VEHICLE DETECTION FROM AERIAL IMAGES

by

Capt Muzzamil Noor
Capt Naveed Sadiq
Capt Fawad Khan

Submitted to the faculty of Computer Science,
Military College of Signals,
National University of Sciences and Technology,
Rawalpindi in partial fulfillment of B.E. degree in
Software Engineering.
May 2005

ABSTRACT

VEHICLE DETECTION FROM AERIAL IMAGES

BY

Muzzamil Noor, Naveed Sadiq, Fawad Khan

The proposed project deals with the automatic detection of vehicles, particularly military vehicles in high resolution aerial imagery. The extraction relies upon local features of vehicles. To model a vehicle on local level, a model representation is used that describes the prominent geometric features of vehicles. The model is adaptive because, during extraction, the expected saliencies of various edge features are automatically adjusted depending on viewing angle, vehicle color measured from the image, and current illumination direction. The extraction is carried out by matching this model ”top-down” to the image and evaluating the support found in the image Hence, training data samples of vehicles are first clustered and statistical parameters corresponding to each cluster are obtained. Vehicles are detected by searching the test image for patches of vehicles at all points in the image and across different scales. Applying this technique to the military vehicles particularly fighting vehicles presents peculiar problems of its own as they differ in geometric and statistical representation from that of the soft vehicles. The project is aimed at facilitating automatic aerial imagery analysis, which is a very tedious job if done manually, simultaneously maximizing the accuracy and performance.

ACKNOWLEDGMENTS

We would like to express our gratitude to all those who gave us the possibility to complete this project. We want to thank computer science department for their support and for encouraging us to complete our project. We would also like to express our special thanks to our supervisor Lt Col Naveed Sarfraz Khattak for his advice and guidance during this project. We would like to give our special thanks to our families whose patient love enabled us to complete this project.

TABLE OF CONTENTS

Chapters **Page No**

List of Figures …………………………………………….................... v

List of Tables ……………………………………………………..…. vii

1. Introduction …………………………………………………………..1

1.1 General ……………………………………………………………..1

1.2 Motivation …………………………………………………………..1

1.3 Objectives of Project ..……………………………………………………3

2. Literature Review ……………………………………………………… 4

2.1 Introduction ………………………………………………………....4

2.2 Aerial Photographs and Satellite Images ……………………………….. 5

2.3 Thresholding Techniques ……………………………………………….. 7

2.4 Miscellaneous Techniques ……………………………………………… 9

2.5 Conclusion ……………………………………………………………… 9

3. Data Sets ………………………………………………………………… 10

3.1 Introduction ...……………………………………………………...10

3.2 Data Set Description ………………………………………………….10

4. Graphical User Interface …………………………………….......... 19

4.1 Introduction ………………………………………………………… 19

4.2 ImageJ Class Structure ………………………………………………… 20

4.3 The Plugin Concept of ImageJ ………………………………………… 20

4.4 Integrating Plugins into the ImageJ GUI ……………………………… 20

4.5 Image Representation in ImageJ ……………………………………….. 20

4.6 GUI of the Project ……………………………………………………….21

4.6.1 User Interface ………………………………………………………… 21

4.6.2 Result Windows ……………………………………………………… 22

5. Methodologies ………………………………………………………..... 24

5.1 Preprocessing for Brightness and Contrast ….…………………………... 24
5.2 Choosing a Smoothing Filter ……………………………………………. 26
5.3 Derivative of Gaussian Filters ………………………………………….. 27
5.4 Identifying Edge Points from Filter Outputs ……………………………. 29
5.5 Steerable Filters ………………………………………………………… 32
5.6 Design of Steerable Filters for Feature Detection ………………………. 34
5.7 Optimality Criterion …………………………………………………..35
5.7.1 Signal to Noise Ratio ………………………………………………… 35
5.7.2 Localization …………………………………………………….. 35
5.7.3 Elimination of False Oscillations ……………………………………... 36
6. Results and Analysis …………………………………………………... 38
6.1 Introduction ……………………………………………………………... 38
6.1.1 Test Image ATR ……………………………………………………… 38
6.1.2 Test Image Img83 …………………………………………………….. 39
6.1.3 Test Image v2 ………………………………………………………… 42
6.1.4 Test Image v10 ………………………………………………………... 44
6.1.5 Test Image v18 ………………………………………………………... 46
6.1.6 Test Image v22 ………………………………………………………... 48
6.1.7 Test Image v23 ………………………………………………………... 50
6.1.8 Test Image v24 ………………………………………………………... 52
6.1.9 Test Image v25 ………………………………………………………... 54
6.1.10 Test Image v26 ……………………………………………………….. 56
6.2 Efficiency Analysis through Data Set ………………………………….... 58
7. Conclusion ……………………………………………………………… 61
8. Future work …………………………………………………………….. 62
9. List of References ……………………………………………….............. 63
10. Appendix 1 ……………………………………………………………. 65
11. Appendix 2 ……………………………………………………………. 70

LIST OF FIGURES

Figure **Page No**

Figure 1.1: Variations causing image unreadable …………………................................ 2

Figure 1.2: Shadows obstructing image……………………………………………... 3

Figure 3.1: Test Image ATR….……………………………………………………... 13

Figure 3.2: Test Image img83……………………………………………………….. 13

Figure 3.3: Test Image v2 ………………………………………………………….. 14

Figure 3.4: Test Image v10 …………………………………………………………. 14

Figure 3.5: Test Image v18 ……….……………………………………………….... 15

Figure 3.6: Test Image v22 …………………………………………………………. 15

Figure 3.7: Test Image v23 …………………………………………………………. 16

Figure 3.8: Test Images v24 ….…………………………………………………….... 16

Figure 3.9: Test Images v25 ……………………………………………………….... 17

Figure 3.10: Test Images v26 ……………………………………………………….. 17

Figure 4.1: User interface …………………………………………………………… 22

Figure 4.2: Result Window ………………………………………………………...... 23

Figure 5.1: Reconstruction of gradient magnitude .……………………………………... 29

Figure 5.2: Image Examples ….……………………………………………………… 31

Figure 5.3: Edge Detection with different Gradient magnitude...……………………..... 32

Figure 6.1: Edge image of ATR ……………………………………………………… 38

Figure 6.2: Final output of ATR …………………………………………………....... 38

Figure 6.3: Edge image of Img 83 ………………………………………………………… 40
Figure 6.4: Final output of Img 83 with concentration …………………………………… 41
Figure 6.5: Final output of Img 83 without concentration ..…………………...……….. 42
Figure 6.6: Edge image of v2 ………………………………………………………... 43
Figure 6.7: Final output of v2 with concentration ………………..………………… 43
Figure 6.8: Final output of v2 without concentration …………………………………... 44
Figure 6.9: Edge image of v10 ……………………………………………………..... 45
Figure 6.10: Final output of v10 with concentration …………………………….….... 46
Figure 6.11: Final output of v10 without concentration ....…………………………. 46
Figure 6.12: Edge image of v18 ……………………………………………………... 47
Figure 6.13: Final output of v18 with concentration ………………………...………. 47
Figure 6.14: Final output of v18 without concentration ………………………….…….. 48
Figure 6.15: Edge image of v22……………………………………………………..…..49
Figure 6.16: Final output of v22 with concentration …..………………………………. 49
Figure 6.17: Final output of v22 without concentration ………………………………... 50
Figure 6.18: Edge image of v23 ……………………………………………………... 51
Figure 6.19: Final output of v23 with concentration ……………………………….... 51
Figure 6.20: Final output of v23 without concentration ………………………….…….. 52
Figure 6.21: Edge image of v24 ……………………………………………….……….. 53
Figure 6.22: Final output of v24 with concentration …..………………………………. 53
Figure 6.23: Final output of v24 without concentration …………………………….... 54
Figure 6.24: Edge image of v25 ……………………………………………………... 54
Figure 6.25: Final output of v25 with concentration …..………………………………. 55
Figure 6.26: Final output of v25 without concentration without concentration………..… 55
Figure 6.27: Edge image of v26 ..……………………………………………….…. 56
Figure 6.28: Final output of v26 with concentration ...………………………………… 57
Figure 6.29: Final output of v26 without concentration ……………………...……. 57
Figure 6.30: Error Analysis ……………………………………………………….. 60
Figure 6.31: Performance Analysis …………………………………………….…. 60

LIST OF TABLES

Table **Page No**

Table 3.1: Test case Descriptions …………………………………………………. 11

Table 6.1: Results with Concentration Conditions …………………………………... 58

Table 6.2: Results without Concentration Conditions ………………………………. 58

C h a p t e r   1

1. Introduction

1.1 General

This project deals with automatic detection of vehicles in high resolution aerial imagery. Detection from aerial image is easier than from detection from an arbitrary viewpoint, in that the viewpoint is constrained. However, it is still not as easy as it may seem to be. Example images are shown in Figure1.1. Although the viewpoint is constrained, there are still variations that make the cars have different appearance. The image resolution is high even then not many details are visible. Some cars are heavily obscured by the environment in the images, mostly tree branches (Figure1.1). 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 (Figure 1.2). 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. It is needed to account for all these difficulties to get a reasonable good system.

