BAB IV
HASIL AKHIR DAN ANALISIS

4.1 Prinsip Kerja Perangkat Lunak *Leaf App*

Aplikasi yang dibuat memiliki cara kerja dalam penggunaannya. Prosesnya adalah pertama pengambilan citra gambar dengan menggunakan kamera *smartphone* kemudian setelah gambar diambil citranya, citra dipotong besarnya sesuai dengan *background* yang berbentuk A4 sebagai luas referensinya. Langkah berikutnya adalah pemilihan citra gambar untuk dihitung luasnya yang kemudian di *threshold* untuk mengubah citra yang ada ke dalam warna putih-hitam. Langkah terakhir adalah penghitungan luas daun dengan menghitung piksel warna putih hasil *threshold* dengan faktor pengali pikselnya maka akan ditampilkan luas citra daun yang ada.

4.2 Analisis

Analisis merupakan salah satu bagian hal yang penting yang harus dilakukan untuk mengetahui apakah alat yang telah direncanakan mampu beroperasi dengan hal yang telah direncanakan dan diharapkan. Hal itu dapat dilihat dari hasil – hasil yang telah dicapai selama pengujian alat. Selain untuk mengetahui apakah alat sudah bekerja dengan baik sesuai dengan yang diharapkan, pengujian juga bertujuan untuk mengetahui kelebihan dan kekurangan dari alat yang dibuat. Hasil – hasil pengujian tersebut nantinya akan dianalisa agar dapat diketahui mengapa terjadi kekurangan. Pengujian pertama dilakukan secara terpisah, dalam artian pengujian tiap fungsi. Kemudian dilakukan pengujian secara keseluruhan ketika semua fungsi sudah disatukan. Pengujian yang telah dilakukan pada bab ini antara lain :

4.2.1 Pengujian Tiap Fungsi

- **Fungsi Untuk Mendapatkan Citra Gambar Melalui Kamera**

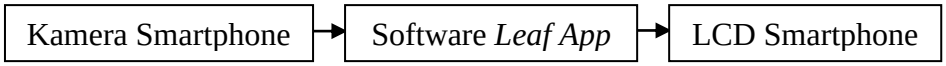
Pengujian ini adalah pengujian awal yang harus dilakukan untuk memastikan kamera dapat bekerja dan mengambil citra gambar.

Komponen yang terlibat :

- Smartphone Android Sony Xperia ST27i

- Pustaka OpenCV 2.4.9
- Kode Sumber *Class* "Application.java dan LabActivity.java"

Alur :



Gambar 4.1 Blok Pengujian Mendapatkan Citra Gambar

- Kamera Smartphone yang dipakai memiliki resolusi terbesar 320 x 240 piksel.
- Smartphone akan mendapatkan inputan dari kamera, diolah oleh software *Leaf App*, dan hasil ditampilkan ke LCD Smartphone.
- Kode Sumber Program *Class* ini adalah:

→ *file source code Application.java :*

```
package com.example.leafapp;

import java.io.File;
import java.util.List;

import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.provider.MediaStore.Images;
import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Intent;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.hardware.Camera.Parameters;
import android.hardware.Camera.Size;
import android.support.v7.app.ActionBarActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;
```

```
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import
org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.JavaCameraView;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.highgui.Highgui;
import org.opencv.imgproc.Imgproc;

// Use the deprecated Camera class.
@SuppressWarnings("deprecation")
public final class Application extends ActionBarActivity
implements CvCameraViewListener2 {

    // A tag for log output.
    private static final String TAG =
            MainActivity.class.getSimpleName();

    // A key for storing the index of the active camera.
    private static final String STATE_CAMERA_INDEX =
"cameraIndex";

    // A key for storing the index of the active image size.
    private static final String STATE_IMAGE_SIZE_INDEX =
            "imageSizeIndex";

    // An ID for items in the image size submenu.
    private static final int MENU_GROUP_ID_SIZE = 2;

    // The index of the active camera.
    private int mCameraIndex;

    // The index of the active image size.
```

```
    private int mImageSizeIndex;

    // Whether the active camera is front-facing.
    // If so, the camera view should be mirrored.
    private boolean mIsCameraFrontFacing;

    // The number of cameras on the device.
    private int mNumCameras;

    // The image sizes supported by the active camera.
    private List<Size> mSupportedImageSizes;

    // The camera view.
    private CameraBridgeViewBase mCameraView;

    // Whether the next camera frame should be saved as a photo.
    private boolean mIsPhotoPending;

    // A matrix that is used when saving photos.
    private Mat mBgr;

    // Whether an asynchronous menu action is in progress.
    // If so, menu interaction should be disabled.
    private boolean mIsMenuLocked;

    // The OpenCV loader callback.
    private BaseLoaderCallback mLoaderCallback =
            new BaseLoaderCallback(this) {
                @Override
                public void onManagerConnected(final int status) {
                    switch (status) {
                        case LoaderCallbackInterface.SUCCESS:
                            Log.d(TAG, "OpenCV loaded
successfully");
                            mCameraView.enableView();
                            //mCameraView.enableFpsMeter();
                            mBgr = new Mat();
                            break;
```

