YILDIZ TECHNICAL UNIVERSITY

ELECTRIC-ELECTRONIC FACULTY

COMPUTER ENGINEERING DEPARTMENT

SENIOR PROJECT

TRAFFIC FLOW SPEED DETECTION

Project Supervisor : Prof. Dr. A.Coşkun SÖNMEZ

Project Group

02011055 Zehra GÜRBÜZ

İstanbul, 2006

INDEX **Page**

1.INTRODUCTION

Video Object Tracking plays a very important role in many vision applications. Apart from applications traditionally for video surveillance, object recognition, and video segmentation and indexing, real-time object tracking is now extensively used in audio-visual speech recognition (Liu 2002), human gesture recognition, and object based video compressions, such as MPEG-4. When priori knowledge about shape or motion of the object is known, a parameterised model of the object can be used. This type of tracking is also known as model-based tracking. It is frequently used to track balls in sports, cars in traffic (Koller 1992) and hands in human body (Ouhaddi 1999).

On the other hand car detection in aerial images has important civilian and military uses, such as traffic surveillance, both for traffic information system or to gather traffic statistics for urban planning. It can also produce strong evidence for road detection. It also provides a good test domain for methods of object detection in difficult situation that require integration of multiple cues.

Our project aims to implement a system that is capable to detect and track cars in video images.Traffic is a big problem in big cities and requires to be controlled for public benefit. The aim of this project is detecting velocity of traffic flow and developing a system avaliable for sending its result to another application simultaneously.

2.SYSTEM ANALYSIS

System Analysis contains avaliable hardware and software system in the firm, its problems and our suggestions for solution.

2.1 Available System in the Firm

Traffic information system is being used by Traffic Control Unit in Metropolitan Municipality of İstanbul. There are nearly 60 traffic cameras at different locations in İstanbul, sending their data via DVS (Digital Video Surveillance System) to Traffic Contol Center. Other system used in Traffic Control Unit is the RTMS (Remote Traffic Microwave Sensor) radar, a traffic sensor which detects presence and measures traffic parameters in multiple independent lines.

2.1.1 DVS (Digital Video Surveillance System)

Digital Video Surveillance System is a digital tracking system that uses MPEG-4 coding technique. It is easy to use and scalable because of its digital substructure characteristic. On the other hand it is useful for storing data at a remote center and transmitting data to Internet. It has a lot of technical superiorities, such as UPS support with remote control to the cameras, properity to change settings of cameras remotely. We will use this substructure to get video data. The locations of traffic cameras in İstanbul are shown below.

Figure 2.1 A traffic camera [1]

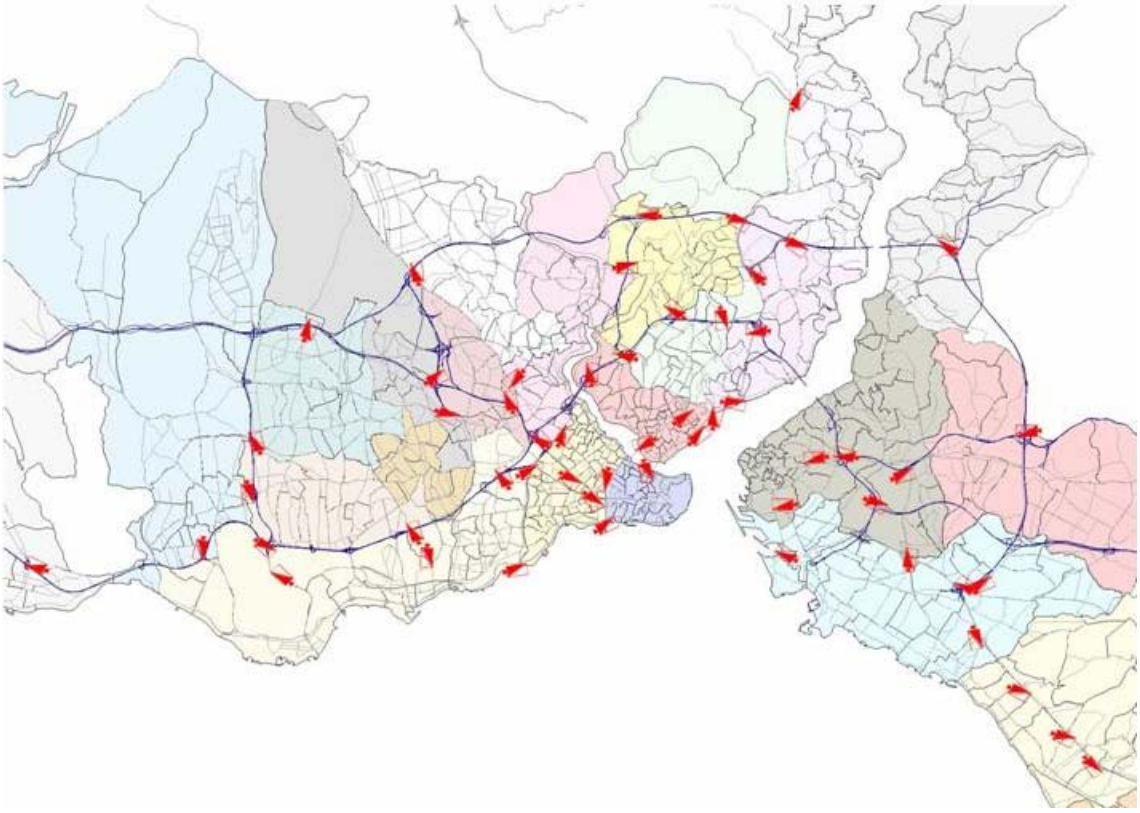

Figure 2.2 Traffic Cameras in İstanbul [1]

