システム実験
実験14回レポート

6119019056 山口力也

2019/07/26 日提出

1　演習

1.1　目的

カラーセンサによる色の表現方法やセンシング技術の理解を深めた. また, カラーセンサについて学んだ知識を統合的に応用した. 特に, 種々の測定環境で, 効率よくカラーセンサのデータを測定する手法を習得した.

1.2　演習 8.2.1

演習 8.1.1 に従い, カラーセンサの配線を行った. 次に, 演習 8.1.2 で使用したサンプルスケッチを用いて配線が正しく行われていることを確認した.

1.3　演習 8.2.2

カラーパターンの白黒部分をセンサで計測し, シリアルモニタに表示された最小値と最大値を記録した. その後,Arduino のサンプルスケッチのパラメータを手動で調整し, カラーセンサのキャリブレーションを行った.

1.4　演習 8.2.3

手動キャリブレーションでは, カラーセンサの RGB 値 (最小・最大) を毎回記録し, プログラムに反映させなければならない. そこで, カラーパターンの白黒部分にセンサを 5 秒間かざせば,RGB 値の最小・最大が決定できる様に改良した.

以下表 1 に, 演習 8.2.2 と演習 8.2.3 の結果を示す.
手動キャリブレーションとオートキャリブレーションの結果の違いは小節 2.1 で考察する.

表 1: カラーパターンの各色の RGB 値

	手動キャリブレーション後			自動キャリブレーション後		
	Red 値	Green 値	Blue 値	Red 値	Green 値	Blue 値
黒	3	4	3	13	12	0
白	219	306	225	469	342	384
赤	136	39	23	232	45	52
緑	49	150	62	93	252	113
青	53	97	145	44	75	182
シアン	41	89	74	138	261	291
マゼンタ	124	74	94	255	144	206
イエロー	187	259	64	326	303	117

1.5 演習 8.2.4

この演習では, 閾値法により色の識別を行なった. 各色の RGB 値を参考にしながら, 色を正しく判定できるように閾値を決めた. 閾値には表 1 を参考にした. 考察は小節 2.3 で行なった.

1.6 演習 8.2.5

試行錯誤による閾値の設定だけでは色を正しく認識するのは難しい. そこで, 閾値を簡単に決める方法として機械学習アルゴリズムの一つである k 近傍法がある. 特徴空間の中で最も近いラベルへ統計的に分類する手法である. これらを用いて色を認識した.

表 2: カラーパターンの各色に対する RGB 値とその平均

	Red 値				Green 値				Blue 値			
	1	2	3	平均	1	2	3	平均	1	2	3	平均
黒	0	6	4	3.3	1	6	4	3.7	0	5	3	2.7
白	262	264	264	263.3	360	360	375	365	261	262	262	262.7

2　課題

2.1　課題 8-2-1

1 の結果を用いて, 手動キャリブレーションと自動キャリブレーションの違いを確認し, その結果が得られた理由を考察せよ. また,Processing の表示と一致するかどうかも報告せよ.

以下図 1, 図 2, 図 4, 図 3 にそれぞれ手動キャリブレーションとオートキャリブレーション時の黒, 白のセンシング時の結果を示す.

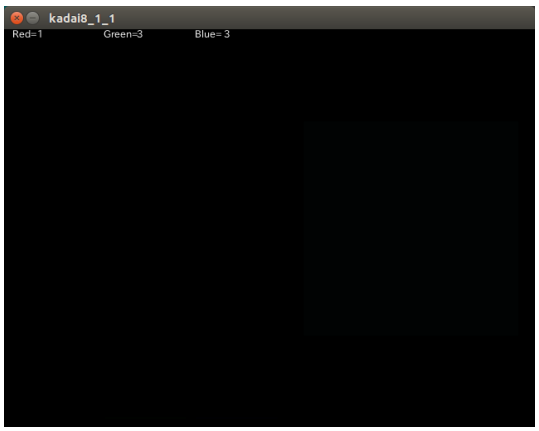

図 1: 手動キャリブレーション (黒)