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

// Suppress backward incompatibility errors because we provide
// backward-compatible fallbacks.
@SuppressLint("NewApi")
@Override
protected void onCreate(final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    final Window window = getWindow();
    window.addFlags(
            WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    if (savedInstanceState != null) {
        mCameraIndex = savedInstanceState.getInt(
                STATE_CAMERA_INDEX, 0);
        mImageSizeIndex = savedInstanceState.getInt(
                STATE_IMAGE_SIZE_INDEX, 0);
    } else {
        mCameraIndex = 0;
        mImageSizeIndex = 0;
    }

    final Camera camera;
    if (Build.VERSION.SDK_INT >=
            Build.VERSION_CODES.GINGERBREAD) {
        CameraInfo cameraInfo = new CameraInfo();
        Camera.getCameraInfo(mCameraIndex, cameraInfo);
        mIsCameraFrontFacing =
                (cameraInfo.facing ==
                        CameraInfo.CAMERA_FACING_FRONT);
        mNumCameras = Camera.getNumberOfCameras();
        camera = Camera.open(mCameraIndex);
```

```
        } else { // pre-Gingerbread
            // Assume there is only 1 camera and it is rear-
facing.
            mIsCameraFrontFacing = false;
            mNumCameras = 1;
            camera = Camera.open();
        }
        final Parameters parameters = camera.getParameters();
        camera.release();
        mSupportedImageSizes =
                parameters.getSupportedPreviewSizes();
        final Size size =
mSupportedImageSizes.get(mImageSizeIndex);

        mCameraView = new JavaCameraView(this, mCameraIndex);
        mCameraView.setMaxFrameSize(size.width, size.height);
        mCameraView.setCvCameraViewListener(this);
        setContentView(mCameraView);
    }

    public void onSaveInstanceState(Bundle savedInstanceState) {
        // Save the current camera index.
        savedInstanceState.putInt(STATE_CAMERA_INDEX,
mCameraIndex);

        // Save the current image size index.
        savedInstanceState.putInt(STATE_IMAGE_SIZE_INDEX,
                mImageSizeIndex);

        super.onSaveInstanceState(savedInstanceState);
    }

    // Suppress backward incompatibility errors because we provide
    // backward-compatible fallbacks.
    @SuppressLint("NewApi")
    @Override
    public void recreate() {
        if (Build.VERSION.SDK_INT >=
```

```
            Build.VERSION_CODES.HONEYCOMB) {
        super.recreate();
    } else {
        finish();
        startActivity(getIntent());
    }
}

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

@Override
public void onResume() {
    super.onResume();
    OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_9,
            this, mLoaderCallback);
    mIsMenuLocked = false;
}

@Override
public void onDestroy() {
    if (mCameraView != null) {
        mCameraView.disableView();
    }
    super.onDestroy();
}

@Override
public boolean onCreateOptionsMenu(final Menu menu) {
    getMenuInflater().inflate(R.menu.activity_camera, menu);
    int numSupportedImageSizes = mSupportedImageSizes.size();
    if (numSupportedImageSizes > 1) {
        final SubMenu sizeSubMenu =
```

```
menu.addSubMenu(R.string.menu_image_size);
            for (int i = 0; i < numSupportedImageSizes; i++) {
                final Size size = mSupportedImageSizes.get(i);
                sizeSubMenu.add(MENU_GROUP_ID_SIZE, i, Menu.NONE,
                        String.format("%dx%d", size.width,
                                size.height));
            }
        }
        return true;
    }

