画像修復におけるFFT高速ブロックマッチングの応用

Image Inpainting using FFT based Fast Block Matching Algorithm

李禎† 貴家仁志‡
†首都大学東京大学院システムデザイン研究科情報通信システム学域

Zhen LI† Hitoshi KIYA†
†Dept. of Information and Communications Systems Engineering, Tokyo Metropolitan Univesity

アブストラクト 画像修復とは，一部分を削除されたデジタル画像を自動的に復元する技術である．より違和感のない画像を生成するため, 様々な手法が研究されていた. 本稿では, その中の一つテクスチャ合成に基づく手法に注目する. この手法ではある最類似パッチの探索を膨大に繰り返す必要があり, 演算量の削減が問題である. そこで,FFT高速ブロックマッチング [13] を適用することを提案する．FFT 高速ブロックマッチングの特徴と最類似パッチ探索の要求との相性がよいため，画像修復の処理速度が飛躍的に向上することができた. シミュレーションでは提案法の有効性を確認した.

1 はじめに

デジタル画像の一部分が，プライバシーや傷，汚れや編集上の考量などにより削除されることがある．その削除される部分（本稿では消失領域と呼ぶ）を自動的に復元させる手法は画像修復や消失領域復元などと呼ばれるデジタル画像処理の中で一つ重要な技術である. より違和感のない画像を生成するため, 様々な手法が研究されていた.

たとえば，消失領域とその周辺領域の輝度値に注目し，輝度値の連続性を保つことを優先とする手法がある [1]–[4]．これらの手法は消失領域が小さい場合うまく復元できるが，消失領域が大きくなると，ぼやけた復元結果になってしまう.

またフーリエ空間やウェーブレット空間などの特徴量を使った修復手法がある [5], [6]．これらの手法では，原画像が周期的なパターンをもつことが前提とする．周期的なパターンをあまり持たない自然画像には向かない.

さらに，消失領域以外の部分から，消失領域内の画素を含むブロックとの最類似パッチを探索し，テクスチャの合成を行う手法がある [7]–[12]．これらの手法は実際画像に存在するものを基に修復するので，より自然な復元結果が得られる．不適切な合成結果を抑えるため，各手法では探索時に用いる尺度関数や修復時に用いる補間関数など多く提案されていた．しかし，細かい探索が膨大に繰り返され，演算量の削減が問題である.

本稿では, テクスチャ合成に基づく画像修復法の演算量の削減に注目する. 文献 [11] の手法を例にする.[11] の最類似パッチの探索では，一つ共有な探索領域で大量なブロックマッチングを行われている. そこで，その探索に FFT 高速ブロックマッチング [13] を適用することを提案する.

FFT 高速ブロックマッチング [13] の特徴として三つあげられる．一つ目は誤差尺度がテクスチャ合成法の探索尺度と同じく SSD(Sum of squared difference) であり，探索の精度はフルサーチと同じになることを保証する．二つ目は二つのブロックが探索領域を共有している場合，二つのブロックの探索は一回の演算で済む．三つ目はブロック数と関係なく，探索領域が一つしかない場合，その部分の FFT 演算は一回でよい．さらに共有するブロック数が多いほど，[13] はフルサーチや他の FFT 高速ブロックマッチング法に比べ演算量の削減がより有効である.

シミュレーションでは，他の FFT 高速ブロックマッチング [14] を利用する手法に比べ文献 [13] を利用する提案法の処理時間は約 1.5 倍速いである．さらに提案法がフルサーチに基づく従来法に比べ，画像修復の処理時間を約 101～148 倍の高速化ができた.

2 準備

ここでは，文献 [11] のテクスチャ合成に基づく画像修復手法と画像修復のためのブロックマッチングを紹介する．文献 [11] の手法は画像だけでなく動画像の修復にも適応できるが，本稿では，画像の修復手法のみ注目する.

