Traffic Monitoring System Using Open-CV

B.E.(CIS)PROJECTREPORT

By

Arisha Yazdani (Group Leader)

Department of Computer and Information Systems Engineering

N.E.D. University of Engg. & Tech.
Karachi-75270
Traffic Monitoring System Using Open-CV

B.E.(CIS)PROJECTREPORT

ProjectGroup:

Arisha Yazdani   CS-26
Hira Butool      CS-19
Anam Siddiqui    CS-51
Zaira Azmat      CS-54

BATCH: 2007-08

ProjectAdvisor(s):

Mr. Shahab Tahzeeb(InternalAdvisor)
Mr. Ali Zaidi (ExternalAdvisor)

November, 2011

Department of Computer and Information Systems Engineering

N.E.D. University of Engg. & Tech.
Karachi-75270
Traffic Monitoring System (TMS) is an integrated framework that uses live or recorded video footage to perform traffic surveillance on public roads and highways. Extending from conventional systems, which require manual monitoring and analysis on limited points, our system provides a 24/7 automated and reliable intelligent monitoring at remote strategic locations. This system consists of three main parts: video processing unit, data analysis and web application. Video processing unit focuses on multiple object detection, object tracking, speed measurement and object counting based on video sequences obtained from a video camera feed, located on a road. In Data Analysis, the gathered data is processed and relevant information is extracted to deduce the traffic congestion on the road. Web Application is developed to improve current traffic system and provide users current traffic updates on click.
ACKNOWLEDGEMENTS

First of all our heartiest thanks go to All Almighty Allah, who gave us the strength and confidence to undertake and accomplish this project.

We are thankful to our course instructor Ms. Saneeha Ahmed for providing us with the idea of this project. Furthermore we are deeply grateful to Mr. Shahab Tehzeeb (internal advisor) for his guidance, support and good nature throughout the project.

We are also thankful to Mr. Ali Zaidi (external advisor) for his consistent support and assistance throughout the term.

Finally we also owe a debt of gratitude to our families, for their patience, support and understanding.
<table>
<thead>
<tr>
<th>S.No.</th>
<th>CONTENTS</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.1 Project Overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.2 Problem Statement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.3 Motivation and Need</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.4 Project Objectives</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Project Description</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.1 Scope</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.2 Purpose</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.3 Domain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.4 Software Process Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5 System Requirement</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.5.1 Software Interfaces</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5.2 Hardware &amp; Communication Interfaces</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.6 Open-CV</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.6.1 C# bindings for Open-CV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.6.2 Open-CV VS MATLAB</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Software Design</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3.1 Use Case Diagrams</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2 Sequence Diagram</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3.3 Collaboration Diagram</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.4 State Transition Diagram</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Project Modules</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>4.1 Modules</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.1.1 Module 1 (Video processing unit)</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.1.2 Module 2 (Data Analysis)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.1.3 Module 3 (Web Application)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Video Processing and Motion Tracking</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5.1 Video Processing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.2 Tracking Motion of Vehicle</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>5.2.1 Tracking of Vehicles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.2.2 Corner Detection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.3 Corner Detection VS Edge Detection</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>5.4 Optical Flow</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>5.4.1 The Shi-Tomasi Corner Detection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.4.2 The change implemented in Harris Corner Detector</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.5 Good Feature to Track</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5.5.1 CvGoodFeaturesToTrack function definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.6 Alternatives</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>5.6.1 Lucas Kanade Tracking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.6.2 Motion Detection using Aforge.Net</td>
<td>19</td>
</tr>
</tbody>
</table>
6 Data Analysis

6.1 Data Analysis
   6.1.1 Processing the Data Obtained From Traffic Video

6.2 Noise Filtration of the Tracked Points by CVGoodFeaturestoTrack

6.3 Clustering
   6.3.1 What is Clustering?
   6.3.2 Need Of Clustering In The Project
   6.3.3 Extraction of number of Clusters
   6.3.4 Algorithmic steps for QT clustering
   6.3.5 Advantages of QT Algorithm

6.4 Why Not K-Means Clustering?

6.5 Database
   6.5.1 Design of Location Table
   6.5.2 Design of User Login Table

7 TraffixGem

7.1 Website Overview
7.2 Google Maps
7.3 Short Message Service (SMS)
7.4 Registration Form for SMS

8 Future Enhancement

8.1 Alternative Routes
8.2 Customized SMS
8.3 Filtering noise by means of Kalman or Particle Filter

9 Appendix A

10 Appendix B

11 References
CHAPTER 1

Introduction

1.1 Project Overview

The project “Traffic Monitoring System” proposes an automatic visual monitoring system that uses video sequences of traffic footage to detect and track multiple vehicles and deduce the traffic congestion on roads.

This project is prompted by the need of reducing the cost of collecting transport data using manual observations with computer vision technology. Computer Vision is the science and technology of machines that see (where see in this case means that the machine is able to extract information from an image that is necessary to solve some task). As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

There has been widespread attention in advancing surveillance systems using computer vision technology. Most of the research is mainly related to detect moving objects for pedestrians and traffic surveillance.