    // Suppress backward incompatibility errors because we provide
    // backward-compatible fallbacks (for recreate).
    @SuppressLint("NewApi")
    @Override
    public boolean onOptionsItemSelected(final MenuItem item) {
        if (mIsMenuLocked) {
            return true;
        }
        if (item.getGroupId() == MENU_GROUP_ID_SIZE) {
            mImageSizeIndex = item.getItemId();
            recreate();

            return true;
        }
        switch (item.getItemId()) {
            case R.id.menu_take_photo:
                mIsMenuLocked = true;

                // Next frame, take the photo.
                mIsPhotoPending = true;

                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

```
    @Override
    public void onCameraViewStarted(final int width,
                                    final int height) {
    }

    @Override
    public void onCameraViewStopped() {
    }

    @Override
    public Mat onCameraFrame(final CvCameraViewFrame inputFrame) {
        final Mat rgba = inputFrame.rgba();

        if (mIsPhotoPending) {
            mIsPhotoPending = false;
            takePhoto(rgba);
        }

        if (mIsCameraFrontFacing) {
            // Mirror (horizontally flip) the preview.
            Core.flip(rgba, rgba, 1);
        }

        return rgba;
    }

    private void takePhoto(final Mat rgba) {

        // Determine the path and metadata for the photo.
        final long currentTimeMillis = System.currentTimeMillis();
        final String appName = getString(R.string.app_name);
        final String galleryPath =
                Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY_PICTURES).toString();
        final String albumPath = galleryPath + File.separator +
                appName;
        final String photoPath = albumPath + File.separator +
```

```
                currentTimeMillis +
LabActivity.PHOTO_FILE_EXTENSION;
        final ContentValues values = new ContentValues();
        values.put(MediaStore.MediaColumns.DATA, photoPath);
        values.put(Images.Media.MIME_TYPE,
                LabActivity.PHOTO_MIME_TYPE);
        values.put(Images.Media.TITLE, appName);
        values.put(Images.Media.DESCRIPTION, appName);
        values.put(Images.Media.DATE_TAKEN, currentTimeMillis);
        // Ensure that the album directory exists.
        File album = new File(albumPath);
        if (!album.isDirectory() && !album.mkdirs()) {
            Log.e(TAG, "Failed to create album directory at " +
                    albumPath);
            onTakePhotoFailed();
            return;
        }

        // Try to create the photo.
        Imgproc.cvtColor(rgba, mBgr, Imgproc.COLOR_RGBA2BGR, 3);
        if (!Highgui.imwrite(photoPath, mBgr)) {
            Log.e(TAG, "Failed to save photo to " + photoPath);
            onTakePhotoFailed();
        }
        Log.d(TAG, "Photo saved successfully to " + photoPath);

        // Try to insert the photo into the MediaStore.
        Uri uri;
        try {
            uri = getContentResolver().insert(
                    Images.Media.EXTERNAL_CONTENT_URI, values);
        } catch (final Exception e) {
            Log.e(TAG, "Failed to insert photo into MediaStore");
            e.printStackTrace();

            // Since the insertion failed, delete the photo.
            File photo = new File(photoPath);
            if (!photo.delete()) {
```

```
                Log.e(TAG, "Failed to delete non-inserted photo");
            }

            onTakePhotoFailed();
            return;
        }

        // Open the photo in LabActivity.
        final Intent intent = new Intent(this, LabActivity.class);
        intent.putExtra(LabActivity.EXTRA_PHOTO_URI, uri);
        intent.putExtra(LabActivity.EXTRA_PHOTO_DATA_PATH,
                photoPath);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                startActivity(intent);
            }
        });
    }

    private void onTakePhotoFailed() {
        mIsMenuLocked = false;

        // Show an error message.
        final String errorMessage =
                getString(R.string.photo_error_message);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(Application.this, errorMessage,
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

→ *file source code LabActivity.java :*

```
package com.example.leafapp;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.provider.MediaStore.Images;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageView;

public final class LabActivity extends ActionBarActivity {

    public static final String PHOTO_FILE_EXTENSION = ".png";
    public static final String PHOTO_MIME_TYPE = "image/png";

    public static final String EXTRA_PHOTO_URI =
            "com.nummist.secondsight.LabActivity.extra.PHOTO_URI";
    public static final String EXTRA_PHOTO_DATA_PATH =

"com.nummist.secondsight.LabActivity.extra.PHOTO_DATA_PATH";

    private Uri mUri;
    private String mDataPath;

