誤りパターン埋込み型ステガノグラフィに関する一考察

A study on steganography based on embedding error patterns

研究者 : 索手　一平　　指導教員 : 福岡　久雄
Researcher : Ippei NAWATE　　Supervisor : Hisao Fukuoka

松江工業高等専門学校情報工学科
Department of Information Engineering, Matsue College of Technology

Abstract

本論文は，テキスト情報をグレースケール画像に埋め込むステガノグラフィ技術を対象とする．データの埋め込み手 法の一つである誤りパターン埋め込み法は，テキスト情報を冗長かつハミング重みの小さいビット列（これを誤りパターンという）に変換し，画像中の LSB 平面との排他的論理和で LSB 平面を置き換える．本研究では，誤りパターン埋め込み法における画質劣化等について考察する．また，Shalkwijk の数え上げ符号を用いた誤りパターンの動的生成方法を提案する．

Keywords

ステガノグラフィ，セキュリティ，誤りパターン

目次

1 はじめに 2

2 ステガノグラフィとは 2

3 誤りパターン埋め込み法 3

4 誤りパターンの生成方法 3

4.1 Slalkwijk の数え上げ符号 4

4.2 誤りパターンの動的生成 4

5 実験 5

6 実験結果 5

6.1 誤り率と埋め込み率 5

6.2 SSIM と誤り率 5

6.3 2 ビットプレーン以降の埋め込みでの実験結果 6

6.4 メッセージ長が同じ場合の劣化について 6

6.5 SSIM の落ちにくい画像の特性についての実験 7

1. はじめに

近年のネットワーク通信の増加に伴い，安全に通信を行うための技術の重要性が高まってきている．特に秘密情報の通信においては重要なものとなる．安全に通信を行う手段として主なものに送信内容の暗号化がある．暗号化により，送信内容を第三者に読み取られない状態とすることで，第三者による盗聴，なりすましといった妨害行為を防ぐことができる．しかし，通信行為そのものは第三者に認知されていることから通信経路の遮断などといった妨害行為をうける可能性を排除できない．この点に対し，ステガノグラフィでは送信内容を別のデータに埋込むことで，第三者から秘密情報の存在そのものを隠ぺいする．つまり，「秘密データの通信を行っている」という事実そのものを隠ぺいすることができる．

本論文ではステガノグラフィ技術の中で最も埋め込みに利用される画像，特にグレイスケール画像に対し，テキスト情報を埋め込むものとして，誤りパターン埋め込み法における埋め込み率と誤り率，画質劣化とのトレードオフ関係を実験的に明らかにする．なお，画質劣化の指標には SSIM[1] を用いる．また，誤りパターンへの変換手法として Shalkwijk の数え上げ符号 [3] を用いたどう適正性手法を提案することで，誤りパターンテーブル法における問題に対する解決案を示す．

2. ステガノグラフィとは

ステガノグラフィとは秘密データを別の媒体に埋め込む技術，研究の総称である．ステガノグラフィでは一般的に埋め込み対象となる画像をカバーデータ，埋め込みによって生成されたデータをステゴデータと呼ぶ．ステガノグラフィを用いた通信は図 1 のように，送信者がステゴデータを生成することで秘密データの存在を隠蔽し，それを送信する．受信者は送信されたステゴデータから埋め込みに対応するアルゴリズムを用いて秘密情報を取り出す．第三者にとってはこの通信はあくまで「何の変哲もない画像通信」であり，送信内容を確認できたとしても秘密情報の存在に気づくことはできない．