1.2 Problem Statement

Every road is designed to support a certain volume of traffic. As the number of vehicles increase, the road network must be upgraded to avoid congestion. However, in some instances, upgrading the road network is difficult or in-feasible. A good example is downtown Karachi where the road network design is almost a century old and cannot be upgraded due to the relics of construction on either side of the road.
Traffic congestion occurs when there is a sudden and often unpredictable surge in the volume of traffic that saturates the capacity of the road network. This saturation is often caused by a small accident. The situation is more volatile during rush hours when the volume of the traffic is already stretching the road network capacity and even the slightest disturbance renders the system out of control.

1.3 Motivation and Need

- Rapid growth of traffic flow has caused traffic monitoring job to become harder.
- Current available surveillance system which is based on electronic sensors and manual observations by human operators may lead to unreliable traffic analysis.
- With the increasing capacity of digital storage, improved processor power, better internet connection, infrastructures and new video compression standards make computer vision one of the best choice for automatic monitoring solution.
- The ultimate need of this project is to improve current system and reduce human errors by implementing computer vision technology.
1.4 Project Objectives

Following are the main objectives of this project.

- To avoid the traffic congestion problem.
- To provide up-to-date information for enquiries about traffic conditions.
- Design a web service which will provide current traffic status powered by Google map.
- To facilitate the travelers, an SMS service will be provided.
CHAPTER 2

Project Description

2.1 Scope

Traffic Congestion is a severe problem in many modern cities like Karachi. It has been causing many critical problems and challenges in the major and most populated cities. Travelling among different places within the city has become a hectic situation due to traffic jams. Therefore, we are designing an online service; Traffic Management System to minimize these problems and letting us know the current traffic situations in the city.

2.2 Purpose

As discussed earlier, we have designed a system that will minimize the problems faced by the travelers. Our system will help travelers in a way that before reaching a specific location, a person can enquire about the traffic situation there.

2.3 Domain

Our project is not domain specific. Currently we have designed it for travelers of Karachi but it can be used for other cities as well.

2.4 Software Process Model

The software process model is a sequence of steps which should be followed in order to maintain and develop the project. We have used Water Fall model for our project. Waterfall Model consists of five main steps; Requirement, Design, Implementation, Verification and Maintenance.
2.5 System Requirements

Following are the hardware and software requirements for the project.

2.5.1 Software Interfaces:

- SQL server 2005
- Visual studio 2010
- ASP.NET
- C#.NET

2.5.2 Hardware & Communication Interfaces:

- Webcam/CCTV camera to be used for live video feed and processed for deducing the traffic congestion on roads.
- The communication interface includes e-mail, web browser (preferably Google Chrome or Firefox).
2.6. Open-CV

- Open-CV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel.
- It consists of many algorithms and examples for image processing and machine vision.
- Open-CV is now available in the form of dynamic link library (DLL).

2.6.1 C# Bindings for Open-CV:

There are many platforms available for binding of Open-CV with C#.net some of them are:

- Emgu-CV
- Open-CVSharp
- Open-CVDotNet
- Open-CV Code Project

In Traffic Monitoring System we have used Emgu-CV and Code Project library for C#.net binding.

2.6.2 Open-CV vs. MATLAB:

MATLAB is built upon Java and Java is built upon C. Whenever MATLAB program runs, computer gets busy, trying to interpret the code in two steps.

- It converts it into Java
- Finally executes the code, thus MATLAB is slower.

Whereas, Open-CV is developed for image processing in C/C++ environment.
CHAPTER 3

Software Design

3.1 Use Case Diagram

User

View Traffic Updates

View Services

Register For SMS Updates

Contact Admin

Contact Service

View Company Profile

Figure 3.1 User-end (Website)
3.2 Sequence Diagram

![Sequence Diagram](image)

**Figure 3.2** Admin-end (Window Application)

**Figure 3.3** User-end (Website)
3.3 Collaboration Diagram

![Collaboration Diagram](image)

**Figure 3.4** User-end (Website)

3.4 State Transition Diagram

![State Transition Diagram](image)

**Figure 3.5** Admin-end (Window Application)
CHAPTER 4

Project Modules

4.1 Modules

TMS- Traffic Gem has been divided into three main modules:

1. Video processing
2. Data Analysis
3. Web development
4. *Embedding SMS Service (Bonus Module)

Fig 4.1 Proposed System of Traffic Management System
4.1.1 Module 1 (Video processing unit)

This is developed using C# language. Video processing is based on open source computer vision library, Open-CV.

4.1.2 Module 2 (Data Analysis)

The second objective was to store the information obtained from video processing in Microsoft SQL database server.

4.1.3 Module 3 (Web Application)

The third objective of this project was to build a WEB application that consists of database server and Graphical User Interface (GUI).
Chapter 5

Video Processing and Motion Tracking

5.1 Video Processing

- Window application is developed using C# language. Video processing is based on open source computer vision library, Open-CV.
- Video processing is done with the help of Capture class of Emgu-CV (Open-CV).
- Capture Class take three parameters as argument.
  1. File Name of the recorded video.
  2. Null in case of Webcam
  3. Index in “integer” for any other kind of camera for Live Streaming

- In our Project we are using Recorded Videos for testing purpose.
- The Frame Rate for capturing frames is 1 second, i.e. after every one second a frame is captured and then that frame is processed for Motion detection
- In our project we have set a total of 60 frames for Processing

