FINAL YEAR PROJECT

Static ASL Detection and Conversion to Speech

AUTHORS:
SABIKAH BATOOL MUKHI(s.b.mukhi@gmail.com)
SARCHINA KUMARI(sarchina1@live.com)
VERDA YOUSUF (verdaly@gmail.com)

Introduction:

What is ASL ?

ASL means AMERICAN SIGN LANGUAGE, which is termed as a universal sign language all over the globe. Sign language is used to communicate with hearing disability. American Sign Language (ASL) is commonly said to be "the fourth most-used language in the United States" (alternatively phrased as "the *third* most-used *non-English* language in the U.S.").

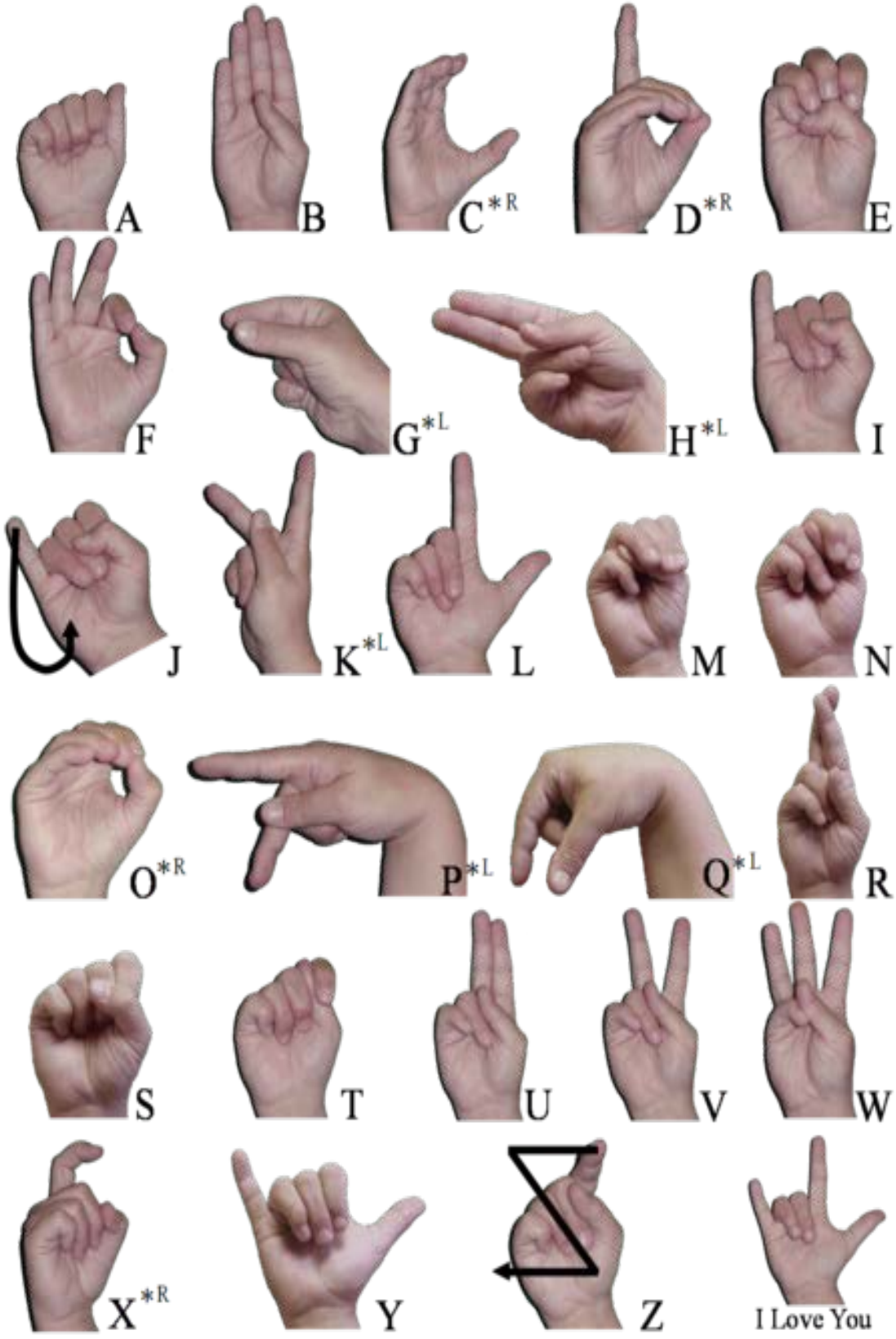
What is "ASL to Speech"?

Need: A dumb may communicate to people around him by using his sign language. But does not a dumb person need a phone as a speech-blessed person does? With the fast moving world and this globalization around, every human is dependent on long-distance communication.

This project aims at breaking the communication barriers that person has to face.

What it does: The user may give sign inputs using the mobile camera, and this sign gets converted to speech, ready to be delivered on other end of the call.

Scope: We are currently restricting the ASL conversion to static words/ gestures only. Dynamic sign language is not included in our project due to the degree of complexity in technicality as well as operating system capability of a smart phone or tablet.


