Vol.2016-CVIM-200 No.12
2016/1/21

乱択化バイラテラルフィルタによる高速エッジ保持平滑化

藤田 秀[1]　木村 誠[2]　福嶋 慶繁[1]

概要：バイラテラルフィルタは，代表的なエッジ保持平滑化フィルタの一つであり，様々なアプリケーションに応用されている．一方で，計算コストの高さが問題であるが，数々の高速化手法が提案されている．しかし，その高速化手法の近似精度は不十分であり，また効率的に処理可能な次元（色）に制限がある．そこで本稿では，乱択アルゴリズムをバイラテラルフィルタへと適応することで，高精度かつ高効率に実行可能な高速化手法を提案する．この手法では，フィルタ時の参照画素を乱択することで参照する画素数を減らし，処理結果の近似高速化する．この際，単純な乱択ではなく，ベイズの定理から確率的に最適なサンプリングを推定することで，高精度な近似を実現する．加えて，計算効率向上のためにフィルタのSIMD ベクトル化を行う．その際発生するストリーキングノイズを抑えつつ，高精度に並列・ベクトル化を行う手法も同時に提案する．実験では，提案する近似高速化手法が，従来提案されてきた手法よりも高精度かつ高効率に計算が可能であることを示す．

1. はじめに

エッジ保持平滑化フィルタ（Edge-Preserving Filter: EPF）は，画像のノイズ除去 [1] を筆頭に，ハイダイナミックレンジ画像 [2]，超解像 [3]，霞除去 [4] やステレオマッチング [5] といった，画像処理やコンピュータビジョンの様々なアプリケーションに応用可能である．その応用範囲の広さから，様々な特性を持つ EPF が提案されている．

バイラテラルフィルタ（Bilateral Filter: BF）[6] は，代表的な EPF の一つである．BF は，適応的な重みを用いる有限インパルス応答（Finite Impulse Response: FIR）フィルタであり，その重みは注目・参照画素間の距離と画素値の差から決まる．この適応的な処理は，エッジ保持を可能とするが，計算コストは高い．そのため，BF の近似高速化手法の提案が多数されてきた．

グレー画像に対して効率的な手法として文献 [2], [7], [8], [9], [10], [11] があり，カラー画像などの高次元画像に効率的な手法として文献 [12], [13], [14] がある．これらの手法により，カーネルサイズに対して定数時間で処理が可能となる，高速化がなされた．しかし，リアルタイム処理には GPU が必要となるなど，CPU で十分に高速で動作するとは言えず，加えて，必要なメモリの量も膨大である．一方で，セパレイブル実装による高速化 [15], [16] も可能であるが，この実装方法はストリーキングノイズが不可避である．

BF の近似とは異なる，その他の高速な EPF 手法も存在する．例えば，局所線形モデルを用いるガイデッドフィルタ [17] や，測地線距離を元にドメイン変換を行うドメイントランスフォームフィルタ [18] などがある．これらの手法は非常に高速である一方で，いくつかの問題点もある．ガイデッドフィルタは局所線形モデルを仮定しているが，大きなカーネルサイズを用いて処理する場合，モデルが破綻しやすく，ノイズが発生する．ドメイントランスフォームフィルタは，セパレイブル実装のため，ストリーキングノイズが不可避となる．

一般的に，任意のアルゴリズムを効率的に計算を行う上で，乱択化は有効な手法の一つである．中でも，モンテカルロ法による乱択化は，乱択の試行回数により精度と計算速度のトレードオフを扱うことができる [19], [20]．この乱択化を利用することで高速化を図ったノイズ除去手法も存在する [21], [22]．このモンテカルロ法を BF に適用し，参照画素を乱択することでカーネルを近似した手法として，文献 [23] がある．図 1 に，この乱択化フィルタの概要を示す．文献 [23] では，最も基本的な方法で乱択を行ったとしても，従来の近似高速化手法の問題を解決しつつ，高速化が可能であることを示している．

本稿では，この乱択化による近似高速化手法を確率的に検証し，より BF に適した形へと展開することで，より高精度かつ頑健に BF を高速化した，乱択化バイラテラルフィルタを提案する．また，近年の CPU は，SIMD 命令によりベクトル化を行うと，処理の効率化が可能である．このベクトル化は乱択化バイラテラルフィルタにも適用可能で

[1] 名古屋工業大学
Nagoya Institute of Technology, Japan
[2] サムスン日本研究所
Samsung R&D Institute Japan

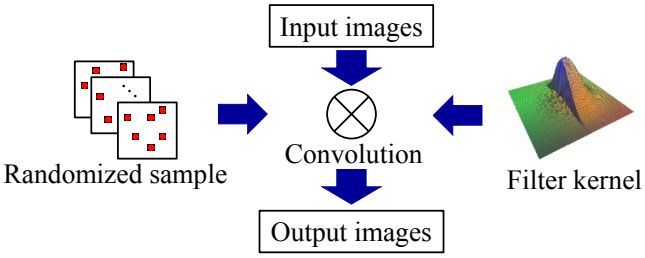


図 1: 乱択化バイラテラルフィルタの概要.

