Date: Thu, 19 May 2011 15:38:20 +0900
From: Tsukasa NAKANO
To: Akira Tsuchiyama, Yoshito Nakashima, Satoshi Okumura, Kentaro UESUGI, Masayuki Uesugi, Michihiko Nakamura, Takashi Matsushima, 道上達広
Subject: Convex-Hull-programs

みなさま、

GSJ/AIST のなかのです。2 および 3 次元画像上の物体像や、CAD 用の STL もしくは PLY/ZCP 形式のファイル上の 3 次元像の凸包（とっぽう；convex hull）を抽出するプログラム群を書きました。ただし、3 次元像の凸包の抽出にはぼくが独自開発したコードではなく、以下の教科書に載っていた分割統治（divide and conquer）法を用いた FORTRAN のサブルーチン "CHULL3" を流用しました。

杉原厚吉
FORTRAN 計算幾何プログラミング
岩波コンピュータサイエンス、岩波書店、1998、402p

CHULL3 の FORTRAN コードが置いてある場所
http://home.mims.meiji.ac.jp/~sugihara/Welcomej.html

ついでに言っておくと、与えられた点の集合に対する凸包構築は「計算幾何学」の重要なテーマのひとつなので、それに関する多数の論文や解説書の類が出回っています。ぼくは以下の教科書にも目を通しました。

M. de Berg, M. van Kreveld, M. Overmars & O. Schwarzkopf（3M & O）
Computational Geometry, Algorithms and Applications
Springer-Verlag, 1997, ???p（未読）

浅野哲夫訳、コンピュータ・ジオメトリ、近代科学社、2000、441p
この和訳は良くないので、原書にあたるべき。

J. O'Rourke
Computational Geometry in C, 2nd ed.
Cambridge Univ. Press., 1998, 376p
逐次追加法（incremental algorithm）のコードの例が載っている。しかし、その実行速度や堅固さ（robustness）は前記の CHULL3 には及ばない。

これら以外にも凸包抽出に関する多数の解説書やプログラムコードを Google で得ることができます。特に、以下の Chan の分割統治法の解説論文は有用です。

http://www.cs.uwaterloo.ca/~tmchan/ch3d/ch3d.pdf

(1) 凸包とは何か

添付した "t_ch_line.gif" をご覧下さい。その上下２個の図の上に赤色で描いた凸多角形がそれぞれの２次元画像上に青色で描いた物体像の凸包です。

> 凸包抽出の際には物体像の「つながり」を考慮しません。画像上に分散している個々の物体像の凸包を求めたい場合、物体像と見なす画素のクラスタラベリングを事前に行っておく必要があります。

また、３次元の場合も同様で、物体像（を構成している画素すべて）がちょうどおさまる凸多面体（polytope と呼ぶらしい）が凸包です。

> 画像上の物体像の凸包抽出では画素が正方形（２次元画像の場合）もしくは立方体（３次元）と仮定しています。実際の画素が長方形や直方体でも、これで得られる凸包は同じになると思われます（これは自明か？）。

ぼくが書いた２次元画像用のプログラムでは物体点の逐次追加法によって凸包を抽出しています。また、３次元画像用の主要部は前記の（分割統治法を使った）CHULL3 ですが、それに渡す物体点（== スライス画像上の物体像に対する２次元凸包の点）を逐次追加法で選別することにより大幅な省メモリと高速化を実現しました。ここでは凸包抽出のアルゴリズムに関するこれ以上詳しい話は省略します。

(2) 凸包を抽出できると何がわかるか

前記の教科書では「３次元凸包抽出プログラムで２次元分布している点の Voronoi 分割ができること」や「CG における粒子衝突の検出への凸包の利用」などが謳われていますが、画像解析の観点から言えばそれらよりも以下の２つの応用が重要だと思います。

まず、画像上の物体像に含まれる空隙（porosity）の識別・分類です。添付した "t_ch.gif" をご覧下さい。そこに先の "t_ch_line.gif" の２個の凸包のそれぞれの内部を塗りつぶし（rasterization）した結果を示しました。ただし、凸包の内部の青色の物体像に囲まれた「孤立した空隙」の画素をクラスタラベリングして識別し、緑色で示しました。そして、それら以外の、そもそもが物体像の外部にあった赤色の領域を「外部につながっている空隙」と見なすことができます。このようにして物体像に付随している２種類の空隙を自動的・機械的に識別することができます。

> この解析法はつちやまさん御愛用の「風呂敷法」を洗練したものです。つまり、Erosion-Dilation（ED）による風呂敷法とは異なり、凸包抽出にはアドホックなパラメータの指定（および試行錯誤）は不要です。物体像を与えればその凸包は一意的に決まります。
>
> とは言え、物体像の凸包近似は「ノイズ」に弱いので、現実的な外形にするためには単純な２値化だけではダメな可能性があります。その場合には ED などによる物体像の加工（ノイズ除去）が必要かもしれません。

それから、これはまだ実現していませんが、物体像の外形を凸包で近似すればその「スペクトル解析」を無理なく行えます。その定義から、凸包内部に置いた「原点」と凸包上の点の間の距離 R は一価関数になります。２次元像の場合、このような R は方位角 θ の一価関数です。 そして、R(θ) は θ に関して周期 360° の関数なので Fourier 級数展開が可能であり、さらに、凸包は折れ線なので個々の Fourier 係数の積分の高精度な計算が可能です。また、３次元像の場合も同様で、その外形を近似した凸包に対して高精度な球面調和関数展開を行うことができます。このような２もしくは３次元像の外形を近似した凸包のスペクトル解析用のコードを今後書くつもりでいます。

(3) 凸包抽出プログラムの概要

前置きが長くなりました。今回紹介する凸包抽出プログラムは以下の７個です。

CAD 用のファイル上の３次元物体像の凸包を抽出するプログラム

zcp_ch：PLY/ZCP 形式のファイルを入出力する。

stl_ch：STL 形式のファイルを入出力する。

画像上の物体像の凸包の polytope を抽出するプログラム

t_ch_line： ２次元画像上の物体像の凸包の多角形を抽出する。

si_ch_zcp

３次元画像上の物体像の凸包を PLY/ZCP 形式ファイルに書き込む。

si_ch_stl

３次元画像上の物体像の凸包を STL 形式ファイルに書き込む。

画像上の物体像の凸包近似によって２種類の空隙を識別するプログラム

t_ch： ２次元画像用

si_ch： ３次元画像用

(4) プログラムのインストール

上記の７個のプログラムのソースコードとそれらの Windows 用実行ファイルは以下の２×２個の書庫ファイルに入れてあります（以下に併記した ZIP 形式の書庫ファイル "*.zip" と gzip'ed TAR ファイル "*.taz" の内容は同じです）。

zcp_ch と stl_ch 用の書庫ファイル

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl_ch.zip

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl_ch.taz

t_ch* と si_ch* 用の書庫ファイル

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch.zip

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch.taz

Windows の場合は上記の ZIP 形式書庫ファイルの中に入っている 7 個の実行ファイル "stl_ch/exe/*.exe" と "ch/*.exe" を実行パスに登録済みの適当なディレクトリにコピーすればインストールは完了です。ただし、これらの実行ファイルは MinGW（Minimalist GNU for Windows）を用いてコンパイルしたもので、Cygwin でも実行可能です。