5.2 Tracking the motion of vehicle

No feature-based vision system can work until good features can be identified and tracked from frame to frame. Although tracking itself is by and large a solved problem, selecting features that can be tracked well and correspond to physical points in the world is still an open problem. [1]

5.2.1 Tracking of Vehicles

Every Object has certain distinct features with which it can be tracked and worked upon. Thus by observing the characteristic and properties of vehicles, we observed that every vehicle can be tracked by its corners while moving on the road.
5.2.2 Corner Detection

It is an approach used within computer vision systems to extract certain kind of features and infer the contents of an image. Corner detection is frequently used in motion detection, image registration, video tracking, 3D modeling and object recognition. [2]

Fig 5.1 the points in circles are good points to track, whereas points in boxes (edges) are poor choices

5.3 Corner Detection vs. Edge Detection

Corner detection works on the principle that if you place a small window over an image and if it is placed on a corner and moved in any direction then there will be a large change in intensity.

If the window is over a flat area of the image then there will obviously be no intensity change when the window moves. If the window is over an edge there will only be an intensity change if the window moves in one direction because an edge is not a physical entity, just like a shadow. It is where the vertical and the horizontal surfaces of an object meet.
If the window is over a corner then there will be a change in all directions, and therefore a corner can be defined as the intersection of two edges.

5.4 The Optical Flow

It is used to determine the movement of objects from the current image with respect to the last image. Note that this means more than one image is needed in order to use Optical Flow.

The Optical Flow of an image is defined as how parts of the image move with respect to the previous image. Each part of the image is broken into square patches and searched for the best match within the previous image. Once the best match has been identified a connection between where that patch current is and where it was is draw as a line. These lines are often referred to as needles due to their appearance when viewed in a moving image.
Optical flow can be implemented using either Shi-Tomasi or Lucas-Kanade Algorithm.

5.4.1 The Shi-Tomasi Corner Detector

The Shi-Tomasi corner detector is based entirely on the Harris Corner Detector. However, one slight variation in selection criteria made this detector much better than the original. It works quite well where even the Harris Corner Detector fails. So here’s the minor change that Shi and Tomasi did to the original Harris Corner Detector.

5.4.1.1 The Change implemented in Harris Corner Detector

The Harris Corner Detector has corner selection criteria. A score is calculated for each pixel, and if the score is above a certain value, the pixel is marked as a corner. The score is calculated using two eigen values. That is, you gave the two eigen values to a function. The function manipulates them, and gave back a score. [3]

Shi and Tomasi suggested that the function should be done away with. Only the eigen values should be used to check if the pixel was a corner or not.

5.5 Good Features to track:

In order to track the Vehicle we used the function Good Features to track. CVGoodFeaturesToTrack is also called Shi-Tomasi Method.

- The function first calculates the minimal eigen value for every pixel of the source image.
- 16 -

- The function CVGoodFeaturesToTrack finds corners with big eigen values in the image.
- A corner can be defined as the intersection of two edges.
- The function ensures that all the corners found are distanced enough from one another by getting two strongest features and checking that the distance between the points is satisfactory.
- This function also implements Harris corner detection.

5.5.1 CVGoodFeaturesToTrack Function Definition

```csharp
public static void cvGoodFeaturesToTrack( IntPtr image, IntPtr eigImage, IntPtr tempImage, IntPtr corners, ref int cornerCount, double qualityLevel, double minDistance, IntPtr mask, int blockSize, int useHarris, double k ) [4]
```

**Parameters**

- **image (IntPtr)**
  The source 8-bit or floating-point 32-bit, single-channel image

- **eigImage (IntPtr)**
  Temporary floating-point 32-bit image of the same size as image

- **tempImage (IntPtr)**
  Another temporary image of the same size and same format as eig_image

- **corners (IntPtr)**
  Output parameter. Detected corners

- **cornerCount (Int32)**
  Output parameter. Number of detected corners

- **qualityLevel (Double)**
Multiplier for the maxmin eigen value; specifies minimal accepted quality of image corners

- **minDistance (Double)**
  
  Limit, specifying minimum possible distance between returned corners; Euclidian distance is used

- **mask (IntPtr)**
  
  Region of interest. The function selects points either in the specified region or in the whole image if the mask is NULL

- **blockSize (Int32)**
  
  Size of the averaging block, passed to underlying cvCornerMinEigenVal or cvCornerHarris used by the function

- **useHarris (Int32)**
  
  If nonzero, Harris operator (cvCornerHarris) is used instead of default cvCornerMinEigenVal.

- **k (Double)**
  
  Free parameter of Harris detector; used only if useHarris != 0
In the Fig 5.5 the motion of the car is tracked. The points in yellow show the detected corners of the car and the arrows in red tell the flow fields.

5.6 Alternatives

5.6.1 Lucas Kanade Tracking

The LK tracker uses three assumptions, (a) brightness constancy between the same pixels from one frame to the next,(b)small movements between frames (requiring image pyramids to track larger movements), and (c) spatial coherence, points near each other are on the same surface. Then the basic concept of the tracker is to estimate the velocity of a moving pixel by the ratio of the derivative of the intensity over time divided by the derivative of the intensity over space. [5]

We tried both the algorithms and observed that CVGoodFeaturestoTrack gives more accurate tracking results as compare to Lucas Kanade, Therefore we opted CVGoodFeaturestoTrack.
5.6.2 Motion Detection using Aforge.Net

AForge.NET is a C# framework designed for developers and researchers in the field of Computer Vision and Artificial Intelligence (image processing, neural networks, genetic algorithms, machine learning, robotics, etc.)

