Mobile Image Processing on the Google Phone with the Android Operating System

by Michael T. Wells

1.0 Introduction

This report analyzes the computation time of several common image processing routines on the HTC G1, also known as the Google Phone released in October 2008 as the first official hardware platform with the Android Operating System.

1.1 Motivation

Image processing on mobile phones is a new and exciting field with many challenges due to limited hardware and connectivity. Phones with cameras, powerful CPUs, and memory storage devices are becoming increasingly common. The need for benchmarking basic image processing routines such as: addition, convolution, thresholding and edge detection is important for comparison of systems. With this information developers and researchers can design complex computer vision and image processing applications while being aware of the current state of the art limitations and bottlenecks on mobile phones.

1.2 Background

For the sake of this project the following summary found in *Figure 1* will be referenced to provide context to the steps in a typical computer vision application.

Figure 1

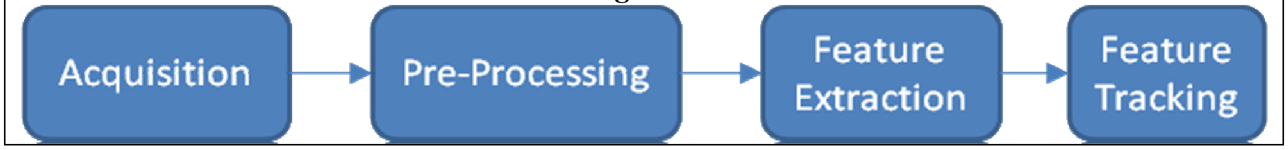


Image *Acquisition* refers to the capturing of image data by a particular sensor or data repository. Once the image data is acquired, *Pre-Processing* often includes rendering the acquired data to a format that can be handled by a set of algorithms for *Feature Extraction* that transform sub-image data to information which are often in turn maintained over time to provide temporal information.

1.2.1 Examples

There are many software applications that focus on *Acquisition* and *Pre-Processing* primarily. These include applications that perform image editing and enhancement such as Adobe Photoshop. Other applications may include *Feature Extraction* in order to make spatial decisions or notify a user of an event such as an augmented reality device. Finally, these extracted features are often tracked over time to render some temporal statistics to make decisions or notify a user of an event such as in early warning or surveillance devices.

1.3 Goals

The goal of this project is to focus on *Image Acquisition* and *Pre-Processing* through implementing image addition, convolution, thresholding and edge detection on the HTC G1 mobile phone using the available Software Development Kit (SDK). Once these image processing routines are implemented the time it takes to perform these operations will be measured on various sample images (see Results and Appendix). Through this effort to quantify processing times for common routines in Image Processing, this information can be used to make decisions on the feasibility of implementing Feature Extraction and Tracking applications.

2.0 Approach and Challenges

The Android operating system is preferable for benchmarking due to its recent growth in popularity with varying hardware manufactures e.g. HTC, Motorola, and Samsung. The Android operating system is supported and a part of the Open Handset Alliance. This alliance positions key manufacturers, cellular providers and the Android operating system in a collaborative environment which has caused large growth since October 2008 when the first Android mobile phone was released.

Using the HTC G1 as the hardware for testing is advantageous because it is the first phone that was officially released with the Android operating system and is therefore a good platform to benchmark and begin developing image processing applications. The capabilities for this hardware include still images at a resolution of 1536 x 2048 and video at a resolution of 320 x 240. By benchmarking key processing functions in Acquisition and Pre-Processing the design of complex image processing and computer vision applications can be designed with this information in mind.

The particular challenges when implementing on the HTC G1 with the Android OS include architecting software and optimizing code for

a. Memory limitations
b. CPU limitations
c. Image Quality limitations

In the software development described in the subsequent sections of this paper items (b) and (c) where the least limiting factor. Item (a) was the most difficult to work around as discussed along with alternatives in section *5.0 Discussion*. Google provides most of the documentation needed to development software applications on their developer web page [7], as well as forums to discuss your challenges and problems. Documentation and online tutorials are becoming increasingly more common as more developers begin to learn the ins and outs while coding in the Android Operating System.

3.0 Previous Work

The bulk of the results and analysis of this report are based primarily on motivation from paper [5]. In this paper six image processing benchmarks are analyzed: Addition, Blending, Convolution, Dot Product, Scaling and Double Threshold. I implemented Addition, Convolution, Single Threshold and Sobel edge detection in my application with the goal to benchmark processing time.

To accomplish any significant image processing application, feature extraction is important and widely a part of many computer vision systems as discussed in section *1.2 Background*. In particular feature extraction includes threshold, image addition and convolution. The papers [1, 2] cover the SURF method for feature extraction. In particular this paper provided me with background on types of invariant features that are quick and easy to compute and provide meaningful information about the image. These papers also give a high level description of the steps in the SURF algorithm along with speed improvements. These papers could be used in conjunction with the timing results obtained in section *4.0 Results* to determine how practical it would be to implement on the HTC G1 as discussed in section *5.0 Discussion and Future Work*.

Once features are extracted in a timely manner these features are often tracked in many computer vision systems. These papers [3, 4] provided details on implementing tracking methods on a mobile device which are used with the results of section *4.0 Results* to hypothesize the feasibility of implementing *Feature Tracking* as discussed in section *1.2 Background*. This paper gives frame rates and mobile device CPU clock speeds and other useful statistics and provides as a good reference point for comparing results.

4.0 Software

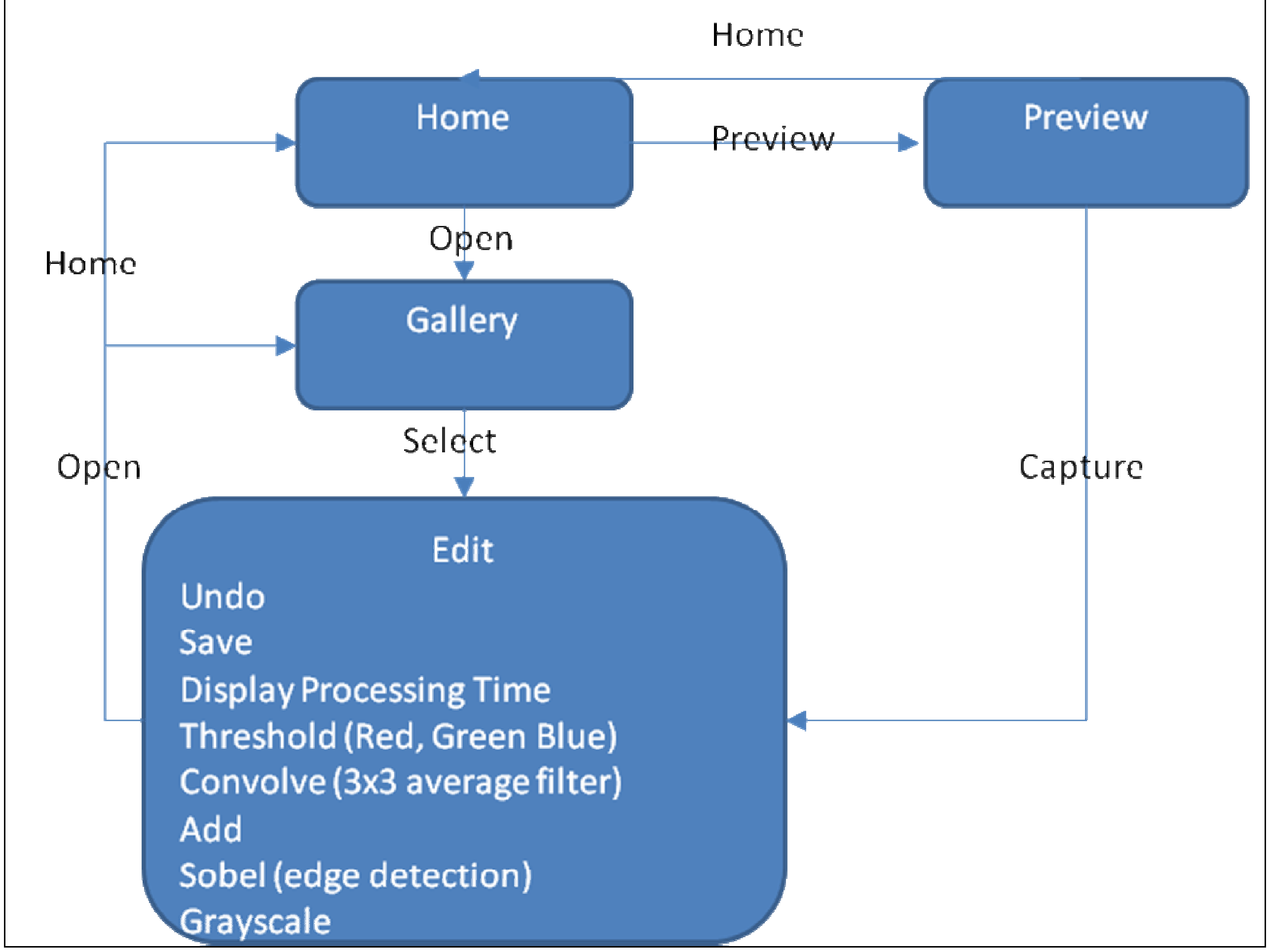
As mentioned in section *1.0 Introduction* the application produced in this work covers Image *Acquisition* and *Pre-Processing* and the goal of the application is to acquire and decode images to byte data that can be processed keeping in mind the limitations discussed in section *2.0 Approach and Challenges*. The software must measure the processing time of processing an individual image independent of decoding the image and displaying it.

