Lampiran 2 Source Code Aplikasi

```
package com.example.wahyuadjieprasetyo.skripsi;

import android.Manifest;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.SurfaceView;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.android.Utils;
import org.opencv.core.DMatch;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfDMatch;
import org.opencv.core.MatOfKeyPoint;
import org.opencv.core.Scalar;
import org.opencv.features2d.DescriptorExtractor;
import org.opencv.features2d.DescriptorMatcher;
import org.opencv.features2d.FeatureDetector;
import org.opencv.features2d.Features2d;
import org.opencv.imgproc.Imgproc;

import java.io.IOException;
import java.io.InputStream;
import java.util.LinkedList;
import java.util.List;

public class DeteksiCAM extends AppCompatActivity implements
CameraBridgeViewBase.CvCameraViewListener2{
int bestIndex=0;
int jumGaris=0;
    String gab="";
    dbHelper helper=null;

    int jd=2;
    Button btnCek;
    String[]arID;
    String[]arNama;

String hasilAnalisa="";
    private static final String TAG = "OCVSample::Activity";
    private static final int REQUEST_PERMISSION = 100;
    private int w, h;
```

```
    private CameraBridgeViewBase mOpenCvCameraView;
    TextView tvName;
    Scalar RED = new Scalar(255, 0, 0);
    Scalar GREEN = new Scalar(0, 255, 0);
    FeatureDetector detector;
    DescriptorExtractor descriptor;
    DescriptorMatcher matcher;

    Mat descriptors2;
    MatOfKeyPoint keypoints2;

    MatOfKeyPoint [] key1;
    Mat desc1[],citra1[];

    static {
        if (!OpenCVLoader.initDebug())
            Log.d("ERROR", "Unable to load OpenCV");
        else
            Log.d("SUCCESS", "OpenCV loaded");
    }

    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();
                    try {
                        initializeOpenCVDependencies();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                break;
                default: {
                    super.onManagerConnected(status);
                }
                break;
            }
        }
    };

    private void initializeOpenCVDependencies() throws IOException {

        helper=new dbHelper(this);
        mOpenCvCameraView.enableView();
        detector = FeatureDetector.create(FeatureDetector.ORB);
        descriptor = DescriptorExtractor.create(DescriptorExtractor.ORB);
        matcher =
DescriptorMatcher.create(DescriptorMatcher.BRUTEFORCE_HAMMING);

        Button btnCek=(Button)findViewById(R.id.btnCek);
        btnCek.setOnClickListener(new View.OnClickListener() {
            public void onClick(View arg0) {
                hasil();

            }});
```

```
        jd=16;////////////////jumlah
arID=new String[jd];
arNama=new String[jd];
desc1=new Mat[jd];
citra1=new Mat[jd];
key1=new MatOfKeyPoint[jd];

jd=15;

//peliang daun
arNama[0]="Peliang Daun1";
arNama[1]="Peliang Daun2";
arNama[2]="Peliang Daun3";
arNama[3]="Peliang Daun4";
arNama[4]="Peliang Daun5";

arNama[5]="embun tepung1";
arNama[6]="embun tepung2";
arNama[7]="embun tepung3";
arNama[8]="embun tepung4";
arNama[9]="embun tepung5";

arNama[10]="Terezia1";
arNama[11]="Terezia2";
arNama[12]="Terezia3";
arNama[13]="Terezia4";
arNama[14]="Terezia5";

arID[0]="dp1.jpg";
arID[1]="dp2.jpg";
arID[2]="dp3.jpg";
arID[3]="dp4.jpg";
arID[4]="dp5.jpg";
arID[5]="dp6.jpg";
arID[6]="dp7.jpg";
arID[7]="dp8.jpg";
arID[8]="dp9.jpg";
arID[9]="dp10.jpg";
arID[10]="dp11.jpg";
arID[11]="dp12.jpg";
arID[12]="dp13.jpg";
arID[13]="dp14.jpg";
arID[14]="dp15.jpg";


        AssetManager assetManager = getAssets();

        for(int k=0;k<jd;k++) {
            citra1[k] = new Mat();
            InputStream istr1 = assetManager.open(arID[k]);//a.jpeg
            Bitmap bitmap1 = BitmapFactory.decodeStream(istr1);
            Utils.bitmapToMat(bitmap1, citra1[k] );
            Imgproc.cvtColor(citra1[k] , citra1[k] ,
Imgproc.COLOR_RGB2GRAY);


            citra1[k] .convertTo(citra1[k] , 0);
            desc1[k] = new Mat();
            key1[k] = new MatOfKeyPoint();
            detector.detect(citra1[k] , key1[k]);
            descriptor.compute(citra1[k] , key1[k], desc1[k]);
        }
        //=============================================1
```

