LI-9

３×３メディアンフィルタの高速アルゴリズム

A Fast Algorithm for 3x3 Median Filtering

浜村倫行† 入江文平†
Tomoyuki Hamamura Bunpei Irie

1．まえがき

画像処理において，従来，各画素周辺に一定の大きさの画像領域（ウィンドウ）を設定し，各画素が持っているデータ（画素値）をウィンドウ内の全データの中央値に置き換える処理を行うメディアンフィルタが知られている．メディアンフィルタは画像データの解像度を損なうことなくスパイクノイズのような孤立点を除去できる特徴をもっており，ノイズ除去手法の一つとして使われている．ウィンドウのサイズは３×３のものがよく用いられている．

メディアンフィルタの一般的な計算方法としては，ウィンドウ内のデータをソートし，中間順位のデータを得る方法がある．例えば，３×３メディアンフィルタで，選択（単純）ソートを用い，昇順に 5 番目のデータが得られた時点でやめるものとすると，比較回数は 30 回となる．

また，３×３に限らず，一般に n 個のデータの中で t 番目に小さいデータを見つけるアルゴリズムとして，Find[4],Select[5]等がある．しかし，その高速性はデータ数が十分多いときに有効なものであり，データ数が少ないときは，データ数に特化したアルゴリズムの方が高速である．

９個のデータに特化したアルゴリズムとしては，Smith によるもの[6]がある．比較回数は 19 回である．

また，隣接画素のウィンドウが重なることを利用して高速化を図る方法もある．文献[2]では，３×３に限らず，一般にウィンドウが重なるメディアンフィルタの各種アルゴリズムについて比較実験している．しかし，やはりデータ数が少ないときには有効ではない．また，３×３に限らず，データの取る値の種類 U が少ないときに有効な方法として，ヒストグラムを用いた方法がある[2][4]．比較回数は，３×３のとき，平均 8+d 回，最大 8+U 回程度である（d は隣接する中央値の差の平均）．

３×３に特化し，かつ隣接画像のウィンドウが重なることを利用した方法として，Kopp によるもの[1]がある．隣接する 2 画素の中央値を同時に求めるもので，比較回数は，縦方向の隣接を考慮しないものが平均 8.87 回，最大 9.5 回であり，考慮したものは更に 0.5 回少ない．

本稿では，これらの従来手法より高速な，３×３メディアンフィルタのアルゴリズムを提案する．

2．９個のデータの中央値を求めるアルゴリズム

まず，一般に９個のデータの中央値を求める高速なアルゴリズムを提案する．ウィンドウの重なりが利用できない場合にも用いることができる．

2.1 アルゴリズム

<step 1> 与えられた 9 個のデータを，3 個ずつの 3 組（この組を unit と呼ぶ）に分け，unit ごとにソートをする．

<step 2> unit を中央値によってソートする．

図１が step 2 終了後の様子である．データを丸，unit を長方形で示す．不等号でデータの大小関係が表しており，各 unit 内では大きいものが上に書かれている．各 unit について，中央値の大きい順に，A，B，C とする．

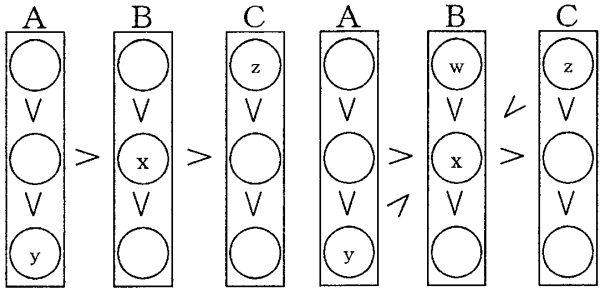

図１ step 2 終了後

図２ case 1 のとき

ここで，unit B の中央値 x に注目すると，x 以上のデータが 3 個，x 以下のデータが 3 個確定している．x との大小関係が確定しないデータは 2 個であり，図１における y，z である．y は unit A で最小，z は unit C で最大のデータである．