The image processing library called JJIL (John's Java Image Library) was used for its image decoding functions contained in the class definitions *RgbImage.java* and *RgbImageAndroid.java* as found in section *10.0 Appendix*. The functionality in this code converts the raw byte array to a raster RGB image which was surprisingly difficult to find. In section 5.0 Discussion and Future Work, other options to the JJIL are mentioned that require less memory and overhead.

4.1 Architecture

Below in *Figure 1* is an overview of the software architecture that is divided into boxes that represent portions of code called an *Activity*. A specific activity communicates through an *Intent*, which are the lines relating each activity in *Figure 2*. Inside each activity are functions that operate on each particular activity. See [6, 7] for definitions and more detail on these software components, *Activity* and *Intent*, which are the fundamental components of producing an Android application.

Figure 2



4.1.1 Activity Descriptions

The Home Activity is the first screen in the application and the user can choose to acquire images through the file system in the *Gallery* activity on the phone or through the camera *Preview* activity. The *Gallery* activity is built into the Operating System and only required coding of the intent to retrieve image files. The *Preview* activity contains code to preview images through the camera before the *Capture* intent is sent upon pressing the image capture button. Upon *Capture* or *Open* each sends a specific intent to the *Edit* activity where the image processing occurs. For the code implementation of each activity, intent and function see section *10.0 Appendix* for the full source.

4.1.2 Image Size Problem

The images acquired from the Gallery and Preview activities where originally at full image resolution 1536 x 2048. However, in testing the application, as described in section 5.0 Testing, it would crash upon image acquisition. The Edit activity contains three static local RgbImage objects defined in JJIL as described in section *4.0 Software*. One static instance for the current acquired image, another holds the previous image for the undo operation and the final image is stored for the add function. Apparently the storage of 3 images at 1536 x 2048 in memory is too much. My quick solution was to sub-sample by a factor of 4 in the Preview and Gallery activities. The more permanent solution for this problem would be to maintain only the raw data for one image in memory and store the others in a file or database on the phone and load only data in memory needed for a specific function.

4.2 Measuring Processing Time

The processing times measured in the application occur within the individual functions listed under the Edit activity. I used the built in java package System to get the system time. I grab a time stamp at the beginning of processing and then a time stamp at the end of processing and take the difference to obtain the total processing time.

It would also be useful work to determine the time to acquire and display an image in the Preview activity but this is not covered in the scope of this project. See section *5.0 Discussion and Future Work* for more on this.

5.0 Testing

The software application was tested with a variety of images run 20 times on the same inputs and an average computation time is calculated. The application was also loaded onto the Android Market. The Android Market is run by Google and makes the application accessible to anyone that has a cellular connection on a device running the Android Operating System. With the application loaded onto the Android Market for only 3 days it had been installed on 135 phones which is a great test environment of different hardware and user configurations. The Market also provides users the opportunity to comment on the application and I received valuable feedback which I used to improve my application.

An important note about the Android Operating System is how it manages resources and applications when a program is in the foreground or the background. The Android Operating System supports the running of simultaneous applications and depending on the priority of the application processing times may be impacted. For instance, if an incoming phone call occurs while an image is being processed the call takes precedence over other applications. See [6,7] for more detail on this subject.

6.0 Results

Table 1

	Average Run Time [seconds] of 20 Runs			
Image Name	Sobel	Addition	3x3 Convolution	Single Threshold
Outdoor.jpeg (512x384)	0.145111	0.020061	0.143640	0.003059
House.jpeg (512x384)	0.154149	0.022400	0.147198	0.002951
Face.jpeg (512x384)	0.144820	0.022368	0.144240	0.002965
Keyboard.jpeg (160x120)	0.015326	0.016865	0.015319	0.0028535
Lamp.jpeg (240x320)	0.054860	0.008577	0.054825	0.001237
Concertina.jpeg (320x240)	0.058465	0.008973	0.05700	0.001358

7.0 Discussion

The average run times prove that the hardware and software can be used to perform basic image processing applications. Consider a basic image processing application that involves thresholding of a sobel image to be used in a Hough Transform for example. If you look at the processing time results for Face.jpeg in section *6.0 Results* you would obtain on average 0.144820[sec] + 0.002965[sec] = 0.147785[sec] to obtain a black/white image of edges for the Hough transform.

Key lessons learned from this work include:

1. Optimizing for memory usage by using file storage or a database.
2. Creating one activity instead of passing multiple intents to save on application overhead.

8.0 Future Work

The next step of for application development on this mobile platform is to implement and test more complex image processing applications that contain aggregates of the benchmarked image processing routines like the Hough Transform or the SURF algorithm [1,2]. A similar runtime analysis to what was performed here in this paper would be performed on these applications.

Some future work would include investigating the time needed to grab a preview image with the camera and overlay data. This is a fundamental step for any augmented reality system and benchmarking that process would be important.

Extending the results to frames per second and comparing to actual video processing run times would also be an important benchmark.

Since more new hardware platforms for the android operating system are being released every month it will also be important to test on these new platforms as they are made available. The Motorola Droid, just released in November 2009 contains a 5.0 mega pixel camera and a flash for night shots which greatly extends the image processing possibilities pas the G1.

9.0 References

[1] Chen, Wei-Chao and Xiong, Yingen and Gao, Jiang and Gelfand, Natasha and Grzeszczuk, Radek. "Efficient Extraction of Robust Image Features on Mobile Devices." In Proc. ISMAR 2007, 2007.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In ECCV (1), pages 404-417, 2006.

[3] Wagner, Daniel and Langlotz, Tobias and Schmalsteig, Dieter."Robust and Unobtrusive Marker Tracking on Mobile Phones," IEEE International Symposium on Mixed and Augmented Reality 2008 15-18 September. Cambridge UK.

[4] Wagner, Daniel and Reitmayr, Gerhard and Mulloni, Alessandro and Drummond, Tom and Schmalsteig, Dieter. "Pose Tracking from Natural Features on Mobile Phones." IEEE International Symposium on Mixed and Augmented Reality 2008 15-18 September, Cambridge UK.

[5] Ranganathan, P., Adve, S., and Jouppi, N. Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions. IEEE 1999.

[6] Meier, Reto. Professional Android Application Development. Wrox Publishing. Nov. 2008.

[7] http://developer.android.com/index.html

10.0 Appendix

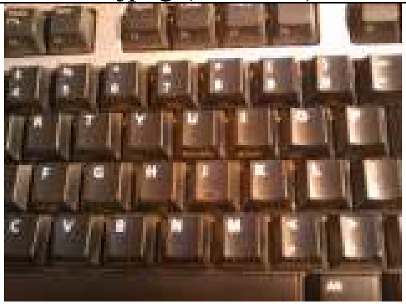
Figure A.1 ***(Image Data)***		
Outdoor.jpeg (512x384)	House.jpeg (512x384)	Face.jpeg (512x384)
		
Keyboard.jpeg (160x120)	Lamp.jpeg (240x320)	Concertina.jpeg (320x240)

Table A.2.1 ***(House.jpeg Results)***	
Sobel	Addition

Convolution

Threshold

Table A.2.2				
Run	Threshold	Addition	Sobel	Convolution
	House	House + Face	House	House
01	0.00293	0.01955	0.17947	0.14054
02	0.00291	0.02004	0.17155	0.14593
03	0.00299	0.02144	0.14541	0.14631
04	0.00292	0.02346	0.18314	0.16557
05	0.00287	0.02039	0.15241	0.14458
06	0.00292	0.01978	0.17594	0.14291
07	0.00287	0.02037	0.14835	0.14324
08	0.00312	0.02385	0.15215	0.14498
09	0.00298	0.02352	0.14615	0.14427
10	0.00314	0.02198	0.1431	0.14337
11	0.00294	0.02295	0.17433	0.14527
12	0.00293	0.02559	0.15057	0.14384
13	0.00299	0.02447	0.14122	0.14885
14	0.00296	0.02385	0.14538	0.14532
15	0.00292	0.02244	0.1554	0.14832
16	0.00293	0.0228	0.14161	0.14401
17	0.00287	0.02386	0.14521	0.14633
18	0.00295	0.02192	0.14566	0.16903

