Date: Wed, 14 May 2008 09:44:08 +0900
From: Tsukasa NAKANO
To: Junji Torii
Cc: Shunji KASAMA
Subject: Erosion-&-Dilation-programs

とりいさま、

産総研の中野司です。前回の E-mail から色々あって間があいてしまいました（実は研究室の引っ越しもまだ終わっていません)。slice シリーズに含まれていた、３次元画像上の物体像に対する Erosion（E）や Dilation（D）処理用のプログラム群の改造版を紹介します。旧版（slice*）と新版（si_*）のプログラムの対応関係は以下の通りです（「E+D 処理」や「非対称」などの語の意味は後で説明します）：

旧版	新版	機能
sliceOSP	si_osp2	スライスごとに物体像の外側の表面を皮むきする
sliceOSP3	si_osp3	３次元物体像の外側の表面を皮むきする
sliceED	si_ed	３次元物体像に E+D 処理を行う
sliceDE	si_de	３次元物体像の D+E 処理を行う
ed_asm	si_eda	非対称 E+D 処理を行う
de_asm	si_dea	非対称 D+E 処理を行う

新版のプログラム群の特徴は以下の通りです：

[1] 旧版よりもかなり高速に処理を行うことができる。
[2] Windows の DOS 窓でも走る（ついでながら、書庫ファイルに入れた Windows 用の実行ファイルは Cygwin でも走ります)。
[3] ２次元画像に対して皮むきや E+D、D+E などの処理を行う（デモもしくはテスト用の）プログラム t_*も書いた。

下記の同じ内容の２個の書庫ファイルに新版のプログラムのソースファイルと Windows 用の実行ファイルを入れておきましたので、どうぞご利用下さい：

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

新しいプログラムの説明を以下に書きます。

(1) Erosion（E）と Dilation（D）処理

物体像を含む２値画像を考えます。Erosion と Dilation はそれぞれ以下の内容の処理です：

Erosion（E）処理とは

物体像に属している画素のうちで物体像に属していない画素と隣接しているものを識別する（物体像の表面から画素の「層」を除去する）処理

Dilation（D）処理とは

物体像に属していない画素のうちで物体像に属している画素と隣接しているものを識別する（物体像の表面に画素の「層」を付加する）処理

ただし、「隣接」とは「２個の画素が辺（２次元画像の場合）もしくは面（３次元画像）を挟んで並んでいる状態」のことです。また、物体像の表面の１画素幅の層ごとの E と D の処理は対象となる画像上の画素すべてに対して同時に行います。

これらは単純な処理ですが、そのための効率的なコードを書くのは意外と難しいです。旧版のものはかなり非効率的なコードだったので、それらを大幅に書き換えました。先の書庫ファイルの中のテキストファイル erosion.txt と dilation.txt のそれぞれに新しい E と D の処理の主要部のコードが書き込まれています。

これらのテキストファイルには１、２、３次元画像それぞれに対する E と D 処理用のコードが書き込まれています。１次元画像に対するコードはデモ用ですが、２、３次元画像用のものは本番でもほぼそのまま使用しています（３次元画像用のものは画素値の格納の仕方が異なりますが)。本番用はインクルードファイル e2.h（２次元画像の E 処理用）などに入っています。

これらのコードはそれぞれ１、２、３次元の配列 cell に格納されている２値画像の上の x1 や x2 などで対角点の座標値を指定した矩形領域に含まれている物体像の表面の１画素幅の層の E や D の処理を行います。ここでは cell 全体のコピーを作らずに、対象領域を１度スキャンするだけで処理するようにしています。この時、対象領域のすべての画素に対して E や D の処理を同時に行うようにするにはそれぞれの画素の処理後の値を cell に遅延格納する必要がありますが、それを０、１、２次元の作業用配列 last を用いて行っています。これは、具体的には、E の処理で物体像の表面から除去した画素（画素値が１→０になる）や D の処理で表面に付加した画素（０→１）に対して cell の値を即座に置換せずに last にとりあえずの値２を記録しておき、後になって cell のもとの値が引用されなくなってからその新しい値（０もしくは１）を last の値をもとにして確定（再格納）するようにしています。

erosion.txt の１、２、３次元画像用コードのそれぞれの中に「/* cell[x-1]==1 */」のような注釈行が埋め込まれていますが、これらは「座標値 x-1 にある画素が E 処理の後も物体像に属していること」などを示しています。つまり、E 処理の後も物体像に属している画素に対する条件判断は必ずそこを通ります。