あるが，乱択した参照画素の分布の都合により，ストリーキングノイズが発生する問題がある．本稿では，この問題を抑えることができるベクトル化手法を検討することで，更なる処理の効率化を図る．

2. 乱択化バイラテラルフィルタ

2.1 定義

本節では，乱択化バイラテラルフィルタ（Randomized BF: RBF）を定義する．まず，ある入力画像 $\boldsymbol{I}$ に対する基本的な BF [6] の処理は以下の式で表現できる．

$$\bar{\boldsymbol{I}}(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} f_{bf}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{I}(\boldsymbol{q})}{\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} f_{bf}(\boldsymbol{p},\boldsymbol{q})} \tag{1}$$

ここで，$\bar{\boldsymbol{I}}$ は BF の出力画像，$\boldsymbol{p},\boldsymbol{q}$ はそれぞれ注目・参照画素，$\mathcal{N}(\boldsymbol{p})$ は注目画素 $\boldsymbol{p}$ の近傍画素集合である．f_{bf} は BF のフィルタ重みを表し，以下の式で求められる．

$$f_{bf}(\boldsymbol{p},\boldsymbol{q}) = \exp\left(\frac{-(||\boldsymbol{p}-\boldsymbol{q}||_2^2}{2\sigma_s^2}\right)\exp\left(\frac{-||\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}||_2^2}{2\sigma_r^2}\right) \tag{2}$$

ここで，σ_s と σ_r はそれぞれ，空間平滑化とエッジ保持の度合いを決めるパラメータである．

基本的な BF は，注目画素 $\boldsymbol{p}$ の近傍画素を全て参照する．一方で，自然画像の局所領域は，似た画素値を持つ画素から構成されていることが多く，冗長性を含んでいる（詳細については 2.2 節で示す）．したがって，自然画像に対してフィルタリングを行う際，全ての参照画素を参照する必要はなく，参照画素の数は間引くことができると考えられる．しかし，規則的に参照画素を間引いた場合，エイリアシングが発生する恐れがある．これを防ぐために RBF では，乱択による参照画素の間引きを行い，以下のように計算を行う．

$$\bar{\boldsymbol{I}}(\boldsymbol{p}) \simeq \bar{\boldsymbol{I}}'(\boldsymbol{p}) = \frac{\sum_{j=1}^{n} f_{bf}(\boldsymbol{p},R_j(\boldsymbol{p}))\boldsymbol{I}(R_j(\boldsymbol{p}))}{\sum_{j=1}^{n} f_{bf}(\boldsymbol{p},R_j(\boldsymbol{p}))} \tag{3}$$

ここで，$\bar{\boldsymbol{I}}'$ は RBF の出力画像，R は参照画素を $\mathcal{N}(\boldsymbol{p})$ から乱択する関数，n は参照する画素の数を表す．この際，式（1）で参照する画素の数を N で表すと，$n << N$ となるように n を設定する．これにより，RBF は通常の BF に比べて，高速に動作することができる．

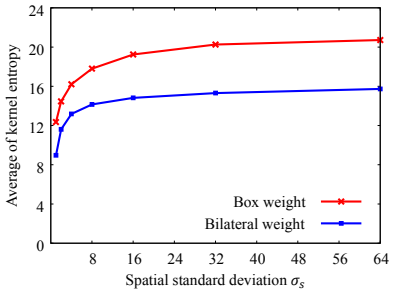


図 2: カーネルエントロピーの平均値．なお，カーネル半径 r は $3\sigma_s$ としている．

2.2 カーネルエントロピー

本節では，前述のフィルタカーネルの間引きが許容できることを，カーネルのエントロピーを定義することで示す．BF に限らず一般的な FIR フィルタは，各画素 $\boldsymbol{p}$ における，局所ヒストグラム h を使用することで表現できる．例えば，BF の処理は，BF の局所ヒストグラム h_{bf} を用いると，以下のように書き表すことができる．

$$h_{bf}(\boldsymbol{p},i) = \sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} f_{bf}(\boldsymbol{p},\boldsymbol{q})\delta(i-\boldsymbol{I}(\boldsymbol{q})) \tag{4}$$

$$\bar{\boldsymbol{I}}(\boldsymbol{p}) = \frac{\sum_i h(\boldsymbol{p},i)i}{\sum_i h(\boldsymbol{p},i)} \tag{5}$$

ここで，δ はクロネッカーのデルタ関数を表し，i は局所ヒストグラムの i 番目のビンである．そして，この局所ヒストグラムは，そのカーネル内の領域におけるエントロピーをそのまま表現している．このエントロピーを本稿では，カーネルエントロピーと呼び，以下の式で定義する．

$$H(\boldsymbol{p}) = -\sum_i P(h(\boldsymbol{p},i))\log_2 P(h(\boldsymbol{p},i)) \tag{6}$$

ここで，$P(h(\boldsymbol{p},i))$ はカーネル内の領域の画素数に対する i 番目のビンの数の割合である．もし，カーネル内の領域が様々な画素値で構成されていれば，このカーネルエントロピーは増加する．逆に，カーネル内の領域がほとんど似た画素値で構成されていれば，このカーネルエントロピーは減少する．つまり，カーネルエントロピーが小さければ，この局所ヒストグラムは圧縮できると言える．

