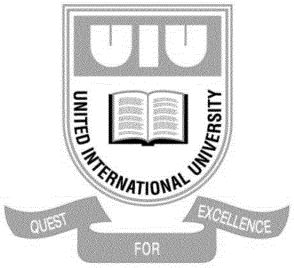
A Real-Time Video-Based Adaptive Vehicle Counting, Speed Measurement and Classification Tool in Java



Amit Ghosh (011 132 134)
Md. Shahinuzzaman Sabuj (011 132 023)
Hamudi Hasan Sonet (011 132 033)
Department of Computer Science and Engineering
United International University

A thesis submitted for the degree of
BSc in Computer Science & Engineering

January 2019

Abstract

Intelligent Transportation System (ITS) is an integral part for efficiently and effectively managing road-transport network in metros and smart cities. ITS provides several important features including public transportation management, route information, safety and vehicle control, electronic timetable and payment system etc. In this paper, we have designed and developed an adaptive video-based vehicle detection, classification, counting, and speed-measurement tool using Java programming language and OpenCV for real-time traffic data collection. It can be used for traffic flow monitoring, planning, and controlling to manage transport network as part of implementing intelligent transport management system in smart cities. The proposed system can detect, classify, count, and measure the speed of vehicles that pass through on a particular road. It can extract traffic data in csv/xml format from real-time video and recorded video, and then send the data to the central data-server. The proposed system extracts image frames from video and apply a filter based on the user-defined threshold value. We have applied MOG2 background subtraction algorithm for subtracting background from the object, which separates foreground objects from the background in a sequence of image frames. The proposed system can detect, classify, and count vehicles of different types and size as a plug & play system. We have tested the proposed system at six locations under different traffic and environmental conditions in Dhaka city, which is the capital of Bangladesh. The overall average accuracy is above 80% for classifying all types of vehicles in Dhaka city.

This work is devoted to our parents.

Acknowledgements

We would like to take this opportunity to express gratitude to our thesis supervisor Dr. Dewan Md. Farid, Associate Professor of Computer Science and Engineering, United International University for providing us guidance, supports and valuable advice in accomplishing our thesis and thanks for his constructive criticism, valuable suggestions and constant encouragement at all stages of thesis work and writing thesis book. He taught us how to work hard and growing confidence in us when we doubted ourselves and brought out the good ideas in us. Without his encouragement and constant guidance, we could not have finished this thesis. We express our sincere thanks and gratitude to Professor Dr. Chowdhury Mofizur Rahman, Vice Chancellor, United International University and Dr. Swakkhar Shatabda, Associate Professor & Undergraduate Program Coordinator United International University for their support and guidance and constant encouragement for completion of this thesis. Our special thanks also to our parents and all other teachers who supported us throughout our academic year.
We would also like to thank "a2i (Access to Information) Innovation Lab" (https://a2i.gov.bd/innovation-lab/) for considering this project, which is the winner of "National Innov-A-thon 2018" held by the ICT division (https://ictd.gov.bd) of the Government of the People's Republic of Bangladesh.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A smart city is a framework that aims at increasing the quality of urban life through improved decision-making using information and communication technologies. An intelligent transport system (ITS) is an essential facet of an advanced smart city system. The purpose of ITS is to improve safety, mobility and efficiency of the traffic system by processing information to minimize congestion, environmental impact and sustain the benefits of transportation. ITS provides the users with information about optimized travel routes, real-time traffic data and predictions, passenger density and seat availability of vehicles and many more. The working model of an ITS can be divided into stages of data collection, transmission, analysis and relay to users. Precise and prompt real-time data of traffic count, location, vehicle weight, delays etc. are collected through hardware devices such as sensors, cameras, GPS locators and transmitted over the internet to the server for further analysis. The collected data is screened, corrected, synthesized and pooled and later analyzed to predict traffic situations accordingly. Information regarding travel time, route, congestion, accidents, road blocks etc. is then extracted from the analysis and sent to the users. With the available information, users are able to make more efficient, safer and smarter use of the transportation network to suit their needs. With new ITS models emerging globally, opportunities for solving modern urbanization challenges are also growing.

1.1 Motivation

An intelligent transport system (ITS) is the basic requirement of a smart city, for doing so we need to have traffic related data. From the first day the biggest challenge we faced is lack of recourse and traffic data. This problem is also faced by other researchers. There is a saying "Necessity is the mother of invention" as this problem motivated us for doing this research work and build a tool which would help other researcher and help to build an intelligent transport system (ITS) which will be the foundation of building a smart city.

1.2 Objectives of the Thesis

In this paper, we have proposed an adaptive video-based vehicle detection, classification, counting, and speed-measurement system for real-time traffic data collection. The proposed system was built using Java programming language (https://www.java.com/) and OpenCV (https://opencv.org). The main objective for developing this system is to collect vehicle count and classification data. So that we can build intelligent transportation network based on historical traffic data. The proposed system can engender traffic data by detecting, classifying, counting, and measuring speed of vehicles and store the in csv/xml file format. It's a plug & play system and applied MOG2 algorithm as a background subtraction technique. The proposed system was tested at different six locations in Dhaka city (Dhaka is the capital of Bangladesh) under different traffic and environmental conditions. It achieved average 81% accuracy for classifying all types of vehicles in Dhaka city.

