I-036

重みの空間分解による高速反復バイラテラルフィルタ
Fast Repetitive Bilateral Filter with Spatial Decomposition of Weight

常 センᵗ　井上 光平ᵗ　浦浜 喜一ᵗ
Chang Jian　Kohei Inoue　Kiichi Urahama

あらまし

バイラテラルフィルタを画像に複数回反復してかけるときに，フィルタの加重係数を固定し，縦方向と横方向に分解して高速化する手法を提案する．

1. まえがき

バイラテラルフィルタ (以下 BF と略す) は画像処理のみならず，コンピュータビジョンやコンピュータグラフィックスなど広い分野で利用が進んでいる [1]．イラスト画作成などのノンフォトリアリスティックレンダリング (NPR) にも利用されているが，BF は指数関数を含み計算量が多いので，NPR などの高速処理が必要な応用のために，近似計算による高速化法が開発されている [2, 3]．それらは BF を 1 回だけ画像にかける場合を想定しているが，実際には 1 回だけでは平滑化が不十分なため，複数回反復して BF がかけられることが多い．そこで本論文では，BF を複数回反復して画像にかける場合に有効な高速化法を提案する．

また，NPR への応用等では BF の階段効果 [4] による擬似輪郭も解消する必要があり，これについても本提案法により改善されることを示す．

2. 定加重反復 BF

BF の出力画素値 f_{ij} は，線形フィルタと同様に入力画素値 d_{ij} の加重平均

$$f_{ij}=\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl}d_{i+k,j+l}/\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl} \quad (1)$$

で与えられる．重みは $w_{ijkl}=e^{-\alpha(k^2+l^2)-\beta(d_{ij}-d_{i+k,j+l})^2}$ であり，重みに d_{ij} を含むので BF は非線形である．

2.1 反復 BF

画像に BF を複数回反復してかける場合，初期値を $f_{ij}^{(0)}=d_{ij}$ として，第 ξ 回めの出力画素値は

$$f_{ij}^{(\xi)}=\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl}^{(\xi-1)}f_{i+k,j+l}^{(\xi-1)}/\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl}^{(\xi-1)} \quad (2)$$

で与えられる．重みは $w_{ijkl}^{(\xi-1)}=e^{-\alpha(k^2+l^2)-\beta(f_{ij}^{(\xi-1)}-f_{i+k,j+l}^{(\xi-1)})^2}$ である．この反復 BF では重みが反復ごとに変わるので，毎回 1 画素当り指数関数を $(2p+1)^2$ 回計算する必要がある．BF の難点は，このように計算量が多いことと，後の図 2(a) に見られるような階段効果 [4] による擬似輪郭である．

2.2 定加重反復 BF

そこでここでは，この両難点の解消策として，重みを入力画像での値 $w_{ijkl}^{(\xi-1)}=w_{ijkl}=e^{-\alpha(k^2+l^2)-\beta(d_{ij}-d_{i+k,j+l})^2}$ に固定することにし，これを定加重反復 BF と呼ぶ．これは

[配列の作成]

Step 1) 全ての i,j,k,l について w_{ijkl} を計算して配列に保存する．

[フィルタ反復]

Step 2) $f_{ij}^{(\xi)}=\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl}f_{i+k,j+l}^{(\xi-1)}/\sum_{k=-p}^{p}\sum_{l=-p}^{p}w_{ijkl}$

を繰り返す．

という手順で計算すれば重みの計算が 1 回だけで済み，重みを毎回計算しなければならない式 (2) の反復 BF よりも計算量が少なくなる．Step 2 の演算量は毎回 1 画素当り $(2p+1)^2$ の乗算と $8p(p+1)$ の加算および 1 回の除算である．但し，メモリ使用量が 2.1 節の反復 BF よりも $(2p+1)^2\times$ 画素数だけ増える．

この定加重反復 BF の 1 回めは BF と同じであるが，2 回め以降では重みは画素値 $f_{ij}^{(\xi-1)}$ によらないので線形フィルタとなり，後の図 2(c) のように階段効果は生じない．なお，2 回め以降の線形フィルタでもエッジ付近の重みは小さいので主要エッジは保存される．