Linux や MacOSX などの場合はソースファイルのコンパイルを行う必要があります。それには書庫ファイルの中の "stl_ch/src/Makefile" と "ch/Makefile" をお使い下さい。ただし、サブルーチン CHULL3 のコンパイルには FORTRAN77 のコンパイラが必要で、その処理コマンドとして "f77" を Makefile に埋め込んであります。

最近の Linux ディストリビューションでは f77（もしくは g77）ではなく FORTRAN95 対応の “gfortran”（GNU FORTRAN）がバンドルされているかもしれません。その場合は Makefile の先頭付近にある設定 "FC = f77" を適当に書き換えて下さい。

いずれにせよ、GNU のもののような FORTRAN と C 言語のコードを統合可能な処理系が必要です。端末から以下のように入力すれば、Linux などでプログラム群をコンパイル・インストールできるはずです。

zcp_ch と stl_ch のインストール

```
% tar   xzf   stl_ch.taz
% cd   stl_ch/src
% make
% make   install
```

これによりディレクトリ "stl_ch/bin/" の下に zcp_ch と stl_ch がコピーされるはずなので、そのディレクトリを実行パスに登録するか、それらの実行ファイルを実行パスに登録済みのディレクトリにコピーする。

t_ch* と si_ch* のインストール

```
% tar   xzf   ch.taz
% cd   ch
% make
```

これによりディレクトリ "ch/" の下に 5 個の実行ファイル（t_ch_line、t_ch、si_ch_zcp、si_ch_stl および si_ch）が作成されるはずなので、それらを実行パスに登録済みの適当なディレクトリにコピーする。

なお、２×２個の書庫ファイルのすべてに入れてあるテキストファイル "readme3d.txt" は作者の杉原厚吉さんが記したサブルーチン CHULL3（ソースファイル "chull3.f"）などに関する文書です。できることなら一読して下さい。

(5) プログラムの起動法

ここで紹介するプログラムはいずれも端末（Windows ならコマンドプロンプト）を開き、そのコマンドラインから以下のようにパラメータを指定して起動します。

zcp_ch　org.zcp　new.zcp

機能

PLY/ZCP 形式ファイルの上の３次元物体像の凸包を抽出する。

起動パラメータ

org.zcp

３次元物体像を記述した PLY/ZCP 形式ファイルの名前。ただし、”-“ を指定するとその内容のバイナリデータを標準入力から読み込む。

new.zcp

その凸包のデータを書き込む PLY/ZCP 形式ファイルの名前。ただし、”-“ を指定するとそのバイナリデータを標準出力に書き出す。

備考

後述するプログラム vf2zcp を使えばテキストファイルから org.zcp を作成できる。また、zcp_dmp によって new.zcp をテキストファイルに変換することもできる。

stl_ch　org.stl　new.stl

機能

STL 形式ファイルの上の３次元物体像の凸包を抽出する。

起動パラメータ

org.stl

３次元物体像を記述した STL 形式ファイルの名前。ただし、”-“ を指定するとその内容のバイナリデータを標準入力から読み込む。

new.stl

その凸包のデータを書き込む STL 形式ファイルの名前。ただし、”-“ を指定するとそのバイナリデータを標準出力に書き出す。

備考

STL 形式のファイルは後述する STL 用プログラム群で処理できる。

t_ch_line　org.tif　rangeList　> new.txt

機能

２次元画像上の物体像の凸包を抽出する。

起動パラメータ

org.tif : TIFF 形式の画像ファイルの名前

rangeList

物体像と見なす画像上の画素の画素値の値域。例えば、"1-"（画素値１以上） や "0,255"（画素値０と 255）のような記法で指定する。

new.txt

標準出力に書き出される凸包の多角形のテキストデータをリダイレクトするファイルの名前。以下の構成の ASCII 文字のデータが書き込まれる。

1行目：多角形の頂点の個数

2行目以降（タブコード区切りで2個の値が並んでいる）

それぞれの頂点の x および y 座標値

si_ch_zcp　directory　nameFile　rangeList　new.zcp

機能

3次元画像上の物体像の凸包の多面体を PLY/ZCP 形式ファイルに書く。

起動パラメータ

directory

3次元画像を構成するスライスの TIFF 形式画像ファイルが格納されているディレクトリの名前

nameFile

スライス画像のファイル名のリストを書き込んだファイルの名前。指定したディレクトリにスライス画像のファイルだけが格納されており、かつ、それらのファイル名の英数字の順番がスライスの順番と一致する場合は "-" を指定すればよい。

rangeList: t_ch_line のものと同じ。

new.zcp

3次元画像上の物体像の凸包の多面体のバイナリデータを書き込む PLY/ZCP 形式ファイルの名前。”-“ を指定するとそのデータを標準出力に書き出す。zcp_ch のものとは異なり、多面体を構成する三角形にはその法線ベクトルの x、y もしくは z 成分値が0か否かに応じた7種類の色を付随させてある。

備考

zcp_ch のものと同じ。

si_ch_stl　directory　nameFile　rangeList　new.stl

機能

3次元画像上の物体像の凸包の多面体を STL 形式ファイルに書く。

起動パラメータ

directory：si_ch_zcp のものと同じ

nameFile：si_ch_zcp のものと同じ

rangeList: t_ch_line のものと同じ。

new.stl

3次元画像上の物体像の凸包の多面体のバイナリデータを書き込む STL 形式ファイルの名前。”-“ を指定するとそのデータを標準出力に書き出す。stl_ch のものとは異なり、多面体を構成する三角形にはその法線ベクトルの x、y もしくは z 成分値が0か否かに応じた7種類の色を付随させてある。

備考

stl_ch のものと同じ。

t_ch　org.tif　rangeList　{new.tif}　> new.txt

機能

２次元画像上の物体像の凸包を抽出し、それを用いて識別した物体像、その内部の孤立した空隙、それら以外の凸包の内部領域（物体像の外部につながった空隙）および凸包の外部領域のそれぞれを色分けした新しい２次元画像を作成する。

起動パラメータ

org.tif : t_ch_line のものと同じ

rangeList : t_ch_line のものと同じ

new.tif（指定の省略が可能；省略すると新しい画像を作らない）

以下の画素値０～３が書き込まれた TIFF 形式画像ファイルの名前。ただし、new.tif には画素値に応じた色情報（カラーマップ）が埋め込まれている。

３：物体像を構成する画素。青色で表示される。

２：物体像の内部の孤立した空隙の画素。緑。

１：上記以外の凸包の内部にある画素。赤。

０：凸包の外部にある画素。白。

new.txt

以下の３行のテキストデータをリダイレクトするファイルの名前。

１行目

new.tif の上の画素値１の画素（物体像外部の空隙の画素）の個数

２行目（タブコード区切りで２個の値が並んでいる）

画素値２の画素（孤立した空隙の画素）の個数と、それらの空隙のクラスタの個数

３行目： 画素値３の画素（物体像の画素）の個数

si_ch　orgDir　nameFile　rangeList　{newDir}　> new.txt

機能

３次元画像上の物体像の凸包を抽出し、それを用いて識別した物体像、その内部の孤立した空隙、それら以外の凸包の内部領域（物体像の外部につながった空隙）および凸包の外部領域のそれぞれを色分けした新しい３次元画像を作成する。