2.1.2 RTMS (Remote Traffic Microwave Sensor)

The RTMS is a true presence traffic detector providing presence, volume, occupancy, speed and classification information in up to 8 discrete user-defined detection zones up to 60 m away. Output information is provided to existing controllers via contact pairs and to computer systems via a RS-232 serial communications port. RTMS receives reflected signals from all objects, stationary or moving i.e. pavement, barriers, vehicles and trees. To distinguish between them, RTMS signal processing maintains and updates a background level of signals for each of the 32 range slices. Presence of a vehicle in a detection zone is determined only if the reflected signal strength in its range slice exceeds the background level by a threshold margin.

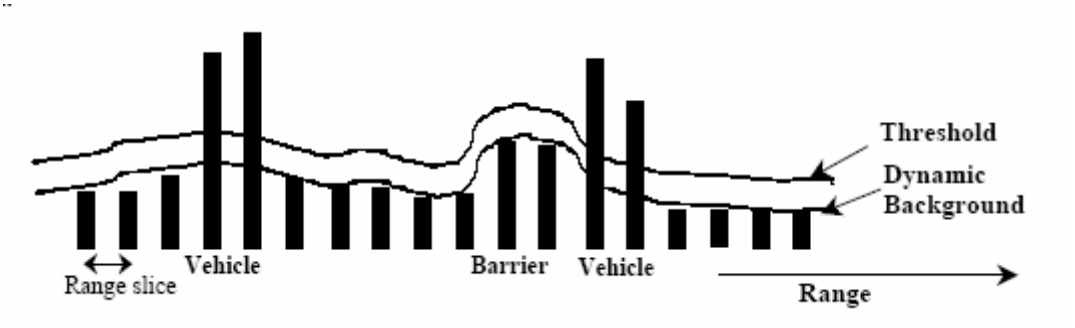

Figure 2.3 Reflected Signals and detection

The RTMS is a versatile sensor, capable of detecting vehicles' presence parameters in multiple zones, in support of various ITS applications. RTMS equipments are being used by Traffic Control Center to get data of traffic density, band based flow velocity.

2.2 Current System Problems

Current system has some problems about RTMS devices. RTMS equipment is very expensive and frequently requires care. You can not be sure whether the data sent is true or not. Its fault rate is %80 [2]. Our aim with this project is processing data come from via VMS, detecting traffic flow and measuring its velocity; so that we will be able to detect RTMS devices' straightness.

2.3 New System Characteristics

Our project will be able to process several video images. This means there isn't unique road characteristic or camera position. Our system is formed with 3 software parts:

- vehicle detection,
- tracking,
- velocity measurement

To implement these requirements some software development tools and hardware devices will be used in the project. The main difficulties lie in the following:

- Although the viewpoint is constrained, there are still variations that make the cars have different appearance.
- The image resolution is low so not many details are visible.
- Cars can be of any intensity in the image, from very dark to very light. Also, some cars' intensity is very close to the road.
- The shadow cast on the ground by sunlight is more salient in an aerial view than in a ground view, which complicates the detection.
- The image quality varies. The brightness, contrast and sharpness of the images change due to factors including illumination, focusing and atmospheric turbulence.
- The expected features of a car differ with its intensity and the existence of shadow. For a simple example, whether or not the boundary of a gray car can be detected depends heavily on its shadow.

We need to account for all these difficulties to get a reasonable good system. There are some frames captured from video images shown below:

Figure 2.4 Images From Different Camera Videos

3. FEASIBILITY STUDY

Feasibility research of project will be structured in separated technical and economical feasibility.

3.1 Technical Feasibility

We will talk about software and hardware feasibility in this section..

3.1.1 Software Feasibility

The development tool for this project is:

- Microsoft Visual C++ 6.0

The software tool is:

- OPENCV

Operating system of the computer will be Windows based. The operating system requirements of our system is:

- Windows XP Professional Edition Sp2

Software subject is very important to evaluate hardware system efficently. Software tool comes more important in image processing, since there are lots of matrix operations and calculations; besides the speed and memory allocations are vital for real time operations. We decided to use C, and its object-oriented successor C++. They are used to write a huge variety of applications and almost all operating systems. There are C/C++ compilers for all major operating systems and hardware platforms. C and C++ are written as a series of functions that call each other for processing. Functions are very flexible, allowing programmers to choose from the standard library that comes with the compiler, to use third party libraries or to develop their own. With this language we will use OPENCV (Open Source Computer Vision Library) for developing tracking part of our project because of its successful classes and functions in this field. Details are given below:

3.1.1.1 OPENCV (Open Source Computer Vision Library)

OpenCV is a library of programming functions mainly aimed at real time computer vision. General description of OPENCV:

- Open source computer vision library in C/C++.
- Optimized and intended for real-time applications.
- OS/hardware/window-manager independent.
- Generic image/video loading, saving, and acquisition.
- Both low and high level API.

