Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2025. Vol. 5. No. 1; e-ISSN: 2808-4519

Analysis of Motor Vehicle Density Based on Digital Images (Case Study on Jenderal Sudirman Street, Palembang City)

Riska Okta Rina[1]*, Rudi Heriansyah[2], Evi Purnamasari[3]

[1] *Universitas Indo Global Mandiri*
2020110059@students.uigm.ac.id[1*], rudi@uigm.ac.id[2], evi.ps@uigm.ac.id[3]

Abstract

Digital image processing has become an essential technology in motor vehicle density analysis, particularly in real-time traffic monitoring. The ability of this system to produce accurate visual data enables researchers and transportation planners to identify traffic patterns, predict congestion, and make data-driven decisions. This study has two main objectives: to determine how motor vehicle density on Jalan Jenderal Sudirman, Palembang City, is analyzed based on digital images and to identify the most appropriate digital imaging method to support this analysis. The Haar Cascade method was chosen as the main algorithm due to its reliability in feature-based object detection. The process begins with image pre-processing to reduce noise and improve lighting in order to enhance image quality. Next, image segmentation is performed to separate vehicle objects from the road background. Detection is performed frame-by-frame, enabling the system to detect vehicles quickly and respond to traffic dynamics. The detected data is then processed using a simple calculation algorithm to determine the number of vehicles within a certain time frame, which is then used to analyze traffic density levels. The results of the study show that the Haar Cascade method is capable of detecting and counting vehicles with an accuracy of 83.6%, making it an efficient solution in intelligent traffic monitoring systems. This research makes a real contribution to the field of modern transportation, particularly in the development of digital image-based systems to support more measurable and technology-based urban transportation planning and decision-making.

Keywords: *density, traffic, transportation, Haar Cascade.*

1. Introduction

Population and economic growth in large cities such as Palembang has led to an increase in motor vehicle ownership every year [1]. This condition is not proportional to the available road capacity, causing traffic congestion, especially on Jalan Jenderal Sudirman, which is the main arterial road. Traffic congestion results in fuel waste, increased air pollution, longer travel times, and reduced comfort and safety for road users [2]. Previous studies have shown that the level of service on Jalan Jenderal Sudirman has been low due to high traffic volumes, and even infrastructure projects such as the LRT and flyovers have caused temporary obstacles [3]. Manual survey methods, which are still dominant, have limitations in terms of accuracy, efficiency, and availability of real-time data. Previous studies have shown that the service level of Jalan Jenderal Sudirman has been low due to high traffic volume, and even infrastructure projects such as the LRT and flyovers have caused temporary obstacles[4]. Manual survey methods, which are still dominant, have limitations in terms of accuracy, efficiency, and real-time data availability [5]. With the development of technology, digital image processing offers a more efficient and adaptive alternative solution for detecting and analyzing vehicle movements. By utilizing the Haar Cascade Classifier algorithm in OpenCV, the traffic monitoring system can support the automatic and accurate collection of vehicle density data [6]. This is relevant for implementation in the city of Palembang to support the formulation of data-based transportation policies.

2. Research Method

The flowchart in the figure illustrates the research process, which begins with a field survey as the first step to obtain an overview of the actual conditions, followed by the collection of both primary and secondary data as required by the research [7]. The collected data then enters the data analysis stage to be processed and interpreted, resulting in findings or research results. Next, conclusions and suggestions are formulated based on these results, which serve as answers to the research questions as well as recommendations for further development or application. Your paper must be in a single-column format with a 0.5 cm space between columns.

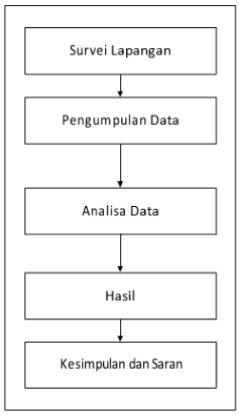


Fig. 1: Research Stages

2.1. Survey Lapangan

In conducting research related to motor vehicle density, the location chosen was Jalan Jenderal Sudirman in Palembang City. Observations were made on busy days and at busy hours, while the observation points were where the number of vehicles was expected to increase (based on research). In this study, the observation points were located around the Grand Mosque of Palembang, Cinde, and the Provincial Police Headquarters. This field survey is intended to gather input and determine suitable observation locations as well as establish the methods to be used, as well as preparations for observation/data collection related to the analysis of motor vehicle density on Jalan Jenderal Sudirman in Palembang City. This field review is to determine the actual conditions in the field and establish the points where motor vehicle density occurs so that data analysis can be carried out manually. can be seen in Figure 2, the locations used in this study

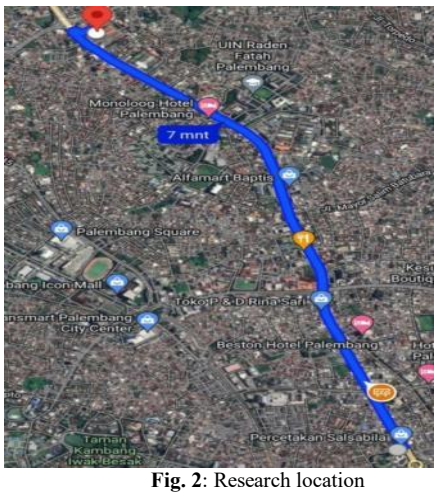


Fig. 2: Research location

At the research location conducted to observe the density of motorized vehicles on Jalan Jenderal Sudirman in Palembang City, observations began at UIGM as the main point for conducting observations until they ended at the Regional Police Headquarters as the final point of the observation results.

2.2. Data Collection

1. Preparation Stage
   The observation of motor vehicle density on Jalan Jenderal Sudirman began at 7:00 a.m. and ended at 5:00 p.m. Because the observation is carried out twice a day, it is done in the morning because at that time people will carry out their activities such as work, school, and so on, and also in the late afternoon, which is the right time to conduct observations because people's activities have ended. These observations are conducted at the beginning and end of the community's active hours, which are believed to increase in line with the specific hours and times of activity of workers and others.