しかし，「別の媒体に埋め込む」という特性からカバーデータによっては多くの情報を埋め込むことが困難となる．また，一般的に多くの情報を埋め込むほど

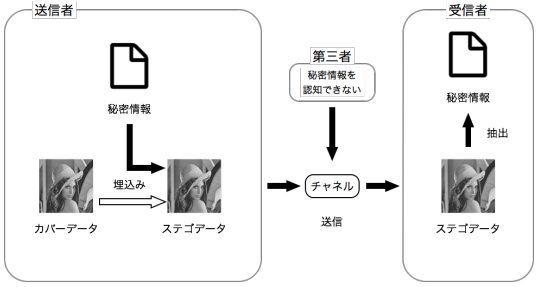

図 1　ステガノグラフィの通信モデル

にステゴデータはカバーデータと比べ劣化し，劣化が大きければ第三者から秘密情報の存在を認知される可能性が高まるためステガノグラフィは意味を成さなくなる．このような特性からステガノグラフィでは以下の 2 点を同時に満たすことが強く望まれる．

1. できるだけ多くの情報の埋め込みが可能である
2. 埋め込みが主観的・客観的に認知されない

埋め込み媒体にはには主にメディアデータが用いられる．ほとんどのメディアデータはいくらかの冗長性を有している．本実験で扱う画像の場合，画像中の各ピクセルにおける LSB などの下位ビットが挙げられる．特に LSB は各ピクセルに対する情報量が少なく，変化させたとしても画像の視覚的変化はほとんど発生しない．この特性を利用した埋め込み手法として LSB 法が提案されており，ステガノグラフィにおける最も一般的な手法として知られている．

この手法は画像の各ピクセルにおける LSB のみを埋め込み対象とすることで，埋め込みによる画像の劣化を抑えた手法である．また，埋め込みは LSB に対ししテキスト情報のバイナリ表現をそのまま置き換えることで行われる．(図 2)

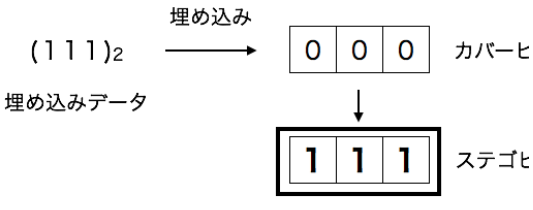

図 2　LSB 法での埋め込み

また，埋め込み時に LSB をテキスト情報のバイナリ表現でそのまま置換することから，LSB のビット列をそのまま抽出することでテキスト情報を画像から抽出することができる．(図 3)

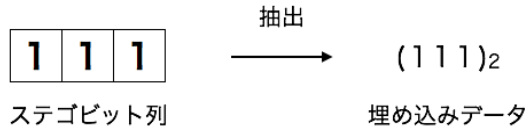

図 3　LSB 法での抽出

単純なアルゴリズムであるため実装が容易である反面，埋め込み後のビット誤りが起きやすく，図 2 のようにすべてのビットが反転してしまう場合もある．このため，ステゴデータの画質が劣化しやすく，同時に隠蔽性に問題を引き起こす可能性がある．

本論文ではカバーデータのうち，埋め込みに対象となるビット列をカバービット列，ステゴデータ中のデータが埋め込まれたビット列をステゴビット列と呼ぶ．また，カバービット列とテキスト情報のバイナリ表現のビット列長をそれぞれ n，m，ステゴビット列中の誤りビットの数を d として，$\frac{m}{n}$ を埋め込み率，$\frac{d}{n}$ を誤り率とする．

これらは埋め込み手法の評価指標である．埋め込み率はいかに効率よくテキスト情報を埋め込めているかを表しており，高いほどに多くの情報を埋め込むことができる．また，誤り率は埋め込みによるビット誤りの発生頻度を表しており，一般的に誤り率が高いほど画質が劣化する．LSB 法では埋め込み率は常に 100 %であるが，一方でビット誤りが起きやすいことから誤り率が高い．

3.　誤りパターン埋め込み法

LSB 法をもとに，より低い誤り率での埋め込みを実現した手法として誤りパターン埋め込み法が知られている．この手法では後述する変換方式 (章 4) を用いてテキスト情報のバイナリ表現を，より長くハミング重みの小さいビット列である誤りパターンに変換する．そして，カバービット列をそのまま置き換えるのではなく，カバービット列と誤りパターンとの排他的論理和でカバービット列を置き換える（図 4)．