Clustering wasn’t possible with the results obtained by Aforge.Net hence we opted for CVGoodFeaturestoTrack which gave more exact tracking results and clustering algorithm was also implementable to deduce traffic congestion on roads.
CHAPTER 6

Data Analysis

6.1 Data Analysis

Data Analysis is the second main module of the project. In this module the data gathered is processed and relevant information extracted to deduce the traffic congestion on roads.

6.1.1 Processing the Data Obtained From Traffic Video

The points that were tracked using “Good Features to Track” on vehicles were used to find out the velocity of vehicles, that is how far vehicles have moved from one captured frame of the video to the second consecutive captured frame, this will tell the speed of vehicles on the road.

The distance between tracked points will tell the distance between cars on the road that is; if cars are very close to each other compared to a threshold value then it will indicate high congestion.

The graph below in Fig6.1 shows the tracked points plotted for only one moving object (car) on the road. Here we can easily figure out the points and identify where each point lies on the graph.

We saved tracked points by the function of “Good Features to Track” and plotted the graphs as shown below.
The graph when plotted for multiple moving objects (that is many cars on the road) the points are so intermixed with each other that it is very difficult and nearly impossible to figure out where each point is.

![Velocity vs Angle graph for single moving object](image1)

**Fig 6.1** tracked points of a single moving object in video

![Velocity vs Angle graph for multiple moving objects](image2)

**Fig 6.2** When multiple moving objects in video
6.2 Noise Filtration of the Tracked Points by CVGoodFeaturesToTrack

The points that were tracked using “Good Features to Track” included points that were not required that is Noise which had to be removed to get those filtered points that will contribute in providing the essential information about the traffic congestion on the road.

These points included:

- The static points on trees and other non moving objects.
- The points on pedestrians crossing the road.
- The points indicating any kind of abnormality such as a car cannot turn over an angle of 90 degrees.
- The points with a random and abnormal change in speed which is practically not possible

All such points add noise and thus are to be removed by efficient filtering. For this purpose we modified our code for the tracked points and obtained the constructive points.

6.3 Clustering

6.3.1 What is Clustering?

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields, including machine learning, data mining, recognition and image analysis.[6]

6.3.2 Need Of Clustering In The Project:

“All the points located near each other belong to the same object”. That is; the points belonging to one car will be positioned close to each other. All the nearby points will form one cluster that is the cluster is actually indicating the car or any vehicle.
By finding the number of clusters of the tracked points, we are finding out the number of vehicles on the road. If the count is above a certain threshold value for the road it will indicate High traffic congestion, below the threshold values will indicate Low traffic congestion and if it falls in a certain range it will indicate Medium traffic congestion.

### 6.3.3 Extraction of Number of Clusters

We have used the Quality Threshold (QT) Algorithm in C#.NET to find out the number of clusters from the tracked points after filtration.

This algorithm requires a priori specification of the threshold distance within the cluster and the minimum number of elements in each cluster.[6] Now from each data point we find all its candidate data points. Candidate data points are those which are within the range of the threshold distance from the given data point. This way we find the candidate data points for all data point and choose the one with large number of candidate data points to form cluster. Now data points that belong to this cluster are removed and the same procedure is repeated with the reduced set of data points until no more clusters can be formed satisfying the minimum size criteria.

### 6.3.4 Algorithmic steps for QT clustering

1) Initialize the threshold distance allowed for clusters and the minimum cluster size.

2) Build a candidate cluster for each data point by including the closest point, the next closest, and so on, until the distance of the cluster surpasses the threshold.

3) Save the candidate cluster with the most points as the first true cluster, and remove all points in the cluster from further consideration.

4) Repeat with the reduced set of points until no more cluster can be formed having the minimum cluster size.
Fig 6.3 Result of QT Clustering with threshold distance =10 & minimum number of element required within a cluster = 5

6.3.5 Advantages of QT Algorithm

1) Quality Guaranteed - Only clusters that pass a user-defined quality threshold will be returned.

2) Number of clusters is not specified a priori.

3) All possible clusters are considered - Candidate cluster is generated with respect to every data points and tested in order of size against quality criteria. [6]

6.4 Why Not K-Means Clustering?

In statistics and data mining, $k$-means clustering is a method of cluster analysis which aims to partition $n$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters).
In our project we tracked the points and searched for an algorithm which can tell us the no of clusters that could be formed from the tracked points. The number of clusters correspond to the number of cars on road at that particular instant and hence the traffic congestion on road. Therefore, k-means which required us to specify the number of clusters as input did not suit our requirements.

6.5 Database:

Our database consist of two tables

- Location Table
- User Login Table

6.5.1 Design of Location Table:

![Location Table](image1)

Fig 6.4 Location Table

6.5.2 Design of User Login Table:

![User Login Table](image2)

Fig 6.5 User Login Table
CHAPTER 7

TraffixGem [Website]

7.1 Website Overview

The basic objective of this website is to improve the current traffic system and to provide users current traffic updates on click.

- We cover Karachi Traffic Situation 24/7 round the clock.

