# 2019 年度版

# 情報通信実験 C

# メディアコンテンツ

# 簡易実験マニュアル

作成者：深田 佳佑

Ver. 1.1

最終更新日：2019/12/02

## 【はじめに】

個人的に，メディアコンテンツ実験においては，実験マニュアルが分かりづらい部分が多く感じた．今後, この実験を受ける人たちにそのような思いをしてほしくない故, この手順書を”簡潔に”執筆した．

また，実験に必要な作業を愚直に手作業で行うのはかなりの時間及び労力を要す．これは非常に時間の無駄である．正直めんどいし，だるい．それにあたり，めんどい部分を勝手に行ってくれるプログラムを組んだので，ぜひ活用してほしい．かなりの楽が出来ると思われる．

なお，本書に実験マニュアルの全てを記載しているわけではなく，かつ，間違いもあるかもしれないので，“本来の”実験テキストもしっかり読むこと．

また，これは 2019 年度版なので来年以降の実験に対応しているかは分からない． 来年以降に本書を扱う場合は，“本来の”実験テキストと照らし合わせてしっかり確認してほしい．これを使用したことで発生した被害は自己責任でよろしく．

尚，本書は Windows のみに対応している．Linux には対応してないので注意すること．
(こんな偉そうに言っといて間違いだらけとかだったらクソ恥ずかしいので, あくまで“参考程度”に読んで欲しい．)

## 【すべての週で共通してやる事】

この節では“半角スペース”を”_”(半角アンダーバー)として表記した．

1. コマンドプロンプトを起動する．

2. 多くの場合，カレントディレクトリが H ドライブなことが多いので,

   cd_/d_hogehoge

   で C ドライブ上で行きたいとこに移動する(Desktop に移動する人が多い).
   hogehoge には絶対パスを入力.

3. PATH を設定する．以下のコマンドを打とう．

(1) echo_%CLASSPATH%
   →%CLASSPATH%が出力されるはず
(2) set_CLASSPATH=C:¥Program_Files_(x86)¥opencv¥build¥java¥opencv-340.jar
(3) set_CLASSPATH=%CLASSPATH%;.
(4) echo_%PATH%
   (これを打つとデフォルトで設定されている PATH がたくさん表示される. 次に設定する PATH が存在しないことを確認する)
(5) set_PATH=%PATH%;C:¥Program_Files_(x86)¥opencv¥build¥java¥x64
   以上で PATH の設定は終わり．

## 【1 週目】

<手順>

1. 配布された Java プログラム(SimpleDitector.java)をコンパイル，実行しよう．
2. 場合によっては，コンパイル時のオプションとして「-encoding UTF-8」が必要．
3. Java ファイルに，新たに int 型変数 i を宣言してそれを for ループの中でインクリメントし，最後に i の中身を表示させると，認識されたオブジェクト(赤枠)を数える手間が省ける．
   下に例を載せるので，こんな感じでやってみては？．

```
    public static void main(String[] args){
    int i=0;
```

～(中略)～

```
    for(Rect rect : detectedObjects.toArray()){
        Imgproc.rectangle(src, new Point(rect.x, rect.y), /*画像 src，始点の座標*/
                new Point(rect.x + rect.width, rect.y + rect.height), /*終点の座標*/
                new Scalar(0, 0, 255), 2); /*線色，線幅*/
        i++;
    }
```

～(中略)～

```
    System.out.println(i); /*生成した赤枠の数を表示する*/
```

(1 週目はあまり書くことが無いので，省略)

## 【2 週目以降】

楽をさせてくれるプログラム(RakuRaku ファイル)を組み，それを GitHub に載せた．以下に GitHub の URL を記載する．コピペ，pull などして是非ともたくさん活用してほしい．

< RakuRaku ファイルの仕様>

1. ファイル名を一括で 0001, 0002, 0003, …と変更してくれる
   (自分で用意した画像ファイルを PC に移すと，ファイル名がぐちゃぐちゃなことが多い．それを手直しで 0001, 0002, 0003,…と名前を変更していくのは非常に手間. 本プログラムを使用することで，数秒で自動的にファイル名を直してくれる．)
2. 正例/負例画像の一覧表.txt ファイルの作成
   (自分でテキストファイルに「./Positive/0001.jpg 0 0 1 640 480 …」というように，手作業で全ての画像に関して打ち込むのはかなり大変．これまた本プログラムが勝手に生成してくれる．)
3. 学習データの反転画像の生成
   (画像認識において，学習データを反転(左右逆)させ，それを使用するのは王道ともいえる(それにデータも簡単に倍増できる)．しかし，自ら反転画像を生成するのは面倒である．それを，これまた本プログラムが(以下略))