Example applications of the OpenCV library are Human-Computer Interaction (HCI); Segmentation and Recognition; Face Recognition; Motion Tracking, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics. Library Areas:

Table 3.1 Library Areas

Image functions	Creation, allocation, destruction of images. Fast pixel access macros.
Data Structures	Static types and dynamic storage.
Geometry	Line and ellipse fitting. Convex hull. Contour analysis.
Morphology	Erode, dilate, open, close. Gradient, top-hat, black-hat.
Background Differencing	Accumulate images and squared images. Running averages.
Distance Transform	Distance Transform
Camera Calibration	Intrinsic and extrinsic, Rodrigues, un-distortion, Finding checkerboard calibration pattern
View Morphing	8 point algorithm, Epipolar alignment of images
Motion Templates	Overlaying silhouettes: motion history image, gradient and weighted global motion.
Optical Flow	HS, L-K, BM and L-K in pyramid.
Drawing Primatives	Line, rectangle, circle, ellipse, polygon. Text on images.
System Functions	Load optimized code. Get processor info.

OpenCV provides transparent interface to Intel® Integrated Performance Primitives (IPP). That is, it loads automatically IPP libraries optimized for specific processor at runtime, if they are available. [3]

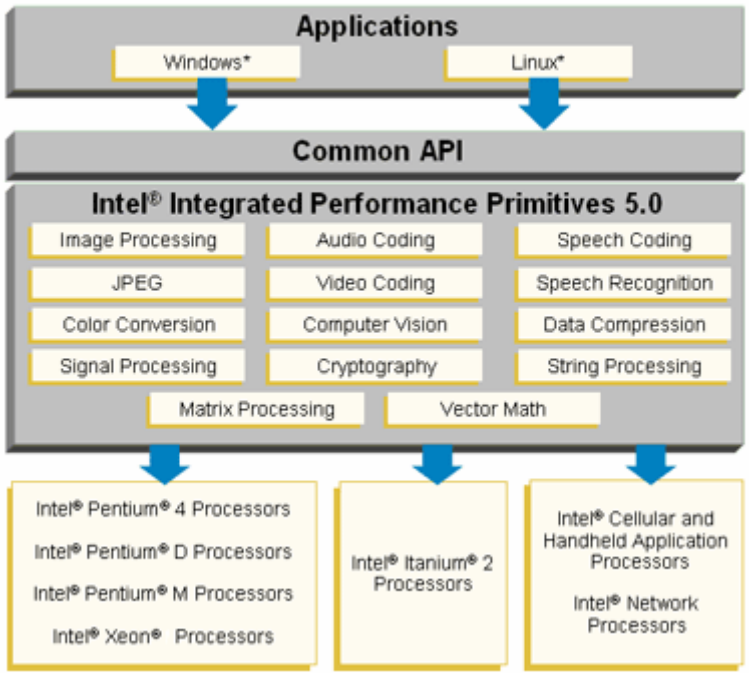

Figure 3.1 Intel IPP Architecture

Intel IPP helps you cut through performance bottlenecks easily in your media and communications, applications, with a rich feature set:

Outstanding Performance

- Advanced tuning techniques for software and hardware make it easy to get maximum performance out of your applications.
- Media applications get the performance headroom to add features and deliver more high-quality video and audio streams to more users.
- Communications applications get the performance headroom to provide more differentiated services to clients.
- Entertainment applications deliver more vivid, interactive user experiences.

New Codec Development Framework

- A new set of Unified Media C++ Classes included in the Intel IPP 5.0 code samples makes it easy for you to create powerful media applications.
- Get to market sooner with your media applications by re-using software designs for multiple media codecs.
- Create versatile multi-format media applications that easily handle and translate MPEG-2, MPEG-4, H.264 video formats, and multiple speech/audio formats. [3]

3.1.2 Hardware Feasibility

Hardware Requirements:

Processor

Intel Pentium processor with MMX technology, 300 MHz performance level

Memory Requirement

256 MB

Free Disk Space

700 MB

Monitor

VGA adapted or high resolution

Camera Equipment

Pelco CC1400HZ16-2 Series Digital CCD Color Camera

Camera Features:

• CompactBodyStyle

• 1/4-Inch Format CCD Imager

• High Resolution

• Built-in 16X Optical Zoom Lens with 8X Digital Zoom

• Horizontal Resolution of 470 TV Lines

• Digital Signal Processing

• AC Line Lock (24 VAC only)

• Auto White Balance

• Auto Iris, Auto Focus

• Controller Interface Available

3.1 Economical Feasibility

According to the project plan the construction of the program will take 3 monthsAll these steps are shown below:

	Traffic Flow Speed Detection System	**March**	**April**	**May**	**June**
SYSTEM ANALYSIS	Start				
	Communicating With Firm				
	Requirement Anlaysis				
	Model Composing				
	Interaction with Firm				
APPLICATION DEVELOPMENT	Program Design				
	Programming				
TEST and SETUP	Test				
	Setup				