1.2 Motivation

Military operation planning is the foremost field which requires presence of vehicles at particular places while analyzing the aerial images. Other fields of application are found in the context of military reconnaissance and extraction of geographical data for Geo-Information Systems (GIS), e.g., for site model generation and up-date. Traffic-related data play an important role in urban and spatial planning, e.g., for road planning and for estimation / simulation of air and noise pollution. In recent years, attempts have been made to derive traffic data also from aerial images, because such images belong to the fundamental data sources in many fields of urban planning. Therefore, an algorithm that automatically detects and counts vehicles in aerial images would effectively support traffic-related analyses in urban planning. Furthermore, because of the growing amount of traffic, research on vehicle detection is also motivated from the strong need to automate the management of traffic flow by intelligent traffic control and traffic guidance systems. The aerial images used are grayscale images taken mostly from a vertical or slightly oblique viewpoint. The length of a typical car in the datasets ranges from 13 to 26 pixels in image. This information has been

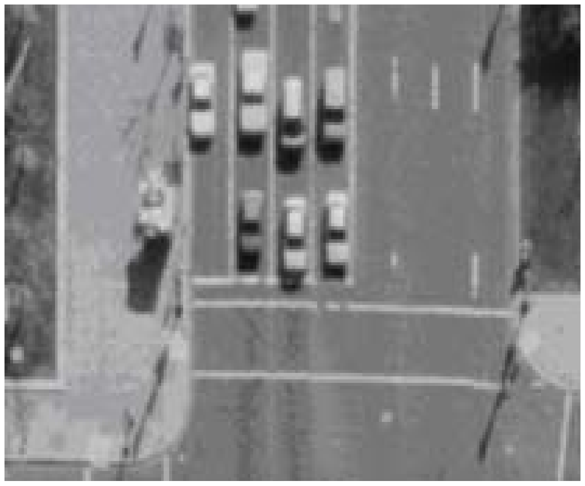


Figure 1.1 Variations in image.

used to make a geometric template. This template is matched at varying sizes and at all locations in the image. The likely candidates are classified as such and then further criterion on the basis of other features in the vicinity is applied for the final classification of the candidate as a vehicle or otherwise.

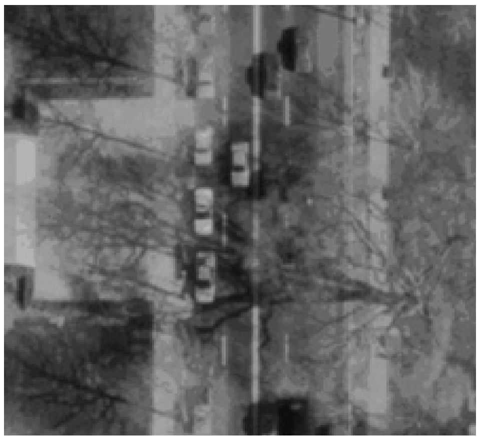


Figure 1.2. Shadows obstructing image

1.3 Objectives of project

The objectives intended to be achieved during course of this project are: To develop an automated system for detection of vehicles from aerial images, provide post detection interactive analysis, system efficiency analysis is to be done for assessing system efficiency. The concept of steerable filters and rectangular feature extraction algorithm are used for feature extraction. These features such as lines, corners are then processed for extraction of relative attributes, for example distance, angle, and orientation, for matching purposes. After that detection algorithm is applied to detect the vehicles. The image is then processed for feature extraction and representation and finally a processed image appears with vehicles earmarked on them under a white rectangle along with an efficiency window highlighting the time taken and number of vehicles detected.

Chapter 2

2. Literature Review

2.1 Introduction

Vehicle detection from satellite imagery is a relatively new topic in the civilian Community. The lack of sufficiently high-resolution satellite data has been the primary cause. However, with the recent launch of several commercial satellites with high resolution data, it should be seeing several inroads in this area in the near future. Previous attempts have primarily focused on using aerial photographs, although these have been limited. In one of the early studies Taylor [1] used an analytical stereo plotter to extract traffic parameters from aerial photographs. Agin [2] tried to use artificial intelligence concepts for this purpose.

Some of the recent attempts for vehicle detection in aerial photographs are described in the next sections. Some of the methods for vehicle detection and tracking being used by the surveillance community are also discussed. Their attempts focus on tracking and threat assessment of vehicle and object movements. It is important to note the significant differences in the problem of vehicle detection from aerial imagery compared to that using other sensor systems like video imagery and satellite photographs. Firstly, the resolution of the satellite images being used for the purposes of this study is 1-m, which is considerably coarser than the resolution on the order of 20-cm for the video imagery and the aerial photographs that were used in other studies. Availability of multiple snapshots of the same area is another advantage that aerial photographs (several overlapping images) and video imagery (multiple frames) enjoy over the satellite imagery for which only one snapshot is available for an area at a given time. Thus, motion cues that can be extracted from video imagery are not available in the case of satellite imagery. Overlapping images can be used to generate a pavement background image that can be compared with a new image to detect regions of change or motion on the pavement.

2.2 Aerial Photographs and Satellite Images

Several attempts have been made for detecting vehicles in aerial images. Here described are some of the approaches taken by researchers for vehicle detection in aerial images. Moon et al. [3] use a simple geometric edge model in conjunction with contextual information for vehicle detection in aerial imagery of parking lots. They construct a vehicle detection operator by combining four elongated edge operators designed to collect edge responses from the sides of the vehicles. The operator collects responses at the centroids of the four operators that are combined to form a rectangle. Suitable thresholds and rules are applied to filter out spurious responses. The algorithm was tested on images with varying conditions of camera angle, camera orientation and illumination. They also propose performance analysis measures of vehicle detection algorithms.

Rajgopalan et al. [4] describe a method that “approximately models the unknown distribution of the images of the vehicles by learning higher order statistics (HOS) information of the vehicle class from sample images.” The basic idea is that the joint density function for the vehicle class is unlikely to be a simple Gaussian distribution. Hence, training data samples of vehicles are first clustered and statistical parameters corresponding to each cluster are obtained. They go on to propose a clustering method based on a HOS-based decision measure. Vehicles are detected by searching the test image for patches of vehicles at all points in the image and across different scales. The statistical information about the background is learned on the fly. The HOS measure is essentially a class-distance measure expressed as a function of series expansion for a multivariate probability density function.

Zhao and Nevatia [5] consider the vehicle detection problem as a 3D object recognition problem. Based on their observations from psychological tests they choose a set of features for recognition. The features include boundary of car body, boundary of windshield, and the vehicle shadow. They then construct a Bayesian network that is used to integrate all the features for the recognition process. Their approach seems to be quite robust, even in cases of partial occlusion by objects such as trees.

Ruskoné et al. [6] use a structure-based approach to identify vehicles in aerial images. They propose a structural hierarchical model that tries to capture the understanding of structure at different levels of organization or detail. The idea of different levels is as follows. At a pixel level the radiometry would be a measure to place a pixel in the vehicle class. This clustering process gives rise to the vehicle level. The vehicles are then organized as lines, say two identically oriented lines, and meta-lines, say two car-lines with relative orientation of 0º, 45º, or 180º. The car detection is validated using the organization of vehicles into lines and meta-lines. They use a neural network for the process. In an attempt to account for all or most of the conditions, like occlusions and less than perfect geometry, they use an exhaustive database to train the network. Sastry [7] reports Ruskoné's observation that finding suitable features to train the network was a difficult part of the problem.

Structural analysis was also the key factor in research by Nagao and Matsuyama [8]. They sought a system that would automatically locate different classes of objects in an aerial photograph by using common sense knowledge about the world. They hypothesized the presence of cars by the absence of a large homogeneous region, such as vegetation, water and shadow. However, they realized that contextual information, like the presence of a road, was required to improve the accuracy of detection. They used the above mentioned hypothesis expression, while also supplying the contextual information for detection. They then use other characteristic features, for example, rectangularity, to further refine the results.

Sastry [7] proposed a wavelets and principal components based method for identification of vehicles in aerial photographs and satellite imagery. He creates a pyramid of texture and mean images. Principal component analysis is performed on this set of bands. One of the resultant principal component bands is then selected for threshold selection. The resulting blobs are then analyzed for size and shape characteristics. Some bands consistently showed desirable properties for applying the threshold selection algorithms.

The NCRST-F group at The Ohio State University has developed a simplified methodology derived from the method described by Sastry [7]. They have also proposed a method that uses a background estimate to associate a probabilistic measure of a pixel being dynamic or stationary [9]. Starting with a prior estimate of the probability, a Bayesian scheme is used to update the probabilities until convergence is achieved. They propose to build the background using a Kalman filter-like approach, i.e., using an average of several images taken at different times.

2.3 Thresholding Techniques

Thresholding is one of the common and simple approaches to image segmentation. It involves limiting the intensity values within an individual image to a certain bounded range or ranges. The threshold value can be selected manually by visually inspecting the histogram of the image or can be selected by automatic software programs. Classical threshold selection methods have relied on the one-dimensional histogram of the image to be segmented. A one-dimensional histogram provides an estimate of the probability of occurrence for each gray level in the gray level range of the image. Several schemes, like Kullback-Leibler [10] distance based threshold, have been proposed to select thresholds from one-dimensional histograms. However, the main drawback of one-dimensional histogram-based methods is that they do not take the pixel dependencies or the spatial distribution into account while defining the information measures. To overcome these deficiencies, methods based on higher order entropies have been proposed. Second-order entropies have found special attention in this regard. Pal and Pal [11] defined the concepts of local and joint entropy. Chang et al. [12] used the concept of relative (or cross) entropy for threshold selection. They first calculate the gray level transition probability distribution of the co-occurrence matrix for the image and for a thresholded binary image. The threshold that minimizes the difference between the two transition probabilities is chosen. Baghdadi et al. [13] use the same concept, but they divide the image into blocks for threshold selection.

Lee et al. [14] have attempted to integrate the concept of local entropy into that of relative entropy to get a more efficient algorithm for threshold selection. Their approach was used for the threshold selection procedure implemented in this study. They also attempt to account for the locality of objects and the different resolutions of thresholding by using the Hilbert space filling sequence for calculating the co-occurrence matrix.

Cheng et al. [15] present a thresholding approach that performs a fuzzy partition on the two-dimensional histogram of the image based on fuzzy relaxation and the maximum entropy principles. They claim to have found better results than the usual two-dimensional hard threshold selection algorithms. Yanowitz and Bruckstein [16] present a method for image segmentation that attempts to address the issue of poor and non-uniform illumination. An adaptive threshold surface is determined by interpolating the image gray levels at points where the gradient is high, indicating probable object edges. This surface is then used to segment the image. The adaptive nature of the surface tends to eliminate the illumination characteristics at a point. The threshold surface may intersect the background creating objects where none exists. However, this is easily rectified since such points will have low gradient values.

2.4 Miscellaneous Techniques

Building extraction from aerial images is an area that can yield useful strategies that might be relevant for our problem. Most building strategies tend to use geometric representation. This involves extracting lines (or edges), breaking them into segments based on curvature criterion, or by identifying corners and junctions and then perceptually organizing them into objects. Some of the significant works in this direction are Sarkar and Boyer [17], Roux and McKeown [18], and Lin and Nevatia [14].