2.1 テクスチャ合成による画像修復手法

図 1 に示すように，文献 [11] の手法は主に三つのステップに分けられる．まずステップ 1 では，一部分削除さらた入力画像を三つの領域に分割する. それぞれは消失領域と, その周辺領域と, それ以外のデータ領域である. 消失領域とは画素が削除された部分である．削除された画素を消失画素と呼ぶ．周辺領域とはある正方形ウィンドウ内に一画素でも消失画素含むウィンドウの中心画素の集合である. そして，周辺領域の平均画素値を初期値としてす

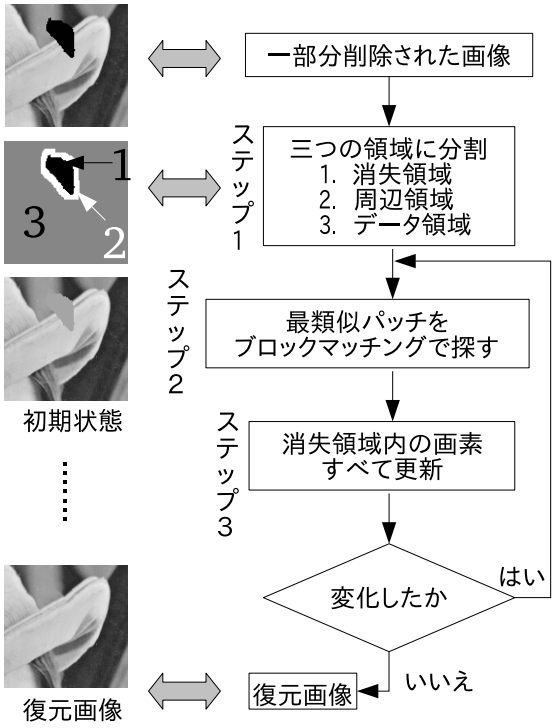

図 1: テクスチャ合成に基づく画像修復手法 [1]

べての消失画素に与える.

次に, ステップ２では最類似パッチの探索を行う．具体的には消失領域と周辺領域の画素を中心とするある正方形ウィンドウWをブロックとし，データ領域を探索領域とし，ブロックマッチングを行う．プロックマッチングの結果をそのウィンドウWの最類似パッチとする．この際, ブロックマッチング法をフルサーチとし, 誤差尺度を SSD とする.

最後に, ステップ３では最類似パッチに基づき消失画素を更新する. ここである消失画素に注目する，その消失画素を含むウィンドウWが複数存在する．すべてのウィンドウWの最類似パッチがステップ２で探索済みである．それぞれの最類似パッチ内に注目する消失画素と同じ位置に画素が存在する．それらの画素値を使い，各パッチの信頼度を考量した重みを付け，ある重み付き平均値が算出される．それが注目する消失画素の新たな値となる．このように，すべての消失画素が更新される.

前回と比べすべての消失画素が変化しなくなるまで, 最類似パッチの探索と画素の更新を続ける. 演算量の面では, ステップ２の演算量が全体の９０%以上占めている.

2.2 画像修復のためのブロックマッチング

文献 [11] のステップ２では，最類似パッチを探索するため，ブロックマッチングを用いた. 他のテクスチャ合成に

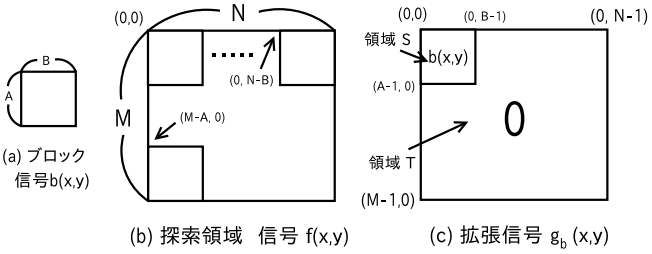

図 2: 信号の表現

基づく画像修復法 [7], [9], [10] においてもブロックマッチングを利用している. ここでは，画像修復のためのブロックマッチングの定義と特徴を紹介する.