1.3 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 provides related works.

Chapter 3 presents the proposed method.

Chapter 4 discusses the results and experimental analysis.

Chapter 5 presents the conclusions, summaries the thesis contributions, and discusses the future works.

Chapter 2

Related Work

Computer vision technology is using for traffic monitoring in many countries [1, 2]. The development of computer vision technology over video based traffic monitoring for detecting moving vehicles in video streams become an essential part in ITS [3, 4]. A good number of work has been done on vehicle tracking and detection using computer vision technology. In 2005, Hasegawa and Kanade [5] introduced a system for detecting and classifying the moving objects by its type and colour. In this process, a series of images of a specific location were supplied and vehicles from these images were identified. In 2013, Nilesh et al. [6] designed and developed a system using visual C++ with OpenCV for detecting and counting moving vehicles. It can automatically identify and count moving objects as vehicle in real-time or from recorded videos, which basically used background subtraction, image filtering, image binary and segmentation method. In 2014, Da Li et al. [7] developed real-time moving vehicle detection, tracking, and counting system also using Visual C++ with OpenCV including adaptive subtracted background method in combination with virtual detector and blob tracking technology. Virtual detector constructs a set of rectangular regions in each input image frame and blob-tracking method generates input image frames, the absolute difference between the background image and foreground blobs corresponding to the vehicles on the road. The above systems have some limitations like tackling shadows, occlusion of multiple vehicles that appear in a single region. Peek Traffic Corporation (https://www.peektraffic.com/index.php) commercially developed several video traffic detection systems at the present time.

Chapter 3

Methodology

3.1 Tools and Techniques

Our proposed system is based on digital image processing technology. The application process real-time video as well as recorded video data to detect the different type of vehicles. By doing so, we can predict the density of any type of vehicles. We have used OpenCV (Open Source Computer Vision Library) for image processing. To developed the proposed system java programming language is used because it runs on a virtual machine called java virtual machine and also it is concurrent, class-based object-oriented. The main motive of this programming language is write once, run anywhere (WORA). It supported all the platform. It does not require any to change our application when we switch our platform on demand.

3.1.1 Java

Java is one of the most popular programming languages. It is a class-based, object-oriented computer programming language. Java was designed and written by James Gosling along with two other people Mike Sheridan and Patrick Naughton, while they were working at Sun Micro-systems. It was initially named oak programming language. Later java alongside with sun micro-systems was acquired by Oracle Corporation. There were five initial objectives for creating Java language. These are, Java language must be,

1. Simple, object-oriented, and familiar
2. Robust and secure

3. Architecture-neutral and portable
4. High performance and
5. Interpreted, threaded, and dynamic.

3.1.2 NetBeans

For developing this app we have used NetBeans integrated development environment (IDE) which is a good IDE for Java development. Along with Java development, NetBeans has extensions for other languages like PHP, C, C++, HTML, JavaScript and CSS. NetBeans supports Microsoft Windows, macOS and Linux. NetBeans is released under the open-source BSD license, thus it free to use and has a huge community of users and developers all over the world. Batch analyzers and converters are provided to search through multiple applications at the same time, matching patterns for conversion to new Java 8 language constructs. With a range of handy and powerful tools, the code can be easily re factored using NetBeans. It has also dragged and drop design generator, nice package manager, interactive user interface for managing project and project file. It has some automated tools for code suggestion, finding files, searching words and it also provides code templates and code generators. Making an executable file is so easy.

3.1.3 OpenCV

OpenCV [8] is an open source computer vision and Machine learning software library. OpenCV stands for Open Source Computer Vision Library. This library has more than 2500 optimized algorithm. These algorithms can be used to detect and recognize faces, human-computer interaction, objects identification, segmentation and recognition, gesture recognition, camera and motion tracking, ego-motion, motion understanding, stereo and multi-camera calibration and depth Computing classify those actions. OpenCV is released under the open-source BSD license. So it is free to use both for academic and commercial. OpenCV is written in C++. Its primary interface is in C++ but it also has Python and Java interfaces. OpenCV runs on Windows, Linux, Mac OS, iOS and Android.

3.1.4 Version Controlling

For version controlling git is used for our proposed system. Bit-bucket has been used also for the remote repository. Git is a delightful version controlling tools where you can maintain a separate branch for separate features. It's easy to rollback in any of the previous version at any time if needed. It is lightweight and easy to use.

3.2 System Overview

Our proposed system can run in two modes, recorded video mode and in real-time mode. The proposed system converts video into a sequence of image frames then extracts background and performs detection of moving objects. Video source may be of two types,

1. Recorded video
2. Real-Time video.

Initially, an area-threshold is set for vehicle detection. A count line and speed line is drawn to count and measure the speed of detected vehicles. Figure 3.1 and figure 3.2 shows the flow chart and activity diagram respectively. We can split our system into several stages. Below the stages have been described in detail.