3. 重みの空間分解による高速化

定加重反復 BF は式 (2) の反復 BF よりも高速であるが，景らの手法 [2] に倣って重みを分解近似して計算量とメモリ使用量を更に減らす．

3.1 反復 BF の高速化

定加重反復 BF の分解近似を述べる前に 2.1 節の反復 BF の分解近似法を示す．式 (2) の重みを i 方向と j 方向の重みの積に分解して $w_{ijkl}^{(\xi-1)}\cong u_{ijk}^{(\xi-1)}v_{i+k,jl}^{(\xi-1)}$ と近似する．ここで $u_{ijk}^{(\xi-1)}=e^{-\alpha k^2-\beta(f_{ij}^{(\xi-1)}-f_{i+k,j}^{(\xi-1)})^2}$, $v_{ijl}^{(\xi-1)}=e^{-\alpha l^2-\beta(f_{ij}^{(\xi-1)}-f_{i,j+l}^{(\xi-1)})^2}$ である．重みをこのように分解すると，式 (2) は以下の手順で計算できる．

Step 1) 全ての i,j について以下の a_{ij} と b_{ij} を計算する．

$$a_{ij}=\sum_{l=-p}^{p}v_{ijl}^{(\xi-1)}f_{i,j+l}^{(\xi-1)},\quad b_{ij}=\sum_{l=-p}^{p}v_{ijl}^{(\xi-1)} \quad (3)$$

Step 2) 全ての i,j について

$$s_{ij}=\sum_{k=-p}^{p}u_{ijk}^{(\xi-1)}a_{i+k,j},\quad t_{ij}=\sum_{k=-p}^{p}u_{ijk}^{(\xi-1)}b_{i+k,j} \quad (4)$$

を計算し，$f_{ij}^{(\xi)}=s_{ij}/t_{ij}$ とする．

このようにすると指数関数の計算回数は毎回 1 画素当り $2(2p+1)$ で済む．

3.2 定加重反復 BF の高速化

同様な空間分解近似を行うと 2.2 節の定加重反復 BF は以下の手順で計算できる．

[配列の作成]

ᵗ九州大学大学院芸術工学研究院視覚情報部門

図 1: 入力画像 (左) とイラスト風画像 (右)

Step 1) 全ての i,j,k について $u_{ijk} = e^{-\alpha k^2-\beta(d_{ij}-d_{i+k,j})^2}$ を計算し，また全ての i,j,l について $v_{ijl} = e^{-\alpha l^2-\beta(d_{ij}-d_{i,j+l})^2}$ を計算し，それぞれ配列に保存する．

Step 2) 全ての i,j について $b_{ij} = \sum_{l=-p}^{p} v_{ijl}$ を計算する．

Step 3) $t_{ij} = \sum_{k=-p}^{p} u_{ijk} b_{i+k,j}$ を全ての i,j について計算して配列に保存する．

[フィルタ反復]

Step 4) 全ての i,j について

$$a_{ij} = \sum_{l=-p}^{p} v_{ijl} f_{i,j+l}^{(\xi-1)}, \quad s_{ij} = \sum_{k=-p}^{p} u_{ijk} a_{i+k,j} \qquad (5)$$

を計算し，$f_{ij}^{(\xi)} = s_{ij}/t_{ij}$ とするのを繰り返す．

このようにすると反復 BF からのメモリ使用量の増加は $[2(2p+1)+1]\times$ 画素数となり，また Step 4 の演算量は毎回 1 画素当り $2(2p+1)$ の乗算と $4p$ の加算および 1 回の除算であり，メモリ使用量と演算量の両方とも 2.2 節の定加重反復 BF よりも減る．なお，本提案法は Guarnieri らの手法 [5] とは方式が異なり，演算の単位量が異なるので演算量の直接の比較はできない．

