Jurnal
Teknologi

Full Paper

VEHICLE COUNTING AND CLASSIFICATION FOR TRAFFIC DATA ACQUISITION

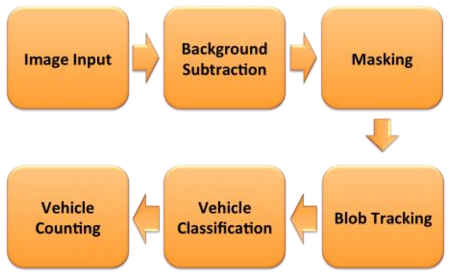
Yohanssen Pratama[a]*, IGB Baskara Nugraha[b], Eka Trisno Samosir[a]

[a]Del Institute of Technology, Sitoluama, Indonesia
Faculty of Informatics and Electrical Engineering
[b]Bandung Institute of Technology, Bandung, Indonesia
School of Electrical Engineering and Informatics

Article history
Received
8 June 2015
Received in revised form
9 September 2015
Accepted
10 December 2015

*Corresponding author
yohanssen.pratama@del.ac.id

Graphical abstract



Abstract

With current traffic condition in Indonesia where congestion and overloaded road capacity become a serious issue, we need more advanced traffic monitoring system to manage and monitor the traffic condition. In this paper, we propose a method and technique to collect the traffic data automatically by using charge-coupled device (CCD) camera as a sensor. The traffic data were collected by using the counting method that proposed and existing image processing methods in computer vision (background subtraction and tracking). Background subtraction is being used here for background modeling and detect object. After the vehicle can be detected as an object we used the two line counting method to get the traffic data (volume and velocity) that will be needed to analyze the traffic condition.

Keywords: Background; subtraction; traffic; monitoring; camera; data; counting

1.0 INTRODUCTION

If we look back to the past, there were many barriers that preventing us to use real-time video processing and acquiring some information about traffic condition around us. But today with the improvement in the hardware capability and vision computation, we have a chance to build some ergonomic and reliable system for the purpose of traffic data acquisition and monitoring.

Actually in the hardware side, now mini PC has become famous and began to enter the miniature era. We can easily found a mini PC that has a small form factor with high computing performance. This kind of PC is inexpensive and has a good capability for data processing. For digital image processing itself, the mini PC processor has an ability to execute the computer algorithm and perform an image modification. With a small form factor, we can pair the mini PC with the CCD camera or other n device in one same enclosure box. So with the mini PC technology today, it is very possible to build a transportable and affordable traffic monitoring system.

For software side actually in computer vision area, visual traffic surveillance has been attracted many people, because the prospect for being used in the Intelligent Transport System (ITS) is widely open. But there still a problem in here. Monitoring traffic in the crowded area could be a tedious task, because we need to find a method that can separate the background and foreground smoothly. In busy environment, the background scene might be fluctuated in some areas between different values. This fluctuation and dynamic environment background condition could lead us to a recent false foreground detections problem [1]. In urban area there are several moving background objects such as people who cross the street and other moving object that is not a vehicle, this event have a contribution in creating a dynamical environment condition. So in order to solve this problem we need a good method to perform the background subtraction. Instead using a good method and algorithm to prevent an unwanted object (non-vehicle moving object) being detected as foreground, we could do another way such as define some area of interest. Defining an area of interest will make our task simpler, because another object outside the area will not be detected as an object (being ignored).

Our goal in here is to create a robust system than can detect the traffic flow automatically and collect it as the data. From there, we can get real-time information about traffic volume, average velocity, and other important information. This information will be useful to manage the traffic condition.

1.1 Previous Work

There are many methods for separating foreground objects from background scenes. The famous one is background subtraction method. Background subtraction (a.k.a. background differencing) is the most fundamental operation in image processing for monitoring movement object from movie or video. But the background subtraction itself has a weakness. Each pixel which difference is high enough from their neighboring pixels will be declared as foreground pixels. However, the background scene can change every time, because a sudden change in light condition or there was a new object added or removed from the background [2]. In this case, the misinterpretation can be happen (background object could be detected as foreground).

