平成２８年度卒業論文

色相判定による通信方法の提案

情報理工学部 先端工学基礎課程

１３２４０２０　岡川　翔子

指導教員　　寺田 実　　准教授

提出日 ２０１７年 １月３１日

概要

目的

スマートフォンやタブレットなどの携帯端末の普及により, 要所に合わせた通信方法というのが数多く確立されている. 多くの端末に搭載されている機能から新しい通信方法が確立できないかと考えた. 具体的には通信可能な場所を限定でき, また機密性の高い通信方法の確立を目的とする.

方法

本研究では, 多くの端末に実装されているカメラ機能に着目し，カメラを用いて取得した動画から色相情報を検出し, エンコードに利用することで通信ができないかと考えた.

結果

カメラと色相判定の実装はできたが通信の実装までは間に合わなかった. 未実装部分の実装だけでなく, 実装したシステムでもたくさんの改善点があることに気付くことができた.

目次

図目次

第 1 章

序論

1.1　背景

スマートフォンやタブレットなど携帯端末が普及している中, 様々な通信方法が確立されている. 通信方法の代表的なものとして赤外線,Bluetooth, wifi, QR コードによる情報通信などが挙げられる. 赤外線は近接させた端末同士の一対一での通信を行うことができる.Bluetooth は登録された媒体間でのみ通信を行うことができる.wifi は通信可能な区域を制限して通信を行うことができる.QR コードは一枚の画像コードを解析することで通信を行っている. しかし,QR コードには写真からのコードの読み取りも可能なため, 画像がインターネット上で簡単に拡散できてしまう昨今では配布場所を制限することができないという欠点が存在する.

1.2　着目点

QR コードのような一枚の画像解析による通信方法では場所の制限ができないという欠点が存在する. そこで本研究では, 拡散が難しいだろう動画を取得し解析することで通信を行うことができないか, という点に着目した.

1.3　目的

この研究では, 色を高速で切り替える映像を端末のカメラから取得し, 色相情報に割り振られているビットを解析することで情報通信を行うことを提案し, 実装, 評価することを目的とする.

1.4　本論文の構成

論文の構成を簡単に説明する.
この章では「序論」として, 本研究の背景・着目点・目的について述べた.
第 2 章 関連研究を紹介し, それらの研究と本研究の相異点を述べる.

第３章 本研究で作成するシステムの設計について述べる.

第４章 本研究の実装について述べる.

第５章 今後の課題について述べる.

第 2 章

関連研究

2.1 Vinteraction:スマート端末のための振動を利用した情報送信インタラクション

2.1.1 概要

Vinteraction[1] とは, 米澤らが開発したシステムで, 振動モータと加速度を利用した情報送信インタラクションシステムである. 具体的なシステムとして, 送信元の端末を送信先の端末の上に配置し, 送信したい情報を振動にエンコードし送信元の端末のバイブレーション機能を利用して振動させ, その振動を送信先の端末の加速度センサで検知しデコードするというものである. システムの概略図を下の図 2.1 に示す.

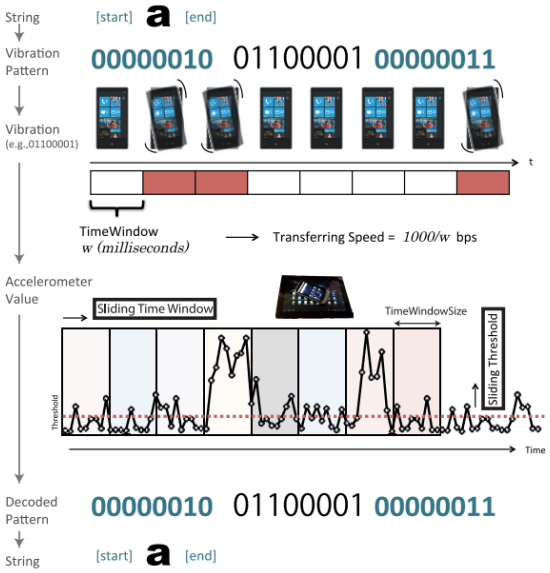

図 2.1 振動モータと加速度センサによる情報通信手法

Vinteraction[1] の研究の目的は, ネットワーク基盤にアクセスすることなく, 特別なハードウェアの追加・変更を必要とせず, 簡単で少ない手順で行える通信方法の提供である.

