

高速多値画像ラベリング手法を用いた画像解析

Image analysis via high-speed multivalued labeling technology

Author: Cryst

高速堅牢な多値ラベリング，大規模データ処理（Voxel を含む）

1 はじめに

伝統的な画像処理[1,2]では、濃淡画像から2値画像を作り、2値画像にラベリング処理[3]を行い、特徴量を算出する手法が多く用いられてきた。2値画像用のラベリング処理では、処理可能な対象物の数の上限が数千から数万個程度に制限されている事が多い。

本処理で使用しているラベリング手法は、ラスタースキャン方式で多値データのラベリング処理を高速に行う。シンプルで拡張性も高く3次元のボクセルデータのラベリング等への応用も可能である。1億ピクセル以上の高解像度画像や、100万個を超える対象物のラベリング処理が容易に行えるので、幅広い利用法が期待出来る。

2 手法に関して

様々な画像処理の局面において利用されるラベリング処理ではあるが、精度、ロバストさ、速度、経済性で満足できるものが入手できなかったので、仕様を満たす多値データに対応した高速ラベリングソフトの独自実装を行った。

従来から行われている一般的な画像処理（2値画像処理）では、次の様な流れで処理が行われてきた。

画像取得 ⇒ 前処理 ⇒ 2値化(対象物の抽出)
⇒ ラベリング ⇒ 特徴量計測

今回提案する手法を用いれば、この手順を以下の様に変更する事が可能となる。

画像取得 ⇒ 画像の微小領域分割 ⇒ 微小領域毎のラベリング(多値データ) ⇒ ラベリングされた領域の特徴量計測 ⇒ 計測済みの特徴量を加味した対象物の抽出 ⇒ 対象物の特徴量計測

この手法は、ミクロ的な領域に着目しその領域から構成される対象部分を認識する為、従来手法では問題となる輝度ムラ(シェーディング)にも対処(隣接領域の変化は少ない)できる。

この多値ラベリング処理は、小さなアレンジではあるが、新たな画像処理アルゴリズムや解析手法の開発が可能となるので、大きな結果が期待できる。ここでは多値データ用のラベリングを用いて、従来の領域分割処理を再評価する。

3 高速多値画像ラベリング

ラベリングの性能は、処理時間が画素数に線形比例する事が望ましい。処理時間の比較が容易に出来る様に、処理時間の基準として、カラー画像の3x3平滑化処理[100]を使用する。以下に画像中の画素数と処理速度(平滑化とラベリング処理)の関連性をグラフで示す。

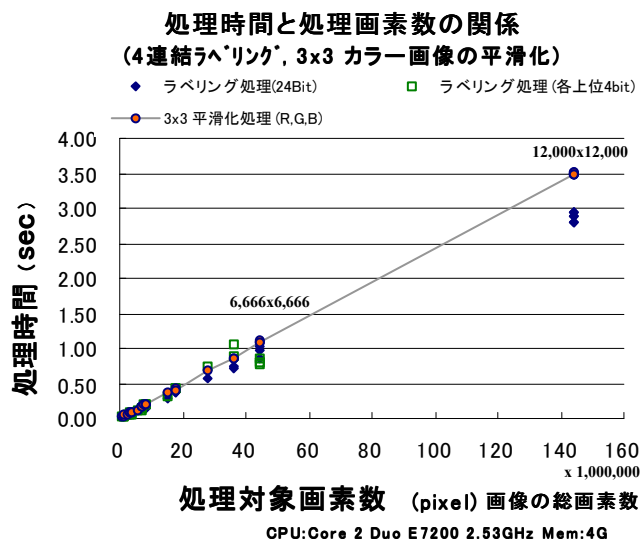


図1 画像中の画素数と各処理時間の関係

平滑化処理は、人間は意識しないがコンピュータで画像を処理する場合に問題となる隣接画素の過剰な色変化を除去するために用いる。図2の原画で確認出来る色の変化を、3x3平均化フィルター処理を1回施す事で、図3で示す様に隣接画素間の色変化はかなり抑えられる。