A
B
C*R
D*R
E
F
G*L
H*L
I
J
K*L
L
M
N
O*R
P*L
Q*L
R
S
T
U
V
W
X*R
Y
Z
I Love You

High Level Design:

1. Rationale and sources of the project idea
   Project has been sighted as a service to a social cause and can be utilized by special people's institutes.
2. Logical structure
   The application's logical structure lays on image processing and comparison with images in database. Next text to speech is utilized for oral feedback.
3. Hardware / Software tradeoffs
   -Java Android Environment
   -OpenCV (Computer Vision)
   -A Handset (Samsung Young)
4. Patents, copyright and trademarks
   -OPEN CV
   - Eclipse

Software / Hardware Design:

1. **Overview**
   We are currently aiming at developing this application for mobiles. Keeping its factor of mobility ease, this application can prove to be an extremely helpful one for communication with the disabled while on move. It can prove to be a handy guide for new learners as well and visual interpretation can also be utilized to tutor, teach and instruct the disabled personnel

2. **Program Details**

➢ *JAVA ANDROID ENVIRONMENT*

- AIDE is an integrated development environment (IDE) for developing real Android Apps directly on Android devices. AIDE supports the full edit-compile-run cycle: write code with the feature rich editor offering advanced features like code completion, real-time error checking, refactoring and smart code navigation, and run your App with a single click.
- AIDE will turn your Android tablet with keyboard into a real development box. We use the Transformer Prime running Ice Cream Sandwich to code with AIDE. AIDE will turn your Android Phone into a small development computer to browse and touch your code on the go.
- AIDE supports building Apps with Java/Xml and the Android SDK as well as Apps with C/C++ and the Android NDK.
- AIDE is fully compatible with Eclipse projects. You can just copy the source code to your device and open the Eclipse project in AIDE to start coding. Alternatively you can keep your source code on your Drop box - AIDE integrates with Drop box and allows to

easily download from your Drop box and sync back your changes. AIDE can also open Android Studio projects, which follow the default folder structure.

- AIDE supports GIT for professional development.

- **AIDE Premium Key is required for the following features:**
  - Saving files in larger projects (5+ Java files)
  - Direct run without user prompt

Git push/commit/branch
- APK publishing

➢ *OPEN CV (COMPUTER VISION)*

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate it.

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

➢ BRISK ALGORITHM USED:

We propose BRISK, a novel method for keypoint detection, description and matching. A comprehensive evaluation on benchmark datasets reveals BRISK's adaptive, high quality performance as in state-of-the-art algorithms, albeit at a dramatically lower computational cost (an order of magnitude faster than SURF in cases). The key to speed lies in the application of a novel scale-space FAST-based detector in combination with the assembly of a bit-string

descriptor from intensity comparisons retrieved by dedicated sampling of each keypoint neighborhood.

1. **Hardware Details**

- *A HANDSET (SAMSUNG Galaxy young)*

Results:

- SPEED OF EXECUTION: Approximately 15 seconds to process one image.
- USABILITY: Great potential with given its credentials.
- ACCURACY: 75%
- SAFETY: Absolutely safe.
- EVIDENCE: Live demonstrations at Final Year Project-1 presentation and video provided.

Conclusions:

Our project has reached its completion successfully within the desired time period and limitations. The expected image processing with all its uses and text to speech conversion to have conformed with our goals for the project. The application is ready to use on any smart phone or portable device and has great potential in the market on pitching. Uptill now we have not confronted any application that uses image processing and vision tactics to convert ASL to speech in real-time. The basic purpose of this project is to enable a dumb person, use a cell phone for distant communication. Thereby we can claim to have pioneered in a real time processing application for ASL communication over the phone. The user will find the application absolutely user friendly and easy. Parents with deaf and dumb children as well as teachers would be greatly benefitted all over the world by this application. It is a major and a constructive stepping stone for furthering education and communication ease for all.

Appendix:

Appendix 1: Code

Appendix 2: Schematic of your hardware

Appendix 3: Software/parts list

Appendix 4: Work distribution

Appendix 5: Project timeline

APPENDIX 1 : CODE

APPENDIX 1 : CODE

```
// ACTIVITY CLASS**********

package org.opencv.samples.tutorial3;

import java.io.BufferedWriter;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Date;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.List;
import java.util.ListIterator;
import java.lang.Object;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfDMatch;
import org.opencv.core.MatOfFloat;
import org.opencv.core.MatOfInt;
import org.opencv.core.MatOfKeyPoint;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Scalar;
import org.opencv.features2d.DMatch;
import org.opencv.features2d.DescriptorExtractor;
import org.opencv.features2d.DescriptorMatcher;
import org.opencv.features2d.FeatureDetector;
import org.opencv.features2d.Features2d;
```

```
import org.opencv.highgui.VideoCapture;
import org.opencv.imgproc.Imgproc;


import org.opencv.video.BackgroundSubtractorMOG;
import org.opencv.video.BackgroundSubtractorMOG2;
import org.opencv.video.Video;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;


import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera.Size;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.SubMenu;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;



public class Tutorial3Activity extends Activity implements CvCameraViewListener2,
OnTouchListener {
    private static final String TAG = "OCVSample::Activity";
```

```
    private static final String TAG_compare = "inComparison";
    private static final String TAG_touch = "ontouch";

    private Tutorial3View mOpenCvCameraView;
    private List<Size> mResolutionList;
    private MenuItem[] mEffectMenuItems;
    private SubMenu mColorEffectsMenu;
    private MenuItem[] mResolutionMenuItems;
    private SubMenu mResolutionMenu;

   BackgroundSubtractorMOG  bs_mog;
    BackgroundSubtractorMOG2 bs_mog2;
    private Mat              mIntermediateMat;
    private Mat              mIntermediateMat2;
    private Mat                          foreground;

    private static MatOfKeyPoint keypointsA,keypointsB,keypointsC, dupKeypoints;
    private Video v;
    //private static Bitmap bmp;

     // List<MatOfPoint> contours;
            Mat hierarchy;
//          MatOfPoint contour;
//          MatOfPoint contour2;
            MatOfInt hull;
            MatOfInt histSize;
     MatOfInt channels ;
   //   ArrayList<Mat> bgr_planes;
     ArrayList<Mat> bgr_planesA;
     ArrayList<Mat> bgr_planesB;
     ArrayList<Mat> bgr_planesC;
     ArrayList<Mat> bgr_planes2;
            int maxIndex=0;


            //


            private static int descriptor = DescriptorExtractor.BRISK;
            private static int min_dist = 150;
     private static int min_matches =50;
     private static long startTime, endTime;
     private static String text;
     private static Mat imgA,imgB,imgC,img;
```

```
        private static Mat img2;
        private static Bitmap bmpimgA,bmpimgB,bmpimgC,bmp;
        private static Bitmap bmpimg2;
        private static int matchesAFound,matchesBFound,matchesCFound;
        //private static Bitmap clickedbmp;
//



private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
    @Override
    public void onManagerConnected(int status) {
        switch (status) {
        case LoaderCallbackInterface.SUCCESS: {
                Log.i(TAG, "OpenCV loaded successfully");
                                        mOpenCvCameraView.enableView();
                mOpenCvCameraView.setOnTouchListener(Tutorial3Activity.this);
               Mat outputImg = new Mat();
                img2=new Mat();
                imgA=new Mat();
                imgB=new Mat();
                imgC=new Mat();
                img=new Mat();

                VideoCapture cap = new VideoCapture(1);
                if (!cap.isOpened())
                {}
                else
                {
                Mat frame= new Mat();
                cap.retrieve(frame);

                }

                mOpenCvCameraView.enableView();
                run();

        }
                break;
        default: {
                super.onManagerConnected(status);
        }
                break;
        }
```

```
    }
};
    public Tutorial3Activity() {
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        setContentView(R.layout.tutorial3_surface_view);

        mOpenCvCameraView                    =                    (Tutorial3View)
findViewById(R.id.tutorial3_activity_java_surface_view);

        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);

        mOpenCvCameraView.setCvCameraViewListener(this);
    }

    public void run() {
                // TODO Auto-generated method stub

          img2=new Mat();


                bmpimgA=BitmapFactory.decodeResource(getResources(), R.drawable.a1);
                bmpimgB=BitmapFactory.decodeResource(getResources(), R.drawable.b1);
                bmpimgC=BitmapFactory.decodeResource(getResources(), R.drawable.c1);

                bmpimgA = Bitmap.createScaledBitmap(bmpimgA, 100, 100, true);
        bmpimgB = Bitmap.createScaledBitmap(bmpimgB, 100, 100, true);
        bmpimgC = Bitmap.createScaledBitmap(bmpimgC, 100, 100, true);

}
    @Override
    public void onPause()
    {
        super.onPause();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
```

```
    }

    @Override
    public void onResume()
    {
        super.onResume();
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_3,                this,
mLoaderCallback);
    }

    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }

    public void onCameraViewStarted(int width, int height) {
            mIntermediateMat = new Mat();
            mIntermediateMat2= new Mat();
            hierarchy= new Mat();
   //       contours= new ArrayList<MatOfPoint>();
            //mBGSub = new BackgroundSubtractorMOG();

    }

    public void onCameraViewStopped() {
            if (mIntermediateMat != null)
            mIntermediateMat.release();
            if (mIntermediateMat2 != null)
            mIntermediateMat2.release();
            if (hierarchy != null)
            hierarchy.release();
  /*        if (contour != null)
            contour.release();
            if (contour2 != null)
            contour2.release();*/
            if (hull != null)
            hull.release();
            if (imgA != null)
            imgA.release();
            if (imgB != null)
        imgB.release();
            if (imgC != null)
        imgC.release();
```

```
        if (img != null)
     img.release();

    mIntermediateMat = null;
    mIntermediateMat2 = null;
    hierarchy=null;
  //   contour=null;
  //   contour2=null;
    hull=null;
}

  public Mat onCameraFrame(CvCameraViewFrame inputFrame) {


       Mat gray=inputFrame.rgba();
   img=gray;
   return gray;
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    List<String> effects = mOpenCvCameraView.getEffectList();

    if (effects == null) {
       Log.e(TAG, "Color effects are not supported by device!");
       return true;
    }

