グラフィクスとCAD 102－9
（2001．2．16）

画像のノイズ領域を利用したデータハイディングの一手法

日置 尋久* 森本 佳延† 櫻川 貴司*

*京都大学総合人間学部 †NTTアドバンステクノロジ(株)
E-mail: hioki@i.h.kyoto-u.ac.jp

本報告では，一枚の画像に大量のデータを埋め込むデータハイディングの手法を新たに提案する．我々の手法は，BPCSステガノグラフィと同様に，画像のノイズ領域の画素データを可逆的にノイズ状に変換したデータと置き換えることで画像にデータを埋め込む．BPCSステガノグラフィではノイズ領域を複雑度という尺度により判別する．本研究では，複雑度の性質を調べ，あわせて均一度という尺度を新たに導入することで，ノイズ領域をより的確に判別できるようにした．またこれらの尺度を利用して埋め込むデータをノイズ状に変換するために，本研究ではM系列を用いた．実験では，フルカラー画像に対して，その51%にあたる大きさのデータを画質を損なうことなく埋め込むことができた．

A Data Hiding Method using Noise Regions in An Image

HIOKI Hirohisa*, MORIMOTO Yoshinobu†, and SAKURAGAWA Takashi*

*Faculty of Integrated Human Studies, Kyoto University †NTT Advanced Technology Corporation
E-mail: hioki@i.h.kyoto-u.ac.jp

We propose a new data hiding method for embedding a lot of data in an image. The principle of our data hiding method is the same as that of BPCS-Steganography. We replace pixel values in noise regions in an image with data converted into noise patterns in a reversible way. A criterion called complexity is defined in BPCS-Steganography for discriminating noise regions in an image. We examine the characteristics of the complexity and introduce another criterion called evenness to properly discriminate noise regions in an image. We use M-sequence for converting data into noise patterns. An experiment of data embedding is performed and we have achieved the embedding ratio of 51% to the size of a full color image without degrading the image quality.

1 はじめに

データハイディングとは，いわゆるマルチメディアコンテンツの著作権の保護や各種コンテンツの管理・統合などの問題に対応するために，コンテンツの中に密かに別のデータを埋め込む技術である[1]．その中でも管理・統合の目的では，画像内の対象にキーワードや説明文を添付するなど大量のデータを埋め込むことが求められる．またデータハイディングを暗号通信に応用することを目的としたステガノグラフィ[2]という技術もある．

本報告では，データ管理あるいはステガノグラフィへの応用を目的として，一枚のビットマップ画像に大量のデータを埋め込む手法を新たに提案する．以下，データを埋め込む画像をホスト画像，また画像に埋め込むデータをゲストデータと呼ぶ．また画像は，24bitフルカラーあるいは8bitグレイスケールで，非可逆圧縮が行われないフォーマットのものを用いるものとする．

これまでにホスト画像に大きなゲストデータを埋め込む方法の一つとして，BPCSステガノグラフィ(以下BPCS法)[3]が提案されている．BPCS法では，画像のノイズ状の領域の画素データを他のノイズデータと置き換えたとしても視覚的には影響がないことを利用して，ホスト画像の各ビットプレーンのノイズ状の領域に，可逆的にノイズ状に変換したゲストデータを埋め込む．これは，各ビットプレーンをブロックに分割し，それらのうちのノイズ状のブロックを，ノイズ状のブロックの列に変換したゲストデータと置き換えることで実現される．ブロックがノイズ状とみなせるかどうかを判別する基準としては，複雑度という尺度を用いる．具体的には，ある閾値以上の複雑度のブロックをノイズ状であるとみなしている．またゲストデータをノイズ状のブロック列に変換するには，コンジュゲートと呼ばれる演算を用いる．もしゲストデータをブロック列にしたときに複雑度が十分でなくノイズとはみなせないブロックがあれば，それらに対してコンジュゲート演算を施

すことで，複雑度を高くすることができる．

ところが実際には，複雑度が高いブロックでも，規則的なパターンをもつ場合や，ノイズ状の部分と単調な部分とが混在している場合がある．そこで本研究では，ノイズ状のブロックをより的確に判別する基準を新たに考案した．まず複雑度に関して下限と上限の2つの閾値を設定し，また均一度という尺度を新たに導入した．このような基準に従うようにゲストデータを変換するために，本研究ではM系列を用いることにした．