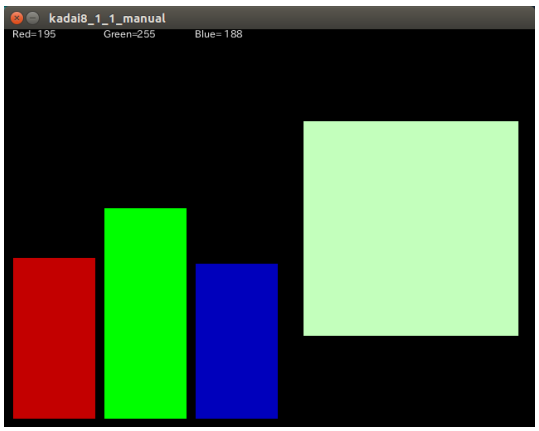

図 2: 手動キャリブレーション (白)

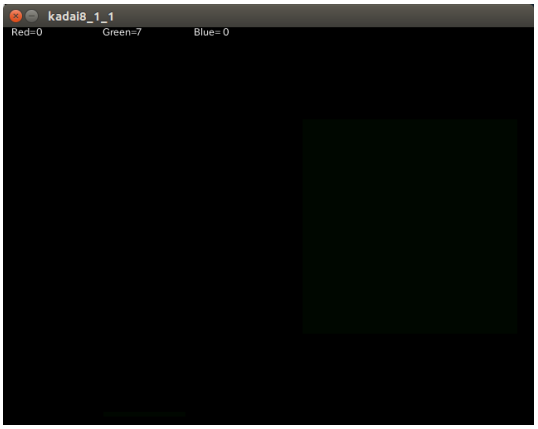

図 3: 自動キャリブレーション (黒)

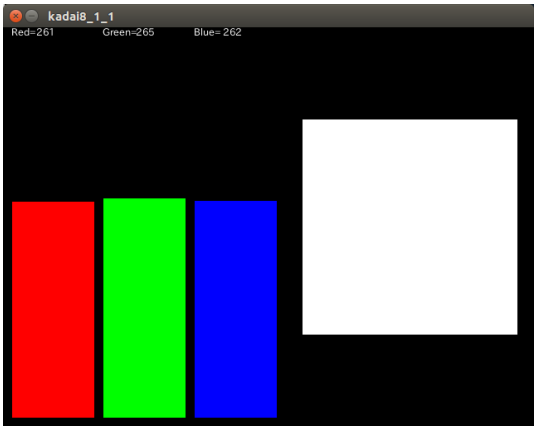

図 4: 自動キャリブレーション (白)

黒色の場合はどちらもうまくセンシングできているが, 白色の場合は自動キャリブレーションはうまくセンシングできているが, 手動キャリブレーションでは若干値が足りなく白色とは異なる色となった. これは手動キャリブレーション時に最大値, 最小値の設定を誤っていたためと考えられる.map 関数を用いて 0~255 の値に変換しているので最大値は合っていても最小値をうまく設定できていない場合, 本来出るべき最大値がでない可能性がある.
手動キャリブレーション時, 自分は 10 個ほどの値しか参考にしなかったため, 自動キャリブレーションとでは比較する数が違うのも原因だと考えられる.

2.2 課題 8.2.2

自動キャリブレーション時間の影響を調べるために,RGB 値の最小, 最大を決める時間を 1 秒と 10 秒でそれぞれ設定し, 黒と白の RGB 値を比較した. 以下表 3 に結果を示す.

表 3: カラーパターンの各色の RGB 値

	自動キャリブレーション (1s)			自動キャリブレーション (10s)		
	Red 値	Green 値	Blue 値	Red 値	Green 値	Blue 値
黒	3	4	3	13	88	14
白	73	77	14	498	506	55

キャリブレーションの時間が短いと最小値がうまくでなかった. が, 逆に長すぎても最大値が大きく, また最小値も大きくなってしまった. そのため実際には適切な秒を選択するべきだと考えられる.

2.3 課題 8.2.3

演習 8.2.4 の閾値法により, カラーパターンの赤, 緑, 青, シアン, マゼンタ, イエローの各色を正しく識別できる様にプログラムを変更せよ.

閾値には表 1 を参考にした. 以下ソースコード 1 にスケッチを示す.