![Fig 7.1 Home Page of Website](image)

- Traffic Updates powered by Google Maps.
- We are using Google Map Dynamic Link Library (DLL) in order to embed Google maps in our website.
7.2 Google Maps

The Dynamic Link Library (DLL) we have used is from googlemaps.subgrim.net. The user is required to select the location on the website of which he needs to know the current traffic situation. The location is then shown on the maps.

We have implemented map, hybrid and satellite views of Google map in the project. Once a location is selected the traffic congestion data is extracted from the database which keeps a record of the traffic situation of all the entered locations, a pop up then appears telling whether the congestion is high, medium or low at present in the entered location.

![Fig 7.2 Traffic Status shown on Google Maps](image-url)
7.3 Short Message Service (SMS)

Short Message Service (SMS) is a text messaging service component of phone, web, or mobile communication systems, using standardized communication protocols that allow the exchange of short text messages between fixed line or mobile phone devices. SMS text messaging is the most widely used data application in the world. SMS as used on modern handsets was originated from radio telegraphy in radio memo pagers using standardized phone protocols and later defined as part of the Global System for Mobile Communications (GSM) series of standards in 1985 as a means of sending messages of up to 160 character. [7]

![Fig 7.3 SMS Service](image)

7.4 Registration Form for SMS:

We have a registration form so that only registered users get Traffic Updates from our website on their mobile phones.
We have used free Online API (for C#.net) to embed GSM service to send traffic updates from our website to registered users. [10]
CHAPTER 8

Future Enhancements

8.1 Alternative routes

An enhancement to the project is the capability to identify the alternative routes of low traffic congestion in case of high traffic congestion on the location entered by the user.

8.2 Customized SMS

The SMS feature can be further customized to include the company name instead of the number. Two way SMS feature can also be added. All registered users get notified if there is a severe traffic jam in any part of the city.

8.3 Filtering noise by means of Kalman or Particle Filter

The techniques of Kalman Filter and Particle Filter can be used to filter out the noise from the tracked points to give more efficient results, hence the clustering when performed to deduce traffic congestion will also be more accurate.
Coding Of Motion Tracking Unit

// Variable Initialization

string vdo;
int index; public string loc;
Timer timer = new Timer();
public int m = 0;
public location locinmain;
IplImage image;
Image<Bgr, byte> image1;
Capture capture;
StreamWriter sw = new StreamWriter("c:\file.csv");
double xx1=0, yy1=0;
double degree, hyp;
public double avg=0, sum=0;
report frm = new report();
ArrayList arr;
SolidBrush mybrush;
public string cls;
private CvPoint2D32f[][] point;

// Function Call For Form Loading

private void Form1_Load(object sender, EventArgs e)
{

this.WindowState = FormWindowState.Maximized;
vdo = ((location)this.locinmain).vdo;
index = ((location)this.locinmain).index;
loc = ((location)this.locinmain).loc;
if (index == 0)
capture = new Capture();
else if (index == 1)
{
capture = new Capture(vdo);
}
timer.Tick += new EventHandler(timer1_Tick); // Everytime timer ticks, timer1_Tick will be called
timer.Interval = 100; // Timer will tick event second
timer.Enabled = true; // Enable the timer
timer.Start();
}

// Structure for QT Clustering(Clustering Code [8])

public partial class main : Form
{
struct Point
{
public int X, Y;
}
public Point(int x, int y)
{
    this.X = x;
    this.Y = y;
}
public override string ToString()
{
    return String.Format("(\{0\}, \{1\})", X, Y);
}

public static int DistanceSquared(Point p1, Point p2)
{
    int diffX = p2.X - p1.X;
    int diffY = p2.Y - p1.Y;
    return diffX * diffX + diffY * diffY;
}

// Method to get clusters from the points store in List(Clustering Code [8])
static List<List<Point>> GetClusters(List<Point> points, double maxDiameter)
{
    if (points == null) return null;
    points = new List<Point>(points); // leave original List unaltered
    List<List<Point>> clusters = new List<List<Point>>() ;
    while (points.Count > 0)
    {
        List<Point> bestCandidate = GetBestCandidate(points, maxDiameter);
        clusters.Add(bestCandidate);
    }
    return clusters;
}

// GetBestCandidate() returns first candidate cluster encountered if there is more than one with the maximum number of points. (Clustering Code [8])
static List<Point> GetBestCandidate(List<Point> points, double maxDiameter)
{
    double maxDiameterSquared = maxDiameter * maxDiameter; // square maximum diameter
    List<List<Point>> candidates = new List<List<Point>>() ; // stores all candidate clusters
    List<Point> currentCandidate = null; // stores current candidate cluster
    int[] candidateNumbers = new int[points.Count]; // keeps track of candidate numbers to which points have been allocated
    int totalPointsAllocated = 0; // total number of points already allocated to candidates
    int currentCandidateNumber = 0; // current candidate number
    for(int i = 0; i < points.Count; i++)
    {
        if (totalPointsAllocated == points.Count) break; // no need to continue further
        if (candidateNumbers[i] > 0) continue; // point already allocated to a candidate
        currentCandidateNumber++;
    }
currentCandidate = new List<Point>(); // create a new candidate cluster
currentCandidate.Add(points[i]); // add the current point to it
candidateNumbers[i] = currentCandidateNumber;
totalPointsAllocated++;