3.2.1 System Initialization

Our proposed system is get installed and set up in this stage. In this stage we set up cameras on lamp-post or pillar of roads in an angle so that we can have the clear view of the road. Count line and speed line is drawn and we also set the distance threshold between the speed line and the count line which is the real distance in meter. We set up other thresholds like minimum size of vehicle, image threshold etc. Camera records stream of data from roads and sends to the system for further analysis.

3.2.2 Image Digitization and Background Subtraction

Background subtraction is a broadly applied method for detecting moving objects in video from static cameras. Our proposed system is using background subtraction for object extraction from video. It is faster, very dynamic and produces a better result than other pattern-based [9] detection. Pattern-based detection is good for detecting

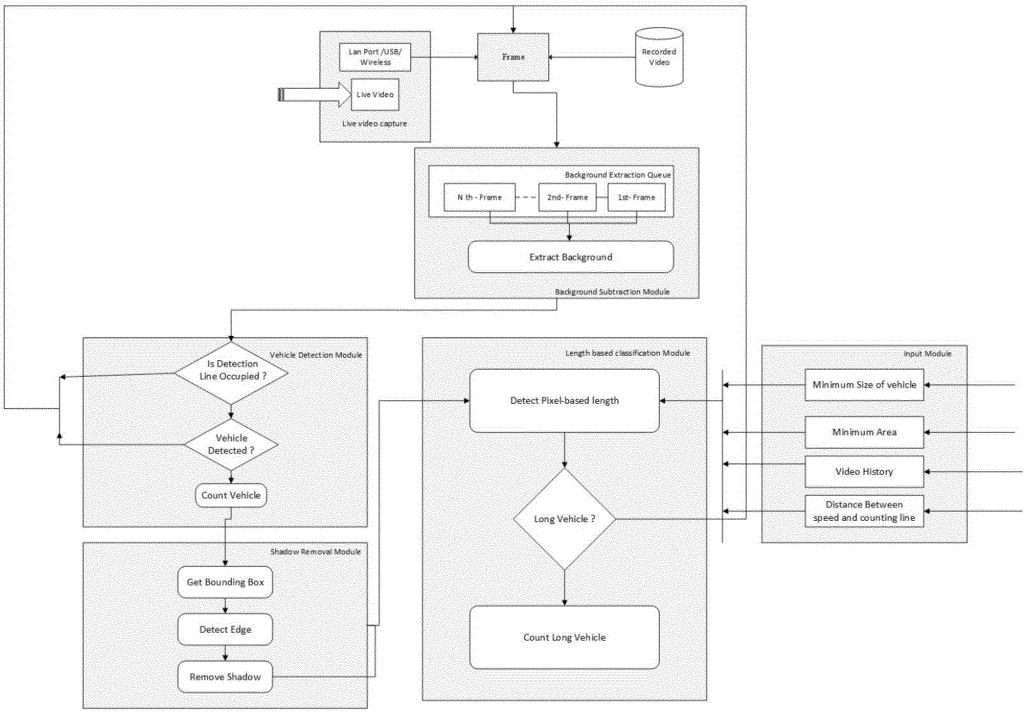


Figure 3.1: Flow chart of the proposed system.

unique objects. But when there is a variety in types of vehicles, pattern-based detection can't produce a better result. In background subtraction technique the moving foreground is extracted from a static background. Background subtraction performs a subtraction between current frame and the background model to determine the foreground mask. Figure 3.3 and 3.4 shows only the moving objects after performing background subtraction from a recorded video.

3.2.3 Vehicle Detection & Classification

After performing background subtraction, there are only the moving vehicles. The system detects each moving vehicle and the detected vehicle is surrounded with a rectangle. The size of the rectangle refers to the area of the detected vehicle. In the proposed system a minimum area can be setup as area-threshold. If the size of a moving object is greater than the defined minimum area-threshold, it will only be considered as a vehicle then, otherwise it will be ignored. That is why human will not be detected in the system. Figure 3.5 shows the detected vehicles. To classify the detected vehicles in

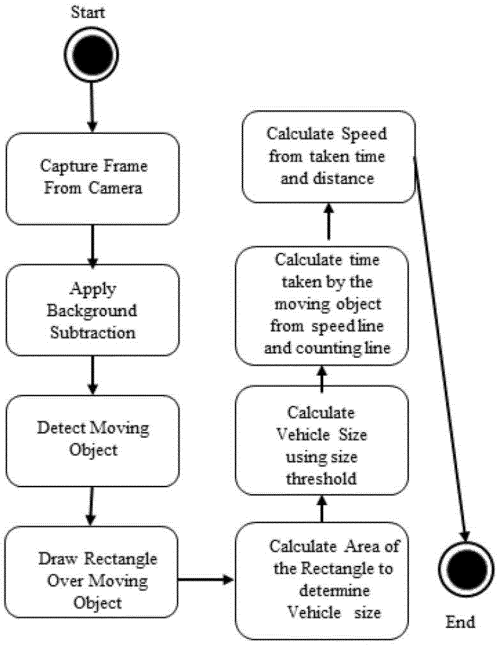


Figure 3.2: Activity diagram of the proposed system.

different types, like large vehicles, medium vehicles and small vehicles, we again consider the size. The size range has been defined earlier for the different types of vehicles in the proposed system. When a detected vehicle size lies in a particular size-range, the vehicle is classified in that specific vehicle type.