ソースコード 1: 課題 8.2.3(Arduino)

```
1  #include <Wire.h>
2  #define cal_time 10000
3  #define COLOR_SENSOR_ADDR 0x39
4  #define REG_BLOCK_READ 0xCF
5  unsigned int readingdata[20];
6  unsigned int i, red, green, blue;
7  int flg = 0;
8  ////////////////////////////////////////////////////////////

9  //map()関数のパラメータ:キャリブレーション後､適切な値に変更
10 //初期値:RGB の最小･最大を決定
11 unsigned int r_min = 30000, g_min = 30000, b_min = 30000;
12 unsigned int r_max = 0, g_max = 0, b_max = 0;
13 ////////////////////////////////////////////////////////////

14 void setup() {
15   Serial.begin(9600); //シリアル通信の初期化
16   Wire.begin(); //I2C バスに接続
```

```
17   Wire.beginTransmission(COLOR_SENSOR_ADDR); //カラーセンサの
         AD 変換開始
18   Wire.write(0x80); //REG_CTL
19   Wire.write(0x03); //CTL_DAT_INIITIATE
20   Wire.endTransmission();
21   //auto calibration
22   Serial.println("Start Auto Calibration for 5 s");
23   auto_calib(); //loop に入る前の最初の 5 秒間で自動キャリブ
         レーション
24   Serial.println("End Auto Calibration");
25 }
26 void loop() {
27   readRGB();
28   Serial.println("RGB values = ");
29   Serial.print(red); Serial.print(",");
30   Serial.print(green); Serial.print(",");
31   Serial.println(blue);
32   delay(100);
33   if (red > 240 && green > 240 && blue > 240) {
34     Serial.println("White"); //白の閾値
35   }
36   else if (red < 20 && green < 20 && blue < 20 ) {
37     Serial.println("Black"); //黒の閾値
38   }
39   else if (red > 120 && green < 50 && blue < 50 ) {
40     Serial.println("Red"); //赤の閾値
41   }
42   else if (red < 80 && green >150 && blue < 80 ) {
43     Serial.println("Green"); //緑の閾値
44   }
45   else if (red < 50 && green < 50 && blue > 110 ) {
46     Serial.println("Blue"); //青の閾値
47   }
48   else if (red < 70 && green > 80 && blue > 70 ) {
49     Serial.println("Syan"); //シアンの閾値
50   }
51   else if (red > 120 && green < 80 && blue > 90 ) {
52     Serial.println("Magenta"); //マゼンタの閾値
53   }
54   else if (red >120 && green > 120 && blue < 80 ) {
55     Serial.println("Yellow"); //イエローの閾値
56   }
57   else Serial.println("Unknown");
58 }
59
60 void auto_calib() {
```

```
61   unsigned long timeInit;
62   timeInit = millis();
63   while (1) {
64     readRGB();
65     if (red > r_max) r_max = red;
66     if (red < r_min) r_min = red;
67     if (green > g_max) g_max = green;
68     if (green < g_min) g_min = green;
69     if (blue > b_max) b_max = blue;
70     if (blue < b_min) b_min = blue;
71     Serial.print("min = ");
72     Serial.print(r_min); Serial.print(",");
73     Serial.print(g_min); Serial.print(",");
74     Serial.println(b_min);
75     Serial.print("max = ");
76     Serial.print(r_max); Serial.print(",");
77     Serial.print(g_max); Serial.print(",");
78     Serial.println(b_max);
79
80     delay(100);
81     if (millis() - timeInit > cal_time){
82       flg = 1;
83       break;
84     }
85   }
86 }
87
88 void readRGB(){
89   Wire.beginTransmission(COLOR_SENSOR_ADDR);
90   Wire.write(REG_BLOCK_READ);
91   Wire.endTransmission(); //送信完了
92   Wire.beginTransmission(COLOR_SENSOR_ADDR); //送受信開始
93   Wire.requestFrom(COLOR_SENSOR_ADDR, 8); //カラーセンサにデ
       ータ要求
94   delay(500);
95   if(Wire.available() >= 8){ //8byte 以上受信したと き
96     for(i =0; i < 8; i++){
97       readingdata[i]=Wire.read(); //データの受信
98     }
99   }
100  green = readingdata[1] * 256 + readingdata[0];
101  red = readingdata[3] * 256 + readingdata[2];
102  blue = readingdata[5] * 256 + readingdata[4];
103  if (flg == 1){ //自動キャリブレーション終了後
104  //map()関数によるキャリブレーション
105    red = map(red, r_min, r_max, 0, 255);
```

```
106     green = map(green, g_min, g_max, 0, 255);
107     blue = map(blue, b_min, b_max, 0, 255);
108   }
109 }
```

黒や白などの大まかなものは大体認識できたが, シアンなどの中間色は誤って認識することが多かった. 手動キャリブレーションでは値の決め方が難しいためだと考えられる.