So to handle the evolving background scene over time, we need to dynamically build a model of the background scene. It can be achieved by implementing an Exponentially Weighted Moving Average (EWMA) method. Robert (1959) is the first person who introduced the Exponentially Weighted Moving Average (EWMA) control scheme. Using simulations to evaluate its properties, he showed that the EWMA control scheme is useful for detecting small shifts in the mean of a process [3]. EWMA control scheme is based on the statistic:

$$A_t = \lambda B_t + (1 - \lambda)A_{t-1}, \quad 0 < \lambda \leq 1 \qquad (1)$$

A_t is the pixel value at a given time t, and A_{t-1} is the previous average value. A_t is a current average value that has been updated. The λ parameter is called the learning rate. If λ value is large, the moving average function will become more sensitive (The system can detect a little change in the observed values in no times).

We have tried this method for monitoring a traffic condition, but the background scene seems to be fluctuated in some area. It happened because the background area in our experimental environment (the city of Bandung) is mostly covered by many object like buildings, trees, etc. From this case we can conclude that the EWMA method only capable for extracting foreground object in simple scenes condition and relatively stable background. Besides that, in our experience, the camera must be placed on the fix and stable position, if the camera become oscillated, the effect is not good, because the background scene will be oscillated too and being detected as foreground.

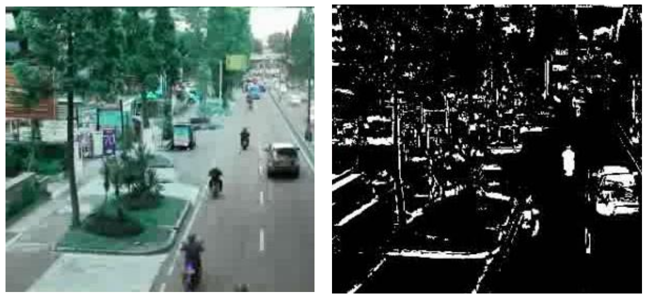


Figure 1 Left side is the current image. The right side is the result image using the EWMA method

1.2 Our Approach and Current Work

In our system, we go through three steps before able to detect moving object (figure 2). The first step is background subtraction followed by masking and blob tracking. Background subtraction is being used for background modeling and blob tracking could be used to prevent the same object being detected twice by the system (for recognize the same object in the different frame). It should be emphasized that we used an existing background subtraction algorithm, which has been developed and tested by people. So in this case, we not build a new algorithm from scratch, but only implemented an existing algorithm into our system.

For background subtraction, instead using EWMA for extract the foreground objects, we need to use more sophisticated background modeling methods. For the rural area case, where the clean background is rare due to the busy environment, it is not necessary to use EWMA. So we used the Pixel Based Adaptive Segmenter (PBAS)[4] which is base on ViBe algorithm [5] and Sigma-Delta algorithm[6]. We used PBAS because this algorithm has a good background modeling. PBAS use a non-parametric background modeling, so the background is modeled by a history of recently observed pixel values [1]. On the other hand sigma delta was used here because it was lightweight algorithm and has a good processing speed in lack performance hardware.

For tracking a blob (blob tracking), we used a tracking algorithm based on "Appearance Models for Occlusion Handling"[7] because this algorithm has a good performance in complex environment. In the last step to count volume and vehicle velocity, we proposed a two-line counting method. Until the third step we use existing algorithm but for last step (vehicle count) we tried to implement the method that we proposed. The detailed about the counting method will be discussed in the next section.

