積積分（モーメント）の値は前述の Gauss の発散定理を利用して求めた。ここでは楕円体近似に関するこれ以上の説明は省略する。

stl_of は 10 個の近似楕円体のパラメータの数値をタブコード区切りで以下の４行にまとめて標準出力に書き出す（stl_of_oblate では以下の長軸と短軸を入れ替えたものを出力する）。

物体像の体積
近似楕円体の中心の座標値（x、y、z 成分の値）
近似楕円体の軸方向を示す角度（長軸方向の経度と緯度、および、中軸方向を示す角度）
近似楕円体の軸半径（短軸、中軸、長軸半径）

上記の中軸方向を示す角度は stl_ar などで物体像を軸回転させる際に指定する回転角 θ の符号を逆にしたものに相当する（その詳細は前述の Ikeda *et al.* の論文を参照のこと）。また、これを含む角度の単位はすべて度である。

of_stl_ih、of_stl_oh および of_stl_th

起動法

of_stl_ih {level} {xO yO zO λ ϕ θ A B C} {scR scG scB} STL_file
of_stl_oh {level} {xO yO zO λ ϕ θ A B C} {scR scG scB} STL_file
of_stl_th {level} {xO yO zO λ ϕ θ A B C} {scR scG scB} STL_file

標準入力から読み込むパラメータ指定行

xO yO zO λ ϕ θ A B C
level xO yO zO λ ϕ θ A B C
xO yO zO λ ϕ θ A B C scR scG scB
level xO yO zO λ ϕ θ A B C scR scG scB

機能と使用法

これらのプログラムはいずれも指定された３軸不等楕円体の形状を近似する多面体を表す STL データを作成し、それを binary もしくは color STL 形式でファイル STL_file （"−" を指定した場合は標準出力）に書き込む。それぞれのプログラムの処理内容の違いや多面体を覆う三角形の総数を決めるパラメータ level などについては後で説明する。ただし、level は 0 以上の整数値で、指定を省略すると 0 と見なされる。パラメータ xO、yO、zO は楕円体の中心の座標値である。 λ、ϕ、θ はいずれも度単位の角度で、 λ と ϕ は楕円体の c 軸の方向を示す経度と緯度、 θ は b（もしくは a）軸の方向を示す角度である。 A、B、C は相互に直交する楕円体の３軸（a、b、c 軸）方向の半径の値である。これらの楕円体のパラメータの値として前述のプログラム stl_of や stl_of_oblate が出力した数値そのものを指定できる。パラメータ scR、scG、scB を指定しなければ楕円体を近似した多面体のデータは binary STL 形式になる。これらに 0 〜 255 の整数値で色の３成分の強度を指定すると、それに相当する色情報を多面体の表面のすべての三角形に埋め込んだ color STL 形式のデータを作成する。

起動時に楕円体の形状パラメータ（xO、yO、zO、λ、ϕ、θ、A、B、C）の指定がない場合には標準入力からそれらの読み込みが行われる。このとき、厳密には STL 形式の仕様に反することだが、プログラム of_stl_[i,o,t]h はそれぞれの楕円体を近似した複数の多面体の STL データをひとつのファイルにまとめて書き込むことができる。すなわち、上記のように４種類の記述法が許されている標準入力からの指定は異なる種類のものを交えて複数回行ってもよい。その際に上記の２番目以降の記述法を使えば楕円体（多面体）ごとに level の値や色の情報（scR、scG、scB）を個別に指定することもできる。なお、上記の１番目の記述法のように level などの指定を省略すると起動時に指定された値が用いられる。

楕円体の多面体近似について

プログラム of_stl_ih、of_stl_oh および of_stl_th はそれぞれ正 20 面体（icosahedron；ih）、正８面体（octahedron；oh）および正４面体（tetrahedron；th）の表面の正三角形のそれぞれを以下に記す（再帰的な）手続きで細分化した多面体を用いて楕円体の像を近似している：

[0] まず、球に内接する適当な正多面体（前記の３つの正多面体のいずれか）を用意する。

[1] 多面体の三角形それぞれを３辺の中点を新しい頂点として４個の三角形に分割し、それらの新しい頂点を球の中心から投影した球面上の点に移動する。このような三角形の細分化処理を球（球面）の近似に十分な多面体が構築できるまで繰り返す。

[2] 最後に、このような球（球面）を近似する多面体を長、中、短軸の方向にそれぞれ伸縮して近似楕円体に合わせた形状に変形する。

上の処理 [1] の三角形の細分化の繰り返しの回数を of_stl_ih などの起動パラメータ level で指定する。ただし、level に０を指定すると三角形の細分化は行われず、元の正多面体を変形したもので楕円体を表すことになる。元の正多面体にもよるが、level として概ね７以上の値を指定すれば楕円体の表面を表すのに十分な多面体（STL データ）になるハズである。

なお、上記の手順で得られる楕円体を近似する STL データ中の三角形の総数は、処理に用いた元の正多面体の面の数に 4^{level} を乗じた値となる。それゆえ、of_stl_[i,o,t]h で複数個の楕円体のデータを作成する場合は、それぞれに体積（もしくは代表的な軸半径）の対数に比例した level の値（非負の整数値）を指定すればよい。