ぼくは E の処理を物体像だけを含む画像領域のトリミングの前処理に使うことが多いです。つまり、「甘い目」のしきい値を与えて識別したおおよその物体像の表面層を数画素分の厚さだけ「皮むき」し、その結果の像がちょうどおさまる長方形や直方体に皮むきした層厚に相当するマージンを加えた領域を切り出してやれば、それはほとんどの場合に処理・解析する対象の物体像を完全に含んでいる、もとの画像よりも画素数が少ない画像なので、その後の処理・解析に有用です。さらに、このような皮むき処理によってX線CT画像に特有の線状のノイズ(偽像;ring artifactsやstreak noise)を除去することもできます。ぼくは E 処理だけを行うプログラム*_osp*（OSP == Object Skin Parer）をこのような用途を想定して書きました。なお、上のような物体像の領域の「ひろがり」を知る際には、物体像の内部の「穴」の表面の皮むきは不要です。そのため、プログラム*_osp*では皮むきの前に「穴埋め」をします。それに用いたアルゴリズムは前に紹介した*_mhl のものと同じです（旧版の皮むきプログラムでは multiple hole labeler とは別のアルゴリズムを使って穴埋めをしていました)。

E 処理だけを行う物体像の皮むきプログラムはぼくが独自に考えたものですが、E と D を組み合わせた Erosion + Dilation 処理はいわゆる画像処理の定番の手法らしいです(ぼくは自分が書いたもの以外の画像処理プログラムを使ったことが皆無なので、この真偽は定かではありません)。また、その逆を行うのが Dilation + Erosion 処理です（これも定番？)。これらの処理内容は以下の通りです：

Erosion + Dilation（E+D）処理では
　２値画像に対して指定した層厚分の E 処理を行い、その後、それと同じ層厚分の D 処理を行う。
Dilation + Erosion（D+E）処理では
　２値画像に対して指定した層厚分の D 処理を行い、その後、それと同じ層厚分の E 処理を行う。

E+D 処理を行えば物体像の表面の「突起」を除去することができます。また、逆に、D+E 処理は物体像内部の「穴」や「湾」を埋めます。なお、これらはいずれも同じ層厚分の E や D の処理を続けて行う「対称」な処理なので、それらの結果の物体像がもとの画像領域からはみ出ることはありません。ぼくが書いた E+D や D+E（および皮むき）処理のプログラム群でも処理結果を入れる画像の画素数はもとの画像のものと同じにしてあります。

つまり、以前に紹介したプログラム*mask* を使えば、E+D や D+E（および皮むき）処理で得られた２値画像はもとの画像とのマスク処理に使用できます。

あまりに多い層厚を指定すると、E+D の処理中に物体像の画素がなくなることがあります（これは皮むきの場合も同様)。この時、ぼくのプログラムはエラー終了します。

D+E の処理中に物体像がもとの画像領域からはみ出してしまうことがあり得ますが、ぼくのプログラムはそれを考慮済みです。

ここでは同じ層厚分の E と D の処理を「対称」と書きましたが、これはもちろん「E と D の処理が完全に対称である」の意味ではありません（もしそうなら、E+D や D+E 処理によって何も起こらないことになります)。長方形もしくは直方体の物体像について考えてみて下さい。E 処理を行うとこれらはもとの形を保ったまま小さくなりますが、D 処理では「角」の部分が欠けた像になってしまいます。E+D や D+E 処理の前後で同じ形を保つのは円もしくは球形の物体像の場合だけです。

