CHAPTER V

Implementation and Testing

5.1 Implementation

5.1.1 Capture Video to Image

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.lang.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.FileReader;
import java.awt.image.WritableRaster;
import com.xuggle.mediatool.IMediaReader;
import com.xuggle.mediatool.MediaListenerAdapter;
import com.xuggle.mediatool.ToolFactory;
import com.xuggle.mediatool.event.IVideoPictureEvent;
import com.xuggle.xuggler.Global;
import java.awt.Image;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Timer;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import java.awt.event.*;
import javax.swing.*;
//diambil dari : http://www.javacodegeeks.com/2011/02/xuggler-tutorial-frames-capture-video.html & lucky
//fungsi :utk mendeklarasi variabel yang akan digunakan
public class TrafficCounter extends JFrame {

    //diambil dari : http://www.javacodegeeks.com/2011/02/xuggler-tutorial-frames-capture-video.html
    public static final double SECONDS_BETWEEN_FRAMES = 50;
    private static int mVideoStreamIndex = -1;
    private static final String outputFilePrefix = "/root/Desktop/TAmobil/capturegambar/";
    private static long mLastPtsWrite = Global.NO_PTS;
    private static int k=0;

    public static final long MICRO_SECONDS_BETWEEN_FRAMES =
            (long)(Global.DEFAULT_PTS_PER_SECOND * SECONDS_BETWEEN_FRAMES);

```

```
309    //diambil dari : http://www.javacodegeeks.com/2011/02/xuggler-tutorial-frames-capture-video.html & lucky
310    //fungsi :utk mengcapture video menjadi gambar
311    private  class ImageSnapListener extends MediaListenerAdapter {
312
313        public void onVideoPicture(IVideoPictureEvent event) {
314
315            if (event.getStreamIndex() != mVideoStreamIndex) {
316                // if the selected video stream id is not yet set, go ahead an
317                // select this lucky video stream
318                if (mVideoStreamIndex == -1)
319                    mVideoStreamIndex = event.getStreamIndex();
320                    // no need to show frames from this video stream
321                else
322                    return;
323            }
324
325            // if uninitialized, back date mLastPtsWrite to get the very first frame
326            if (mLastPtsWrite == Global.NO_PTS)
327                mLastPtsWrite = event.getTimeStamp() - MICRO_SECONDS_BETWEEN_FRAMES;
328
329             // if it's time to write the next frame
330             if (event.getTimeStamp() - mLastPtsWrite >= MICRO_SECONDS_BETWEEN_FRAMES)
331             {
332
333                 String outputFilename = dumpImageToFile(event.getImage());
334
335                 // indicate file written
336                 double seconds = ((double) event.getTimeStamp()) / Global.DEFAULT_PTS_PER_SECOND;
337                 System.out.printf("at elapsed time of %6.3f seconds wrote: %s\n",seconds, outputFilename);
338
339                 // update last write time
340                 mLastPtsWrite += MICRO_SECONDS_BETWEEN_FRAMES;
341             }
342
343        }
```

Proses capture video to image is the process of capturing a video to image every time (seconds), which has been determined by the desired. The image is taken while the video is running and when the video goes it will be processed into an image every time (seconds) specified the less time given the number of images that will produce more and more.