    mColorEffectsMenu = menu.addSubMenu("Color Effect");
    mEffectMenuItems = new MenuItem[effects.size()];

    int idx = 0;
    ListIterator<String> effectItr = effects.listIterator();
    while(effectItr.hasNext()) {
      String element = effectItr.next();
      mEffectMenuItems[idx] = mColorEffectsMenu.add(1, idx, Menu.NONE, element);
      idx++;
    }

    mResolutionMenu = menu.addSubMenu("Resolution");
    mResolutionList = mOpenCvCameraView.getResolutionList();
    mResolutionMenuItems = new MenuItem[mResolutionList.size()];

    ListIterator<Size> resolutionItr = mResolutionList.listIterator();
```

```
        idx = 0;
        while(resolutionItr.hasNext()) {
            Size element = resolutionItr.next();
            mResolutionMenuItems[idx] = mResolutionMenu.add(2, idx, Menu.NONE,
                    Integer.valueOf(element.width).toString() + "x" +
Integer.valueOf(element.height).toString());
            idx++;
         }

        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        Log.i(TAG, "called onOptionsItemSelected; selected item: " + item);
        if (item.getGroupId() == 1)
        {
            mOpenCvCameraView.setEffect((String) item.getTitle());
            Toast.makeText(this, mOpenCvCameraView.getEffect(),
Toast.LENGTH_SHORT).show();
        }
        else if (item.getGroupId() == 2)
        {
            int id = item.getItemId();
            Size resolution = mResolutionList.get(id);
            mOpenCvCameraView.setResolution(resolution);
            resolution = mOpenCvCameraView.getResolution();
            String caption = Integer.valueOf(resolution.width).toString() + "x" +
Integer.valueOf(resolution.height).toString();
            Toast.makeText(this, caption, Toast.LENGTH_SHORT).show();
        }

        return true;
    }

    @SuppressLint("SimpleDateFormat")
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        Log.i(TAG_touch,"onTouch event");

            Mat temp= new Mat();

        temp= img;
         work(temp);
    //    run();
```

```
        return false;
    }

    public void work(Mat temp)
            {
            bmpimg2 = Bitmap.createBitmap(temp.cols(),temp.rows(),Bitmap.Config.ARGB_8888);
            Utils.matToBitmap(temp, bmpimg2) ;
            bmpimg2 = Bitmap.createScaledBitmap(bmpimg2, 100, 100, true);
            Utils.bitmapToMat(bmpimg2, temp);
                    Imgproc.cvtColor(temp, temp, Imgproc.COLOR_RGBA2GRAY);
            temp.convertTo(temp, CvType.CV_32F);

                new asyncTask(Tutorial3Activity.this).execute();

      startTime = System.currentTimeMillis();

            }

//

    public static class asyncTask extends AsyncTask<Void, Void, Void> {
        private static Mat imgA,imgB,imgC,img2,sign,  dupDescriptors;
        private static Mat descriptorsA,descriptorsB,descriptorsC;
        private static FeatureDetector detector;
        private static DescriptorExtractor DescExtractor;
        private static DescriptorMatcher matcher;
        //private static MatOfKeyPoint keypointsA, dupKeypoints;
        private static MatOfDMatch matchesA, matchesA_final_mat;
        private static MatOfDMatch matchesB, matchesB_final_mat;
        private static MatOfDMatch matchesC, matchesC_final_mat;
        private static ProgressDialog pd;
        private static boolean isDuplicate = false;
        private Tutorial3Activity asyncTaskContext=null;
        private static Scalar RED = new Scalar(255,0,0);
        private static Scalar GREEN = new Scalar(0,255,0);

        private static Integer maxValue;
        private static String maxKey;

        public asyncTask(Tutorial3Activity context)
        {
              asyncTaskContext=context;
             // sign=img2;
```

```
	}
	   @Override
	   protected Void doInBackground(Void... arg0) {
		   // TODO Auto-generated method stub
		   compare();
		   Log.i(TAG_compare, "im comin till compare");
	   return null;

	   }
	   @Override
	protected void onPreExecute() {
		pd = new ProgressDialog(asyncTaskContext);
		pd.setIndeterminate(true);
		pd.setCancelable(true);
		pd.setCanceledOnTouchOutside(false);
		pd.setMessage("Processing...");
		pd.show();
	}



	   @Override
		   protected void onPostExecute(Void result) {
			   try {
			   /*	Mat img3 = new Mat();
				   MatOfByte drawnMatches = new MatOfByte();
				   Features2d.drawMatches(imgB, keypointsB, img2, dupKeypoints,
						   matchesA_final_mat, img3, GREEN, RED,
drawnMatches, Features2d.NOT_DRAW_SINGLE_POINTS);
				   bmp = Bitmap.createBitmap(img3.cols(), img3.rows(),
						   Bitmap.Config.ARGB_8888);
				   Imgproc.cvtColor(img3, img3, Imgproc.COLOR_BGR2RGB);
				   Utils.matToBitmap(img3, bmp);*/

				   endTime = System.currentTimeMillis();
				   if (maxValue > min_matches)// dev discretion for

	   // number of matches to

	   // be found for an image

	   // to be judged as

	   // duplicate
```

```
                                {
                                        text = maxValue
                                                        + " matches were found. \n Answer is
"+maxKey+"\nTime taken="
                                                        + (endTime - startTime) + "ms";
                                        isDuplicate =true;
                                } else {
                                        text = maxValue
                                                        + " \nMinimum but Answer is
"+maxKey+"\nTime taken="
                                                        + (endTime - startTime) + "ms";

                                        isDuplicate = false;
                                }
                                pd.dismiss();
                                final AlertDialog.Builder alertDialog = new AlertDialog.Builder(
                                                asyncTaskContext);
                                alertDialog.setTitle("Result");
                                alertDialog.setCancelable(true);
                                LayoutInflater factory = LayoutInflater.from(asyncTaskContext);
                                final View view = factory.inflate(R.layout.image_view, null);
                                ImageView matchedImages = (ImageView) view
                                                .findViewById(R.id.finalImage);
                                matchedImages.setImageBitmap(bmp);
                                matchedImages.invalidate();
                                final CheckBox shouldBeDuplicate = (CheckBox) view
                                                .findViewById(R.id.checkBox);
                                TextView message = (TextView)
view.findViewById(R.id.message);
                                message.setText(text);
                                alertDialog.setView(view);

                                alertDialog.show();
                        } catch (Exception e) {
                                e.printStackTrace();
                                Toast.makeText(asyncTaskContext, e.toString(),
                                                Toast.LENGTH_LONG).show();
                        }
                }