図 2(a)(b) に示すある二次元信号 $b(x, y)$ をブロック（ステップ２の正方形ウィンドウW），二次元信号 $f(x, y)$ を探索領域（ステップ２のデータ領域）と呼ぶことにする. ここで $x, y \in \mathbf{Z}$ であり, $M, N, A, B \in \mathbf{Z}$ のもとで，探索領域サイズ $M \times N$ がブロックサイズ $A \times B$ より相当大きい場合を想定する．すなわち，それぞれ次のように定義される.

$$b(x, y), \quad x = 0, 1, \ldots, A-1, \quad y = 0, 1, \ldots, B-1 \tag{1}$$

$$f(x, y), \quad x = 0, 1, \ldots, M-1, \quad y = 0, 1, \ldots, N-1 \tag{2}$$

$$A \ll M, B \ll N$$

探索領域内にブロックと同じサイズのブロックは整数ピクセル精度で，合計 $(N-B+1) \times (M-A+1)$ パターンが存在する．本稿では，その中からブロック $b(x, y)$ と一番近いブロックを探し出すことをブロックマッチングと定義する. ブロック間の誤差尺度は二乗誤差和 (sum of squared differences, SSD)

$$\mathrm{SSD}_{b,f}(u, v) = \sum_{x=0}^{A-1} \sum_{y=0}^{B-1} \{f(x+u, y+v) - b(x, y)\}^2 \tag{3}$$

$$(u, v) \in R_{IN}, \quad u, v \in \mathbf{Z}$$

$$R_{IN} = \{(u, v) | 0 \leqslant u \leqslant M - A \text{ and } 0 \leqslant v \leqslant N - B\} \tag{4}$$

を使用する．ここで，u，v をシフト量とし領域 R_{IN} はシフト可能な範囲とする．このとき，ブロックマッチングは，SSD 誤差が一番小さいときのシフト量 (u_0, v_0) を探し出すことを目的として,

$$\mathrm{SSD}(u_0, v_0) = \min_{(u,v) \in R_{IN}} \{\mathrm{SSD}_{b,f}(u, v)\} \tag{5}$$

と表現される. 上式は, 可能なシフト量 (u, v) について全パターンに調べることを意味する．この操作を特にフルサーチと呼ぶ.

画像修復では正方形ウィンドウWが多数存在する. それに対してデータ領域は一つしかない. したがって，最類似パッチの探索では一つの探索領域から大量のブロックのマッチングを行うことになる. さらに，画素値を更新するたびに最類似パッチの探索を繰り返される. よって，画像修復の全体演算量はこのブロックマッチングに大きく左右される.

3 提案法

画像修復法の演算量を削減し，処理速度を向上するため，文献 [7], [9]–[11] では実装時にフルサーチではなく，近似的な探索法 [7], [9], [11] や FFT ブロックマッチング [10] を利用している.

本稿では，フルサーチの代わりに文献 [13] の FFT 高速ブロックマッチング法を適応することを提案する.

3.1 FFT 高速ブロックマッチングの基本

ここでは，FFT を用いた高速ブロックマッチングアルゴリズム [13] を簡単に説明する. まず, サイズの異なる信号の相互相関を FFT を用いて実行するため, ブロック信号 $b(x, y)$ を信号 $g_b(x, y)$ に拡張する (図 2(c) 参照). 信号 $g_b(x, y)$ が信号 $f(x, y)$ と同じサイズで，T と S と二つの領域から構成される．領域 T はゼロ詰めした部分である．領域 S はブロック信号 $b(x, y)$ である.

次に, 式 (3) の $\mathrm{SSD}_{b,f}(u, v)$ は次式のように書き直すことができる,

$$SSD_{b,f}(u,v) = C_{g_b} - 2cor_{g_b,f}(u,v) + \sum_{(x,y)\in S} \{f(x+u, y+v)\}^2 \tag{6}$$

ただし,

$$C_{g_b} = \sum_{y=0}^{B-1} \sum_{x=0}^{A-1} \{b(x,y)\}^2 \tag{7}$$

$$cor_{g_b,f}(u,v) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} g_b(x,y) f(x+u, y+v), \tag{8}$$

$$(u,v) \in R_{IN}$$

さらに, 式 (6) を式 (5) に代入すると, ブロックマッチングで誤差最小となる座標 (u_0, v_0) は

$$\mathrm{SSD}(u_0,v_0) = C_{g_b} - \max_{(u,v)\in R_{IN}} \{2cor_{g_b,f}(u,v) - S_{f^2}(u,v)\} \tag{9}$$