<step 3> y と z について，x との大小関係を調べる．

step 3 の結果により，次の３通りに場合分けされる．

● case 1 y<=x<=z もしくは z<=x<=y のとき

この場合，x が中央値であることが確定する．

● case 2 x<y かつ x<z のとき

この場合，x が大きい方から 6 番目のデータであることが確定する．図２にその様子を示す．

すると，中央値は，x 以上のデータ 5 個の中で最小のものになる．x 以上のデータ 5 個とは，unitA に属する３個のデータと，unitB で最大のデータ（w とする）と，z であるが，unit A に属する 3 個のデータの中では y が最小であるため，中央値になり得るデータは y，z，w の３個となる．

<step 4> y，z，w の最小値を求める．

step 4 の結果が，case 2 のときの中央値である．

● case 3 x>y かつ x>z のとき

case 2 と対称であり，同様の処理で中央値を決定できる．

2.2 性能の理論的評価

本アルゴリズムの性能を，比較回数という観点で見る．

まず，一般的に，３個のデータのソートにかかる比較回数は，2 回になる確率が 1/3，3 回になる確率が 2/3 であるため，平均 8/3 回，最大 3 回である．

本アルゴリズムにおいては，step 1，step 2 で「３個のデータのソート」を 4 回行い，step 3 で 2 回の比較が行われる．ここまでで，合計の比較回数は以下のようになる．

平均 $C_{ave}'=8/3\times4+2=38/3$ 回

最大 $C_{max}'=3\times4+2=14$ 回

その後，case 2，case 3 になった場合のみ比較が行われる．case 2 では step4 を行うのに，比較が 2 回行われる．case 3 も同じである．ここで，平均の比較回数を求めるために，case 1,2,3 になる確率 P_1,P_2,P_3 を求めておく．case 2 となるの

†㈱東芝 柳町 e-ソリューション工場 要素技術部

は，9個のデータを大きい順に $n_1,n_2,\ldots,n_9$ とすると，n_6,n_7,n_8,n_9のうち2つが同じunitに入り，残り2つも別の同じunitに入った場合である．この確率は以下である．

$P_2=\{3\times(5\times4)\times3!\times3!^3\}/9!=3/14$

$P_3=P_2$であり，case 1になる確率は $P_1=1-P_2-P_3=4/7$ である．よって，場合分け後の比較回数は，

平均 $C_{ave}''=P_1\times0+P_2\times2+P_3\times2=6/7$回

最大 $C_{max}''=2$回

となる．以上より，本アルゴリズムにおける比較回数は，

平均 $C_{ave}=C_{ave}'+C_{ave}''=284/21\fallingdotseq13.52$回

最大 $C_{max}=C_{max}'+C_{max}''=16$回

である．Smithの方法における比較回数（19回）を，最大でも下回り，平均的には28.8%も下回っている．

3. 3×3メディアンフィルタへの応用

ここでは，前章で提案したアルゴリズムを，3×3メディアンフィルタに応用し，本手法が3×3メディアンフィルタと非常に親和性が高く，高速に動作するアルゴリズムであることを説明する．

画像に用いられる3×3メディアンフィルタの場合，隣接した画素ではウィンドウが重なる．これを利用することで，高速化が可能である．

画像を幅w、高さhとし、iを横方向，jを縦方向，左端i=0，右端i=w-1，上端j=0，下端j=h-1とする．位置[i,j]の3×3ウィンドウを $W_{i,j}$ とする．横優先のラスタスキャンでメディアンフィルタ処理を行うものとする．

ウィンドウを横に3つに分け，横1×縦3の3つの領域にし，それぞれをunitとして，前章のアルゴリズムを適用する．その様子を図3に示す．すると，先に $W_{i-1,j}$ の中央値を求めているので，3つのunitのうち2つでは前のソート結果を用いることができ，比較回数を「3個のデータのソート」2回分削減できる．