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        final Intent intent = getIntent();
        mUri = intent.getParcelableExtra(EXTRA_PHOTO_URI);
        mDataPath = intent.getStringExtra(EXTRA_PHOTO_DATA_PATH);
```

```
        final ImageView imageView = new ImageView(this);
        imageView.setImageURI(mUri);

        setContentView(imageView);
    }

    @Override
    public boolean onCreateOptionsMenu(final Menu menu) {
        getMenuInflater().inflate(R.menu.activity_lab, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(final MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_delete:
                deletePhoto();
                return true;
            case R.id.menu_edit:
                editPhoto();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    /*
     * Show a confirmation dialog. On confirmation ("Delete"), the
     * photo is deleted and the activity finishes.
     */
    private void deletePhoto() {
        final AlertDialog.Builder alert = new AlertDialog.Builder(
                LabActivity.this);
        alert.setTitle(R.string.photo_delete_prompt_title);
        alert.setMessage(R.string.photo_delete_prompt_message);
        alert.setCancelable(false);
        alert.setPositiveButton(R.string.delete,
                new DialogInterface.OnClickListener() {
```

```
                    @Override
                    public void onClick(final DialogInterface
dialog,
                                        final int which) {
                        getContentResolver().delete(
                                Images.Media.EXTERNAL_CONTENT_URI,
                                MediaStore.MediaColumns.DATA +
"=?",
                                new String[]{mDataPath});
                        finish();
                    }
                });
        alert.setNegativeButton(android.R.string.cancel, null);
        alert.show();
    }

    /*
     * Show a chooser so that the user may pick an app for editing
     * the photo.
     */
    private void editPhoto() {
        final Intent intent = new Intent(Intent.ACTION_EDIT);
        intent.setDataAndType(mUri, PHOTO_MIME_TYPE);
        startActivity(Intent.createChooser(intent,
                getString(R.string.photo_edit_chooser_title)));
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        finish();
    }
}
```

→ Jalankan Program dengan cara klik tombol "*Capture*" pada GUI program.

Hasil dan Analisa :

Pada LCD Smartphone akan ditampilkan hasil *capture* dari kamera smartphone berupa citra 2 dimensi. Gambar di bawah ini merupakan hasil sebelum diintegrasikan dengan fungsi lain.

Gambar 4.2 Pengambilan Citra dan Pemotongan Citra

Pada gambar di atas menampilkan hasil tangkapan dari kamera smartphone, jendela dengan nama "*Leaf App*" jendela kiri atas adalah *screenshot* tangkapan dari kamera smartphone pada saat fungsi kamera dijalankan, jendela kanan atas adalah hasil tangkapan dari kamera smartphone yang kemudian dapat di *edit* ataupun di hapus, jendela kiri bawah adalah tampilan pada saat akan melakukan pemotongan citra dengan menggunakan fungsi *edit*, dan jendela kanan bawah adalah hasil dari pemotongan citra yang sudah dilakukan proses pemotongan sebelumnya.

- **Fungsi Untuk Mendapatkan Gambar *Threshold* dan Luas Citra**

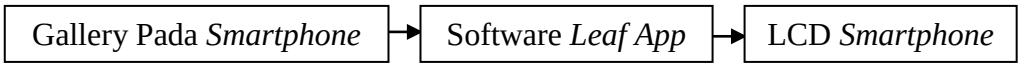
Pengujian ini untuk menguji apakah kode sumber mampu menghasilkan gambar *Threshold* dan menampilkan luas citra gambar.

Komponen yang terlibat :

- Smartphone Android Sony Xperia ST27i
- Pustaka OpenCV 2.4.9

- Kode Sumber *Class* "MainActivity.java"

Alur :



Gambar 4.3 Blok Pengujian Gambar Threshold dan Luas Citra

- Citra yang di ambil pada Gallery di Smartphone.
- Citra akan diolah oleh software *Leaf App*, dan hasil ditampilkan ke LCD Smartphone.
- Kode Sumber Program *Class* ini adalah:

→ *file source code MainActivity.java :*

```
package com.example.leafapp;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

