画素置換型ステガノグラフィを用いた Ａｎｄｒｏｉｄアプリの作成

島根大学　総合理工学部　数理・情報システム学科

計算機科学講座　田中研究室

S103104　廣瀬正高

目次

第1章　序論

1.1 研究背景

近年では、スマートフォンなどの普及により写真を撮るという事が日常的に行われるようになっている。

それに伴い、何故撮ったのか、何を撮ったのか分からなくなり、撮影された写真が溢れ返り整理しきれなくなる問題が出てきた。

本研究では、このような問題を解決する手段として、撮影された写真の裏にメモを書く感覚でスマートフォンで撮影した写真にその写真の情報を埋め込むことのできるAndroidアプリを作成した。

1.2 研究概要

本研究では、ステガノグラフィ技術を用いて、画像データ内にユーザが入力した文字情報を埋め込み、さらに埋め込まれた情報を書き出し見ることのできるプログラムを作成しることを目的とする。

第2章　ステガノグラフィの紹介

2.1 ステガノグラフィとは[1]

ステガノグラフィとは、情報隠匿技術の一部をなすものであり、デジタルコンテンツ中に秘密情報を埋め込む技術である。秘密情報を他人に分からないように何か別のデータに埋め込んで伝えることにより、秘密通信の存在自体を隠すことができる。そのため秘密情報が埋め込まれていることを認知されにくい特性を持っている。

通常画像ステガノグラフィは、人間の知覚の鈍い場所や、画像に対する影響が少ない場所に埋め込む必要がある。

2.2 隠蔽の原理

ステガノグラフィでは、秘密データを埋め込んでも、簡単には画像データや音声データが大 きく変化しない手法をとり、情報を隠匿する。 その方法は大きく分けて2種類に分類され、コンテンツの中の冗長性を含むデータ内に埋め込む手法 と人間の生理的特性を利用した手法 である。

コンテンツの中の冗長性を含むデータ内に埋め込む手法 は、もともとデジタル画像やデジタル音声に含まれている冗長部分を利用しようというものである。画像でいえば、冗長性というのは画像の骨格をなす部分ではなく、彩りを添えている部分に相当する。冗長性が高い場合、わずかな量の情報を変更しても、画像全体に決定的な影響を与える怖れはない。

生理学的特徴の利用した手法は、人間の感覚のルーズさを利用して情報を埋め込もうというものである。例えば、人間は輝度情報に対する感度より色に対する感度が低いという特徴がある。

図1はRGB成分が(000,000,000)の画像と(001,001,001)の画像である。人間はこの違いを認識することが出来ない。これらの生理学的特徴を利用して埋め込みを行う。

図1.RGB成分が左(000,000,000)の画像と右(001,001,001)の画像

2.3 ステガノグラフィの種類[2]

本研究では静止画像を対象に情報を埋め込んでいく。情報を埋め込む方法としては大まかに分けて以下の 2 種類に分類される。

・画素置換型ステガノグラフィ
・周波数特性を利用したステガノグラフィ

2.3.1 画素置換型ステガノグラフィ

画素置換型ステガノグラフィは明るさ情報を利用した埋め込み方法で、人の視覚の特性と画像の冗長性を利用する。

カラー画像の場合だと、R（赤）、G（緑）、B（青）の各成分の強さをそれぞれ 8 ビットで表現することが多い。そのような形式での画像データの第 1 ビット目を最上位、第 8 ビット目を最下位ビットとして扱うことにする。各画素の第 1 ビット目から第 8 ビット目までをそれぞれのビットプレーンで集めると、画像を 8 枚のビットプレーンに分けることができる(図 2)。

画素置換型というのは、プレーンそのものを秘密情報と置き換える方法である。最上位ビットがほぼ画像の特徴を反映しているのに対し、カラー画像や白黒濃淡画像の最下位ビットは画像全体への視覚的影響が少なく、別のデータと置き換えても変化はほとんどない。このことを利用して最下位ビットに秘密を埋め込んでも、視覚的な埋め込みの証拠は残らず、秘密データを隠すことができる。