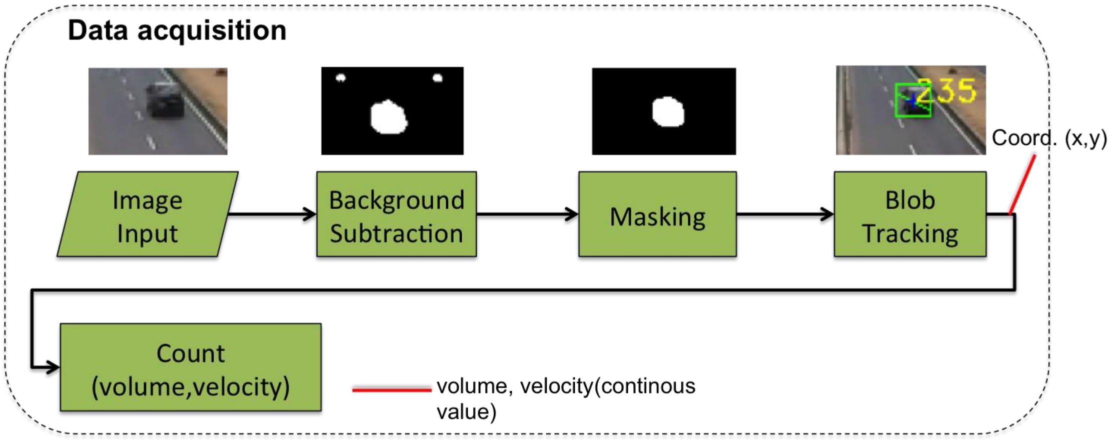


Figure 2 Block diagram for object detection

2.0 METHODS

When the video signals were captured by CCD camera, we will have sequences of image called frames. With an adequate standard computer, it is very possible to perform advanced visual analysis on images sequences. In our experiment, we used mini PC as the processing unit. Mini PC is being used here because can meet a minimum requirements that's need to perform an image analysis. All image processing step that shown by figure 2 took a place in mini PC. In this section we will describe the counting method that used to collect the traffic data volume.

2.1 Vehicle Counting

Once we could detect a vehicle as an object, the vehicle will be represented as a blob. Each blob has it own centroid. Centroid is a point where diagonals of the bounding box that is located around the blob intersect. In this case we use centroid as vehicle position in pixel coordinate (x, y).

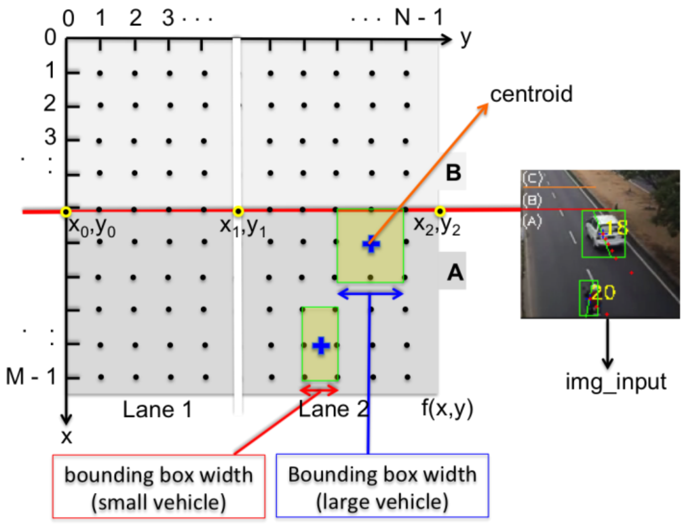

0 1 2 3 · · · · · · N - 1
y
0
1
2
3
M - 1
x
centroid
B
A
x_0,y_0
x_1,y_1
x_2,y_2
img_input
Lane 1
Lane 2
f(x,y)
bounding box width (small vehicle)
Bounding box width (large vehicle)

Figure 3 Image input representation in pixel coordinate

To be able to count the number of vehicles using centroid position, we must determine the counting line. Counting line serves to divide an area that traversed by centroid into two parts. In the figure 3 it can be seen that the counting line dividing the counting area into area A and area B. When a centroid crossed the counting line, the centroid state will be change from A to B. If the state of centroid changed from A to B, the traffic volume variable (i) which count the number of passing vehicles will be increased by one. Pseudocode below show us the centroid that move in x coordinate and passes the counting line.

```
Function Count(centroid, line, state)
1     BEGIN
2       i ← 0;
3       j ← 0;
4       if centroid.x < line.x
5         state ← A;
6           else state ← B;
7             if currentstate != oldstate;
8               i++;
9         if  y1 < lane2 < y2
10              j++;
11    END
```

If the road has more than 1 lane (in figure 3 the road has 2 lane), we can count the number of passing vehicles in lane 2 (depicted as j variable) by applied the boundary in y coordinate from y1 until y2.