起動パラメータ

orgDir : si_ch_zcp のものと同じ

nameFile : si_ch_zcp のものと同じ

rangeList : t_ch_line のものと同じ

newDir（指定の省略が可能；省略すると新しい画像を作らない）

以下の画素値０～３が書き込まれた TIFF 形式画像ファイルを格納するディレクトリの名前。ただし、newDir は事前に作成しておく必要がある。また、その下に

格納される新しい画像ファイルの名前は orgDir の下のものと同じになる。そして、それらのファイルには画素値に応じた色情報（カラーマップ）が埋め込まれる。

３：物体像を構成する画素。青色で表示される。

２：物体像の内部の孤立した空隙の画素。緑。

１：上記以外の凸包の内部にある画素。赤。

０：凸包の外部にある画素。白。

new.txt

以下の３行のテキストデータをリダイレクトするファイルの名前。

１行目

newDir に格納する３次元画像上の画素値１の画素（物体像の外部の空隙の画素）の個数

２行目（タブコード区切りで２個の値が並んでいる）

３次元画像上の画素値２の画素（孤立した空隙の画素）の個数と、それらの空隙クラスタの個数

３行目：３次元画像上の画素値３の画素（物体像の画素）の個数

(6) PLY/ZCP および STL 形式ファイルについて

以前に発表したプログラム群を使えば(PLY/ZCP と STL 形式ファイルの相互の変換を含む)STL 形式ファイルに対する様々な処理を行うことができます。

~~http://www.gsj.jp/GDB/openfile/files/no0448/0448index.html~~

https://www.gsj.jp/researches/openfile/openfile2006/openfile0448.html

or

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl.pdf

また、後者のマニュアルには記していませんが、TAR 形式書庫ファイル

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl.tar

に入っているプログラム zcp_dmp と vf2zcp を使えば PLY/ZCP 形式のファイルとテキストファイルの相互変換が可能です。これらの起動法は以下の通りです。

zcp_dmp　org.zcp　> new.txt

機能

PLY/ZCP 形式ファイルのデータをテキストデータに変換する。

起動パラメータ

org.zcp：PLY/ZCP 形式ファイルの名前

new.txt

標準出力に書き出されたテキストデータをリダイレクトするファイルの名前。以下の構成の行のそれぞれにタブコード区切りで複数の値が並んでいる。

１行目（２個の値が並んでいる）

org.zcp の上の物体像を構成する頂点と三角形の個数

その後の頂点の個数分の行（３個の値が並んでいる）

頂点の x、y および z 座標値（一般には浮動小数点数）

その後の三角形の個数分の行（６個の値が並んでいる）

三角形の３個の頂点の番号（０から始まる上記のファイル上での頂点の並びの順番に応じた番号）とその三角形の色の R、G および B 成分の値（０～255 の整数値）

vf2zcp　new.zcp　< org.txt

機能

テキストデータを変換して PLY/ZCP 形式のファイルに書き込む。

起動パラメータ

org.txt

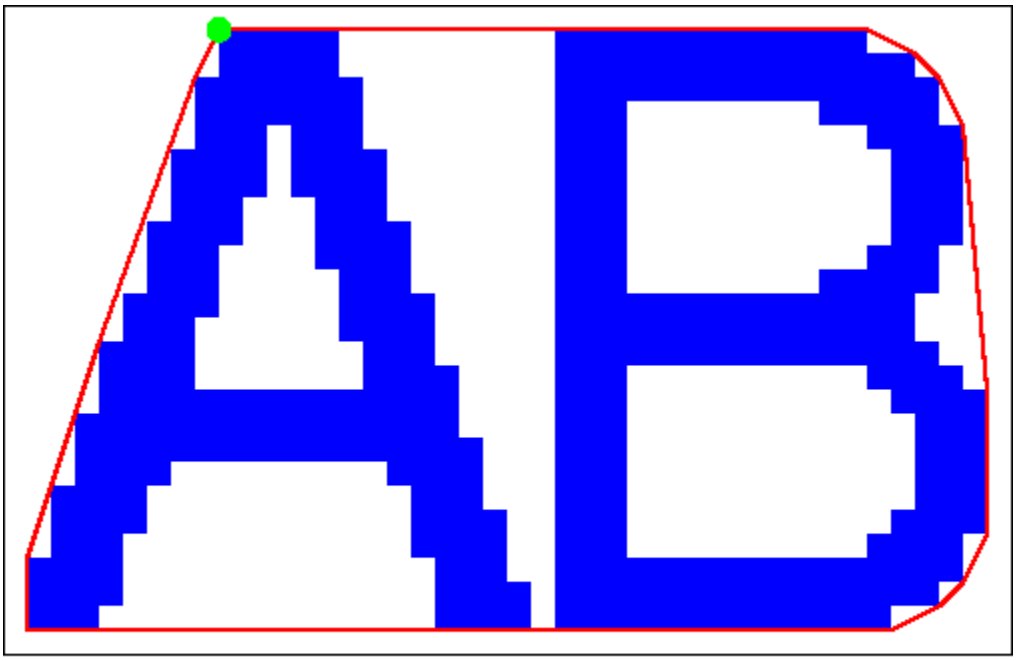
zcp_dmp が出力したもの（new.txt に書き込んだもの）と同じ形式のテキストデータが記されているファイルの名前。なお、vf2zcp はこのデータを本来は標準入力から読み込むが、ここではそれをファイルからリダイレクトした。