3.2.4 Vehicle Counting & Speed Measurement

While counting vehicles, it is very much important to count each vehicle only once. For this purpose, a count line has been introduced in the system. When detected vehicles pass over the count line will only be counted otherwise not.

The proposed system also has a speed line to measure the speed of moving vehicles. While configuring the system, it is needed to draw both count line and the speed line. The distance between the count line and the speed line will be fixed earlier at the time of setting up the camera . As the distance between the speed line and the count line is known and our developed system can calculate the time for crossing the speed line to

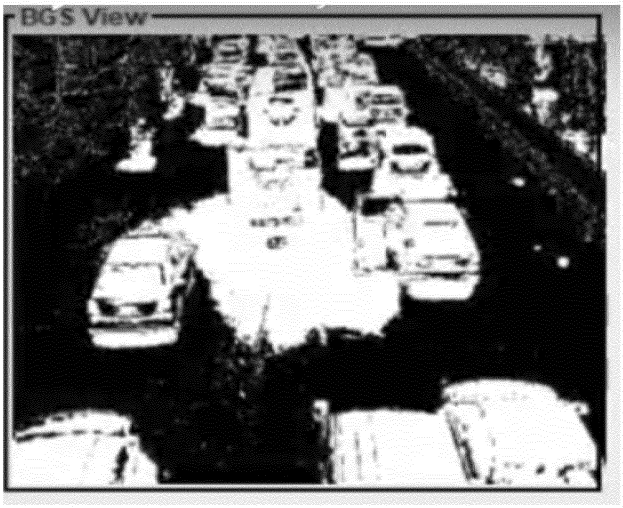


Figure 3.3: Background Subtraction Initial state

count line. Thus the system can easily calculate the speed of a vehicle. Table **??** shows some measured speed of vehicles.

3.2.5 Shadow Identification and Removal

If there is a shadow of a vehicle and the shadow is moving as well, the background subtraction considers it as moving foreground and thus the shadow may be detected as a moving vehicle. As simple background subtraction can't detect the moving shadow of vehicles, some good algorithms are introduced like BackgroundSubtractorMOG2 etc. to deal with the shadow problem. In BackgroundSubtractorMOG2 [10] algorithm, there has an option for selecting whether a shadow is to be detected or not. When detectShadows = True, the shadows are detected and BackgroundSubtractorMOG2 marks shadows and shadows are marked in gray color. For shadow identification and removal, it has been used the BackgroundSubtractorMOG2 in the proposed system.

3.3 Graphical User Interface

Figure 3.6 shows the functions and functionalities of our proposed system that have been described below:
1.1 Represents the rectangle which defines the area.

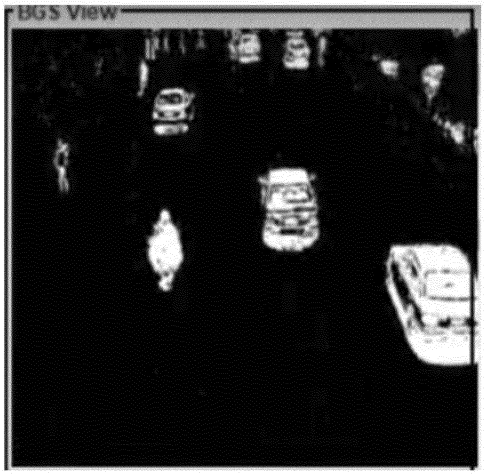


Figure 3.4: Background Subtraction Stable state

Figure 3.5: Vehicle Detection

1.2 Shows the area of the corresponding rectangle that is actually the detected vehicle size.

1.3 The red line appears is the count line. All detected vehicles pass through this line will be counted.

1.4 The speed line which is marked as green is measuring the speed of the passing vehicles.

1.5 The car enclosed by the rectangle is the detected object from the video.

2. Shows the subtracted background of objects in real time.

3. This input field indicates the location of the recorded video.

4. Although the data-set name is automated, user can set a default name of a specific

data-set.

5. Shows the button to draw the count line.

6. Noticed the button to draw the custom speed line. The line needs to cover the whole road.

7. Indicates the path to save the data-set captured via camera which we can also chose manually .

8. For loading the recorded video in the system.

9. It is to select the camera which is attached to the device.

10. Resets all the changes to initial state.

11. Plays all the process and pauses all the process.

12. Stops all the process and save the data-set in the chosen path.

13. To define the minimum size of vehicles. The value can be changed at any time according to the camera position. We have considered this threshold size to deduct the walking people in the road, tiny creatures like birds that may also appear in front of the camera.

14. To define the minimum size of objects that to be detected.

15. How much time an object will remain in the BGS view after it first appears.

16. This is a threshold value for controlling the darkness of an image. When an image is subtracted from a known background the color, contrast, brightness is a very important issue. So we need to use this threshold for adjusting the color in a different situation.