また，カーネルエントロピーは局所領域の画素値の構成だけでなく，フィルタ重みにも依存する．例えば，注目画素からの距離と輝度差に応じて重みが与えられる BF と，一様に同じ重みを用いるボックスフィルタとでは，カーネルエントロピーは異なる．実際に図 2 に，自然画像「Lenna」に使用して，BF とボックスフィルタのカーネルエントロピーの平均値を計算したときの結果を示す．ただし，ボックスフィルタの重み f_{box} は以下のように計算される．

$$f_{box}(\boldsymbol{p},\boldsymbol{q}) = 1 \tag{7}$$

図 2 に示すように，カーネルエントロピーは常に 24[*1] よ

[*1] ランダム画像のとき，カーネルエントロピーは 24 となる．

りも小さい．加えて，BF のフィルタ重みを使用すると，さらにエントロピーは低くなる．このことから，BF のカーネルは偏りがあり，冗長であるということがわかる．したがって，局所ヒストグラムはカーネルの間引きにより近似可能と言える．

しかしながら，文献 [23] でも示されている通り，規則的にカーネルを間引くと，エイリアシングが発生する．次節では，適切に近似を行うためには，どのように参照画素を間引けばよいかについてを検討する．

2.3 乱択手法

参照画素の乱択を行う際，単純な乱択を行うと，サンプリングした画素の分布に偏りが生じることがある．これを避けるために，文献 [23] では乱択手法として，ポアソンディスクサンプリング（Poisson-Disk Sampling: PDS）[24] を適用した．これにより，文献 [23] はよい近似を得ていたが，これは実験的によいとされたものである．本稿では，RBF に適した乱択手法について，理論的に検討する．

2.3.1 最適乱択パタン

一般的には，各領域で最も良い近似を得るための乱択パタンは唯一ではない．このことから，最適な乱択パタンは，確率密度分布として表現ができると考えられ，これはベイズの定理により導くことができる．ここで，θ をある分布に従って生成される乱択パタンとし，ある領域を最適な乱択パタンを用いてフィルタした結果を観測データ D とする．このとき，θ の事後確率 $P(\theta|D)$ は，ベイズの定理から以下の式で表すことができる．

$$P(\theta|D) = \frac{P(D|\theta)\pi(\theta)}{P(D)} \propto P(D|\theta)\pi(\theta) \tag{8}$$

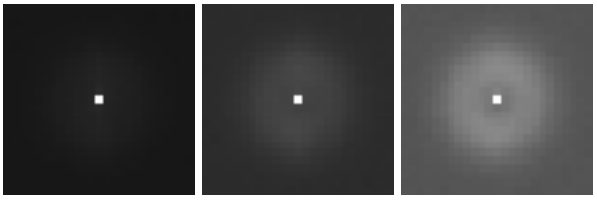
ここで，$P(D|\theta)$ は D の尤度，$\pi(\theta)$ は θ の事前確率を表す．また，$P(D)$ については定数と考えられるため，無視することができる．事前確率 $\pi(\theta)$ は，事前に考えられたある分布 [*2] に従うため，式 (8) において支配的な要素は，尤度 $P(D|\theta)$ である．しかし，厳密な尤度の算出は，フィルタカーネルのモデルが複雑であることから，高い計算コストが必要となる．

尤度計算が困難な問題に対する，効率的な計算方法としては，近似ベイズ計算（Approximate Bayesian Computation: ABC）[25] がある．ABC はまず，$\pi(\theta)$ から最適なパタンの候補となる θ' を生成し，それを用いてシミュレーションデータ D' を作成する．そして，D と D' 間の距離が正の定数 ϵ 以内であれば，D' を許容するというものである．つまり，事後確率 $P(\theta|D)$ を以下のように近似する．

$$P(\theta|D) \simeq \hat{P}(\theta|D) = P(\theta|\rho(D, D') \leq \epsilon) \tag{9}$$

ここで，$\hat{P}(\theta|D)$ は近似事後確率，ρ はデータ間の距離評価関数であり，本稿では平均二乗誤差（Mean Square Error: MSE）を使用する．なお，観測データ D には，乱択を行わないフィルタの結果を用いることで，最適な乱択フィルタの結果として用いる．

[*2] 本稿では簡単のため，一様分布に従うものとする．



(a) $\sigma_r = 8$　(b) $\sigma_r = 16$　(c) $\sigma_r = 32$

図 3: バイラテラルフィルタの最適な確率分布．

この ABC の実際の計算には，棄却サンプリング [25] を用いる．この場合，許容された回数が K になるまで以下の三つの処理を繰り返し，近似事後確率を得る．

(1) 候補となる θ' を事前確率 $\pi(\theta)$ から生成する．
(2) θ' から D' をシミュレーションする．
(3) $\rho(D, D') \leq \epsilon$ ならば許容し，θ' を記録する．

したがって，近似事後確率は，以下の式で求められる．

$$\hat{P}(\theta|D) = \frac{1}{nK}\sum_{k=1}^{K} C(\theta'_k) \quad (\rho(D, D'(\theta'_k)) \leq \epsilon) \tag{10}$$

ここで，θ'_k は k 番目の候補パタンであり，$D'(\theta'_k)$ は θ'_k から生成されるシミュレーションデータである．$C(\theta'_k)$ は θ'_k の構成要素に 1 を加えていく関数である．