テキスト情報の抽出時は，カバービット列とステゴビット列との排他的論理和をとることで誤りパターンを抽出し，その誤りパターンを逆変換することでテキスト情報を抽出することができる（図 5)．

排他的論理和を用いることから，誤りパターンにおける「1」の対応するビットのみがカバービット列中で置き換わるため，ビット誤りを大幅に抑えることができる．一方でより長いビット列を使用することから必然的に埋め込み率は減少する．

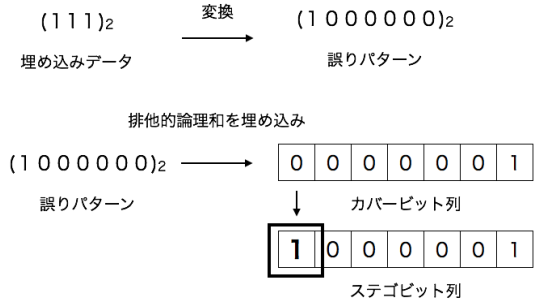

図 4　誤りパターン埋め込み法での埋め込み

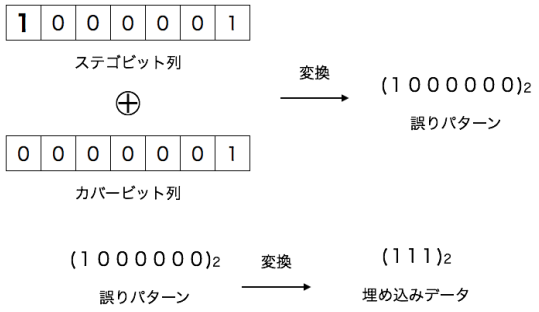

図 5　誤りパターン埋め込み法での抽出

また，この手法はにおける誤り率はカバービット列との比較を行う一方で埋め込まれるテキスト情報の内容にのみ依存する．テキスト情報の抽出時にカバービット列とステゴビット列の比較を用いることから，抽出時にステゴデータとともにカバーデータを必要とする．

4.　誤りパターンの生成方法

一般的な生成方法として表 1 に示すような各文字のバイナリ表現と誤りパターンとの対応を示した誤りパターンテーブルを利用した方法がある．この方法では埋め込み時，抽出時にテーブルを参照することで相互の変換う. しかし，この方法ではテキスト情報の各文字のバイナリ表現を m ビットとしたとき，2^m 個の要素を対応付けたテーブルを用意する必要があることから，文字サイズの増加に伴ってテーブルサイズが指数的に増加し，膨大な大きさとなるため強いメモリ制約下での実装が困難であるという問題点があげられる．

この問題の解決策として文献 [2] では，コスト付き符号化を用いた誤りパターンテーブルの効率的実装方法が提案されている．本論文では誤りパターンテーブ

表 1　誤りパターンテーブル

埋め込みデータ	誤りパターン
000	0000000
001	0000001
010	0000010
011	0000100
100	0001000
101	0010000
110	0100000
111	1000000

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

図 6　パスカルの三角形

ルを用いない方法として，Shalkwijk の数え上げ符号 [3] を用いた誤りパターンの動的生成手法を提案する．

4.1.　Slalkwijk の数え上げ符号

Shalkwijk の数え上げ符号とは長さ n，ハミング重み k の 2 進数列 x の集合に対し 10 進数 $i(x)$ を一意に割り当てることができる符号化手法である．なお i(x) は

$$0 \leq i(x) \leq {}_nC_k$$

を満足する．

またパスカルの三角形を用いた x，$i(x)$ 間の相互変換が可能である．一例として $x = 010100$ としたときの符号化について考える．この 2 進数列 x は $n = 6$, $k = 2$ であるからまず頂点の 1 から数えて 7 段目，左から 3 番目の値である 15 に注目する．ここから x 中の MSB から 1 ビットずつ値を取り出し，0 であれ