17. The path for saving video where all the recorded video will be stored. It saves current-time and date and also a determine the extension name.

19. The actual distance between the count line and speed line. Speed calculation depends on this threshold.

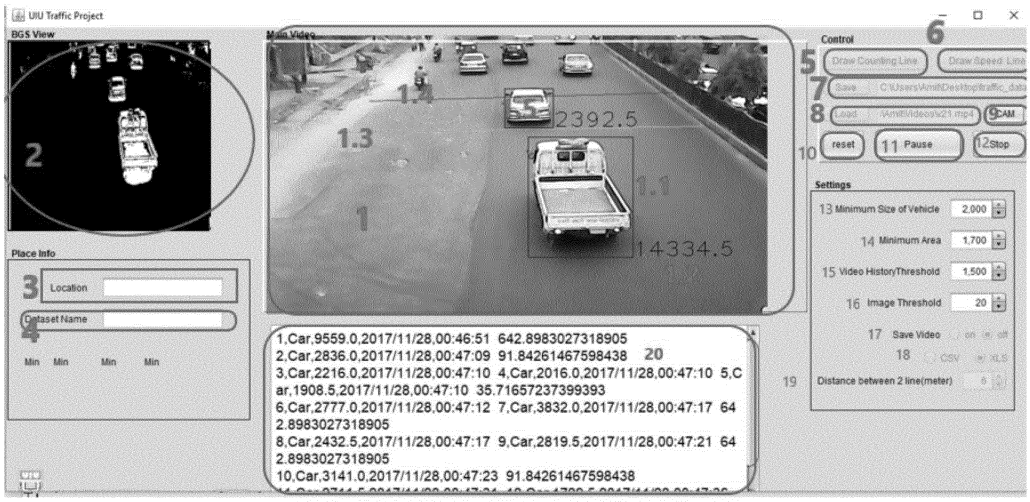


Figure 3.6: GUI of the proposed system.

Chapter 4

Experimental Tests & Result

We have tested the proposed system on a laptop powered by an Intel Core i5-3230M processor (2.6 GHZ) CPU and 8 GB RAM equipped with 720p HD web-cam on image sequences of city roads. The system is able to track and classify most vehicles successfully. We have tested the proposed system with real-time and recorded video at six different locations under different traffic and environmental conditions in Dhaka city, which is the capital of Bangladesh. Test results are tabulated in Table 4.1 and Table 4.2.

Table 4.1: Result on real-time video

Location	Data	Minute	Accuracy
Mirpur Road	1090	45	70 %
Satmosjid Road	510	15	87 %
Dhanmondi 27	460	30	89 %
Total	2320	90	82 %

Table 4.2: Result on recorded video

Location	Data	Minute	Accuracy
Farmgate	380	20	70 %
Gulsan 2	690	20	87 %
Dhanmondi 15	460	25	85 %
Total	1530	65	81 %

Our experiment ends with generating a dataset. We have a sample dataset that is generated by our tool. Below there is dataset description and the description of the attribute of the dataset.

4.0.1 Data Set Description

Title: Traffic Dataset

Source: Our Developed System

Dataset Characteristic: Multivariate

Attribute Characteristic: Numeric, Nominal

Area: Sat Mosjid Road, Dhaka

4.0.2 Data Set Information

The data set contains 6 attributes and 500 instances.

Table 4.3: Attribute Description

Attribute Name	Type	Description
No.	numeric	Number of vehicles detected
Vehicle type	nominal	Different types of vehicles
Speed [km/h]	numeric	Speed of vehicles
Size	numeric	Different vehicles size
Date	yyyy/mm/dd	Date of data collection
Time	hh:mm:ss	Time of data collection

Our system has generated some sample dataset from the collected video and real-time video. The below table is referring to some sample instances from 500 instances. We have run the above dataset in weka, generated by our developed tool. And we observed the ratio of vehicle moving in the road of Dhaka city. Figure 4.1 reffers the ratio of vehicles using the data extract from our proposed system.

4.1 Summary

In this chapter, we have discussed the environment where the developed tool runs. We have tested the system on real-time video and recorded video on different places in Dhaka and find out the accuracy. It is quite good. We learned that our developed tool can generate dataset in csv or XML format from video frames. Our experimental results indicate that the method we have proposed is effective to detect, track, measure speed

Table 4.4: Sample Traffic Dataset

SL No.	Vehicle type	Speed[Km/h]	Size	Date	Time
1	BUS	28.4818212	11278	2017/10/30	15:51:52
2	CNG	38.618212	2941.5	2017/10/30	15:52:02
3	BIKE	25.64134813	1823	2017/10/30	15:52:05
4	CAR	31.84826663	4158	2017/10/30	15:52:07
5	BIKE	46.73896764	1883.5	2017/10/30	15:52:10
6	BIKE	53.20573051	1789.5	2017/10/30	15:52:10
7	CNG	27.17428227	2887	2017/10/30	15:52:11
8	CAR	34.76447177	6135	2017/10/30	15:52:18
9	CAR	60.09590226	7969.5	2017/10/30	15:52:23
10	CNG	54.68656872	2444	2017/10/30	15:52:25
11	BIKE	48.71949885	1895.5	2017/10/30	15:52:25
12	BIKE	52.64005541	1728	2017/10/30	15:52:25
13	CNG	41.53079131	2792.5	2017/10/30	15:52:27
14	CAR	46.39157098	3727	2017/10/30	15:52:28