理想としては，あらゆる領域のフィルタカーネルに対する事後確率を導出することだが，それが現実的ではない．そのため本稿では，様々なフィルタカーネルから，一般的な事後確率を求める．この際，各フィルタカーネルに対して t 個の θ' と D' を生成し，その中で $\rho(D, D')$ が最小となる θ' のみを許容する．これは特定のフィルタカーネルによって，事後確率が偏ることを防ぐためである．つまり，式 (10) において許容される θ'_k の条件が，以下の式を満たすものとする．

$$\theta'_k = \arg\min_{\theta'_s}\{\rho(D_k, D'_k(\theta'_s))\} \tag{11}$$

ここで，D_k は k 番目のフィルタカーネル，$D'_k(\theta'_s)$ は θ'_s から生成されるシミュレーションデータである．また，s は $1 \leq s \leq t$ を満たす整数である．

図 3 は，BF の最適な乱択パタンの確率分布を推定したときの結果である．なおこの推定では，様々な自然画像から生成される 5×5 のパッチに対してフィルタを行い，パラメータとして，$\sigma_s = 4$，$n = 30$，$t = 100$，$K = 1048576$ を与えた．この確率分布にしたがって行った乱択結果を本稿では，OS（Optimal Sampling）と呼ぶ．乱択手法で OS を用いると，単純な乱択や PDS を行った場合に比べ，高い近似精度が得られる．この結果については，第 4 節で示す．

2.3.2 近似最適乱択

前節では，RBF に適した乱択パタンの確率分布を推定した．しかし，図 3 に示すように，最適な乱択パタンは平滑化パラメータ毎に異なる．これはパラメータ毎に傾向の異なる乱択を行わなければならず，高い計算コストが必要となる．本節では，この最適な乱択パタンの確率分布から，近似最適乱択手法を検討する．

図 3 の確率分布から，注目画素はその近傍画素に比べて，重要度が高い．また，その傾向はエッジ保持効果を高めるほど，つまり σ_r が小さくなるほど顕著である．一方で，注目画素の近傍画素では大きな差異はない．したがって，この確率分布にしたがう乱択は，注目画素と一様乱択との組み合わせにより近似できると考えられる．この際，一様乱択の手法は，文献 [23] でも議論されているように，単純な乱択よりも PDS の方が高い効果を持つ．つまり，近似最適な乱択手法は，PDS に注目画素を加えたもので実現可能である．この近似的な最適乱択手法を，本稿では AOS（Approximated Optimal Sampling）と呼ぶ．

3. 効率的な実装

3.1 乱択結果のルックアップテーブル化

フィルタリングにおいて，どの参照画素を使用するかの選択方法は，二種類存在する．一つは動的に乱択を行う方法，もう一つはルックアップテーブル（Look-Up Table: LUT）を用いる方法である．動的に乱択を行う方法は，各注目画素で乱択を行うため，素朴で直感的な方法である．しかし，OS や PDS のような，サンプリングパタンが複雑な乱択手法を用いると，計算コストが高くなる．一方，LUT を用いる方法の場合，乱択パタンを事前に計算して LUT に格納するため，乱択の計算コストを省くことができる．従来手法 [23] では，処理する参照画素を表す二値マスクを，Crnley-Patterson 法 [26], [27] を用いることでサイズを縮小し，LUT に格納していた．しかし，それでもメモリの使用量を十分削減できているとはいえず，フィルタサイズが大きくなるほど，メモリの使用量は増加する．その場合，乱択する画素数 n は大きくは変わらないため，LUT は冗長となる．

そのため本稿では，カーネル内で乱択された座標のみを格納した，メモリ使用量が n にのみ依存する LUT を導入する．この LUT では，キャッシュ効率を向上させるために，ラスタ順に座標を格納する．なお，乱択パタンを増やすために，LUT は m 個生成される．この LUT の概要を図 4 に示す．したがって LUT を用いると，RBF は以下の式のように処理を行う．

$$O'(\boldsymbol{p}) \simeq O'_s(\boldsymbol{p}) = \frac{\sum_{j=1}^{n} f(\boldsymbol{p}, L_u(j)) I(L_u(j))}{\sum_{j=1}^{n} f(\boldsymbol{p}, L_u(j))} \qquad (12)$$

ここで，L_u は u 番目の LUT を表し，u ($\in \{0, \cdots, m-1\}$) は各注目画素 $\boldsymbol{p}$ でランダムに選択される．また，$L_u(j)$ は L_u 中の j 番目に格納されている参照画素位置を表す．

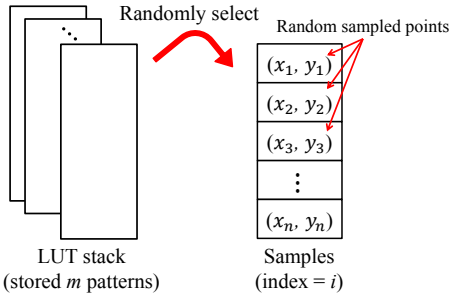


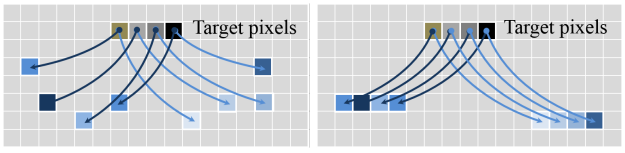
図 4: LUT の概要．

LUT を用いる実装は，計算コストは大幅に削減される．一方で，LUT 方式の実装は，ランダム性の再現性と，LUT ためのメモリ領域の二点で課題がある．ランダム性を補うためには，式 (12) では m 個の LUT を用意し，各注目画素で LUT をランダムに選択する．次にメモリの課題については，まず提案した LUT の必要メモリ量を解析する．一つの点について考えたとき，座標を格納するために必要なメモリ量は，たかだか 4 バイト（short 型の領域 $\times 2$）である．このことから，一つの LUT に必要なメモリ量は，$4n$ バイトである．したがって，最終的には m 個の LUT 分を使用するため，必要なメモリ量は $4nm$ バイトである．n と m の数によっては必要なメモリ量は大きくなるが，RBF に必要な参照画素数 n と LUT の数 m は少ない（詳細第 4 節にて示す）．そのため，実質的に必要なメモリ量は大きくはない．