2.2 Vehicle Classification

In this paper we classified the vehicle into 3 category based on the bounding box width. The small category is for small moving vehicle such as motorcycle, the medium category is for car type vehicle, and the large category is for truck or bus type vehicle. We did an averaging operation for all bounding box width value to categorize boundary value for each vehicle type (table 1).

Table 1 Vehicle by type

Vehicle Type	Bounding Box Width
Small	width < a
Medium	a ≤ width ≤ b
Large	width > b

The pseudocode to count the number of vehicles based on its type was shown below.

```
Function CountType(centroid, line, state, width)
1     BEGIN
2        i ← 0;
3        k ← 0;
4        l ← 0;
5        m ← 0;
7           if centroid.x < line.x
8              state ← A;
9           else state ← B;
10              if currentstate != oldstate;
11                 i++;
12                    if width < a
13                       k++;
14                    else if a ≤ width ≤ b
15                       l++;
16             else m++;
17    END
```

Condition which added to former pseudocode (Function Count) is a condition of vehicle category based on its bounding box width. For vehicle that has been crossed the counting line and has a width within predefined range will be counted and increased the amount of volume data that represented by variables which has been assigned into that range (see table 1) by one. Variable k is assigned as small vehicle volume, l as medium vehicle volume, and m as large vehicle volume.

2.2 Vehicle Velocity

To obtain the vehicle velocity we could use the basic motion equation. Based on the equation of motion to get velocity we need time and distance data and to get both of them we need an additional counting line for this case. Additional counting line is useful to create a distance r between the first counting line (line1) and an additional counting line itself (in Figure 4 line 2 serves as an additional counting line).

The new counting line also useful for capturing time when the vehicle (centroid) which travelled in x coordinate has the same x value with line 2 or we can say when the state of the centroid were changed (the previous counting line 1 also serves for capturing time in here). After we acquired the centroid passing time for each line, we can get the range of time t that needed by centroid to travel from line 1 to line 2. If we divide distance r with t then we will get the vehicle velocity.

So far we have already obtained the information that needed to determine the velocity, but an unit for the velocity that has been obtained was in pixel / milliseconds. We have time unit in milliseconds because we use GetTickCount function to capture the time [8]. To be able to change to another unit conversion that commonly used such as meter/second or kilometers /hour we require a conversion factor. To determine the conversion factor from pixel to meters in this study we used a comparative method directly, that is comparing the size of an object inside an image with their actual size.

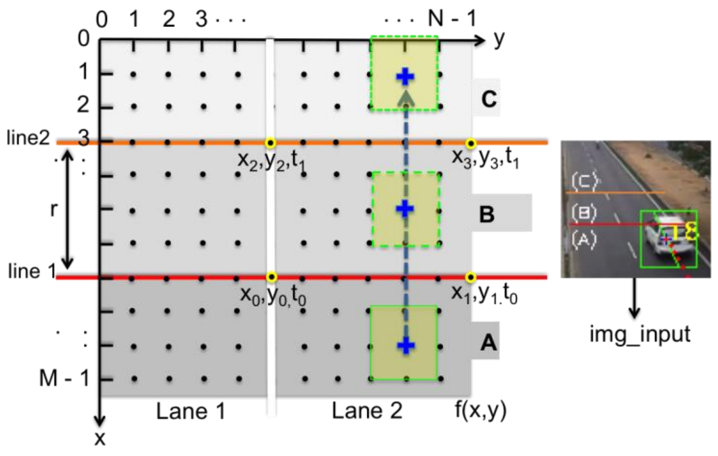


Figure 4 Additional line 2 to get the vehicle velocity

Below is pseudocode to count velocity of the vehicle that move in x coordinate. If we want automatically detect whether the vehicle move in x or y coordinate, we need to added the new if conditional in line.

```
Function Velocity(centroid, line, state, width)
1     BEGIN
2        if centroid.x < line1.x &&
      centroid.x < line2.x
3        state ← A;
4           else if centroid.x > line1.x &&
      centroid.x < line2.x
5           state ← B;
7              else centroid.x > line1.x
      && centroid.x > line2.x
8              state ←C;
9        if oldstate == A && currentstate
      == B;
10       $t_0$ = GetTickCount();
11          else if oldstate == B &&
      currentstate == C && oldstate !=A
      && currentstate !=B
12             $t_1$ = GetTickCount();
```

```
13    t=t1-t0 ;
14    v=(r/t)*conversion factor;
15    END
```

In pseudocode above when the state of centroid were change from one state to another state then the GetTickCount function will capture the current time at instant. When the state change from A → B the t_0 time will be captured and this also applies for t_1 time that will be taken when the state change from B → C.