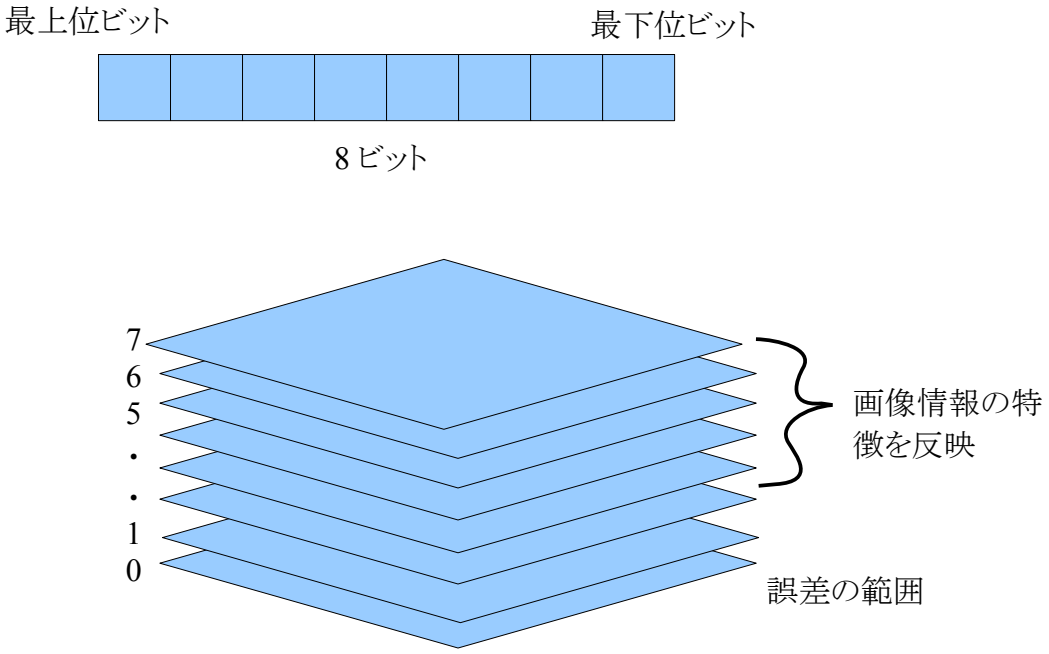

図 2 ビットプレーン

複雑な処理ではないため、後述する周波数成分を利用したステガノグラフィに比べ埋め込みが容易であり、処理時間が少なく済むというメリットがある。
一方で隠匿されたデータはそのまま画像のデータとして残るため、画像の変質 には弱く画像の加工、圧縮などによって埋め込んだデータが破損してしまうという問題がある。

2.3.2 周波数特性を利用したステガノグラフィ

一旦画像を周波数成分に変換し、その中で画像 に影響の少ないとされる部分に周波数を埋め込んでいく手法である。
周波数領域にすると画像の特徴が把握しやすくなり、画像の骨格を形成しているのがどの周波数で、どの部分は無駄であるかということが分かりやすくなる。
画像や音声データは、フーリエ変換やスペクトル拡散、離散コサイン変換などを使い、周波数成分に変換して扱うことができる。そのような変換の際に、画質に影響しないように、ダミーデータの特定周波数帯域の成分を秘密データと置き換える方式である。
低周波成分は画像情報そのものを表し、高周波成分は圧縮などで消えてしまうため、中間周波数成分の一部を置き換えるような方法がとられる。

第3章　情報の埋め込み・抽出方法について

本研究では、画素置換型ステガノグラフィを利用し、情報の埋め込まれた画像の作成を行う。本章では、実際に画像への情報の埋め込み・抽出方法についての方法を述べる。

3.1 情報を埋め込むアルゴリズム

埋め込む情報は、文字であれ画像であれビット0か1のいずれかとなる。例えば2進数01100011を埋め込んでいくとすると、これを最下位ビットから1ビットずつ埋め込んでいくことになる。

処理内容は以下のとおりである。

1. 画像データを読み込み、画像のヘッダ情報を取得する。
2. 読み込んだ画像に隠蔽できる秘密データの容量求める。

埋め込み可能容量(byte)=画像の幅　×　画像の高さ　×　3　／　8

今回扱う画像はフルカラー画像で各ピクセルがRGBの3バイトで構成されているので、3倍することにより情報の埋め込みが可能な容量がビット単位で求まる。これを8で割り、バイト単位で求めておく。

3. 埋め込む情報を読み込み、埋め込みが可能な容量と比較。

読み込んだ情報が埋め込みが可能な容量以下なら4へ。

4. 情報の埋め込み処理。

ユーザが入力した文字情報を文字コード変換し、最下位から1ビットずつ画像データのRGB値の最下位ビットに書き込んでいく。この処理を文字情報がすべて書き込まれるまで繰り返す。