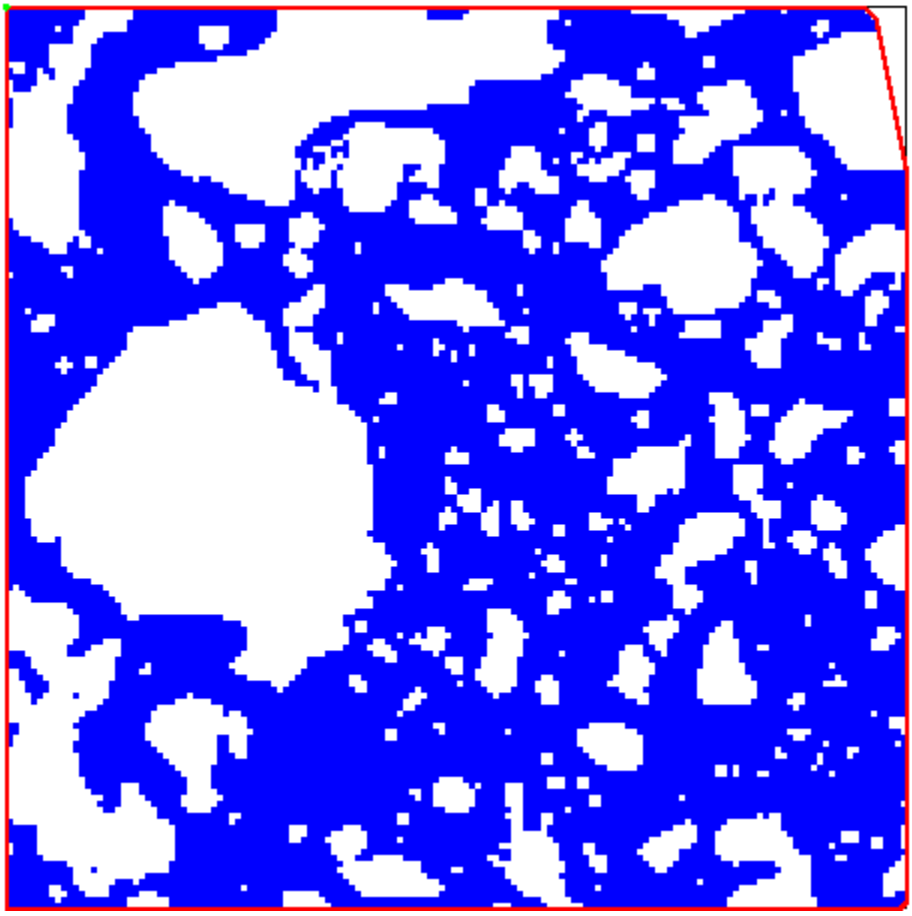
new.zcp

org.txt の内容を変換したデータを書き込む PLY/ZCP 形式ファイルの名前

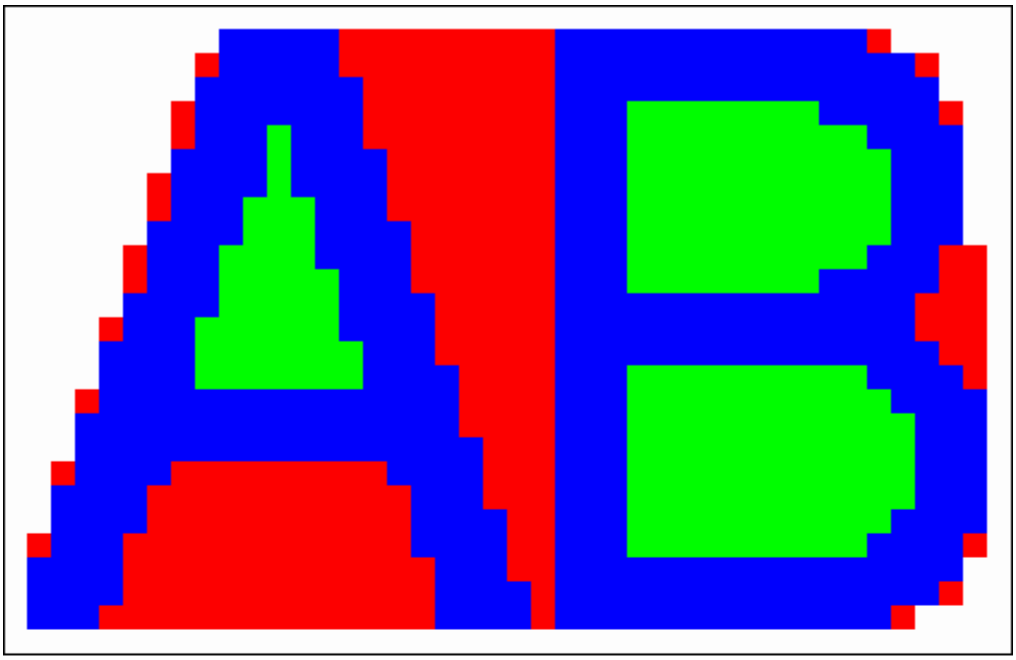
長い E-mail になりました。とりあえず以上です。

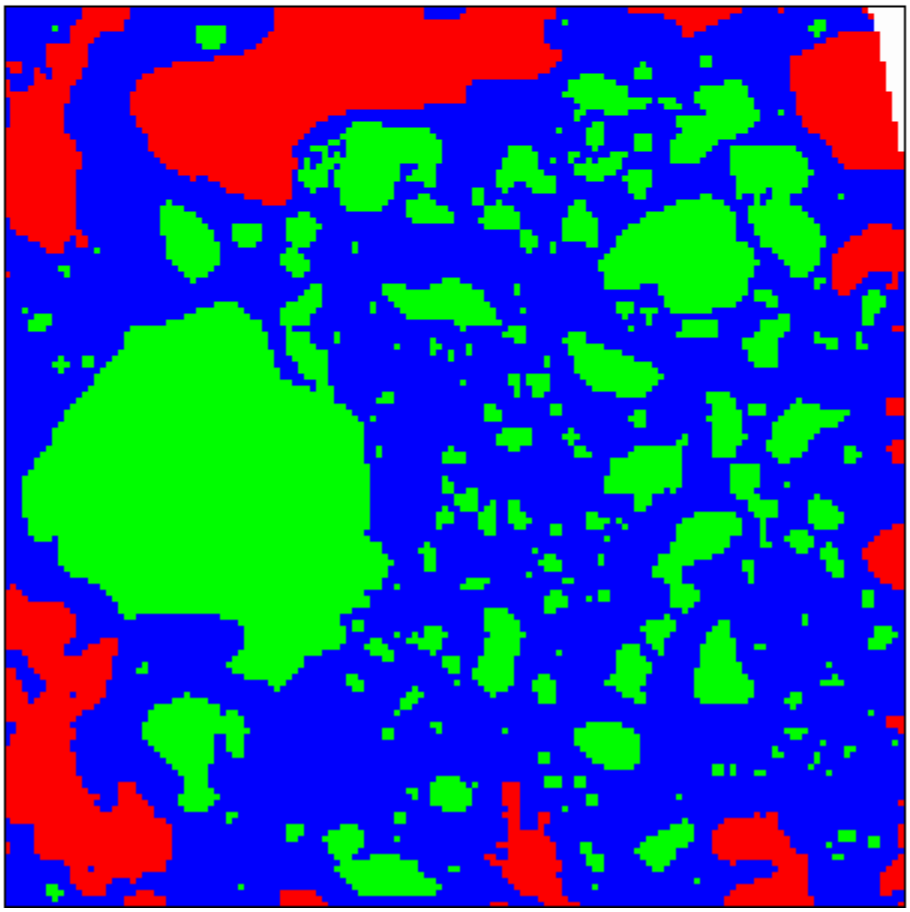
添付ファイル “t_ch_line.gif”





添付ファイル “t_ch.gif”





Date: Fri, 27 May 2011 18:29:31 +0900
From: Tsukasa NAKANO
To: Akira Tsuchiyama, Yoshito NAKASHIMA, Satoshi Okumura, Kentaro Uesugi, Masayuki Uesugi, Michihiko Nakamura, Takashi Matsushima, 道上達広
Subject: Convex-Hull-examples