3.0 EXPERIMENTAL RESULT

3.1 Counting Result

If we implemented the methods that has been described above to count a vehicles based on its type, we will get the vehicles number data from real time video within interval of 15 minutes as below:

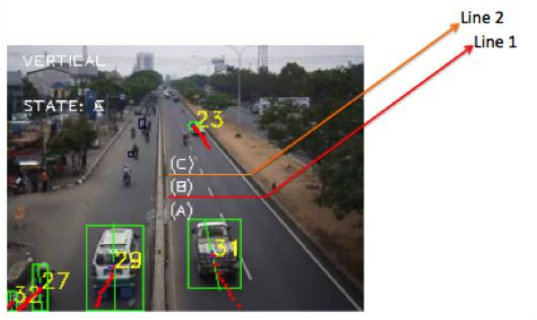


Figure 5 Counting line in image input

Table 2 Counting Result

Line	Total	Large	Medium	Small	Algorithm
Line2	433	36	286	111	PBAS
Line1	452	74	287	91	
Line2	433	51	306	76	Sigma-Delta
Line1	440	68	286	86	
	516	61	347	108	manual

In table 2 we can see the total number of vehicle based on its type that traversing the line1 and line2. Because 2 background subtraction algorithm implemented here so we will get 2 different result from each algorithm. We can compare the counting result from both algorithm with manual counting (based on human perception) and for each counting line the result is only differ by a small margin. (Figure 6).

Table 3 Algorithm Accuracy

Line	Total	Large	Medium	Small	Algorithm
Line2	83.91	59.02	82.42	97.30	PBAS
Line1	87.60	82.43	82.71	94.51	
Line2	83.91	83.61	88.18	70.37	Sigma-
Line1	85.27	89.71	82.42	79.63	Delta

3.2 Accuracy Result

We can get the counting accuracy by comparing the counting result in table 2 with manual counting. The accuracy rate for medium vehicle type seem higher than the other type. This because the medium vehicle has a relative constant bounding box size when its centroid position was captured by counting line. The bounding box size varies from large to small based on object size and the distance of the vehicle object from the camera. Sometimes when the object become too small to detect, the bounding box will disappear from it. It happened because the object has small size and its position was far away from the camera. This case is more common occurs for small vehicle type.

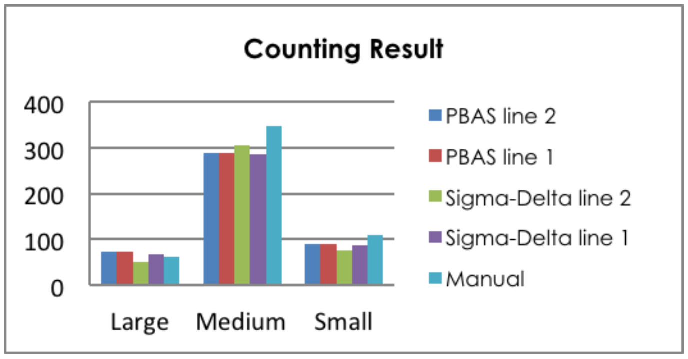


Figure 6 Counting result chart

4.0 CONCLUSION

We have tried and compared two existing algorithms (Pixel Based Adaptive Segmenter and Sigma-Delta) to separate the foreground from background object and implementing the two line counting method to collect the traffic data. The combination within background subtraction technique and the counting method that proposed could be used for collecting a vehicle volume and velocity data in rural area.

For the future work, we will try to implement the real time road traffic density classification based on the traffic data that collected using current method and an existing machine learning algorithm that has been improved.