        void compare() {
```

```
try {
        detector = FeatureDetector.create(FeatureDetector.PYRAMID_FAST);
   DescExtractor = DescriptorExtractor.create(descriptor);
   matcher = DescriptorMatcher
             .create(DescriptorMatcher.BRUTEFORCE_HAMMING);

     bmpimgA = bmpimgA.copy(Bitmap.Config.ARGB_8888, true);
     imgA = new Mat();
     Utils.bitmapToMat(bmpimgA, imgA);
     Imgproc.cvtColor(imgA, imgA, Imgproc.COLOR_BGR2RGB);
     keypointsA = new MatOfKeyPoint();
     descriptorsA = new Mat();

     //B
     bmpimgB = bmpimgB.copy(Bitmap.Config.ARGB_8888, true);
     imgB = new Mat();
     Utils.bitmapToMat(bmpimgB, imgB);
     Imgproc.cvtColor(imgB, imgB, Imgproc.COLOR_BGR2RGB);
     keypointsB = new MatOfKeyPoint();
     descriptorsB = new Mat();

    //C
     bmpimgC = bmpimgC.copy(Bitmap.Config.ARGB_8888, true);
     imgC = new Mat();
     Utils.bitmapToMat(bmpimgC, imgC);
     Imgproc.cvtColor(imgC, imgC, Imgproc.COLOR_BGR2RGB);
     keypointsC = new MatOfKeyPoint();
     descriptorsC = new Mat();

     //image 2
     bmpimg2 = bmpimg2.copy(Bitmap.Config.ARGB_8888, true);
     img2 = new Mat();
     Utils.bitmapToMat(bmpimg2, img2);
     Imgproc.cvtColor(img2, img2, Imgproc.COLOR_BGR2RGB);

     dupKeypoints = new MatOfKeyPoint();
     dupDescriptors = new Mat();
     matchesA = new MatOfDMatch();
     matchesB = new MatOfDMatch();
     matchesC = new MatOfDMatch();

     //A
     detector.detect(imgA, keypointsA); //keypoints containc detected points of imgA
     Log.d("LOG!", "number of query Keypoints= " + keypointsA.size());
```

```
DescExtractor.compute(imgA, keypointsA, descriptorsA);
Log.d("LOG!", "number of descriptorsA= " + descriptorsA.size());
//B
detector.detect(imgB, keypointsB); //keypoints containc detected points of imgA
DescExtractor.compute(imgB, keypointsB, descriptorsB);
//C
detector.detect(imgC, keypointsC); //keypoints containc detected points of imgA
DescExtractor.compute(imgC, keypointsC, descriptorsC);


detector.detect(img2, dupKeypoints); //dupkeypoints containc detected points of
img2
Log.d("LOG!", "number of dup Keypoints= " + dupKeypoints.size());
// Descript keypoints
DescExtractor.compute(img2, dupKeypoints, dupDescriptors);
Log.d("LOG!","number of dupDescriptors= " + dupDescriptors.size());
// matching descriptorsA

matcher.match(descriptorsA, dupDescriptors, matchesA);
matcher.match(descriptorsB, dupDescriptors, matchesB);
matcher.match(descriptorsC, dupDescriptors, matchesC);

Log.d("LOG!", "Matches Size " + matchesA.size());
// New method of finding best matches
List<DMatch> matchesAList = matchesA.toList();
List<DMatch> matchesA_final = new ArrayList<DMatch>();
for (int i = 0; i < matchesAList.size(); i++) {
      if (matchesAList.get(i).distance <= min_dist) {
            matchesA_final.add(matchesA.toList().get(i));
      }
}
matchesAFound=matchesA_final.size();

//B
List<DMatch> matchesBList = matchesB.toList();
List<DMatch> matchesB_final = new ArrayList<DMatch>();
for (int i = 0; i < matchesBList.size(); i++) {
      if (matchesBList.get(i).distance <= min_dist) {
            matchesB_final.add(matchesB.toList().get(i));
      }
}
matchesBFound=matchesB_final.size();
//C
List<DMatch> matchesCList = matchesC.toList();
```

```
                List<DMatch> matchesC_final = new ArrayList<DMatch>();
                for (int i = 0; i < matchesCList.size(); i++) {
                        if (matchesCList.get(i).distance <= min_dist) {
                                matchesC_final.add(matchesC.toList().get(i));
                        }
                }
                matchesCFound=matchesC_final.size();

                matchesA_final_mat = new MatOfDMatch();
                matchesA_final_mat.fromList(matchesB_final);

                Hashtable<String,Integer> h = new Hashtable<String,Integer>();

                h.put("A", matchesAFound);
                h.put("B", matchesBFound);
                h.put("C", matchesCFound);

                int max = Collections.max(h.values());
                Log.d("inComparison", "max match= " +max);



                Collection<Integer> values = h.values();
                maxValue = Collections.max(values);
                Enumeration<String> keys = h.keys();
                while(keys.hasMoreElements()){
                   String key = keys.nextElement();
                   if((h.get(key)).equals(maxValue))
                      maxKey=key;
                }
        } catch (Exception e) {
                e.printStackTrace();

         }
   }
   } // end of async class
}
// VIEW CLASS*****************
package org.opencv.samples.tutorial3;

import java.io.FileOutputStream;
import java.util.List;

import org.opencv.android.JavaCameraView;
```

```
import android.content.Context;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.Size;
import android.util.AttributeSet;
import android.util.Log;

public class Tutorial3View extends JavaCameraView implements PictureCallback {

    private static final String TAG = "Sample::Tutorial3View";
    private String mPictureFileName;

    public Tutorial3View(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public List<String> getEffectList() {
        return mCamera.getParameters().getSupportedColorEffects();
    }

    public boolean isEffectSupported() {
        return (mCamera.getParameters().getColorEffect() != null);
    }

    public String getEffect() {
        return mCamera.getParameters().getColorEffect();
    }

    public void setEffect(String effect) {
        Camera.Parameters params = mCamera.getParameters();
        params.setColorEffect(effect);
        mCamera.setParameters(params);
    }

    public List<Size> getResolutionList() {
        return mCamera.getParameters().getSupportedPreviewSizes();
    }

    public void setResolution(Size resolution) {
        disconnectCamera();
        mMaxHeight = resolution.height;
        mMaxWidth = resolution.width;
        connectCamera(getWidth(), getHeight());
```

```
    }

    public Size getResolution() {
        return mCamera.getParameters().getPreviewSize();
    }

    public void takePicture(final String fileName) {
        Log.i(TAG, "Taking picture");
        this.mPictureFileName = fileName;
        // Postview and jpeg are sent in the same buffers if the queue is not empty when performing a capture.
        // Clear up buffers to avoid mCamera.takePicture to be stuck because of a memory issue
        mCamera.setPreviewCallback(null);

        // PictureCallback is implemented by the current class
        mCamera.takePicture(null, null, this);
    }

    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        Log.i(TAG, "Saving a bitmap to file");
        // The camera preview was automatically stopped. Start it again.
        mCamera.startPreview();
        mCamera.setPreviewCallback(this);

        // Write the image in a file (in jpeg format)
        try {
            FileOutputStream fos = new FileOutputStream(mPictureFileName);

            fos.write(data);
            fos.close();

        } catch (java.io.IOException e) {
            Log.e("PictureDemo", "Exception in photoCallback", e);
        }

    }
}
```