Figure 3.2 Gannt Diagram

Table 3.2 Human Resources

Development Steps	Time	Cost of 1 staff / day (s/d)	Total
System Analysis	40 days	20$	800$
Coding	40 days	20$	800$
Education	5 days	80$	400$
			2000$

Costs of all requirements are:

Table 3.3 Total cost of the process

	Cost
Main Devices (PC)	700$
Peripherals (camera equipment) [4]	500$
Development Tools [5]	700$
Operating System	300$
Development – Education (content of Table 3.2)	2000$
Total	4200$

4. SYSTEM DESIGN

In this project, the Lucas Kanade method for image registration was implemented to track simple features in traffic avi files. The flow chart of the application is below:

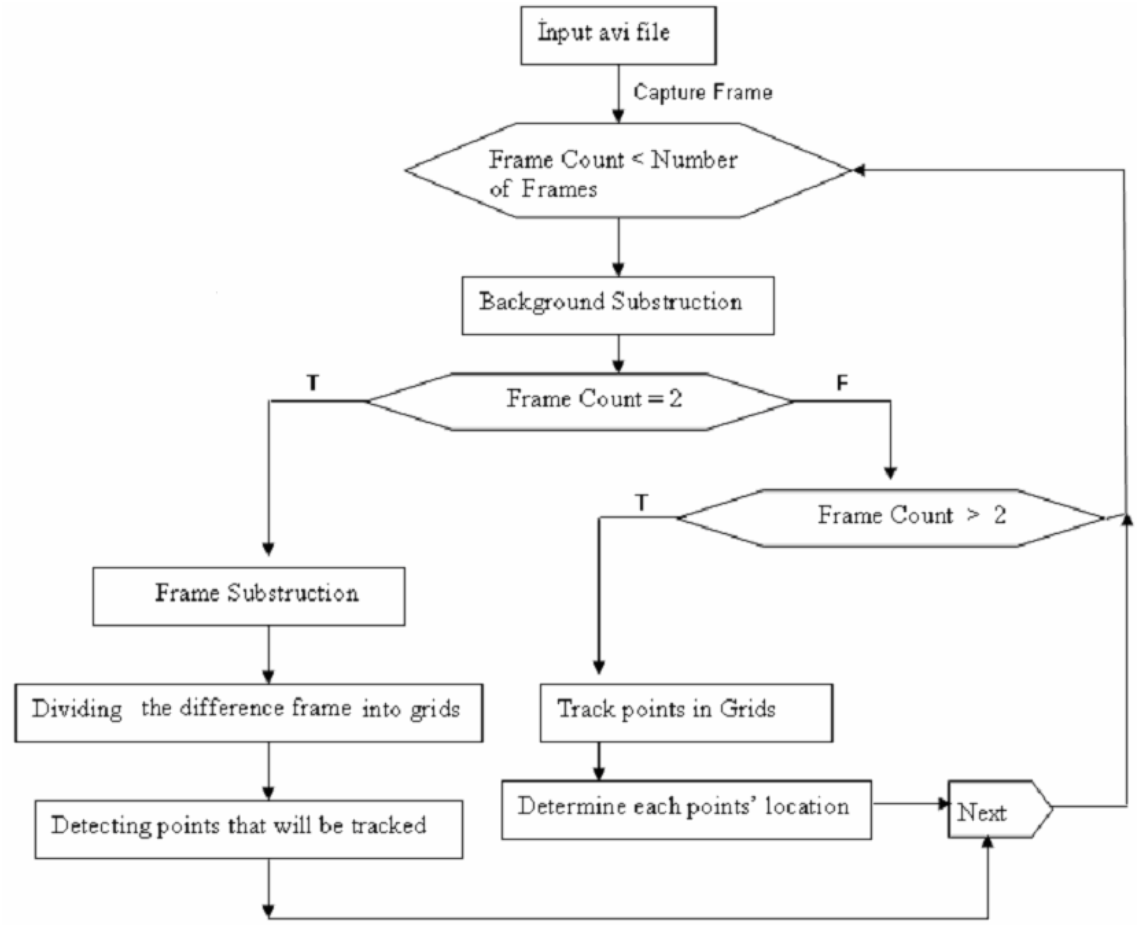

Figure 4.1

4.1 Background Subtraction

This section describes basic functions that enable building statistical model of background for its further subtraction. In this chapter the term "background" stands for a set of motionless image pixels, that is, pixels that do not belong to any object, moving in front of the camera. This definition can vary if considered in other techniques of object extraction. For example, if a depth map of the scene is obtained, background can be determined as parts of scene that are located far enough from the camera. The simplest

background model assumes that every background pixel brightness varies independently, according to normal distribution.The background characteristics can be calculated by accumulating several dozens of frames, as well as their squares. That means finding a sum of pixel values in the location $S(x,y)$ and a sum of squares of the values $Sq(x,y)$ for every pixel location.