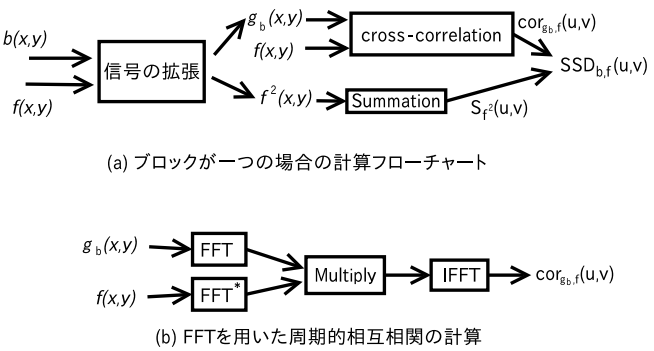

図 3: 計算フローチャート (FFT^* は FFT 後に複素共役をとる演算である)

ただし,

$$S_{f^2}(u,v) = \sum_{(x,y)\in S} f^2(x+u, y+v). \tag{10}$$

と関係し，式 (9) より，$\max_{(u,v)\in R_{IN}}\{2cor_{g_b,f}(u,v) - S_{f^2}(u,v)\}$ を与える座標として求められる.

信号 $g_b(x, y)$ と信号 $f(x, y)$ の相互相関 $cor_{g_b,f}(u, v)$ は FFT を用いて高速に計算することができる.

一方, $S_{f^2}(u, v)$ の計算は再帰型加算によって効率的に計算可能である. 簡単のため，一次元の信号を例に，再帰型加算法を式 (11) で表現する.

$$S_{f^2}(u) = \begin{cases} \displaystyle\sum_{k=0}^{A-1} f^2(u+x), \quad u=0, \\ S_{f^2}(u-1) - f^2(u-1) + f^2(u+A-1), \quad u>0. \end{cases} \tag{11}$$

一般に式 (9) の計算量は，$S_{f^2}(u, v)$ に比べ $cor_{g_b,f}(u, v)$ の計算量が支配的となる．図 3 が上述の手法の計算フローチャートである.FFT 高速ブロックマッチングはフルサーチと同じ精度となることを保証し，演算量が少なくなる.

3.2 画像修復への応用

文献 [11] の画像修復の最類似パッチ探索では，SSD フルサーチを用いられている．FFT 高速ブロックマッチングの特徴の一つは SSD フルサーチの結果を保証することである.

そして，画像修復ではひとつの探索領域に対して，大量のブロックのマッチングが必要である．これに対して，文献 [13] の FFT 高速ブロックマッチング法は二つのブロックを一回の演算でマッチングすることが可能である.

簡単な説明のため，一次元の信号を例にする．$g_a(n)$, $g_b(n)$ と $f(n)$ が同じ長さ N の実数信号で，それぞれの

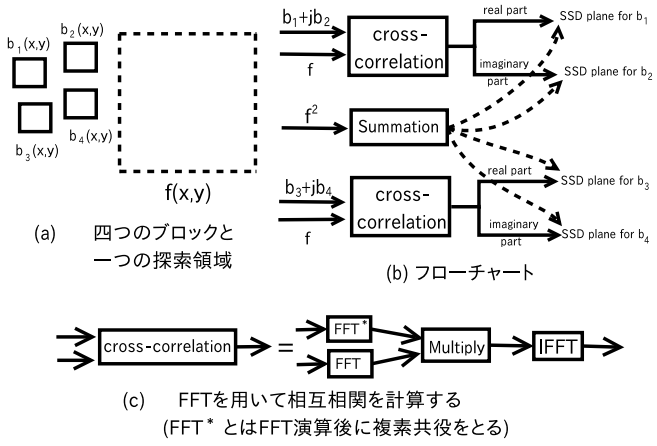

図 4: ブロック４つの場合の FFT 高速ブロックマッチング [13]