2.5 Conclusion

This chapter presented a selection of approaches that have been developed for detecting and tracking vehicles and objects in imagery derived from several different classes of sensors. Various strategies, each trying to exploit a different set of invariant characteristics in the sensor data, were described.

C h a p t e r 3

3. Data Set

3.1 Introduction

The objective of this study is to test the performance of an image processing system for vehicle detection, classification and counting from high-resolution satellite data. For this purpose we have extracted a set of images to be tested so that the software performance can be characterized under conditions of varying pavement material and geometry, and atmospheric conditions. Images used for this study are aerial photographs scanned to produce 1-m resolution digital data. The general characteristics of the two types of imagery are described in the next section. The salient features of the dataset are described in the next section. Conditions of each image area are listed to characterize performance.

3.2 Data Set Description

After extensive search from internet and military resources we have gathered a small but representative image data set which fulfills the constraints and assumptions made for the project. These images are 256 level gray scale (8-bit) images. As stated before, the test cases are extracts of roadway from several scanned aerial photographs and satellite images. These images present various conditions under which the performance of the image processing algorithms will be characterized. Careful attention has been paid to bring all the conditions likely to pose a challenge to the system in detection of vehicles in the data set. Table 3.1 provides a summary of the characteristics of these selected images. The images are shown in Figure 3.1 to Figure 3.10.

Image Label	**Characteristics**
ATR	1. Cross country terrain back ground 2. Tank image 3. Size of image: 134 × 100 4. Memory size 13k.
Img83	1. Lot of built up area, with rectangular shape buildings. 2. Road passing through the centre of image. 3. Trees present along the roads and buildings 4. Mostly vehicles at the border of the road. 5. Straight roadway section. 6. Size of image: 505 × 477. 7. Memory size 235k.
v2	1 Curved road segment 2. Predominantly dark image. 3. Presence of large water body. 4. Bridge. 5. Built up area with trees shadowing some vehicles present. 6. Size of image: 693 × 540. 7. Memory size 365k.
v10	1. Haphazardly scattered vehicles. 2. Wide variety of vehicles greys scales. 3. Straight road segment but most of the vehicles are off the road. 4. Vehicles of varying sizes and orientations. 5. Size of image: 666 × 400. 6. Memory size 260k.
v18	1. Mountainous terrain. 2. Low resolution image. 3. Predominantly dark image. 4. Widely dispersed vehicles 5. Size of image: 720 × 540. 6. Memory size 379k.

Image Label	**Characteristics**
v22	1. All vehicles on the road. 2. Few buildings with rectangular profiles are present. 3. Some vehicles are merging with the road texture. 4. Vehicles of varying sizes. 5. Vehicles are fairly dispersed. 6. Size of image: 600 × 3297. Memory size 192k.
v23	1. Road crossing. 2. Good contrast between vehicles and pavement. 3. Presence of rectangular shadows. 4. Vehicles are fairly dispersed. 5. Size of image: 335 × 219 6. Memory size 71k.
v24	1. Poor contrast between vehicles and surroundings 2. Cross country terrain. 3. Tanks are parked in clusters. 4. Noisy image. 5. Size of image: 324 × 249. 6. Memory size 78k.
v25	1. Parking lot. 2. Absence of any feature other than vehicles. 3. Vehicles of large variety of shades. 4. Occluded vehicles. 5. Large no of vehicles, evenly dispersed. 6. Size of image: 368 × 64. 7. Memory size 94k.
v26	1. Multi tiered road structures. 2. Less vehicles widely dispersed. 3. Different shades of vehicles. 4. Back ground is dark and has good contrast with road texture. 5. Size of image: 500 × 300. 6. Memory size 146k.

Table 3.1 Test case descriptions

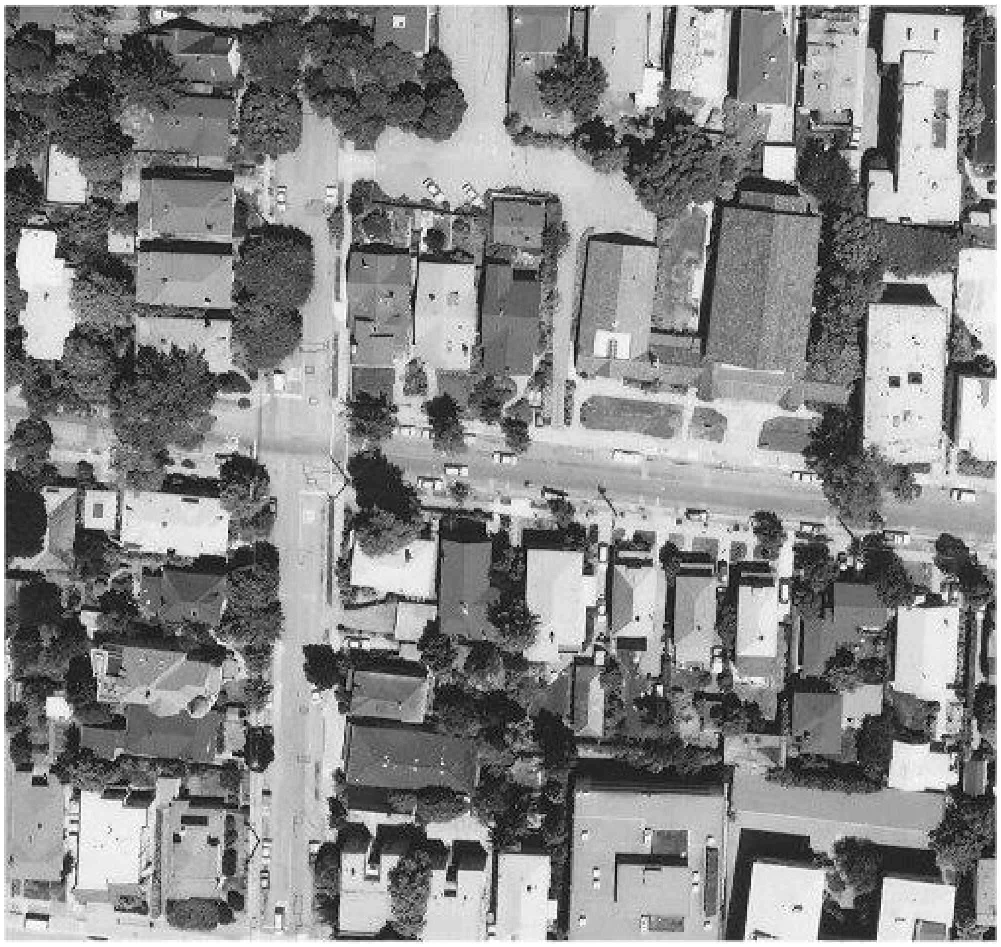


Figure 3.2 Test Image Img83

These conditions include: Vehicle contrast with respect to the pavement, which is affected by atmospheric conditions, such as presence of cloud cover, and by the pavement type and vehicle tone. Presence of vehicle-like objects in the image, for example, pavement markings. Road geometry, whether curved or straight, with respect to the image orientation. This analysis does not restrict itself to highways and freeways only. The reason being that at the level of arterials, several other factors would need to be considered. Like the orientation and the relative positioning of other features in the surroundings.

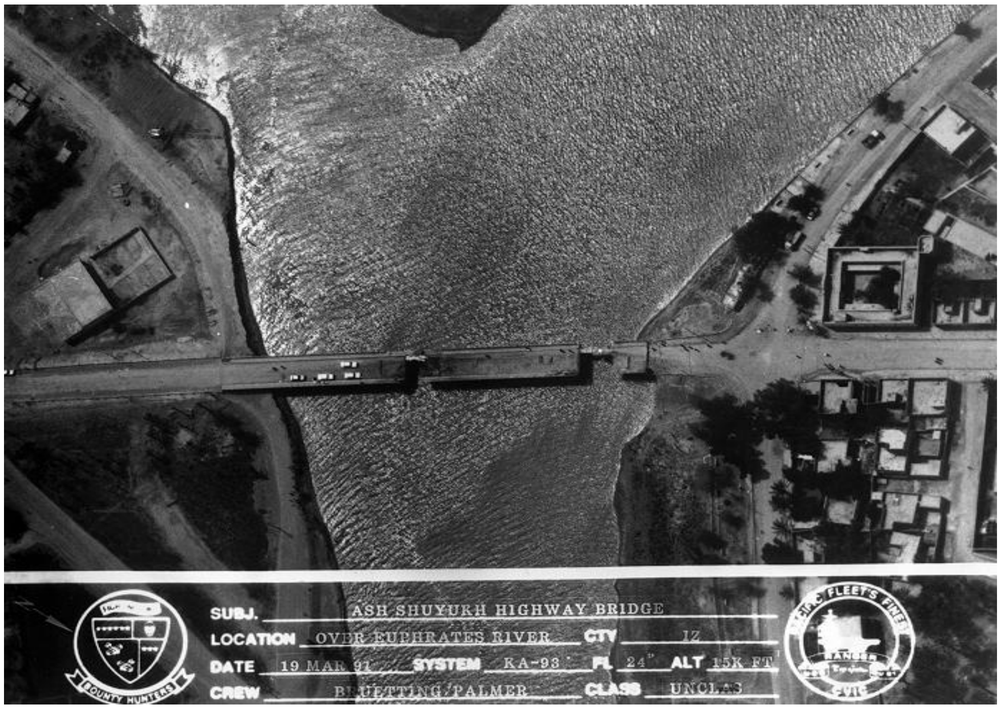

BOUNTY HUNTERS
SUBJ. ASH SHUYUKH HIGHWAY BRIDGE
LOCATION OVER EUPHRATES RIVER CTY IZ
DATE 19 MAR 91 SYSTEM KA-93 FL 24" ALT 15K FT
CREW BRUETTING/PALMER CLASS UNCLAS
PACIFIC FLEET'S FINEST
RANGER
CV-61