なおBPCS法において埋め込んだゲストデータを取り出すには，どのブロックにコンジュゲート演算を施したかを示す情報をマップとして埋め込んでおく必要があり，場合によっては，このマップ自体にもコンジュゲート演算を施さなければならない．また埋め込まれたゲストデータを取り出す際には，マップがコンジュゲートされているかどうかを判定しなければならない．一方，我々の手法では，全てのブロックに対して，必ず変換をかけるようにしたため，そのようなマップは不要である．またそれにより，埋め込みおよび取り出しのアルゴリズムを単純にすることができた．

2 ノイズ状のブロックの判別

直観的にいえば，画像には意味の見いだせる領域とそうでない領域がある．意味の見いだせない領域はノイズ状になっており，それを他のノイズデータと置き換えたとしても，視覚的には認知されにくい．よって，ゲストデータを可逆的にノイズ状に変換できれば，それをホスト画像のノイズ領域のデータと置き換えることで，画質を損なうことなく，ゲストデータをホスト画像に埋め込むことができる．

BPCS法では，ノイズ領域を判別するためにビットマップ画像の各ビットプレーンを $2^m \times 2^m$ 画素の小さなブロックに分割し，各ブロックに対して次の複雑度 α を定義している．

$$\alpha = k/(2^m \times 2^{m-1} \times 2) \qquad (1)$$

ここで k はブロック内部で白黒の画素が隣接している画素の境界線の長さである．一方，分母はブロック内部の画素の全境界線の長さである．この定義から，複雑度が低いブロックは白黒のパターンの変化が少ない単調な領域に対応し，複雑度が高いブロックは白黒のパターンが入り組んだ領域に対応するものと考えられる．BPCS法では，ある閾値以上の複雑度をもつブロックをノイズ領域としている．しかし文献[3]でも指摘されている通り，複雑度が最大値(=1)に近い場合は，ブロックは市松模様のパターンに近くなり，逆に規則的なパターンを持つため，ノイズ領域に対応するとは考えにくい．

またホスト画像はその内容によらず均一にブロック分割されているため，図2(a)のように，ブロックがノイズ領域の境界に位置することがありうる．このようなブロックにノイズ状に変換したデータを埋め込むと，ノイズ領域が広がり，ホスト画像の画質が損なわれる危険性がある．しかし複雑度 α では，このようなブロック内部の偏りは評価できない．そこで本研究では，次の均一度 β を新たに導入し，複雑度 α とあわせて用いることにした．

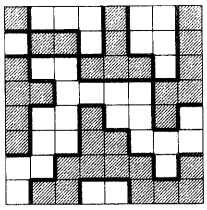
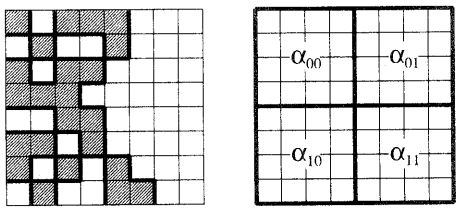


図 1: ブロックの複雑度の計算例



(a) (b)

図 2: (a) ノイズ領域の境界にあるブロック (b) サブブロック分割

$$\beta = \begin{cases} 2\alpha_m/(\alpha_m + \alpha_M) & \alpha_M > 0 \\ 0 & \alpha_M = 0 \end{cases} \qquad (2)$$

$$\alpha_M = \max\{\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}\} \qquad (3)$$

$$\alpha_m = \min\{\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}\}, \qquad (4)$$

ここで $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ は，ブロックを図2(b)のように分割した4つのサブブロックのそれぞれの複雑度である．この定義から，均一度 β は，サブブロックの複雑度の最大値と最小値の差が小さいときに大きな値をとる．すなわち，ブロック内部の白黒の境界線が全体に偏りなく広がっているとき均一度は大きくなる．なお α_M が0のとき，すなわちブロック全体の複雑度が0のときは，均一度は0と定義する．

以上の考察から，本研究では，複雑度 α に関して下限と上限の2つの閾値 α_t, α_T，および均一度 β に関して1つの閾値 β_t を設定し，$\alpha \in [\alpha_t, \alpha_T]$ かつ $\beta \geq \beta_t$ という条件により，ノイズ領域に対応するブロックを判別するものとする．以下，この条件を満たすブロックをノイズブロックと呼ぶ．なお閾値 $\alpha_t, \alpha_T, \beta_t$ は各ビットプレーン毎に設定する．これにより，画像への視覚的な影響が小さい下位のプレーンでは条件を緩やかに設定し，上位に移るにしたがい条件を厳しくすることが可能となる．