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.imgproc.Imgproc;

import java.io.FileNotFoundException;

public class MainActivity extends Activity {
    Button bCapt, bPickCal;
    ImageView iv;
```

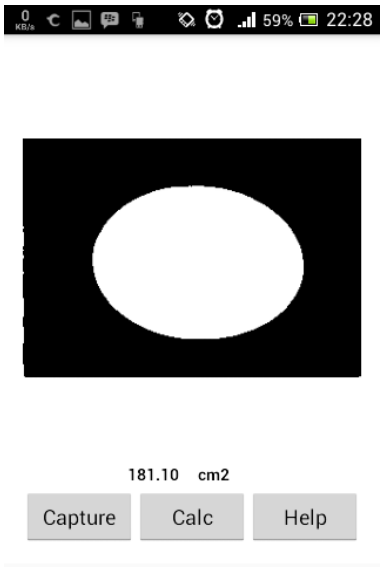
```
    Uri source;
    Bitmap bitmap, bitmapMaster;
    TextView Result;

    private static final int LOAD_IMAGE = 100;

    private static final String TAG = "MainActivity";

    static {
        if (!OpenCVLoader.initDebug()) {
            Log.d(TAG, "OpenCV not loaded");
        } else {
            Log.d(TAG, "OpenCV loaded");
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bCapt = (Button) findViewById(R.id.bCapture);
        bPickCal = (Button) findViewById(R.id.bPick);
        iv = (ImageView) findViewById(R.id.imageView);
        Result = (TextView) findViewById(R.id.source);
        bitmap = BitmapFactory.decodeResource(this.getResources(),
R.id.imageView);


        bCapt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
Application.class);
                startActivity(intent);
            }
        });
        bPickCal.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(intent, LOAD_IMAGE);
            }
        });

    }


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

        if (resultCode == RESULT_OK) {
            switch (requestCode) {
                case LOAD_IMAGE:
                    source = data.getData();

                    try {
                        bitmapMaster =
BitmapFactory.decodeStream(getContentResolver().openInputStream(
                                source));
                        LoadBitmap(bitmapMaster);

                    } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    break;
            }
        }
    }

    private void LoadBitmap(Bitmap bitmap) {
        Mat mat = new Mat(bitmap.getWidth(), bitmap.getHeight(),
```

```
CvType.CV_8UC1);
        // Convert
        Utils.bitmapToMat(bitmap, mat);

        Mat gray = new Mat(bitmap.getWidth(), bitmap.getHeight(),
CvType.CV_8UC1);
        // Convert the color
        Imgproc.cvtColor(mat, gray, Imgproc.COLOR_RGB2GRAY);
        // Convert back to bitmap
        Mat purpose = new Mat(gray.rows(), gray.cols(),
gray.type());

        Imgproc.adaptiveThreshold(gray, purpose, 255,
Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY_INV, 255,
5);

        Utils.matToBitmap(purpose, bitmap);
        iv.setImageBitmap(bitmap);