APPENDIX 2: MAJOR FUNCTIONS USED

APPENDIX 2: MAJOR FUNCTIONS USED

VideoCapture : Class for video capturing from video files or cameras. The class provides C++ API for capturing video from cameras or for reading video files.

ConcurrentLinkedQueue: This implementation employs an efficient "wait-free" algorithm based on one described in Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms by Maged M. Michael and Michael L. Scott.

Iterators are weakly consistent, returning elements reflecting the state of the queue at some point at or since the creation of the iterator. They do not throw ConcurrentModificationException, and may proceed concurrently with other operations. Elements contained in the queue since the creation of the iterator will be returned exactly once.

Beware that, unlike in most collections, the size method is NOT a constant-time operation. Because of the asynchronous nature of these queues, determining the current number of elements requires a traversal of the elements, and so may report inaccurate results if this collection is modified during traversal. Additionally, the bulk operations addAll, removeAll, retainAll, containsAll, equals, and toArray are not guaranteed to be performed atomically. For example, an iterator operating concurrently with an addAll operation might view only some of the added elements.

This class and its iterator implement all of the optional methods of the Queue and Iterator interfaces.

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a ConcurrentLinkedQueue happen-before actions subsequent to the access or removal of that element from the ConcurrentLinkedQueue in another thread.

Mat: The class Mat represents an n-dimensional dense numerical single-channel or multi-channel array. It can be used to store real or complex-valued vectors and matrices, grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms (though, very high-dimensional histograms may be better stored in a SparseMat).

Bitmap: can be used to store or read a Bitmap from a file or video

Core.getTickFrequency();//no of ticks/second

Core.getTickCount();//no. of tick b/w the start n end time of certain event

Handler: There are two main uses for a Handler: (1) to schedule messages and runnables to be executed as some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.

SurfaceView: Provides a dedicated drawing surface embedded inside of a view hierarchy. You can control the format of this surface and, if you like, its size; the SurfaceView takes care of placing the surface at the correct location on the screen

The surface is Z ordered so that it is behind the window holding its SurfaceView; the SurfaceView punches a hole in its window to allow its surface to be displayed. The view hierarchy will take care of correctly compositing with the Surface any siblings of the SurfaceView that would normally appear on top of it. This can be used to place overlays such as buttons on top of the Surface, though note however that it can have an impact on performance since a full alpha-blended composite will be performed each time the Surface changes.

The transparent region that makes the surface visible is based on the layout positions in the view hierarchy. If the post-layout transform properties are used to draw a sibling view on top of the SurfaceView, the view may not be properly composited with the surface.

Access to the underlying surface is provided via the SurfaceHolder interface, which can be retrieved by calling getHolder().

The Surface will be created for you while the SurfaceView's window is visible; you should implement surfaceCreated(SurfaceHolder) and surfaceDestroyed(SurfaceHolder) to discover when the Surface is created and destroyed as the window is shown and hidden.

One of the purposes of this class is to provide a surface in which a secondary thread can render into the screen. If you are going to use it this way, you need to be aware of some threading semantics:

- All SurfaceView and SurfaceHolder.Callback methods will be called from the thread running the SurfaceView's window (typically the main thread of the application). They thus need to correctly synchronize with any state that is also touched by the drawing thread.

- You must ensure that the drawing thread only touches the underlying Surface while it is valid -- between SurfaceHolder.Callback.surfaceCreated() and SurfaceHolder.Callback.surfaceDestroyed().

Paint:The Paint class holds the style and color information about how to draw geometries, text and bitmaps

GestureDetector: Detects various gestures and events using the supplied MotionEvents. The GestureDetector.OnGestureListener callback will notify users when a particular motion event has occurred. This class should only be used with MotionEvents reported via touch (don't use for trackball events). To use this class:

- Create an instance of the GestureDetector for your View

• In the onTouchEvent(MotionEvent) method ensure you call onTouchEvent(MotionEvent). The methods defined in your callback will be executed when the events occur.

AND MANY MORE.

APPENDIX 3 : SCHEMATIC OF YOUR HARDWARE

APPENDIX 3 : SCHEMATIC OF YOUR HARDWARE

Schematic of your hardware

The specifications are as follow:

n as Samsung Galaxy Young Duos S6312, Samsung GT-S6310L

GENERAL	2G Network	GSM 850 / 900 / 1800 / 1900 - GT-S6310
		GSM 850 / 900 / 1800 / 1900 - GT-S6312 (SIM 1 & SIM 2)
	3G Network	HSDPA 900 / 2100
	SIM	Optional Dual SIM (Mini-SIM, dual stand-by)
	Announced	2013, February
	Status	Available. Released 2013, March
BODY	Dimensions	109.4 x 58.6 x 12.5 mm (4.31 x 2.31 x 0.49 in)
	Weight	112 g (3.95 oz)
DISPLAY	Type	TFT capacitive touchscreen, 256K colors
	Size	320 x 480 pixels, 3.27 inches (~176 ppi pixel density)
	Multitouch	Yes
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
MEMORY	Card slot	microSD, up to 64 GB
	Internal	4 GB (2 GB user available), 768 MB RAM
DATA	GPRS	Yes
	EDGE	Yes
	Speed	HSDPA, 7.2 Mbps; HSUPA, 5.76 Mbps
	WLAN	Wi-Fi 802.11 b/g/n, Wi-Fi hotspot, Wi-Fi Direct
	Bluetooth	Yes, v3.0 with A2DP
	USB	Yes, microUSB v2.0
CAMERA	Primary	3.15 MP, 2048 x 1536 pixels, check quality