みなさま、

GSJ/AIST のなかのです。先日の E-mail でお知らせした３次元物体像の凸包の抽出プログラムを用いた３種類の処理の例を紹介します。

(1) CAD 用のファイル上の３次元物体像の凸包抽出

添付した E-mail で紹介しているイトカワなどの小惑星の形状データからそれらの凸包を抽出してみました。例えば、gzip 圧縮した STL 形式ファイル

http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/itokawa_aizu5.04.stl.gz

に入っている会津大学が作成した ver. 5.04 のイトカワの形状データの場合、そのファイルの圧縮を解いた後に以下のように入力すれば、凸包のデータが入った STL 形式ファイル "ch.stl" を得ることができます。

stl_ch itokawa_aizu5.04.stl ch.stl

このようにしてイトカワの７種類の形状モデルとそれ以外の小惑星の 22 個のモデル（添付したテキストファイル "asteroids_obj.txt" に記したもの）の凸包を抽出し、それぞれの鳥瞰アニメーションを作りました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/itokawa_bev/
http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/asteroids_bev/
形状モデルから描いた元の像の鳥瞰アニメーションが入っている。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/itokawa_o_bev/
http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/asteroids_o_bev/
元の像にその形状を近似した楕円体を重ねて描いたもの（オマケ）。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/itokawa_ch_bev/
http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/asteroids_ch_bev/
元の像にその凸包を重ねて描いた鳥瞰アニメーションが入っている。

なお、鳥瞰図の「重ね合わせ」には昔作ったプログラム si_cim（color image multiplier）を使いましたが、これは単純な方法なので出来は良くありません。

(2) 画像上の単一の３次元物体像の凸包抽出

SPring-8 の測定 060410g で FZP-CT を使って撮影した X 線 CT 画像の上の Stardust サンプル（81P/Wild-2 彗星の粒子、C2054.0.35.4 #10）の物体像の凸包を抽出してみました。ただし、FZP-CT の測定条件は以下の通りです。

X 線エネルギー：8 keV
投影数：3600
画素の辺長：42.5 nano meter
３次元 CT 画像の画素数：1060×1060×700

この粒子を切断して分析するため、2007 年の夏頃に CT 画像から物体像を抽出して石膏模型を作りました。その時に使ったファイルが以下の場所にあります。

http://www-bl20.spring8.or.jp/~sp8ct/hd2/stardust/060410_grains_FZP/060410g/

byte_60.gif : X 線 CT 画像のスライス画像の browse 画像
bin_49G/ や bin_49G.gif など
　単純な２値化で得た物体像。サンプルを糊付けした棒の像が付いている。
cut_60E8/、rainbow_60E8/ と cut_60E8.gif
　ED などにより不要な部分を除去した物体像。cut_60E8/ は色なし、それに派手な色付け（詳細は不明）をして石膏模型を作った。cut_60E8/ の画素値１～255 が物体像で、そのサイズは 483×579×500 画素（およそ 20^3 μm^3）。

上記のディレクトリ cut_60E8/ の３次元像の凸包を抽出してみました。ただし、そこには連結していない物体像が含まれていたので、以下のようにしてクラスタラベリングした後に（それで得られたクラスタ画像上で画素値１となっている）最大の物体像のクラスタの凸包を抽出しました。

```
mkdir　ch
si_mcl　cut_60E8　-　1-255　ch
→　91　　← 物体像のクラスタの総数＋１
　178809669　　0　　0　　0　　550　　756　　520
　　← 物体像以外の画素の個数とその bounding box
　38501275　　59　　27　　13　　541　　605　　512
　　← 物体像の最大クラスタの画素数とその bounding box
　634　　108　　190　　360　　130　　199　　373
　　← ２番目のクラスタの情報。最大のものと比べて十分小さい。
　...
si_ch　ch　-　1　ch
→　13624715　　← 凸包内部かつ物体像外部にある画素の個数
　48601　522　　← 物体像内部にある空隙の画素数とそのクラスタ数
　38501275　　← 物体像の画素数
```

この結果の凸包を示す鳥瞰アニメーションを作りました。ご覧下さい。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/060410g.mpg

(3) 画像上の複数の３次元物体像全体の凸包抽出

最後の例は「ストロー」に入れた直径が 0.3～0.5 mm のガラス球の３次元像の凸包です。SPring-8 の測定 040711j で得た X 線 CT 画像から抽出しました。

X 線エネルギー：25 keV
投影数：360
画素の辺長：5.83 μm
３次元 CT 画像の画素数：1000×1000×720

ただし、これは HASPET（Hayabusa Asteroidal Sample Preliminary Examination Team）で用いた画像再構成ソフトウェアの実行例にも使った測定データで、その CT 画像の「作り方」は以下の文書に書いてあります。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/haspet.pdf

さて、測定 040711j の８ビット画素値の CT 画像上の画素値と LAC（X 線線吸収係数）は LAC の理論値０が画素値０、石英（組成 SiO2、密度 2.65 g/cc と仮定）のそれが画素値 125 になる線形な関係にしました。その browse 画像と画素値ヒストグラムは以下の通りです。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/040711j.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/040711j.pdf

このように物体像に相当するヒストグラムのピークの画素値は 180 程度の値なので、この CT 画像を画素値 125 で２値化して物体像を識別することにしました。

まず、以下のようにしてディレクトリ byte/ に入っている 040711j の８ビット画素値の画像を単純に２値化して得た物体像の凸包を抽出しました。

```
mkdir  ch_0
si_ch  byte  -  125-  ch_0
→  13624715
     48601   522
     38501275
```

その結果を示す鳥瞰アニメーションを作りました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/040711j_0.mpg

これからわかるように、単純な２値化では物体像の上部と下部にある「ノイズ」の影響で凸包が変な形になっています。そこで、これらを ED で除去することしました。その理由の説明は省略

しますが、３画素幅の表面層の ED を行いました。

```
mkdir  ch_1
si_ed  byte  -  125-  3  ch_1
→  ...
si_ch  ch_1  -  1  ch_1
→  110157038
     8826     4531
     110343471
```

この結果を示す以下の鳥瞰アニメーションを眺めると、凸包はガラス球を入れたストローと同様な滑らかな（側面および底面の）形状になりました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/040711j_1.mpg

ところで、表面張力もしくは静電気のため、上部に分布しているガラス球の配置は下に凹のすり鉢状になっています。このため、ガラス球の集合体の全体を凸包で近似するのは不適切です。そこで、すり鉢状の部分に相当する番号 z= 0～269 のスライス画像を除外した３次元画像から凸包を抽出してみました。

```
ls  byte  |  tail  -n  +271  > nameFile.txt
mkdir  ch_2
si_ed  byte  nameFile.txt  125-  3  ch_2
→  ...
si_ch  ch_2  -  1  ch_2
→  60468982
     5868     3041
     87877945
```

以下の鳥瞰アニメーション（ガラス球をニョキニョキ生えさせる表示はやめました）からわかるように、ガラス球の全体を近似した凸包はそれらしい形になりました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ch/040711j_2.mpg