Figure 3.3 Test Image v2

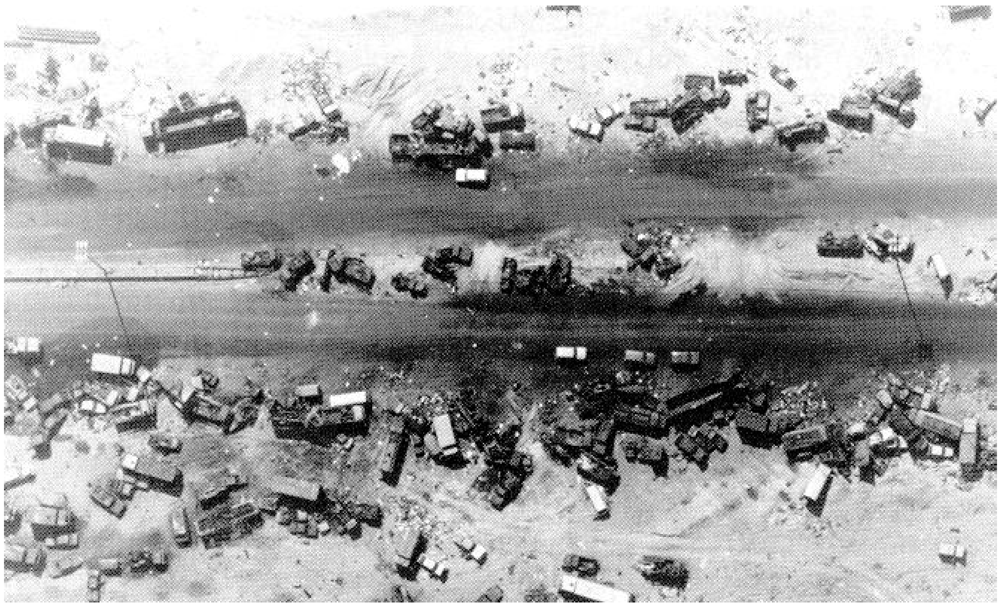


Figure 3.4 Test Image v10

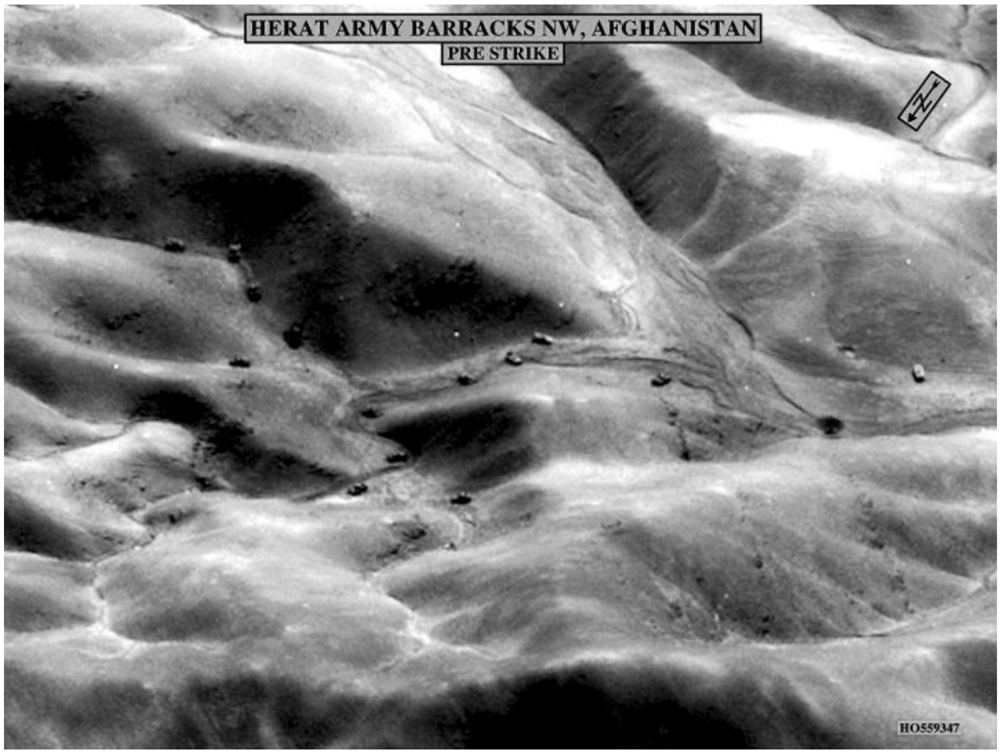


Figure 3.5 Test Image v18

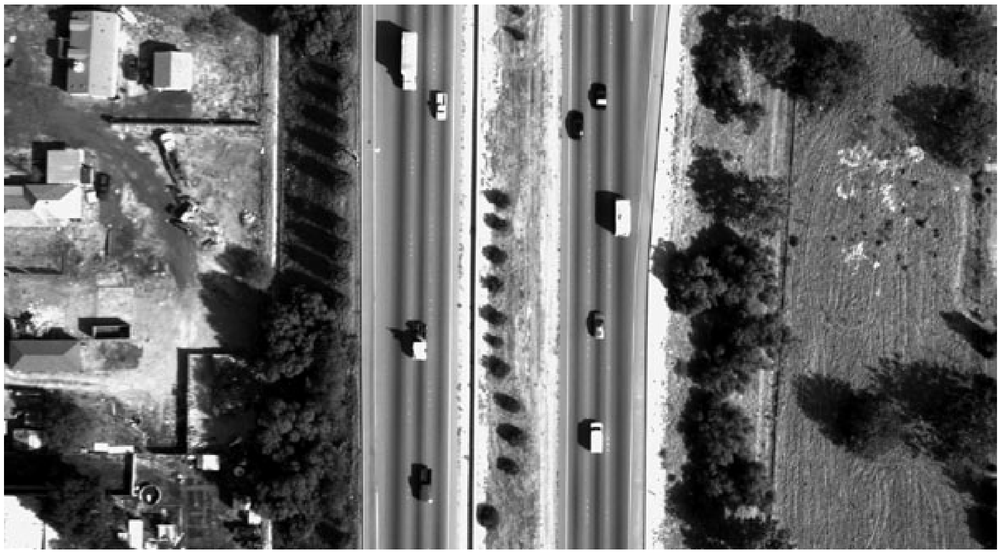


Figure 3.6 Test Image v22

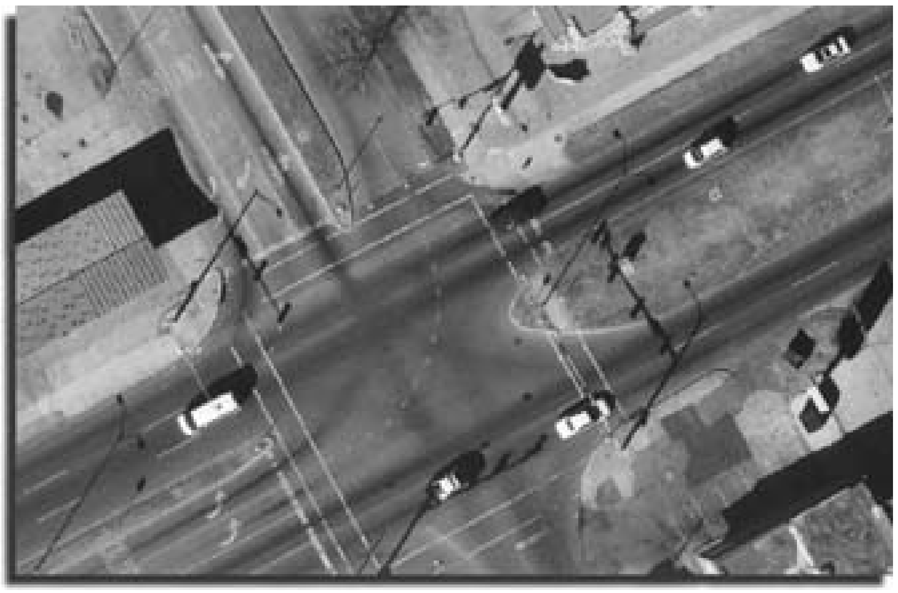


Figure 3.7 Test Image v23

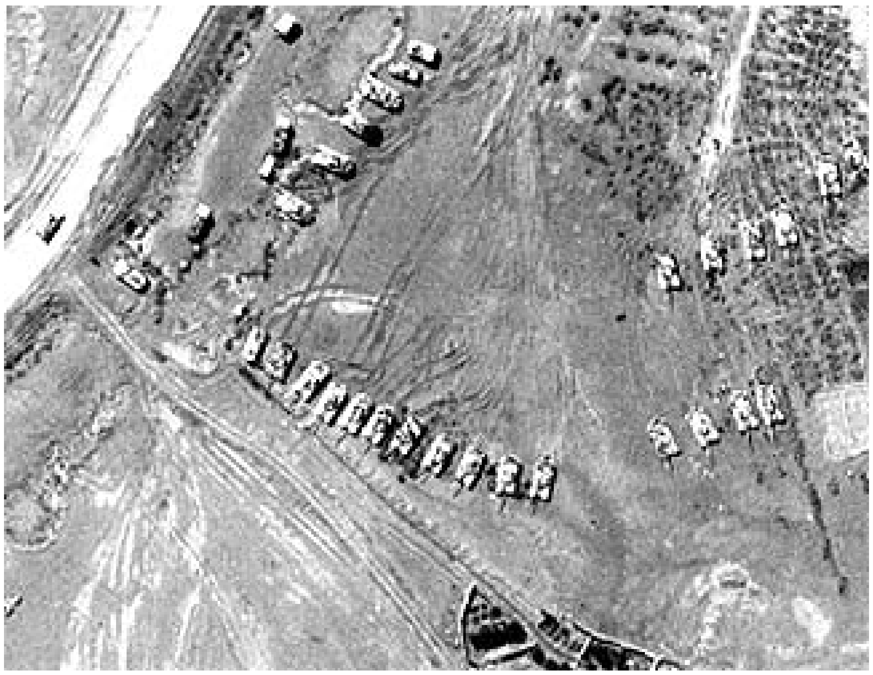


Figure 3.8 Test Image v24

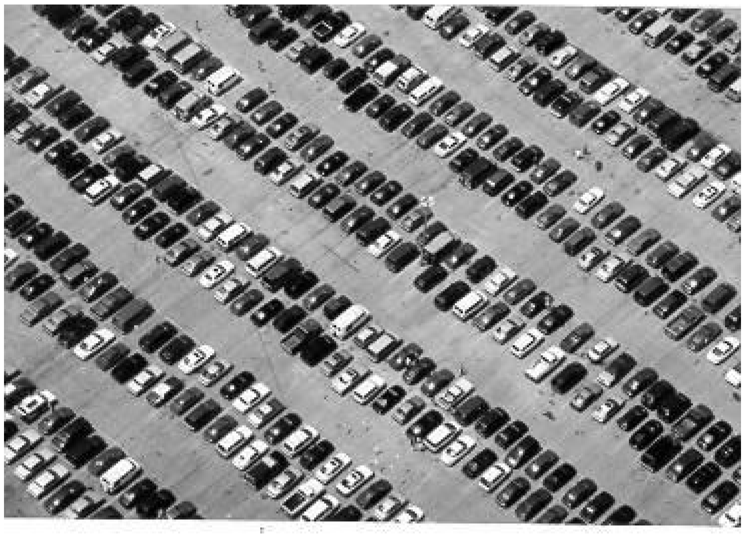


Figure 3.9 Test Image v25

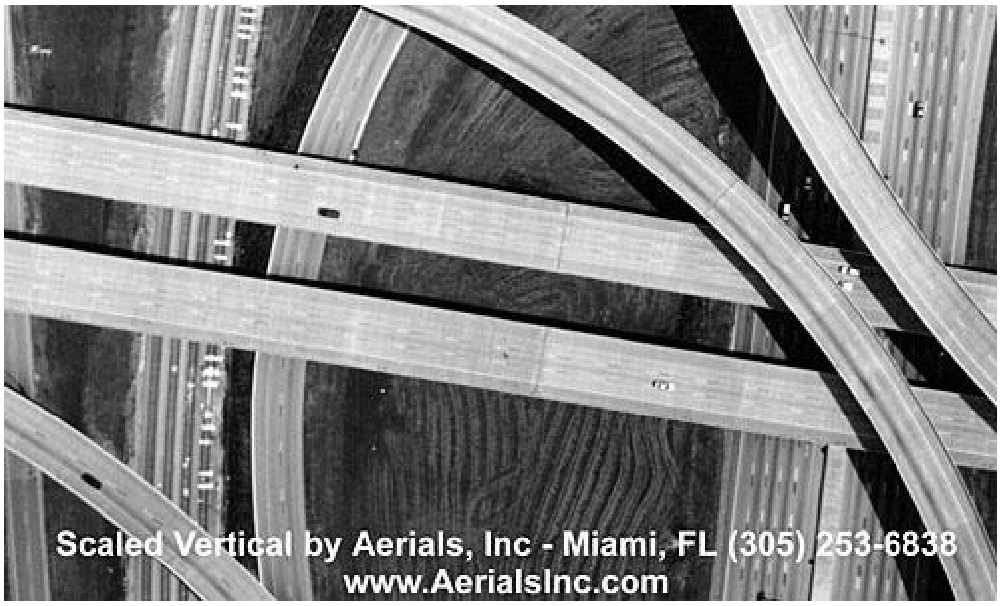


Figure 3.10 Test Image v26

As can be seen from the images, primary interest is in a combination of three features or conditions that may affect the vehicle detection algorithms. These conditions include: Vehicle contrast with respect to the pavement, which is affected by atmospheric conditions, such as presence of cloud cover, and by the pavement type and vehicle tone. Presence of vehicle-like objects in the image, for example, pavement markings. Road geometry, whether curved or straight, with respect to the image orientation. This analysis restricts itself to highways and freeways only. The reason being that at the level of arterials, several other factors would need to be considered. These factors include vehicles are more likely to be obscured, the presence of objects having signatures similar to vehicles, and the presence of shadows from surrounding buildings that would be likely to occur when an arterial passes through an area that has high-rise buildings.

C h a p t e r 4

4. Graphical User Interface

4.1 Introduction

ImageJ is a public domain Java image processing program inspired by NIH Image for the Macintosh. It runs, either as an online applet or as a downloadable application, on any computer with a Java 1.1 or later virtual machine. It can display, edit, analyze, process, save and print 8-bit, 16-bit and 32-bit images. It can read many image formats including TIFF, GIF, JPEG, BMP, DICOM, FITS and "raw". It supports "stacks", a series of images that share a single window. It is multithreaded, so time-consuming operations such as image file reading can be performed in parallel with other operations. It can calculate area and pixel value statistics of user-defined selections. It can measure distances and angles. It can create density histograms and line profile plots. It supports standard image processing functions such as contrast manipulation, sharpening, smoothing, edge detection and median filtering. It does geometric transformations such as scaling, rotation and flips. Images can be zoomed up to 32:1 and down to 1:32. All analysis and processing functions are available at any magnification factor. The program supports any number of windows (images) simultaneously, limited only by available memory. Spatial calibration is available to provide real world dimensional measurements in units such as millimeters. Density or gray scale calibration is also available. ImageJ was designed with an open architecture that provides extensibility via Java plug-in. Custom acquisition, analysis and processing plugins can be developed using ImageJ's built in editor and Java compiler. User-written plug-in make it possible to solve almost any image processing or analysis problem. The program supports any number of windows (images) simultaneously.

4.2 ImageJ Class Structure