19	0.00293	0.02287	0.13832	0.14626
20	0.00295	0.02287	0.14761	0.14503
AVG	**0.002951**	**0.0224**	**0.154149**	**0.147198**

Table A.3.1
(Lamp.jpeg Results)

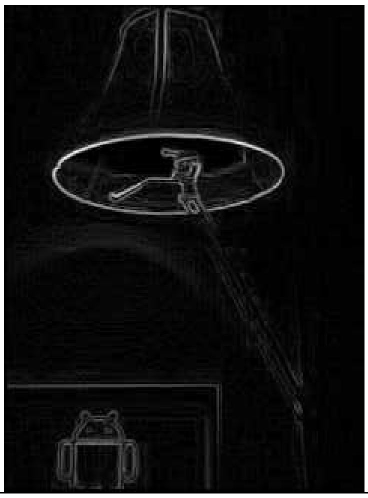
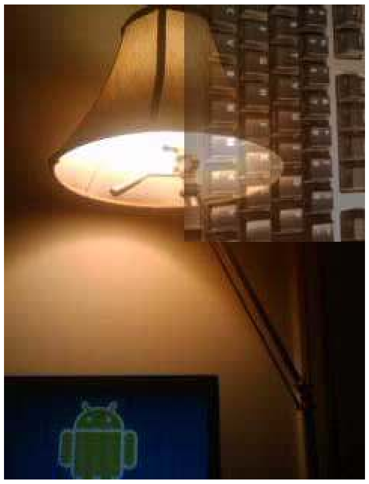
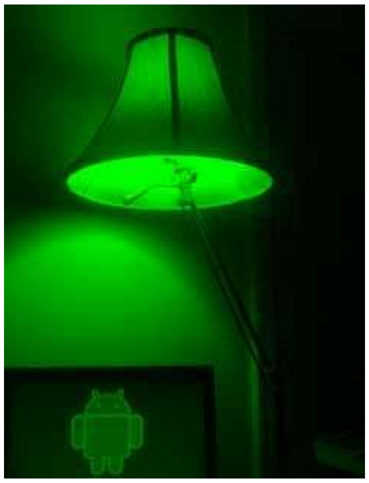
Sobel	Addition
	
Convolution	**Threshold**
	

Table A.3.2

Run	Threshold	Addition	Sobel	Convolution
	Lamp	Lamp+Keyboard	Lamp	Lamp
01	0.00118	0.00995	0.05421	0.05577
02	0.00116	0.00767	0.054	0.05513
03	0.00127	0.00769	0.054	0.05522
04	0.00123	0.00942	0.05375	0.05495
05	0.00125	0.00789	0.05497	0.05396

06	0.00121	0.00845	0.06562	0.05446
07	0.00136	0.00966	0.05407	0.05527
08	0.00114	0.00862	0.05584	0.0545
09	0.00122	0.00861	0.05375	0.05466
10	0.00114	0.00902	0.05486	0.05511
11	0.00121	0.00871	0.05662	0.05491
12	0.0012	0.00773	0.05329	0.05594
13	0.00131	0.00548	0.0543	0.05413
14	0.00119	0.00971	0.05483	0.0549
15	0.00121	0.00982	0.05337	0.05503
16	0.00123	0.00709	0.05605	0.05491
17	0.00125	0.01017	0.05005	0.05506
18	0.00131	0.00703	0.0533	0.05403
19	0.00141	0.0104	0.05588	0.05429
20	0.00126	0.00842	0.05443	0.05427
AVG	**0.001237**	**0.008577**	**0.05486**	**0.054825**

Table A.4.1
(Face.jpeg Results)

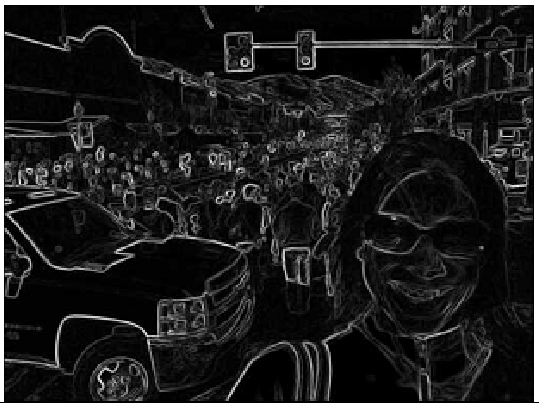
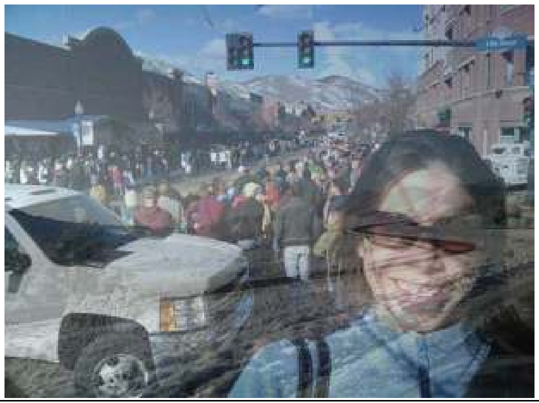
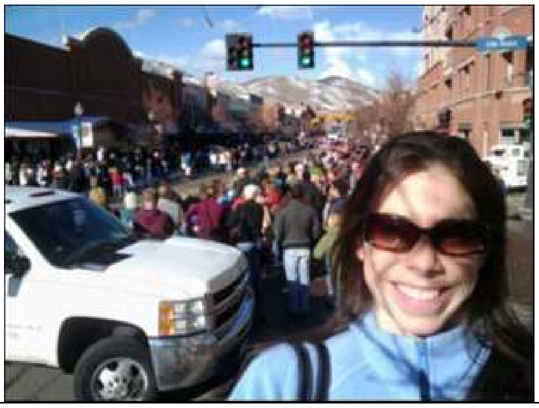
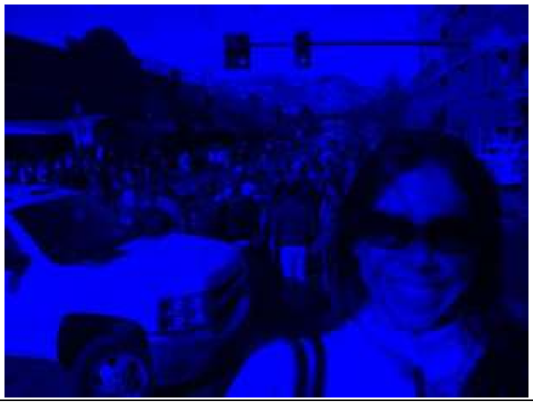
Sobel	Addition
	
Convolution	Threshold
	

Table A.4.2

Run	Threshold	Addition	Sobel	Convolution
	Face	Face + Landscape	Face	Face
01	0.00293	0.02423	0.1501	0.14134
02	0.00298	0.02207	0.14437	0.16277
03	0.0032	0.02044	0.1411	0.14247
04	0.00315	0.02442	0.14311	0.13963
05	0.0029	0.02151	0.14993	0.16365
06	0.00292	0.0252	0.14384	0.14024
07	0.00289	0.02042	0.15015	0.13944
08	0.00292	0.0222	0.1426	0.13924
09	0.00295	0.02143	0.14661	0.14381
10	0.00303	0.02317	0.14248	0.13934
11	0.00309	0.02628	0.14382	0.14329
12	0.00289	0.02985	0.14713	0.14912
13	0.00298	0.02285	0.14731	0.14428
14	0.00299	0.0269	0.14117	0.13996
15	0.00292	0.0185	0.14174	0.14041
16	0.00289	0.01817	0.14308	0.14138
17	0.00292	0.01887	0.14963	0.15277
18	0.00292	0.02057	0.14258	0.14179
19	0.00287	0.01997	0.14246	0.14031
20	0.00301	0.02031	0.14318	0.13956
AVG	**0.0029675**	**0.022368**	**0.14482**	**0.14424**

Table A.4.1
(Keyboard.jpeg Results)