さらに、このガラス球の集合体の空間充填率は以下の値だと推定できます。

(5868 + 87877945) / (60468982 + 5868 + 87877945) ～0.592

これは FCC（0.740）や BCC（0.680）の充填率に比べるとかなり低い値です。

長い E-mail になりました。とりあえず以上です。

添付した E-mail

Date: Fri, 04 Mar 2011 10:30:55 +0900
From: Tsukasa NAKANO
To: Takashi Matsushima
Cc: 道上達広, Akira TSUCHIYAMA, Masayuki Uesugi, Kentaro UESUGI, Yuta Imai, Junya Matsuno, Ryo Noguchi, 松本徹, 永野宗
Subject: Re: レゴリス粒子の３軸比

みなさま、

GSJ/AIST の中野 司です。ぼくも「2:√2:1」の話に加わらせて下さい。

(0) --- 削除しました。

(1) 「2:√2:1」が砂粒や岩石片で成り立っていることの「物理」はさておき、道上さんがつちやまさんに渡したと聞いている「３軸比」のデータに「Asteroid by Spacecraft」が含まれていることに興をそそられました。そのような巨大な物体でも「2:√2:1」が成り立っているなら非常におもしろい。そこで、その真偽を実際の小惑星の形状モデルを用いて調べてみました。

(2) まず、前置きですが、「2:√2:1」の３軸不等楕円体は以下のような感じです：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/asteroids/pebble.gif

(3) 前置きその２です。松島さんが「応用力学論文集 11」に概略を書いた「物体像の３軸不等楕円体近似」の方法論は画像上の物体像だけではなく、多面体近似した物体像に対しても適用することができます。つまり、この近似法の詳細は

http://www-bl20.spring8.or.jp/~sp8ct/tmp/ovoid.pdf

に記した通りですが、そこで行っている体積積分を Gauss の発散定理で面積分に置換すれば、物体像の（表面の）形状データからその近似楕円体を計算できます。具体的には、CAD 用のソフトウェアで広く使われている STL（STereo Lithography）形式データ（すべての面が三角形の多面体で表現した物体像の形状データ）から楕円体近似を行えます。詳細は以下をご覧下さい：

~~http://www.gsj.jp/GDB/openfile/files/no0448/0448index.html~~
https://www.gsj.jp/researches/openfile/openfile2006/openfile0448.html

ただし、そこに置いた２個のファイルの代わりに以下をお使い下さい：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl.pdf
http://www-bl20.spring8.or.jp/~sp8ct/tmp/stl.tar

(4) まず、イトカワの形状を調べました。そのデータは「はやぶさ」のランデブーの頃に集めたものなので詳細は忘れました。以下の場所に置いてあります：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/

ここにある gzip 圧縮したファイル "itokawa.*.gz" が形状データです。ただし、"itokawa.dxf.gz" と "itokawa.obj.gz" は JPL のサイトから取ってきたもので、今は亡き Ostro さん達がレーダ観測で作った同じデータだと思います。これらは STL 形式ではないですが、プログラム言語 "awk" を使えば STL 形式に変換できます。"dxf2stl.awk" と "obj2stl.awk" がそれを実行するスクリプトです。また、"itokawa.*.stl.gz" は「はやぶさ」のデータから作った形状データで、"*.aizu5.04.*" は会津大（出村くん）の作品、それ以外は JPL（Gaskell？）が作ったものです（複数の「版」があり、また、最新版もあるかもしれない）。C-shell scripts "bev_*.csh" を使ってこれらの動画を作ってみました：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/obj.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/dxf.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/aizu5.04.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/f0049152.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/f0196608.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/f0786432.gif
http://www-bl20.spring8.or.jp/~sp8ct/tmp/itokawa/f3145728.gif

これらそれぞれの形状データから計算した、イトカワを近似した 3 軸不等楕円体の軸半径 A、B、C（ただし、A < B < C）を添付したファイル itokawa_abc.txt に書き込みました。なお、これらの半径の単位は不明です（以下同様）。

(5) JPL のサイトにあったレーダー観測で得られた小惑星の形状データも調べました。添付したファイル asteroids_obj.txt にサイトの URL（1 行目）と、データファイルの名前と簡単な説明（2 行目以降）が書き込まれています。これら 22 個のデータファイルはいずれも "*.obj" 形式ですが、先の "obj2stl.awk" を使えば STL 形式に変換できます。そのテスト用に作成した動画のファイル "*.gif" を以下の場所に置いておきました：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/asteroids/

これらのデータから計算した小惑星それぞれの形状を近似した 3 軸不等楕円体の軸半径 A、B、C を添付したファイル asteroids_abc.txt に書き込みました。

(6) このようにして計算した 3 軸不等楕円体の軸半径の比 A/C と B/C を添付したファイル ratio.txt に書き込みました。残念なことに「1/2:1/√2」に近い比の値になっているものはわずかです：

Mithra.v1.PA.prograde.mod.obj(4486 Mithra)	0.489074	0.587704
Nereus_alt1.mod.wf（4660 Nereus）	0.487155	0.624306
kw4b.obj(1999 KW4 Beta)	0.575918	0.755171

もちろんイトカワもダメです。まぁ、素性を考えずに網羅的に調べただけなので、ダメでもともとのような気もします。今後の検討が必要ですね。

(7) 以上のもの以外の天体の形状データも JPL の PDS（Planetary Data System）のサイトにありました（宇宙研は PDS のミラーサイトだったような気がする）。

http://sbn.psi.edu/pds/archive/shape.html

ここにあるデータ（Eros のものもあります）をこれから調べてみます。

とりあえず以上です。

添付した E-mail の添付ファイル itokawa_abc.txt --- 削除しました。

添付した E-mail の添付ファイル asteroids_obj.txt

http://echo.jpl.nasa.gov/asteroids/shapes/

kleo.obj	216 Kleopatra
betulia.obj	1580 Betulia
geographos.obj	1620 Geographos
bacchus.obj	2063 Bacchus
rashalom.obj	2100 Ra-Shalom
toutatis.obj	4179 Toutatis
hirestoutatis.obj	4179 Toutatis high-resolution
Mithra.v1.PA.prograde.mod.obj	4486 Mithra
Nereus_alt1.mod.wf	4660 Nereus
castalia.obj	4769 Castalia (1989 PB)
golevka.obj	6489 Golevka
sk.obj	(10115) 1992 SK
sf36.v.mod.wf	25143 Itokawa smooth model
1950DA_ProgradeModel.wf	(29075) 1950 DA prograde
1950DA_RetrogradeModel.wf	(29075) 1950 DA retrograde
wt24.obj	(33342) 1998 WT24
ml14.obj	(52760) 1998 ML14
kw4a.obj	(66391) 1999 KW4 Alpha
kw4b.obj	(66391) 1999 KW4 Beta
yorp.obj	54509 YORP (2000 PH5)
ky26.obj	1998 KY26
ce26.obj	2002 CE26 primary