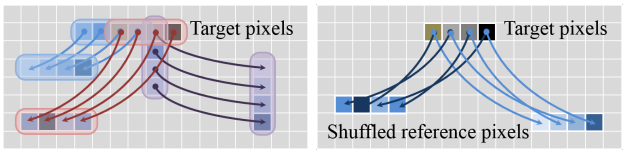
3.2 乱択化バイラテラルフィルタのベクトル化

近年のほとんどの CPU は，SIMD などのベクトル化処理機構を備えている．SIMD 命令は従来の命令セットに比べて，連続したデータに対して高速な処理が可能である．しかし，ランダムにカーネルを間引く乱択化と，連続にデータを処理するベクトル化の発想は，互いに相反したものである．

図 5 (a) は，参照画素に対してランダムアクセス（RA: random access）している様子を，視覚化したものである．なお，この図では各注目画素につき，二点が乱択されている．この RA を行う上で，最も大きなオーバーヘッドは，キャッシュミスである．このキャッシュ効率の問題に対する率直な解決策が，連続したメモリへのアクセスを行う，ベクトル化である．このメモリアクセスの方法を本稿では，SA（sequential access）と呼ぶ．図 5 (b) に示すように，SA のメモリアクセスはキャッシュ効率が向上するように，常にベクトルの長さの分だけ連続している．しかしこれは，注目画素から参照画素への相対的な位置が，ベクトルの長さの間隔で，固定されることを意味する．つまり，ベクトルの長さの間隔で，使用される LUT が共有されな



(a) Random access (RA) (b) Sequential access (SA)



(c) Overlapped SA (OSA) (d) Shuffled SA (SSA)

図 5: ベクトル化処理の概要.

ければならない．この場合，RBF に対して SIMD を適用可能だが，ベクトルの長さの間隔で画素値が類似しやすくなるため，ストリーキングノイズが発生しやすいという欠点が存在する．

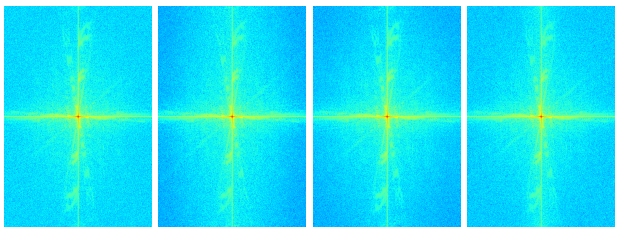
この問題を解決するために，本稿では OSA（Overlapped Sequential Access）を提案する．OSA で重要な点は，通常の SA で連続して使用する画素群に対し，位相のずれた画素群を重複させることである．図 5 (c) は二つの OSA の例を表している．一つ目の例は，図中の赤と青の画素群で示されており，水平方向にずれた画素群の重複領域を平均化することで，ストリーキングノイズの影響を抑えている．もう一つは，紫の画素群で示されており，転置した画像に対して SA を行う．そして通常の結果と転置したときの結果とを平均化することで，最終的な結果を得る．この転置を行う処理は，水平方向のストリーキングノイズの特徴に対し，特に有効である．

また，OSA に加えて，SSA（Shuffled Sequential Access）という方法も提案する．これは図 5 (d) に示すように，ベクトル化した参照画素群を，実行時にシャッフルするという方法である．この SSA により，LUT を共有した画素群における，乱択パタンの統一性を抑えることができる．SSA は，たった一つの SIMD 命令（*mm_shuffle_ps*）で実現することできる．したがって，SSA を適用することで発生する計算コストは，極めて少ないといえる．

図 6 は，RA，SA，OSA，SSA を用いて RBF を行ったときの周波数画像を示している．図 5 (a) の RA を使用時と比べて，図 5 (b) は水平方向の高周波領域の成分が減衰している．これは SA により，水平方向のテクスチャの同一化によるストリーキングノイズが発生していることを示している．一方で，OSA や SSA を使用して処理を行った場合，高周波領域における減衰は改善されている．

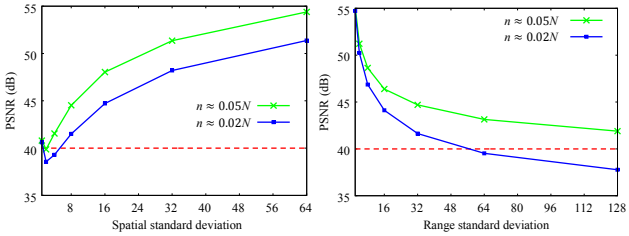
4. 実験結果

本節では，RBF の近似精度，処理結果の安定度，計算効率について，他の BF の高速化手法と比較を行いつつ，性能を検証する．その後，RBF の近似精度と計算効率のトレードオフの性能について，検証を行う．



(a) RA (b) SA (c) OSA (d) SSA

図 6: 各ベクトル化手法使用時の RBF 後の周波数画像.