図3は、本研究での文字を埋め込んでいく処理の流れである。

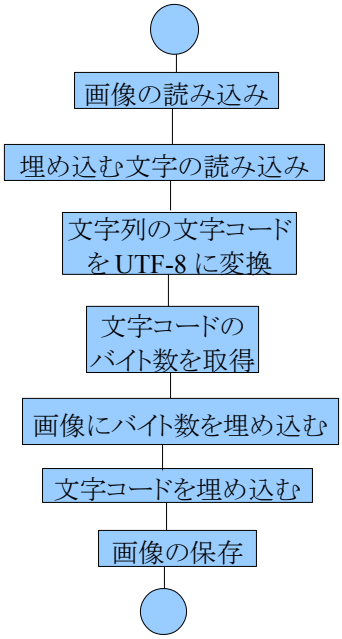

図3. 埋め込み処理の流れ

3.2 情報を抽出するアルゴリズム

　文字が埋め込まれた画像データから埋め込まれた文字を抽出するには、RGB 値の画像データの最下位ビットから書き出せばいい。

1. 隠蔽後画像を読み込む。
2. 抽出処理。

　画像データの最下位ビットを抽出し、データを格納していく。文字データは最下位から埋め込まれているため、取り出す際も最下位ビットから取り出していくことになる。

　図 4 は、本研究での文字を埋め込んでいく処理の流れである。

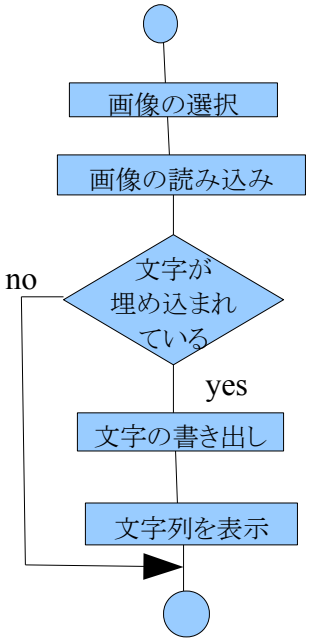

図 4. 書き出し処理の流れ

第4章　作成したアプリケーションについて

4.1UTF-8とは［3］

画像データに文字列を埋め込むにあたり、文字列を文字コードに変換する。

UTF-8は文字を1byteから4byteで表現され、英数の場合は1byte、日本語は3byteまたは4byteで多くは3byteで表現する。例えば「あ」はUTF-8コードでは14909824(10進数)であり、2進数表記だと111000111000000110000000である。

文字範囲が広く、どの国の文字も文字化けすることがないことが大きなメリットとなる。そのため本研究ではUTF-8を用いて文字コード変換を行う事とした。

4.2アプリケーション概要

ユーザが画像を選択し、入力された画像の文字情報を画像の中に埋め込むＡｎｄｒｏｉｄアプリを画素置換型ステガノグラフィを用いて作成する。

本研究は、画素置換型ステガノグラフィを用いてRGB成分のそれぞれ8bit中の最下位ビットに文字コードと文字コードのデータ量2つを埋め込んでいく。順番はRGBの順に埋め込んでいく。画像データ内に文字コードが埋め込まれているか判断するため、2つのデータ量をbyte単位で埋め込む（図3）。

埋め込める容量は画像の　縦×横×3/8byte　で、文字コードのデータ量を埋め込むのに3byteずつ6byte消費し、残った部分に文字コードを埋め込む。そのため、埋め込める文字は画像の大きさに依存し、最大数は画像によって異なる。

3byte　　6byte　　可変長

図3.埋め込むデータ

例えば「ステガノグラフィ」という文字列を画像内に埋め込むとする。

「ステガノグラフィ」はUTF-8で文字コード変換した場合、データ量は24byteとなる。これを2進数で表すと11000となり、これを画像の最終ビットに埋め込んでいく。データ量を埋め込む容量は3byte確保されているので実際には000000000000000000011000を埋め込むことになる。データ量を2つ埋め込む必要があるので、もう1度同じ数値を画像内に埋め込む。

次に埋め込む文字「ステガノグラフィ」をUTF-8で文字コード変換を行う。スの場合であれば、UTF8で111000111000001010111001となり、それを8bit単位に分け最下位から1ビットすつ右シフトしながら画像のRGB地の最下位ビットに埋め込んでいく。

最後に埋め込み処理を行った画像を保存すれが処理は完了である。

4.3 アプリケーションの操作の流れ

アプリケーション実行の大まかな流れは次のとおりである。

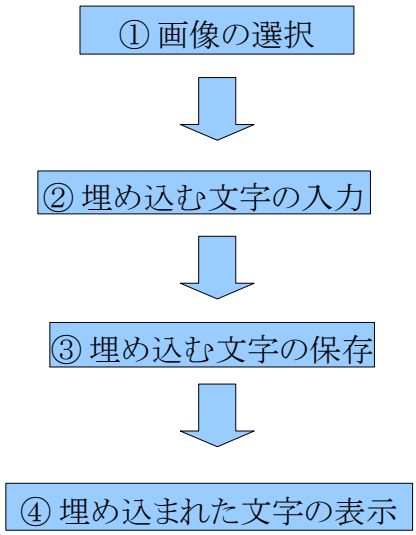

図 4. 実行の流れ

①文字を埋め込む画像を選択する。
②テキスト box に埋め込む文字を入力する。
③保存ボタンを押すと入力された文字が画像に埋め込まれ保存される。
④①②③の操作後、埋め込まれた文字が表示さる。