2.1.2 本研究との差異

Vinteraction[1] では, 多くの端末に搭載されているだろうという理由でバイブレーション機能と加速度センサを利用したとあったが, 本研究ではカメラ機能を利用する.

第 3 章

提案システム

3.1　システム概要

本研究では端末に搭載されているカメラ機能を利用し, 取得した映像から色相を解析することで行う通信システムを提案する. 具体的には, 送信したいものを色相を利用してエンコードし順に描写する動画を作成する. この動画を流しているところを端末のカメラ機能を利用して取得し, 逐一検知した色相をデコードすることで通信を行う. システムの概略図を下の図 3.1 に示す.

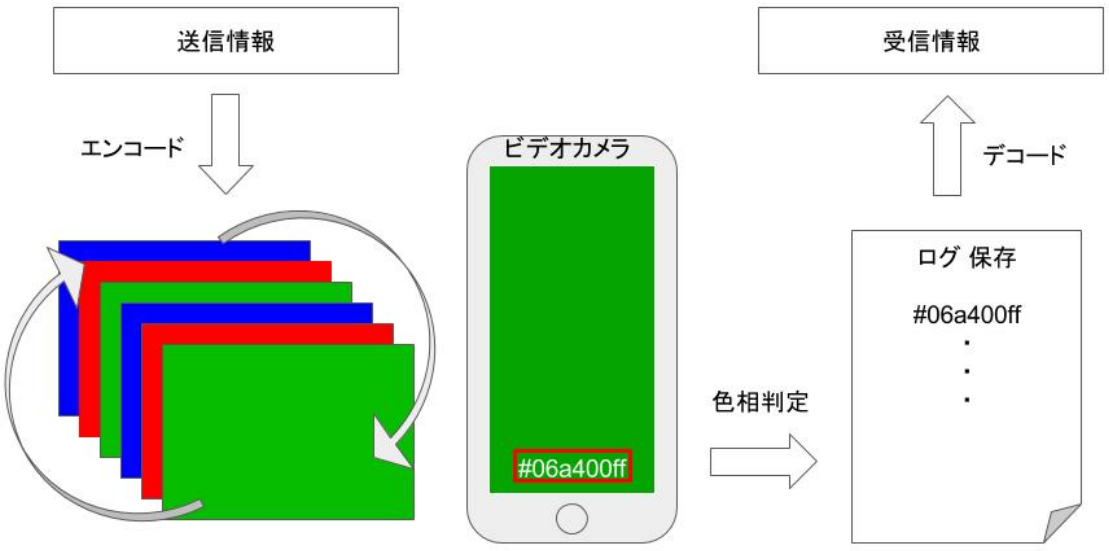

図 3.1　色相判定による通信システムの概要

提案システムのメリットとしては以下の点が挙げられると考えている.

- 色の高速なフラッシュ動画であるため, 人が記憶することは困難であるため機密性が高い
- 動画を流す液晶の大きさに比例し, 配信できる領域・人数を制限できる

以上の観点から, ライブやイベントのような場所と人を制限したいような場合, このシステムは有効である. イベントの参加者のみへのデータの送受信が可能になり,SNS で拡散される確率や QR コード受け渡しだったときに起こりうる転売の被害を減らすことができるだろう.

3.2 実装したシステム

3.2.1 ビデオカメラ

送信元の情報から作成された色情報の動画を取得するためのビデオカメラのアプリケーションを実装した. ビデオカメラの起動中に取得された色情報は画面に表示されるようになっている.

3.2.2 色相判定

カメラを起動している間, カメラが捉えている場所の中央付近のピクセルから色相情報を取得し, ログとしてテキストファイルに保存している. 色相情報は最初に YUV という色空間として取得し,YUV の値を利用して RGB カラーモデルを計算して求めている. この時 RGB は 16 進法で #FF0000 というように表記されているが, ログとして保存する前に RGB=（255,0,0）という表記に変換してから保存している.

YUV とは輝度信号 Y と色差信号 U-V から表現される色空間のことである.[2]
また,RGB とは赤, 緑, 青の三原色から幅広い色の種類を再現することが可能なカラーモデルのことである.