This is an overview of the class structure of ImageJ. It is by far not complete, just the most important classes for plug-in programming are listed and briefly described. ImageJ can be run as applet or as application. This is the applet class of ImageJ. The advantage of running ImageJ as applet is that it can be run (remotely) inside a browser; the biggest disadvantage is the limited access to files on disk because of the Java applet security concept, if the applet is not signed.

4.3 The Plugin Concept of Imagej

The functions provided by ImageJ's menu commands (most of them are in fact plugins themselves) can be extended by user plugins. These plugins are Java classes implementing the necessary interfaces that are placed in a certain folder. Plugins can be written with ImageJ's built-in plugin editor (accessible via the menus "Plugins/New..." and "Plugins/Edit..."), with a text editor of your choice or they can be generated using ImageJ's plugin recorder. In any case plugins can be compiled and run inside ImageJ. Plugins found by ImageJ are placed in the Plugins menu or in submenus of it.

4.4 Integrating Plugins into the ImageJ GUI

Like commands, plugins can be accessed via hot-keys. One can create a new hot-key by selecting "Create Shortcut" from the menu "Plugins / Shortcuts". When the plugin interfaces were discussed it was talked about arguments that can be passed to plugins. Installing a plugin using the menu command "Plugins / Shortcuts / Install Plugin ..." places the plugin into a selected menu, assigns a hot-key and passes an argument. "Plugins / Shortcuts / Remove ..." removes a plugin from the menu.

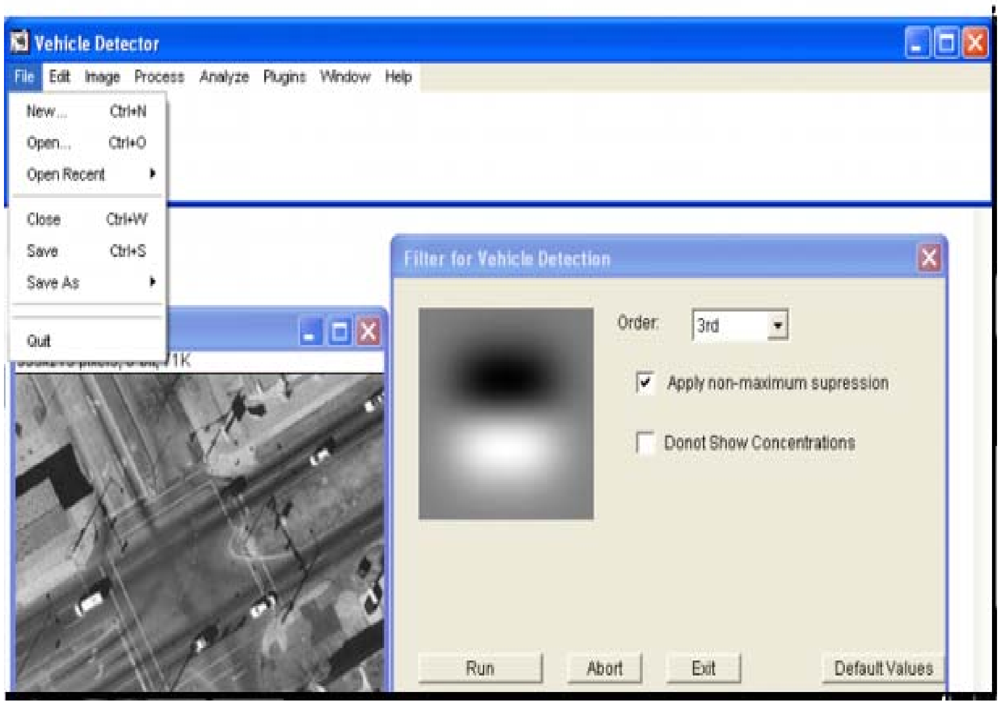
4.5 Image Representation in ImageJ

In ImageJ, images are represented by ImagePlus and ImageProcessor objects in ImageJ. In this section a closer look at the way images are handled by ImageJ. Images are large arrays of pixel values. But it is important to know how these pixel values should be interpreted. This is specified by the type of the image. ImageJ knows five image types. 8 bit grayscale image can display 256 grayscales and a pixel is represented by a byte variable. 8 bit color image can display 256 colors that are specified in a lookup table (LUT) and a pixel is represented by a byte variable. 16 bit grayscale image can display 65, 536 grayscales, and a pixel is represented by a short variable. RGB color image can display 256 values per channel and a pixel is represented by an int variable. In 32 bit floating point grayscale image, a pixel is represented by a float variable.

4.6 GUI of the project

4.6.1 User Interface

User Interface of the project consists of a main window which gives the user menus as shown in Figure 4.1. File menu, includes the options of new, open, close, save, save as and quit. Edit menu includes the options of undo, cut, copy, paste, clear, fill and draw. Image menu includes the options of type, properties, and colors, duplicate, rename, scale, rotate and zoom. Process menu contains the important functions of smooth, sharpen, contrast, noise, shadows etc. Analyze menu includes measure, summarize, label, clear results, set measurements etc. Plugins, In addition to compilation and run plugins menu includes the most important function of detector which implements all the functions automatically described above. Windows menu includes the functions cascade, put behind etc. Image menu includes the options of type, properties, and colors, duplicate, rename, scale, rotate and zoom. Process menu contains the important functions of smooth, sharpen, contrast, noise, shadows etc. Analyze menu includes measure, summarize, label, clear results, set measurements etc. Edit menu includes the options of undo, cut, copy, paste, clear, fill and draw.