Then mean is calculated as :

$$\bar{m}_{(x,y)} = \frac{S_{(x,y)}}{N},$$

where N is the number of the frames collected, and standard deviation as

$$\sigma_{(x,y)} = sqrt\left(\frac{Sq_{(x,y)}}{N} - \left(\frac{S_{(x,y)}}{N}\right)^2\right)$$

After that the pixel in a certain pixel location in certain frame is regarded as belonging

to a moving object if condition

$$abs(m_{(x,y)} - p_{(x,y)}) > C\sigma_{(x,y)}$$

is met, where C is a certain constant. If C is equal to 3, it is the well-known "three sigmas" rule. To obtain that background model, any objects should be put away from the camera for a few seconds, so that a whole image from the camera represents subsequent background observation.The above technique can be improved. First, it is reasonable to provide adaptation of background differencing model to changes of lighting conditions and background scenes, e.g., when the camera moves or some object is passing behind the front object.

4.2 Object Tracking

In object tracking problem there are several functions for calculating optical flow between two images. Most papers devoted to motion estimation use the term optical flow. Optical flow is defined as an apparent motion of image brightness. Let $I(x,y,t)$ be the image brightness that changes in time to provide an image sequence. Two main assumptions can be made:

1. Brightness $I(x,y,t)$ smoothly depends on coordinates x, y in greater part of the image.

2. Brightness of every point of a moving or static object does not change in time.

Let some object in the image, or some point of an object, move and after time dt the object displacement is $(dx,\ dy)$. Using Taylor series for brightness $I(x,y,t)$ gives the following:

$$I(x+dx, y+dy, t+dt) = I(x, y, t)+\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial t}dt+\dots,$$

where "…" are higher order terms. Then, according to Assumption 2:

$$I(x+dx, y+dy, t+dt) = I(x, y, t),$$

and

$$\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial t}dt+\dots = 0.$$

Dividing (18.3) by dt and defining

$$\frac{dx}{dt} = u,\ \frac{dy}{dt} = v$$

gives an equation

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v,$$

usually called *optical flow constraint equation*, where u and v are components of optical flow field in x and y coordinates respectively. Some variants of further steps may be chosen. Below follows a brief overview of the options available.

4.2.1 Lucas & Kanade Technique

Using the optical flow equation for a group of adjacent pixels and assuming that all of them have the same velocity, the optical flow computation task is reduced to solving a linear system. In a non-singular system for two pixels there exists a single solution of the system. However, combining equations for more than two pixels is more effective. In this case the approximate solution is found using the least square method. The equations are usually weighted. Here the following 2x2 linear system is used:

$$\sum_{x,y} W(x,y) I_x I_y u + \sum_{x,y} W(x,y) I_y^2 v = -\sum_{x,y} W(x,y) I_y I_t ,$$

$$\sum_{x,y} W(x,y) I_x^2 u + \sum_{x,y} W(x,y) I_x I_y v = -\sum_{x,y} W(x,y) I_x I_t ,$$

where $W(x,y)$ is the Gaussian window. The Gaussian window may be represented as a composition of two separable kernels with binomial coefficients. Iterating through the system can yield even better results. It means that the retrieved offset is used to determine a new window in the second image from which the window in the first image is subtracted, while It is calculated.

4.2.2 Horn & Schunck Technique

Horn and Schunck propose a technique that assumes the smoothness of the estimated optical flow field. This constraint can be formulated as

$$S = \iint_{\text{image}} \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right] (dx)\, dy$$

This optical flow solution can deviate from the optical flow constraint. To express this deviation the following integral can be used:

$$c = \iint_{\text{image}} \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right)^2 dx\,dy$$

The value $s + \lambda c$, where λ is a parameter, called Lagrangian multiplier, is to be minimized. Typically, a smaller λ must be taken for a noisy image and a larger one for

a quite accurate image. To minimize $s + \lambda c$, a system of two second-order differential equations for the whole image must be solved:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \lambda \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right) \frac{\partial I}{\partial x},$$

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = \lambda \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right) \frac{\partial I}{\partial x}.$$

Iterative method could be applied for the purpose when a number of iterations are made for each pixel. This technique for two consecutive images seems to be computationally expensive because of iterations, but for a long sequence of images only an iteration for two images must be done, if the result of the previous iteration is chosen as initial approximation.

4.2.3 Block Matching

This technique does not use an optical flow equation directly. Consider an image divided into small blocks that can overlap. Then for every block in the first image the algorithm tries to find a block of the same size in the second image that is most similar to the block in the first image. The function searches in the neighborhood of some given point in the second image. So all the points in the block are assumed to move by the same offset that is found, just like in Lucas & Kanade method. Different metrics can be used to measure similarity or difference between blocks - cross correlation, squared difference, etc.

4.3 Project Development

This section is formed by system confirmation and interpretation tools.

4.3.1 System Confirmation

The recorded developments are shown below:

1) Capturing the first frame from the file:

Figure 4.2

2) Converting the frame to the grey level:

Figure 4.3

3) Capture and convert the second frame to the grey level:

Figure 4.4

4) Getting difference of those two frames:

Figure 4.5

5) Detecting the moving objects from the difference frame:

Figure 4.6

6) Tracking the objects that move:

