```
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 11/30/16.The Sobel Edge filter transforms the input image first
9   * into a grayscale brightnessimage. Then use different edge filter to high light edges in the image.
10  * By applyingdifferent edge filters, we can obtain the gradients in horizontal and vertical direction.
11  * Once the pixel values are set, we can find the result
12  */
13
14 public class SobelEdge {
15
16
17     int w ;
18     int h ;
19     int[][] redcontainer;
20     int[][] greencontainer;
21     int[][] bluecontainer;
22     int[][] gray;
23     int[][] sx;
24     int[][] grx;
25     int[][] gry;
26     int[][] gr;
27     int[][] sy;
28     String TAG = "SobelEdge";
29
30     //Function: main method in this class which calls all the other methods to implement the sobel edge transform
31     //Input: bitmap object
32     //Output: bitmap object
33     public Bitmap sEdge (Bitmap bmp, int[] args1){
34
35         h = bmp.getHeight();
36         w = bmp.getWidth();
37         redcontainer = new int[w][h];
38         greencontainer = new int[w][h];
39         bluecontainer = new int[w][h];
40         gray = new int[w][h];
41         sx = new int[][]{{-1,0,1},{-2,0,2},{-1,0,1}};
42         sy = new int[][]{{-1,-2,-1},{0,0,0},{1,2,1}};
43
44         grx = new int[w][h];
45         gry = new int[w][h];
```

```
46        gr = new int[w][h];
47
48        Log.d(TAG, "Starts");
49        getColorValue(bmp);
50        grayScale();
51        setGrayScale(bmp);
52        gradient(bmp, args1[0]);
53
54        Log.d(TAG, "Ends");
55
56        return bmp;
57    }
58
59    //Function: extract color values from the bitmap
60    //Input: bitmap object
61    //Output: bitmap object
62
63    public void getColorValue(Bitmap bmp){
64        for ( int x = 0; x < w; x++){
65            for ( int y = 0; y < h; y++){
66                redcontainer [x][y] = Color.red(bmp.getPixel(x, y));
67                greencontainer [x][y] = Color.green(bmp.getPixel(x, y));
68                bluecontainer [x][y] = Color.blue(bmp.getPixel(x, y));
69
70            }
71        }
72    }
73
74    //Function: convert the color array into grayscale values
75    //Input: void
76    //Output: null
77    public void grayScale(){
78
79        for (int x = 0; x < w; x++){
80            for (int y = 0; y < h; y++){
81                redcontainer[x][y] = (int)(0.2989*redcontainer[x][y]);
82                greencontainer[x][y] = (int)(0.5870*greencontainer[x][y]);
83                bluecontainer[x][y] = (int)(0.1140*bluecontainer[x][y]);
84                gray[x][y] = redcontainer[x][y] + greencontainer[x][y] + bluecontainer[x][y];
85
86            }
87        }
88
89    }
90
91
92    //Function: sets the grayscale values into the image bitmap object
93    //Input: bitmap object
94    //Output: null
```

```
 95
 96     public void setGrayScale(Bitmap bmp){
 97         for (int x = 0; x < w; x++){
 98             for (int y = 0; y < h; y++){
 99                 // bmp.setPixel(x, y, Color.argb(255, redcontainer[x][y], greencontainer[x][y],
    bluecontainer[x][y]));
100                 bmp.setPixel(x, y, Color.argb(255, gray[x][y], gray[x][y], gray[x][y]));
101
102             }
103
104         }
105
106     }
107
108     //Function: sets the grx values into the image bitmap object
109     //Input: bitmap object, input args (0: Sx, 1: Sy)
110     //Output: null
111
112
113     public void gradient(Bitmap bmp, int args){
114         switch (args){
115             case 0:
116                 getGrx();
117                 for (int x = 1; x < w-1; x ++){
118                     for (int y = 1; y < h-1; y++){
119
120                         bmp.setPixel(x, y, Color.argb(255, grx[x][y], grx[x][y], grx[x][y]));
121 //
122                     }
123                 }
124
125                 break;
126             case 1:
127                 getGry();
128                 for (int x = 1; x < w-1; x ++){
129                     for (int y = 1; y < h-1; y++){
130
131                         bmp.setPixel(x, y, Color.argb(255, gry[x][y], gry[x][y], gry[x][y]));
132                     }
133                 }
134                 break;
135             case 2:
136                 getGrx();
137                 getGry();
138                 for (int x = 1; x < w-1; x ++) {
139                     for (int y = 1; y < h - 1; y++) {
140
141                         bmp.setPixel(x, y, Color.argb(255, (int)Math.sqrt(grx[x][y]*grx[x][y]),
142                                 (int)Math.sqrt(grx[x][y]*grx[x][y]),
```

```
143                         (int)Math.sqrt(grx[x][y]*grx[x][y])));
144
145                 }
146
147             }
148
149             break;
150         default:
151             break;
152     }
153
154
155 }
156
157 //Function: finds the grx values
158 //Input: void
159 //Output: null
160
161 public void getGrx(){
162     for (int x = 1; x < w-1; x ++){
163         for (int y = 1; y < h-1; y++){
164             grx[x][y] = (-1)*gray[x-1][y-1] + (0)*gray[x][y-1] + (1)*gray[x+1][y-1]+
165                     (-2)*gray[x-1][y] + (0)*gray[x][y] + (2)*gray[x+1][y]+
166                     (-1)*gray[x-1][y+1] + (0)*gray[x][y+1]+ (1)*gray[x+1][y+1];
167             if(grx[x][y] < 0) {
168                 grx[x][y] = 0;
169             } else {
170
171             }
172         }
173     }
174
175
176 }
177
178 //Function: finds the gry values
179 //Input: void
180 //Output: null
181
182 public void getGry(){
183
184     for (int x = 1; x < w-1; x ++){
185         for (int y = 1; y < h-1; y++){
186                 grx[x][y] = (-1)*gray[x-1][y-1] + (-2)*gray[x][y-1] + (1)*gray[x+1][y-1]
    +
187                         (0)*gray[x-1][y] + (0)*gray[x][y] + (0)*gray[x+1][y]+
188                         (1)*gray[x-1][y+1] + (2)*gray[x][y+1]+ (1)*gray[x+1][y+1];
189                 if(grx[x][y] < 0) {
190                     grx[x][y] = 0;
```

```
191                     } else {
192
193                     }
194                 }
195             }
196
197
198
199
200         }
201     }
202
203
```