LAMPIRAN

Lampiran 1 Source Code

MainActivity.java

```
package com.example.text_to_speech;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Rect;
import android.graphics.RectF;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.os.Environment;
import android.os.Handler;
import android.os.Looper;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.Frame;
import com.google.android.gms.vision.text.TextBlock;
```

```
import com.google.android.gms.vision.text.TextRecognizer;
import com.googlecode.tesseract.android.TessBaseAPI;

import org.opencv.android.BaseLoaderCallback;
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.imgproc.Imgproc;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Locale;

public class MainActivity extends AppCompatActivity{
    public static MainActivity instance;
    StringBuilder word = new StringBuilder();

    static {
        if (!OpenCVLoader.initDebug()){

        }
    }

    private TessBaseAPI tessBaseAPI;
    private      static      final      String      DATA_PATH      =
Environment.getExternalStorageDirectory().toString();

    public Mat mat;
    public Mat matCrop, matTreshold;
    public Mat matACapital, matA, matBCapital, matB, matCCapital,
matC, matDCapital, matD, matECapital
            ,   matE,   matFCapital,   matF,   matGCapital,   matG,
matHCapital, matH, matICapital, matI, matJCapital
```

```
            ,   matJ,   matKCapital,   matK,   matLCapital,   matL,
matMCapital, matM, matNCapital, matN, matOCapital
            ,   matO,   matPCapital,   matP,   matQCapital,   matQ,
matRCapital, matR, matSCapital, matS, matTCapital
            ,   matT,   matUCapital,   matU,   matVCapital,   matV,
matWCapital, matW, matXCapital, matX, matYCapital
            , matY, matZCapital, matZ, mat1, mat2, mat3, mat4, mat5,
mat6, mat7, mat8, mat9, mat0;

    Bitmap aCapital, a, bCapital, b, cCapital, c, dCapital, d,
eCapital, e, fCapital, f, gCapital, g, hCapital, h, iCapital, i,
jCapital, j, kCapital
    , k, lCapital, l, mCapital, m, nCapital, n, oCapital, o, pCapital,
p, qCapital, q, rCapital, r, sCapital, s, tCapital, t, uCapital, u,
vCapital
    , v, wCapital, w, xCapital, x, yCapital, y, zCapital, z, satu,
dua, tiga, empat, lima, enam, tujuh, delapan, sembilan, nol;

    SurfaceView cameraView;
    ImageView preview;
    CameraSource cameraSource;
    final int RequestCameraPermissionID = 1001;
    String huruf = "", huruf2 = "";
    TextBlock niel2, niel;
    String character = "", characterTemplate = "";
    TextToSpeech tts;
    private String stringResult = null;
    double nielresult = 0;

    private boolean safeToTakePicture = false;

    private     BaseLoaderCallback     mLoaderCallback     =     new
BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status){
                case LoaderCallbackInterface.SUCCESS: {
                    Log.i("OpenCV   status",   "Open   CV   loaded
sucessfully");
                    mat = new Mat();
                    matCrop = new Mat();
```

```
                    matTreshold = new Mat();
                    matACapital = new Mat();
                    matA = new Mat();
                    matBCapital = new Mat();
                    matB = new Mat();
                    matCCapital = new Mat();
                    matC = new Mat();
                    matDCapital = new Mat();
                    matD = new Mat();
                    matECapital = new Mat();
                    matE = new Mat();
                    matFCapital = new Mat();
                    matF = new Mat();
                    matGCapital = new Mat();
                    matG = new Mat();
                    matHCapital = new Mat();
                    matH = new Mat();
                    matICapital = new Mat();
                    matI = new Mat();
                    matJCapital = new Mat();
                    matJ = new Mat();
                    matKCapital = new Mat();
                    matK = new Mat();
                    matLCapital = new Mat();
                    matL = new Mat();
                    matMCapital = new Mat();
                    matM = new Mat();
                    matNCapital = new Mat();
                    matN = new Mat();
                    matOCapital = new Mat();
                    matO = new Mat();
                    matPCapital = new Mat();
                    matP = new Mat();
                    matQCapital = new Mat();
                    matQ = new Mat();
                    matRCapital = new Mat();
                    matR = new Mat();
                    matSCapital = new Mat();
                    matS = new Mat();
                    matTCapital = new Mat();
                    matT = new Mat();
```

```
                    matUCapital = new Mat();
                    matU = new Mat();
                    matVCapital = new Mat();
                    matV = new Mat();
                    matWCapital = new Mat();
                    matW = new Mat();
                    matXCapital = new Mat();
                    matX = new Mat();
                    matYCapital = new Mat();
                    matY = new Mat();
                    matZCapital = new Mat();
                    matZ = new Mat();
                    mat0 = new Mat();
                    mat1 = new Mat();
                    mat2 = new Mat();
                    mat3 = new Mat();
                    mat4 = new Mat();
                    mat5 = new Mat();
                    mat6 = new Mat();
                    mat7 = new Mat();
                    mat8 = new Mat();
                    mat9 = new Mat();
                } break;
                default: {
                    super.onManagerConnected(status);
                } break;
            }
        }
    };

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permission, @NonNull int[] grantResults) {
        switch (requestCode) {
            case RequestCameraPermissionID: {
                if                (grantResults[0]                ==
PackageManager.PERMISSION_GRANTED) {
                    if     (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
                        return;
                    }
```

```
                    try {
                        cameraSource.start(cameraView.getHolder());
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            break;
        }
    }

        @Override
    protected void onCreate (Bundle savedInstanceState){
        if (!OpenCVLoader.initDebug()) {
            Log.d("OpenCV", "Internal OpenCV library not found. Using
OpenCV Manager for initialization");
            //OpenCVLoader.initAsync
(OpenCVLoader.OPENCV_VERSION_3_0_0.this, mLoaderCallback);
        } else {
            Log.d("OpenCV", "OpenCV library found inside package.