Order: 3rd

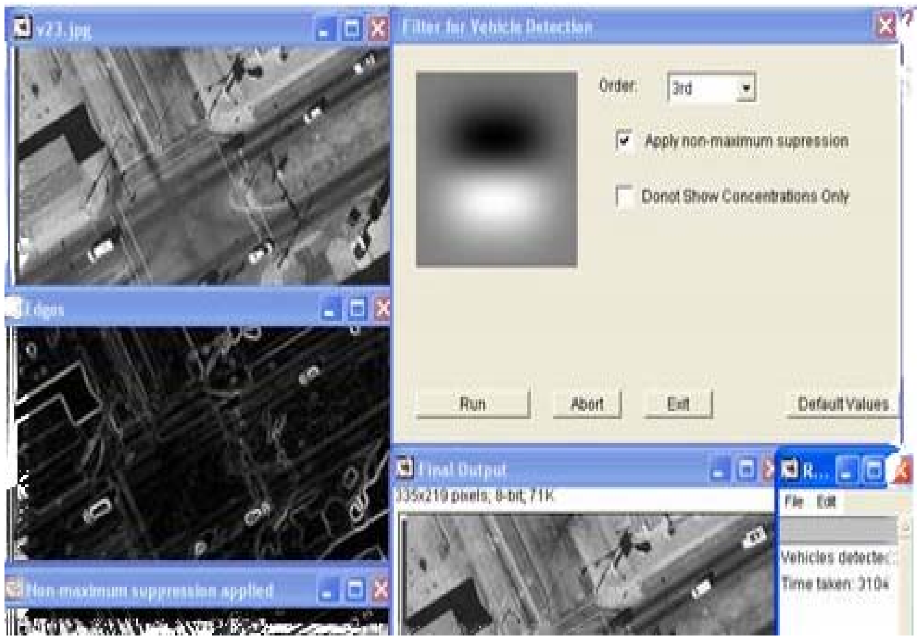
[✓] Apply non-maximum supression

[ ] Donot Show Concentrations

Figure 4.1 User interface

4.6.2 Result Windows

After the Run function is called, four windows (Figure 4.2) appear, three containing the processed images and one portraying the efficiency in terms of time consumed and number of vehicles detected. The processed windows include edge detected image and Non Maximum suppression applied image. The final result window is the replica of the original image with a white rectangle placed over each detected vehicle. Windows menu includes the functions cascade, put behind etc. Image menu includes the options of type, properties, and colors, duplicate, rename, scale, rotate and zoom. Process menu contains the important functions of smooth, sharpen, contrast, noise, shadows etc. Analyze menu includes measure, summarize, label, clear results, set measurements. The result window also shows the number of vehicles detected in a test image.



Order: 3rd

[✓] Apply non-maximum supression

[ ] Donot Show Concentrations Only

335x219 pixels; 8-bit; 71K

Vehicles detectec

Time taken: 310

Figure 4.2 Result Window

The processed windows include edge detected image and Non Maximum suppression applied image. The final result window is the replica of the original image with a white rectangle placed over each detected vehicle. Windows menu includes the functions cascade, put behind etc. Image menu includes the options of type, properties, and colors, duplicate, rename, scale, rotate and zoom.

C h a p t e r 5

5. Methodologies

5.1 Preprocessing for Brightness and Contrast

After aligning the features, there is one remaining major source of variation (apart from intrinsic differences between features). This variation is caused by lighting and camera characteristics, which can result in brightly or poorly lit images, or images with poor contrast. These problems are first addressed by using a simple image processing approach. This preprocessing technique first attempts to equalize the intensity values in across the window. A function is applied, which varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. If the intensity of a pixel *x, y* is *I*(*x, y*), then to fit this linear model parameterized by *a, b, c* to the image:

$$\begin{pmatrix} x & y & z \end{pmatrix} \bullet \begin{pmatrix} a \\ b \\ c \end{pmatrix} = I(x, y) \cdots\cdots\cdots\cdots\cdots (1)$$

The choice of this particular model is somewhat arbitrary. It is useful to be able to represent brightness differences across the image, so a non-constant model is useful. The variation is limited to linear to keep the number of parameters low and allow them to be fit quickly. Collecting together the contributions for all the pixels in the oval window gives an over-constrained matrix equation, which is solved by the pseudo-inverse method. This linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions.

Next, histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases. The algorithm for this step is as follows. First compute the intensity histogram of the window, where each intensity level is given its own bin. This histogram is then converted to a cumulative histogram, in which the value at each bin says how many pixels have intensities less than or equal to the intensity of the bin. The goal is to produce a flat histogram, which is an image in which each pixel intensity occurs an equal number of times. The cumulative histogram of such an image will have that property that the number of pixels with an intensity less than or equal to a given intensity is proportional to that intensity.

Given an arbitrary image, one can produce an image with a linear cumulative histogram by adjusting the pixel intensities. Each intensity will be mapped to the value of the cumulative histogram for that bin. This guarantees that the number of pixels matches the intensity, which is the property required. In practice, it is impossible to get a perfectly flat histogram (for example, the input image might have a constant intensity), so the result is only an approximately flat intensity histogram.

In some parts of this project, only histogram equalization with subtracting the linear model is used. This is done when it is not know which pixels in the input window are likely to be foreground or background, and cannot apply the linear correction to just the face. Instead, just apply the histogram equalization to the whole window, hoping that it will reduce the variability somewhat, without the background pixels having too much effect on the appearance of the features in the foreground.

5.2 Choosing a Smoothing Filter

The smoothing filter can be chosen by taking a model of an edge and then using some set of criteria to choose a filter that gives the best response to that model. It is difficult to pose this problem as a two dimensional problem, because edges in 2D can be curved. Conventionally, the smoothing filter is chosen by formulating a one-dimensional problem, and then using a rotationally symmetric version of the filter in 2D.

The one-dimensional filter must be obtained from a model of an edge. The usual model is a step function of unknown height, in the presence of stationary additive Gaussian noise: where

$$edge(x) = AU(x) + n(x) \cdots\cdots\cdots\cdots (2)$$

$$U(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases} \cdots\cdots\cdots\cdots\cdots\cdots (3)$$

(The value of U(0) is irrelevant for this purpose). A is usually referred to as the contrast of the edge. In the 1D problem, finding the gradient magnitude is same as finding the square of the derivative response. For this reason, one usually seeks a derivative estimation filter rather than a smoothing filter (which can then be reconstructed by integrating the derivative estimation filter). Canny established the practice of choosing a derivative estimation filter by using the continuous model to optimize a combination of three criteria. Signal to noise ratio: the filter should respond more strongly to the edge at x = 0 than to noise. Localization: the filter response should reach a maximum very close to x = 0. Low false positives: there should be only one maximum of the response in a reasonable neighborhood of x = 0. It is difficult to pose this problem as a two dimensional problem, because edges in 2D can be curved.
Once a continuous filter has been found, it is discredited. The criteria can be combined in a variety of ways, yielding a variety of somewhat different filters.

It is a remarkable fact that the optimal smoothing filters that are derived by most combinations of these criteria tend to look a great deal like Gaussians — this is intuitively reasonable, as the smoothing filter must place strong weight on center pixels and less weight on distant pixels, rather like a Gaussian. In practice, optimal smoothing filters are usually replaced by a Gaussian, with no particularly important degradation in performance.

5.3 Derivative of Gaussian Filters

Smoothing an image and then differentiating it is the same as convolving it with the derivative of a smoothing kernel. This fact is most easily seen by thinking about continuous convolution. Firstly, differentiation is linear and shift invariant. This means that there is some kernel — it dodges the question of what it looks like — that differentiates. That is, given a function I(x,y)

$$\frac{\partial I}{\partial x} = K\frac{\partial}{\partial x} * I \cdots\cdots\cdots\cdots (4)$$

Now the derivative of a smoothed function is required. The convolution kernel for the smoothing is written as S. Recalling that convolution is associative,

$$\left(K\frac{\partial}{\partial x} * (S * I)\right) = \left(K\frac{\partial}{\partial x} * S\right) * I = \left(\frac{\partial S}{\partial x}\right) * I \cdots\cdots\cdots\cdots (5)$$

This fact appears in its most commonly used form when the smoothing function is a Gaussian;

$$\frac{\partial(G\sigma * I)}{\partial x} = \left(\frac{\partial G\sigma}{\partial x}\right) * I \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$

i.e. only convolve with the derivative of the Gaussian, rather than convolve and then differentiate. A similar remark applies to the Laplacian. The Laplacian of a function in 2D is defined as:

$$\left(\nabla^2 f\right)\bullet(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7)$$

Again, because convolution is associative,

$$\left(K\nabla^2 * \left(G\sigma * I\right)\right) = \left(K\nabla^2 * G\sigma\right) * I = \left(\nabla^2 G\sigma\right) * I \cdots\cdots\cdots\cdots (8)$$

This practice results in much smaller noise responses from the derivative estimates. Non-maximum suppression obtains points where the gradient magnitude is at a maximum along the direction of the gradient. The Figure 5.1 shows how to reconstruct the gradient magnitude. The dots are the pixel grid. At pixel q, attempting to determine whether the gradient is at a maximum; the gradient direction through q does not pass through any convenient pixels in the forward or backward direction, so interpolation is done to obtain the values of the gradient magnitude at p and r; if the value at q is larger than both, q is an edge point. Typically, the magnitude values are reconstructed with a linear interpolate, which in this case would use the pixels to the left and right of p and r respectively to interpolate values at those points. On the right, it sketches how to find candidates for the next edge point, given that q is an edge point; an appropriate search direction is perpendicular to the gradient, so that points s and t should be considered for the next edge point. Notice that, in principle, it is not needed to restrict to pixel points on the image grid, because it is known where the predicted position lies between s and t, so that it can again interpolate to obtain gradient values for points off the grid. Typically, the magnitude values are reconstructed with a linear interpolate, which in this case would use the pixels to the left and right of p and r respectively to interpolate values at those points. This reconstruction is followed by tracing of the edges found out by the algorithm. Thresholding of the image then brings out an image in which edges are visible clearly. Notice that, in principle, it is not needed to restrict to pixel points on the image grid, because it is known where the predicted position lies between s and t, so that it can again interpolate to obtain gradient values for points off the grid.

