情報メディアプロジェクトⅠ：演習課題レポート 2

澤 祐里（21–1–037–0801）

2021 年 7 月 1 日

目次

1　手順

1.1　課題 1

1.1.1　計測対象

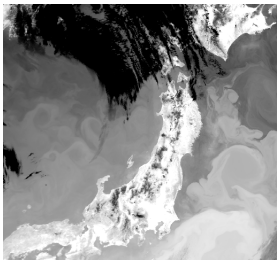
- 計測対象である画像処理は、画像のノイズを除去することができる「3 × 3 メディアンフィルタを使った平滑化」である。この実験では、図 1 のようなプログラムを使う。
- 画像処理に用いる画像は、「幅 512 ピクセル × 高さ 479 ピクセル」の図 2 のような画像である。

図 1　メディアンフィルタを使った平滑化のプログラム

```
private static GImage noisedelete(GImage inputimg,int range){   //rangeは原点からフィルタにする範囲
    int width = inputimg.getWidth();
    int height = inputimg.getHeight();
    GImage outputimg = new GImage(height,width);    //出力用画像を生成
    int[] median = new int[(int)Math.pow(2*range+1 , 2)];//入力された範囲からフィルタ用の配列を生成
    int count;  //フィルタに画素値を挿入していくためのカウント用変数
    int exchange;   //ソートで使う交換用変数

    for(int y=0;y<height;y++) {
        for(int x=0;x<width;x++) {
            if(y-range >= 0 && y+range < height && x-range >= 0 && x+range < width){
                count = 0;
                for(int i=-range;i <= range;i++){
                    for(int j=-range;j <= range;j++){
                        median[count] = inputimg.pixel[y + i][x + j];
                        count++;
                        if(count == median.length){
                            for(int a=0;a<median.length;a++){
                                for(int b=median.length-1;b>=0;b--){
                                    if(median[a] > median[b]){
                                        exchange = median[a];
                                        median[a] = median[b];
                                        median[b] = exchange;
                                    }
                                }
                            }
                            outputimg.pixel[y][x] = median[(median.length-1)/2];
                        }
                    }
                }
            }
            else{
                outputimg.pixel[y][x] = GImage.MIN_GRAY;
            }
        }
    }
    return outputimg;
}
```

図 2　計測対象の画像処理に使う画像



1.1.2　計測方法

- java では、「System.currentTimeMillis()」というメソッドを使うことで、long 型でエポック秒から経過した時間をミリ秒単位で知ることができる。この実験では、このメソッドを用いて画像処理の処理時間を計測していく。
- 具体的な計測方法は「計測したい処理」を「System.currentTimeMillis()」で囲いこみ、処理後の時間から処理前の時間を引くことで、処理時間を計算する。
- 本題に入る前に、この計測方法が正確であるかどうかを確認する。java の「Thread.sleep();」メソッドでは、引数に入れた数字×ミリ秒だけ、プログラムを一時停止することができる。これを利用して「Thread.sleep();」を処理とみなして、処理時間を計測し、「Thread.sleep();」の引数との比較を行うことで、計測方法が正確かどうかを判断する。
- この実験では、「Thread.sleep();」の引数を「1000」として、また、信頼性のあるデータをとるために、計測処理を for 文で 20 回繰り返えす図 3 のようなプログラムを作成して計測した。その結果は、図 4 のようになり、結果をまとめると「1001ms」が 9 回、他は全て「1000ms」となった。1ms の誤差が約 1/2 の確率で発生したが、この程度の誤差はマシンの性能などによる誤差の範疇だと考察できるため、この実験では、この計測方法は正確であると判断した。
- この実験では、画像処理の処理時間のみを正確に計測したいため、「System.currentTimeMillis()」を使って画像処理の処理時間を計測する前に、そもそもこの「System.currentTimeMillis()」という処理自体が時間を使っていないかを確認する。
- 計測方法は、図 5 のように、「System.currentTimeMillis()」を「System.currentTimeMillis()」で囲いこみ、処理後の時間から処理前の時間を引くことで、処理時間を計算する。また、信頼性のあるデータをとるために、これらの処理を for 文で 20 回繰り返している。
- 「System.currentTimeMillis()」の処理時間の計測結果は、図 6 のように、試行回数 20 回の全てで「0ms」となった。これらの結果より、「System.currentTimeMillis()」の処理時間はこの実験では考慮しない。

図 3　「Thread.sleep();」を用いた計測方法の正確さ確認のプログラム

```
private static void Checkaccuracy(){
    for(int i=0;i<20;i++){
        long starttime = System.currentTimeMillis();
        try {
            Thread.sleep(1000);
        }
        catch (InterruptedException e) {
        }
        long endtime = System.currentTimeMillis();
        System.out.println("開始時刻:"+starttime+"ms");
        System.out.println("終了時刻:"+endtime+"ms");
        System.out.println("処理時間:"+(endtime - starttime)+"ms");
    }
}
```

図 4　「Thread.sleep();」を用いた計測方法の正確さ確認の出力結果

```
開始時刻:1625135123951ms
終了時刻:1625135124951ms
処理時間:1000ms
開始時刻:1625135124951ms
終了時刻:1625135125952ms
処理時間:1001ms
開始時刻:1625135125952ms
終了時刻:1625135126952ms
処理時間:1000ms
```

1.2　課題 2

1.2.1　計測対象

1.2.2　計測方法

図 5　「System.currentTimeMillis()」の処理時間を計測するプログラム

```
private static void milltime(){
    for(int i=0;i<20;i++){
        long starttime = System.currentTimeMillis();
        System.currentTimeMillis();
        long endtime = System.currentTimeMillis();
        System.out.println("開始時刻:"+starttime+"ms");
        System.out.println("終了時刻:"+endtime+"ms");
        System.out.println("処理時間:"+(endtime - starttime)+"ms");
    }
}
```

図 6　「System.currentTimeMillis()」の処理時間の計測結果

```
開始時刻:1625128208561ms
終了時刻:1625128208561ms
処理時間:0ms
開始時刻:1625128208561ms
終了時刻:1625128208561ms
処理時間:0ms
開始時刻:1625128208561ms
終了時刻:1625128208561ms
処理時間:0ms
開始時刻:1625128208562ms
終了時刻:1625128208562ms
処理時間:0ms
```