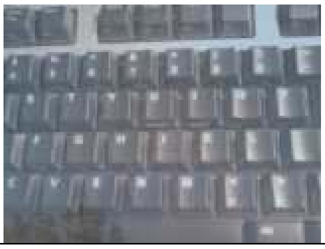
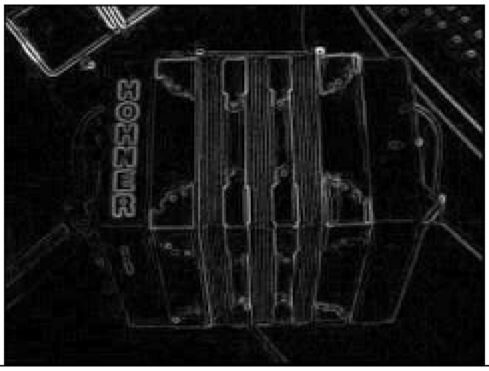
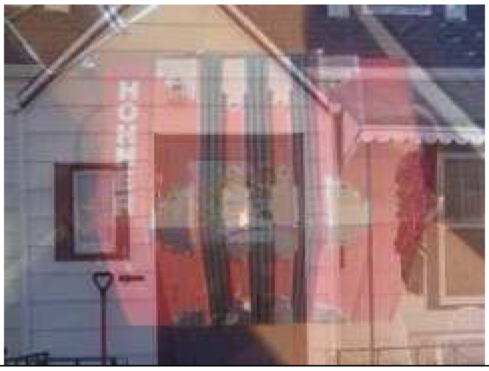
Sobel	Addition
	
Convolution	Threshold

Table A.4.2

Run	Threshold	Addition	Sobel	Convolution
	Keyboard	Keyboard+Outdoor	Keyboard	Keyboard
01	0.000294	0.01733	0.01609	0.01633
02	0.000340	0.01680	0.01546	0.01529
03	0.000290	0.01544	0.01539	0.01550
04	0.000310	0.01664	0.01504	0.01482
05	0.000291	0.01690	0.01704	0.01642
06	0.000300	0.01783	0.01436	0.01532
07	0.000281	0.01591	0.01458	0.01663
08	0.000251	0.01699	0.01477	0.01456
09	0.000295	0.01721	0.01523	0.01505
10	0.000311	0.01696	0.01514	0.01526
11	0.000300	0.01550	0.01463	0.01475
12	0.000294	0.01742	0.01475	0.01497
13	0.000318	0.01680	0.01556	0.01561
14	0.000279	0.01688	0.01602	0.01462
15	0.000287	0.01792	0.01511	0.01483
16	0.000220	0.01688	0.01535	0.01467
17	0.000292	0.01794	0.01557	0.01528
18	0.000202	0.01597	0.01490	0.01653
19	0.000247	0.01720	0.01492	0.01467
20	0.000305	0.01677	0.01661	0.01526
AVG	**0.0028535**	**0.016865**	**0.015326**	**0.015319**

Table A.5.1 ***(Concertina.jpeg Results)***	
Sobel 	Addition 
Convolution	Threshold

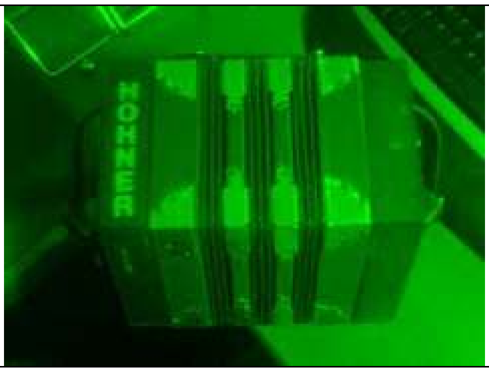
	

Table A.5.2				
Run	Threshold	Addition	Sobel	Convolution
	Concertina	Concertina+House	Concertina	Concertina
01	0.00132	0.00987	0.05669	0.05586
02	0.00145	0.00988	0.05611	0.05612
03	0.00114	0.00843	0.06163	0.05642
04	0.00146	0.00786	0.05860	0.05686
05	0.00123	0.00880	0.05160	0.05628
06	0.00144	0.00952	0.05654	0.05554
07	0.00147	0.00960	0.06110	0.06380
08	0.00128	0.00914	0.05602	0.05802
09	0.00140	0.00974	0.06115	0.05716
10	0.00127	0.00998	0.05672	0.05670
11	0.00141	0.00963	0.06890	0.05552
12	0.00145	0.00902	0.05606	0.05794
13	0.00145	0.00956	0.06772	0.05701
14	0.00135	0.00882	0.05621	0.05655
15	0.00126	0.00856	0.05821	0.05723
16	0.00115	0.00884	0.05719	0.05626
17	0.00147	0.00782	0.05634	0.05613
18	0.00123	0.00766	0.05512	0.05714
19	0.00148	0.00892	0.06022	0.05774
20	0.00145	0.00780	0.05716	0.05571
AVG	**0.001358**	**0.008973**	**0.058465**	**0.05700**

Table A.6.1 *(Outdoor.jpeg Results)*	
Sobel	Addition

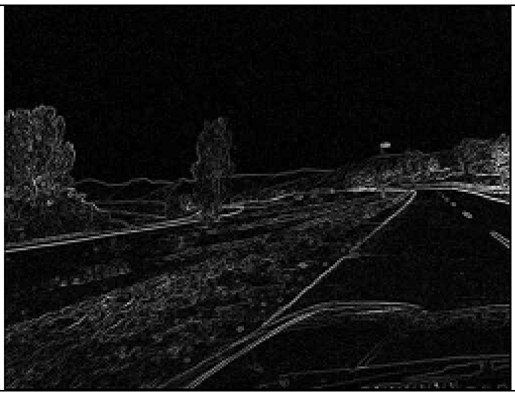
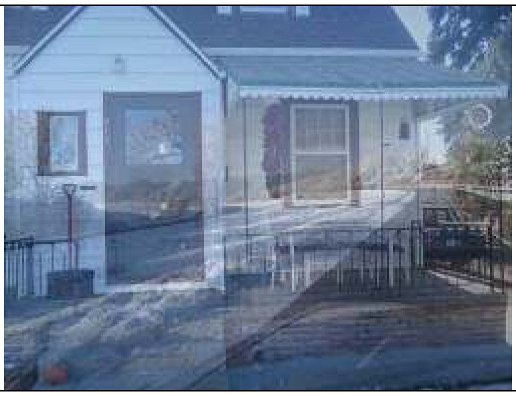
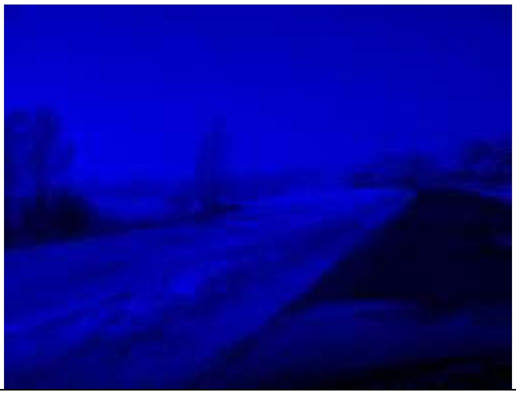
	
Convolution 	Threshold 

Table A.6.2				
Run	Threshold	Addition	Sobel	Convolution
	Outdoor	Outdoor+House	Outdoor	Outdoor
01	0.00293	0.01913	0.14618	0.14187
02	0.00323	0.01724	0.14023	0.14233
03	0.00307	0.02203	0.14550	0.14567
04	0.00321	0.02113	0.14622	0.13187
05	0.00292	0.01944	0.14863	0.14453
06	0.00327	0.01923	0.14744	0.14988
07	0.00289	0.01977	0.14549	0.14870
08	0.00305	0.02120	0.14543	0.14522
09	0.00292	0.01966	0.14654	0.13990
10	0.00315	0.02269	0.14230	0.14144
11	0.00298	0.01929	0.14561	0.13952
12	0.00292	0.01837	0.14432	0.14079
13	0.00293	0.01844	0.14409	0.15110
14	0.00331	0.02210	0.14331	0.13965
15	0.00289	0.02050	0.14289	0.14114
16	0.00309	0.01915	0.14293	0.14150
17	0.00309	0.02076	0.14682	0.14396
18	0.00306	0.01841	0.14504	0.14157

19	0.00305	0.02055	0.14591	0.15926
20	0.00322	0.02213	0.14733	0.14290
AVG	**0.003059**	**0.020061**	**0.145111**	**0.14364**