E や D の処理に別々の層厚を指定できる「非対称」E+D もしくは D+E 用のプログラム群も用意しました。これらを使えば単体の E や D 処理はもちろんのこと、「対称」な E+D や D+E 処理を実行することもできます。ただし、これらのプログラムは処理結果の物体像がちょうどおさまる領域の画像（通常はもとの画像と異なった画素数の画像）を作ります。この画像をもとの画像のマスク処理に使いたい場合は、以前に紹介した画像上の矩形領域の切り出し（*trim*）や埋め込み（*paste*）用のプログラムを使用する必要があります。

(2) 物体像の外側の表面の皮むきを行うプログラム

t_osp

起動法

t_osp　orgTIFF　rangeList　thickness　{newTIFF}

機能

２次元画像 orgTIFF を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、物体像内部の穴を埋めた後、thickness で指定した層厚（０以上の整数値）の E 処理を行う。画像ファイルの名前 newTIFF を指定すると E 処理の結果の２値画像をそこに格納する。このプログラムは処理の最中に

[1] ２値化によって識別した物体像
[2] E 処理の結果の物体像

のそれぞれがちょうどおさまる長方形領域の対角点（の画素）の座標値(x1,y1)と(x2,y2)の４個の整数値をこの順に、タブコード区切りで１行に並べて標準出力に

書き出す（正常にプログラムが動作すると合計２行の出力が行われる)。

si_osp2

起動法

si_osp2　orgDir　nameFile　rangeList　thickness　{newDir}

機能

ディレクトリ orgDir の下の nameFile で指定された３次元画像を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、スライス画像上の物体像ごとに２次元の穴埋めをした後、thickness で指定した２次元の E 処理を行う。既存のディレクトリの名前 newDir を指定すると E 処理の結果の２値画像をその下の、nameFile で指定された名前の画像ファイルに格納する。このプログラムは処理の最中に t_osp の場合と同様な２種類の３次元物体像がちょうどおさまる直方体領域の対角点の座標値(x1,y1,z1)と(x2,y2,z2)の６個の整数値をこの順に、タブコード区切りでそれぞれ１行に並べて標準出力に書き出す。

注意

これは「金太郎飴」のようなスライス画像ごとの相異が少ない準３次元状のサンプルを想定して作成したプログラムである。その処理内容は t_osp で３次元画像を構成するスライス画像のそれぞれを処理すれば実現できる場合もある。ただし、スライス画像のいずれかに物体像がない（あるいは E 処理によりスライス画像のいずれかの物体像がなくなった）場合に t_osp ではエラーになるが、si_osp2 では３次元画像の全域でそれが起こらない限りエラーにはならない。

si_osp3

起動法

si_osp3　orgDir　nameFile　rangeList　thickness　{newDir}

機能

t_osp の３次元画像版。起動パラメータやプログラムの出力に関することは si_osp2 の場合と同様。

(3-1) 物体像の E+D 処理を行うプログラム

t_ed

起動法

t_ed　orgTIFF　rangeList　layers　{newTIFF}

機能

２次元画像 orgTIFF を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、layers で指定した層厚（０以上の整数値）の E 処理を

行った後に同じ層厚の D 処理を行う。画像ファイル名 newTIFF を指定するとこれらの処理の結果の２値画像をそこに格納する。このプログラムは処理の最中に

[1] ２値化によって識別した物体像
[2] E 処理の結果の物体像
[3] D 処理の結果の物体像

のそれぞれがちょうどおさまる長方形領域の対角点（の画素）の座標値(x1,y1)と(x2,y2)の４個の整数値をこの順に、タブコード区切りで１行に並べて標準出力に書き出す（正常にプログラムが動作すると合計３行の出力が行われる）。なお、E 処理の間に物体像に属する画素がなくなるとエラーになる。

si_ed

起動法

si_ed　orgDir　nameFile　rangeList　layers　{newDir}

機能

ディレクトリ orgDir の下の nameFile で指定された３次元画像を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、layers で指定した層厚の E 処理を行った後に同じ層厚の D 処理を行う。既存のディレクトリの名前 newDir を指定するとこれらの処理の結果の２値画像をその下の、nameFile で指定された名前の画像ファイルに格納する。このプログラムは処理中に t_ed の場合と同様な３種類の３次元物体像がちょうどおさまる直方体領域の対角点の座標値(x1,y1,z1)と(x2,y2,z2)の６個の数値をこの順に、タブコード区切りで１行に並べて標準出力に書き出す。なお、E 処理の間に物体像に属する画素がなくなるとエラーになる。