第 4 章

実装

4.1　実装環境

本システムは Android 5.0 以上の Android 端末で動作するアプリケーションである. プログラミング言語は java で実装している.
また, 開発に使用した端末の機種は Xperia Z3 (SO-01G),Android バージョンは Android 6.0.1 である.

4.2 ビデオカメラ

カメラの実装は,Android の API Level 21 から実装された Camera2API を利用した. 以下のような実装になる.

Listing 4.1 カメラ実装

```
try {
    builder = camera2.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
} catch (CameraAccessException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

imagereader = ImageReader.newInstance(320, 240, ImageFormat.YUV_420_888, 2);

builder.addTarget(surface);

builder.addTarget(imagereader.getSurface());

List<Surface> outputs = Arrays.asList(surface, imagereader.getSurface());
textview.append(outputs + "\n");

try {
    camera2.createCaptureSession(outputs, new CameraCaptureSession.StateCallback() {

        @Override
        public void onConfigured(CameraCaptureSession session) {
            textview.append(session + "\n");
            capturesession = session;
            configured();
        }

        @Override
        public void onConfigureFailed(CameraCaptureSession session) {
            // TODO Auto-generated method stub

        }
    }, null);
} catch (CameraAccessException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

imagereader.setOnImageAvailableListener(new OnImageAvailableListener() {

    @Override
```

```
        public void onImageAvailable(ImageReader reader) {
            // TODO Auto-generated method stub
            image_available();
        }
    }, null);

     private void image_available(){
    byte ycopy[] = new byte[WIDTH*HEIGHT];
    byte ucopy[] = new byte[WIDTH*HEIGHT / 2 - 1];
    byte vcopy[] = new byte[WIDTH*HEIGHT / 2 - 1];

    Image image = imagereader.acquireLatestImage();
    if(image == null) return;
    Plane[] plane = image.getPlanes();
    ByteBuffer ys = plane[0].getBuffer();
    ByteBuffer us = plane[1].getBuffer();
    ByteBuffer vs = plane[2].getBuffer();
    ys.get(ycopy);
    us.get(ucopy);
    vs.get(vcopy);
    image.close();

    conv_image(ycopy, ucopy, vcopy);
    bitmap.setPixels(rgbimage, 0, WIDTH, 0, 0, WIDTH, HEIGHT);
    imageview.setImageBitmap(bitmap);
}
```

4.3　色相判定・ログ取得

カメラから色情報を YUV で取得し,RGB へ変換する.YUV から RGB への変換には下の公式を応用する.[2]

$$R = clip(round(1.164383 * C + 1.596027 * E)) \tag{4.1}$$
$$G = clip(round(1.164383 * C - (0.391762 * D) - (0.812968 * E))) \tag{4.2}$$
$$B = clip(round(1.164383 * C + 2.017232 * D)) \tag{4.3}$$

以上の公式を利用して実装した YUV から RGB への変換は以下のようになる.

Listing 4.2　色相（YUB → RGB）変換

```
byte ycopy[] = new byte[WIDTH*HEIGHT];
byte ucopy[] = new byte[WIDTH*HEIGHT / 2 - 1];
byte vcopy[] = new byte[WIDTH*HEIGHT / 2 - 1];
conv_image(ycopy, ucopy, vcopy);

private void conv_image(byte ys[], byte us[], byte vs[]){
    for(int h = 0; h < HEIGHT; h++){
        for(int w = 0; w < WIDTH; w++){
            int i = h * WIDTH + w;
            int y = (int)ys[i] & 0xff;
            int j = h / 2 * WIDTH + (w & 0xfffffffe);
            int u = (int)us[j] & 0xff;
            int v = (int)vs[j] & 0xff;
            int r = clamp256((int) (y + (1.370705 * (v-128))));
            int g = clamp256((int) (y - (0.698001 * (v-128)) - (0.337633 * (u-128))));
            int b = clamp256((int) (y + (1.732446 * (u-128))));
            rgbimage[i] = 0xff000000 | (r << 16) | (g << 8) | b;
        }
    }
 private int clamp256(int v){
    return (v < 0) ? 0 : (v >= 256) ? 255 : v;
 }
```

また, 求められた RGB（16 進法）から RGB(10 進法) へ変換し,log.text へログとして記録していった.16 進法から 10 進法への変換式の例は以下のようになる.