5.1.2 Grayscale & Tresholding

```
File fileGambar = new File(outputFilename);
BufferedImage img = ImageIO.read(fileGambar);//read File fileGambar
//ImageIcon icon = new ImageIcon(outputFilename);
//Image img = icon.getImage();
//System.out.println(icon);
Image newImg = img.getScaledInstance(300, 300, java.awt.Image.SCALE_SMOOTH);
ImageIcon iconz= new ImageIcon(newImg);
jLabel2.setText("");
jLabel2.setIcon(iconz);

File fileGambar1 = new File(outputFilename);
BufferedImage image1 = ImageIO.read(fileGambar1); //read File fileGambar

int x = 0; int y = 0;
w = image1.getWidth(); //get the width of selected image
h = image1.getHeight(); //get the height of selected image
double Sx, Sy;
BufferedImage gambar = new BufferedImage(w, h, BufferedImage.TYPE_BYTE_GRAY); //make Buffered image for grayscale image
/*Grayscale Process & Treshold*/
for (int i = 0; i < w; i++) {
    for (int j = 0; j < h; j++) {
        int c = image1.getRGB(i, j);
        int r = (c & 0x00ff0000) >> 16;
        int g = (c & 0x0000ff00) >> 8;
        int b = c & 0x000000ff;