Using it!");

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }

        super.onCreate (savedInstanceState);
        setContentView (R.layout.activity_main);
        aCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.a_capital4).copy(Bitmap.Config.ARGB_8888, true);
        a       =       BitmapFactory.decodeResource(getResources(),
R.drawable.a4).copy(Bitmap.Config.ARGB_8888, true);
        bCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.b_capital4).copy(Bitmap.Config.ARGB_8888, true);
        b       =       BitmapFactory.decodeResource(getResources(),
R.drawable.b4).copy(Bitmap.Config.ARGB_8888, true);
        cCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.c_capital4).copy(Bitmap.Config.ARGB_8888, true);
        c       =       BitmapFactory.decodeResource(getResources(),
R.drawable.c4).copy(Bitmap.Config.ARGB_8888, true);
        dCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.d_capital4).copy(Bitmap.Config.ARGB_8888, true);
```

```
        d       =       BitmapFactory.decodeResource(getResources(),
R.drawable.d4).copy(Bitmap.Config.ARGB_8888, true);
        eCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.e_capital4).copy(Bitmap.Config.ARGB_8888, true);
        e       =       BitmapFactory.decodeResource(getResources(),
R.drawable.e4).copy(Bitmap.Config.ARGB_8888, true);
        fCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.f_capital4).copy(Bitmap.Config.ARGB_8888, true);
        f       =       BitmapFactory.decodeResource(getResources(),
R.drawable.f4).copy(Bitmap.Config.ARGB_8888, true);
        gCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.g_capital4).copy(Bitmap.Config.ARGB_8888, true);
        g       =       BitmapFactory.decodeResource(getResources(),
R.drawable.g4).copy(Bitmap.Config.ARGB_8888, true);
        hCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.h_capital4).copy(Bitmap.Config.ARGB_8888, true);
        h       =       BitmapFactory.decodeResource(getResources(),
R.drawable.h4).copy(Bitmap.Config.ARGB_8888, true);
        iCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.i_capital4).copy(Bitmap.Config.ARGB_8888, true);
        i       =       BitmapFactory.decodeResource(getResources(),
R.drawable.i4).copy(Bitmap.Config.ARGB_8888, true);
        jCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.j_capital4).copy(Bitmap.Config.ARGB_8888, true);
        j       =       BitmapFactory.decodeResource(getResources(),
R.drawable.j4).copy(Bitmap.Config.ARGB_8888, true);
        kCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.k_capital4).copy(Bitmap.Config.ARGB_8888, true);
        k       =       BitmapFactory.decodeResource(getResources(),
R.drawable.k4).copy(Bitmap.Config.ARGB_8888, true);
        lCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.l_capital4).copy(Bitmap.Config.ARGB_8888, true);
        l       =       BitmapFactory.decodeResource(getResources(),
R.drawable.l4).copy(Bitmap.Config.ARGB_8888, true);
        mCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.m_kapital4).copy(Bitmap.Config.ARGB_8888, true);
        m       =       BitmapFactory.decodeResource(getResources(),
R.drawable.m4).copy(Bitmap.Config.ARGB_8888, true);
        nCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.n_capital4).copy(Bitmap.Config.ARGB_8888, true);
```

```
        n       =       BitmapFactory.decodeResource(getResources(),
R.drawable.n4).copy(Bitmap.Config.ARGB_8888, true);
        oCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.o_capital4).copy(Bitmap.Config.ARGB_8888, true);
        o       =       BitmapFactory.decodeResource(getResources(),
R.drawable.o4).copy(Bitmap.Config.ARGB_8888, true);
        pCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.p_capital4).copy(Bitmap.Config.ARGB_8888, true);
        p       =       BitmapFactory.decodeResource(getResources(),
R.drawable.p4).copy(Bitmap.Config.ARGB_8888, true);
        qCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.q_capital4).copy(Bitmap.Config.ARGB_8888, true);
        q       =       BitmapFactory.decodeResource(getResources(),
R.drawable.q4).copy(Bitmap.Config.ARGB_8888, true);
        rCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.r_capital4).copy(Bitmap.Config.ARGB_8888, true);
        r       =       BitmapFactory.decodeResource(getResources(),
R.drawable.r4).copy(Bitmap.Config.ARGB_8888, true);
        sCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.s_capital4).copy(Bitmap.Config.ARGB_8888, true);
        s       =       BitmapFactory.decodeResource(getResources(),
R.drawable.s4).copy(Bitmap.Config.ARGB_8888, true);
        tCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.t_capital4).copy(Bitmap.Config.ARGB_8888, true);
        t       =       BitmapFactory.decodeResource(getResources(),
R.drawable.t4).copy(Bitmap.Config.ARGB_8888, true);
        uCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.u_capital4).copy(Bitmap.Config.ARGB_8888, true);
        u       =       BitmapFactory.decodeResource(getResources(),
R.drawable.u4).copy(Bitmap.Config.ARGB_8888, true);
        vCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.v_capital4).copy(Bitmap.Config.ARGB_8888, true);
        v       =       BitmapFactory.decodeResource(getResources(),
R.drawable.v4).copy(Bitmap.Config.ARGB_8888, true);
        wCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.w_capital4).copy(Bitmap.Config.ARGB_8888, true);
        w       =       BitmapFactory.decodeResource(getResources(),
R.drawable.w4).copy(Bitmap.Config.ARGB_8888, true);
        xCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.x_capital4).copy(Bitmap.Config.ARGB_8888, true);
```

```
        x       =       BitmapFactory.decodeResource(getResources(),
R.drawable.x4).copy(Bitmap.Config.ARGB_8888, true);
        yCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.y_capital4).copy(Bitmap.Config.ARGB_8888, true);
        y       =       BitmapFactory.decodeResource(getResources(),
R.drawable.y4).copy(Bitmap.Config.ARGB_8888, true);
        zCapital    =    BitmapFactory.decodeResource(getResources(),
R.drawable.z_capital4).copy(Bitmap.Config.ARGB_8888, true);
        z       =       BitmapFactory.decodeResource(getResources(),
R.drawable.z4).copy(Bitmap.Config.ARGB_8888, true);
        satu      =      BitmapFactory.decodeResource(getResources(),
R.drawable.satu).copy(Bitmap.Config.ARGB_8888, true);
        dua       =      BitmapFactory.decodeResource(getResources(),
R.drawable.dua).copy(Bitmap.Config.ARGB_8888, true);
        tiga      =      BitmapFactory.decodeResource(getResources(),
R.drawable.tiga).copy(Bitmap.Config.ARGB_8888, true);
        empat     =      BitmapFactory.decodeResource(getResources(),
R.drawable.empat).copy(Bitmap.Config.ARGB_8888, true);
        lima      =      BitmapFactory.decodeResource(getResources(),
R.drawable.lima).copy(Bitmap.Config.ARGB_8888, true);
        enam      =      BitmapFactory.decodeResource(getResources(),
R.drawable.enam).copy(Bitmap.Config.ARGB_8888, true);
        tujuh     =      BitmapFactory.decodeResource(getResources(),
R.drawable.tujuh).copy(Bitmap.Config.ARGB_8888, true);
        delapan    =    BitmapFactory.decodeResource(getResources(),
R.drawable.delapan).copy(Bitmap.Config.ARGB_8888, true);
        sembilan    =    BitmapFactory.decodeResource(getResources(),
R.drawable.sembilan).copy(Bitmap.Config.ARGB_8888, true);
        nol       =      BitmapFactory.decodeResource(getResources(),
R.drawable.nol).copy(Bitmap.Config.ARGB_8888, true);

        preview = findViewById(R.id.imageView);
        cameraView = findViewById(R.id.surfaceView);
        tts     =     new     TextToSpeech(MainActivity.this,     new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if (status == TextToSpeech.SUCCESS) {
                    int result = tts.setLanguage(Locale.US);
                    if (result == TextToSpeech.LANG_MISSING_DATA ||
result == TextToSpeech.LANG_NOT_SUPPORTED) {
```

```
                        Log.e("Error TTS", "This Language is not
supported");
                    } else {
                    }
                } else {
                    Log.e("Error TTS", "Initialization Failed!");
                }
            }
        });
        startocr();
    }

    public void startocr () {
        TextRecognizer textRecognizer = new
TextRecognizer.Builder(getApplicationContext()).build();
        if (!textRecognizer.isOperational()) {
            Log.w("MainActivity", "Detector dependencies are not yet
available");
        } else {
            cameraSource = new
CameraSource.Builder(getApplicationContext(),
textRecognizer).setFacing(CameraSource.CAMERA_FACING_BACK)
//            .setRequestedPreviewSize (1280, 1024)
//                    .setRequestedFps(15.0f)
                    .setAutoFocusEnabled(true).build();

            cameraView.getHolder().addCallback(new
SurfaceHolder.Callback() {
                @Override
                public void surfaceCreated(@NonNull SurfaceHolder
holder) {
                    try {
                        if
(ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.CAMERA}, RequestCameraPermissionID);
                            return;
                        }
                        cameraSource.start(cameraView.getHolder());
```

```
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                @Override
                public void surfaceChanged(@NonNull SurfaceHolder
holder, int format, int width, int height) {
                    safeToTakePicture = true;
                }

                @Override
                public void surfaceDestroyed(@NonNull SurfaceHolder
holder) {
                    if (cameraSource != null) {
                        cameraSource.release();