2. Data Collection
   At this advanced stage, data collection is carried out using a video camera. The use of a video camera is based on considerations to ensure that the data obtained is accurate (no data is lost). In other words, the margin of error will be small. The day and time of observation are determined based on the results of the preliminary observation stage. How to determine/establish the observation point: The video camera is placed at a point so that all traffic passing through two fixed points can be recorded with a predetermined distance to be traveled by the vehicle. This observation is conducted for a duration of 1 hour. The timing of this 1-hour observation is based on the peak hours of motor vehicle traffic density on Jalan Jenderal Sudirman in Palembang City.

3. Required data

The data required for this study consists of primary and secondary data. Primary data was obtained through field surveys conducted directly at the research site by observing the surrounding conditions, communicating with samples, and recording and collecting relevant information. The primary data includes observations of traffic signs, road sections, number of vehicles, vehicle speeds, and environmental conditions. Meanwhile, secondary data is obtained from previous literature studies to strengthen the theoretical basis and help solve research problems. Secondary data includes literature studies, average daily traffic data, and supporting books.

4. Data Analysis
In this study, the author applies digital image methods as the main analytical strategy for traffic density phenomena. This method was chosen because of its ability to present visual data, both in the form of images and videos that can be processed quantitatively and qualitatively to understand the actual conditions in the field, especially on Jalan Jenderal Sudirman, Palembang, which is known as one of the points prone to traffic congestion. The use of surveillance cameras enables the detection and counting of vehicles, the identification of movement patterns, and real-time localization of traffic congestion points, so that the information obtained is more accurate and relevant as a basis for traffic policy and engineering A similar study in Magelang by Pangestu et al. (2024) shows that advanced object detection algorithms such as YOLOv8 can be used to analyze the density of CCTV recordings automatically, even though the accuracy reaches around 59.2% [8]. However, this result already shows the great potential of digital images as a tool in traffic monitoring.

In another study by Tawalujan, Sompi, and Manembu (2024), digital images were used to build an adaptive traffic light system based on vehicle queue length. By utilizing YOLO, Python, and image processing, the system is able to detect vehicles on each lane with an accuracy of 81% for cars and 54% for motorcycles. Going further, Whardana and Rentelinggi (2024) combined YOLOv5, optical flow, and RNN to analyze traffic videos and semantically classify vehicle density levels [9]. This approach achieved up to 87% accuracy in classifying density, indicating the effectiveness of a systematic approach based on digital images in urban traffic studies.

2.3. Digital Imaging Methods

In general, digital image processing is a process for manipulating, analyzing, and extracting information from two-dimensional images with the help of computers (Gonzalez and Woods, 2018). In a broader sense, digital image processing includes the processing of any two-dimensional data, whether in the form of static images or dynamic data such as video [10]. A digital image can be defined as a two-dimensional array containing real or complex values, represented in a specific bit sequence. Each element in the array is called a pixel, which is the smallest unit of an image that contains brightness (intensity) and color information. Thus, a digital image is a numerical representation of a real-world object that has been converted into digital form so that it can be processed by a computer. Images are basically divided into two types, namely analog images and digital images. Analog images are continuous images, such as images on a television monitor, X-ray photos, or images from a tube camera. Meanwhile, digital images are the result of the representation of analog images captured by a sensor and then processed through sampling and quantization (Munir, 2013).

Sampling refers to the size of pixels or the number of boxes arranged in rows and columns in an image. The denser the sampling, the higher the image resolution and the closer it is to the original object. Quantization is the process of converting the continuous intensity values of an analog image into a discrete form with a specific number of levels. In grayscale images, quantization refers to the number of gray levels available, usually expressed in binary bits. For example, an 8-bit grayscale image has 256 gray levels (0–255), while RGB color images typically use 24 bits, meaning each red, green, and blue channel has 256 intensity possibilities. In other words, sampling determines the spatial resolution of an image, while quantization determines the resolution of the intensity or the number of colors/grayscale levels that can be represented by the image. These two processes are fundamental in the formation of digital images, thus greatly affecting the visual quality and information that can be obtained from the image.

The development of digital image processing technology has enabled its application in various fields, including facial recognition, medical image analysis, remote sensing (remote sensing), traffic surveillance systems, and object classification. The advantages of digital images over analog images are their ability to be processed automatically, stored in digital media, transmitted easily, and manipulated for various analytical needs (Kumar, 2020). It is known that a digital image is represented by a matrix, where x and y represent the position of the pixel based on its column and row, and the value of the matrix element represents its intensity. Because a digital image is a matrix, the process of manipulating pixels is done by manipulating the elements of the matrix. The manipulation process can involve only a single matrix element, a set of matrix elements, or all matrix elements. Figure 3 shows the types of image processing operations.

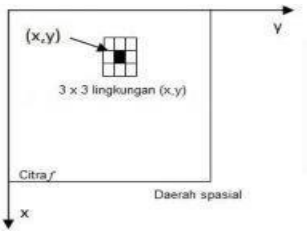


Fig. 3: 3×3 Neighborhood of a Point (x,y) in an Image in Spatial Domain

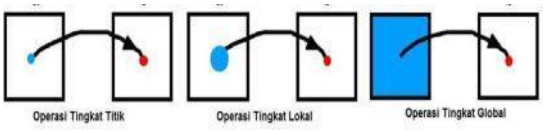


Fig. 4: Illustration of Image Processing Operation Types

Based on the image manipulation process, image processing can be grouped into three levels of operation, namely:

1. Point Level Operations
   Point level operations are image processing operations performed on a single pixel in an image. In this operation, the pixel at the desired location is modified with a specified operation, and the result, which is a new pixel value, is placed at the corresponding location in the new image. This operation is performed on all pixels in the image. Point-level operations are divided into three types of operations, namely point-level operations based on intensity, point operations based on geometry, and point operations based on a combination of intensity and geometry. In intensity-based point operations, the new image is a similar image, but with the intensity of each pixel changed. Meanwhile, in geometric-based operations, the new image is a similar image but with a changed position. Examples of applications of this operation are image brightness adjustment, binarization, contrast adjustment, image sharpening, and others.