3 ゲストデータのノイズ状のブロック列への変換

本研究では，ゲストデータがノイズブロックの列になるように変換して，それらをホスト画像のノイズブロックと置き換えることにより埋め込みを行う．以下では，ゲストデータをブロックサイズ毎に区切ったもののそれぞれをデータブロックと呼ぶ．

BPCS 法では，データブロックの複雑度が閾値より低い場合には，コンジュゲートと呼ばれる演算を施してから埋め込みを行う．これは，対象ブロックと市松模様ブロックとの画素毎の排他論理和を取る演算であり，ブロック b に対してコンジュゲート演算を施したものを b^* で表すと，$\alpha(b^*) = 1-\alpha(b), (b^*)^* = b$ という性質がある．しかし，コンジュゲート演算では任意のデータブロックを，前節で定義したノイズブロックに変換することはできない．そこで本研究では，M 系列から順に得られるビット列をブロック化したものと全データブロックとの画素毎の排他論理和をとることにした．

ただし M 系列の位相 0 からのビット列との排他論理和で全てのデータブロックをノイズブロックに変換できるとは限らないため，M 系列の初期位相を順次ずらし，うまく変換ができる位相を探すようにする．よって，埋め込んだデータを取り出す際には，この初期位相が必要となる．既知の n 次 M 系列については，長さ n のビット列が与えられたとき，その位相を同定できるので [4]，それをブロックとしてゲストデータの先頭に付加して埋め込むことにする．このブロックを位相キーと呼ぶ．もちろん位相キーがノイズブロックでないような位相は避ける．本研究では，以上のようにデータブロックの列を (位相キーを加えた) ノイズブロックの列に変換することを MXOR 変換と呼ぶ．また逆にノイズブロックの列の先頭の位相キーから位相を特定して，残りの部分から元のデータブロックの列を復元することを MXOR 逆変換と呼ぶ．なお M 系列はブロックサイズと同じ次数のものを用いる．

ところで，一般にゲストデータが大きくなるほど，またノイズブロックの判別条件を厳しくするほど，データブロックの列全体を一度に MXOR 変換することが困難になる．そこで本研究では，データブロック列を L_{bs} 個づつのセクションに分割して，それぞれを独立に MXOR 変換することにした．これは，各セクション毎に位相キーが必要となることを意味する．セクション長 L_{bs} は，1 から無限大 (制限なし) まで設定できる．実験では，ブロックサイズが $2^3 \times 2^3$ のとき，セクション長 L_{bs} が数百から数千の値であれば問題なく埋め込みができた．

なお BPCS 法においては，埋め込んだデータを取り出すために，どのデータブロックにコンジュゲートを施したかを示すマップをゲストデータ本体とともに埋め込む必要があり，このマップ自体についても，必要ならばコンジュゲート演算を施さなければならない．また埋め込まれたゲストデータを取り出す際には，マップ自体がコンジュゲートされているかどうかを判定する必要がある．一方，我々の手法では，MXOR 変換は全てのデータブロックに施すため，そのようなマップは不要であり，埋め込みおよび取り出しのアルゴリズムは単純なものになっている．

4 アルゴリズムの詳細

4.1 埋め込みアルゴリズム

埋め込みの際には，

1. ブロックサイズ $n = 2^m \times 2^m$
2. ビットプレーン毎の複雑度の閾値 α_t^i, α_T^i，および均一度の閾値 β_t^i
3. MXOR 変換のための n 次 M 系列の特性多項式およびその初期ビット列を生成するための $2m$ 次 M 系列の特性多項式と $2m$ ビットのシード
4. セクションの長さ L_{bs}(0 なら無制限)

をパラメタとして与える．これらのうち 3 番目の M 系列の情報については，位相 0 からの $2n$ ビットをそのままブロックとしてゲストデータとともに埋め込むことにする．これによりゲストデータの取り出しの際に M 系列を特定することが可能になる [4]．このブロックを M 系列キーと呼ぶ．M 系列としては，この M 系列キーがノイズブロックとなるものを選ぶ必要がある．またセクション長 L_{bs} も埋め込む．