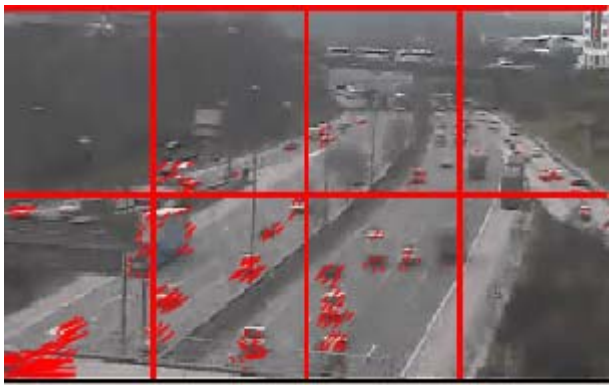

Figure 4.7

7) Determine moving points in each grid and showing them in the graph shown below:

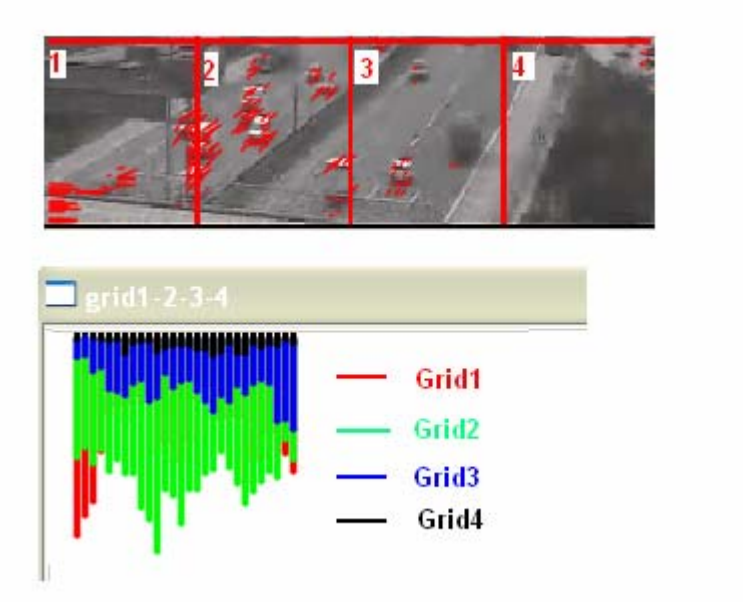

Figure 4.8 First Row's Grids

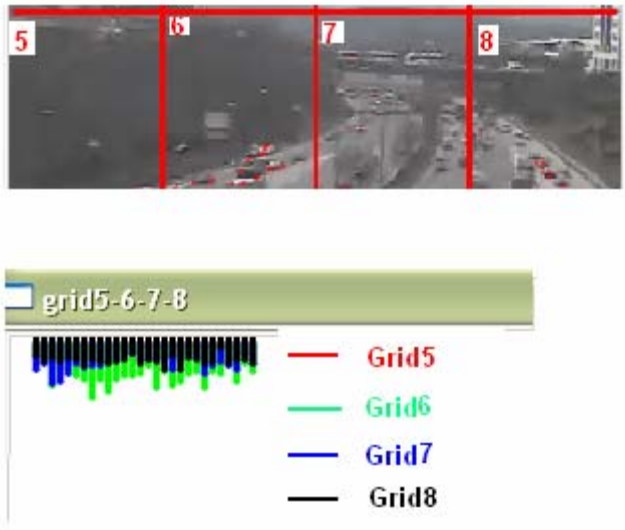

Figure 4.9 Second Row's Grids

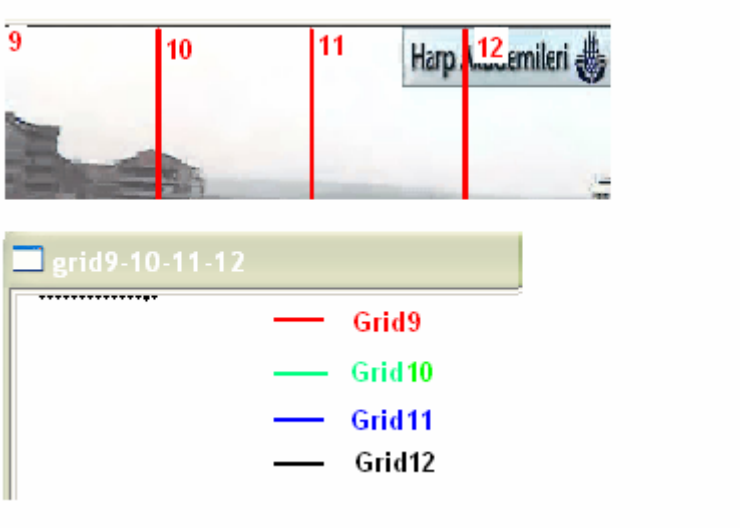

Figure 4.10 Third Row's Grids

4.3.2 Interpretation Tools

We used Fourier Transform for getting inference from the graphics.