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.wellsmt.ImageDetect"
      android:versionCode="7"
      android:versionName="1.6">
       <uses-sdk android:minSdkVersion="4"></uses-sdk>
    <uses-permission android:name="android.permission.CAMERA" />
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name"
                 android:debuggable="false">
        <activity android:name=".Progress"
                  android:label="@string/app_name"
                  android:configChanges="orientation|keyboardHidden"
                  android:screenOrientation="landscape"
                  >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".HomeScreen"
                        android:label="@string/app_name"
                        android:configChanges="orientation|keyboardHidden"
                  android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="com.wellsmt.ImageDetect.HomeScreen" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".ImageDetect"
                  android:label="@string/app_name"
                  android:configChanges="orientation|keyboardHidden"
                  android:screenOrientation="landscape"
                  android:theme="@android:style/Theme.NoTitleBar">
            <intent-filter>
                <action android:name="com.wellsmt.ImageDetect.Preview" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".OpenImage"
                        android:label="@string/app_name"
                        android:configChanges="orientation|keyboardHidden"
                        android:screenOrientation="landscape">
               <intent-filter>
                       <action android:name="com.wellsmt.ImageDetect.OpenImage" />
                       <category android:name="android.intent.category.DEFAULT" />
               </intent-filter>
               <intent-filter>
                       <action android:name="android.intent.action.GET_CONTENT" />
                       <category android:name="android.intent.category.DEFAULT" />
                       <data android:mimeType="image/*"/>
               </intent-filter>
        </activity>
        <activity android:name=".ModImage"
                  android:label="@string/app_name"
                  android:configChanges="orientation|keyboardHidden"
                  android:screenOrientation="landscape"
                  android:theme="@android:style/Theme.NoTitleBar">
            <intent-filter>
                <action android:name="com.wellsmt.ImageDetect.ModImage" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Progress.java

```
package com.wellsmt.ImageDetect;

import com.wellsmt.ImageDetect.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class Progress extends Activity{
        public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.progress);
         try {
                Thread t = new Thread() {
                        public void run() {
                                startActivity(new Intent("com.wellsmt.ImageDetect.HomeScreen"));

                        }
                };
                 t.start();
        } catch (Exception ex) {
                ex.printStackTrace();
            System.err.print("Exception: " + ex.toString()); //$NON-NLS-1$ //$NON-NLS-2$
         }
        }
}
```

HomeScreen.java

```
package com.wellsmt.ImageDetect;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Toast;

public class HomeScreen extends Activity{
        public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.home);

         final ImageView mImageView = (ImageView) findViewById(R.id.image);
         mImageView.setOnClickListener(new View.OnClickListener() {
             public void onClick(View v) {
                Toast.makeText(HomeScreen.this, "2D Gaussian...", Toast.LENGTH_LONG);
             }
         });

         final ImageButton buttonCapture = (ImageButton) findViewById(R.id.button_capture);
         buttonCapture.setOnClickListener(new View.OnClickListener() {
             public void onClick(View v) {
                startActivity(new Intent("com.wellsmt.ImageDetect.Preview"));
             }
         });

         final ImageButton buttonOpen = (ImageButton) findViewById(R.id.button_open);
         buttonOpen.setOnClickListener(new View.OnClickListener() {
             public void onClick(View v) {
                 // Perform action on click
                startActivity(new Intent("com.wellsmt.ImageDetect.OpenImage"));
             }
         });
        }
}
```

OpenImage.java

```
package com.wellsmt.ImageDetect;

import java.io.InputStream;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;

public class OpenImage extends Activity{

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.image);

                Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
                photoPickerIntent.setType("image/*");

                startActivityForResult(photoPickerIntent, 1);
        }

        // Bitmap bytes have to be created via a direct memory copy of the bitmap

    protected void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        super.onActivityResult(requestCode, resultCode, data);

        if (resultCode == RESULT_OK)
        {
                Uri chosenImageUri = data.getData();

                if( chosenImageUri != null)
                {
                                try
                                {
                                        InputStream photoStream =
getContentResolver().openInputStream(chosenImageUri);
                                        BitmapFactory.Options opts = new BitmapFactory.Options();
                                        opts.inSampleSize = 4;


                                        Bitmap mBitmap = BitmapFactory.decodeStream(photoStream,
null, opts);
                                        if( mBitmap != null)
                                        {
                                                ModImage.setJpegData(mBitmap);
                                        }

                                        photoStream.close();

                                        startActivity(new
Intent("com.wellsmt.ImageDetect.ModImage"));
                                }
                                catch (Exception e)
                                {
                                        e.printStackTrace();
                                }
                }
                }
    }

        }
```

ImageDetect.java

```java
package com.wellsmt.ImageDetect;

import java.io.IOException;
import android.app.Activity;
import android.content.Intent;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.os.Bundle;

import android.view.KeyEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.WindowManager;
import android.widget.Toast;

public class ImageDetect extends Activity implements SurfaceHolder.Callback,
View.OnClickListener{

        final int RESTART_PREVIEW = 1;
        final int PROGRESS = 2;
        final int RESTART_PREVIEW2 = 3;


        private boolean boolCaptureOnFocus = false;
        private boolean boolFocusButtonPressed = false;
        private boolean boolFocused = false;
        private boolean boolFocusing = false;
        private boolean boolPreviewing = false;
        private Camera camera = null;
    private int nPreviewWidth, nPreviewHeight;
    private SurfaceView preview = null;
    private SurfaceHolder surfaceHolder = null;


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Toast.makeText(this, "Welcome to ImageProx", Toast.LENGTH_LONG).show();

               setContentView(R.layout.capture);
               preview = (SurfaceView) findViewById(R.id.Preview);
        SurfaceHolder s = preview.getHolder();
        s.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        s.addCallback(this);
    }

    private final class AutoFocusCallback implements Camera.AutoFocusCallback {
        public void onAutoFocus(boolean focused, Camera camera) {
            boolFocusing = false;
            boolFocused = focused;
            if (focused) {
                if (boolCaptureOnFocus) {
                    Camera.Parameters parameters = camera.getParameters();
                    parameters.set("jpeg-quality", 75);
                    parameters.setPictureSize(320, 240);
                    camera.setParameters(parameters);
                    camera.takePicture(null, null, new JpegPictureCallback());
                    clearFocus();
                }
                boolCaptureOnFocus = false;
            }
        }
    };

    private final class JpegPictureCallback implements PictureCallback {
        public void onPictureTaken(byte [] jpegData, android.hardware.Camera camera) {
               ModImage.setJpegData(jpegData);
               startActivity(new Intent("com.wellsmt.ImageDetect.ModImage"));
               stopPreview();
        }
    };

    private void autoFocus() {
        if (!this.boolFocusing) {
            if (this.camera != null) {
               this.boolFocusing = true;
                this.boolFocused = false;
                this.camera.autoFocus(new AutoFocusCallback());
            }
        }

    }

    private void clearFocus() {
        this.boolFocusButtonPressed = false;
        this.boolFocused = false;
        this.boolFocusing = false;
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        switch (keyCode) {
            case KeyEvent.KEYCODE_CAMERA:
            case KeyEvent.KEYCODE_DPAD_CENTER:
                if (event.getRepeatCount() == 0) {
                    if (this.boolFocused || !this.boolPreviewing) {
                        clearFocus();
                    } else {
                        this.boolCaptureOnFocus = true;
                    }
                    if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER && !this.boolFocusButtonPressed)
{
                        autoFocus();
                    }
                }
                return true;
            case KeyEvent.KEYCODE_FOCUS:
                this.boolFocusButtonPressed = true;
                if (event.getRepeatCount() == 0) {
                    if (this.boolPreviewing) {
                        autoFocus();
                    }
                }
                return true;
       }
        return super.onKeyDown(keyCode, event);
    }

    private void startPreview(int nWidth, int nHeight) {
        this.nPreviewWidth = nWidth;
        this.nPreviewHeight = nHeight;
        if (this.boolPreviewing) {
               return;
        }
        if (this.camera == null) {
               this.camera = android.hardware.Camera.open();
        }
        if (this.camera != null && this.surfaceHolder != null) {
               Camera.Parameters parm = this.camera.getParameters();
               parm.setPreviewSize(nWidth, nHeight);
               this.camera.setParameters(parm);
               try {
                              this.camera.setPreviewDisplay(this.surfaceHolder);
                      } catch (IOException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                this.camera.startPreview();
                this.boolPreviewing = true;
        }
    }

    private void stopPreview() {
        if (this.camera != null) {
                this.camera.stopPreview();
                this.camera.release();
                this.camera = null;
                this.boolPreviewing = false;
        }
    }

    public void onClick(View v) {
                // TODO Auto-generated method stub

        }

        public void surfaceChanged(SurfaceHolder holder, int format, int width,
                        int height) {
                if (holder.isCreating()) {
                        startPreview(width, height);
                }
        }

        public void surfaceCreated(SurfaceHolder holder) {
                this.surfaceHolder = holder;
        }

        public void surfaceDestroyed(SurfaceHolder holder) {
                stopPreview();
                this.surfaceHolder = null;
        }


}
```