	Features	Geo-tagging
	Video	Yes, VGA@24fps
	Secondary	No
FEATURES	OS	Android OS, v4.1.2 (Jelly Bean)
	Chipset	Qualcomm MSM7227A Snapdragon
	CPU	1 GHz Cortex-A5
	GPU	Adreno 200
	Sensors	Accelerometer, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM, RSS
	Browser	HTML
	Radio	Stereo FM radio with RDS, FM recording
	GPS	Yes, with A-GPS support
	Java	Yes, via Java MIDP emulator
	Colors	White
		- SNS integration - MP4/WMV/H.264/H.263 player - MP3/WAV/eAAC+/FLAC player - Organizer - Image/video editor - Document viewer/editor - Google Search, Maps, Gmail, YouTube, Calendar, Google Talk, Picasa - Voice memo/dial - Predictive text input
BATTERY		Li-Ion 1300 mAh battery
	Stand-by	(2G) / Up to 250 h (3G)
	Talk time	(2G) / Up to 6 h 40 min (3G)
MISC	SAR US	1.18 W/kg (head) 1.28 W/kg (body)
	SAR EU	0.88 W/kg (head) 0.56 W/kg (body)
	Price group	
TESTS	Display	Contrast ratio: 701:1 (nominal) / 1.220:1 (sunlight)

	Loudspeaker	Voice 65dB / Noise 64dB / Ring 65dB
	Audio quality	Noise -91.4dB / Crosstalk -92.5dB
	Camera	Photo

APPENDIX 4: SOFTWAREPARTS/ LIST

APPENDIX 4: SOFTWAREPARTS/ LIST

- *JAVA ANDROID ENVIRONMENT*
- *OPEN CV (COMPUTER VISION)*

APPENDIX 5: WORK DISTRIBUTION

APPENDIX 5: WORK DISTRIBUTION

- **Initial research:**
  Sabikah, Sarchina, Verda
- **Scope identification and boundary setting:**
  Sabikah, Sarchina, Verda
- **Code and resources hunting:**
  Sabikah, Verda
- **Device expertise and installation on handset**:
  Verda
- **Software installations on desktop:**
  Sabikah, Sarchina
- **Coding:**
  Sabikah
- **Ongoing research and debugging of code:**
  Sabikah, Sarchina, Verda
- **Testing and retesting of code:**
  Sabikah
- **Software testing:**
  Sabikah, Sarchina, Verda

APPENDIX 6: PROJECT TIMELINE

APPENDIX 6: PROJECT TIMELINE

MONTHLY PROJECT PLANNER : ASL DETECTION & CONVERSION
BY: SABIKAH MUKHI, SARCHINA KUMARI & VERDA YOUSUF

DATE (DEADLINE)	MONTH	WEEK	OBJECTIVES	STATUS
4/9/2013	SEPTEMBER	1	UNDERSTANDING FINAL YEAR PROJECT GUIDELINES AND PROCEDURES	SUCCESSFULLY ACCOMPLISHED
11/9/2013	SEPTEMBER	2	GROUP MEMEBERS TO BE DECIDED	SUCCESSFULLY ACCOMPLISHED
20/9/2013	SEPTEMBER	3	GROUP TO DECIDE PROJECT SCOPE	SUCCESSFULLY ACCOMPLISHED
30/9/2013	SEPTEMBER	4	DECISION TO BE FINALIZED ABOUT FINAL YEAR PROJECT	SUCCESSFULLY ACCOMPLISHED
3/10/2013	OCTOBER	1	MEETING WITH SUPERVISOR, UNOFFICIAL SUPERVISOR CONCERNING PROJECT	SUCCESSFULLY ACCOMPLISHED
10/10/2013	OCTOBER	2	GUIDELINES AND SCOPE SETTING FOR THE PROJECT	SUCCESSFULLY ACCOMPLISHED
20/10/2013	OCTOBER	3	EXPLORATION OF OPEN CV LIBRARIES FOR ANDROID EXPLORATION  Links	SUCCESSFULLY ACCOMPLISHED
27/10/2013	OCTOBER	4	EXPLORATION OF OPEN CV AND JAVA ANDROID APPLCATIONS LIBRARIES AND FUNCTIONS  Android Applicatio...	SUCCESSFULLY ACCOMPLISHED
5/11/2013	NOVEMBER	1	RESEARCH OF SIMILAR PROJECTS OR RELATIVES OF SCOPE FIRST FYP PROGRESS REPORT SUBMISSION	SUCCESSFULLY ACCOMPLISHED
15/11/2013	NOVEMBER	2	RESEARCH ON HARDWARE RECOGNITION OF FACIAL RECOGNITION IN ANDROID	SUCCESSFULLY

15/11/2013	NOVEMBER	2	RESEARCH ON HARDWARE RECOGNITION OF FACIAL RECOGNITION IN ANDROID FUNCTIONS AND THEIR DESCRIPTION  Functions description	SUCCESSFULLY ACCOMPLISHED
20/11/2013	NOVEMBER	3	PDF STUDY RESEARCH: OPEN CV FACE DETECTION, MASTERING OPEN CV, LEARNING OPENCV  mASTERING OPENCV_C...	SUCCESSFULLY ACCOMPLISHED
28/11/2013	NOVEMBER	4	WORKING WITH OPEN CV AND INTELIMAGE PROCESSING LIBRARIES PROCESSING IMAGE DATA TOOLS  Working with Open...	SUCCESSFULLY ACCOMPLISHED
5/12/2013	DECEMBER	1	IMAGE PROCESSING CAPTURING IMAGE ON SMART PHONE IMAGE SAVING AND RETRIVAL  opencvAndr oidFaceDe...	SUCCESSFULLY ACCOMPLISHED
14/12/2013	DECEMBER	2	WORKING IN IMAGE PROCESSING  HardwareIm plementati...	SUCCESSFULLY ACCOMPLISHED