                        cameraSource = null;
                    }
                }
            });

            textRecognizer.setProcessor(new
Detector.Processor<TextBlock>() {
                @Override
                public void release() {

                }

                @Override
                public                                          void
receiveDetections(Detector.Detections<TextBlock> detections) {
                    SparseArray<TextBlock>       sparseArray       =
detections.getDetectedItems();
                    StringBuilder      stringBuilder      =      new
StringBuilder();

                    for (int i = 0; i < sparseArray.size(); ++i){
                        TextBlock            textBlock            =
sparseArray.valueAt(i);
                        if (textBlock != null && textBlock.getValue()
!= null){
```

```
stringBuilder.append(textBlock.getValue() + " ");
                        }
                    }

                    final        String        stringText        =
stringBuilder.toString();

                    Handler          handler          =          new
Handler(Looper.getMainLooper());
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            stringResult = stringText;
                            Log.d("Result ", stringResult);
                            if (safeToTakePicture) {
                                cameraSource.takePicture(null,   new
CameraSource.PictureCallback() {
                                    @Override
                                    public                       void
onPictureTaken(byte[] bytes) {
                                        cameraSource.release();
                                        cameraSource.stop();
                                        Log.d("Size         bites:",
String.valueOf(bytes.length));
                                        Bitmap       bitmap       =
BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
                                        prosess(bitmap);

                                        safeToTakePicture = true;
                                    }
                                });
                                safeToTakePicture = false;
                            }
                        }
                    });
                }
            });
        }
    }
```

```
    public void prosess(Bitmap bitmap){
        TextRecognizer txtRecognizer = new
TextRecognizer.Builder(getApplicationContext()).build();
        if (!txtRecognizer.isOperational()){

        } else {
            Frame frame = new
Frame.Builder().setBitmap(bitmap).setRotation(1).build();
            SparseArray items = txtRecognizer.detect(frame);
            Log.d("Status", "prosess");
            Log.d("Jmlh deteksi", String.valueOf(items.size()));
            if (items.size() == 0) {

                startocr();
                try {
                    if
(ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.CAMERA}, RequestCameraPermissionID);
                        return;
                    }
                    cameraSource.start(cameraView.getHolder());
                } catch (IOException e) {
                    e.printStackTrace();
                }
                Log.d("Status", "Kosong");
            } else {
                for (int i = 0; i < items.size(); i++){
                    TextBlock item = (TextBlock) items.valueAt(i);
                    huruf2 = item.getValue();
                    Object obj = huruf2.charAt(0);
                    character = item.getValue();
                    niel2 = item;
                    if (!huruf2.equals("")){
                        Toast.makeText(getApplicationContext(),
"Proses 2 success",Toast.LENGTH_LONG).show();
                        RectF rect = new
RectF(niel2.getBoundingBox());
                        Bitmap brotated;
```

```
                        float degrees = 90;//rotation degree
                        Matrix matrix = new Matrix();
                        matrix.setRotate(degrees);
                        brotated = Bitmap.createBitmap(bitmap, 0, 0,
bitmap.getWidth(), bitmap.getHeight(), matrix, true);
                        if (i == items.size()-1){
                            String status = "Finish";
                            getCropBitmapByCPU(brotated,       rect,
status, niel2.getValue().length(), niel2.getValue());
                        } else {
                            String status = "Not yet";
                            getCropBitmapByCPU(brotated,       rect,
status, niel2.getValue().length(), niel2.getValue());
                        }
                    } else {
                        Toast.makeText(getApplicationContext(),
"Proses 2 Gagal", Toast.LENGTH_LONG).show();
                        //                    startocr();
                        Intent          a          =          new