以下埋め込み処理、書き出し処理について操作の流れを実際に表示される画面を用いて説明する。

4.4 実行結果

4.4.1 書き出し処理手順

1. アプリを実行するとまず図 5 のように表示される。

図 5. 初期画面

各ボタンは以下のような機能を持っている。

・ファイルを開く　　：保存されている画像を選択し読み込むボタン
・文字を埋め込む　　：入力された文字を画像に埋め込み保存するボタン
・編集　　：埋め込まれていた文字をテキスト BOX に挿入するボタン

2. ファイルを開くボタンを押し、保存されている画像の中から文字を埋め込みたい、または埋め込まれた文字を見たい画像を選択し読み込む。

　読み込んだ画像は表示され、文字が埋め込まれていない場合は図 6、埋め込まれている場合は図 7 のようになる。

図 6. 画像に文字が埋め込まれている場合

図 7. 画像に文字が埋め込まれていない場合

4.4.2 埋め込み処理手順

1.抽出処理手順のように画像を選択し表示させる（図）。

2.画像に埋め込みたい文字を入力する。
　　入力方法には以下の2つがある。
　1：テキストBOXにそのまま入力する方法(図8)。
　2：編集ボタンを押して画像に元々埋め込まれていた文字を編集する方法(図9)。これは、画像内に文字が埋め込まれていなかった場合は使用することができない。

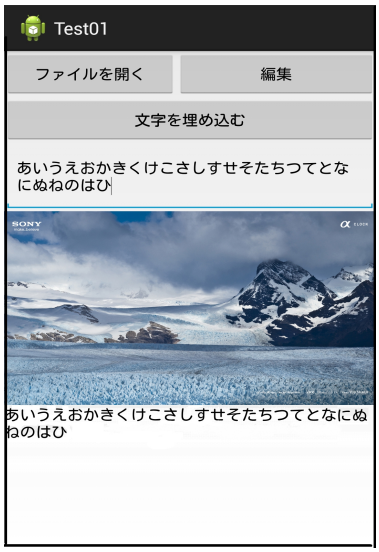

図8.テキストBOXにそのまま入力する　図9.テキストBOXに埋め込まれた文字を挿入

3.文字を埋め込むボタンを押してテキストBOX内の文字を画像に埋め込み、埋め込まれた画像を保存する。

第5章　結論

5.1 まとめ

画素置換型ステガノグラフィを用いて比較的大容量の文字情報を埋め込むことができた。また、文字を埋め込まれた画像を可逆圧縮のPNGで保存することができた。文字が埋め込まれた画像から文字を書き出し表示することができたことで、必要になってくる基本的な処理を行えるアプリが完成した。実際に撮影した写真400x300の中に文字を処理時間5301msで埋め込むことができた。

本研究で作成したアプリを用いることで、画像に情報を埋め込み撮影された写真の整理が容易になると考えられる。

5.2 今後の課題

基本的な処理を行うことができるようにはなったが、処理時間を短縮すると共に、可逆圧縮のPNGでのみ対応している状態であるので、非可逆圧縮のデータ圧縮方法のJPEGやその他の圧縮にも対応したプログラムを考えていくことが今後の課題である。また、4.2節で用いた先頭バイト長の内容がデータ量を埋め込むことようにしているが、その場合文字が埋め込まれていない場合でも埋め込まれていると判断される恐れがある。そのため、より正確に埋め込みを探知するために、データの誤りを検出するための巡回冗長符号であるCRC16[4]などを用いてよりロバストにする必要があろう。

謝辞

本研究にあたり、最後まで熱心な御指導をいただきました田中章司郎教授には心より御礼申し上げます。

また田中研究室皆様には、本研究に関して数々の御協力と御助言をいただきました。厚く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等のすべての知的財産権を本研究の指導教官である田中章司郎教授に譲渡致します。

引用文献

[1]　松井甲子雄,「電子透かしの基礎　マルチメディアのニュープロテクト技術」
[2] 小野束,「電子透かしとコンテンツ保護」
[3] 国際規格番号　ISO/IEC 10646:2011
[4] crc の基礎知識　http://www.wdic.org/w/WDIC/CRC