添付した E-mail の添付ファイル asteroids_abc.txt --- 削除しました。

添付した E-mail の添付ファイル ratio.txt --- 削除しました。

Date: Wed, 27 Jul 2011 19:32:42 +0900
From: Tsukasa NAKANO
To: Satoshi Okumura
Cc: Akira TSUCHIYAMA, Yoshito NAKASHIMA, Kentaro Uesugi, Michihiko Nakamura
Subject: porosity_mesurement_using_convex_hull_program

おくむらさま、

GSJ/AIST のなかのです。si_ch（３次元画像上の物体像の凸包抽出と空隙の識別用プログラム）を使って、昨年１月に奥村くんが SPring-8 で撮影した X 線 CT 画像上の変形した発泡マグマの空隙率（vesicularity）を調べてみました。昨年６月頃の E-mails で紹介した ECS 処理に用いた、測定 1001122[k,l,v,w]の８ビット画素値の byte 画像を使いました。

これまでに今回と同じ測定条件で撮影した、今回と同じ原料物質（和田峠の黒曜石）のサンプルの CT 画像の画素値ヒストグラムの解析から、100122[k,l,v,w]の byte 画像では画素値 71～255 の画素をサンプル像（固体）と見なせば良いことがわかっています。以前に報告したように、測定 100122[k,l,v,w] に対して、サンプル像の内部だけが含まれる直方体領域を切り出した、CT 値と画素値の対応関係が byte 画像のものと同じ trim 画像を用いて以下の空隙率の値を得ました：

> On Fri, 04 Jun 2010 21:51:13 +0900, Tsukasa NAKANO wrote:
>
> > 実は、測定 100122[k,l,v,w]の CT 画像に対してあなたが決めてくれた領域を
> > 切り出した後に画素値 70 をしきい値として２値化して ECS を始めました。
> > このしきい値で識別した泡のクラスタラベリングの結果は以下の通りです：
>
> | 測定番号 | P0 | P1 | C |
> |---|---|---|---|
> | 100122k | 0.292415 | 0.212329 | 0.726121 |
> | 100122l | 0.310379 | 0.133505 | 0.430134 |
> | 100122v | 0.480821 | 0.464246 | 0.965526 |
> | 100122w | 0.510978 | 0.507144 | 0.992497 |
>
> > ただし、
> > P0：すべての泡の空隙率（奥村くんの用語で vesicularity ？）
> > P1：最大かつ画像の端から端までつながっている泡のクラスタの体積
> > C= P1 / P0：connectivity

また、以下がこのような２値化した trim 画像の鳥瞰図です：

http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122k_b_bev.mpg
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122l_b_bev.mpg
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122v_b_bev.mpg
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122w_b_bev.mpg

さて、今回はプログラム si_ch を用いて、2000^2×1312 画素の byte 画像の上のサンプル像全体を対象にその空隙率を推定しました。具体的には以下の処理を行いました：

[1] まず、画像上のノイズ除去のために Erosion-Dilation（ED）を行う。

```
mkdir　CH
si_ed　byte　-　71-255　LAYERS　CH
```

See http://www-bl20.spring8.or.jp/~sp8ct/tmp/osp.pdf

[2] その結果の２値画像からサンプルの凸包を抽出し、空隙を識別する。

```
si_ch　CH　-　1　CH
```

ここで LAYERS は ED を加える物体像の表面層の層厚（単位は画素幅）で、今回は測定ごとに０～３の４通りを試しました（ただし、０の場合は ED を行わない）。また、それらそれぞれの結果の画像を入れるディレクトリ CH を "ch_0" ～ "ch_3" としました。

si_ch が出力した画素数 N1、N2、N3 を使って２種類の空隙率 Pa と Pb を算出しました：

すべての空隙の割合 Pa = (N1 + N2) / (N1 + N2 + N3)
サンプル内部の孤立した空隙の割合 Pb = N2 / (N1 + N2 + N3)

ただし、

N3：サンプル（固体）の像の画素の個数
N2：サンプル内部にある空隙の画素の個数
N1：凸包内部にある上記の２種類以外の画素の個数

こうして得た画像 ch_[0,1,2,3] の空隙率は以下の値になりました：

	測定	ch_0	ch_1	ch_2	ch_3
Pa	100122k	0.866620	0.466794	0.323749	0.340809
	100122l	0.889564	0.471848	0.379374	0.402769
	100122v	0.874235	0.685280	0.502506	0.532267
	100122w	0.850542	0.707950	0.629642	0.696383
Pb	100122k	0.008781	0.020993	0.013071	0.006407
	100122l	0.008618	0.021750	0.011197	0.005372
	100122v	0.002932	0.004748	0.003478	0.001327
	100122w	0.000878	0.000883	0.000563	0.000270

また、si_ch が識別した空隙は以下の browse 画像に示したようになっています：

http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122k_ch.gif
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122l_ch.gif

http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122v_ch.gif
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122w_ch.gif

ただし、これらの browse 画像の左端の数値は（byte 画像の）スライス番号で、参考のために ch_[0-3]のものに加えて byte 画像のスライス画像も並べておきました。また、右端にある２種類の画像 ch_[2,3]+0 については後で説明します。

browse 画像や空隙率 Pa の値からわかるように、byte 画像に含まれるノイズのため、層厚が２以上の ED を事前に実行しておかないと尤もらしい凸包にはなりません。また逆に、尤もらしい凸包になっている ch_[2,3]では ED によってサンプル内部の固体の「組織」が壊れています。そのため Pb の値が減少し、Pa が増加して browse 画像上のスライスが「赤っぽく」なっています。

ch_[2,3]の画像をサンプルの内部を識別する「マスク」として使い、かつ、ED を行っていない ch_0 の画像でサンプル内部の組織を評価すればことにすれば、上のような ED によるサンプルの内部組織の破壊を回避できます。例えば ch_2 の画像の画素値１～３の画素がサンプルの内部を指すマスクだとすると、それらの画素の値を ch_0 のもので置き換えた画像を作れば良い訳です。これは具体的には以下のような処理になります：

[1] ch_2 をマスク、ch_0 を「前景」としたマスク処理を行う。

```
mkdir  ch_2+0
si_mask  ch_2  -  1-3  -0  ch_0  -  ch_2+0
```

ここで、プログラム si_mask に指定したパラメータの意味は以下の通り：

ch_2 - 1-3：ch_2 の画像の画素値１～３の画素をマスクに使う。
-0：マスクされていない「背景」の領域には画素値０を埋め込む。
ch_0 -：マスクされた前景領域は ch_0 の画素に置き換える。
ch_2+0：結果の画像をディレクトリ ch_2+0 に格納する。

See http://www-bl20.spring8.or.jp/~sp8ct/tmp/mask+trim.pdf

[2] 作成した画像上の画素値の出現頻度分布を調べる。

```
si_pvr  ch_2+0  -
```

See http://www-bl20.spring8.or.jp/~sp8ct/tmp/pvr.pdf

[3] 必要ならプログラム si_cm_rgb などを使って画像ファイルに色情報を埋め込む。

```
tcm_rgb  ch_0/0000.tif  |  si_cm_rgb  ch_2+0  -  -  ch_2+0
```