In conclusion, this method is able to collected traffic volume data for each type of vehicle and its velocity. The background subtraction method is being used to detect an object, after that the object will be represented by centroid. By using the centroid coordinate we collected the vehicle data that passing the counting line. The result of the accuracy test shows that the volume data for medium vehicle type is almost consistent whereas the accuracy for each algorithm is within 82,42% and 88.18%. As for the other vehicle type, the minimum accuracy is 59.02% where the maximum accuracy is 97.30%. The accuracy of the all vehicle types and algorithms is about 85%.

Acknowledgement

We are grateful for the IT DEL support and all the facilities that have been provided by Blackberry Innovation Center ITB to Author 1.

References

[1] Laganiere, R. 2011. *OpenCV2 Computer Vision Application Programming Cookbook*. Packt Publishing.

[2] Bradski, G. and Kaehler, A. 2008. Learning OpenCV. O'Reilly.

[3] Lucas, J.M. and Saccucci, M.S. 1990. Exponentially Weighted Moving Average Control Schemes: Properties And Enhancements. *Technometrics.* 32: 1-12.

[4] Hofmann, M., Tiefenbacher, P. and Rigoll, G. 2012. *Background Segmentation With Feedback: The Pixel-Based Adaptive Segmenter*. IEEE Workshop on Change Detection. 38-43.

[5] Barnich, O. and Van Droogenbroeck, M. 2011. ViBe: A Universal Background Subtraction Algorithm for Video Sequences. IEEE Transactions on Image Processing. 1709-1724.

[6] Manzanera, A. and Richefeu, J. 2007. A New Motion Detection Algorithm Based On Sigma-Delta Background Estimation. *Pattern Recognition Letters*. 320–328.

[7] Senior, A., Hampapur, A., Tian, Y., Brown, L., Pankanti, S. and Bolle, R. 2006. Appearance Models For Occlusion Handling. *Image and Vision Computing.* 24: 1233-1243.

[8] Friedman, N. and Russell, S. 1997. *Image Segmentation in Video Sequences: A Probabilistic Approach. In* Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence. San Fransisco

[9] Dempster, A., Laird, N. and Rubin, D. 1977. *Maximum Likelihood From Inclomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society. 39(Series B): 1-38.

[10] Shi, J. and Tomasi, C. 1994. *Good Features to Track*. IEEE Conference on Computer Vision and Pattern Recognition. 593-600.

[11] KaewTrakulPong, P. and Bowden, R. 2001. An Improved Adaptive Background Mixture Model For Real-Time Tracking With Shadow Detection. *2nd European Workshop on Advanced Video Based Surveillance Systems*.

[12] Zivkovic, Z. 2004. Improved Adaptive Gaussian Mixture Model For Background Subtraction. In Pattern Recognition, 2004. ICPR 2004. *Proceedings of the 17th International Conference on*. 2: 28-31

[13] Grimson, W., Stauffer, C., Romano, R. and Lee, L. 1998. Using Adaptive Tracking To Classify And Monitor Activites In A Site. In: *Conference on Computer Vision and Pattern Recognition.* 22-29

[14] Tao, H., Sawhney, H. and Kumar, R. 2000. Dynamic Layer Representation With Applications To Tracking. In: *Proc. International Conference on Pattern Recognition*.

[15] Bouwmans, T., Porikli, F. Hoferlin, B and Vacavant, A. 2015. Handbook on Background Modeling and Foreground Detection for Video Surveillance. *CRC Press.*

[16] Sobral, A., Baker, C., Bouwmans, T. and Zahzah, E. 2014. Incremental and Multi-Feature Tensor Subspace Learning Applied For Background Modeling And Subtraction. *International Conference on Image Analysis and Recognition, ICIAR 2014*.

[17] Elgammal, A., Harwood, D. and Davis, L. 2000. Non-Parametric Model For Background Subtraction. *In Frame-Rate Workshop, IEEE*. 751-767

[18] Lucas, B. and Kanade, T. 1981. An Iterative Image Registration Technique With An Application To Stereo Vision. *Joint Conference in Artificial Intelligence*. 674-679.

[19] Stauffer, C. and Grimson, W. 1999. Adaptive Background Mixture Models For Real-Time Tracking. Publisher in proceedings of the *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2: 246-252.

[20] Piccardi, M. 2004. Background Subtraction Technique A Review. *In Systems, Man and Cybernetics. 2004 IEEE International Conference on*. 4: 3099-3104.