        Mat m = new Mat();
        Core.extractChannel(purpose, m, 0);
        int n = Core.countNonZero(m);
        int integer = Integer.parseInt(String.valueOf(n));
        // A4 Area in cm2
        float a = 623.7f;
        //Pixel Value A4 Area in 226x320 resolution
        int r = 72320;
        double result = (a / r) * integer;
        Result.setText(String.valueOf(result));
    }
}
```

→ Jalankan Program dengan cara klik tombol "*Calc*" pada GUI program.

Hasil dan Analisa :

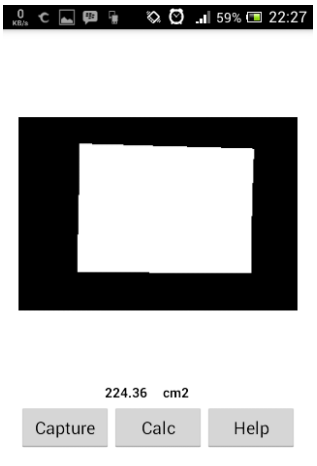


Gambar 4.4 Penggunaan *Threshold* Pada Citra

Ini adalah hasil ketika program dijalankan, hasil menunjukkan perubahan pada citra yang semula berwarna hijau berubah menjadi warna putih yang disebabkan digunakannya fungsi *Threshold* pada citra. Setelah citra dirubah ke dalam warna hitam putih, citra kemudian dihitung jumlah nilai piksel yang berwarna putih yang dikalikan dengan faktor pengkali pikselnya sehingga di dapatkan nilai luasan dari citra.

4.3 Pengukuran Luas Citra Terhadap Objek Citra

1. Pengukuran Luas Citra Bentuk Persegi Panjang

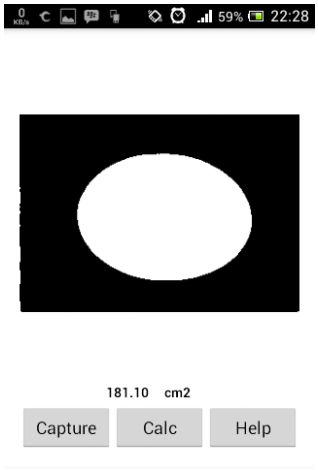


Tabel 4.1 Pengukuran Citra Persegi Panjang

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	234,0	234,7
2	234,0	234,1
3	234,0	235,3

Gambar 4.5 Hasil *Threshold* pada Bentuk Persegi Panjang

2. Pengukuran Luas Citra Bentuk *Ellips*

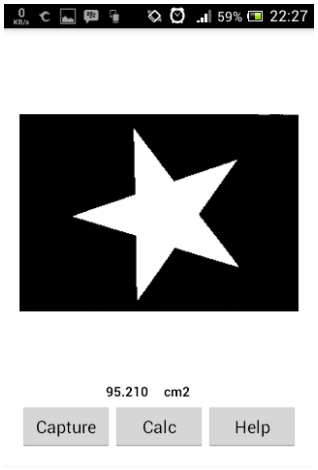


Tabel 4.2 Pengukuran Citra *Ellips*

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	183,8	180,7
2	183,8	180,8
3	183,8	188,2

Gambar 4.6 Hasil *Threshold* pada Bentuk *Ellips*

3. Pengukuran Luas Citra Bentuk Bintang

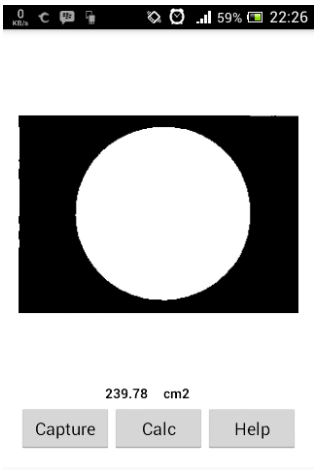


Tabel 4.3 Pengukuran Citra Bintang

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	100,1	101,5
2	100,1	101,8
3	100,1	101,2

Gambar 4.7 Hasil *Threshold* pada Bentuk Bintang

4. Pengukuran Luas Citra Bentuk Lingkaran

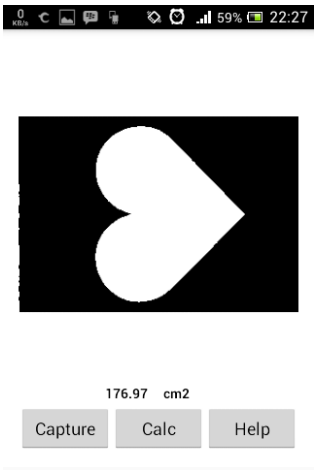


Tabel 4.4 Pengukuran Citra Lingkaran

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	254,5	251,2
2	254,5	256,5
3	254,5	256,1