2. Local Operations
   Local operations are image processing operations performed on the desired pixels and influenced by the intensity of the surrounding pixels. In this operation, the desired pixel is modified using an operation where the neighboring pixels influence the value produced by the operation. This new value is placed in the appropriate location in the new image and is performed for all pixels in the image. Examples of applications of this operation are image smoothing and sharpening.

3. Global-level operations
   Global operations are image processing operations performed on a pixel to produce a new pixel value at the corresponding location, and the determination of this new value is influenced by the intensity of all pixels in the image. An example of an application of this operation is the averaging operation. of all pixels in the image. An example of the application of this operation is the histogram equalization operation. The algorithm that will be used in this system is created using the Python programming language. The Python programming language has many advantages over other programming languages, namely that it is suitable for creating prototypes. and is also suitable for use in embedded systems, allowing for future development.

The following is a flowchart of the system used with the digital image method.

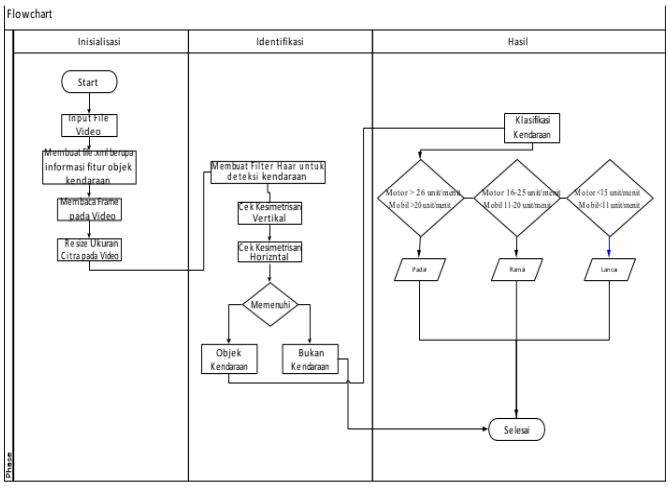


Fig. 5: System Flowchart

The application starts by opening a video file. After that, the system runs a process to create an XML file in the form of a vehicle object filter. The XML file is used to classify vehicles by reading frames in the video. It then changes the image size in the video. Then it creates a Haar filter for vehicle detection. This allows for checking vertical symmetry and horizontal symmetry to determine whether the object is a vehicle or not. If it is not, the process stops. If it is, the vehicle object is marked with a box. Next, it identifies the classification of the vehicle, then calculates the number of vehicles present and calculates the speed so that it can obtain a calculation of the vehicle density.

Below, we will provide a comprehensive explanation covering the entire process, starting from the input, continuing to the process itself, and ending with the output.

1. Video File Input
   To start the process, a program is used to prepare the video. The video is added by entering the file name “sudirman.mp4” which was recorded in the afternoon, with the camera capturing the perspective from the middle of the road.
2. Creating XML Files

All vector samples that have been created are combined into one vector sample file. Training vector samples for classification and converting the results. The combined vector feature file is trained using OpenCV training cascade into an *.xml file format. Use the *.xml file to detect objects. The *.xml file is used for the detection process.

3. Reading Frames in Videos

The algorithm will be created using the video processing capabilities of OpenCV (Open Computer Vision). The video to be processed by the algorithm is taken from real-time surveillance videos.

a. Image Processing

The system algorithm begins by opening the video, then enters the image preprocessing stage where each video frame represented by an image will be processed here. Each image in the video frame is processed based on the difference between the background and the object (foreground). The separation of the background from the foreground will use the Background Subtraction method from OpenCV.

b. Object Detection

The object detection process consists of the following 4 sub-processes:

1) Background Substraction

The Background Subtraction method is used to convert each frame of an image from color to black and white (grayscale), where the difference between black and white indicates the movement of objects in the video. Moving objects in each frame will be displayed in white, while stationary objects will be displayed in black. The background extractor works in three main stages: building a background model from previous images or videos, which can be static images or more complex representations of backgrounds that change over time; separating objects by comparing the current frame with the background model to detect significant differences that are considered new objects or important changes; and periodically updating the model to remain adaptive to slow changes in the background, such as variations in lighting or shifts in the position of fixed objects.

Fig. 6: Results of the Background Extractor Program Execution

2) Object edge detection

After distinguishing between moving objects and stationary objects using the Background Subtraction method, it will be easier to place the (region of interest or ROI) on the object. Mark placement on moving objects is done using the edge detection method. This edge detection method will provide a series of points that form a line on the outer edge of the object. There are several well-known and widely used methods for edge detection in images, namely the Robert operator, the Preswit operator, and the Sobel operator. The Sobel method is most widely used as an edge detector because of its simplicity and effectiveness (Munir, 2004). The advantage of this method is its ability to reduce noise before performing edge detection calculations. The Sobel operator consists of a 3x3 matrix, each of which is and . The mask matrix is designed to provide a maximum response to the edges of objects, both horizontal and vertical. The mask can be applied separately to the image input. The Sobel operator uses a 3 x 3 gradient kernel with predetermined coefficients. It can be expressed as follows

$$G_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \text{ dan } G_y=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \quad (1)$$

The above kernel is designed to solve edge detection problems both vertically and horizontally. These kernels can be used together or separately (Purnomo and Muntasa, 2010). To obtain the maximum value from the Sobel

operator, the next step is to calculate the edge strength of the image in relation to its brightness by finding the magnitude value, which can be calculated using the following equation:

$$M = \sqrt{G_x{}^2 + G_y{}^2} \tag{2}$$

Because calculating the root is a complex problem and produces real values, the calculation can be simplified when finding the edge strength (magnitude). The magnitude of the gradient can be calculated even faster by using the following equation

$$M = |G_x| + |G_y| \tag{3}$$

Figure 7 shows edge detection using the Sobel operator. The convolution operation works by shifting the pixels one by one, and the results are then stored in a new matrix. The first convolution is performed on pixels with a value of 1 (at the center point of the mask).