(a) $\sigma_r = 32$ (b) $\sigma_s = 8$

図 7: RBF の近似精度

近似精度

RBF の近似精度についての検証を行う．なお，近似精度の客観評価には，基本的な BF の処理結果を正解画像とした PSNR を用いて行う．使用した入力画像は，Kodak 社により提供されているカラー画像 24 枚を使用し，その平均値を結果としている．なお，フィルタカーネルの半径 r は，$3\sigma_s$ とした．

図 7 は，一定数乱択したときの結果を表している．一般的には，PSNR 値が 40 dB あれば，視覚的な差異がほぼわからないということから，RBF の近似精度は十分に高いと言える．また，σ_s が大きいとき，つまりフィルタサイズが大きいときほど，近似精度が向上する．この結果では，全体の参照画素数 N に対する，使用している参照画素数 n の割合は固定であるため，大きいフィルタサイズでは n の割合は減らせると言える．一方で，σ_r が大きいとき，つまりエッジ保持効果が小さいときは，近似精度が低下する．しかし σ_r が大きいときの処理結果は，ガウシアンフィルタの結果に近くなり，その場合はより効率的なガウシアンフィルタの近似 [28] でよい．そのため，大きい σ_r を使用するのであれば，RBF を適用する必要はない．

図 8 は，AOS と PDS を用いて RBF を処理したときの近似精度を示す．なお，PDS は従来手法 [23] で適用された手法である．そのため，図から，従来手法と比べて RBF の近似精度が向上したと言える．

また，RBF の処理結果を視覚的に評価するために，図 9 に RBF の処理結果を示す．この図から，RBF の処理結果は基本的な BF の処理結果と比較して，非常に類似してい

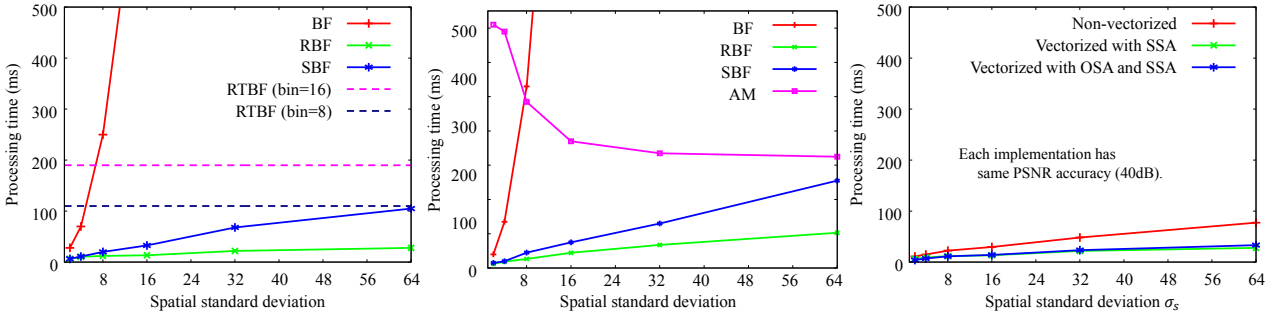
図 9: バイラテラルフィルタリングの結果．図 (a)，(d)，(g) は入力画像，図 (b)，(e)，(h) は従来のバイラテラルフィルタ [6] の結果，図 (c)，(f)，(i) は乱択化バイラテラルフィルタの結果を表す．図 (c)，(f)，(i) の近似精度はそれぞれ，43.86 dB，41.69 dB，44.68 dB である．なお，使用した平滑化パラメータは $\sigma_s = 8$，$\sigma_r = 48$，カーネル半径は $3\sigma_s$，サンプリング数は約 $0.03N$（$N = (2 * r + 1) \times (2 * r + 1) = 2401$）である．

ることがわかる．なお，この RBF に使用した参照画素数 n は，全体の参照画素数 N に対して 3%程度である．したがって，フィルタリングの際に，大幅に参照画素数を間引いたとしても，画質に大きな影響がない．

処理結果の安定度

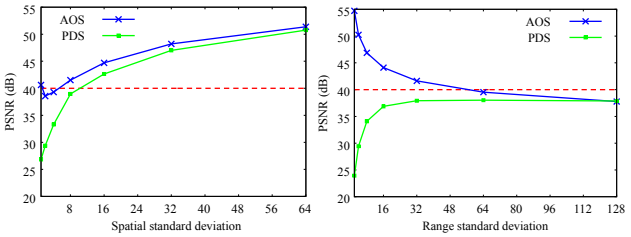
フィルタリングを乱択化する上では，処理結果の安定度は重要な課題である．この安定度については，処理結果の PSNR に対する 95%信頼区間を用いて評価する．なお，この実験で行った試行回数は 1000 回である．

図 10 は，LUT の有無，そして乱択手法の違いで安定度を評価したときの結果である．この結果から，ほとんどの場合で，PSNR の違いが 0.002 dB 以内に収まっているため，処理結果は安定していると言える．また，LUT を用いると安定度は低下するが，この差は非常に小さい．そのため，LUT による高速化は出力結果の画質に対して，大きな影響はない．同時に，AOS の処理結果は，PDS の処理結果と比べて安定していることも確認できる．