Gambar 4.8 Hasil *Threshold* pada Bentuk Lingkaran

5. Pengukuran Luas Citra Bentuk Waru

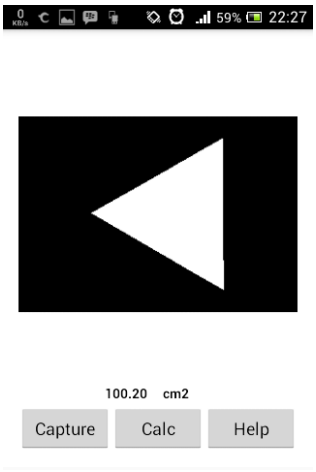


Tabel 4.5 Pengukuran Citra Waru

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	177,3	175,1
2	177,3	176,7
3	177,3	174,3

Gambar 4.9 Hasil *Threshold* pada Bentuk Waru

6. Pengukuran Luas Citra Bentuk Segitiga

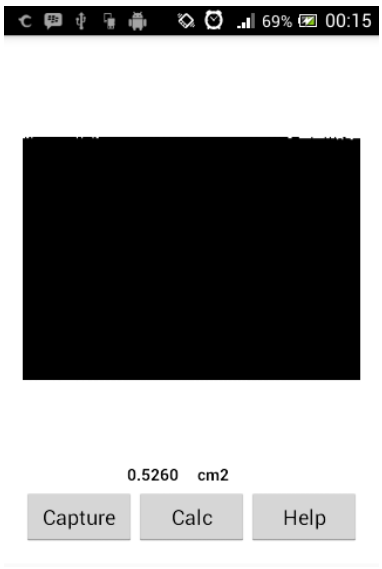


Tabel 4.6 Pengukuran Citra Segitiga

No.	Manual (Cm2)	*Leaf App* (Cm2)
1	105,4	104,5
2	105,4	103,8
3	105,4	103,5

Gambar 4.10 Hasil *Threshold* pada Bentuk Segitiga

7. Pengukuran Luas Citra Dengan Luas A4



Gambar 4.11 Hasil *Threshold* pada Luas A4

Ini adalah hasil ketika program dijalankan pada bentuk bidang yang memiliki luas area sama dengan luas A4 yang digunakan sebagai referensi. Dari hasil menunjukkan bahwa dalam pengukuran ini, referensi A4 sebagai alas pada saat pengambilan memiliki sebagai bidang yang dikenai fungsi *threshold* sehingga luas area objek yang akan dihitung tidak diperbolehkan sama dengan sama luas A4 yang berfungsi sebagai bidang referensinya agar pengukuran dapat dilakukan.

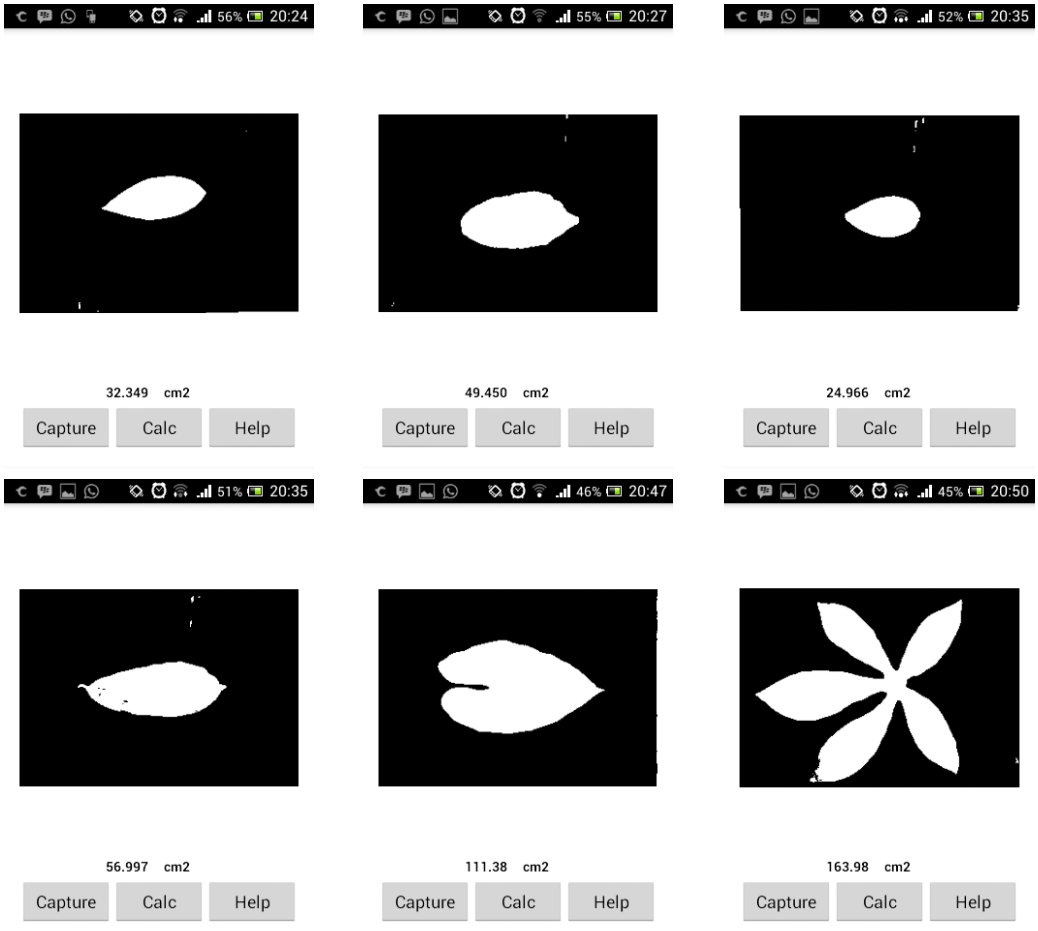
Tabel 4.7 Data Pengukuran Luas Citra Terhadap Objek Citra

Bentuk		**Luas**		**Perbedaan Luas (Manual – *Leaf App*)**	**Kesalahan(%)**
		Manual (cm2)	***Leaf App* (cm2)**		
Persegi Panjang	Besar	468	434,6	33,4	7,14
	Sedang	234	234,7	0,7	0,3
	Kecil	117	119,7	2,7	2,3
Ellips	Besar	367,6	362,6	5	1,36
	Sedang	183,8	183,2	0,5	0,27
	Kecil	91,9	94,2	2,3	2,5
Bintang	Besar	100,17	101,5	1,4	1,39
	Sedang	56,3	56,5	0,3	0,53
	Kecil	25,0	25,4	0,4	1,6
Lingkaran	Besar	254,5	254,6	0,1	0,04
	Sedang	143,1	145,6	2,5	1,74
	Kecil	63,6	64,9	1,3	2,04
Waru	Besar	177,3	175,4	1,9	1,07
	Sedang	99,7	101.9	2,2	2,2
	Kecil	44,3	45,4	1,1	2,48
Segitiga	Besar	140,4	139,2	1,2	0,85
	Sedang	105,4	103,9	1,5	1,42
	Kecil	35,1	36,5	1,4	3,99