Intent(getApplicationContext(), NextActivity.class);
                        a.putExtra("Bilangan", "");
                        startActivity(a);
                        finish();
                    }
                }
            }
        }
    }

    public void loopTemplate(Bitmap bmp32, Mat mat
            , int currentLoop
            , int maxLoop
            , String status
            , char karakter
    ){
        Log.d("Status", "Matching");
        int sizeCount = 0;
        String karakter2 = Character.toString(karakter);
        for (int i = 1; i <= 63; i++){
            if (i == 1) {
                Utils.bitmapToMat(aCapital, matACapital);
```

```
                Matching(matACapital, mat, Imgproc.TM_SQDIFF, "A");
            } else if (i == 2) {
                Utils.bitmapToMat(a, matA);
                Matching(matA, mat, Imgproc.TM_SQDIFF, "a");
            } else if (i == 3) {
                Utils.bitmapToMat(bCapital, matBCapital);
                Matching(matBCapital, mat, Imgproc.TM_SQDIFF, "B");
            } else if (i == 4) {
                Utils.bitmapToMat(b, matB);
                Matching(matB, mat, Imgproc.TM_SQDIFF, "b");
            } else if (i == 5) {
                Utils.bitmapToMat(cCapital, matCCapital);
                Matching(matCCapital, mat, Imgproc.TM_SQDIFF, "C");
            } else if (i == 6) {
                Utils.bitmapToMat(c, matC);
                Matching(matC, mat, Imgproc.TM_SQDIFF, "c");
            } else if (i == 7) {
                Utils.bitmapToMat(dCapital, matDCapital);
                Matching(matDCapital, mat, Imgproc.TM_SQDIFF, "D");
            } else if (i == 8) {
                Utils.bitmapToMat(d, matD);
                Matching(matD, mat, Imgproc.TM_SQDIFF, "d");
            } else if (i == 9) {
                Utils.bitmapToMat(eCapital, matECapital);
                Matching(matECapital, mat, Imgproc.TM_SQDIFF, "E");
            } else if (i == 10) {
                Utils.bitmapToMat(e, matE);
                Matching(matE, mat, Imgproc.TM_SQDIFF, "e");
            } else if (i == 11) {
                Utils.bitmapToMat(fCapital, matFCapital);
                Matching(matFCapital, mat, Imgproc.TM_SQDIFF, "F");
            } else if (i == 12) {
                Utils.bitmapToMat(f, matF);
                Matching(matF, mat, Imgproc.TM_SQDIFF, "f");
            } else if (i == 13) {
                Utils.bitmapToMat(gCapital, matGCapital);
                Matching(matGCapital, mat, Imgproc.TM_SQDIFF, "G");
            } else if (i == 14) {
                Utils.bitmapToMat(g, matG);
                Matching(matG, mat, Imgproc.TM_SQDIFF, "g");
            } else if (i == 15) {
```

```
                Utils.bitmapToMat(hCapital, matHCapital);
                Matching(matHCapital, mat, Imgproc.TM_SQDIFF, "H");
            } else if (i == 16) {
                Utils.bitmapToMat(h, matH);
                Matching(matH, mat, Imgproc.TM_SQDIFF, "h");
            } else if (i == 17) {
                Utils.bitmapToMat(iCapital, matICapital);
                Matching(matICapital, mat, Imgproc.TM_SQDIFF, "I");
            } else if (i == 18) {
                Utils.bitmapToMat(this.i, matI);
                Matching(matI, mat, Imgproc.TM_SQDIFF, "i");
            } else if (i == 19) {
                Utils.bitmapToMat(jCapital, matJCapital);
                Matching(matJCapital, mat, Imgproc.TM_SQDIFF, "J");
            } else if (i == 20) {
                Utils.bitmapToMat(j, matJ);
                Matching(matJ, mat, Imgproc.TM_SQDIFF, "j");
            } else if (i == 21) {
                Utils.bitmapToMat(kCapital, matKCapital);
                Matching(matKCapital, mat, Imgproc.TM_SQDIFF, "K");
            } else if (i == 22) {
                Utils.bitmapToMat(k, matK);
                Matching(matK, mat, Imgproc.TM_SQDIFF, "k");
            } else if (i == 23) {
                Utils.bitmapToMat(lCapital, matLCapital);
                Matching(matLCapital, mat, Imgproc.TM_SQDIFF, "L");
            } else if (i == 24) {
                Utils.bitmapToMat(l, matL);
                Matching(matL, mat, Imgproc.TM_SQDIFF, "l");
            } else if (i == 25) {
                Utils.bitmapToMat(mCapital, matMCapital);
                Matching(matMCapital, mat, Imgproc.TM_SQDIFF, "M");
            } else if (i == 26) {
                Utils.bitmapToMat(m, matM);
                Matching(matM, mat, Imgproc.TM_SQDIFF, "m");
            } else if (i == 27) {
                Utils.bitmapToMat(nCapital, matNCapital);
                Matching(matNCapital, mat, Imgproc.TM_SQDIFF, "N");
            } else if (i == 28) {
                Utils.bitmapToMat(n, matN);
                Matching(matN, mat, Imgproc.TM_SQDIFF, "n");
```

```
            } else if (i == 29) {
                Utils.bitmapToMat(oCapital, matOCapital);
                Matching(matOCapital, mat, Imgproc.TM_SQDIFF, "O");