2.4 課題 8.2.4

演習 8.2.5 の k 近傍法により, カラーパターンの赤, 緑, 青, シアン, マゼンタ, イエローの各色を正しく認識できる様にプログラムを変更せよ. 以下表 4 に各色に対する RGB 値を示す.

表 4: カラーパターンの各色に対する RGB 値とその平均

	Red 値				Green 値				Blue 値			
	1	2	3	平均	1	2	3	平均	1	2	3	平均
黒	0	6	4	3.3	1	6	4	3.7	0	5	3	2.7
白	262	264	264	263	360	360	375	365	261	262	262	263
赤	157	152	150	153	75	75	75	75	15	15	13	14
緑	44	54	47	48	170	182	176	176	56	45	47	49
青	53	56	57	55	75	80	78	78	181	184	179	181
シアン	41	42	45	43	89	95	92	92	75	80	78	78
マゼンタ	124	128	134	129	76	82	77	78	100	105	103	103
イエロー	190	186	193	190	263	254	266	261	70	76	88	78

この表 4 を参考に k 近傍法による色の識別を行なった. 作成したコードを以下ソースコード 2 に示す.

ソースコード 2: 課題 8.2.4(Arduino)

```
1 #include <Wire.h>
2 #define cal_time 10000
3 #define COLOR_SENSOR_ADDR 0x39
4 #define REG_BLOCK_READ 0xCF
5 unsigned int readingdata[20];
6 unsigned int i, red, green, blue;
7 int flg = 0;
8 unsigned int r_min = 30000, g_min = 30000, b_min = 30000;
```

```
unsigned int r_max = 0, g_max = 0, b_max = 0;

#define max_colors 10//NN: 識別したい色の数
unsigned int ave_colors[max_colors][3] = {
  {263, 365, 263}, //白識別:red, green, blue 値
  { 3, 3, 3}, //黒識別:red, green, blue 値
  { 64, 64, 64}, //Other1：中間色 1
  {192, 192, 192},//Other2：中間色 2
  {153, 75, 14},//赤
  {48, 176, 49},//緑
  {55, 78, 181},//青
  {43, 92, 78},//シアン
  {129, 78, 103},//マゼンタ
  {190, 266, 78}//イエロー
};
////////////////////////////////////////////////////////////

void setup() {
  Serial.begin(9600); //シリアル通信の初期化
  Wire.begin(); //I2C バスに接続
  Wire.beginTransmission(COLOR_SENSOR_ADDR); //カラーセンサの
       AD 変換開始
  Wire.write(0x80); //REG_CTL
  Wire.write(0x03); //CTL_DAT_INIITIATE
  Wire.endTransmission();
  //auto calibration
  Serial.println("Start Auto Calibration for 5 s");
  auto_calib(); //loop に入る前の最初の 5 秒間で自動キャリブ
      レーション
  Serial.println("End Auto Calibration");
}
void loop() {
  readRGB();
  Serial.println("RGB values = ");
  Serial.print(red); Serial.print(",");
  Serial.print(green); Serial.print(",");
  Serial.print(blue); Serial.print(",");
  delay(100);

  //NN 関数の呼び出し
  Nearest_Neighbor();
  delay(500);
}

void auto_calib() {
  unsigned long timeInit;
```

```
52   timeInit = millis();
53   while (1) {
54     readRGB();
55     if (red > r_max) r_max = red;
56     if (red < r_min) r_min = red;
57     if (green > g_max) g_max = green;
58     if (green < g_min) g_min = green;
59     if (blue > b_max) b_max = blue;
60     if (blue < b_min) b_min = blue;
61     /*
62     Serial.print("min = ");
63     Serial.print(r_min); Serial.print(",");
64     Serial.print(g_min); Serial.print(",");
65     Serial.println(b_min);
66     //Serial.print("max = ");
67     Serial.print(r_max); Serial.print(",");
68     Serial.print(g_max); Serial.print(",");
69     Serial.println(b_max);
70   */
71     delay(100);
72     if (millis() - timeInit > cal_time) {
73       flg = 1;
74       break;
75     }
76   }
77 }
78
79 void readRGB() {
80   Wire.beginTransmission(COLOR_SENSOR_ADDR);
81   Wire.write(REG_BLOCK_READ);
82   Wire.endTransmission(); //送信完了
83   Wire.beginTransmission(COLOR_SENSOR_ADDR); //送受信開始
84   Wire.requestFrom(COLOR_SENSOR_ADDR, 8); //カラーセンサにデ
       ータ要求
85   delay(500);
86   if (Wire.available() >= 8) { //8byte 以上受信したと き
87     for (i = 0; i < 8; i++) {
88       readingdata[i] = Wire.read(); //データの受信
89     }
90   }
91   green = readingdata[1] * 256 + readingdata[0];
92   red = readingdata[3] * 256 + readingdata[2];
93   blue = readingdata[5] * 256 + readingdata[4];
94   if (flg == 1) { //自動キャリブレーション終了後
95     //map()関数によるキャリブレーション
96     red = map(red, r_min, r_max, 0, 255);
```

```
97      green = map(green, g_min, g_max, 0, 255);
98      blue = map(blue, b_min, b_max, 0, 255);
99    }
100 }
101
102 int Nearest_Neighbor() {
103   int count = 0;
104   int color = -1;
105   float minDistance = 9999999;
106   float distance;
107   for (int i = 0; i < max_colors; i++) { // i は設定したカラーのラベル
108     distance = sqrt( (red - ave_colors[i][0]) * (red - ave_colors[i][0])
109                     + (green - ave_colors[i][1]) * (green - ave_colors[i][1])
110                     + (blue - ave_colors[i][2]) * (blue - ave_colors[i][2]));
111     if ( distance < minDistance ) { //最短距離の判定
112       minDistance = distance;
113       color = i;
114     }
115   }
116   if (color == 0) Serial.println("White");
117   else if (color == 1) Serial.println("Black");
118   else if (color == 2 || color == 3) Serial.println("Others");
119   else if (color == 4) Serial.println("Red");
120   else if (color == 5) Serial.println("Green");
121   else if (color == 6) Serial.println("Blue");
122   else if (color == 7) Serial.println("Syan");
123   else if (color == 8) Serial.println("Magenta");
124   else if (color == 9) Serial.println("Yellow");
125   else Serial.println("Not detected!"); //例外処理:未検出の場合
126 }
```