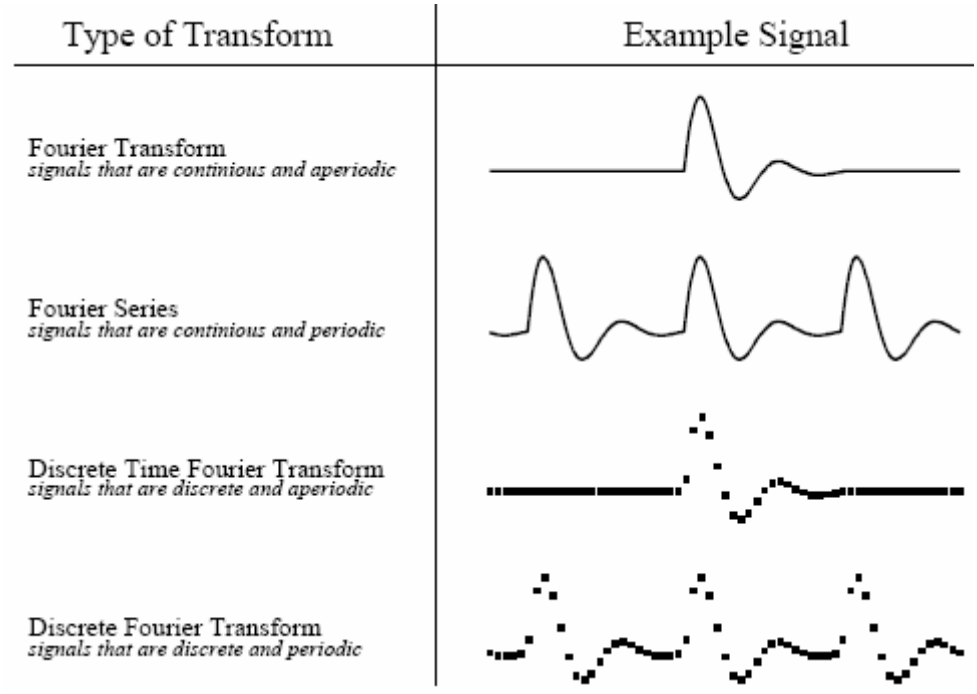

Figure 4.11 Illustration of the four Fourier transforms.

Discrete Time Fourier Transform is used as our time domain grid graphics are formed by aperiodic signals.

Aperiodic-Continuous

This includes, for example, decaying exponentials and the Gaussian curve.These signals extend to both positive and negative infinity *without* repeating in a periodic pattern. The Fourier Transform for this type of signal is simply called the **Fourier Transform**.

Periodic-Continuous

Here the examples include: sine waves, square waves, and any waveform that repeats itself in a regular pattern from negative to positive infinity. This version of the Fourier transform is called the **Fourier Series**.

Aperiodic-Discrete

These signals are only defined at discrete points between positive and negative infinity, and do not repeat themselves in a periodic fashion. This type of Fourier transform is called the **Discrete Time Fourier Transform.**

Periodic-Discrete

These are discrete signals that repeat themselves in a periodic fashion from negative to positive infinity. This class of Fourier Transform is sometimes called the Discrete Fourier Series, but is most often called the **Discrete Fourier Transform.**

Fourier transform changes an *N* point input signal into two *N*/2%1 point output signals. The input signal contains the signal being decomposed, while the two output signals contain the *amplitudes* of the component sine and cosine waves (scaled in a way we will discuss shortly). The input signal is said to be in the **time domain**. This is because the most common type of signal entering the DFT is composed of samples taken at regular intervals of *time*. Of course, any kind of sampled data can be fed into the DFT, regardless of how it was acquired. When you see the term "time domain" in Fourier analysis, it may actually refer to samples taken over time, or it might be a general reference to any discrete signal that is being decomposed. The term **frequency domain** is used to describe the amplitudes of the sine and cosine waves. The frequency domain contains exactly the same information as the timedomain, just in a different form. If you

know one domain, you can calculate the other. Given the time domain signal, the process of calculating the frequency domain is called **decomposition**, **analysis,** the **forward DFT**, or simply, **the DFT**. If you know the frequency domain, calculation of the time domain is called **synthesis**, or the **inverse DFT**.

Both synthesis and analysis can be represented in equation form and computer algorithms. The number of samples in the time domain is usually represented by the **variable *N***. While *N* can be any positive integer, a power of two is usually chosen, i.e., 128, 256, 512, 1024, etc. There are two reasons for this. First, digital data storage uses binary addressing, making powers of two a natural signal length. Second, the most efficient algorithm for calculating the DFT, theFast Fourier Transform (FFT), usually operates with *N* that is a power of two.Typically, *N* is selected between 32 and 4096. In most cases, the samples run from 0 to *N*&1, rather than 1 to *N*.