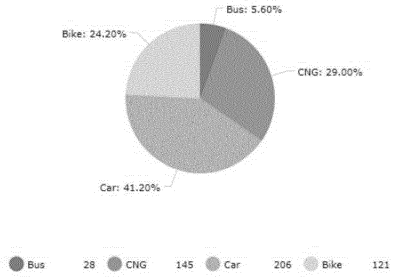


Figure 4.1: Transport Mediums

and count moving vehicle accurately. In the next chapter, we discussed the conclusion and future work.

Chapter 5

Conclusions & Future Work

The demands of Intelligent Transport System (ITS) are increasing gradually and steadily. We have developed video-based vehicle detection, classification, counting, and speed-measurement adaptive system for real-time traffic data collection. We have used BackgroundSubtractionMoG2 algorithm, OpenCV, and Java SE Development Kit 8 for developing the system. We have used git for remote repository. In the proposed system, we have considered all day and night shadowing, and different lighting situations. Also, we have considered the moving shadow of moving vehicles. We have tested the proposed system at three different locations under different traffic and environmental conditions in Dhaka city. The proposed system can generates traffic data by analyzing video with classification of vehicle type, vehicle length, vehicle speed, and time & date. It can generate traffic data in csv or xml file format. It's a plug & play system. The experimental results indicates that the proposed system can effectively detect, classify, count, and measure speed of moving vehicles with above 80% accuracy. In future, we will collect more traffic data from different locations of Dhaka city using our proposed system and will apply several machine learning algorithms for mining traffic patterns of Dhaka city.

Bibliography

[1] S. Gupte, O. Masoud, R. Martin, and N. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 1, pp. 37–47, August 2002. 4

[2] J. J. Kim, S. Smorodinsky, M. Lipsett, B. C. Singer, A. T. Hodgson, and B. Ostro, "Traffic-related air pollution near busy roads: the east bay children's respiratory health study," *American journal of respiratory and critical care medicine*, vol. 170, no. 5, pp. 520–526, May 2004. 4

[3] A. H. Lai, G. S. Fung, and N. H. Yung, "Vehicle type classification from visual-based dimension estimation," *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, pp. 201–206, August 2001. 4

[4] A. Rhodes, D. Bullock, J. Sturdevant, Z. Clark, and D. C. Jr, "Evaluation of the accuracy of stop bar video vehicle detection at signalized intersections," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1925, pp. 134–145, 2005. 4

[5] O. Hasegawa and T. Kanade, "Type classification, color estimation, and specific target detection of moving targets on public streets," *Machine Vision and Applications*, vol. 16, no. 2, pp. 116–121, December 2005. 4

[6] N. J. Uke and R. C. Thool, "Moving vehicle detection for measuring traffic count using OpenCV," *Journal of Automation and Control Engineering*, vol. 1, no. 4, pp. 349–352, December 2013. 4

[7] D. Li, B. Liang, and W. Zhang, "Real-time moving vehicle detection, tracking, and counting system implemented with OpenCV," *2014 4th IEEE International Conference on Information Science and Technology*, pp. 631–634, October 2014. 4

[8] "Opencv user site: http://opencv.org/." 6

[9] S. Deep, R. A. Shaikh, J.-P. Li, M. H. Memon, A. Khan, and Z. Tao, "Pattern based object recognition in image processing," *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP)*, pp. 276–279, December 2014. 7

[10] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," *Video-based surveillance systems*, pp. 135–144, 2002. 10

Appendix A

Code

A.1 Main Classes

In this Appendix section we will attached some of our main classes that we have implemented.