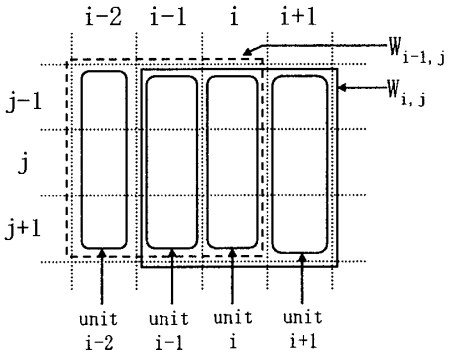

図3 隣接ウィンドウとunitの関係

更に，Koppの方法[1]と同様のアイデアを用い，横に隣接した2画素について同時にメディアンフィルタ処理をした場合を考える．step 2において，両ウィンドウに含まれる2つのunit（図3のunit i-1, unit i）同士の比較を先に行うことで，比較回数を1回減らすことができる．画像1画素あたりに直せば，比較回数1/2回の削減になる．

以上の工夫により，画像1画素あたりの比較回数は，

平均 $C_{ave2}=C_{ave}-8/3\times2-1/2=323/42\fallingdotseq7.69$回

最大 $C_{max2}=C_{max}-3\times2-1/2=9.5$回

となる．従来手法で最も高速である，Koppの方法と比べると，比較回数で見て，平均回数は13.3%減り，最大回数は変わらない．

4. 更なる高速化

更に高速化するために，Rankaらのテクニックを応用する．1次元のデータ系列に対し，幅3のウィンドウのメディアンフィルタを系列順に連続に施す問題を考える．ウィンドウの重なりを利用しない場合は，「3個のデータのソート」と同様の比較回数がかかるが，Rankaらは，ウィンドウの重なりを利用し，平均，最大，共に1回削減するアルゴリズムを考案している[3]．

この方法を用いると，中央値だけでなく，最小値と最大値も同時に得ることができる．つまり，幅3のウィンドウ内のソート結果を得られたことになる．

そこで，3章で用いたKoppの方法と同様のアイデアの代わりに，Rankaらのテクニックを用いることができる．これにより，step2における比較回数の削減は，1/2回→1回になり，より高速化される．

更に，3章では横方向の重なりのみを利用したが，縦方向の重なりを利用することもできる．step 1の，unit内のソートにおいて，縦方向の重なりを考えると，Rankaらの方法を適用できる．これにより，更に比較回数を1回削減できる．以上の方法により，画像1画素あたりの比較回数は，

平均 $C_{ave3}=C_{ave2}+1/2-1-1=130/21\fallingdotseq6.19$回

最大 $C_{max3}=C_{max2}+1/2-1-1=8$回

となる．Koppの方法に比べ，平均で30.2%，最大で15.8%削減している．

ただし，Rankaらの方法は，比較回数は減少するが，その他の演算などが増えるため，実際にはそれほど速くはならないことに注意する必要がある．また縦方向の重なりの利用は，縦方向はキャッシュに乗りにくいため逆に遅くなる可能性もある．

5. まとめ

本稿では，3×3メディアンフィルタの高速なアルゴリズムを提案した．従来手法に比べ，平均比較回数で見て，ウィンドウの重なりを使えない場合で28.8%，使える場合で13.3%，減らすことができた．更に，Rankaらのテクニックの応用により，重なりを利用できる場合は30.2%減らすことができた．但し，Rankaらのテクニックによる削減は処理時間短縮と比例はしない．

参考文献

[1] Manfred Kopp, "Efficient 3x3 Median Filter Computations," Machine Graphics & Vision, pp. 79-82, Vol. 4, No. 1/2, 1995.

[2] Martti Juhola, Jyrki Katajainen and Timo Raita,"Comparison of Algorithms for Standard Median Filtering," IEEE Transactions on Signal Processing, pp. 204-208, Vol. 39, No. 1, Jan 1991.

[3] Sanjay Ranka and Sartaj Sahni, "Efficient Serial and Parallel Algorithms for Median Filtering," IEEE Transactions on Signal Processing, pp. 1462-1466, Vol. 39, No. 6, Jun 1991.

[4] 島内剛一他，"アルゴリズム辞典," 共立出版，1994.

[5] R.W.Floyd and R.L.Rivest, "Expected Time Bounds for Selection, Comm.ACM, 18, 3 (1975), 165-172.

[6] John Smith, "Implementing median filters in XC4000E FPGAs," http://www.xilinx.com/xcell/xl23/xl23_16.pdf