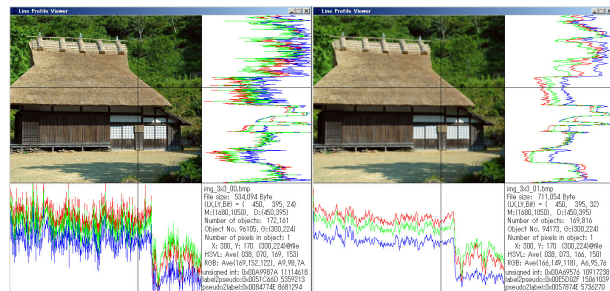


図2 原画

図3 3x3 平均化 1回

4 領域分割処理（応用処理例）

4.1 領域分割処理の流れ

本手順での処理フローを図4に示す。

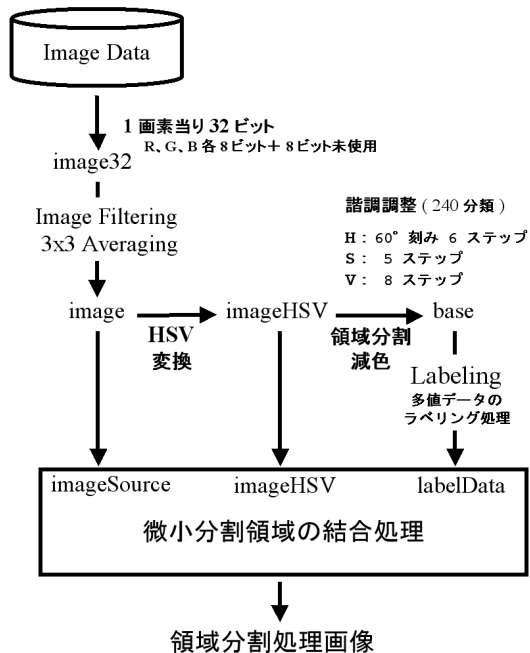


図4 領域分割処理の手順

入力された画像を、RGB 各8ビットで構成される画像に変換し、更に平滑化処理を行う。

同画像を、色情報を表現するHSV系に変換し、同時に諧調数を減らす事で領域を分割した画像を生成する。領域分割された画像に、多値ラベリング処理を行い、各領域を認識可能にする。ラベルデータと画像より、各領域内の特徴量計測を行う。

この分割領域は、過剰分割されているので、各領域の特徴量と、領域と領域が接している部位の画素の特徴量から、領域の同一性判断ルールを作成し概ルールに従って領域の結合を行う。

4.2 処理の経過と結果

処理の流れに沿って、処理結果を示す。



図5 入力画像

図6 領域分割したデータ

読み込んだ画像データを、画素の色と明るさの情報を基準にし、11,000以上の小領域（セル）に分割。

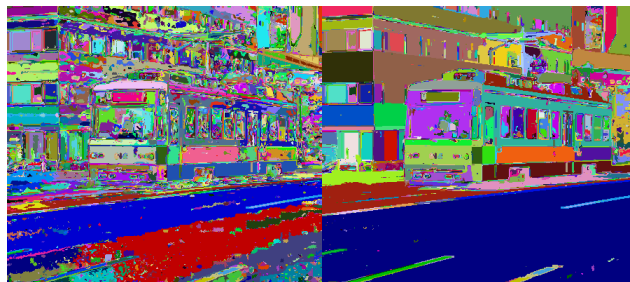


図7 ラベリング後

図8 領域分割画像

擬似カラー化

分割した領域にラベリング処理を施し、領域毎に特徴量を計測する。原画、HSVデータ、ラベル情報、各領域の計測結果を利用して、領域を結合

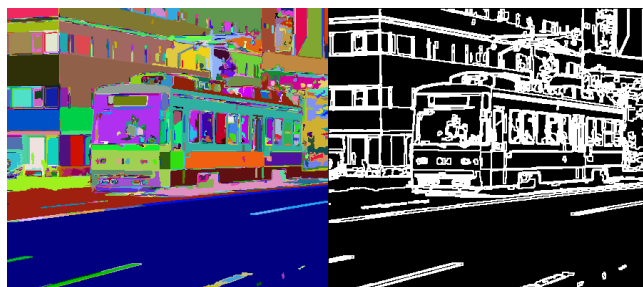


図9 領域分割後

図10 領域の境界部分

微小領域除去

不要な微小領域を除去し、包括する領域に統合した結果と、境界部分を強調した画像。

本手法によって画像中の領域認識が容易に行える事が確認できた。本例では、領域の同一性判断に単純なルールのみを適応しているので、分割精度が高くない事にも言及しておく。

5 終わりに

今回選択した処理は単なる一例で、ラベリング処理は非常にプリミティブな技術であり、今まで普通に行われていなかった、多値データに対応したラベリング処理が、許容できる処理時間内で利用できる事は、従来の研究され尽くされた手法や処理にも、新しい手法を加える事が可能となり、新たな発展が期待できる。