2.5 課題 8.2.5

閾値法または k 近傍法どちらか一つを選び,Arduino で識別した色をシリアル通信で Processing 側に送信し, センシングした色とともに識別結果を表示できる様にプログラムを変更せよ.

作成したコードを以下ソースコード 3 に示す.

ソースコード 3: 課題 8.2.5(Processing)

```
1  import processing.serial.*;
2  Serial myPort;
3  float Red, Green, Blue;
4  float judge;
5  void setup() {
6    size(640, 480);
7    myPort = new Serial(this, "COM3", 9600);
8    //myPort = new Serial(this, "COM10", 9600);
9    myPort.bufferUntil('\n'); //改行までメッセージ受信
10 }
11 void draw() { //受信した値で描画
12   background(0);
13   fill(Red,0,0); rect(10,470,100,-Red);
14   fill(0,Green,0); rect(120,470,100,-Green);
15   fill(0,0,Blue); rect(230,470,100,-Blue);
16   fill(Red, Green, Blue); rect(360,110,260,260);
17   print("R="+Red+", "); print("G="+Green+", "); println("B
         ="+Blue);
18   fill(255);
19   text("Red=",10,10); text(int(Red),40,10);
20   text("Green=",120,10); text(int(Green),160,10);
21   text("Blue=",230,10); text(int(Blue),265,10);
22
23   if(judge == 0.0 ) text("White",400,30);
24   else if(judge == 1.0 ) text("Black",400,30);
25   else if(judge == 2.0 ) text("Red",400,30);
26   else if(judge == 3.0 ) text("Green",400,30);
27   else if(judge == 4.0 ) text("Blue",400,30);
28   else if(judge == 5.0 ) text("Syan",400,30);
29   else if(judge == 6.0 ) text("Magenta",400,30);
30   else if(judge == 7.0 ) text("Yellow",400,30);
31
32
33 }
34 void serialEvent(Serial myPort) {
35   String myString = myPort.readStringUntil('\n'); //シリアル
         バッファー読込み
36   if (myString != null){
37     myString = trim(myString); //空白文字など消去
38     float data[] = float(split(myString, ',')); //カンマ区切
         りで複数の情報を読込む
39     if (data.length >1){
40
41       // キャリブレーションなし
42       Red = data[0];
```

```
43        Green = data[1]; //ここに Green の処理を入れる
44        Blue = data[2]; //ここに Blue の処理を入れる
45        judge = data[3];
46        /*
47        // キャリブレーションあり
48        Red = map(data[0], 7680, 40705, 0, 255); //下線部をシ
              リアルモニタの最大値に変更
49        Green = map(data[1], 16648, 56072, 0, 255); //ここに
              Green の処理を入れる
50        Blue = map(data[2], 11520, 50944, 0, 255);; //ここに
               Blue の処理を入れる
51        // max 40705,56072,50944
52        // min 7680,16648,11520
53    */
54
55
56        myPort.clear();
57      }
58    }
59  }
```