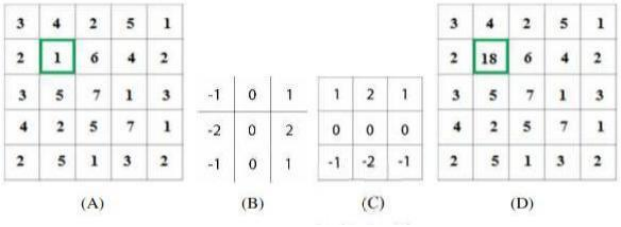


Fig. 7: (A) Original image, (B), (C), (D) Convolution results

$$G_x = (3)(-1) + (2)(-2) + (3)(-1) + (2)(1) + (6)(2) + (7)(1) = 11$$

$$G_y = (3)(1) + (4)(2) + (2)(1) + (3)(-1) + (5)(-2) + (7)(-1) = -7$$

$$M = \sqrt{G_x^2 + G_y^2} = \sqrt{(11)^2 + (-7)^2} \cong |G_x| + |G_y| = |11| + |-7| = 18$$

The value of 18 in the convolution result is obtained with the following calculation (Munir, 2004):

Thus, the value 1 is changed to the value 18 in the output image. In the convolution process, there are two possibilities that need to be resolved, namely if the convolution result produces a negative value, then that value is made 0, and if the convolution result produces a pixel value greater than the maximum gray value, then that value is made the maximum gray value. In the Sobel matrix with a 3 × 3 kernel, not all pixels are convolved, particularly the rows and columns located at the edges of the image (border), because the pixels at the edges of the image do not have complete neighbors, so the convolution formula cannot be applied to these pixels (Kadir and Susanto, 2013)S. This is illustrated in Figure 8, where convolution is not possible at positions A and B.

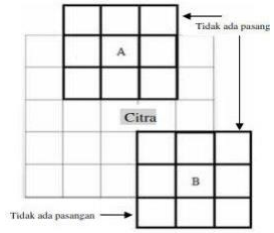


Fig. 8: Problems with convolution

The convolution problem with pixels that have no neighbors always occurs with pixels on the right, left, top, and bottom edges of the image. There are two solutions to this problem: ignore the pixels at the edges, where the pixels are not convolved because their neighbors are incomplete, so the resulting image in that area is filled with zeros, filled with the original image, or not included in the resulting image, resulting in a smaller image size; or by adding rows and columns at the edges, where the new rows and columns are given a value of zero so that the convolution process can still be carried out. This edge detection process uses Background Subtractor as input, with the output

being objects that were previously white and are then changed to black, except for the edges of the object. Thus, edge detection can be said to display the outer lines that form the object

The program for detecting edges is shown by the following program listing, and the results of the program are shown in Figure 9.

```
contours,hierarchy =
cv2.findContours(fgmask.cv2,RETR_EXTERNAL,cv2
.CHAIN_APPROX_SIMPLE)
```

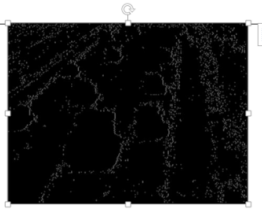


Fig. 9: Edge Detection Results

3) Resize Image Size
Resizing is a process used to change the size of a digital image in pixels, either making it smaller or larger than its actual size. The method used is not a specific method, but rather uses a size comparison of each image with the desired image size.

4) Creating a Haar Filter for vehicle detection
Once the markings on each edge of the object have been formed, it will be easier to determine the coordinate values for the marking coordinate points (region of interest or ROI). Coordinates required to create a mark (region of interest or ROI) are the maximum edge coordinates (represented by the coordinate values of the outermost points of the object, the height and width values of the object. The data that must be obtained before obtaining the Region of Interest (ROI) is the data on the coordinates of the outermost point (x, y), the height value, and the width value. These three values can be obtained with the cv2.contour command, which is a library from OpenCV. As in the previous step, the output from the previous program will be the input for the next program. In this case, the output from the edge detection program will be processed to obtain the third value. Program to obtain the third valu

```
try: hierarchy = hierarchy[0]
except: hierarchy = []
for contour, hier in zip(contours, hierarchy):
(x,y,w,h) = cv2.boundingRect(contour)
```

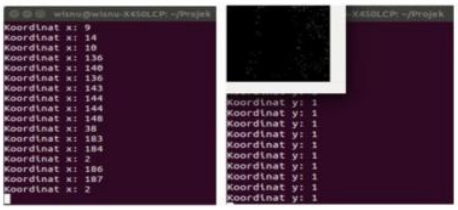


Fig. 10: Results of executing the x.y coordinate value display program

a. Determining the Region of Interest
Region of Interest (ROI) is a part of an image that is selected for processing with the aim of limiting a specific area so that processing is only carried out on the relevant parts. The ROI determination process differs from block processing, because in ROI only part of the image is processed and can consist of more than one region or area, even in the form of a polygon consisting of contiguous pixels or based on intensity ranges, so that the pixels are not always related to each other. In vehicle detection, the ROI determination step is carried out by finding the outermost point of the counting line at the edge of the road, then using the outermost points on the right and left sides, reducing the number of pixels at the outermost point on the left side by a certain constant and adding the number of pixels at the outermost point on the right side by the same constant, then creating an ROI by drawing straight lines parallel to the four points to form an area from the top left point to the bottom right point. Next, the ROI function is declared and directed to be applied to the frame to be processed. Once the coordinates of the outer points of the object, the height (ymax), and the width (xmax) are known, the ROI can be automatically drawn on each edge of the object. The classification and masking process is used to distinguish regions, where if the pixels on the mask are not equal to zero, the image will be