            } else if (i == 30) {
                Utils.bitmapToMat(o, matO);
                Matching(matO, mat, Imgproc.TM_SQDIFF, "o");
            } else if (i == 31) {
                Utils.bitmapToMat(pCapital, matPCapital);
                Matching(matPCapital, mat, Imgproc.TM_SQDIFF, "P");
            } else if (i == 32) {
                Utils.bitmapToMat(p, matP);
                Matching(matP, mat, Imgproc.TM_SQDIFF, "p");
            } else if (i == 33) {
                Utils.bitmapToMat(qCapital, matQCapital);
                Matching(matQCapital, mat, Imgproc.TM_SQDIFF, "Q");
            } else if (i == 34) {
                Utils.bitmapToMat(q, matQ);
                Matching(matQ, mat, Imgproc.TM_SQDIFF, "q");
            } else if (i == 35) {
                Utils.bitmapToMat(rCapital, matRCapital);
                Matching(matRCapital, mat, Imgproc.TM_SQDIFF, "R");
            } else if (i == 36) {
                Utils.bitmapToMat(r, matR);
                Matching(matR, mat, Imgproc.TM_SQDIFF, "r");
            } else if (i == 37) {
                Utils.bitmapToMat(sCapital, matSCapital);
                Matching(matSCapital, mat, Imgproc.TM_SQDIFF, "S");
            } else if (i == 38) {
                Utils.bitmapToMat(s, matS);
                Matching(matS, mat, Imgproc.TM_SQDIFF, "s");
            } else if (i == 39) {
                Utils.bitmapToMat(tCapital, matTCapital);
                Matching(matTCapital, mat, Imgproc.TM_SQDIFF, "T");
            } else if (i == 40) {
                Utils.bitmapToMat(t, matT);
                Matching(matT, mat, Imgproc.TM_SQDIFF, "t");
            } else if (i == 41) {
                Utils.bitmapToMat(uCapital, matUCapital);
                Matching(matUCapital, mat, Imgproc.TM_SQDIFF, "U");
            } else if (i == 42) {
                Utils.bitmapToMat(u, matU);
```

```
                Matching(matU, mat, Imgproc.TM_SQDIFF, "u");
            } else if (i == 43) {
                Utils.bitmapToMat(vCapital, matVCapital);
                Matching(matVCapital, mat, Imgproc.TM_SQDIFF, "V");
            } else if (i == 44) {
                Utils.bitmapToMat(v, matV);
                Matching(matV, mat, Imgproc.TM_SQDIFF, "v");
            } else if (i == 45) {
                Utils.bitmapToMat(wCapital, matWCapital);
                Matching(matWCapital, mat, Imgproc.TM_SQDIFF, "W");
            } else if (i == 46) {
                Utils.bitmapToMat(w, matW);
                Matching(matW, mat, Imgproc.TM_SQDIFF, "w");
            } else if (i == 47) {
                Utils.bitmapToMat(xCapital, matXCapital);
                Matching(matXCapital, mat, Imgproc.TM_SQDIFF, "X");
            } else if (i == 48) {
                Utils.bitmapToMat(x, matX);
                Matching(matX, mat, Imgproc.TM_SQDIFF, "x");
            } else if (i == 49) {
                Utils.bitmapToMat(yCapital, matYCapital);
                Matching(matYCapital, mat, Imgproc.TM_SQDIFF, "Y");
            } else if (i == 50) {
                Utils.bitmapToMat(y, matY);
                Matching(matY, mat, Imgproc.TM_SQDIFF, "y");
            } else if (i == 51) {
                Utils.bitmapToMat(zCapital, matZCapital);
                Matching(matZCapital, mat, Imgproc.TM_SQDIFF, "Z");
            } else if (i == 52) {
                Utils.bitmapToMat(z, matZ);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "z");
            } else if (i == 53) {
                Utils.bitmapToMat(satu, mat1);
                Matching(mat1, mat, Imgproc.TM_SQDIFF, "1");
            } else if (i == 54) {
                Utils.bitmapToMat(dua, mat2);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "2");
            } else if (i == 55) {
                Utils.bitmapToMat(tiga, mat3);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "3");
            } else if (i == 56) {
```

```
                Utils.bitmapToMat(empat, mat4);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "4");
            } else if (i == 57) {
                Utils.bitmapToMat(lima, mat5);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "5");
            } else if (i == 58) {
                Utils.bitmapToMat(enam, mat6);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "6");
            } else if (i == 59) {
                Utils.bitmapToMat(tujuh, mat7);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "7");
            } else if (i == 60) {
                Utils.bitmapToMat(delapan, mat8);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "8");
            } else if (i == 61) {
                Utils.bitmapToMat(sembilan, mat9);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "9");
            } else if (i == 62) {
                Utils.bitmapToMat(nol, mat0);
                Matching(matZ, mat, Imgproc.TM_SQDIFF, "0");
            } else if (i == 63) {