埋め込みアルゴリズムの詳細は以下の通りである．

1. ゲストデータを読み込んで圧縮する．
2. ホスト画像の各画素値をグレイコードに変換し，さらに画像データを順に読んだときに，ブロックが下位のビットプレーンのものから順に得られるようにデータの順序を並び替える．フルカラー画像の場合には，R・G・B のブロックが一つずつ順に現れるようにする．なお，各ビットプレーンの右端と下端でブロックサイズに満たない余りの部分は別にして保存しておく (埋め込みには使用しない)．また各閾値を最下位ビットプレーンのものに設定する．
3. M 系列キーを構築する．画像を読みながらノイズブロックを探して M 系列キーを埋め込む．以下，このように画像のノイズブロックを探して埋め込む操作を単に「埋め込む」と表現する．
4. ヘッダブロックを構築する．ヘッダは，ヘッダブロック数，ゲストデータのファイル名とサイズ，セクションの長さ L_{bs} からなる．このヘッダブロック全体を MXOR 変換して埋め込む．
5. ゲストデータのデータブロックの列を長さ L_{bs} のセクションに分けて，各セクションを MXOR

変換して埋め込む．なお，埋め込みを行うビットプレーンが変わる度に閾値を変更する．

6. 埋め込み終了後，画像データを元の順序に戻す．このとき埋め込みに使わなかった右端と下端の部分もあわせて戻す．最後にグレイコードを純二進コードに逆変換して画像を出力する．

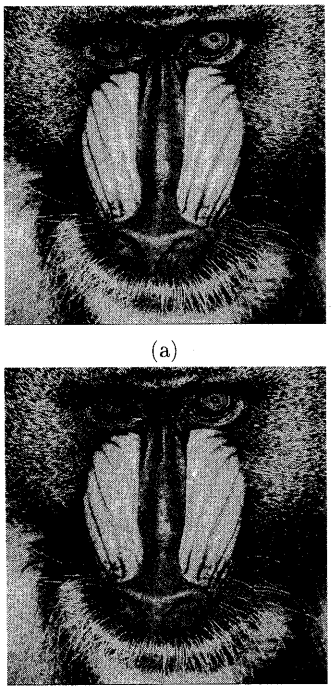
なお，ホスト画像のノイズブロックが埋め込むゲストデータに対して少なすぎる場合や，MXOR 変換できる位相がなかった場合には，エラー終了となる．

4.2 取り出しアルゴリズム

埋め込まれたゲストデータを取り出すには，埋め込み時に用いられたパラメタのうちブロックサイズの乗数 m と閾値 $\alpha_t^i, \alpha_T^i, \beta_t^i$ が必要である．

取り出しアルゴリズムは以下の通りである．

1. 埋め込み時と同様に，ホスト画像をグレイコード変換し，データ順を並べ替える．また各閾値を最下位ビットプレーンのものに設定する．
2. 画像を読んで，ノイズブロックを探しながら，まず M 系列キーを取り出し，M 系列を特定する．
3. 次に MXOR 逆変換によりヘッダブロックを取り出す[1]．ヘッダから，ゲストデータのサイズとセクションの長さ L_{bs} を得る．またこれらの値からセクション数 N_{bs} を求めておく．
4. さらに画像を読み進め，N_{bs} 回 MXOR 逆変換を行い，ゲストデータのデータブロックを全て得る．なお埋め込み時と同様に取り出しを行うビットプレーンが変わる度に閾値を変更する．
5. 取り出されたゲストデータは圧縮されているため，最後に展開してから出力する．



(a)

(b)

図 3: [実験 1] (a) ホスト画像 (b) 埋め込みの結果

ファイル名	サイズ	ファイル名	サイズ
rfc822.txt	106299	rfc1945.txt	137582
rfc1035.txt	122549	rfc1983.txt	123008
rfc1700.txt	458860	rfc2535.txt	110958
rfc1941.txt	150980	rfc2743.txt	229418

表 1: [実験 1] ゲストデータの内容

5 実験

我々の手法の有効性を示すために，3つの実験を行った．実験ではブロックサイズを $2^3 \times 2^3 (= 64)$ とした．また 64 次の M 系列の特性多項式として $F(x) = x^{64} + x^{11} + x^2 + x + 1$ を用い，この M 系列の初期ビット列として，特性多項式 $I(x) = x^6 + x^5 + x^2 + x + 1$ とシード "010101" で生成される M 系列を用いた．実験に用いた計算機は，CPU が Dual PentiumIII 700MHz，メモリが 384MB，OS は Linux 2.2.12 である．なお以下では，画像の上位のビットプレーンから順に第 0 プレーン，第 1 プレーンのように呼ぶことにする．

