```
1  package edu.asu.msrs.artcelerationlibrary;
2
3  import android.graphics.Bitmap;
4  import android.graphics.Color;
5  import android.util.Log;
6
7  /**
8   * Created by yitaochan on 11/27/16.
9   *
10  */
11
12 public class GaussianBlur {
13
14     String TAG = "GaussianBlur";
15
16
17     public Bitmap gblur(Bitmap bmp, int[] r, float[] sigma) {
18 //         int r = 5;
19 //         float sigma = 3f;
20         int redValue;
21         int greenValue;
22         int blueValue;
23         float red_f, green_f, blue_f;
24         Log.d(TAG, "GaussianBlur Starts");
25
26
27         //Log.d(TAG, "The red value is "+ String.valueOf(redValue));
28
29         float k1 = gKernel(1,3f);
30         float k2 = gKernel(0,3f);
31         float k3 = gKernel(-1,3f);
32         float tot = 2*k1+2*k2+2*k3;
33         float round = Math.round(tot);
34         Log.d(TAG, "The k1 value is "+ String.valueOf(k1));
35         Log.d(TAG, "The k2 value is "+ String.valueOf(k2));
36         Log.d(TAG, "The k3 value is "+ String.valueOf(k3));
37         Log.d(TAG, "The total value is "+ String.valueOf(tot));
38         Log.d(TAG, "The total value is "+ String.valueOf((int)tot));
39         Log.d(TAG, "The round value is "+ String.valueOf((int) round));
40
41
42         // Reminder: need to round instead of just int to convert the gaussian kernel;
43
44
45
46         for (int x = 300; x < 600; x++){
47             for(int y = 300; y < 600; y++){
48                 float temp = gKernel(0,sigma[0]);
49                 //Log.d(TAG,"temp "+String.valueOf(temp));
```

```
50              redValue = (int) (temp * Color.red(bmp.getPixel(x, y)));
51              greenValue = (int) (temp * Color.green(bmp.getPixel(x, y)));
52              blueValue = (int) (temp * Color.blue(bmp.getPixel(x, y)));
53              //Log.d(TAG,"Red blue green "+String.valueOf(redValue)+String.valueOf(
    blueValue)+String.valueOf(greenValue));
54
55              for (int k = 1; k <=r[0]; k++) {
56                  redValue += (int)(gKernel(-k,sigma[0])*Color.red(bmp.getPixel(x-k,y))
    +gKernel(k,sigma[0])*Color.red(bmp.getPixel(x+k,y)));
57                  greenValue += (int)(gKernel(-k,sigma[0])*Color.green(bmp.getPixel(x-
    k,y))+gKernel(k,sigma[0])*Color.green(bmp.getPixel(x+k,y)));
58                  blueValue += (int)(gKernel(-k,sigma[0])*Color.blue(bmp.getPixel(x-k,y
    ))+gKernel(k,sigma[0])*Color.blue(bmp.getPixel(x+k,y)));
59                //    Log.d(TAG,"RGB "+String.valueOf(redValue)+String.valueOf(
    greenValue)+String.valueOf(blueValue));
60              }
61              bmp.setPixel(x, y, Color.argb(255, redValue, greenValue, blueValue));
62          }
63
64
65
66          }
67
68      for (int x = 300; x < 600; x++){
69          for(int y = 300; y < 600; y++){
70
71              float temp = gKernel(0,sigma[0]);
72              redValue = (int) (temp * Color.red(bmp.getPixel(x, y)));
73              greenValue = (int) (temp * Color.green(bmp.getPixel(x, y)));
74              blueValue = (int) (temp * Color.blue(bmp.getPixel(x, y)));
75
76              for (int k = 1; k <=r[0]; k++){
77                  redValue += (int)(gKernel(-k,sigma[0])*Color.red(bmp.getPixel(x,y-k))+
    gKernel(k,sigma[0])*Color.red(bmp.getPixel(x,y+k)));
78                  greenValue += (int)(gKernel(-k,sigma[0])*Color.green(bmp.getPixel(x,y-k
    ))+gKernel(k,sigma[0])*Color.green(bmp.getPixel(x,y+k)));
79                  blueValue += (int)(gKernel(-k,sigma[0])*Color.blue(bmp.getPixel(x,y-k))+
    gKernel(k,sigma[0])*Color.blue(bmp.getPixel(x,y+k)));
80              }
81
82
83              bmp.setPixel(x, y, Color.argb(255, redValue, greenValue, blueValue));
84          }
85      }
86
87      Log.d(TAG, "GaussianBlur Ends");
88
89      return bmp;
90  }
```

```
 91
 92     public float gKernel(int k, float t){
 93
 94         float g;
 95         g = (float)Math.exp(-(k*k)/(2*(t*t)));
 96         g = g*(float)1/(float)Math.sqrt(2*Math.PI*t*t);
 97
 98         return g;
 99     }
100
101
102 }
103
```