                sizeCount++;
                if (karakter2.equals(characterTemplate)){
                    End(characterTemplate, currentLoop
                            , maxLoop, status
                            , bmp32);
                    nielresult = 0;
                } else {
                    End(karakter2, currentLoop
                            , maxLoop, status
                            , bmp32);
                    nielresult = 0;
                }
                character = "";
            }
        }
    }

    private
    void
//    Bitmap
```

```
    getCropBitmapByCPU(Bitmap source
            , RectF cropRectF
            , String status
            , int length
            , String karakter){
        Log.d("Status", "Crop (Textblock)");
        float top = cropRectF.top;
        float bottom = cropRectF.bottom;
        float left = cropRectF.left;
        float right = cropRectF.right;
        float tinggi = bottom-top;
        float panjang = right-left;
        float panjangBagi = panjang/length;
//
//        Log.d("Top", String.valueOf(top));
//        Log.d("Bottom", String.valueOf(bottom));
//        Log.d("Left", String.valueOf(left));
//        Log.d("Right", String.valueOf(right));
//        Log.d("Tinggi", String.valueOf(tinggi));
//        Log.d("Panjang", String.valueOf(panjang));
//        Log.d("Panjang", String.valueOf(length));
//        Log.d("Bagian", String.valueOf(panjangBagi));

        for (int i = 0; i < length; i++) {
//            Log.d("Width", String.valueOf(panjangBagi));
//            Log.d("Height", String.valueOf(tinggi));
            Bitmap resultBitmap = Bitmap.createBitmap((int)
(panjangBagi + 20), (int) (cropRectF.height() + 20),
Bitmap.Config.ARGB_8888);
            Canvas canvas = new Canvas(resultBitmap);

            Paint paint = new Paint(Paint.FILTER_BITMAP_FLAG);
            paint.setColor(Color.WHITE);
            canvas.drawRect(new RectF(0, 0, (panjangBagi + 20),
(cropRectF.height() + 20)), paint);

            Matrix matrix = new Matrix();
            matrix.postTranslate(-(cropRectF.left + (panjangBagi *
i)), -cropRectF.top);
            canvas.drawBitmap(source, matrix, paint);
                Bitmap tempResultBitmap = resultBitmap;
```

```
                tempResultBitmap = thresholdBiner(tempResultBitmap);
                tempResultBitmap   =   resizeImage(tempResultBitmap,
20);
                Bitmap                    bitmap32                   =
tempResultBitmap.copy(Bitmap.Config.ARGB_8888, true);
                Utils.bitmapToMat(bitmap32, mat);
                loopTemplate(bitmap32,   mat,   i,   length,   status,
karakter.charAt(i));
        }
    }

    public Bitmap thresholdBiner(Bitmap bitmap){
        Log.d("Status", "Threshold Biner");
        Mat  tmp  =  new  Mat(bitmap.getWidth(),  bitmap.getHeight(),
CvType.CV_8UC1);

        // Convert
        Utils.bitmapToMat(bitmap, tmp);
        Mat  gray  =  new  Mat(bitmap.getWidth(),  bitmap.getHeight(),
CvType.CV_8UC1);

        // Convert the color to gray
        Imgproc.cvtColor(tmp, gray, Imgproc.COLOR_RGB2GRAY);

        // Convert to bitmap
        Mat   destination   =   new   Mat(gray.rows(),   gray.cols(),
gray.type());
        Imgproc.threshold(tmp,      destination,      100,      255,
Imgproc.THRESH_BINARY);
        Utils.matToBitmap(destination, bitmap);
        return bitmap;
    }

    public static Bitmap resizeImage(Bitmap src, float newSize){
        Log.d("Status", "Resize");
        int maxImageSize = 20;
        float scale = 1;
        int new_width, new_height;

        scale = newSize / src.getHeight();
        Log.d("Scale", String.valueOf(scale));
```

```
        new_height = (int) newSize;
        new_width = (int) (src.getWidth() * scale);
        Log.d("new width", String.valueOf(new_width));
        Log.d("new height", String.valueOf(new_height));

//        if (new_height > maxImageSize){
//            scale = maxImageSize / src.getHeight();
//            new_height = maxImageSize;
//            new_width = (int) (src.getWidth() * scale);
//        }
//        Log.d("final width", String.valueOf(new_width));
//        Log.d("final height", String.valueOf(new_height));

        return Bitmap.createScaledBitmap(src, new_width, new_height,
true);
    }

    public void End(String character
            , int currentLoop
            , int maxLoop
            , String status
            , Bitmap bitmap){

        Log.d("Next?", status);
        Log.d("currentLoop", String.valueOf(currentLoop));
        Log.d("maxLoop", String.valueOf(maxLoop));
        if (currentLoop == (maxLoop-1) && status == "Finish"){
            Log.d("Status", "Finish");
            word.append(character);

            ByteArrayOutputStream        stream        =        new
ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);