5.1 実験 1

まず我々の手法で画像に大量のデータを埋め込めることを確認するために，図 3(a) に示したホスト画像に，表 1 に挙げた RFC 文書を tar 形式でまとめたファイルを埋め込む実験を行った．この画像は SIDBA 画像の 1 つであり，24bit カラーで 786432 バイト (512×512×3)，ブロック数は 98304 である．またゲストデータのサイズは，圧縮前で 1454080 バイト，圧縮後で 400799 バイトであった．

図 3(b) は我々の手法による埋め込みの結果を示したものである．埋め込みパラメタは表 2 に示した通りである．このとき画像中の埋め込み可能なブロック数は 58990 個 (471920 バイト) であり，埋め込んだデータ量は全部で 50169 ブロック (401352 バイト) で，ホスト画像の 51.0% に及んだ．それにも関わらず，PSNR[2]は 30.9dB であった．よって，この実験では，画質を大きく損なうことなく大量のデータの埋め込みができたといえる．なお，埋め込みに要した時間は 11.0 秒，ゲストデータを取り出すのに要した時間は，1.9 秒であった．

[1] ヘッダにはヘッダブロック数が最初に含まれており，ブロックを 1 つづつ復元していくことで，ヘッダブロック数を知ることができる

[2] ピーク S/N 比

$[\alpha_t, \alpha_T]$	[0.40, 0.80]@0−3, [0.00, 1.00]@4−7
β_t	0.75, 0.75, 0.50, 0.25, 0.00@4−7
L_{bs}	800

※閾値は，第 0 プレーンの値から順に並べてある．ただし「@i_0-i_1」という記法は，第 i_0〜第 i_1 プレーンで同じ値を使うことを意味する．

表 2: [実験 1] 埋め込みパラメタ

$[\alpha_t, \alpha_T]$	[0.36, 0.70]@0−3, [0.36, 1.00]@4−7
β_t	0.60, 0.60, 0.40, 0.20, 0.00@4−7
L_{bs}	1600

※閾値の記法は，表 2 と同じ．

表 3: [実験 2] 埋め込みパラメタ

5.2 実験 2 — BPCS 法との比較

次に図 4 に示したホスト画像に，図 5 に示す SIDBA 画像[3]をゲストデータとして，BPCS 法および我々の手法のそれぞれにより埋め込む実験を行い，その結果を比較した．このホスト画像は「Lenna」の背景部分を輝度値 127 と 0 の市松模様に置き換えたグレイスケール画像であり，185697 バイト (423×439)，ブロック数は 22464 である．この画像をグレイコードに変換すると，複雑度 1 のブロックが第 1 プレーンに集中して現れた．ゲストデータのサイズは 47425 バイト，圧縮後のサイズは 46791 バイトであった．

図 6(a) は BPCS 法による埋め込みの結果を示している．複雑度 α の閾値は 0.36 とした．一方，図 6(b) は我々の手法による埋め込みの結果を示している．埋め込みパラメタは表 3 の通りである．表 4 に結果の比較を示す．表中のオーバーヘッドとは，ゲストデータ本体以外に埋め込んだデータの量を意味する．

BPCS 法では，複雑度が 1 のブロックも埋め込みの対象となるため，我々の手法に較べて，埋め込み可能なブロック数は多くなっているが，埋め込みによって第 1 プレーンの複雑度 1 のブロックが変更をうけた結果，画質の劣化が起こった．我々の手法の場合も，同様に第 1 プレーンまでデータが埋め込まれたが，画質は損なわれていない．これは，複雑度の上限の閾値 α_T により，背景部分のブロックへの埋め込みが回避され，さらに均一度 β の閾値 β_t により，Lenna と背景との境界にあるブロックへの埋め込みも回避されたためである．またオーバーヘッドの量においても差がみられる．

比較項目	我々の手法	BPCS 法
埋め込み可能ブロック数	5866	6978
埋め込みデータブロック数	5859	6121
オーバーヘッド (byte)	81	2177
PSNR(dB)	33.4	24.4

表 4: [実験 2] 結果の比較

[3]jpeg 形式に変換したものを用いた．

図 4: [実験 2] ホスト画像

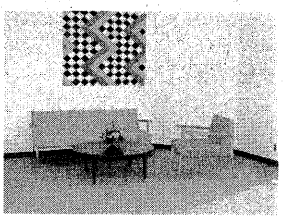


図 5: [実験 2] 埋め込んだ SIDBA 画像

(a)