        double gray =  (0.2989*r + 0.5870*g + 0.1140*b);  // NTSC formula
        //c = new Color(gray, gray, gray);
        //raster.setSample(i, j,0, gray);
        if((gray) < tres1) {
            gambar.setRGB(i,j,TrafficCounter.MixColor(0, 0,0));//change RGB with black colour
            // System.out.println(gambar);
        }
        else {
            gambar.setRGB(i,j,TrafficCounter.MixColor(255, 255,255));//change RGB with white colour
            //System.out.println(gambar);
        }
        //System.out.println(i);
        //System.out.println(j);
        //System.out.println(gray);

    }
}
```

Grayscale is the process undertaken to convert an RGB image has three color channels of red, green, blue be single a gray color chanel within a certain range of gray colors.

Tresholding is the process image that has been converted into gray, and then given a certain threshold value in the range 0-255 so that the image will be black (0) and white (255).

Examples of images that have been in grayscale and threshold :

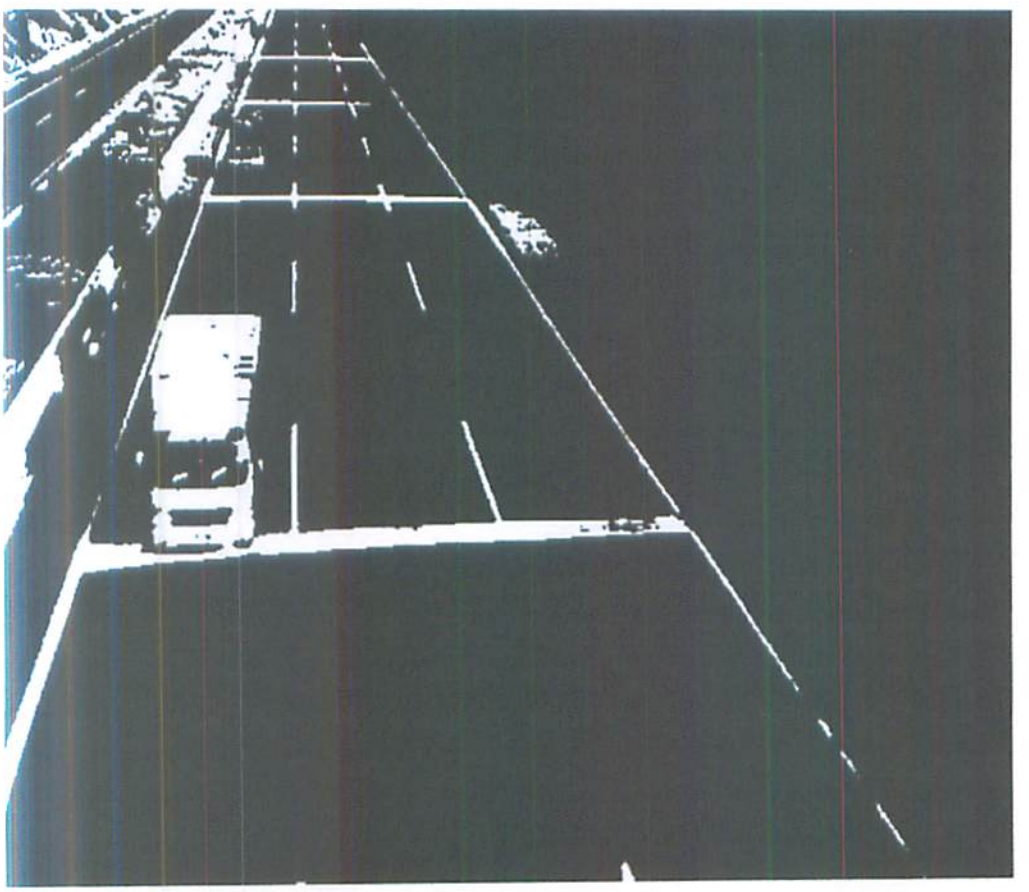


Figure 5.1 Image Grayscale & Treshold

5.1.3 Determine The Detection Area

```
419 int pix [][];
420 pix=new int[w][h];
421 int color;
422 for (int i = 0; i < w; i++) {
423     for (int j = 0; j < h; j++) {
424         color=gambar.getRGB(i,j);
425         pix[i][j]=getRed(color);//save nilai pixel in pix[i][j]
426     }
427 }
428
429 int mobil=0;
430 for (int w = 105; w < 170; w++) {//determine area to detect objek mobil color white(255)
431     for (int h = 180; h < 279; h++) {//determine area to detect objek mobil color white(255)
432         if(pix[w][h]==255 )//compare nilai pix[i][j] with 255
433         {
434             mobil=mobil+1;//count if nilai pix[i][j] same with 255
435         }
436         gambar.setRGB(w,h,TrafficCounter.MixColor(255, 255,255));//make RGB with white colour
437     }
438 }
439
440 System.out.println(mobil);
441 int jmlm=mobil/tres2;//treshold if mobil more than 10
442 System.out.println(jmlm);
443
444 if(jmlm>0){//if 255 more than 10 write in file txt