processed, while if they are equal to zero, no processing will be done. The ROI placement program is shown in the program listing, the results of which are visualized in Figure 11.

```
def get_roi(grabbed, frame):
(LINE_BOTTOM, _) = frame.shape[:2]
 roi =
frame[LINE_TOP:LINE_BOTTOM,LINE_LEFT:LINE_RIGHT]
 return grabbed,roi
frame_cp = frame.copy()
grab_roi, frame_roi = get_roi(grabbed, frame_cp)
```

Fig. 11: The program execution results display the Region of Interest (ROI)

b. Object Tracking

After the object is marked with a box or Region of Interest (ROI), the coordinate value of the center point of the rectangle can then be determined. The coordinate value of the center point is obtained from the sum of the two results divided by the two results of the object width. Once the coordinates of the center point are known, a small circle is drawn or marked at those coordinates to represent the object tracking indicator. A digital image has RGB value components (a combination of red, green, and blue colors). From the RGB values, the grayscale value (degree of grayness) can be determined using the following formula

$$\text{Grayscale\_pixel} = 0.2989R + 0.5870G + 0.1140B \quad (4)$$

Fig. 12: Grayscale Picture

Object tracking can be performed simply by placing the Region of Interest (ROI) as in the previous step. However, the calculation algorithm used employs the concept of cutting a line, so tracking using the Region of Interest (ROI) as an indicator will have many possibilities that can reduce the accuracy of the system. The program to display tracking in the form of dots on objects is shown in the following program listing

```
if w > 20 and h > 25:
  cv2.rectangle(frame, (x,y), (x+w,y+h), (180, 1, 0), 1)
  x1=w/2
  y1=h/2
  cx=x+x1
  cy=y+y1
  centroid= (cx,cy)
  titik=cv2.circle(frame,(int(cx),int(cy)),4,(0,255,0),-
  1)
```

Fig. 13: Vehicle Tracking Execution Results

c. Check Symmetry
The coordinates of the midpoint can be obtained using the following formula: Formula for the midpoint coordinates of the x-axis:

$$x = \frac{x}{2} \tag{5}$$

Formula for the midpoint coordinates of the y-axis:

$$x = \frac{x}{2} \tag{6}$$

Every object whose mean value (centroid) has exceeded the calculation limit will be included in the calculation. The calculation process using the calculation line can be determined using an equation.

$$\frac{x}{lc} > 0 \tag{7}$$

$$\frac{x}{lc} \leq 0 \tag{8}$$

Explanation: lc is the line boundary used to update vehicle calculations.
In addition to being used to draw boxes or rectangles, the outer coordinate data (x, y), the height coordinate (ymax), and the width coordinate (xmax) can also be used to classify vehicle types according to the desired threshold. Vehicle type classification itself uses high value coordinate data and/or or width coordinate data so that it can be classified into at least two types, namely two-wheeled vehicles and four-wheeled vehicles, and the results of the program are shown in Figure

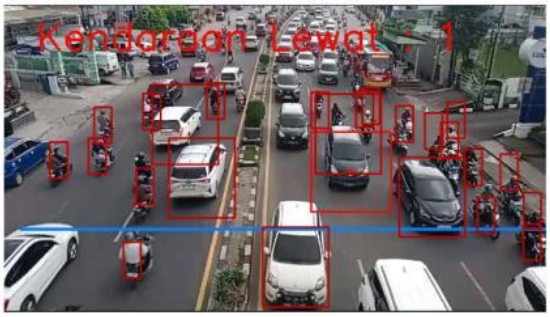


Fig. 14: The program execution results show vehicle density.

d. Vehicle Classification
The vehicle classification process can be carried out by determining threshold values for the physical attributes of objects, such as the height or width of vehicle images. This method is quite effective in distinguishing relatively small two-wheeled vehicles from four-wheeled vehicles, which tend to be larger and more proportional. However, the threshold-based approach has limitations, especially in conditions of heavy traffic, varying lighting, image capture angles, and camera quality. To improve accuracy, more adaptive methods are used, such as Haar-like features combined with the AdaBoost algorithm, where vehicle images (side, front, and rear views) are used as positive samples with uniform sizes, while negative samples come from images of other objects. The results of this training form a cascade classifier that is able to recognize objects with an output value of "1" if a vehicle is detected and "0" if not.

e. Vehicle Density Calculation
After the object tracking indicator is obtained, namely from the results of drawing a square box or the results of the Region of Interest, it will be easier to calculate each object. The method of counting objects is done by drawing lines in the form of coordinates (x1,y1) and (x2,y2) with the value y1 equal to y2 . The line must extend from the right side of the road to the left side of the road. The calculation will be performed when the object tracking indicator (in the form of a small circle or dot) intersects the coordinates of the line that was created. Therefore, tracking in this script uses dots as a tracking function. The concept of a dot as a tracking indicator places the dot at the center point of a previously drawn square or rectangle. The data required to draw the dot is the coordinate value of the center point (x, y) on the square or rectangle. Enter the final stage, which is the result. The result of this application is a file. txt file containing the number of vehicles detected from the input video and the results of traffic density calculations on this system are determined by comparing the number of white pixels with the frame pixel area, then converting the comparison result into a percentage (%). This provides the result for determining vehicle density. This .txt file makes it easy for users to view the history of vehicle density detected from the input video.

```
garis =
cv2.line(frame,(10,130),(300,130),(200,200,0),2)
   if cy==210:
      counter=counter+1
   if cy==205:
        counter=counter+1
   if cy==200:
        counter=counter+1
   if cy==190:
        counter=counter+1
   else: counter=counter+0
```