ModImage.java

```
package com.wellsmt.ImageDetect;

import java.io.InputStream;
import java.io.OutputStream;

import jjil.core.RgbImage;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.provider.MediaStore.Images.Media;
import android.view.ContextMenu;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.Toast;
```

```
public class ModImage extends Activity implements Runnable{
        final short RED_CONTEXT = 0;
        final short GREEN_CONTEXT = 1;
        final short BLUE_CONTEXT = 2;
        final short EDGE_CONTEXT = 3;
        final short GRAY_CONTEXT = 4;
        final short ADD_CONTEXT = 5;
        final short BLEND_CONTEXT = 6;
        final short CONV_CONTEXT = 7;
        final short DOT_PROD_CONTEXT = 8;
        final short SCALE_CONTEXT = 9;
        final short D_THRESH_CONTEXT = 10;

        private static final int DIALOG_SINGLE_CHOICE = 0;

        private int end = 0;
        private int start = 0;

        final short MENU_SAVE = 0;
        final short MENU_UNDO = 1;
        final short MENU_PROC_TIME = 2;

        private ProgressDialog myProgressDialog = null;

        private RgbImage prevRgbImage = null;
        private static RgbImage mRgbImage = null;
        private static RgbImage mSecRgbImage = null;

        private static int width;
        private static int height;
        private static int mSecWidth;
        private static int mSecHeight;
        private int secondImage;
        private short currContextSelect;

        private ModImageView mImageView;

        public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);

         Toast.makeText(ModImage.this, "Obtained Image to Process...", Toast.LENGTH_LONG).show();

         setContentView(R.layout.image);
         this.mImageView = (ModImageView) findViewById(R.id.detectedImage);
         this.mImageView.resetFaces();
                this.mImageView.resetShowX();

                registerForContextMenu(mImageView);
                if( mRgbImage != null)
        {
                this.mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width,
height, Bitmap.Config.ARGB_8888) );
        }
        }
        @Override
        protected Dialog onCreateDialog(int id) {
         switch (id) {
         case DIALOG_SINGLE_CHOICE:
             return new AlertDialog.Builder(ModImage.this)
                 .setIcon(R.drawable.icon)
                 .setTitle("To Be Added Soon...")
                 //.setSingleChoiceItems(R.array.select_dialog_items2, 0, new
DialogInterface.OnClickListener() {
                 //    public void onClick(DialogInterface dialog, int whichButton) {

                         /* User clicked on a radio button do some stuff */
                 //     }
                 //})
                 .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int whichButton) {
                         /* User clicked Yes so do some stuff */
                     }
                 })
                 .create();
         }
         return null;
        }

        public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)
        {
                super.onCreateContextMenu(menu, v, menuInfo);
                menu.add(0, RED_CONTEXT, 0, "Red");
            menu.add(0, GREEN_CONTEXT, 0, "Green");
            menu.add(0, BLUE_CONTEXT, 0, "Blue");
            menu.add(0, EDGE_CONTEXT,0, "Sobel");
            menu.add(0, GRAY_CONTEXT,0, "Gray");
            menu.add(0, ADD_CONTEXT,0, "Add");
            menu.add(0, BLEND_CONTEXT,0, "Blend");
            menu.add(0, CONV_CONTEXT,0, "Convolve");
            menu.add(0, DOT_PROD_CONTEXT,0, "Dot Product");
            menu.add(0, SCALE_CONTEXT,0, "Scale");
            menu.add(0, D_THRESH_CONTEXT,0, "Double Threshold");
        }

        public boolean onContextItemSelected(MenuItem item) {
                currContextSelect = (short)item.getItemId();
                switch (currContextSelect) {
            case RED_CONTEXT:
                if (mRgbImage != null) {
                        showOnlyRed();

                mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width, height,
Bitmap.Config.ARGB_8888) );
                }
                 return true;
            case GREEN_CONTEXT:
                if (mRgbImage != null) {
                        showOnlyGreen();
                     mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width,
height, Bitmap.Config.ARGB_8888) );
                }
                 return true;
            case BLUE_CONTEXT:
                if (mRgbImage != null) {
                        showOnlyBlue();
                     mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width,
height, Bitmap.Config.ARGB_8888) );
                }
                 return true;
            case EDGE_CONTEXT:
                 if (mRgbImage != null)
                 {
                        sobelImage();



                }
                 return true;
            case GRAY_CONTEXT:
                if (mRgbImage != null)
                {
                        greyScale(true);
                     mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width,
height, Bitmap.Config.ARGB_8888) );
                }
                return true;
            case ADD_CONTEXT:
                if( mRgbImage != null)
                {
                        additionImage();
                }
                return true;
            case BLEND_CONTEXT:
                if( mRgbImage != null)
                {
                        showDialog(DIALOG_SINGLE_CHOICE);
                }
                return true;

            case CONV_CONTEXT:
                if( mRgbImage != null)
                {
                        convolveImage();
                }
                return true;
            case DOT_PROD_CONTEXT:
                if( mRgbImage != null)
                {
                        showDialog(DIALOG_SINGLE_CHOICE);
                }
                return true;
            case SCALE_CONTEXT:
                if( mRgbImage != null)
                {
                        showDialog(DIALOG_SINGLE_CHOICE);
                }
                return true;
            case D_THRESH_CONTEXT:
                if( mRgbImage != null)
                {
                        showDialog(DIALOG_SINGLE_CHOICE);
                }
                return true;
            default:
                    return super.onContextItemSelected(item);
            }
        }

        public boolean onCreateOptionsMenu(Menu menu)
        {
                menu.add(0, MENU_UNDO, 0, "Undo Last change");
                menu.add(0, MENU_SAVE, 0, "Save Image");
                menu.add(0, MENU_PROC_TIME, 0, "Processing Time");
            return true;
        }

        /* Handles item selections */
        public boolean onOptionsItemSelected(MenuItem item) {
                switch (item.getItemId()) {
                  case MENU_UNDO:
                        if( prevRgbImage != null)
                        {
                                mRgbImage = (RgbImage)prevRgbImage.clone();
                                mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(),
width, height, Bitmap.Config.ARGB_8888) );
                        }
                        return true;
                    case MENU_SAVE:
                        if (mRgbImage != null)
                        {
                                saveImage();
                         }
                        return true;
                    case MENU_PROC_TIME:
                        if (prevRgbImage != null)
                        {
                                displayProcTime();
                         }
                        return true;
                  }
            return false;
        }
```

```
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        switch (keyCode) {
            case KeyEvent.KEYCODE_CAMERA:
            case KeyEvent.KEYCODE_FOCUS:
                startActivity(new Intent("com.wellsmt.ImageDetect.Preview"));
                finish();
                return true;
       }
       return super.onKeyDown(keyCode, event);
    }

    public void showOnlyRed( )
    {
        thresholdColorPixels( 0,255,255 );

    }

    public void showOnlyGreen()
    {
        thresholdColorPixels( 255,0,255 );

    }

    public void showOnlyBlue()
    {
        thresholdColorPixels( 255,255,0 );

    }

    public static void setJpegData(byte[] jpegData)
    {
       Bitmap bitmap = BitmapFactory.decodeByteArray(jpegData, 0, jpegData.length, null);
       mRgbImage = RgbImageAndroid.toRgbImage(bitmap);
       width = bitmap.getWidth();
       height = bitmap.getHeight();
       bitmap.getPixels(mRgbImage.getData(), 0, width, 0, 0, width, height);
       }

    public static void setJpegData(Bitmap temp)
    {
       Bitmap bitmap = temp.copy(Bitmap.Config.ARGB_8888, true);
       mRgbImage = RgbImageAndroid.toRgbImage(bitmap);
       width = bitmap.getWidth();
       height = bitmap.getHeight();
       bitmap.getPixels(mRgbImage.getData(), 0, width, 0, 0, width, height);
    }

    public static void setSecJpegData(Bitmap temp)
    {
       Bitmap bitmap = temp.copy(Bitmap.Config.ARGB_8888, true);
       mSecRgbImage = RgbImageAndroid.toRgbImage(bitmap);
       mSecWidth = bitmap.getWidth();
       mSecHeight = bitmap.getHeight();
       bitmap.getPixels(mSecRgbImage.getData(), 0, mSecWidth, 0, 0, mSecWidth, mSecHeight);
    }

    public void saveImage() {
               ContentValues values = new ContentValues(3);

               values.put(Media.MIME_TYPE, "image/jpeg");

               // Add a new record without the bitmap, but with the values just set.
               // insert() returns the URI of the new record.
               Uri uri = getContentResolver().insert(Media.EXTERNAL_CONTENT_URI, values);
               // Now get a handle to the file for that record, and save the data into it.
               // Here, sourceBitmap is a Bitmap object representing the file to save to the
database.
               try {
                   OutputStream outStream = getContentResolver().openOutputStream(uri);
                   Bitmap.createBitmap(mRgbImage.getData(), width, height,
Bitmap.Config.ARGB_8888).compress(Bitmap.CompressFormat.JPEG, 50, outStream);
                   outStream.close();
                   Toast.makeText(ModImage.this, "Image Saved...", Toast.LENGTH_LONG).show();
               } catch (Exception e) {
                   Toast.makeText(this, "Image Failed to Save...", Toast.LENGTH_LONG).show();
               }
    }

    private void displayProcTime()
    {
        Toast.makeText(this, "(" + width + "x" + height + "), Image Processing Time = " + (end -
start)*10e-6 + "[sec].", Toast.LENGTH_LONG).show();
    }

    public void thresholdColorPixels( int rthresh,int gthresh, int bthresh ) {
        int[] rgbData = mRgbImage.getData();
        prevRgbImage = (RgbImage) mRgbImage.clone();

        start = (int) System.currentTimeMillis();
        for(int y = 0; y < height; y++)
        {
               int outputOffset = y*width;
               for(int x = 0;x < width; x++)
               {
                      int index = outputOffset + x;

                      int R = ((rgbData[index]>>16) & 0xff);
                   int G = ((rgbData[index]>>8) & 0xff);
                   int B = ((rgbData[index]) & 0xff);

                      if( R <= rthresh){
                             R = 0;
                      }
                      if( G <= gthresh){
                             G = 0;
                      }
                      if( B <= bthresh){
                             B = 0;
                      }

                      rgbData[index] = 0xff000000 | (R << 16) | (G << 8) | B;
               }
        }
        end = (int) System.currentTimeMillis();
    }

    // converts yuv camera data format to gray scale
    public void greyScale(boolean storePrevious) {
        int[] rgbData = mRgbImage.getData();
        if( storePrevious )
        {
               prevRgbImage = (RgbImage)mRgbImage.clone();
        }

        for(int y = 0; y < height; y++)
        {
               int outputOffset = y*width;
               for(int x = 0;x < width; x++)
               {
                      int index = outputOffset + x;

                      int R = ((rgbData[index]>>16) & 0xff);
                   int G = ((rgbData[index]>>8) & 0xff);
                   int B = ((rgbData[index]) & 0xff);

                      int grey = (R + G + B)/3;

                      rgbData[index] = 0xff000000 | (grey << 16) | (grey << 8) | grey;

               }
        }
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        super.onActivityResult(requestCode, resultCode, data);

        if (resultCode == RESULT_OK)
        {
                Uri chosenImageUri = data.getData();

                if( chosenImageUri != null)
                {
                                try
                                {
                                        InputStream photoStream =
getContentResolver().openInputStream(chosenImageUri);
                                        BitmapFactory.Options opts = new BitmapFactory.Options();

                                        opts.inSampleSize = 4;

                                        Bitmap bitmap = BitmapFactory.decodeStream(photoStream,
null, opts);
                                        if( bitmap != null)
                                        {
                                                setSecJpegData(bitmap);

                                                myProgressDialog =
ProgressDialog.show(ModImage.this,
                                                 "Calculating...", "Image addition", true, false);
                                         Thread thread= new Thread(this);
                                         thread.start();

                                                switch( secondImage)
                                                {
                                                        case ADD_CONTEXT:

                                                        break;
                                                        case BLEND_CONTEXT:
                                                                blend();
                                                        break;
                                                }

                                        mImageView.setImageBitmap(
Bitmap.createBitmap(mRgbImage.getData(), width, height, Bitmap.Config.ARGB_8888) );

                                        }

                                        photoStream.close();

                                }
                                catch (Exception e)
                                {
                                        e.printStackTrace();
                                }
                }
                }
    }

    public void blend()
    {

    }

    public void additionImage()
    {
        Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
                photoPickerIntent.setType("image/*");

                startActivityForResult(photoPickerIntent, 1);
    }
    public void convolveImage()
    {
       myProgressDialog = ProgressDialog.show(ModImage.this,
               "Calculating...", "Convolve with Averaging Filter", true, false);
          Thread thread= new Thread(this);
          thread.start();
    }

    public void sobelImage()
    {
        myProgressDialog = ProgressDialog.show(ModImage.this,
             "Calculating...", "Sobel", true, false);
        Thread thread= new Thread(this);
        thread.start();
    }

       public void run()
       {
              switch(this.currContextSelect)
              {
                     case EDGE_CONTEXT:
                            sobel();
                     break;
                     case CONV_CONTEXT:
                            double[]
template={(1/9.0),(1/9.0),(1/9.0),(1/9.0),(1/9.0),(1/9.0),(1/9.0),(1/9.0),(1/9.0)};
                            convolve(template, 3, 3);
                     break;
                     case ADD_CONTEXT:
                            addition();
                     break;
              }
              handler.sendEmptyMessage(0);
       }

    private Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {

                     myProgressDialog.dismiss();
                mImageView.setImageBitmap( Bitmap.createBitmap(mRgbImage.getData(), width,
height, Bitmap.Config.ARGB_8888) );
                displayProcTime();
        }
};

    public void sobel()
    {
       prevRgbImage = (RgbImage)mRgbImage.clone();

       start = (int) System.currentTimeMillis();

       float[] template={-1,0,1,-2,0,2,-1,0,1};

              int templateSize=3;

              int[] rgbData = mRgbImage.getData();
              int[] total = new int[width*height];
              int sumY=0;
              int sumX=0;
              int max=0;
for( int n = 0; n<1; n++)
{

              for(int x=(templateSize-1)/2; x<width-(templateSize+1)/2;x++)
              {
                     for(int y=(templateSize-1)/2; y<height-(templateSize+1)/2;y++)
                     {
                            sumY=0;
```

```
                                for(int x1=0;x1<templateSize;x1++)
                                {
                                        for(int y1=0;y1<templateSize;y1++)
                                        {
                                                int x2 = (x-(templateSize-1)/2+x1);
                                                int y2 = (y-(templateSize-1)/2+y1);
                                                float value = (rgbData[y2*width+x2] & 0xff) * 
(template[y1*templateSize+x1]);

                                                sumY += value;
                                        }
                                }

                                sumX = 0;

                                for(int x1=0;x1<templateSize;x1++)
                                {
                                        for(int y1=0;y1<templateSize;y1++)
                                        {
                                                int x2 = (x-(templateSize-1)/2+x1);
                                                int y2 = (y-(templateSize-1)/2+y1);
                                                float value = (rgbData[y2*width+x2] & 0xff) * 
(template[x1*templateSize+y1]);

                                                sumX += value;
                                        }
                                }
                                total[y*width+x] = (int)Math.sqrt(sumX*sumX+sumY*sumY);

                                if(max < total[y*width+x])
                                        max = total[y*width+x];
                        }
                }


                float ratio=(float)max/255;

                for(int x=0; x<width;x++)
                {
                        for(int y=0; y<height;y++)
                        {
                                sumX = (int)(total[y*width+x]/ratio);
                                total[y*width+x] = 0xff000000 | ((int)sumX << 16 | (int)sumX << 8 | 
(int)sumX);
                        }
                }
}
                System.arraycopy(
                 total,
                 0,
                 rgbData,
                 0,
                 width*height);

                end = (int) System.currentTimeMillis();
    }

    public void convolve(double[] mat, int rows, int cols)
    {
       if((rows % 2) == 0 || (cols % 2) == 0)
       {
       }
       else
       {
                start = (int) System.currentTimeMillis();

                int[] rgbData = mRgbImage.getData();
                int[] conv = new int[width*height];
                int sumR = 0;
                int sumG = 0;
                int sumB = 0;
                prevRgbImage = (RgbImage)mRgbImage.clone();

                for(int x=(cols-1)/2; x<width-(cols+1)/2;x++)
                {
                        for(int y=(rows-1)/2; y<height-(rows+1)/2;y++)
                        {
                                sumR=0;
                                sumG=0;
                                sumB=0;

                                for(int x1=0;x1<cols;x1++)
                                {
                                        for(int y1=0;y1<rows;y1++)
                                        {
                                                int x2 = (x-(cols-1)/2+x1);
                                                int y2 = (y-(rows-1)/2+y1);
                                                int R = ((rgbData[y2*width+x2]>>16) & 0xff);
                                            int G = ((rgbData[y2*width+x2]>>8) & 0xff);
                                            int B = ((rgbData[y2*width+x2]) & 0xff);

                                                sumR += R * (mat[y1*cols+x1]);

                                                sumG += G * (mat[y1*cols+x1]);
                                                sumB += B * (mat[y1*cols+x1]);
                                        }
                                }

                                conv[y*width+x] = 0xff000000 | ((int)sumR << 16 | (int)sumG << 8 |
(int)sumB);
                        }
                }
                System.arraycopy(
                     conv,
                     0,
                     rgbData,
                     0,
                     width*height);
                end = (int) System.currentTimeMillis();
        }
    }

    public void addition()
    {
        secondImage = ADD_CONTEXT;
        prevRgbImage = (RgbImage)mRgbImage.clone();


        if( mSecRgbImage != null && mRgbImage != null )
        {
                start = (int) System.currentTimeMillis();

                int largerWidth = 0;
                int largerHeight = 0;

                if( mSecWidth < width)
                {
                        largerWidth = width;
                }
                else
                {
                        largerWidth = mSecWidth;
                }

                if( mSecHeight < height)
                {
                        largerHeight = height;
                }
                else
                {
                        largerHeight = mSecHeight;
                }
                int[] rgbData = mRgbImage.getData();
                int[] mSecRgbData = mSecRgbImage.getData();
                int[] total = new int[largerWidth*largerHeight];

                for(int y = 0, y2 = 0; y < height || y2 < mSecHeight; y++, y2++)
                {
                        int rgbOutputOffset = y*width;
                        int secRgbOutputOffset = y2*mSecWidth;
                        for(int x = 0,x2 = 0;x < width || x2 < mSecWidth; x++,x2++)
                        {
                                int index = rgbOutputOffset + x;
                                int secIndex = secRgbOutputOffset + x2;

                                int R,G,B,sR,sG,sB;

                                if( x < width && y < height)
                                {
                                        R = ((rgbData[index]>>16) & 0xff);
                                        G = ((rgbData[index]>>8) & 0xff);
                                        B = ((rgbData[index]) & 0xff);
                                }
                                else
                                {
                                        R = ((mSecRgbData[secIndex]>>16) & 0xff);
                                        G = ((mSecRgbData[secIndex]>>8) & 0xff);
                                        B = ((mSecRgbData[secIndex]) & 0xff);
                                }

                                if( x2 < mSecWidth && y2 < mSecHeight)
                                {
                                        sR = ((mSecRgbData[secIndex]>>16) & 0xff);
                                        sG = ((mSecRgbData[secIndex]>>8) & 0xff);
                                        sB = ((mSecRgbData[secIndex]) & 0xff);

                                }
                                else
                                {
                                        sR = ((rgbData[index]>>16) & 0xff);
                                        sG = ((rgbData[index]>>8) & 0xff);
                                        sB = ((rgbData[index]) & 0xff);
                                }

                                double mR,mG,mB;

                            mR = (R + sR)/2.0;
                            mG = (G + sG)/2.0;
                            mB = (B + sB)/2.0;

                                total[index] = 0xff000000 | ((int)(mR) << 16) | ((int)(mG) << 8) |
(int)(mB);
                        }
                }

                System.arraycopy(
                     total,
                     0,
                     rgbData,
                     0,
                     width*height);

                end = (int) System.currentTimeMillis();
        }
    }
}
```