Point latestPoint = points[i]; // latest point added to current cluster
int[] diametersSquared = new int[points.Count]; // diameters squared of each point when added to current cluster

// iterate through any remaining points
    // successively selecting the point closest to the group until the threshold is exceeded
while (true)
{
    if (totalPointsAllocated == points.Count) break; // no need to continue further
    int closest = -1; // index of closest point to current candidate cluster
    int minDiameterSquared = Int32.MaxValue; // minimum diameter squared, initialized to impossible value
    for (int j = i + 1; j < points.Count; j++)
    {
        if (candidateNumbers[j] > 0) continue; // point already allocated to a candidate
        // update diameters squared to allow for latest point added to current cluster
        int distSquared = Point.DistanceSquared(latestPoint, points[j]);
        if (distSquared > diametersSquared[j]) diametersSquared[j] = distSquared;
        // check if closer than previous closest point
        if (diametersSquared[j] < minDiameterSquared)
        {
            minDiameterSquared = diametersSquared[j];
            closest = j;
        }
    }
    // if closest point is within maxDiameter, add it to the current candidate and mark it accordingly
    if ((double)minDiameterSquared <= maxDiameterSquared)
    {
        currentCandidate.Add(points[closest]);
        candidateNumbers[closest] = currentCandidateNumber;
        totalPointsAllocated++;
    }
    else // otherwise finished with current candidate
    {
    break;
    }
}

// add current candidate to candidates list
candidates.Add(currentCandidate);

// now find the candidate cluster with the largest number of points
int maxPoints = -1; // impossibly small value
int bestCandidateNumber = 0; // ditto
for(int i = 0; i < candidates.Count; i++)
{
    if (candidates[i].Count > maxPoints)
    {
        maxPoints = candidates[i].Count;
        bestCandidateNumber = i;
    }
}
bestCandidateNumber = i + 1; // counting from 1 rather than 0
}
}

// iterating backwards to avoid indexing problems, remove points in best candidate from points list
  // this will automatically be persisted to caller as List<Point> is a reference type

for(int i = candidateNumbers.Length - 1; i >= 0; i--)
{
  if (candidateNumbers[i] == bestCandidateNumber) points.RemoveAt(i);
}

// return best candidate to caller
return candidates[bestCandidateNumber - 1];

// Timer Event

private void timer1_Tick(object sender, EventArgs e)
{

m++;

image1 = capture.QueryFrame();

pictureBox1.Image = image1.ToBitmap();
List<Point> points = new List<Point>();
IntPtr kj = (IntPtr)image1;
IntPtr image = (IplImage)Marshal.PtrToStructure(kj, typeof(IplImage));

int index = 1;
int corner_count = 100;

// create gray scale image
IplImage gray = cvlib.CvCreateImage(new CvSize(image.width, image.height),
(int)cvlib.IPL_DEPTH_8U, 1);
IplImage eig_image = cvlib.CvCreateImage(new CvSize(image.width, image.height),
(int)cvlib.IPL_DEPTH_32F, 1);
IplImage tmp_image = cvlib.CvCreateImage(new CvSize(image.width, image.height),
(int)cvlib.IPL_DEPTH_32F, 1);

CvPoint2D32f[] pts = new CvPoint2D32f[corner_count];
CvPoint2D32f[] pts_new = new CvPoint2D32f[200];

// Do color conversion

cvlib.CvCvtColor(ref image, ref gray, cvlib.CV_BGR2GRAY);

GCHandle h;
// Calling “CVGOODFEATURESTOTRACK”

cvlib.CvGoodFeaturesToTrack(ref gray, ref eig_image, ref tmp_image, cvtools.Convert1DArrToPtr(pts, out h), ref corner_count, 0.01, 1, IntPtr.Zero, 3, 1, 0.04);

foreach (CvPoint2D32f p in pts)
{//create circles on the features to track and has arguments of void
 cvlib.CvCircle(ref image, new CvPoint((int)p.x, (int)p.y), 2, new CvScalar(0, 255, 255, 0), 2, 8, 0);

pts_new[index]=pts[index-1];
index++;
}

for(int i = 0; i<corner_count; i++)
{

// Stanford Code begins [9]

int line_thickness;
line_thickness = 1;
CvScalar line_color;  line_color = cvlib.CV_RGB(255,0,0);
/* Let's make the flow field look nice with arrows. */
//The arrows will be a bit too short for a nice visualization because of the
//high framerate
//   * (ie: there's not much motion between the frames). So let's lengthen them
//by a factor of 3.