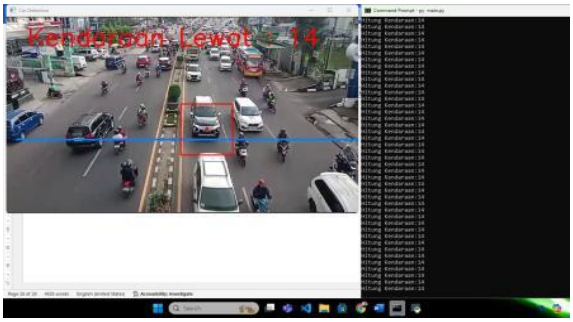


Fig. 15: Results of the vehicle density calculator program execution

There are three parameters that can be used to determine traffic density. Density refers to the number of vehicles on a particular road section. Typically, density is reported in units of vehicles per mile or vehicles per kilometer. High density indicates that vehicles are very close together, while low density indicates that there is more distance between vehicles.

Table 1: Vehicle Density Parameter

No	Parameter	Scope
1	smooth	Motor 0-15 units/minute Car 0-10 units/minute
2	busy	Motor 16-25 units/minute Car 11-20 units/minute
3	crowded	Motor 26-n units/minute Car 20-n units/minute

3. Results and Discussion

3.1. Implementation

This study developed a Python and OpenCV-based CarVelocity system to detect and analyze traffic density on Jalan Jenderal Sudirman, Palembang. The process began with 1-minute video input with a resolution of 1280×720 pixels and 30 fps, which went through pre-processing stages of ROI determination, image stabilization, and noise reduction. Vehicle detection is performed using the Haar Cascade or background subtractor method, followed by tracking and unique ID assignment using the Kalman Filter and matching techniques. Next, the system performs homographic transformation to a bird's-eye view to facilitate distance measurement and speed estimation, which is converted to km/h. In the final stage, the system classifies vehicles based on their dimensions into two categories, namely two-wheeled (R2) and four-wheeled (R4), with the output being real-time information on speed and vehicle type to support comprehensive and accurate traffic analysis.

```
import cv2

delay= 500
detec = []
pos_line=300
offset=3
car= 0


car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')

# Inisialisasi video capture
cap = cv2.VideoCapture('sudirman3.mp4')

```

Fig. 16: Video Initialization

After the video is imported into the application, the system calls up the Haar Cascade XML file that has been previously trained using a collection of positive images (containing vehicles) and negative images (without vehicles). During the training process, Haar features are extracted and then selected using the AdaBoost algorithm to form a strong classifier, and the results are stored in XML format. In the implementation, each candidate object in the video frame is checked in stages through a cascade classifier, starting from simple to complex features, so that non-relevant areas can be eliminated early on. If the candidate object passes all stages, it is categorized as a vehicle and marked with a red bounding box for easy visualization. With this approach, the system is capable of real-time vehicle detection even though the video has variations in lighting, shooting angle, and traffic density, making this method effective for use in research and practical applications in the field of digital image processing.

Fig. 17: Marking blue lines and setting up square tents

This is done to reduce the risk of detection errors. Thus, vehicle object recognition will become more focused. This will make the detection of vehicle types more accurate. Then, it continues by detecting vehicle objects using a classifier. The classifier will match the vehicle object with the XML file that has been loaded previously. If the conditions are met, the detected vehicle object will then be checked for symmetry. The detected vehicle object will be checked to see if it is symmetrical vertically and horizontally or not. If it is symmetrical vertically and horizontally, the object is considered a vehicle. The application will mark the object considered a vehicle with a square. The next step is to identify the type of vehicle. The type of vehicle is identified by measuring the area of the rectangle used to mark the vehicle object. The final stage is the result. The result of this application is a .txt file containing the vehicle types detected from the input video. This .txt file makes it easy for users to find out the types of vehicles detected from the input video.

3.1.1. Main Functions of the Program

Vehicle density detection software has three main functions, namely identifying vehicles in videos, determining vehicle types, and saving detection results in .txt files. The process begins with object detection on each frame using the Haar Cascade Classifier method from OpenCV. Detected vehicles are marked with rectangular bounding boxes, and the area of the box is used for simple classification: small objects are categorized as two-wheeled vehicles, while large objects are categorized as four-wheeled vehicles. After that, the system records the detection results each time a vehicle passes the counting line strategically placed on the video frame. Data in the form of the number of vehicles by type is stored in a text file for easy further analysis. The final stage is determining traffic density based on the number of vehicles passing within a certain period. If the number of vehicles exceeds the threshold, traffic conditions are categorized as congested, while if it is still below the threshold, conditions are considered smooth. With this approach, the software is able to provide real-time traffic density information that can be used in traffic light control and transportation policy formulation

3.2. Test Scenarios

The test was conducted using a video provided in the application with the following parameters:

1. Scale Factor: A parameter that determines how much the image size is reduced at each image scale. It is usually a number that scales or multiplies a quantity, in this case the width and height of the image. This helps to keep the aspect ratio intact and maintain display quality. So the image does not appear distorted. By changing the scale of the input image, you can change the size of larger objects to smaller ones, so that they can be detected by the algorithm.
2. Height: Object height parameter
3. Frames: Number of frames/images extracted
4. Total vehicles in real terms, i.e., the number of vehicles counted manually based on the actual conditions in the field
5. Total vehicles passing by system, i.e., the number of vehicles read by the system
6.