(a) グレー画像での結果　(b) カラー画像での結果　(c) ベクトル化手法による違い

図 11: 様々な高速化手法の処理時間



(a) $\sigma_r = 32$　(b) $\sigma_s = 8$

図 8: AOS と PDS の違いによる RBF の近似精度の差．使用した参照画素数 n は約 $0.02N$ である．

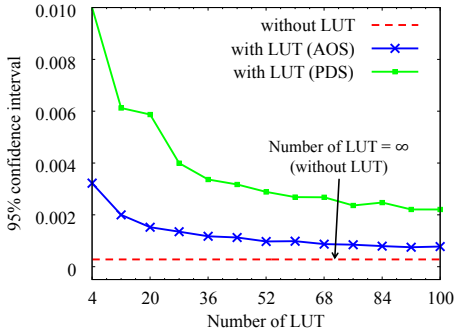


図 10: 処理結果の PSNR の 95%信頼区間

計算効率

計算効率の評価では，RBF と，様々な BF の高速化手法と比較する．比較手法としては，セパレイブル BF（SBF: separable BF）[15]，リアルタイム $O(1)$BF（RTBF: real-time $O(1)$ BF）[9], [29]，AM（adaptive manifolds）[14] を用いる．RTBF はグレー画像では最も高速な定数時間アルゴリズムであり，AM は高次元を処理するときに高速な手法である．SBF は古くから存在する高速化手法であるが，近年においても十分に高速な手法である．

RBF に用いる乱択手法は，AOS を使用する．使用する参照画素数には近似精度が 40 dB を超える最小数を使用し，LUT の数は 20 とする．処理する画像は，解像度 1024×1024 である画像を用いる．なおこの実験では，提案手法や比較手法は，Visual Studio 2012 における OpenMP や Intel TBB，SIMD 命令を用いてベクトル化，並列化されており，C++により実装されている．実験では，CPU として Intel Core i7-3770K 3.50GHz を用いている．

比較結果を図 11 に示す．結果から，RBF はほとんど全ての場合において，他の高速化手法よりも高速である．なお，カラー画像における RTBF の結果は，7000 ms をこえるため，図からは省略した．参考として，BF の近似ではない，他の高速なエッジ保持平滑化フィルタ手法である，ガイデッドフィルタ（GF: guided filter）[17] と，ドメイントランスフォームフィルタ（DTF: domain transform filter）[18] との比較を，カラー画像で行った．ただし，DTF のインパルス応答には正規化畳み込み（normalized convolution）を使用した．このとき，GF と DTF の処理時間はそれぞれ，293 ms と 175 ms であった．したがって，RBF は他の高速な手法と比較を行っても，高速であることがわかる．

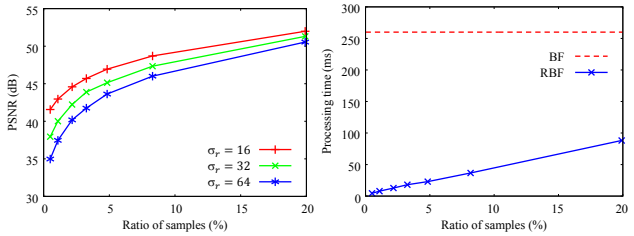
また，RBF のベクトル化手法の提案を行ったが，この処理時間の増加は少ない．図 11c はグレー画像の処理で，ベクトル化の有無，ベクトル化手法の違いによる処理時間を示している．ただし，SA と SSA の処理時間の差はほとんどない．このことから，OSA や SSA によるベクトル化は，画質に影響を与えずに RBF を高速化できることがわかる．

近似精度と計算効率とのトレードオフ

乱択化による高速化では，フィルタリングの計算効率と近似精度の制御を，使用する参照画素数 n で簡単に制御することができる．図 12 は実際に，近似精度と計算効率とのトレードオフ関係を示している．乱択化による高速化は，n を変更することで，連続して近似精度と計算効率を制御できる．インタラクティブな処理が求められる写真編集などでは，こうしたトレードオフの操作性は重要視され，RBF は有効に作用する．

5. まとめ

本稿では，より効率的にバイラテラルフィルタを処理するために，乱択化による近似高速化手法を提案した．提案手法は，フィルタカーネルをランダムに間引くことで高速化を図り，その際，近似最適な乱択手法を導入する．また，



(a) PSNR accuracy (b) Processing time

図 12: 近似精度と計算効率とのトレードオフ

より効率的な処理のために，新たなベクトル化手法や LUT による，実装の側面からも高速化を行った．その結果，提案した乱択化バイラテラルフィルタは，高精度にバイラテラルフィルタを近似し，近年提案されている高速化手法よりも高速であることを示した．また，提案した乱択化手法は，バイラテラルフィルタに限らず，様々なエッジ保持平滑化フィルタへの運用が可能である．そのため，様々なエッジ保持平滑化フィルタの高速化への展開が期待できる．

参考文献

[1] Buades, A., Coll, B. and Morel, J. M.: A Non-Local Algorithm for Image Denoising, *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 60–65 (2005).