DFT は $G_a(k)$,$G_b(k)$,$F(k)$ とする．$g_a(n)$ と $g_b(n)$ をブロックとして，$f(n)$ を探索領域として考えられる．

まずは複素数信号 $h(n) = g_a(n) + jg_b(n)$ を作る．$h(n)$ の DFT は $H(k) = G_a(k) + jG_b(k)$ で表れる．$h(n)$ と $f(n)$ の相互相関 $cor_{hf}(n)$ は次式で表現される．

$$\begin{aligned} cor_{hf}(n) &= \frac{1}{N}\sum_{k=0}^{N-1}\overline{H(k)}F(k)W_N^{-nk} \\ &= \frac{1}{N}\sum_{k=0}^{N-1}(\overline{G_a(k)} - j\overline{G_b(k)})F(k)W_N^{-nk} \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\overline{G_a(k)}F(k)W_N^{-nk} \\ &\quad - j\frac{1}{N}\sum_{k=0}^{N-1}\overline{G_b(k)}F(k)W_N^{-nk} \end{aligned} \tag{12}$$

ただし，$\overline{H(k)}$ は $H(k)$ の複素共役であり，$W_N^{-nk} = e^{-j2\pi/N}$ である．$cor_{hf}(n)$ の実数部分は $g_a(n)$ と $f(n)$ の相互相関となり，虚数部分が $g_b(n)$ と $f(n)$ の相互相関となる．したがって，式 (9) より一回の演算で，二つのブロックマッチング結果が得られる. 図 4 に四つのブロックが一つの探索領域を共有している場合のフローチャートを示す.

他の FFT 高速ブロックマッチング [14] に比べ，文献 [13] のこの長所は画像修復において大きく発揮できる. さらに，共有しているブロック数が多いほど文献 [13] の演算効率がよりよくなる.

以上のことから，画像修復 (図 1) に文献 [13] の FFT 高速ブロックマッチングを適用することで演算量の大幅な削減が考えられる.

3.3 提案法の手順

ここでは，図 1 のステップ 2，すなわち，最類似パッチの探索に FFT 高速ブロックマッチング [13] を適用する際の手順について述べる.

- ステップ 2-1

  探索領域 $f(x, y)$ の FFT 演算と $f^2(x, y)$ の再帰型加算を行う．この際，$f(x, y)$ は入力画像のデータ領域以外の画素値を全てゼロとする．この部分の演算は一回で済む.

- ステップ 2-2

  周辺領域と消失領域の画素を中心とするウィンドウ W をブロック $b(x, y)$ とする．ブロックが多数存在するので，ふたつずつ取り出し図 4(b) のように複素数信号 $b_1(x, y) + jb_2(x, y)$ を作る.

  初期状態では入力画像の消失領域の画素値をすべて周辺領域の画素値の平均値とする．二回目以降は更新された画素値を使う.

- ステップ 2-3

  FFT を用いて複素数信号と探索領域 $f(x, y)$ との相互相関を計算する (図 4(c))．式 (9) にしたがって，二つのブロックそれぞれの最類似パッチを決める.

- ステップ 2-4

  以上のようにすべてのブロック (ウィンドウ W) の最類似パッチを決定し，ステップ 3 に進む.

このステップ 2 の演算は消失画素が更新されるたびに，繰り返して行うが．手順ステップ 2-1 の探索領域についての演算は，消失画素の更新によって結果は変化しないため，一回のみ必要である.

4 シミュレーション

ここでは，図 5,6,7 に示す画像 1,2,3 を使い, 手法 [11] に基づく自動的画像修復を行う. 詳しい条件を表 1 に示す. すべての画像はグレースケールであり，階調数は 256 である.

比較対象として三つの方法を使う．それぞれフルサーチを利用する従来法と，他の FFT 高速ブロックマッチング [14] を利用する手法と，文献 [13] を利用する提案法である.