Table 2: Scale Factor Test Parameters 1.1

Parameter	Value
Frame	100
Height	100
Scale factor	1.1

The value 100 represents the height of the pixel and the full frame size, which is 100%. The results obtained after testing the scenario on the above parameters are as follows:

Fig. 18: Results Obtained After Conducting Scenario Testing

Table 3: Scale Factor Test Parameters 1.1

Total vehicles passing by in real time	69 Unit
Total vehicles that pass through system	101 Unit
Accuracy of detection	68,3%
Error Detection	31,7 %

$$\text{accuracy} = \frac{\text{Actual number of vehicles}}{\text{number of vehicles read by the system}} \times 100\%$$

$$\text{Accuracy Detection} = \frac{69}{101} \times 100\%$$

Accuracy Detection = 63,8%
Error Detection = 100% - Accuracy Detection
Error Detection = 100% - 63,8% = 36,2%

Table 4: Scale Factor Test Parameters 1.2

Parameter	**Value**
Frame	100
Height	100
Scale factor	1.2

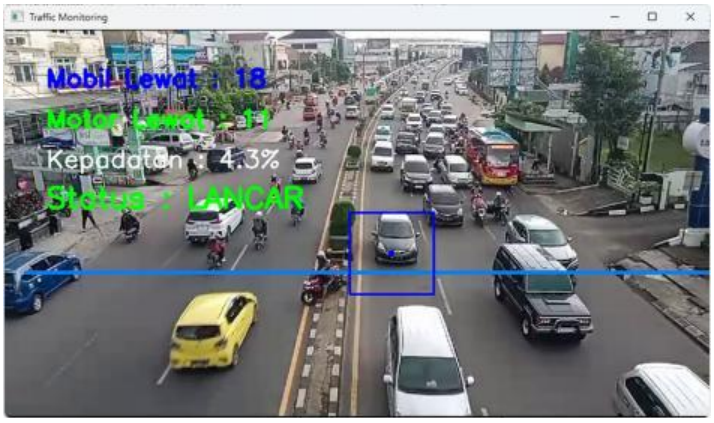


Fig. 19: Results Obtained After Conducting Scenario Testing

Table 5: Scale Factor Test Parameters 1.2

Total vehicles passing by in real time	69 Unit
Total vehicles that pass through	32 Unit

system	
Accuracy of detection	46 %
Error Detection	54 %

$$\text{accuracy} = \frac{\text{Actual number of vehicles}}{\text{number of vehicles read by the system}} \times 100\%$$

$$\text{Accuracy Detection} = \frac{32}{69} \times 100\%$$

Accuracy Detection = 46%
Error Detection = 100% - Accuracy Detection
Error Detection = 100% - 46% = 54%

Table 6: Scale Factor Test Parameters 1.3

Parameter	Value
Frame	100
Height	100
Scale factor	1.3

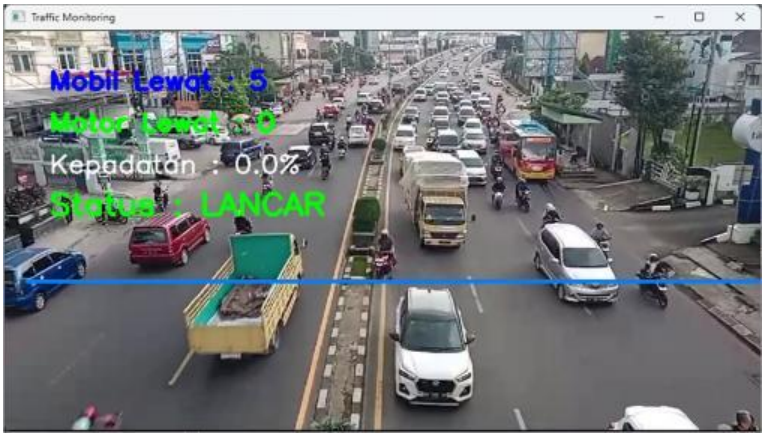


Fig. 20: Results Obtained After Conducting Scenario Testing

Table 7: Scale Factor Test Parameters 1.3

Total vehicles passing by in real time	69 Unit
Total vehicles that pass through system	29 Unit
Accuracy of detection	42 %
Error Detection	58 %

$$\text{accuracy} = \frac{\text{Actual number of vehicles}}{\text{number of vehicles read by the system}} \times 100\%$$

$$\text{Accuracy Detection} = \frac{29}{69} \times 100\%$$

Accuracy Detection = 42%
Error Detection = 100% - Accuracy Detection
Error Detection = 100% - 42% = 58%

Table 8: Scale Factor Test Parameters 1.4

Parameter	Value
Frame	100
Height	100

Scale factor	1.4

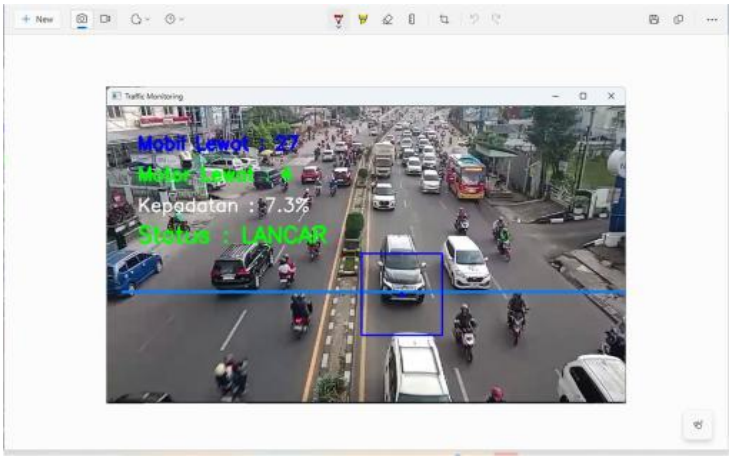


Fig. 21: Results Obtained After Conducting Scenario Testing

Table 9: Scale Factor Test Parameters 1.4

Total vehicles passing by in real time	69 Unit
Total vehicles that pass through system	57 Unit
Accuracy of detection	83,8 %
Error Detection	16,2 %

$$\text{accuracy} = \frac{\text{Actual number of vehicles}}{\text{number of vehicles read by the system}} \times 100\%$$

$$\text{Accuracy Detection} = \frac{57}{69} \times 100\%$$

Accuracy Detection = 83,8%
Error Detection = 100% - Accuracy Detection
Error Detection = 100% - 83,8% = 16,2%