A.1.1 Class Count Vehicles

```
package trafficAnalizer.classes;
import java.util.ArrayList;
import java.util.List;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import trafficAnalizer.model.Veichle;
public class CountVehicles {
public CountVehicles(int areaThreshold, int vehicleSizeThreshold, Point
lineCount1, Point lineCount2, Point lineSpeed1, Point lineSpeed2, boolean
crossingLine, boolean crossingSpeedLine) {
this.areaThreshold = areaThreshold;
this.vehicleSizeThreshold = vehicleSizeThreshold;
this.lineCount1 = lineCount1;
this.lineCount2 = lineCount2;
this.lineSpeed1 = lineSpeed1;
this.lineSpeed2 = lineSpeed2;
this.crossingLine = crossingLine;
this.crossingSpeedLine = crossingSpeedLine;
```

```
this.checkRectLine = new CheckCrossLine(lineCount1, lineCount2);
this.checkSpeedLine = new CheckCrossLine(lineSpeed1, lineSpeed2);
}

public Mat findAndDrawContours(Mat image, Mat binary) {
ArrayList<MatOfPoint> contours = new ArrayList<MatOfPoint>();
this.image = image;
Imgproc.findContours(binary, contours, new Mat(), 1, 2);
Imgproc.line(image, this.lineCount1, this.lineCount2, new Scalar(0.0, 0.0, 255.0), 1);
Imgproc.line(image, this.lineSpeed1, this.lineSpeed2, new Scalar(0.0, 255.0, 0.0), 1);
for (int i = 0; i < contours.size(); ++i) {
MatOfPoint currentContour = contours.get(i);
double currentArea = Imgproc.contourArea(currentContour);
if (currentArea <= (double) this.areaThreshold) {
continue;
}
this.goodContours.add(contours.get(i));
this.drawBoundingBox(currentContour, currentArea);
}
return image;
}

public boolean isVehicleToAdd() {
for (int i = 0; i < this.goodContours.size(); ++i) {
Rect rectangle = Imgproc.boundingRect(this.goodContours.get(i));
if (!this.checkRectLine.rectContainLine(rectangle)) {
continue;
}
this.contourVehicle = this.getGoodContours().get(i);
this.countingFlag = true;
break;
}
if (this.countingFlag) {
if (!this.crossingLine) {
this.crossingLine = true;
return true;
}
return false;
}
this.crossingLine = false;
return false;
```

```
}

public Veichle classifier() {
double currentArea = Imgproc.contourArea(this.contourVehicle);
System.out.println(currentArea);

//here we apply our techniqe to classify our data
//we can use like this y=b0+b1*x
if (currentArea <= (double) this.vehicleSizeThreshold) {
return new Veichle(currentArea, "Car");
}

if (currentArea <= 1.9 * (double) this.vehicleSizeThreshold) {
return new Veichle(currentArea, "Van");
}

return new Veichle(currentArea, "Lorry");
}

private void drawBoundingBox(MatOfPoint currentContour, double area) {
Rect rectangle = Imgproc.boundingRect(currentContour);
Imgproc.rectangle(this.image, rectangle.tl(),rectangle.br(), new Scalar(255,0,0),1);
Imgproc.putText(image, +area + "", rectangle.br(), 1, 2, new Scalar(0.0, 0.0, 0.0));
}

public boolean isToSpeedMeasure() {
for (int i = 0; i < this.goodContours.size(); ++i) {
Rect rectangle = Imgproc.boundingRect(this.goodContours.get(i));
if (!this.checkSpeedLine.rectContainLine(rectangle)) {
continue;
}
this.speedFlag = true;
break;
}
if (this.speedFlag) {
if (!this.crossingSpeedLine) {
this.crossingSpeedLine = true;
return true;
}
return false;
}
```

```
this.crossingSpeedLine = false;
return false;
}

public boolean isCrossingSpeedLine() {
return this.crossingSpeedLine;
}

public boolean isCrossingLine() {
return this.crossingLine;
}

public List<MatOfPoint> getGoodContours() {
return this.goodContours;
}

private Mat image;
public List<MatOfPoint> goodContours = new ArrayList<MatOfPoint>();
private int areaThreshold;
private int vehicleSizeThreshold;
private Point lineCount1;
private Point lineCount2;
private Point lineSpeed1;
private Point lineSpeed2;
CheckCrossLine checkRectLine;
CheckCrossLine checkSpeedLine;
boolean countingFlag = false;
boolean speedFlag = false;
boolean crossingLine;
boolean crossingSpeedLine;
MatOfPoint contourVehicle;
}
```

A.1.2 Check Cross Line

```
\begin{lstlisting}
package trafficAnalizer.classes;
import org.opencv.core.Point;
import org.opencv.core.Rect;

public class CheckCrossLine {
```

```
public int lineAx;
public int lineAy;
public int lineBx;
public int lineBy;
public double a;
public double b;
Point line1;
Point line2;

public CheckCrossLine(Point l1, Point l2) {
this.line1 = l1;
this.line2 = l2;
this.lineAx = (int) Math.min(l1.x, l2.x);
this.lineAy = (int) Math.min(l1.y, l2.y);
this.lineBx = (int) Math.max(l1.x, l2.x);
this.lineBy = (int) Math.max(l1.y, l2.y);
}

public boolean rectContainLine(Rect rect) {
int PrA = (int) ((rect.tl().x + rect.br().x) / 2.0);
int PrB = (int) ((rect.tl().y + rect.br().y) / 2.0);
int pktCy = (int) rect.tl().y;
int pktDy = (int) rect.br().y;
int pktEx = (int) rect.tl().x;
int pktFx = (int) rect.br().x;
if (this.lineBx != this.lineAx && this.lineBy != this.lineAy) {
this.a = (this.line2.y - this.line1.y) / (this.line2.x - this.line1.x);
this.b = this.line1.y - this.a * this.line1.x;
int crax = PrA;
double cray = this.a * (double) PrA + this.b;
double crbx = ((double) PrB - this.b) / this.a;
int crby = PrB;
if(this.lineAx<=crax && this.lineBx>=crax && (double)
this.lineAy <= cray
&& (double) this.lineBy >= cray
&& (double) pktCy <= cray && (double) pktDy >= cray) {
return true;
}
if ((double) this.lineAx <= crbx && (double) this.lineBx >= crbx
&& this.lineAy <= crby && this.lineBy >= crby
```

```
&& (double) pktEx <= crbx && (double) pktFx >= crbx) {
return true;
}
return false;
}
return false;
}
}
```