また、本ラベリング処理のコンセプト [101] は、ラベリングに限定されたものではなく汎用的な概念であり、本処理の領域情報管理にも同じアルゴリズムを使用している。概アルゴリズムは、2つ以上のメンバーにより構成される範囲において、任意の2つのメンバーを比較し同一性が見出せる場合に、2つのメンバーに結合情報を付与する手段をもちい、範囲の全メンバーを1度検査（交換法則が成り立つ事が前提）すれば、相互の結合性を決定できる事の特徴とする手法（芋蔓方式：one after another）であり、ラベリングはその特殊例の1つに過ぎない事を付記する。

6 参考文献

- [1] 村上伸一 画像処理工学 東京電機大学出版局 2004 (ISBN4-501-32370-1)
- [2] 田村 秀行 コンピュータ画像処理 コンピュータ画像処理 オーム社 2002 (4-274-13264-1)

- [3] HE Lifeng, CHAO Yuyan, SUZUKI Kenji, NAKAMURA Tsuyoshi, ITOH Hidenori "高速2回走査ラベル付けアルゴリズム(パターン認識) An Efficient Two-Scan Connected-Component Labeling Algorithm" 電子情報通信学会論文誌. D, 情報・システム J91-D(4), 1016-1024, 2008-04-01

参考資料

Copyright (c) 2011, Cryst.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the Cryst nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2011, Cryst.
All rights reserved.

ソースコード形式かバイナリ形式か、変更するかしないかを問わず、以下の条件を満たす場合に限り、再頒布および使用を許可します。

- * ソースコードを再頒布する場合、上記の著作権表示、本条件一覧、および 下記免責条項を含めること。
- * バイナリ形式で再頒布する場合、頒布物に付属のドキュメント等の資料に、上記の著作権表示、本条件一覧、および下記免責条項を含めること。
- * 書面による特別の許可なしに、本ソフトウェアから派生した製品の宣伝または販売促進に、Cryst の名前またはコントリビューターの名前を使用してはならない。

本ソフトウェアは、著作権者およびコントリビューターによって「現状のまま」提供されており、明示黙示を問わず、商業的な使用可能性、および特定の目的に対する適合性に関する暗黙の保証も含め、またそれに限定されない、いかなる保証也没有。著作権者もコントリビューターも、事由のいかんを問わず、損害発生の原因いかんを問わず、かつ責任の根拠が契約であるか厳格責任であるか（過失その他の）不法行為であるかを問わず、仮にそのような損害が発生する可能性を知らされていたとしても、本ソフトウェアの使用によって発生した（代替品または代用サービスの調達、使用の喪失、データの喪失、利益の喪失、業務の中断も含め、またそれに限定されない）直接損害、間接損害、偶発的な損害、特別損害、懲罰的損害、または結果損害について、一切責任を負わないものとします。

[100] 処理速度の基準用 3x3 平滑化(カラー画像)プログラム (C 言語: gcc-3.4.4 -O3 -Wall -m486)

```
typedef struct PIXEL { unsigned char B, G, R, A; } PIXEL ;
// ライセンスに関しては、前頁を参照の事
void **alloc2D( void *data, int sizeofUnit, int lx, int ly )
{ // 連続データを、配列データとしてアクセス可能にする
register void **d, *base=data ;
register int i, s=sizeofUnit, llx=lx, lly=ly ;

    if( (data==NULL) || (sizeofUnit<1) || (lx<1) || (ly<1) ) return NULL ;
    if( (d=(void **)malloc(ly*sizeof(void *))) == NULL ) return NULL ;
    for( i=0; i<lly; i++ ) d[i] = (void *)((unsigned char *)base + llx*s*i) ;
    return d ;
}

int filterAve3x3(PIXEL *src, PIXEL *dst, int wlx, int wly)
{ // カラー画像を R, G, B 各チャンネル毎に平滑化処理する
register int x, y, lx=wlx, ly=wly, ex=lx-1, ey=ly-1, rw, gw, bw, st=1 ;
register PIXEL **w=(PIXEL **)alloc2D(dst, sizeof(PIXEL), wlx, wly) ;
register PIXEL **p=(PIXEL **)alloc2D(src, sizeof(PIXEL), wlx, wly) ;

    if( (p==NULL) || (w==NULL) || (wlx<3) || (wly<3) ) goto GOOD_BY ;
    for( y=0; y<ly; w[y][0]=p[y][0], w[y][ex]=p[y][ex], y++ ) ;
    for( x=0; x<lx; w[0][x]=p[0][x], w[ey][x]=p[ey][x], x++ ) ;
    for( y=1; y<ey; y++ ) {
        for( x=1; x<ex; x++ ) {
            rw = p[y-1][x-1].R + p[y-1][x].R + p[y-1][x+1].R
                + p[ y ][x-1].R + p[ y ][x].R + p[ y ][x+1].R
                + p[y+1][x-1].R + p[y+1][x].R + p[y+1][x+1].R ;
            gw = p[y-1][x-1].G + p[y-1][x].G + p[y-1][x+1].G
                + p[ y ][x-1].G + p[ y ][x].G + p[ y ][x+1].G
                + p[y+1][x-1].G + p[y+1][x].G + p[y+1][x+1].G ;
            bw = p[y-1][x-1].B + p[y-1][x].B + p[y-1][x+1].B
                + p[ y ][x-1].B + p[ y ][x].B + p[ y ][x+1].B
                + p[y+1][x-1].B + p[y+1][x].B + p[y+1][x+1].B ;
            w[y][x].R = rw / 9 ; w[y][x].G = gw / 9 ; w[y][x].B = bw / 9 ;
        }
    }
    st = 0 ;
GOOD_BY: free(p) ; free(w) ; return st ;
}
```