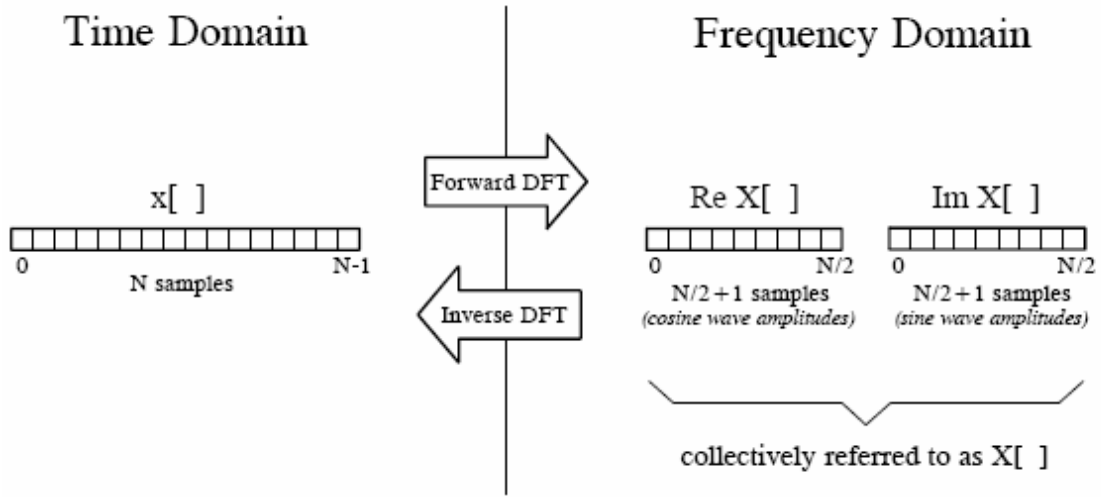

Figure 4.12 DFT Terminology.

DFT Basis Functions

The sine and cosine waves used in the DFT are commonly called the DFT basis functions. In other words, the output of the DFT is a set of numbers that represent amplitudes. The basis functions are a set of sine and cosine waves with unity amplitude. If you assign each amplitude (the frequency domain) to the proper sine or cosine wave (the basis functions), the result is a set of scaled sine and cosine waves that can be

added to form the time domain signal. The DFT basis functions are generated from the equations:

$$c_k[i] = \cos(2\pi ki/N)$$

$$s_k[i] = \sin(2\pi ki/N)$$

where: *c* is the cosine wave for the amplitude held in , and is *k*[ ] *ReX*[*k*] *sk*[ ] the sine wave for the amplitude held in *Im X*[*k*]. The parameter, *k*, sets the frequency of each sinusoid. In particular, *c* is the 1[ ] cosine wave that makes *one* complete cycle in *N* points, *c* is the cosine 5[ ] wave that makes *five* complete cycles in *N* points, etc. This is an important concept in understanding the basis functions; the frequency parameter, *k*, is equal to the number of complete cycles that occur over the *N* points of the signal.

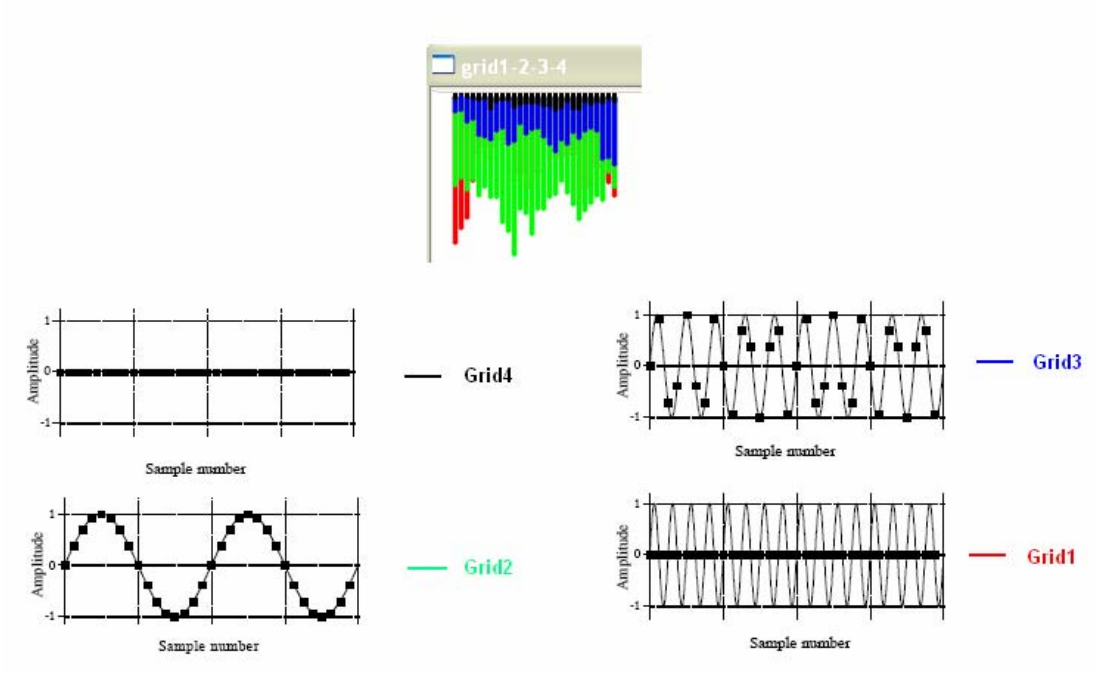

Figure 4.13 Getting Inference by DFT

BIBLIOGRAPHY

[1] http://www.isbak.com.tr/dvs.html

[2] Traffic Control Center Authority

[3] http://www.intel.com/cd/software/products/asmo-na/eng/perflib/ipp/238685.htm

[4] http://www.camerasuperstore.com/pecc1hire16x.html

[5]http://about.pricegrabber.com/search_attrib.php/page_id=189/form_keyword=Visual +C+++6.0

[6] OPENCV Tutorial

CV

Name - Surname: Zehra Gürbüz

Date of Birth: 17/04/1983

Date of Place: Elazığ

High School: Malatya Science High School

Internships: İSKİ

E-Grup Software Hause

İTÜ Information Process Center

Divit Multimedia