Pada tabel di atas, terdapat 5 kolom, kolom pertama adalah bentuk citra bidang datar yang digunakan, kolom kedua adalah data luas sesungguhnya dari citra bidang datar, kolom ketiga adalah hasil pengukuran yang didapat dari penghitungan citra bidang datar menggunakan software *Leaf App*, selanjutnya adalah kolom selisih luas sesungguhnya dengan luas berdasarkan software, dan yang terakhir adalah nilai presentase nilai selisih luas. Pada pengambilan data ini nilai yang terdapat di kolom 3 adalah nilai yang muncul pada LCD Smartphone.

Software ini belum mampu mengukur luas suatu citra pada besaran milimeter, hanya mampu mengukur luas pada besaran sentimeter. Dalam penelitian ini, pengaruh cahaya pada saat pengambilan citra objek yang akan dihitung luasnya cukup mempengaruhi karena posisi saat pengambilan gambar yang akan mengakibatkan adanya bayangan pada hasil pengambilan citra yang berdampak pada kualitas hasil penghitungan citra. Sehingga kondisi cahaya yang cukup dan posisi pengambilan citra harus diperhatikan.

4.4 Pengujian Aplikasi Terhadap Citra Daun



Gambar 4.12 Hasil *Threshold* pada Beberapa Citra Daun

Pada hasil pengukuran beberapa citra daun diatas didapatkan hasil memiliki tingkat akurasi yang cukup. Tingkat akurasi ini didapat dengan melihat dari hasil pengukuran yang dilakukan menggunakan beberapa macam bentuk bidang datar yang telah dilakukan.

4.5 Prosedur Penempatan Objek yang akan dihitung

1. Siapkan selembar kertas dengan ukuran A4.
2. Letakkan objek yang akan dihitung diatas kertas A4.
3. Kondisi lingkungan pada objek yang akan dihitung harus terpapar dengan kondisi cahaya yang cukup(sinar matahari atau cahaya lampu).
4. Letakkan posisi *smartphone* pada jarak kurang lebih 32 cm di atas permukaan objek yang akan dihitung.

4.6 Prosedur Penggunaan Aplikasi Perhitungan Luas Daun

1. Tekan Tombol "*Capture*" untuk mengambil citra.
2. Setelah muncul fungsi kamera, atur kamera sedekat mungkin dengan luasan kertas A4.
3. Kemudian tekan tombol "*Take Photo*".
4. Kemudian akan muncul pilihan "*Delete*" dan "*Edit*". Pilih "*Delete*" jika ingin mengambil citra yang berbeda dan pilih "*Edit*" memotong citra.
5. Pilih "*Save*" untuk menyimpan citra.
6. Tekan tombol "*Calc*" untuk memilih citra yang akan dihitung luasnya.