CvPoint p,q;
p.x = (int)pts[i].x;
p.y = (int) pts[i].y;
   // difference between the old and new tracked points
q.x = (int)pts_new[i+1+1].x;
q.y = (int)pts_new[i+1+1].y;
x1 = (double)p.x - q.x;
yy1 = (double)p.y - q.y;
points.Add(new Point((int)xx1, (int)yy1));
if (xx1 <= 0)
    continue;
if (yy1 <= 0)
    continue;

double angle;
double hypotenuse;
angle = Math.Atan2(yy1, xx1);//angel in radian
//shortening the arrows by a factor of 1000

degree =Math.Round( angle * (180 / Math.PI)); //angle in degree
if (degree == 0 || degree>60 || degree <0)
    continue;
hyp = Math.Round(Math.Sqrt((Math.Pow(yy1, 2) + Math.Pow(xx1, 2))));
if (hyp <= 0 || hyp > 100)
    continue;
arr.Add(hyp);
string file = degree.ToString() + "," + hyp.ToString();
sw.WriteLine(file);
hypotenuse = Math.Sqrt((Math.Pow(yy1, 2) + Math.Pow(xx1, 2)))/1000;

/* Here we lengthen the arrow by a factor of three. */
q.x = (int) (p.x - 3 * hypotenuse * Math.Cos(angle));
q.y = (int) (p.y - 3 * hypotenuse * Math.Sin(angle));
//Now we draw the main line of the arrow.
//"frame1" is the frame to draw on.
// * "p" is the point where the line begins.
// * "q" is the point where the line stops.
// * "CV_AA" means antialiased drawing.
    // /* "0" means no fractional bits in the center coordinate or radius.
    cvlib.CvLine( ref image, p, q, line_color, line_thickness, cvlib.CV_AA, 0 );
    //Now draw the tips of the arrow. I do some scaling so that the
    // /* tips look proportional to the main line of the arrow.
    // /*
    float pi = Convert.ToSingle(3.142);
    p.x = (int) (q.x + 9 * Math.Cos(angle + pi / 4));
    p.y = (int) (q.y + 9 * Math.Sin(angle + pi / 4));
    cvlib.CvLine( ref image, p, q, line_color, line_thickness, cvlib.CV_AA, 0 );
    p.x = (int) (q.x + 9 * Math.Cos(angle - pi / 4));
    p.y = (int) (q.y + 9 * Math.Sin(angle - pi / 4));
    cvlib.CvLine( ref image, p, q, line_color, line_thickness, cvlib.CV_AA, 0 );
}

// Stanford Code ends
// Now display the image we drew on.

cvlib.CvReleaseImage(ref eig_image);
cvlib.CvReleaseImage(ref tmp_image);
cvlib.CvReleaseImage(ref gray);
cvtools.ReleaseHandel(h);
pictureBox2.Image = (Bitmap)image;

if (timer.Interval * m == 6000)
{
    timer.Stop();
    sw.Close();
    for (int i = 0; i < m; i++)
    {
        sum += (double)arr[i];
    }
    double maxDiameter = 25.0;
    avg = Math.Round( sum / m);
List<List<Point>> clusters = GetClusters(points, maxDiameter);
Object clts = clusters.Count;
cls = clts.ToString();
frm frm1inForm2 = this;
frm.Show();

Showing Report of the Traffic Congestion

public partial class report : Form
{
    public main frm1inForm2;
    string location;
    double avg_vel=0,frames=0;
    string insert = null;
    int vel = 0;
    string loc_name =null;
    string cls = null;

    //Calling Constructor
    public report()
    {
        InitializeComponent();
    }

    private void Form2_Load(object sender, EventArgs e)
    {
        this.WindowState = FormWindowState.Maximized;
        location = ((main)this.frm1inForm2).loc;
        maxcongctxt.Text = "25 to 46";
        medcongctxt.Text = "47 to 65";
        maxnocongctxt.Text = "66 to 100";
        avg_vel = ((main)this.frm1inForm2).avg;
        frames = ((main)this.frm1inForm2).m;
        cls = ((main)this.frm1inForm2).cls;
        textBox1.Text = cls;
        avgtext.Text = avg_vel.ToString();
        maxmtext.Text = frames.ToString();
        loctxtbox.Text = location;
        vel=Convert.ToInt32(avg_vel);

        if (avg_vel>=66 & & avg_vel<=100 )
            congestion.Text = "No congestion";
        else if (avg_vel >= 47 & & avg_vel <= 65)
            congestion.Text = "congestion is medium";
        else if (avg_vel >= 25 & & avg_vel <= 46)
            congestion.Text = "congestion is high"; 
    }
// Opens Connection of SQL Database to store results from Motion Tracking

SqlConnection cn = new SqlConnection("Data Source=HOME-6875CA74ED;Initial Catalog=TMS1;Integrated Security=True;MultipleActiveResultSets=True");
try
{
    cn.Open();
    loc_name = "select locName from Location where locName=" + location.ToUpper() + "";
    SqlCommand cmd1 = new SqlCommand(loc_name, cn);
    SqlDataReader dr1 = cmd1.ExecuteReader();

    if (dr1.Read())
    {
        insert = "update Location set trafficStatus=" + congtext.Text.ToUpper() + ",avgVelocity=" + vel + ",clusterCount=" + cls + " where locName=" + location.ToUpper() + "";
    }
    else
    {
        insert = "insert into Location (locName,trafficStatus,avgVelocity,clusterCount) values(" + location.ToUpper() + "," + congtext.Text.ToUpper() + "," + vel + "," + cls + ");"
    }