RgbImage.java

```
/*
 * RgbImage.java
 *
 * Created on August 27, 2006, 12:47 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 *
 * Copyright 2006 by Jon A. Webb
 *     This program is free software: you can redistribute it and/or modify
 *    it under the terms of the GNU Lesser General Public License as published by
 *    the Free Software Foundation, either version 3 of the License, or
 *    (at your option) any later version.
 *
 *    This program is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *    GNU Lesser General Public License for more details.
 *
 *    You should have received a copy of the Lesser GNU General Public License
 *    along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

package jjil.core;

/**
 * RgbImage is the type used to hold an RGB image, which
 * is stored as an ARGB image type (32-bits) with the
 * A byte ignored.
 * <p>
 * Implementation-specific libraries define methods that allow the creation
 * of an RgbImage from a native image type. RgbImage is therefore the first and
 * last jjil.core object used after capture and before display of an image.
 * @author webb
 */
public class RgbImage extends Image {
    /** A pointer to the image data
     */
    private final int[] wImage;

    /** Creates a new instance of RgbImage
     *
     * @param cWidth   the image width
     * @param cHeight  the image height
     */
    public RgbImage(int cWidth, int cHeight) {
        super(cWidth, cHeight);
        this.wImage = new int[getWidth()*getHeight()];
    }

    public RgbImage(int cWidth, int cHeight, int[] rnData) {
        super(cWidth, cHeight);
        this.wImage = rnData;
    }

    /**
     * Creates a new instance of RgbImage, assigning a constant value
     * @param bR the red color value to be assigned.
     * @param bG the green color value to be assigned.
     * @param bB the blue color value to be assigned.
     * @param cWidth the image width
     * @param cHeight the image height
     */
    public RgbImage(int cWidth, int cHeight, byte bR, byte bG, byte bB) {
        super(cWidth, cHeight);
        this.wImage = new int[getWidth()*getHeight()];
        int nRgb = RgbVal.toRgb(bR, bG, bB);
        for (int i=0; i<this.getWidth()*this.getHeight();i++) {
            this.wImage[i] = nRgb;
        }
    }

    /**
```

```
     * Creates a new instance of RgbImage, assigning a constant value
     * @param nRgb the packed RGB value to assign
     * @param cWidth the image width
     * @param cHeight the image height
     */
    public RgbImage(int cWidth, int cHeight, int nRgb) {
        super(cWidth, cHeight);
        this.wImage = new int[getWidth()*getHeight()];
        for (int i=0; i<this.getWidth()*this.getHeight();i++) {
            this.wImage[i] = nRgb;
        }
    }

    /** Creates a shallow copy of this image
     *
     * @return the image copy.
     */
    public Object clone()
    {
        RgbImage image = new RgbImage(getWidth(), getHeight());
        System.arraycopy(
                this.getData(),
                0,
                image.getData(),
                0,
                getWidth()*getHeight());
        return image;
    }

    /**
     * Fill a rectangle in an RgbImage with a given value
     * @param r the Rect to fill
     * @param nRgb the color to assign
     * @return the modified RgbImage (i.e., this)
     * @throws Error if the bounds are outside the image
     */
    public RgbImage fill(Rect r, int nRgb) throws Error
    {
        if (r.getTop() < 0 || r.getBottom() > this.getHeight() ||
                r.getLeft() < 0 || r.getRight() > this.getWidth()) {
            throw new Error(Error.PACKAGE.CORE,
                    ErrorCodes.BOUNDS_OUTSIDE_IMAGE,
                    r.toString(),
                    null,
                    null);
        }
        for (int i=r.getTop(); i<r.getBottom(); i++) {
            for (int j=r.getLeft(); j<r.getRight(); j++) {
                this.wImage[i*this.getWidth()+j] = nRgb;
            }
        }
        return this;
    }

    /** Get a pointer to the image data.
     *
     * @return the data pointer.
     */
    public int[] getData()
    {
        return this.wImage;
    }


    /** Return a string describing the image.
     *
     * @return the string.
     */
    public String toString()
    {
        return super.toString() + " (" + getWidth() + "x" + getHeight() + //$NON-NLS-1$ //$NON-
NLS-2$
                ")"; //$NON-NLS-1$
    }
}
```