三つの手法による一回の更新にかかる処理時間を表 2 にまとめる．フルサーチを利用する従来法により FFT 高速ブロックマッチングに基づく方法は速いことが確認できた．そして提案法は文献 [14] の手法と従来法とに対し

表 1: 各画像の詳しい条件

	画像サイズ	消失画素数	周辺画素数	更新回数
画像 1	128 × 128	702	856	24
画像 2	256 × 256	4374	1600	77
画像 3	288 × 352	4834	2314	33

表 2: 一回の更新にかかる処理時間 (秒)

	提案法	[14] に基づく方法	従来法	比例
画像 1	2.8	4.1	414.2	1:1.46:148
画像 2	54.7	83.2	6892	1:1.52:126
画像 3	129	212	13129	1:1.64:101

てそれぞれ約 1.46〜1.64 倍と 101〜148 倍との高速化ができた. 三つの手法は同じ結果になることを確認した.

シミュレーションで使用するコンピューターの CPU は intel Core2 (2.40GHz,2.40GHz), メモリは 2GB であり, 使用する言語は MATLAB 6.5 である.

5 おわりに

本稿では，文献 [13] の FFT 高速ブロックマッチングを画像修復手法の探索に適用することを提案した．提案法は従来法と同じ修復結果になることを保証し，処理速度の大幅な向上が実現できた．シミュレーションにより提案法の有効性を確認した.

今後の課題として，カラー画像や動画像などに提案法を適用することと，重複保持法による探索領域の分割やサブピクセル精度の FFT ブロックマッチング法を利用することで，提案法の処理速度と修復精度のさらなる向上が考えられる.

参考文献

[1] 前田浩幸，高橋健一，太田正光, “欠損画像の修復処理の一方式,” **電子情報通信学会論文誌**, vol.J69-D No1, pp.91-97，Jan. 1986.

[2] S.Mansnou and J.Morel, “Level lines based disocclusion,” *Proc. IEEE Int. Conf. on Image Processing*, vol.3, pp.259-263，Oct. 1998.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image Inpainting,” *ACM Proc. SIGGRAPH'00*，pp.417-424，Jul. 2000.

[4] 小川貴弘，長谷川美紀，北島秀夫, “オプティカルフローを用いた静止画像における失われた輝度値の復元,” **電子情報通信学会論文誌**，vol.J87-D-II No9, pp.1786-1795，Sep. 2004.

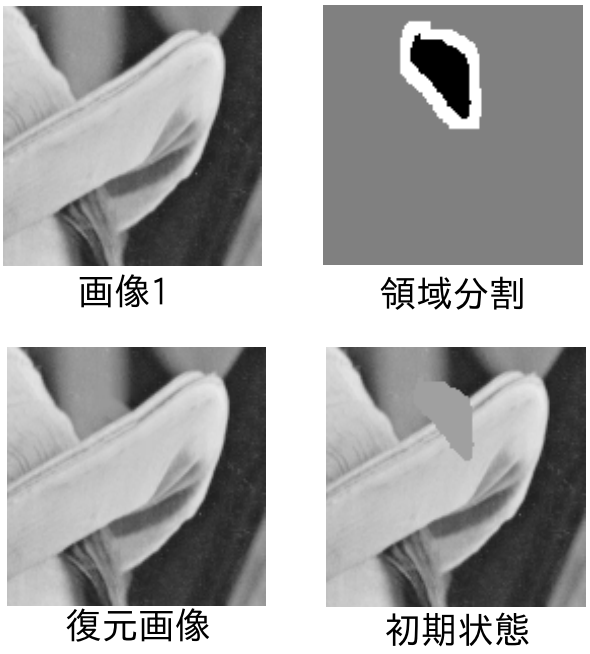

図 5: シミュレーションで使う画像 1 と修復結果

[5] 東海林健二, “テクスチャ画像における欠損部修復の一手法,” **電子情報通信学会論文誌**, vol.J71-D No9, pp.1701-1708，Sept. 1988.

[6] 天野敏之，佐藤幸男, “固有空間法を用いた BPLP による画像補間,” **電子情報通信学会論文誌**，vol.J85-D-II No3，pp.457-465，Mar. 2002.

[7] A. Efros and T. Leung,“Texture Synthesis by Non-Parametric Sampling,” *Proc. IEEE Int. Conf. on Computer Vision*, vol. 3，pp. 1033-1038，Sep. 1999.