19/12/2013	DECEMBER	3	SECOND FYP PROGRESS REPORT SUBMISSION  fyp report	SUCCESSFULLY ACCOMPLISHED
24/12/2013	DECEMBER	4	FINAL EXAMS AND PRESENTATIONS CODE HUNTING	SUCCESSFULLY ACCOMPLISHED
4/1/2013	JANUARY	1	IMAGE PROCESSING FOR IMAGE DETECTION	SUCCESSFULLY ACCOMPLISHED
10/1/2014	JANUARY	2	BACKGROUND DETECTION FOR DISTINGUISHING IMAGE	SUCCESSFULLY ACCOMPLISHED
16/1/2014	JANUARY	3	BACKGROUND DETECTION DISTINGUISHING IMAGE	SUCCESSFULLY ACCOMPLISHED
16/1/2014	JANUARY	4	EDGE DETECTION IN THE IMAGE  snapsJavaC V	SUCCESSFULLY ACCOMPLISHED
4/2/2014	FEBRUARY	1	IMAGE TAKING FROM DIFFERENT ANGLES AND EVALUATING RESULTS	SUCCESSFULLY ACCOMPLISHED
14/2/2014	FEBRUARY	2	TAKING TWO STATIC IMAGES PERCENTAGE SIMILARITY IN TWO IMAGES	SUCCESSFULLY ACCOMPLISHED
19/2/2014	FEBRUARY	3	PERCENTAGE SIMILARITY IN TWO STATIC IMAGES	SUCCESSFULLY ACCOMPLISHED
27/2/2014	FEBRUARY	4	IMAGE COMPARISON	SUCCESSFULLY ACCOMPLISHED
6/3/2014	MARCH	1	IMAGE DATABASE FORMATION COMPARISON OF NEW IMAGE WITH DATABASE	SUCCESSFULLY ACCOMPLISHED
12/3/2014	MARCH	2	IMAGE DATABASE FORMATION COMPARISON OF NEW IMAGE WITH DATABASE	SUCCESSFULLY ACCOMPLISHED
20/3/2014	MARCH	3	IMAGE DATABASE FORMATION COMPARISON OF NEW IMAGE WITH DATABASE	SUCCESSFULLY ACCOMPLISHED

20/3/2014	MARCH	3	IMAGE DATABASE FORMATION COMPARISON OF NEW IMAGE WITH DATABASE	SUCCESSFULLY ACCOMPLISHED
27/3/2014	MARCH	4	IMTEGRATION OF TEXT WITH IMAGES IN DATABASE FOR INPUT IMAGES	SUCCESSFULLY ACCOMPLISHED
5/4/2014	APRIL	1	TEXT TO SPEECH CONVERSION	SUCCESSFULLY ACCOMPLISHED
12/4/2014	APRIL	2	TEXT TO SPEECH CONVERSION	SUCCESSFULLY ACCOMPLISHED
19/4/2014	APRIL	3	FINAL STAGE FOR APPLICATION TO ENSURE 'APP' FORM	SUCCESSFULLY ACCOMPLISHED
26/4/2014	APRIL	4	FINAL STAGE FOR APPLICATION TO ENSURE 'APP' FORM	SUCCESSFULLY ACCOMPLISHED
6/5/2014	MAY	1	TESTING PHASE BUGS FIXING SCOPE ANALYSIS	SUCCESSFULLY ACCOMPLISHED
13/5/2014	MAY	2	TESTING PHASE OF PROJECT BUGS FIXING SCOPE ANALYSIS	SUCCESSFULLY ACCOMPLISHED
20/5/2014	MAY	3	TESTING PHASE OF PROJECT BUGS FIXING	PENDING
27/5/2014	MAY	4	FINAL TESTING!!	PENDING

References:

Books, Inspirations for code and designs, Papers, Background sites:

- https://opencv.org/
- https://code.google.com/p/javacv/downloads/detail?name=javacv-0.3-bin.zip
- http://shashindrasri.blogspot.com/2013/01/setting-up-javacv-on-netbeans-and.html
- http://stackoverflow.com/questions/12091095/configure-java-cv-with-eclipse-juno
- http://opencvlover.blogspot.com/2012/04/javacv-setup-with-eclipse-on-windows-7.html
- https://code.google.com/p/javacv/wiki/ConvertingOpenCV
- http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/
- https://github.com/MasteringOpenCV/code
- http://jayrambhia.com/blog/beginning-android-opencv/
- http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/
- http://www.blogosfera.co.uk/2013/03/screen-capture-using-asl-library/
- https://code.google.com/p/android-screenshot-library/source/browse/emo/src/pl/polidea/asl/demo/ScreenshotDemo.java?r=b6fbe6432404759edebeaabbfae7aa32777b1a28
- MASTERINGOPENCV_Chapter_04.pdf
- opencvAndroidFaceDetection.pdf
- openCvComputerVision.pdf
- OReilly Learning OpenCV.pdf
- snapsJavaCV.pdf
- Working with OpenCV and Intel Image Processing Libraries. Processing image Data Tools.pdf
- HardwareImplementationOfFacialRecognitionAndroid.pdf

Acknowledgements:

We would like to thank all the owners and writers of resources used and above all DR. SAJJAD HAIDER for providing us support and help in areas and times we required for the project to be a success.

We also thank DR. FAISAL IRADAT for mentoring us on organizing and formulating our final year project and its documentation.

****