[2] Durand, F. and Dorsey, J.: Fast Bilateral Filtering for the Display of High-Dynamic-Range Images, *ACM Trans. on Graphics*, Vol. 21, No. 3, pp. 257–266 (2002).

[3] Kopf, J., Cohen, M., Lischinski, D. and Uyttendaele, M.: Joint Bilateral Upsampling, *ACM Trans. on Graphics*, Vol. 26, No. 3 (2007).

[4] He, K., Sun, J. and Tang, X.: Single Image Haze Removal Using Dark Channel Prior, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 12, pp. 2341–2353 (2011).

[5] Scharstein, D. and Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms, *International Journal of Computer Vision*, Vol. 47, No. 1, pp. 7–42 (2002).

[6] Tomasi, C. and Manduchi, R.: Bilateral Filtering for Gray and Color Images, *Proc. IEEE International Conference on Computer Vision (ICCV)*, pp. 839–846 (1998).

[7] Porikli, F.: Constant Time O(1) Bilateral Filtering, *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8 (2008).

[8] Paris, S. and Durand, F.: A Fast Approximation of the Bilateral Filter using A Signal Processing Approach, *International Journal of Computer Vision*, Vol. 81, No. 1, pp. 24–52 (2009).

[9] Yang, Q., Tan, K. H. and Ahuja, N.: Real-Time O(1) Bilateral Filtering, *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 557–564 (2009).

[10] Yoshizawa, S., Belyaev, A. and Yokota, H.: Fast Gauss Bilateral Filtering, *Computer Graphics Forum*, Vol. 29, No. 1, pp. 60–74 (2010).

[11] Chaudhury, K., Sage, D. and Unser, M.: Fast O(1) Bilateral Filtering Using Trigonometric Range Kernels, *IEEE Trans. on Image Processing*, Vol. 20, No. 12, pp. 3376–3382 (2011).

[12] Adams, A., Gelfand, N., Dolson, J. and Levoy, M.: Gaussian KD-Trees for Fast High-Dimensional Filtering, *ACM Trans. on Graphics*, Vol. 28, No. 3 (2009).

[13] Adams, A., Baek, J. and Davis, M. A.: Fast High-Dimensional Filtering Using the Permutohedral Lattice, *Computer Graphics Forum*, Vol. 29, No. 2, pp. 753–762 (2010).

[14] Gastal, E. S. L. and Oliveira, M. M.: Adaptive Manifolds for Real-Time High-Dimensional Filtering, *ACM Trans. on Graphics*, Vol. 31, No. 4 (2012).

[15] Pham, T. Q. and Vliet, L. J. V.: Separable Bilateral Filtering for Fast Video Preprocessing, *Proc. IEEE International Conference on Multimedia and Expo (ICME)* (2005).

[16] Fukushima, N., Fujita, S. and Ishibashi, Y.: Switching Dual Kernels for Separable Edge-Preserving Filtering, *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015).

[17] He, K., Shun, J. and Tang, X.: Guided Image Ffiltering, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 6, pp. 1397–1409 (2013).

[18] Gastal, E. S. L. and Oliveira, M. M.: Domain Transform for Edge-Aware Image and Video Processing, *ACM Trans. on Graphics*, Vol. 30, No. 4 (2011).

[19] Motwani, R. and Raghavan, P.: *Randomized Algorithms.*, Cambridge University Press, New York, NY, USA (1995).

[20] Mitzenmacher, M. and Upfal, E.: *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.*, Cambridge University Press, New York, NY, USA (2005).

[21] Chan, S. H., Zickler, T. and Lu, Y. M.: Monte Carlo Non-Local Means: Random Sampling for Large-Scale Image Filtering, *IEEE Trans. on Image Processing*, Vol. 23, No. 8, pp. 3711–3725 (2014).

[22] Fujita, S., Fukushima, N., Kimura, M. and Ishibashi, Y.: Randomized Redundant DCT: Efficient Denoising by Using Random Subsampling of DCT Patches, *Proc. ACM SIGGRAPH Asia Technical Briefs* (2015).

[23] Banterle, F., Corsini, M., Cignoni, P. and Scopigno, R.: A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain, *Computer Graphics Forum*, Vol. 31, No. 1, pp. 19–32 (2012).

[24] Cook, R. L.: Stochastic Sampling in Computer Graphics, *ACM Trans. on Graphics*, Vol. 5, No. 1, pp. 51–72 (1986).

[25] Tavare, S., Balding, D. J., Griffiths, R. C. and Donnelly, P.: Inferring Coalescence Times from DNA Sequence Data., *Genetics*, Vol. 145, No. 5, pp. 505–518 (1997).

[26] Kollig, T. and Keller, A.: Efficient Multidimensional Sampling, *Computer Graphics Forum*, Vol. 21, No. 3 (2008).

[27] Schlömer, T., Heck, D. and Deussen, O.: Farthest-point Optimized Point Sets with Maximized Minimum Distance, *Proc. ACM SIGGRAPH Symposium on High Performance Graphics*, pp. 135–142 (2011).

[28] Sugimoto, K. and Kamata, S.: Fast Gaussian Filter with Second-Order Shift Property of DCT-5, *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 514–518 (2013).

[29] Yang, Q., Ahuja, N. and Tan, K. H.: Constant Time Median and Bilateral Filtering, *International Journal of Computer Vision* (2014).