[8] I. Drori, D. Cohen-Or, and H. Yeshurun, “Fragment-Based Image Completion,” *ACM Trans. SIGGRAPH'03*，pp. 303-312，Aug. 2003.

[9] A. Criminisi, P. Perez, and K. Toyama, “Region Filling and Object Removal by Exempler-Based Image Inpainting,” *IEEE Trans. Image Processing*, vol.13，No.9，pp.1200-1212，Sept. 2004.

[10] S. Kumar, M. Biswas, S. J. Belongie and T. Q. Nguyen, “Spatio-temporal texture synthesis and image inpainting for video applications,” *Proc. IEEE Int. Conf. on Image Processing*，vol.2, pp.85-88，Sept. 2005.

[11] Y. Wexler, E. shechtman, and M. Irani,“Space-Time Completion of Video,” *IEEE Trans.PAMI*, vol.29, pp.463–476, Mar. 2007.

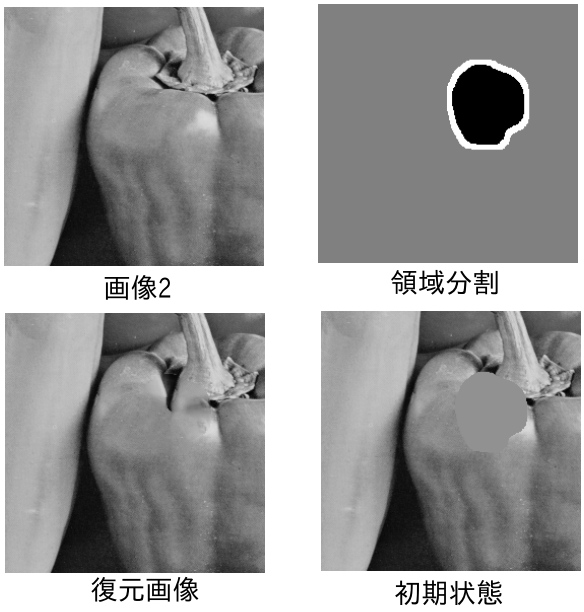

図 6: シミュレーションで使う画像 2 と修復結果

[12] 河合紀彦，佐藤智和，横矢直和,“テクスチャの明度変化と局所性を考慮したパターン類似度を用いたエネルギー最小化による画像修復,” **電子情報通信学会論文誌**, vol.J91-D No9，pp.2293-2304，Sept. 2008.

[13] Z. Li, A. Uemura, and H. Kiya,“An FFT-based full-search block-matching algorithm with Sum of Squared Differences criterion,” *IEICE Trans. Fundamentals.*, to be published.

[14] F. Essannouni, R. Oulad Haj Thami, D. Aboutajdine and A. Salam,“Simple noncircular correlation method for exhausitve sum square difference matching,” *Optical Engineering* 46(10), 107004, Oct. 2007.

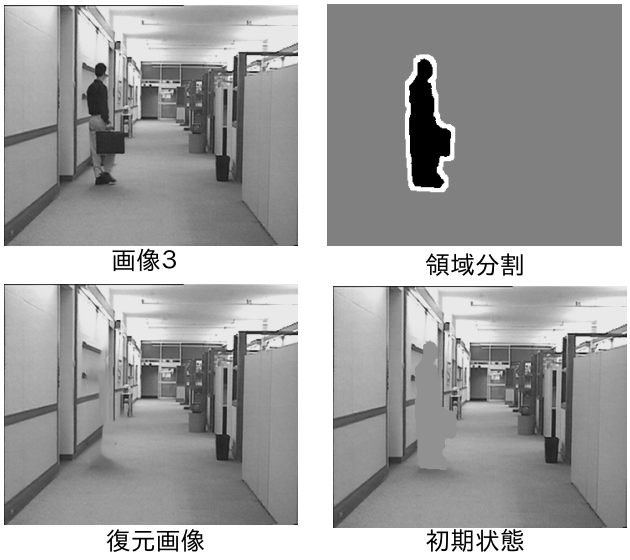

図 7: シミュレーションで使う画像 3 と修復結果