[101] アルゴリズムの骨格部 (ラベリング及び、統合処理用) の Cryst によるサンプル実装

```
// C でコンパイル可能コード (構造体の定義は必要) ライセンスに関しては、前頁を参照の事
LEAD      scanningPoint, searchPoint ;
TARGET    something, neighborhood ;
OBJECT    thing1, thing2 ;

for( scanningPoint.here=0; scanningPoint.here<ScanningField; scanningPoint.here++ ) {
    if( (something=PickUp(scanningPoint)).entity == EUREKA ) { // Scanning MODE
        if ( something.type == ATOM ) //Registration or Acquisition of object
            thing1 = registrationAsObject(something, getObjectGroup(NEWGROUP)) ;
        else thing1 = getObjectGroup(something) ; // something.type == OBJECT
        SearchField = (searchPoint=initSearchField(scanningPoint)).field ;
        for( searchPoint.here=0; searchPoint.here<SearchField; searchPoint.here++ ) {
            if( (neighborhood=PickUp(searchPoint)).entity == EUREKA ) { // Search MODE
                if ( neighborhood.type == ATOM ) // Registration or Acquisition of object
                    thing2 = registrationAsObject(neighborhood, getObjectGroup(NEWGROUP)) ;
                else thing2 = getObjectGroup(neighborhood) ; // neighborhood.type == OBJECT
                if( checkOfRelationForConfirmation(thing1, thing2) == CANDIDATES ) {
                    if( checkObjectAttribute(thing1, thing2) == SAME) binding(thing2, thing1) ;
                }
            }
        }
    }
}
```

要素のグループ化アルゴリズムに関する記載

対象物（特定グループに属する要素群）は、ATOM（1 番小さな構成単位）の 1 つ以上の集まりであり、同一対象物を構成する可能性がある（隣接した）ATOM が同じ属性を持つ場合、隣接した対象物は 1 つの対象物（同一のグループに帰属）であると定義する。

- 1、対象領域内を走査し処理すべき ATOM を探す。
- 2、見つかった ATOM が登録済みの対象物(の一部)でない場合は、新たに対象物として登録する。
（概対象物を、対象物 A とする）
- 3、関連性を持つ可能性のある ATOM 又は対象物（概対象物を、対象物 B ※ 1 とする）を探索する。【 2 次元のラベリング処理の場合は、周辺（隣接条件を満たす、4 連結又或いは 8 連結）を探索。】未登録の ATOM が見つければ、新規の対象物（対象物 B）として登録する。
※ 1：対象物 B は、1 つ以上の対象物の総称
- 4、関連性を持つ可能性のある対象物 B が探索により見つかった場合は、処理対象の同一性を決定する属性に従い、対象物 A と対象物 B の同一性判定を行い、同一であると判断できる場合は、1 つの対象物として、統合する処理を行う。
【2次元のラベリング処理の場合、同一性があり1つの対象物であるとの判断基準は、対象物A、対象物Bが隣接した、異なる対象物と認識された場合】
- 5、全領域走査の完了により、全対象物に対する同一性の確認処理が完了する