445     File file = new File("/root/Desktop/TAmobil/capturegambar/jumlahmobil.txt");
446     if (!file.exists()) {
447         file.createNewFile();
448     }
449     String jml = Integer.toString(jmlm);
450     FileWriter fw = new FileWriter(file.getAbsoluteFile(),true);
451     BufferedWriter bw = new BufferedWriter(fw);//write file txt
452     bw.write(k+","+jml+"\n");
453     bw.close();
454 }
455
456 String outputFilename1 = outputFilePrefix +
457         "b"+ k +".jpg";
458 ImageIO.write(gambar, "JPG", new File(outputFilename1));
```

Determination of the detection area is done by using a mouse listener contained in java to get the coordinates (x, y) of the detection area and the predetermined detection area is colored white so if there is a white car in the area of detection it will be calculated and included in a txt file so knowing how much a car is detected.

Examples of cars that enter the detection area :

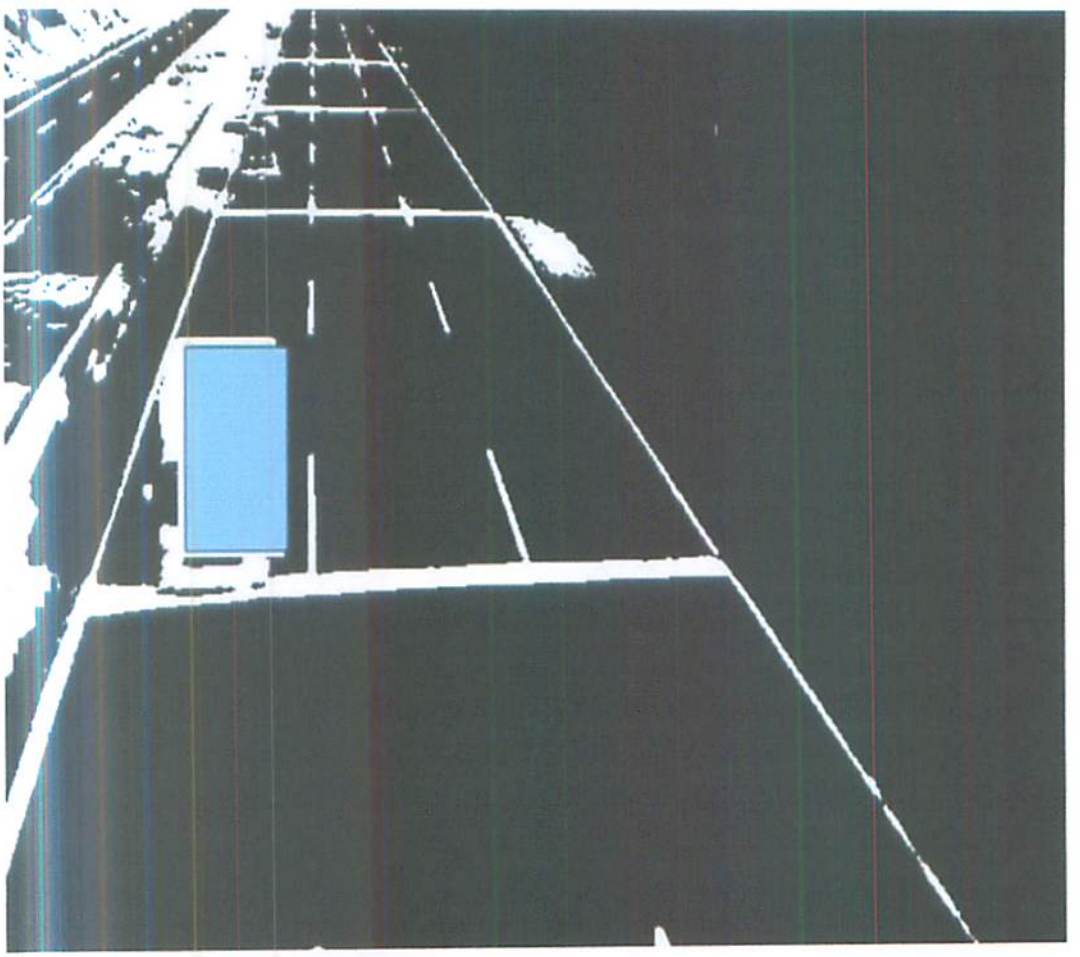


Figure 5.2 Cars That Enter The Detection Area

5.2 Testing

5.2.1 Initial Application View