16 進数=12 の時

$$1 * 16^1 + 2 * 16^0 = 18 \tag{4.4}$$

色相の判別では,RGB の情報から赤, 青, 緑の三色へ判別している. 判別方法は以下のようになる.

- R=255〜128, G=64〜0, B=64〜0　の時,　赤
- R=64〜0, G=255〜128, B=64〜0　の時,　緑
- R=64〜0, G=64〜0, B=255〜128　の時,　青

上記の判別方法から漏れてしまった値は各色の区分との距離を以下の式を使用して計算し, 三原色の中で最も近くの色相に区分させるようにした.

$$\sqrt{(r - r')^2 + (g - g')^2 + (b - b')^2} \tag{4.5}$$

解析された色相はこの段階で（255,0,0）,(0,255,0),(0,0,255) の三種類に分けることができた.

4.4　未実装

4.4.1　通信システム

求められた色情報に情報を載せて実際に送受信を行うことができなかった．計測できた色情報の 255 を 1,0 を 0 とビットを指定し, 送信情報を 2 進数に変換することでデータの送受信が行えることの確認を行おうと考えていた．以上のことが確認できた後に, 判別する色の種類を増やすことでビット数を増やしていきどれほど大きな情報がやり取りできるようになるのか計測する予定であった．

第 5 章

結論

カメラアプリは正常に動作させることができ, 取得した色相のデータは実際の色相と 10〜20 の誤差はあれど正確に計測できた. また, 色相の判別は無事三色に判別することが確認できた.

5.1 課題

5.1.1 未実装部分の実装

4.4.1 で記載したシステムの未実装部分である通信システムの実装が最優先であると言える.

5.1.2 取得色相の細分化

現在, 取得された色情報は赤, 青, 緑の三原色に分類されているが, より複雑な通信を可能にするため分類可能な色彩を増やす必要があると思われる. 赤, 青, 緑のうち二つを重ねることで生まれる紫, 黄, 水色, さらに全て重ねなかった時の黒と, 逆に全て重ねたときの白の計 8 色に増やしたいと思う.

5.1.3 色相判別の正確化

5.2 で述べた色相の細分化をするためには, 判別の正確化は必要不可欠であると思われる. 細分化が行われるということは一色が抱えられる色の範囲が狭くなるため, 現在の実装のままで対応できるかという問題が生じると考えられる.

5.1.4 対応機種の拡大

現在カメラの実装に使用している Camera2 API は Android5.0 以上の機種しか対応することができない.Android4.4 以下の Android バージョンが搭載されている端末には, 同様に公開されている Camera API を使用する必要がある. もしくは,C#を用いることで Android だけでなく iphone にも対応可能にすることができるかもしれない.

謝辞

本研究は, 電気通信大学情報理工学部情報・通信工学科コンピュータサイエンスコース寺田研究室において, 寺田実准教授の指導の下で卒業研究として行われました. 寺田 実准教授には卒業研究のアイデア, アドバイス, 参考論文の探し方や論文の書き方, 発表時の方法や注意など様々な部分でご指導頂きました. 心から感謝を申し上げます. 修士課程 2 年の阿部 真之さん, 鈴木 佑樹さん, 平田 吉久さん, 本田 裕人さん, 修士課程 1 年の安部 文紀さん, 山本 愛美さん, 渡邊 裕貴さん, 同期生の佐々木 透さん, 下澤 一輝さん, 肥後 亮佑さん, 藤本 明優さん, 村松 啓寛さん その他多数の方々には研究のアイデアやアドバイス, 協力等をいろいろな面で頂きました. 大変感謝を致します.

参考文献

[1] 米澤拓郎, ”Vinteraction:スマート端末のための振動を利用した情報送信インタラクション”, 情報処理学会論文誌, Vol.54, No.4, 1498-1506, (Apr.2013).

[2] Kyo Shinyuu, ”KlabGames Tech Blog”, 2016/07/01 公開, (最終閲覧日:2017/01/31). http://klabgames.tech.blog.jp.klab.com/archives/1054828175.html

[3] yanzm, ”Y.A.M の 雑記帳” 2010/04/06 公開, (最終閲覧日:2016/10/28). http://y-anz-m.blogspot.jp/2010/04/androidbitmap-tips-jpg-png.html