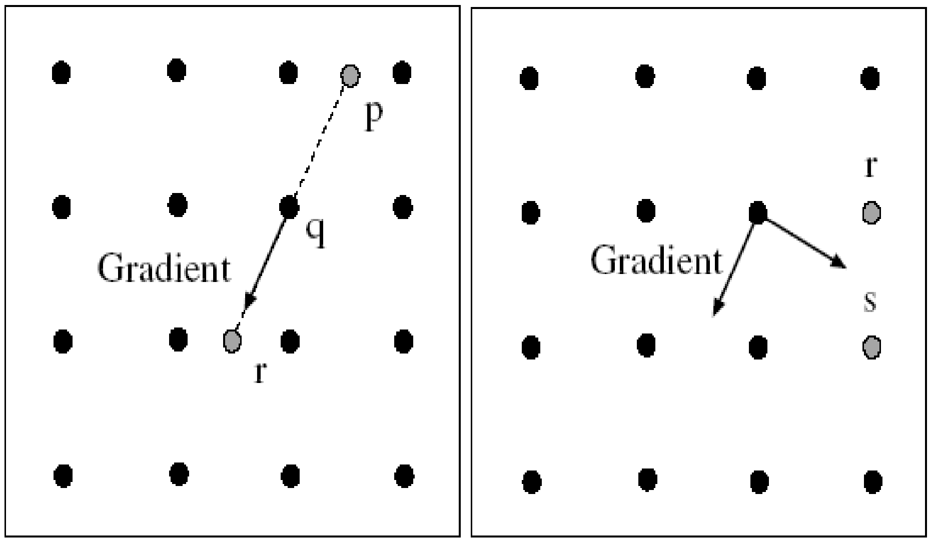
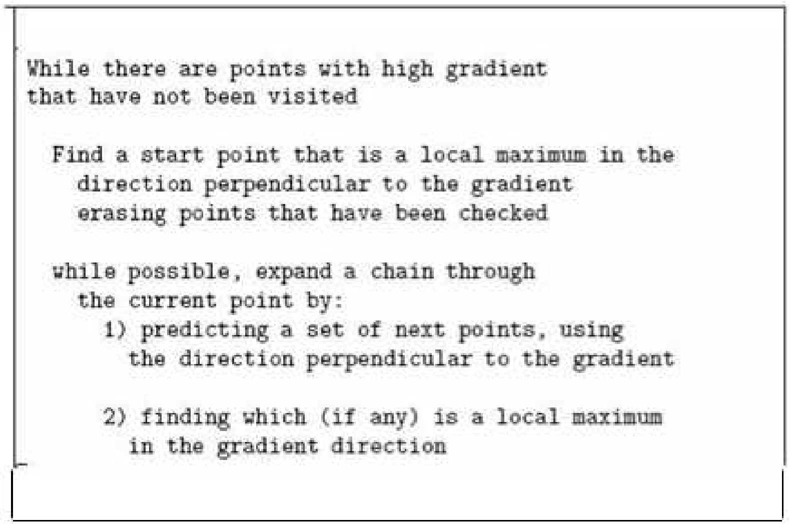


Figure 5.1 Reconstruction of gradient magnitude

5.4 Identifying Edge Points from Filter Outputs

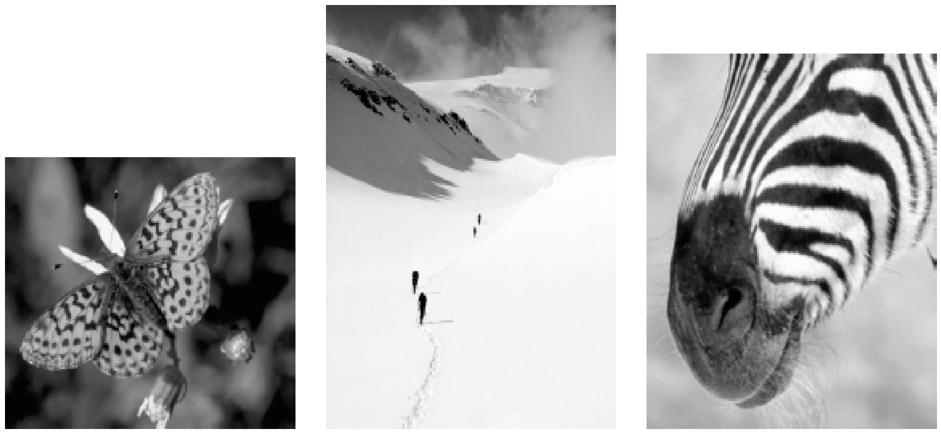
Given estimates of gradient magnitude it is indented to obtain edge points. Again, there is clearly no objective definition, and proceed by reasonable intuition. The gradient magnitude can be thought of as a chain of low hills. Marking local extrema would mark isolated points—the hilltops in the analogy. A better criterion is to slice the gradient magnitude along the gradient direction — which should be perpendicular to the edge — and mark the points along the slice where the magnitude is maximal. This would get a chain of points along the crown of the hills in the chain; the process is called non-maximum suppression. Typically, it is expected that edge points to occur along curve-like chains. The significant steps in non maximum suppression are: Determining whether a given point is an edge point; and, if it is, finding the next edge point. Once these steps are understood, it is easy to enumerate all edge chains. First edge point is found, marked, expanded all chains through that point exhaustively, marking all points along those chains, and continue to do this for all unmarked edge points.



```
While there are points with high gradient
that have not been visited

  Find a start point that is a local maximum in the
    direction perpendicular to the gradient
    erasing points that have been checked

  while possible, expand a chain through
    the current point by:
      1) predicting a set of next points, using
        the direction perpendicular to the gradient

      2) finding which (if any) is a local maximum
        in the gradient direction
```

Algorithm 5.1: Non-maximum suppression

The two main steps are simple. For the moment, assume that edges are to be marked at pixel locations (rather than, say, at some finer subdivision of the pixel grid). It can be determined whether the gradient magnitude is maximal at any pixel by comparing it with values at points some way backwards and forwards along the gradient direction. This is a function of distance along the gradient; typically it steps forward to the next row (or column) of pixels and backwards to the previous to determine whether the magnitude at the pixel is larger. The gradient direction does not usually pass through the next pixel, so we must interpolate to determine the value of the gradient magnitude at the points we are interested in; a linear interpolate is usual. A better criterion is to slice the gradient magnitude along the gradient direction which should be perpendicular to the edge and mark the points along the slice where the magnitude is maximal. This would get a chain of points along the crown of the hills in the chain; the process is called non-maximum suppression.



(i) (ii) (iii)

Figure 5.2 Image Examples (i) Butterfly on a blurred background (ii) Strong contrast between snow and objects (iii) Zebra with fine scale details

Here three images (Figure 5.2) are used to illustrate properties of a gradient based edge detector. The butterfly is on a blurred background; there is strong contrast between the figures on the snow and the background; and the zebra's nose has fine scale detail — its whiskers — as well as coarse scale detail. If the pixel turns out to be an edge point, the next edge point in the curve can be guessed by taking a step perpendicular to the gradient. This step will not, in general, end on a pixel; a natural strategy is to look at the neighboring pixels that lie close to that direction. This approach leads to a set of curves that can be represented by rendering them in black on a white background, as in Figure 5.3. Edge points marked on the pixel grid for the three images shown in Figure 5.2. The top row shows edge points obtained using a Gaussian smoothing filter at σ one pixel, and the center and bottom rows show points obtained using a smoothing filter at σ four pixels. For the top two cases, Gradient magnitude has been tested against a high threshold to determine whether a point is an edge point or not; for the bottom row, gradient magnitude was tested against a low threshold.

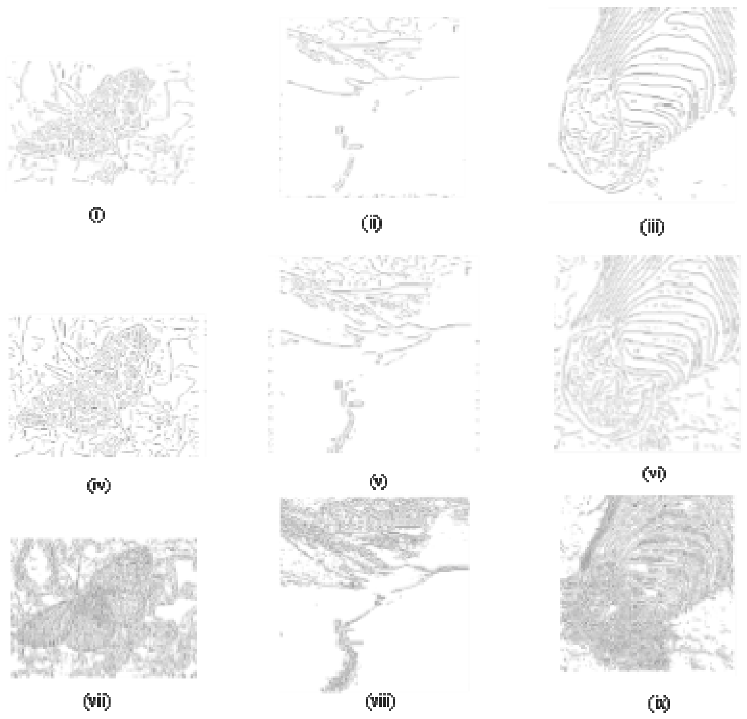


Figure 5.3. Edge Detection with different Gradient magnitude for smoothing (i)(ii)(iii) Gaussian smoothing filter at σ = 1 pixel,(iv)(v)(vi) Gaussian smoothing filter at σ = 2 pixel,(vii)(viii)(ix) Gaussian smoothing filter at σ = 2 pixel.