ば右斜め上に，1 ならば左斜め上に移動する．また取り出した値が 1 であったとき移動前の座標から見た右斜め上の値を記憶し加算していく．今回の場合，MSB が 0 であるから右斜め上の 10 に移動する．そして次のビットは 1 であるから右斜め上の 6 を記憶し，左斜め上の 4 に移動する．これを繰り返していくことによって最終的に 8 が得られ，これが x に割り当てられる 10 進数 $i(x)$ となる．

この逆変換である 8 からそれに対応する 2 進数列 x を求める方法は，は前述のものと同様，上から 7 段目，左から 3 番目の 15 を起点として求めることができる．逆変換では起点から見て右斜め上の値と変換元である 8 とを比較することで MSB から順に書くビットの値を決定することができる．まず，15 の位置から見て右斜め上の値は 10 であり，これよりも 8 は小さいことから MSB は 0 であることが決まり，そのまま右斜め上の 10 に移動する．10 のいちから見ると右斜め上の値は 6 であり，8 のほうが大きいことから左斜め上の 4 に移動し，次のビットは 1 となる．このとき，値 8 から 6 を引いた値である 2 を記憶し今後の比較では 2 を用いる．これを繰り返していくことによって最終的に 010100 が得られる．これが $n = 6, k = 2$ とした場合に 8 に対応する 2 進数列 x ということになる．$n = 6, k = 2$ の 2 進数列 x の集合に対する 10 進数 $i(x)$ の割り当てを表 2 に示す．

表 2　$n = 6, k = 2$ の場合の対応付け

$i(x)$	x	$i(x)$	x
0	000011	8	010100
1	000101	9	011000
2	000110	10	100001
3	001001	11	100010
4	001010	12	100100
5	001100	13	101000
6	010001	14	110000
7	010010		

4.2.　誤りパターンの動的生成

実際の生成では Shalkwijk の数え上げ符号における 2 進数列 x をテキスト情報の各文字のバイナリ表現，$i(x)$ を誤りパターンとして生成を行う．章 4.1 で述べた手順を用いて誤りパターンの生成を行うが，テキスト情報は各文字で異なるハミング重みを持つため，図 **??**のように異なる文字に対して同じ $i(x)$ が割り当てられてしまう．そこで 2 進数列 x の長さ n，ハミング重み k を元に式 (1) で求められる offset_k を求められた i(x) に加算することで一意の誤りパターンとすること

ができる．

$$\begin{aligned} \mathrm{offset}_k &= \sum_{i=0}^{k-1} {}_nC_i \\ \mathrm{offset}_0 &= 0 \end{aligned} \tag{1}$$

この方法を用いることで誤りパターンを動的に生成し，強いメモリ制約下での実装を可能にすることができる．

5. 実験

実験手順の概要を図**??**に示す．以下に示す手順で実験を行った．実験には 256×256px の 8 ビットグレイスケール Bitmap である SIDBA 画像 30 枚を使用し，埋め込みに使用するテキスト情報は当確率で発生する 8bit コードの列とした．テキスト情報の生成にはコジゲンに分布する乱数発生器である SFMT を用いた．

(1) テキスト情報の各コードを章 4.2 にて述べた手法で動的に誤りパターンへと変換し，変換した誤りパターンと画像の LSB 平面との排他的論理和を画像へと埋め込む．
(2) 埋め込み前後の画像を比較し，誤り率，SSIM を算出する．
(3) 誤りパターン長を 8bit から 256bit まで変化させ（1），（2）を繰り返す．
(4) 画像を入れ替えて（3）を繰り返す．