    SqlCommand cmd2 = new SqlCommand(insert, cn);
    SqlCommand cmd = new SqlCommand("insert_loc", cn); //inserting data into Database
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@locName", SqlDbType.VarChar).Value = location.ToUpper();
    cmd.Parameters.Add("@trafficStatus", SqlDbType.VarChar).Value = congtext.Text.ToUpper();
    cmd.Parameters.Add("@avgVelocity", SqlDbType.Int).Value = vel;
    cmd.Parameters.Add("@clusterCount", SqlDbType.Int).Value = cls;
    cmd2.ExecuteNonQuery();
}
catch (Exception exp)
{
    MessageBox.Show(exp.Message);
}
finally
{
    cn.Close(); // Closing Connection
}
APPENDIX-B

Code to embed Google Maps on TMS Website

//Initializing Variables

string location = null, status = null, velocity = null;
string sStreetAddress = null;
string info = null, image_path = null;

//Retrieving Data From Database

protected void Button1_Click(object sender, EventArgs e)
{
SqlConnection cn = new SqlConnection("Data Source=HOME-6875CA74ED;Initial Catalog=TMS1;Integrated Security=True");
try {
    cn.Open();
    select = "select locName,trafficStatus,avgVelocity from Location where locName=" + locationtext.Text.ToUpper() + "";
    SqlCommand cmd = new SqlCommand(select, cn);
    SqlDataReader dr = cmd.ExecuteReader();
    dr.Read();
    if (dr.HasRows == true)
    {
        Session["loc"] = locationtext.Text;
        Session["vel"] = Convert.ToInt32(dr["avgVelocity"].ToString());
        Session["status"] = dr["trafficStatus"].ToString();
    } else
    
    Response.Write("No data Avalible...");
    Response.Redirect("gmap.aspx", false);
}
catch (Exception exp)
{
    Session["error"] = exp.Message.ToString();
    Response.Redirect("Error.aspx", false);
} finally
{
    cn.Close();
}
// Page load event

protected void Page_Load(object sender, EventArgs e)
{
    location = Session["loc"].ToString();
    status = Session["status"].ToString();
    velocity = Session["vel"].ToString();
sStreetAddress = location + " Karachi. Pakistan";
if (status.ToLower() == "congestion is high")
{
    image_path = "img/3.png";
}
else if (status.ToLower() == "congestion is medium")
{
    image_path = "img/1.png";
}
else if (status.ToLower() == "no congestion")
{
    image_path = "images/bg.gif";
}
else
{
    // ic.markerIconOptions = new Subgurim.Controles.MarkerIconOptions(50, 50, Color.Blue);
    image_path = "img/1.png";
    location = "Invalid Location";
    status = "Invalid Traffic Status";
    velocity = "Invalid Speed";
}
GeoCode = GMap1.geoCodeRequest(sStreetAddress);
Subgurim.Controles.GLatLng gLatLng = new Subgurim.Controles.GLatLng(GeoCode.Placemark.coordinates.lat,
GeoCode.Placemark.coordinates.lng);
GMap1.setCenter(gLatLng, 16, Subgurim.Controles.GMapType.GTypes.Normal);
GMap1.enableRotation = true;
Subgurim.Controles.GControl(Subgurim.Controles.GControl.preBuilt.LargeMapControl));
Subgurim.Controles.MarkerManager mManager = new Subgurim.Controles.MarkerManager();
Subgurim.Controles.GIcon ic = new Subgurim.Controles.GIcon(image_path);
Subgurim.Controles.GMarker oMarker = new Subgurim.Controles.GMarker(gLatLng, ic);
mManager.Add(oMarker, 2);
List<Subgurim.Controles.GMarker> mks = new List<Subgurim.Controles.GMarker>();
Random r = new Random();
double ir1, ir2;
Subgurim.Controles.GMarker mkr;
for (int i = 0; i < 10; i++)
{
    ir1 = (double)r.Next(40) / 10.0 - 2.0;
    ir2 = (double)r.Next(40) / 10.0 - 2.0;

    mkr = new Subgurim.Controles.GMarker(oMarker.point + new Subgurim.Controles.GLatLng(ir1, ir2));
    mks.Add(mkr);
    }
    mManager.Add(mks, 6, 8);
    GMap1.markerManager = mManager;
    GMap1.addGMarker(oMarker);
    GMap1.addGMarker(oMarker);
}

**Emailing Class**

public void mail1(string body, string to)
{
    //sending an email

    SmtpClient client = new SmtpClient();
    client.DeliveryMethod = SmtpDeliveryMethod.Network;
    client.EnableSsl = true;
    client.Host = "smtp.gmail.com"; // using gmail server as the host
    client.Port = 587;

    // For Hotmail
    // client.Host = "smtp.live.com";
    // client.Port = 25;

    System.Net.NetworkCredential credentials =
    new System.Net.NetworkCredential("email", "password");
    client.UseDefaultCredentials = true;
    client.Credentials = credentials;

    string str_to = to;
    MailMessage msg = new MailMessage();
    msg.From = new MailAddress("email address ");
    msg.To.Add(new MailAddress(str_to)); // can be hotmail, yahoo or any other
    msg.Subject = "TMS Admin";
    msg.IsBodyHtml = true;
    msg.Body = body;
    client.Send(msg);
    string sent = "An mail has been sent....";
    //return sent;
}
REFERENCES

[1] Gary Bradski, and Adrian Kaebler, *LEARNING OPNECV (OREILLY)*.


[4] http://www.emgu.com/wiki/files/1.5.0.0/Help/html/f491f397-11a4-4999-20f4-0cfa329a5d89.htm