RgbImageAndroid.java

```
package com.wellsmt.ImageDetect;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import jjil.core.RgbImage;
import android.content.Context;
import android.graphics.Bitmap;

public class RgbImageAndroid {
      /**
       * The sole way to create an RgbImage from an image captured from the
camera.
       * The parameters are the pointer to the byte data passed to the JPEG
picture
       * callback and the width and height image you want. You must reduce
the
       * image size because otherwise you will run out of memory. Width and
height
       * reduction by a factor of 2 works on the GPhone.<p>
       * Ex. usage<p>
       * public void onPictureTaken(byte [] jpegData, android.hardware.Camera
camera) {
     *      RgbImage rgb = RgbImageAndroid.toRgbImage(jpegData,
     *                  camera.getParameters().getPictureSize().width/2,
     *                  camera.getParameters().getPictureSize().height/2);
     * }
       * @param jpegData image data supplied to JpegPictureCallback
       * @param nWidth target width image to return
       * @param nHeight target height image to return
       * @return RgbImage initialized with the image from the camera.
       */
    static public RgbImage toRgbImage(Bitmap bmp) {
      int nWidth = bmp.getWidth();
      int nHeight = bmp.getHeight();
      RgbImage rgb = new RgbImage(nWidth, nHeight);
      bmp.getPixels(rgb.getData(), 0, nWidth, 0, 0, nWidth, nHeight);
      return rgb;
    }

    static public Bitmap toBitmap(RgbImage rgb)
    {
      return Bitmap.createBitmap(
                  rgb.getData(),
                  rgb.getWidth(),
                  rgb.getHeight(),
                  Bitmap.Config.ARGB_8888);
    }
    static public void toDisplay(Context context, RgbImage rgb)
    {
      Bitmap bmp = toBitmap(rgb);
    }

    static public void toFile(Context context, RgbImage rgb, int nQuality,
String szPath)
      throws IOException
    {
      OutputStream os = new FileOutputStream(szPath);
      try {
            Bitmap bmp = toBitmap(rgb);
            Bitmap.CompressFormat format = Bitmap.CompressFormat.JPEG;
            szPath = szPath.toLowerCase();
            if (szPath.endsWith("jpg") || szPath.endsWith("jpeg")) { //$NON-
NLS-1$ //$NON-NLS-2$
                  format = Bitmap.CompressFormat.JPEG;
            } else if (szPath.endsWith("png")) { //$NON-NLS-1$
                  format = Bitmap.CompressFormat.PNG;
            }
            bmp.compress(format, nQuality, os);
      } finally {
            os.close();
      }
    }
}
```