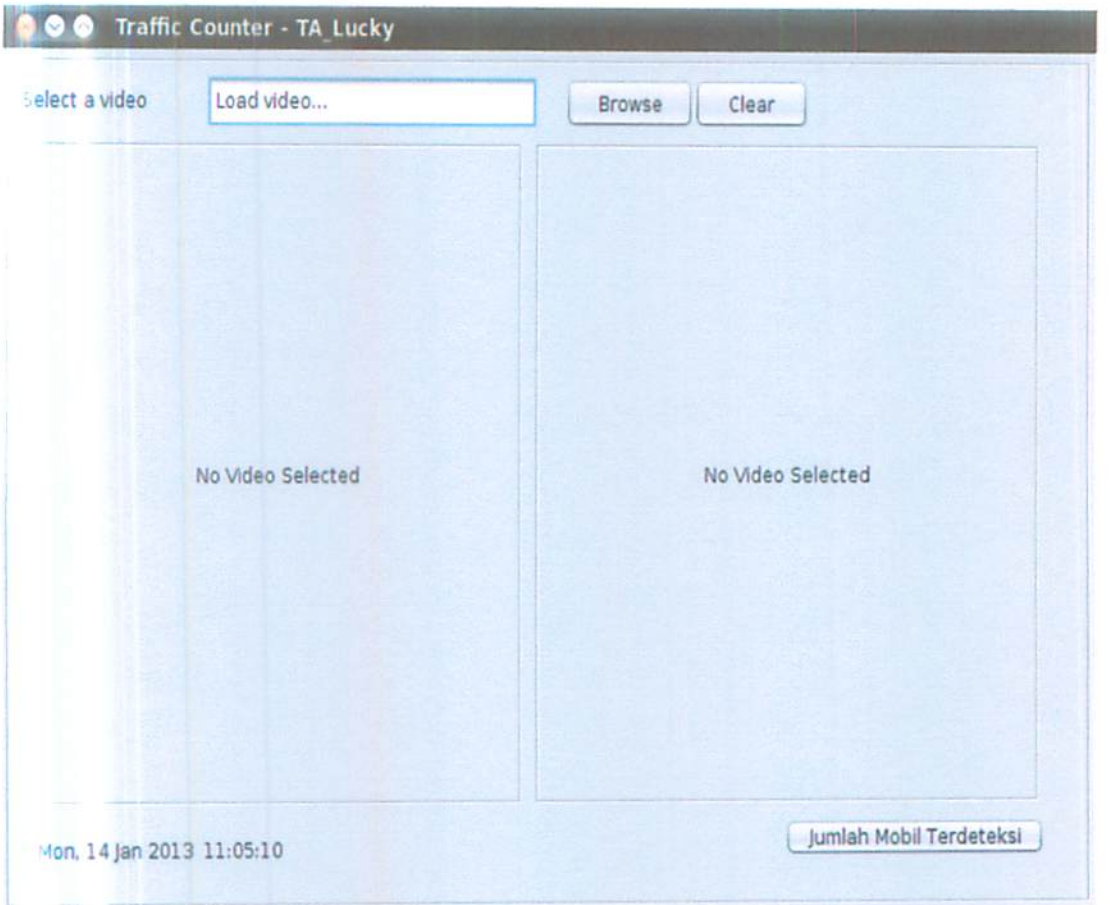


Figure 5.3 Initial Application View

Browse button useful to select the video to be in the video have extension *. MPG, 'jumlah mobil terdeteks' button useful to know the number of cars is detected result from txt file, clear button is used to delete all data if the process has been on the run or an error occurs in the process of selecting a video file.

5.2.2 Display The Result Of The Car Being Detected

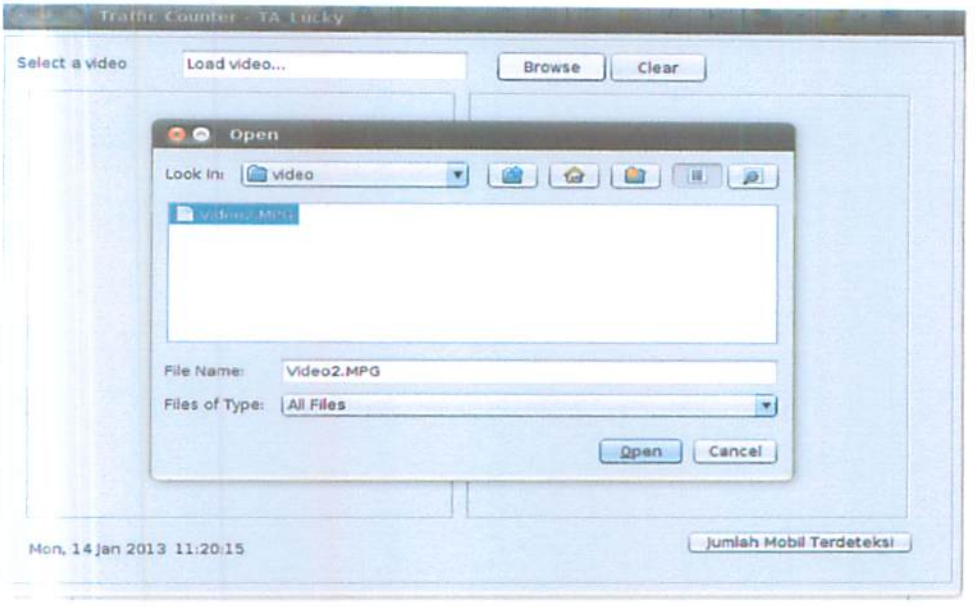


Figure 5.4 Application View Browse Video

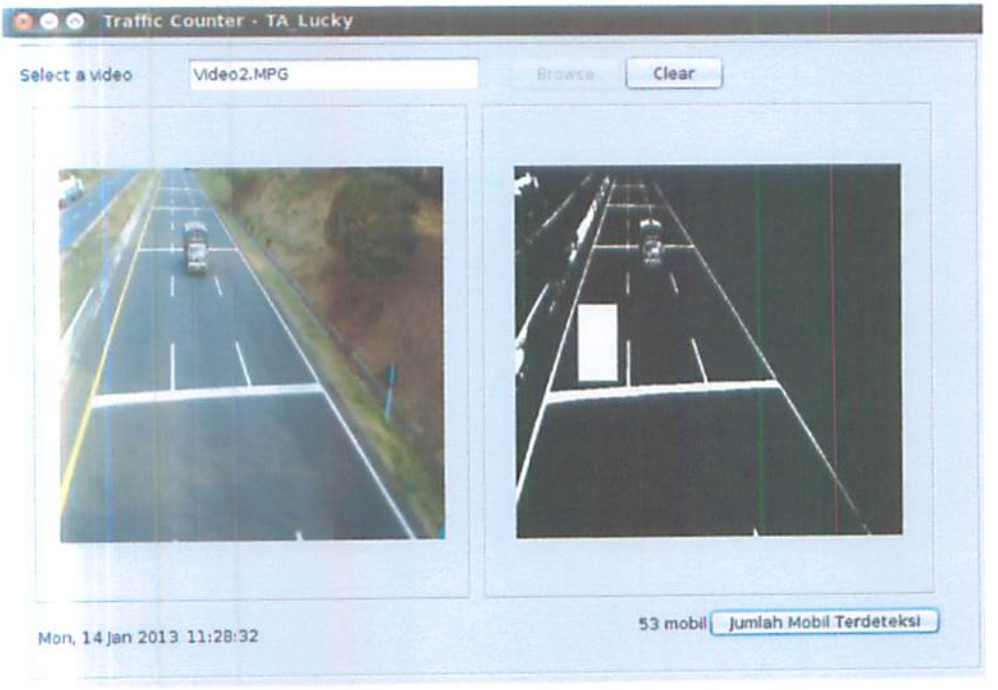
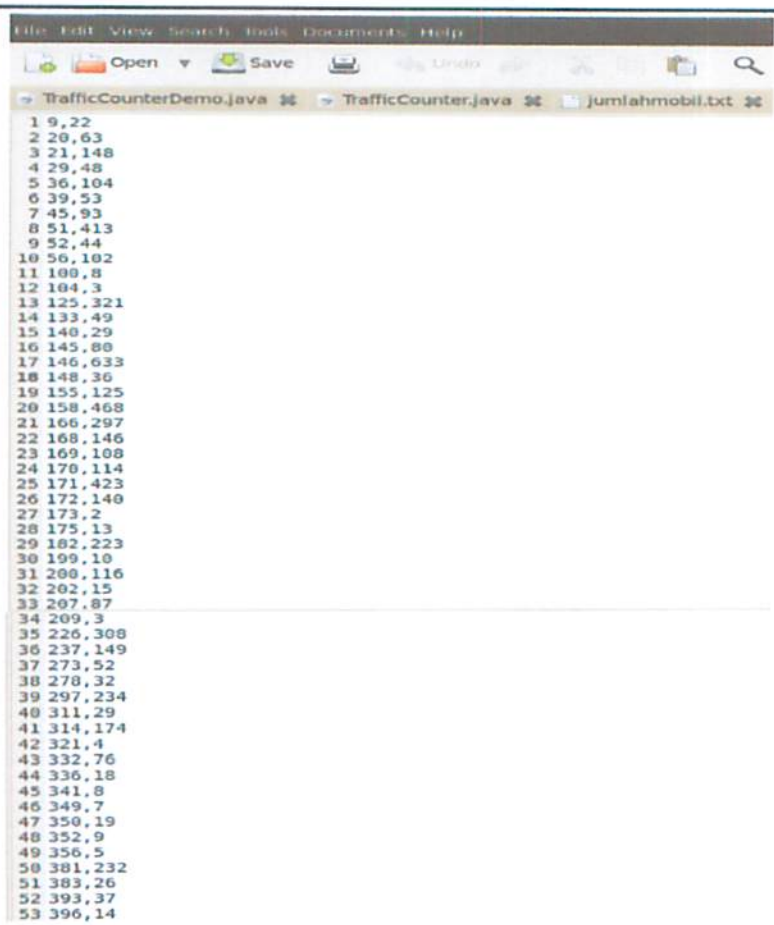
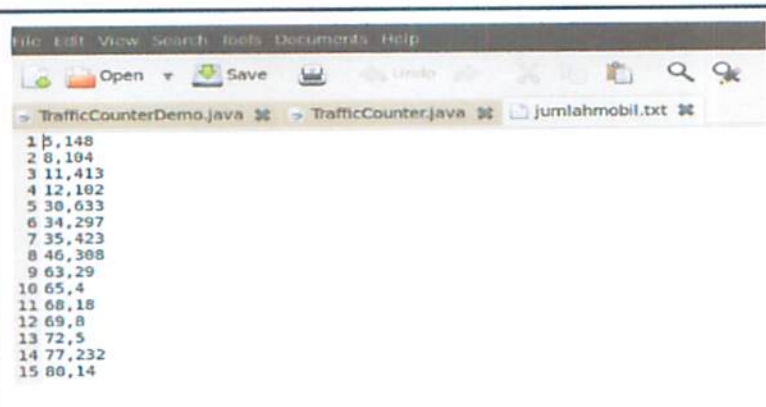


Figure 5.5 Application View The Result Of The Car Being Detected

The results of experiments with different times:

Time	Number of Images	The number of cars is detected is stored in a file jumlahmobil.txt
2 second	466 images	
10 second	94 images	

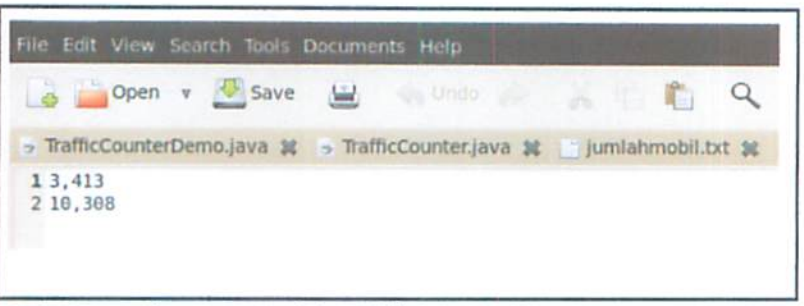
50 second	19 images	

Table 5.1 Results Of Experiments With Different Times

Several factors influence the process of calculating car detection via video:

1. The less time specified in the video capture image will become more and more cars are detected due to the resulting image more and more so that the car can also be detected much.
2. If the dark car will not be detected because the car will be transformed into a dark black color while the process is done to detect the color white.
3. If the threshold is given in the process of grayscale too small then the white color in the image would be too much, causing the car difficult to detect.
4. The greater the limits given for the detection areas that have been determined to detect then the more cars that are detected but it led to the detection results become inaccurate because if the three cars entered the detection area along the car will still be detected as a single car.
5. If given the greater threshold for detecting white color through the detection area will cause the car detection to be less accurate because if there is a car (white color) that's just a little touch detection area will not be detected.