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        setContentView(R.layout.activity_deteksi);


        if(ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA}, REQUEST_PERMISSION);
        }
    mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.deteksi);

        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }


    @Override
    public void onPause() {
        super.onPause();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }
    @Override
    public void onResume() {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
            Log.d(TAG, "Internal OpenCV library not found. Using OpenCV
Manager for initialization");
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0,
DeteksiCAM.this, mLoaderCallback);
        } else {
            Log.d(TAG, "OpenCV library found inside package. Using it!");

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }

        }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }
    @Override
    public void onCameraViewStarted(int width, int height) {
        w = width;
        h = height;
    }

    @Override
    public void onCameraViewStopped() {
```

```
    }


    public Mat recognize(Mat aInputFrame) {//aInputFrame=objek dari
Kamera
        double nmin = 10000000;
        int indexKe=0;

        Imgproc.cvtColor(aInputFrame, aInputFrame,
Imgproc.COLOR_RGB2GRAY);
        descriptors2 = new Mat();
        keypoints2 = new MatOfKeyPoint();
        detector.detect(aInputFrame, keypoints2);
        descriptor.compute(aInputFrame, keypoints2, descriptors2);


        MatOfDMatch matches = null;

        int JM = 0;
        Double max_dist = 0.0;
        Double min_dist = 100.0;

        matches = new MatOfDMatch();
        for(int k=0;k<jd;k++){
        if (citra1[k].type() == aInputFrame.type()) {
            try {

                matcher.match(desc1[k], descriptors2, matches);
                List<DMatch> matchesList1 = matches.toList();

                JM = matchesList1.size();
                for (int i = 0; i < JM; i++) {
                    Double dist = (double) matchesList1.get(i).distance;
                    if (dist < min_dist)
                        min_dist = dist;
                    if (dist > max_dist)
                        max_dist = dist;
                }

                if (min_dist < nmin) {
                    indexKe = k;
                    nmin = min_dist;
                }
                Log.d("MAXMIN", k+"." + max_dist + "@" + min_dist + "@" +
(1.2 * min_dist)+"#"+indexKe);
            } catch (Exception ee) {
            }
        } else {
            return aInputFrame;
        }
    }//loop

//=============================================================================
==========3
        matches = new MatOfDMatch();
        if (citra1[indexKe].type() == aInputFrame.type()) {
            try {
                matcher.match(desc1[indexKe], descriptors2, matches);
            } catch (Exception ee) {
            }

        } else {
```

```
            return aInputFrame;
        }

        gab="";
        List<DMatch> matchesListOut = matches.toList();
        JM = matchesListOut.size();
        List<DMatch>  listBest=null;
        Mat imgBest=new Mat();
        MatOfKeyPoint keyBest=null;

            listBest=matchesListOut;
            imgBest=citra1[indexKe];
            keyBest=key1[indexKe];
          hasilAnalisa=arNama[indexKe];


//=========================================================================
==========================LAST
        int mm=0;
        LinkedList<DMatch> good_matches = new LinkedList<DMatch>();
        for (int i = 0; i < JM; i++) {
            if (listBest.get(i).distance <= (1.2 * min_dist)) {//1.5 *
min_dist
                good_matches.addLast(listBest.get(i));
                mm++;
            }
        }
        gab=gab+String.valueOf(mm)+"#";
        Log.d("MAXMINJUM",indexKe+"#"+gab );

        if(jumGaris<mm){
            bestIndex=indexKe;
            jumGaris=mm;
            Log.d("MAXMINJUM2",bestIndex+"#"+jumGaris+"="+hasilAnalisa );
        }

        MatOfDMatch goodMatches = new MatOfDMatch();
        goodMatches.fromList(good_matches);
        Mat outputImg = new Mat();


        MatOfByte drawnMatches = new MatOfByte();
        if (aInputFrame.empty() || aInputFrame.cols() < 1 ||
aInputFrame.rows() < 1) {
            return aInputFrame;
        }
        try {
            Features2d.drawMatches(imgBest, keyBest, aInputFrame,
key1[bestIndex], goodMatches, outputImg, GREEN, RED, drawnMatches,
Features2d.NOT_DRAW_SINGLE_POINTS);
            Imgproc.resize(outputImg, outputImg, aInputFrame.size());
        }
        catch(Exception rr){}
        return outputImg;
    }


        @Override
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
        return recognize(inputFrame.rgba());
    }
```

```
    public void hasil(){
        new AlertDialog.Builder(this)
                .setTitle("HASIL ANALISA:"+hasilAnalisa)
                .setMessage("#"+hasilAnalisa)
                .setNeutralButton("OK", new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dlg, int sumthin)
{
                        helper.insertarsip("1 Januari 2019", "13:00:00",
"-",hasilAnalisa,"-");

                        finish();
                    }})
                .show();
    }

}
```