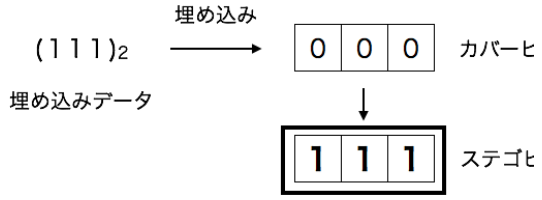

図 7　割り当て

6. 実験結果

6.1. 誤り率と埋め込み率

誤り率の計測結果と文献 [2] において示されている誤りパターン埋め込み法における誤り率の理論的下限曲線を図 8 に示す．なお，誤りパターン埋め込み法における誤り率は埋め込まれるテキスト情報にのみ依存するため，同じテキスト情報が埋め込まれたすべての画像について計測される誤り率は等しい．

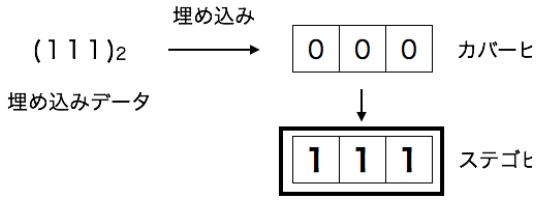

図 8　埋込率に対する誤り率の変化

図 8 より，実際に計測した誤り率は理論的下限曲線と同様，埋め込み率が増加するとともに誤り率も増加し，理論的下限曲線に比べ曲率の低い曲線を描いた．

6.2. SSIM と誤り率

30 枚の SIDBA 画像のうち，SSIM の最も高い画像 10 枚と最も低い 10 枚の埋込率に対する SSIM の変化を図 3 に示す．

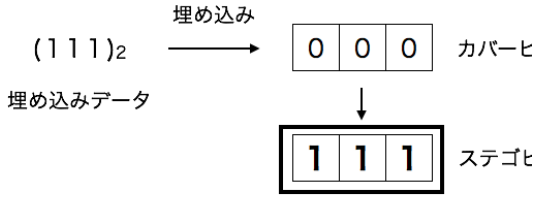

図 9　埋込率に対する誤り率の変化

図 9 よりすべての埋め込み率について SSIM が 0.99 を上回っていることがわかる．一般的に SSIM は 0.98 以上で比較された画像間の見分けがつかないと言われていることから，どのような埋め込み率に対しても画質が大きく劣化することはないと考えられる．このことから，埋め込みを LSB 平面に限定した場合，埋め込み率，誤り率によって画質が大きく劣化することはなく，画質劣化の面でステガノグラフィ技術の隠蔽性に大きく影響を与えることはないということがわかる．同時に図 9 より，画像ごとに SSIM の変化の様子に異なりが見られることがわかる．この 2 点についてさらなる実験・考察を行った，その内容について章 6.3，6.5 にて述べる．

6.3. 2 ビットプレーン以降の埋め込みでの実験結果

LSB のみの埋め込みでは画質劣化がほとんど起きないことから，隠蔽性に影響を及ぼさない範囲で 2 ビットプレーン以降への埋め込みも可能なのではないかと考え，2~4 ビットプレーンまでの埋め込みを行った．実験手順はこれまでと同様の手順で行い，実験対象となるビットプレーン以下のビットプレーンについては埋め込み率 100 %で埋め込みを行った．実験結果をそれぞれ図 10~12 に示す．