(3-2) 物体像の D+E 処理を行うプログラム

t_de

起動法

t_de　orgTIFF　rangeList　layers　{newTIFF}

機能

D+E 処理を行うことを除いてこのプログラムの機能は（E+D 処理用の）t_ed のものと同じである。なお、t_ed とは異なり、このプログラムでは E 処理の間に物体像に属している画素がなくなることはない（それに伴うエラーは生じない）。

si_de

起動法

si_de orgDir nameFile rangeList layers {newDir}

機能

D+E 処理を行うことを除いてこのプログラムの機能は（E+D 処理用の）si_ed のものと同じである。なお、si_ed とは異なり、このプログラムでは E 処理の間に物体像に属している画素がなくなることはない（それに伴うエラーは生じない)。

(4-1) 非対称 E+D 処理を行うプログラム

t_eda

起動法

t_eda orgTIFF rangeList erosion dilation {newTIFF}

機能

２次元画像 orgTIFF を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、erosion で指定した層厚（０以上の整数値）の E 処理を行った後に dilation で指定した層厚の D 処理を行う。画像ファイル名 newTIFF を指定するとこれらの処理結果の物体像がちょうどおさまるひろがり（画素数）の２値画像をそこに格納する。このプログラムは処理の最中に

[1] ２値化によって識別した物体像
[2] E 処理の結果の物体像
[3] D 処理の結果の物体像

のそれぞれがちょうどおさまる長方形領域の対角点（の画素）の座標値(x1,y1)と(x2,y2)の４個の整数値をこの順に、タブコード区切りで１行に並べて標準出力に書き出す（正常にプログラムが動作すると合計３行の出力が行われる)。

si_eda

起動法

si_eda orgDir nameFile rangeList erosion dilation {newDir}

機能

ディレクトリ orgDir の下の nameFile で指定された３次元画像を読み込み、その画素値を rangeList に従って２値化して物体像を識別する。その後、erosion で指定した層厚の E 処理を行った後に dilation で指定した層厚の D 処理を行う。既存のディレクトリの名前 newDir を指定するとこれらの処理結果の物体像がちょうどおさまる画素数の２値画像をその下の、スライス番号（０以上の整数値）を名前とする画像ファイルに格納する。このプログラムは処理中に t_eda の場合と同様な３種類の３次元物体像がちょうどおさまる直方体領域の対角点の座標値(x1,y1,z1)と(x2,y2,z2)の６個の数値をこの順に、タブコード区切りで１行に並べて

標準出力に書き出す。

(4-2) 非対称 D+E 処理を行うプログラム

t_dea

起動法

t_dea orgTIFF rangeList dilation erosion {newTIFF}

機能

非対称 D+E の処理を行うことを除いてこのプログラムの機能は（非対称 E+D 処理用の）t_eda のものと同じである。

si_dea

起動法

si_dea orgDir nameFile rangeList dilation erosion {newDir}

機能

非対称 D+E の処理を行うことを除いてこのプログラムの機能は（非対称 E+D 処理用の）si_eda のものと同じである。

長くなりました。とりあえず、以上です。

P.S.

ここで紹介したプログラムはすべて２値化した画像を計算機のメモリに保持して処理を行います。1000×1000×1000 画素の３次元画像なら必要なメモリの量はおよそ 128 MB です。また、皮むきプログラム*_osp*ではクラスタラベリング用にさらにメモリが必要です。こちらの処理に要するメモリの量は物体像の形状の複雑さに依存するので正確な見積もりが不可能ですが、３次元画像ならその総画素数の 27/64 = 0.421875 倍のバイト数（1000×1000×1000 画素ならおよそ 430 MB）を越えることはないハズです（これは物体像に属する画素を乱数で決めた random site percolation の場合の上限値です）。