

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†

Tomoyuki Hamamura

入江文平†

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持っており、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、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。

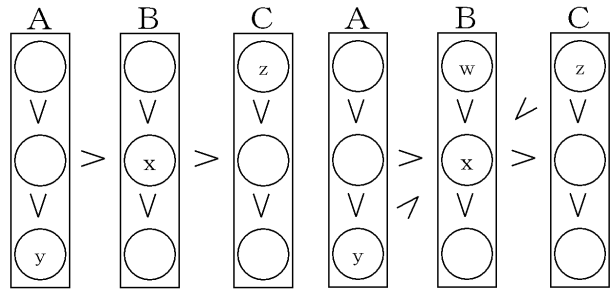


図1 step 2 終了後

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 $y \leq x \leq z$ もしくは $z \leq x \leq y$ 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、unit Aに属する3個のデータと、unit B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と、

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ようになる。

平均 $C_{ave} = 8/3 \times 4 + 2 = 38/3$ 回

最大 $C_{max} = 3 \times 4 + 2 = 14$ 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step 4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

† (株)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, \dots, 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。

$$P_2 = \{3 \times (5 \times 4) \times 3! \times 3!^3\} / 9! = 3/14$$

$P_3 = P_2$ 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、

$$\text{平均 } C_{ave}'' = P_1 \times 0 + P_2 \times 2 + P_3 \times 2 = 6/7 \text{ 回}$$

$$\text{最大 } C_{max}'' = 2 \text{ 回}$$

となる。以上より、本アルゴリズムにおける比較回数は、

$$\text{平均 } C_{ave} = C_{ave}' + C_{ave}'' = 284/21 \quad 13.52 \text{ 回}$$

$$\text{最大 } C_{max} = C_{max}' + C_{max}'' = 16 \text{ 回}$$

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。

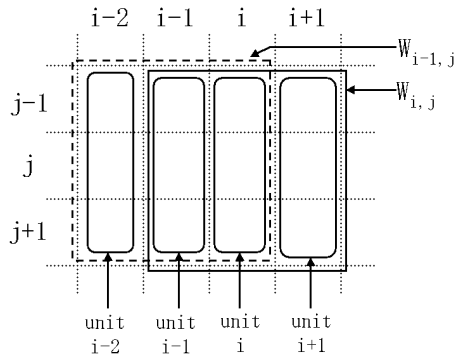


図3 隣接ウィンドウと unit の関係

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) 同士を先に比較を行うことで、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、

$$\text{平均 } C_{ave2} = C_{ave} - 8/3 \times 2 - 1/2 = 323/42 \quad 7.69 \text{ 回}$$

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

となる。従来手法で最も高速である、Kopp の方法と比べると、比較回数で見ても、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、

$$\text{平均 } C_{ave3} = C_{ave2} + 1/2 - 1 - 1 = 130/21 \quad 6.19 \text{ 回}$$

$$\text{最大 } C_{max3} = C_{max2} + 1/2 - 1 - 1 = 8 \text{ 回}$$

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利用は、縦方向はキャッシュに乗りにくい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も、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