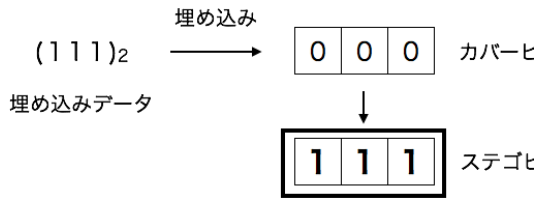

図 10　埋込率に対する誤り率の変化

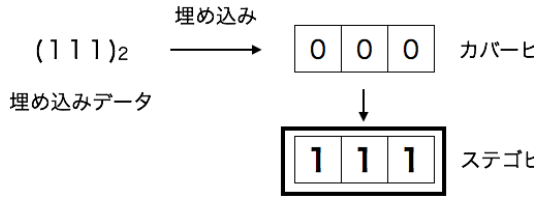

図 11　埋込率に対する誤り率の変化

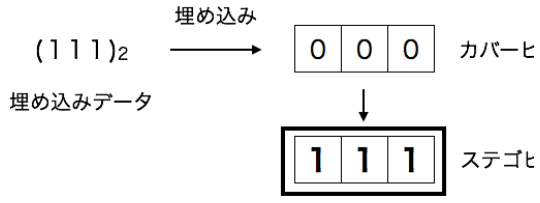

図 12　埋込率に対する誤り率の変化

図 10 より埋め込み率 0.9 前後で SSIM が 0.98 を下回り始めていることがわかる．このことから，LSB への埋め込みを 100 %として 2 ビットプレーンまで埋め込みを行うことで，ほとんどの画像に対し，ステガノグラフィの視覚的な隠蔽性に影響を及ぼさない範囲で，約 1.9 倍のテキスト情報の埋め込みが実現できるといえる．

図 11 において 3 ビットプレーンまで埋め込み率 100 %での埋め込みを行ったとしても SSIM が 0.98 を上回る画像があることがわかる．また，図 12 においても最大で埋め込み率 50 %前後までテキスト情報を埋め込むことができることがわかる．これらのことから適切な画像を選択することでより多くのテキスト情報の埋め込みが可能になるということが言える．

6.4. メッセージ長が同じ場合の劣化について

埋め込み範囲を変えて同じメッセージ量を埋め込む場合，埋め込み範囲を広げることで誤りパターン長を長くすることができることからステゴデータの誤り率を下げることができる．例えば，LSB のみへ埋め込み率 100 %，つまり誤りパターン長を 8 ビットとしてで埋め込んだ場合，256×256px の画像であれば 8192 文字の埋め込みが可能である．これを 2 ビットプレーンにまで埋め込み範囲を拡張し 8192 文字を埋め込んだ場合，単純に埋め込み範囲が倍になることから誤りパターン長を 2 倍の 16 ビットとして埋め込みを行うことができるため，誤り率を半減させることができる．このことから，メッセージ長を同じ長さにして LSB のみへ埋め込んだ場合と，LSB に加え 2 ビットプレーンにまで埋め込んだ場合の SSIM について，同様の実験手順で比較を行った．実験結果を図 13 に示す．この結果から LSB のみへ埋め込みを行った場合の SSIM が埋め込み率にかかわらず，常に 2 ビットプレーンにまで埋め込み範囲を拡張した場合を上回っていることがわかる．このことから埋め込み範囲を拡張することによる画質劣化の抑制は基本的に不可能であるということが言える．しかし，この図において SSIM は 0.98 を上回っており，視覚的な隠蔽性に問題がないことから，誤りによる埋め込みの検出を回避する目的に利用するという面では有用であるといえる．

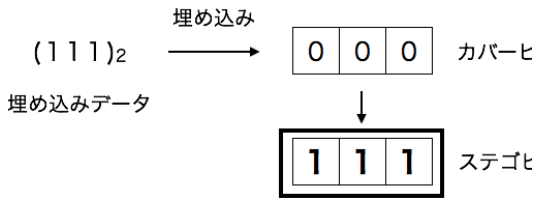

図 13　埋込率に対する誤り率の変化

6.5. SSIM の落ちにくい画像の特性についての実験

章 6.2 にて，画像によって SSIM の変化が大きくことなることを示した．これについて実験者の主観で画像評価を行ったところ，複雑度の高い画像ほど SSIM が下がりにくくなる傾向にあるということがわかった．ここで複雑度とは画素値の明暗の変化の激しさを表しており，複雑度が高いほどにノイズ画像のようになる．これを客観的に評価するため，２つの評価方法を考案し実験を行った．これらの実験では画像の特性が正規分布であることを前提とするため，第 6.2 の実験で使用した 30 枚の SIDBA 画像に加え，同サイズの 8 ビットグレースケールビットマップ 70 枚を Web 上で収集し使用した．