            byte[] byteArray = stream.toByteArray();

            Intent    a    =    new    Intent(getApplicationContext(),
NextActivity.class);
            String finalWord = word.toString();
            a.putExtra("picture", byteArray);
            a.putExtra("word", finalWord);
            Log.d("Hasil AKhir", finalWord);
```

```
            startActivity(a);
            finish();
        } else {
            Log.d("Status", "Next character");
            word.append(character);
        }
    }

    public void Matching(Mat templ, Mat img, int match_method, String
hasil){
        // Create the result matrix
        int result_cols = templ.cols() - img.cols() + 1;
        int result_rows = templ.rows() - img.rows() + 1;
        Mat    result    =    new    Mat(result_rows,    result_cols,
CvType.CV_8UC3);
        // Do the Matching and Normalize
        Imgproc.matchTemplate(templ, templ, result, match_method);

        // Localizing the best match with minMaxLoc
        Core.MinMaxLocResult mmr = Core.minMaxLoc(result);

        org.opencv.core.Point matchLoc;
        if  (match_method  ==  Imgproc.TM_SQDIFF  ||  match_method  ==
Imgproc.TM_SQDIFF_NORMED) {
            matchLoc = mmr.minLoc;
            if (nielresult == 0) {
                nielresult = mmr.minVal;
                characterTemplate = hasil;
            } else if (nielresult > mmr.minVal){
                nielresult = mmr.minVal;
                characterTemplate = hasil;
            }
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
            Log.d("OpenCV", "Internal OpenCV library not found. Using
OpenCV Manager for initialization");
```

```
OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0,      this,
mLoaderCallback);
        } else {
            Log.d("OpenCV", "O" +
                    "penCV library found inside package. Using it!");

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }
}
```

NextActivity.java

```
package com.example.text_to_speech;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.example.text_to_speech.databinding.ActivityNextBinding;

import java.util.Locale;
```

```
public class NextActivity extends AppCompatActivity {

    private AppBarConfiguration appBarConfiguration;
    private ActivityNextBinding binding;
    private TextToSpeech textToSpeech;

    String kata = "";
    Button btn;

    TextView tvHasil;
    ImageView imageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("");
        setContentView(R.layout.activity_next);
        Intent intent = getIntent();

        byte[] byteArray = intent.getByteArrayExtra("picture");
        Bitmap bmp = BitmapFactory.decodeByteArray(byteArray, 0,
byteArray.length);

        kata = intent.getStringExtra("word");
        Log.d("Hasil: ", kata);
        btn = findViewById(R.id.button);
        btn.setVisibility(View.VISIBLE);
        tvHasil = findViewById(R.id.tv_hasil);
        imageView = findViewById(R.id.imageView2);
        tvHasil.setText(kata);
        imageView.setImageBitmap(bmp);
        imageView.setVisibility(View.INVISIBLE);

        textToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                textToSpeech.setLanguage(new Locale("id","ID"));
                textToSpeech.speak(kata, TextToSpeech.QUEUE_FLUSH,
null, null);
            }
```

```
        });

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent a = new Intent(getApplicationContext(),
MainActivity.class);
                startActivity(a);
                finish();
            }
        });
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController =
Navigation.findNavController(this,
R.id.nav_host_fragment_content_next);
        return NavigationUI.navigateUp(navController,
appBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

Lampiran 2 Verifikasi Abstrak Bahasa Inggris dan Tata Tulis Buku Laporan Skripsi

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA
JL. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122

No. Skripsi : 684

FORM VERIFIKASI

ABSTRAK BAHASA INGGRIS DAN TATA TULIS BUKU LAPORAN SKRIPSI

Nama Mahasiswa : Ahmad Musyadad Aminullah **NIM** : 1741720141

Tanggal Ujian : 5 Agustus 2021

Judul : Implementasi Pengolahan Citra Untuk Penderita Tunanetra Dalam Membaca Teks Berbasis Android

NO	BAGIAN YANG DIVERIFIKASI	NAMA VERIFIKATOR	TANGGAL VERIFIKASI	TTD
1	Abstrak Berbahasa Inggris	Atiqah Nurul Asri, S.Pd, M.Pd	13 September 2021	
2	Tata Tulis Buku Laporan Skripsi	Cahya Rahmad, ST., M.Kom., Dr. Eng.	23 September 2021	

Lampiran 3 Biodata Penulis

DATA PRIBADI	
Nama	: Ahmad Musyadad Aminullah
NIM	: 1741720141
Tempat, Tanggal Lahir	: Malang, 20 Agustus 1999
Jenis Kelamin	: Laki-laki
Agama	: Islam
Jurusan	: Teknologi Informasi / D-IV Teknik Informatika
Alamat	: Jalan Kalpataru no. 49B RT/RW 04/01, Kecamatan Lowokwaru, Kelurahan Jatimulyo, Kota Malang, Jawa Timur
Nomor Telepon	: 081233207604
E-mail	: ahmadmusyadadaminullah@gmail.com