4. 実験

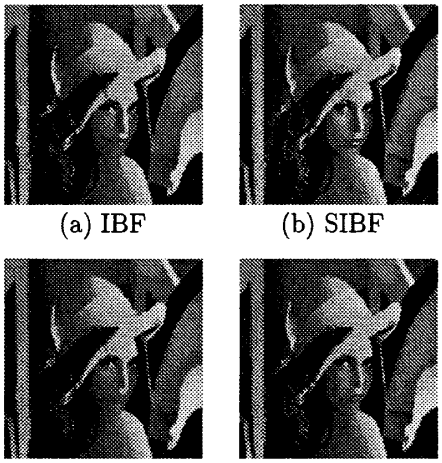
図 1 左の 500×500 の画像にフィルタを 20 回かけた結果を図 2 に示す．$\alpha = 0.001, \beta = 0.01, p = 5$ とした．図 2(a) は反復 BF(Iterated BF:IBF)，(b) は重み分解反復 BF(Separable IBF:SIBF)，(c) は定加重反復 BF(Fixed IBF:FIBF)，(d) は重み分解定加重反復 BF(SFIBF) である．図 2(a) では頬や肩などに階段効果による擬似輪郭が生じているが，図 2(d) ではそれが解消されている．

各反復フィルタの計算時間を図 3 上に示す．IBF に対する SFIBF の高速化率は回数が増すほど大きくなり，20 回めでは約 25 倍速い．なお，図 3 下は入力画像 (図 1 左) との PSNR であり，いずれも平滑化により減少しているが，SFIBF は IBF と近い PSNR 値になっている．

但し，NPR への応用等では定量的な誤差よりも視覚的印象が重要であり，例えばイラスト風画像の生成では平滑化画像から抽出したエッジを重ね描きするので，擬似輪郭の抑制が重要である．高速性に加え，この点でも SFIBF は NPR への応用等に有用であると思われる．図 1 右は，図 2(d) から抽出したエッジを重ね描きして作ったイラスト風画像である．

5. むすび

バイラテラルフィルタを反復してかける場合の高速な方法として重み分解定加重反復 BF を提案し，階段効果による擬似輪郭の発生が抑制されることを検証した．平滑化フィルタの基本用途であるノイズ除去の性能についても検証するのが今後の課題である．



(a) IBF (b) SIBF (c) FIBF (d) SFIBF

図 2: 反復 20 回めの出力画像

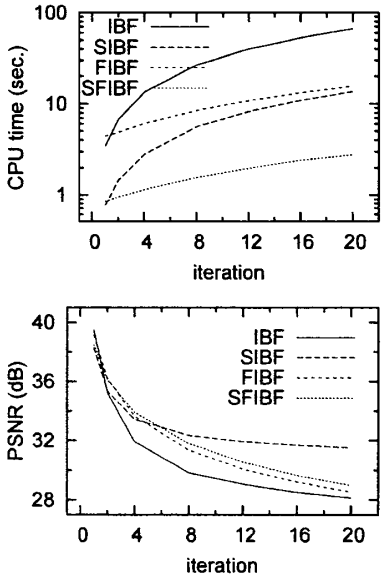


図 3: 計算時間 (上) と PSNR(下)

参考文献

[1] S. Paris, P. Kornprobst, J. Tumblin and F. Durand, "A gentle introduction to bilateral filtering and its applications," SIGGRAPH Course Note, 2007.

[2] 景琳琳, 浦浜喜一, "重みの空間方向分解による非線形フィルタの高速化," 信学論, J89-A, 2, pp.175-178, 2006.

[3] J. Chen, S. Paris and F. Durand, "Real-time edge-aware image processing with the bilateral grid," Proc. SIGGRAPH, 2007.

[4] A. Buades, B. Coll and J. M. Morel, "The staircasing effect in neighborhood filters and its solution," IEEE Trans. Image Process., 15, 6, pp.1499-1505, 2006.

[5] G. Guarnieri, S. Marsi and G. Ramponi, "Fast bilateral filter for edge-preserving smoothing", Elect. Let., 42, pp.396-397, 2006.