また, プログラムの実行結果を以下図 5 に示す.

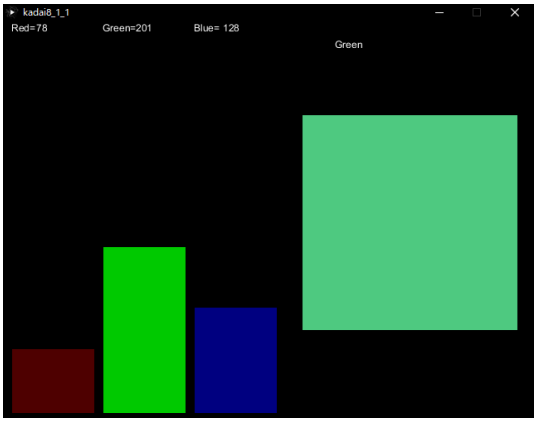

図 5: k 近傍法による色の識別

2.6 課題 8.2.6

課題 8.2.5 のプログラムを用いてカラーパターン境界域すべてについてセンシングを行い,Processing の表示により色を正しく識別できているか確認せよ. またその結果からどのような条件のとき色の誤認識が怒るのか, またどの様にすれば誤認識を改善できるか考察せよ.

以下図 6 に結果の一例を示す.

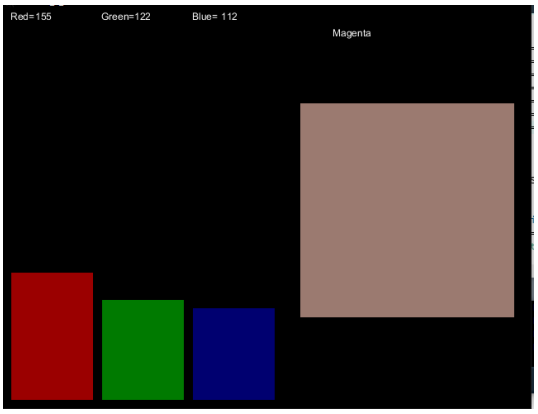

図 6: 境界域での色の識別

図 6 はシアンとマゼンタの境界域における実行結果である.
境界域では二つの色が混ざった様な RGB 値が出てきて, 今回はマゼンタと識別しているが他の境界域では全く異なる色と認識してしまう場合があった. 改善するには k 近傍法の値を変えるか, あるいは最近傍法などの他の手法を用いるなどが考えられる.

2.7 課題 8.2.7

課題 8.2.5 で作成したプログラムを用いて, 手でセンサを素早く (またはゆっくり) カラーパターンの上で移動させた時,RGB 値がどのように変化しているか確認せよ. また, カラーセンサが誤認識を起こす場合, その理由と改善策を考察せよ.

手を素早く移動させたとき, カラーセンサの RGB 値が変化し, 誤認識することが多かった. しかしながら RGB 値の変化に統一性はなかったように思われる. 手をゆっくり移動させたとき, カラーセンサの RGB 値は大きく増え, 大体の場合「白」と認識することが多かった.
手を素早く移動させたときの状態を改善するには直近の何回かのセンシングの結果を保存しておきそれらの平均をとるなどをして急激な変化に耐えれるようにするなどが考えられる. 手をゆっくり移動させたときの状態を改善するには,RGB 値が一定の値を超えたら識別しないようにすることが考えられる. 手をゆっくり移動させた時, 大体の場合白に識別され, また RGB 値も普通の白の場合を大きく超えていた. それらを考慮して RGB 値がある一定の値を超えたら識別を行わないなどをすると良いと考えられる.