(本当は，用意した画像を，自分で指定したサイズにリサイズする機能も含めたかったのだが，時間が足りなかったので断念…．)

**使用言語：Python，動作保証環境：Jupyter Notebook 5.7.8．”glob”, “os”, “PIL”の各ライブラリがインストールされていないと本ファイルを使用できないので注意すること．Anaconda を使っている人なら動かせると思うが，それ以外の環境だと分からない(そこら辺はあまり詳しくないので，すみません)．**

**多分，Google Colaboratory を用いれば Jupyter Notebook のような環境構築無しで動かせると思う．**

<手順>

1. (必要なら)正例画像はピースの部分だけを切り取る．
2. 用意した正例，負例画像ともに，0001, 0002…のように名前を付ける．
3. 正例画像の一覧を矩形表現も含めてテキストファイルで記述する(詳細はテキストを参照)
4. 負例画像の一覧をテキストファイルで記述する(詳細はテキストを参照)．
5. 空の cascade ディレクトリを新規作成
6. 実行ディレクトリ(多くの場合，Win ディレクトリ)の中に
   - (空の)cascade ディレクトリ(学習後に中身が勝手に出来るよ)
   - Positive ディレクトリ(正例画像一覧)
   - Negative ディレクトリ(負例画像一覧)
   - PositiveList.txt(正例画像一覧表)
   - NegativeList.txt(負例画像一覧表)
   - SimpleDitector.java(プログラムコード)

があることを確認．なお，ファイル名はあくまでも例なので，自分の好きなように設定するとよい．また，ベクトルファイルは後で生成される．

< Command Propmt 上で学習させよう>

ここでは半角スペースが分かりづらいので注意！

1. コマンドプロンプトで以下のコマンドを打つ.

opencv_createsamples -info PositiveList.txt -vec vec_V-SignList.vec -num 250 -h 24 -w 48

以下にコマンドの説明を簡潔に述べる．詳細はテキストにも記述されているのでそちらも参照すること.

- ファイル名を間違えないように注意.
- ベクトルファイルは各自好きな名前にして良い.
- このコマンドにおける「250」は自分で用意した正例の画像数．500 枚用意したなら「500」とする.
- -h 24 -w 48：指定した高さ，幅に画像を正規化する．縦長，横長，どちらの画像の方が多いかに注意する.
- 実行後に動作が停止したら，学習不可能な画像があることを意味する．その画像は取り除かなければならないので，それを手探りで地道に探す．コマンドにおける「250」の数字を小さくして何回も実行していこう.
- 枚数が「100」で動いて「101」で動かなくなったら 101 枚目が該当画像なので，101 枚目の画像を消す＆テキストファイルの 101 枚目の記述を消す．二分探索的に探すのがベスト.

2. 上記 1. のコマンドを打った後，ベクトルファイルが正しく生成されたら次のコマンドを打つ.

opencv_traincascade -data cascade -vec vec_V-SignList1.vec -bg NegativeList.txt -numPos 220 -numNeg 212 -featureType HAAR -w 24 -h 48

以下にコマンドの説明を簡潔に述べる．詳細はテキストにも記述されているのでそちらも参照すること.

- 「220」は用意した正例画像の 9 割程度の数にする.
- 「212」は用意した負例の枚数．正例画像では 9 割程度の枚数にしたが，負例画像では用意した全ての枚数を記述して良い.
- HAAR：ここでは Haar-Like 特徴量を用いた．実際には，Haar-Like，LBP など自分の好きな特徴量でよい.
- -w 24 -h 48：上記 1. のコマンドと統一させる.

正しくコマンドが打てたら，学習を開始する．なお，学習時間は用意した枚数によって異なる.

## 【学習終了後】

学習終了後，cascade ディレクトリに cascade.xml 及びその他いくつかの xml ファイルが生成されるので，SimpleDitector.java のプログラムコードにおけるパスを「〜〜〜cascade/cascade.xml」に変更する．変更したら，用意したテスト画像で実行し，生成された識別器の精度を確認しよう．なお，使用する識別器は「cascade.xml」ではなく，「stage〇〇.xml」でも良い．余裕があれば，それらの違いも確認してみよう.

(顔と V サインの両方の認識を行う手順に関しては，書くのが面倒なので各自善処すること．)