A.1.3 Image Processor

```
package trafficAnalizer.classes;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import org.opencv.core.Mat;
public class ImageProcessor {
public BufferedImage toBufferedImage(Mat matrix) {
int type = 10;
if (matrix.channels() > 1) {
type = 5;
}
int bufferSize = matrix.channels() * matrix.cols() * matrix.rows();
byte[] buffer = new byte[bufferSize];
matrix.get(0, 0, buffer);
BufferedImage image = new BufferedImage(matrix.cols(), matrix.rows(), type);
byte[] targetPixels = ((DataBufferByte) image.getRaster().getDataBuffer()).getData();
System.arraycopy(buffer, 0, targetPixels, 0, buffer.length);
return image;
}
}
```

A.1.4 Video processor

```
package trafficAnalizer.classes;

import org.opencv.core.Mat;

public interface VideoProcessor {

public Mat process(Mat var1);
```

```
public void setImageThreshold(double var1);

public void setHistory(int var1);
}
```

A.1.5 Mixture of Gaussian Background

```
package trafficAnalizer.classes;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;
import org.opencv.video.BackgroundSubtractorMOG2;
import org.opencv.video.Video;

import java.awt.*;

public class MixtureOfGaussianBackground
implements VideoProcessor {

private BackgroundSubtractorMOG2 mog;
private Mat foreground = new Mat();
private double learningRate = 0.001;

public MixtureOfGaussianBackground(double imageThreshold, int history) {
this.mog = Video.createBackgroundSubtractorMOG2(history, imageThreshold, true);
this.mog.setShadowValue(0);
}

@Override
public Mat process(Mat inputImage) {
//Imgproc.cvtColor(inputImage,inputImage, Imgproc.COLOR_RGB2GRAY);
this.mog.apply(inputImage, this.foreground, this.learningRate);
return this.foreground;
}

@Override
public void setImageThreshold(double imageThreshold) {
this.mog.setVarThreshold(imageThreshold);
this.mog.setVarThresholdGen(imageThreshold);
}
```

```
@Override
public void setHistory(int history) {
this.mog.setHistory(history);
}
}
```

A.1.6 Resetting Default

```
package trafficAnalizer.classes;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import jxl.write.WriteException;
import org.opencv.core.Size;
import org.opencv.videoio.VideoWriter;
import trafficAnalizer.Loop;
import trafficAnalizer.MainGUI;

public class Reseting
implements Runnable {

MainGUI view;

public Reseting(MainGUI view) {
this.view = view;
}

@Override
public void run() {
while (view.lineSpeed2 == null || view.lineCount2 == null) {
}
view.playPauseButton.setEnabled(true);

view.resetButton.setEnabled(true);
view.resetButton.setEnabled(true);
view.onButton.setEnabled(false);
view.offButton.setEnabled(false);
view.xlsButton.setEnabled(false);
view.csvButton.setEnabled(false);
if (view.saveFlag.equals("On")) {
view.videoWriter = new VideoWriter(view.savePath +"Video.avi",
```

```
VideoWriter.fourcc('P', 'I', 'M', '1'),view.videoFPS, new Size(640.0, 360.0));
}
Thread mainLoop = new Thread(new Loop(view));
mainLoop.start();
String xlsSavePath = view.savePath + "\\Results.xls";
view.fileToSaveXLS = new File(xlsSavePath);
try {
view.writeToExel(view.fileToSaveXLS);
} catch (IOException | WriteException e) {
e.printStackTrace();
}
if (!view.isExcelToWrite) {
String csvSavePath = view.savePath + "\\Results.csv";
try {
view.filetoSaveCSV = new FileWriter(csvSavePath);
view.writeToCSV(view.filetoSaveCSV);
} catch (IOException e) {
e.printStackTrace();
}
}
view.isWritten = false;
}
}
```

A.1.7 Live data visualization

```
package trafficAnalizer;

public class LiveDataVisualization extends javax.swing.JFrame {

/**
* Creates new form LiveDataVisualization
*/
public LiveDataVisualization() {
initComponents();
}

/**
* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
```

```
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGap(0, 880, Short.MAX_VALUE)
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGap(0, 631, Short.MAX_VALUE)
);

pack();
}// </editor-fold>//GEN-END:initComponents

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
try {
for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager
.getInstalledLookAndFeels()) {
if ("Nimbus".equals(info.getName())) {
javax.swing.UIManager.setLookAndFeel(info.getClassName());
break;
}
}
} catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(LiveDataVisualization.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(LiveDataVisualization.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(LiveDataVisualization.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(LiveDataVisualization.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new LiveDataVisualization().setVisible(true);
}
});
}

// Variables declaration - do not modify//GEN-BEGIN:variables
// End of variables declaration//GEN-END:variables
}
```