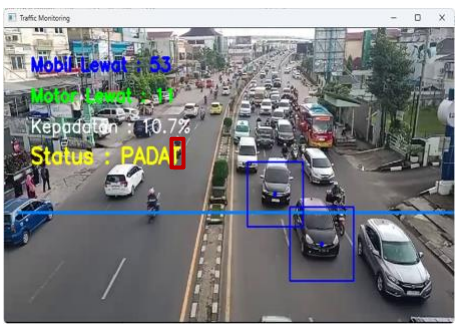
From the scale factor testing, the most optimal scale factor parameter to be implemented in the scenario testing stage is scale factor 1. This is because when the scale factor is changed, it must be adjusted to the object to be detected in order to achieve maximum detection and obtain an accuracy of 83.8%, so the author will use a scale factor parameter of 1.4 for scenario testing.

3.3. Determination of Vehicle Density

In this system, traffic density calculations are not only performed visually, but also using a quantitative approach based on the area of road space occupied by vehicles. The density percentage is calculated by comparing the area occupied by vehicles with the total road area in the video frame. The formula used is as follows:

$$density(\%) = \frac{A_{vehicle}}{A_{road}} \times 100\%$$

The traffic density detection system is designed to calculate the percentage of road usage and average it over a five-minute interval for more stable results. Based on this percentage, traffic conditions are classified into four categories: congested (>50%), busy (30–50%),



smooth (1–29%), and light (0%). Testing was conducted with varying lighting conditions using video contrast adjustment (alpha value 0.2–2) so that the system could consistently detect vehicles. The vehicle type classification process utilizes bounding box coordinates to distinguish between two-wheeled and four-wheeled vehicles based on dimensional thresholds. The analysis results are visualized with colored bounding boxes, allowing for quick and accurate monitoring of vehicle type distribution and traffic conditions.

Fig. 22: The program execution results show vehicle density and
The program execution results show vehicle density at the third minute.

The final stage is the result. The result of this application is a .txt file containing the number of vehicles detected from the input video and the result of traffic density data on this system is determined by comparing the number of white pixels with the frame pixel area, and then converting the result of the comparison into a percentage (%). This provides a result for determining vehicle density. This .txt file makes it easy for users to view the history of vehicle density detected from the input video.

4. Conclusion

The conclusions obtained in this study are as follows:

1. The method used to analyze vehicle density on Jalan Sudirman Palembang uses OpenCV and Haarcascade.
2. The determination of vehicle density on Sudirman Road observed using OpenCV and Haarcascade was successfully carried out with an accuracy of 83. 6%, making it usable for the process of detecting vehicles on the road.
3. The results of the road density percentage calculation can determine the traffic density status, namely congested, crawling, busy but flowing, and flowing.

References

[1] B. P. Statistik, “Provinsi Sumatera Selatan,” Bruto Provinsi Sumatra Selatan 1993-2003 l, p. 66, 2024.
[2] MG. Brilliant, Roy Raja Sukmanta Meilala, and Delfia Herwanis, “Manajemen Transportasi : Kerugian Transportasi Akibat Kemacetan Lalu Lintas di Aceh,” Sammajiva J. Penelit. Bisnis dan Manaj., vol. 2, no. 4, pp. 42–53, 2024, doi: 10.47861/sammajiva.v2i4.1480.
[3] A. R. Pribadi, I. Yunus, M. Kasmuri, J. Jendral, A. Yani, and N. Palembang, “1 2 , 3,” vol. 15, no. 2, 2018.
[4] W. Y. M. Nurokhman1, Suryanto1, Singgih Subagyo1, “P-issn 2798-4869 e-issn 2798-4060,” Dampak Transp. Sist. Ligh Rail Transits Terhadap Kemacetan Lalu Lintas Dan Emisi Carbon Di Jakarta, vol. 7, no. 1, p. Hal. 1-57, 2025, [Online]. Available: https://jurnal.ucy.ac.id/index.php/CivETech/issue/archive
[5] M. S. A. R. Rustam, Rekayasa Lalu Lintas, vol. 53, no. 9. 2019.
[6] H. Sunardi, N. Suhandi, J. R. Coyanda, and R. Rachmansyah, “Studi Pendahuluan Sistem Keamanan Parkir Berbasis Opencv Di Kampus Universitas Indo Global Mandiri,” J. Ilm. Inform. Glob., vol. 13, no. 3, 2023, doi: 10.36982/jiig.v13i3.2693.
[7] R. Gustriansyah, N. Suhandi, and F. Antony, “The design of UML-based sales forecasting application,” Int. J. Recent Technol. Eng., vol. 7, no. 6, pp. 1507–1511, 2019.
[8] Z. R. Mair and M. A. Rahmanda, “Perbandingan Versi Terbaik YOLO Dalam Mendeteksi Jarak Spasi Antar Baris Tulisan Tangan,” J. Sains, Nalar, dan Apl. Teknol. Inf., vol. 4, no. 2, pp. 103–110, 2025, doi: 10.20885/snati.v4.i2.40414.
[9] T. R. I. A. Jabar, “Pengenalan pola huruf bahasa isyarat menggunakan framework you only look once (yolo),” vol. 15, no. 2, pp. 327–335, 2025.
[10] S. P. Collins et al., “Pemrosesan Citra image Processing,” pp. 167–186, 2021.