(b)

図 6: [実験 2] 埋め込みの結果: (a)BPCS 法 (b) 我々の手法

$[\alpha_t, \alpha_T]$	$[0.38, 0.80]$@0−3, $[0.36, 1.00]$@4−7
β_t	0.00@0−7
L_{bs}	1600

※閾値の記法は，表 2 と同じ．

表 5: [実験 3] 埋め込みパラメタ

図 7: [実験 3] 埋め込みの結果

5.3 実験 3

実験 2 と同じホスト画像とゲストデータに対して，ノイズブロックの判別条件を変えて実験を行った．ここでは，均一度の有用性を調べるために，表 5 に示したように均一度 β の閾値を全てのビットプレーンに対して 0.00 とした．

図 7 が埋め込みの結果を示している．Lenna と背景の境界部分にノイズが現れていることが分かる．これは，均一度の閾値を 0.00 にしたことにより，Lenna と市松模様の背景との境界に位置するブロックもノイズブロックと判定され，それらに対しても埋め込みが行われたためである．先ほどの実験 2 の結果では，このようなノイズは視認できないことから，均一度が有効に働いていることが分かる．

6 考察

実験により，我々の手法を用いれば，ホスト画像のノイズ状の領域をうまく利用して，大量のゲストデータを密かに埋め込めることが分かった．また簡単な計算により，おおまかにいって，セクション長をブロックサイズよりも大きくしたとき我々の手法は BPCS 法に較べてオーバーヘッドが少なくなることも分かる．

ただし我々の手法では，ビットプレーン毎の閾値を適切に設定する必要がある．これらを手探りで決定するのは非常に繁雑であるので，閾値の設定を一つの画質パラメタで代表させることが考えられる．たとえば要求する PSNR をパラメタとして受け取り，それを満たすように内部で実際の閾値を自動調整することが考えられる．

ところで，一般に埋め込むデータの量とホスト画像への編集などに対する埋め込んだデータの頑強性はトレードオフの関係にある．我々の手法は大量のデータを埋め込むことを目的としており，現状では BPCS 法と同様に，ホスト画像への編集などに対する頑強性はない．これを改善するには，まず埋め込みに用いる領域に制限を加えることが考えられる．たとえば画像に関する説明文を埋め込む際には，説明の対象となる物体に対応するように埋め込み領域を設定すれば，対象物体に対して変更が加えられない限り，埋め込まれたデータは取り出し可能となる．またこれにより，画質を落としたくない領域への埋め込みを避けることもできる．

7 まとめ

本報告では，データ管理，ステガノグラフィを目的として画像のノイズ領域のデータをノイズ状に変換した埋め込みデータと置き換えることで，大量のデータを埋め込む新しいデータハイディングの手法を提案した．我々の手法は，複雑度と均一度を用いて，単純なパターンのブロック，パターンに偏りのあるブロックを避けて，画像のノイズ領域をより的確に判別できるようになっている．ゲストデータをノイズ状に変換するために M 系列を利用した．実験では，画像の 51% にあたる大きさのゲストデータを画質を損なうことなく埋め込むことができた．またゲストデータ以外に埋め込むオーバーヘッドは BPCS 法より小さくすることができた．

今後の課題として，まず埋め込み領域の指定が挙げられる．これにより，埋め込んだデータと画像の内容との関連付け，埋め込み禁止領域の設定による当該領域の画質保持が可能になる．画質パラメタによる閾値の自動調整も課題として挙げられる．また応用として，WWW ブラウザへの組み込みが考えられる．たとえば，URI を埋め込んだ画像をクリッカブルマップとして使えば，HTML ファイル自体にマップの情報を書く必要がなくなる．

参考文献

[1] Bender, W., Gruhl, D., Morimoto, N. and Lu, A.: Techniques for data hiding, *IBM Systems Journal*, Vol. 35, No. 3&4, pp. 313–336 (1996).

[2] 松井甲子雄: 画像深層暗号, 森北出版 (1993).

[3] Kawaguchi, E. and Eason, R. O.: Principle and application of BPCS-Steganography, *Proceedings of SPIE: Multimedia Systems and Applications*, Vol. 3528, pp. 464–473 (1998).

[4] 柏木濶: M 系列とその応用, 昭晃堂 (1996).