See http://www-bl20.spring8.or.jp/~sp8ct/tmp/cm.pdf

ch_[2,3]のそれぞれをマスク画像として上の処理を行い、画像 ch_[2,3]+0 を得ました。それらの画素値の出現頻度分布から先と同様にして算出した空隙率 Pa と Pb の値は以下の通りです：

	測定	ch_2+0	ch_3+0
Pa	100122k	0.310057	0.307466
	100122l	0.362918	0.361371
	100122v	0.479909	0.478844
	100122w	0.571557	0.569120
Pb	100122k	0.045433	0.045605
	100122l	0.049719	0.049840
	100122v	0.012130	0.012155
	100122w	0.002519	0.002534

このようにして計算したサンプル全体の空隙率 Pa は ch_[2,3]+0 でほぼ同じ値になっています：

100122k	0.31	← 2010/6/4 付けの E-mail の P0 の値はおよそ 0.29
100122l	0.36	← 0.31
100122v	0.48	← 0.48
100122w	0.57	← 0.51

これらは先の 2010/6/4 付けの E-mail に記した P0 の値と大小関係は合っていますが、絶対値が違うものもあります。この辺りをもう少し議論しないとダメですが、今日のところはここまで。

Date: Fri, 29 Jul 2011 17:34:46 +0900
From: Tsukasa NAKANO
To: Satoshi Okumura
Cc: Akira TSUCHIYAMA, Yoshito NAKASHIMA, Kentaro Uesugi, Michihiko Nakamura
Subject: Re: ch.pdf

おくむらさま、

なかのです。先程はつれない返事をしましたが、よく考えると、３次元画像を水平方向（画像の z 軸と垂直な方向）から見た時の「側面」が凹になっているだけの「砂時計」状のサンプルなら、プログラム t_ch でスライスごとの凸包を抽出すればその内外を識別できることに気づきました。

> On Fri, 29 Jul 2011 13:16:32 +0900, Tsukasa NAKANO wrote:
>
> > si_ch が行うのは物体像の凸包近似ですから、外形が凹の像には無力です。
> >
> > > On Fri, 29 Jul 2011 13:02:54 +0900, Satoshi Okumura wrote:
> > >
> > > > 凸包プログラムはさっそく使わせてもらってます。
> > > > ただ、なかなか癖がありそうですね。
> > > > 例えばこちらの実験試料で砂時計形のものがありますが、こいつに使ってみると
> > > > 砂時計のくぼみをうまくトレースしてくれなくて、太った形になります。

具体的な処理の説明の前に２つのことをはっきりさせておきます。まず、ぼくは凸包抽出の前に ED を行いますが、これは凸包抽出の必要条件ではありません。尤もらしい凸包の抽出の邪魔になるノイズの除去は ED 以外の方法で行ってもかまいません。逆に言うと、ED と凸包抽出は独立な処理です。それゆえ、以下ではサンプル像の２次元凸包を抽出しますが、その前のノイズの除去には２次元の ED ではなく従来通りの３次元の ED を行うことにします。それから、凸包には物理的な意味がありますが、ここではその抽出を単なる「物体像の内外の識別のためのマスクを作る処理」と割り切ることにします。もしくは、ここで行うのは「z 軸と概ね一致する中心軸を持った円筒状の物体像の場合にのみ使える手法」と言った方が良いかもしれません。

ということで、以下の処理を行ってみました。

[1] まず、byte 画像に対して３次元の ED を行ってノイズを除去する。
[2] プログラム t_ch でそのスライスそれぞれの２次元凸包を抽出する。
[3] この「３次元画像」の上の画素値１～３の画素がサンプルの内部を指すマスクであると見なす。これを用いて、固体、孤立空隙とそれ以外の像を識別した、ED によってサンプル内部の構造が破壊されていない画像をそこに埋め込む。
[4] このようにして作った画像の画素値の出現頻度分布を調べる。
[5] この画像のファイルに色情報を埋め込む。

端末からの具体的な入力内容は以下の通りです：

[1] ここでは層厚が２と３の ED だけを行った。ただし、以前に得た ch_[2,3] の画像を流用したので、新たに ED を行う必要はなかった（端末から何も入力しなかった）。

[2] 以下は２次元凸法を ch_2 の画像から抽出する場合の例：

```
mkdir  ch_2+0_2d          ← 結果を入れるディレクトリを作っておく。
```

以下は C-shell 環境で実行する場合

```
foreach  tiff  (`ls  ch_2`)
  t_ch  ch_2/$tiff  3  ch_2+0_2d/$tiff  > /dev/null
end
```

以下は B-shell 環境で実行する場合

```
for  tiff  in  `ls  ch_2`
  do  t_ch  ch_2/$tiff  3 ch_2+0_2d/$tiff  > /dev/null
done
```

Windows のコマンドプロンプトの場合

for 文を使えば良いはずですが、...。

[3] 以前の E-mail で紹介したマスク処理の際の入力と概ね同じ（画像 ch_2+0_2d をマスクとして、ED で内部構造が破壊されていない ch_0 の画像を前景に埋め込む）：

si_mask ch_2+0_2d - 1-3 -0 ch_0 - ch_2+0_2d

[4] si_pvr ch_2+0_2d -

[5] tcm_rgb ch_0/0000.tif | si_cm_rgb ch_2+0_2d - - ch_2+0_2d

こうして得た ch_[2,3]+0_2d のスライス画像を以前と同様な browse 画像に示しました（比較のため、以前の browse 画像に載せた ch_2+0 などのスライス画像も一緒に並べました）：

http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122k_ch_2d.gif
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122l_ch_2d.gif
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122v_ch_2d.gif
http://www-bl20.spring8.or.jp/~sp8ct/hd1/okumura_bubble/1001_20b2/100122w_ch_2d.gif

これらより、ch_[2,3]+0 に比べて ch_[2,3]+0_2d の画像ではサンプル像（固体；青色）の外部にあった空隙（赤色）が減っていることがわかります。

また、プログラム si_pvr で調べた画素数から Pa（すべての空隙の割合）と Pb（サンプル内部の孤立した空隙の割合）を計算しました：

	測定	ch_2+0_2d	ch_3+0_2d
Pa	100122k	0.272423	0.271165
	100122l	0.316335	0.315100
	100122v	0.458178	0.457272
	100122w	0.550148	0.543232
Pb	100122k	0.047916	0.048007
	100122l	0.053360	0.053469
	100122v	0.012638	0.012663
	100122w	0.002647	0.002692

そして、これまでに調べた「すべての空隙の割合」の値をまとめると以下のようになります：

測定	P0	Pa（ch_[2,3]+0）	Pa（ch_[2,3]+0_2d）
100122k	0.29	0.31	0.272
100122l	0.31	0.36	0.316
100122v	0.48	0.48	0.458
100122w	0.51	0.57	0.547

以前に得た ch_[2,3]+0 の Pa よりも今回得た ch_[2,3]+0_2d の値の方がサンプル像の内部の領域だけを切り出した trim 画像から計算した P0 の値に近いような気がします。とりあえず以上です。