6.5.1. 画素周辺との分散値による実験

画像を図 11 のように少区間に分割し，その少区間中での画素値の分散値の平均を求めることで，画像の複雑度を判別できると考え以下の手順で実験を行った．

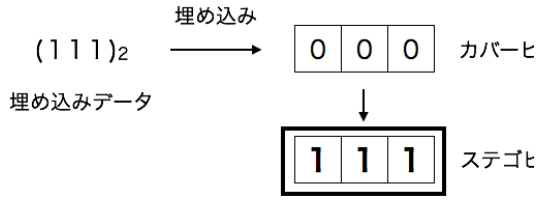

図 14　埋込率に対する誤り率の変化

1. 各画像の少区間ごとの画素値の分散を計算し，それらの平均を求める．
2. 少区間のサイズを変え，再度 (**??**) を繰り返す．
3. 画像を入れ替えて (2) を繰り返す．

この実験では 256×256px の画像を使用したことから，少区間のサイズは画素の漏れや重複なく計算が行えるよう一辺が画素数 2,4,8,16,32,64 個分である正方形とした．少区間ごとの画素値の分散の平均値と埋め込み率 100 %で LSB に埋め込みを行った場合の SSIM との相関を表 4 に示す．この結果から少区間を細かく設定することで SSIM との高い相関が得られていることがわかる．このことからは予想通り SSIM と小区間での画素値の変化が高いこと，つまり複雑度の高い画像ほど SSIM が高く保たれる傾向にあるといえる．

表 3　誤りパターンテーブル

少区間の一辺の画素数	相関
2	0.721631188
4	0.717703982
8	0.65476536
16	0.569234861
32	0.466591644
64	0.337251306

6.5.2. 画素区切りと面積

画素値を 0〜5，6〜10 というように細かい区間で区切り，その区間内の画素値を持つ画素のみを画像から抽出する（以下抽出画像とする）．抽出された画素によって構成される面積を計算することで，画像の複雑度を判別できると考え以下の実験を行った．また，本実験は章 6.5.1 と同様の画像 100 枚を用いて実験を行った．

1. 画像から画素値区間ごとの画素のみを抽出し，抽出画像を生成する
2. 抽出画像を区間内の画素とそれ以外の画素とで 2 値化する
3. 生成された画像それぞれに対しラベリング処理を行い，抽出画像ごとの最大面積を計算する
4. 抽出画像ごとの最大面積の中で最大の面積を求める
5. 画像を入れ替えて (1)〜(4) を繰り返す
6. 画素値区間の分割数を変更し，(5) を繰り返す．

画素は 10，20，30 個の区間に分割し，それぞれ計測を行った．計測された最大面積と埋め込み率 100 %で LSB に埋め込みを行った場合の SSIM との相関を表 **??**に示す．この結果から画素値区間の分割数をを 20 または 30 とすることで高い相関が得られていることがわかる．このことから近しい画素値によって構成される大きな領域を含む画像ほど SSIM が減少しやすいといえる．

表 4　誤りパターンテーブル

分割数	相関
10	0.721631188
20	0.717703982
30	0.65476536

参考文献

[1] Z.Wang, A.C.Bovik, H.R.Sheikh, E.P.Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, Vol. 13, No. 4, pp. 600–612, April 2004.

[2] 合田翔, 渡辺峻, 松本和幸, 吉田稔, 北研二. コスト付き符号化を用いたステガノグラフィ. 信学技法 IT, Vol. 113, No. 153, pp. 5–9, 7 2013.

[3] J.P.M.Shalkwijk. An algorithm for source coding. *IEEE TRANSACTIONS ON INFORMATION THEORY*, Vol. IT-18, No. 3, pp. 395–399, May 1972.