```
1  package edu.asu.msrs.artcelerationlibrary;
2
3  import android.util.Log;
4
5  /**
6   * Created by yitaochan on 12/1/16.
7   */
8
9  public class GaussianBlurByte {
10
11 //    int r = 3;
12    //float sigma = 3f;
13    int height = 1066;
14    int width = 1600;
15 //    int[][] red = new int[height][width];
16 //    int[][] green = new int[height][width];
17 //    int[][] blue = new int[height][width];
18    //byte[] processedbyte = new byte[height*width*pixelLength];
19
20  static public byte[] gBlurByte(byte[] b, int w, int h) {
21        String TAG = "GaussianBlurByte";
22        Log.d(TAG, "GaussianBlur Starts");
23        float sigma = 5f;
24        int r = 20;
25 //        int[][] red = new int[h][w];
26 //        int[][] green = new int[h][w];
27 //        int[][] blue = new int[h][w];
28
29     float[][] red = new float[h][w];
30     float[][] green = new float[h][w];
31     float[][] blue = new float[h][w];
32
33        convertToInt(b,w,h,red,green,blue);
34
35        processOne(red,w,h,sigma,r);
36        processTwo(red,w,h,sigma,r);
37
38        processOne(green,w,h,sigma,r);
39        processTwo(green,w,h,sigma,r);
40
41        processOne(blue,w,h,sigma,r);
42        processTwo(blue,w,h,sigma,r);
43
44
45        for (int pixel = 0, row = 0, col = 0; pixel < h*w*4; pixel += 4) {
46
47            b[pixel + 0] = (byte)(red[row][col]);
48            b[pixel + 1] = (byte)((green[row][col]));
49            b[pixel + 2] = (byte)((blue[row][col]));
```

```
50          b[pixel + 3] = (byte)255;
51          col++;
52          if (col == w) {
53              col = 0;
54              row++;
55          }
56      }
57
58      Log.d(TAG,"Ends");
59
60      return b;
61  }
62
63  static public void convertToInt(byte[] b, int w, int h, float[][] red, float[][] green,
    float[][] blue) {
64      for (int pixel = 0, row = 0, col = 0; pixel < h*w*4; pixel += 4) {
65          red[row][col] = (b[pixel + 0] & 0xff);
66          green[row][col] = (b[pixel + 1] & 0xff);
67          blue[row][col] = (b[pixel + 2] & 0xff);
68          col++;
69          if (col == w) {
70              col = 0;
71              row++;
72          }
73      }
74  }
75
76
77  static public void processOne(float[][] color, int w, int h, float sigma, int r) {
78      // float temp;
79      for (int row = 0; row < h; row++ ){
80          for (int col = 0; col < w; col++){
81  //            color[row][col] = (int)(color[row][col]*gKernel(0,sigma)/unify(r,sigma));
82              color[row][col] = (color[row][col]*gKernel(0,sigma));
83
84              //      temp = color[row][col]*(gKernel(0,sigma)/unify(r,sigma));
85              for (int k = 1; k<=r; k++){
86  //                if(row+k<h){
87  //                    color[row][col] += (color[row+k][col]*(gKernel(k,sigma)));
88  //
89  //                }
90  //                if(row>=h){
91  //                    color[row][col] += (color[row-k][col]*(gKernel(-k,sigma)));
92  //
93  //                }
94
95                  if((row < k) || ( row + k >= h)){
96                      color[row][col] += 0;
97                  } else{
```

```
 98                     color[row][col] += (color[row+k][col]*(gKernel(k,sigma)));
 99                     color[row][col] += (color[row-k][col]*(gKernel(-k,sigma)));
100
101 //                     temp += (temp*(gKernel(k,sigma)/unify(r,sigma)));
102 //                     temp += (temp*(gKernel(-k,sigma)/unify(r,sigma)));
103 //                     color[row][col] = (int)temp;
104 //                     temp = 0;
105                 }
106
107             }
108
109         }
110     }
111
112 }
113
114
115
116 static public void processTwo(float[][] color, int w, int h, float sigma,int r) {
117 //   float temp ;
118     for (int row = 0; row < h; row++ ){
119       for (int col = 0; col < w; col++){
120         color[row][col] = color[row][col]*(gKernel(0,sigma));
121 //          temp = color[row][col]*(gKernel(0,sigma));
122
123           for (int k = 1; k<=r; k++){
124
125             if((col < k) || ( col + k >= w)){
126               color[row][col] += 0;
127             } else{
128               color[row][col] += (color[row][col+k]*(gKernel(k,sigma)));
129               color[row][col] += (color[row][col-k]*(gKernel(-k,sigma)));
130 //                 temp += (color[row][col]*(gKernel(k,sigma)));
131 //                 temp += (color[row][col]*(gKernel(-k,sigma)));
132 //                 color[row][col] = (int)temp;
133 //                 temp = 0;
134             }
135
136           }
137
138       }
139     }
140
141 }
142
143
144 static public float gKernel(int k, float t){
145
146     float g;
```

```
147         g = (float)Math.exp(-(k*k)/(2*(t*t)));
148         g = g*(float)1/(float)Math.sqrt(2*Math.PI*t*t);
149 
150         return g;
151     }
152 
153 //   public float unify(int r, float sigma){
154 //       float sum = gKernel(0,sigma);
155 //       for(int i = 1; i < 2*r+1; i++){
156 //           sum += gKernel(-r,sigma) + gKernel(r,sigma);
157 //       }
158 //
159 //       return sum;
160 //   }
161 }
```