At a fine scale, fine detail at high contrast generates edge points, which disappear at the coarser scale — for example, the zebra's whiskers disappear. When the threshold is high, curves of edge points are often broken because the gradient magnitude dips below the threshold; for the low threshold, a variety of new edge points of dubious significance are introduced.

5.5 Steerable Filters

To cut down on the computational load, we select our detector within the class of steer able filters introduced by Freeman and Adelson. These filters can be rotated very efficiently by taking a suitable linear combination of a small number of filters. Specifically, we consider templates of the form

$$h(x,y)=\sum_{k=1}^{M}\sum_{i=0}^{k}\alpha_{k,i}\frac{\partial^{k-i}}{\partial x^{k-i}}\frac{\partial^{i}}{\partial y^{i}}g(x,y)\cdots\cdots(9)$$

where $g(x,y)$ is an arbitrary isotropic window function. It is called an Mth order detector. Once the f $_{ki}$ (x,y) are available, f(x)* h(Rx) can be evaluated very efficiently via a weighted sum with its coefficients that are trigonometric polynomials of Since the number of partial differentials in (5) for a general Mth order template is M(M+3)/2, h(x) is steer able in terms of as many individual separable functions. Using some simplification, it can be shown that such a general h(x) can also be rotated using 2M + 1 non separable filters.

$$f(x)\bullet h(\Re_\theta x)=\sum_{k=1}^{M}\sum_{i=0}^{k}\beta_{k,i}(\theta)f_{k,i}(x)\cdots\cdots(10)$$

$$f_{k,i}(x,y)=f(x,y)*\underbrace{\left(\sum_{k=1}^{M}\sum_{i=0}^{k}\frac{\partial^{k-i}}{\partial x^{k-i}}\frac{\partial^{i}}{\partial y^{i}}g(x,y)\right)}_{g(x,y)}\cdots\cdots(11)$$

$$\beta_{k,i}(\theta)=\left(\sum_{i=0}^{k}\alpha_{k,i}\sum_{\ell m\in\delta(k,j))}\binom{k-j}{i}\binom{j}{m}(-1)\cos(\theta)\sin(\theta)\right)\cdots\cdots(12)$$

A case of special interest corresponds to g(x) being the Gaussian; indeed, the Gaussian is optimally localized in the sense of the uncertainty principle and the corresponding filters in (6) are all separable. Interestingly, the Gaussian family is equivalent to the class of moment filters (polynomials multiplied by Gaussian window), but the filters are not identical.

5.6 Design of Steer able Filters for Feature Detection

As already observed by Freeman and Adelson, the widely used Canny edge detection algorithm can be reinterpreted in terms of steer able filters. This algorithm involves the computation of the gradient-magnitude of the Gaussian smoothed image. The direction of the gradient gives the orientation of the edge. Mathematically,

$$\theta^* = \arctan\left(\frac{(f*g)_y}{(f*g)_x}\right)\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(13)$$

$$r^* = \sqrt{((f*g)_x)}^2 + \sqrt{((f*g))_y}^2\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(14)$$

$$\theta^*(x) = \arg\max\left(f(x)*g_x(R_\theta x)\right)\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(15)$$

$$\theta^*(x) = \arg\max\left(f(y)*g_y(R_\theta x)\right)\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(16)$$

$$\theta^*(x) = \arg\max\left(f*(g_x\cos(\theta)+g_x\sin(\theta))\right)\cdots\cdots\cdots\cdots\cdots(17)$$

$$\theta^*(x) = \arg\max\left(f*(g_x\cos(\theta)-g_x\sin(\theta))\right)\cdots\cdots\cdots\cdots\cdots(18)$$

Here, the steer ability of gx from (5) is used. To compute the maximum of the above expression, it sets the differential of (11) with respect to zero.

The widely-used contour extraction algorithm has three steps: feature detection, non maximum suppression, and threshold. In this section, a general strategy for the design of steer able filters for feature detection is presented, while keeping in mind the subsequent steps. It proposes a criterion similar to that of Canny and analytically derives the optimal filter—or, equivalently, the optimal weights—within the particular class of steer able functions specified by (4). The widely-used contour extraction algorithm has three steps: feature detection, non maximum suppression, and threshold. In this section, a general strategy for the design of steer able filters for feature detection is presented, while keeping in mind the subsequent steps.

5.7 Optimality Criterion

A review of Canny's criterion and modifying it slightly to make it analytically optimized. To derive the optimal 2D operator, it is assumed that the feature (edge/ridge) is oriented in some direction (say, along the x axis) and derives an optimal operator for its detection. As the operator is Rotation-steer able by construction, its optimality properties will be independent of the feature orientation. The three different terms in Canny's criterion are as follows.

5.7.1 Signal-to-Noise Ratio

The key term in the criterion is the signal-to-noise ratio. The response of a filter to a particular signal (e.g., an idealized edge) centered at the origin is given by

$$s = \int_{\Re^2} f_o(x,y)h(-x,-y)dxdy \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (19)$$

S is given by the height of the response at its maximum. If the input is corrupted by additive white noise of unit variance, then the variance of the noise at the output is given by the energy of the filter:

$$Noise = \int_{\Re^2} |h(x,y)|^2 dxdy \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (20)$$

It is desired to have a high value of S for a given value of Noise; S2 Noise is the amplification of the desired feature provided by the detector.

5.7.2 Localization

The detection stage is preceded by non maximum suppression. The estimated feature position corresponds to the location of the local maximum of the response in the direction orthogonal to the feature boundary (y axis in our case). The presence of noise can cause an undesirable shift in the estimated feature location. The direct extension of Canny's expression for the shift-variance (due to white noise of unit variance) to 2D gives

$$E\left[(\Delta y)^2\right] = \frac{\int_{\Re^2} \left|h_y(x,y)\right|^2 dxdy}{\left|\int_{\Re^2} f_o(x,y) h_{yy}(-x,-y) dxdy\right|^2} \cdots\cdots\cdots\cdots\cdots\cdots (21)$$

Canny has proposed to maximize the reciprocal of this term. The numerator of (19) is a normalization term which will be small automatically if the impulse response of the filter is smooth along the y axis (low norm for the derivative). Since the approach is imposing this type of smoothness constraint elsewhere via an additional regularization term, it is not necessary to optimize this term here, which also keeps the effects well separated. Therefore, it is proposed to maximize the second derivative of the response, orthogonal to the boundary, at the origin

$$Loc = -\frac{d^2}{dy^2}(f_o * h) = -\int_{\Re^2} f_o(x,y) h_{yy}(-x,-y) dxdy \cdots\cdots\cdots\cdots\cdots\cdots (22)$$

which is the square-root of the denominator in (20). The above expression is ensured to be positive because the second derivative of the response is negative at the maximum (assuming $S > 0$). Note that the new localization term is a measure of the width of the peak. The drift in position of the maximum due to noise will decrease as the response becomes sharper. This work is neglecting the effect of neighboring features in deriving the localization term.

5.7.3 Elimination of False Oscillations

Canny observed that when the criterion is optimized only with the SNR and the localization constraint, the optimal operator has a high bandwidth; the response will be oscillatory and, hence, have many false maxima. In 2D, it is desired that the response be relatively free of oscillations orthogonal to the feature boundary. This can be achieved by penalizing the term:

$$R_x = \int_{R^2} \left|h_{yy}(x,y)\right|^2 dxdy \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (23)$$

Note that this term is the numerator of the expression for the mean distance between zero crossings proposed by Canny. It is a thin-plate spline-like regularization which is a standard technique to constrain a solution to be smooth (low bandwidth). The thresholding step is easier if the response is flat along the boundary. The oscillation of the response along the boundary (x axis) can be minimized by penalizing above equation. These terms will force the filter to be smooth making the response is less oscillatory, thus resulting in less false detection.

Chapter 6

6. Results and Analysis

6.1 Introduction

After tuning the edge detection operator to optimally detect the vehicle edges, testing was carried out on the image data set to verify the efficacy of the technique. Testing was carried out on Pentium IV 1.6 GHz machine with 256 MB RAM, with Microsoft Windows XP (SP2). The dataset, as already described was selected to test the system for the objectives already stated. The results obtained are described image wise:

6.1.1 **Test Image ATR**

This image contained a vertical photo of a tank with no other objects in the surroundings. This was the simplest image. The edge response was good (Figure 6.1), the rectangular profile of the tank was clearly visible. Hence, results as shown in Figure 6.2, the system was able to correctly classify it.

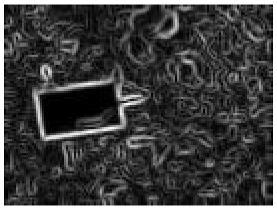


Figure 6.1 Edge image of ATR

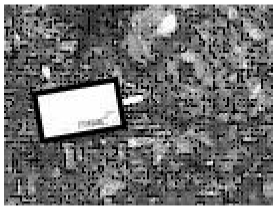


Figure 6.2 Final Output of ATR

The image contained only one vehicle and the rectangular profile was successfully detected as the image was noise free and had no other features. The concentration detection condition had no impact on the results because only one vehicle was present in the image.

6.1.2 Test Image Img83

This image depicted a densely populated area. Large number of buildings, some of which were giving good rectangular profile. Moreover the presence of trees was also a hindrance in the detection of vehicles. The mentioned problems did pose a challenge in the edge detection phase, as can be seen from the edge image (Figure 6.3). There were 20 vehicles present in the image; the system classified 19 objects as vehicles (Figure 6.4), with 13 false positives and 14 false negatives. Here it is pertinent to mention that these results are with the concentration condition applied. Without the concentration condition (Figure 6.5) there were 6 correct classifications and 27 errors. The errors are fairly high, but can be justified on the ground that the system is designed to work without any image context information and is assumed to work in cross country also. The non use of road and pavement alignment information though cause this large number of errors, but at the same time proves the efficiency of the system where it has successfully, classified vehicles from other similar objects. The system objective as has been mentioned before also is tom detect the vehicles in cross country terrain, therefore in this image the results can be accepted. This image was included in the test images to check its efficacy in urban area images and find out any areas where further amendments can make the system more efficient. The chimneys present on the roof tops specially give out the edge shape similar to that of the vehicles as can be seen in Figure 6.3.

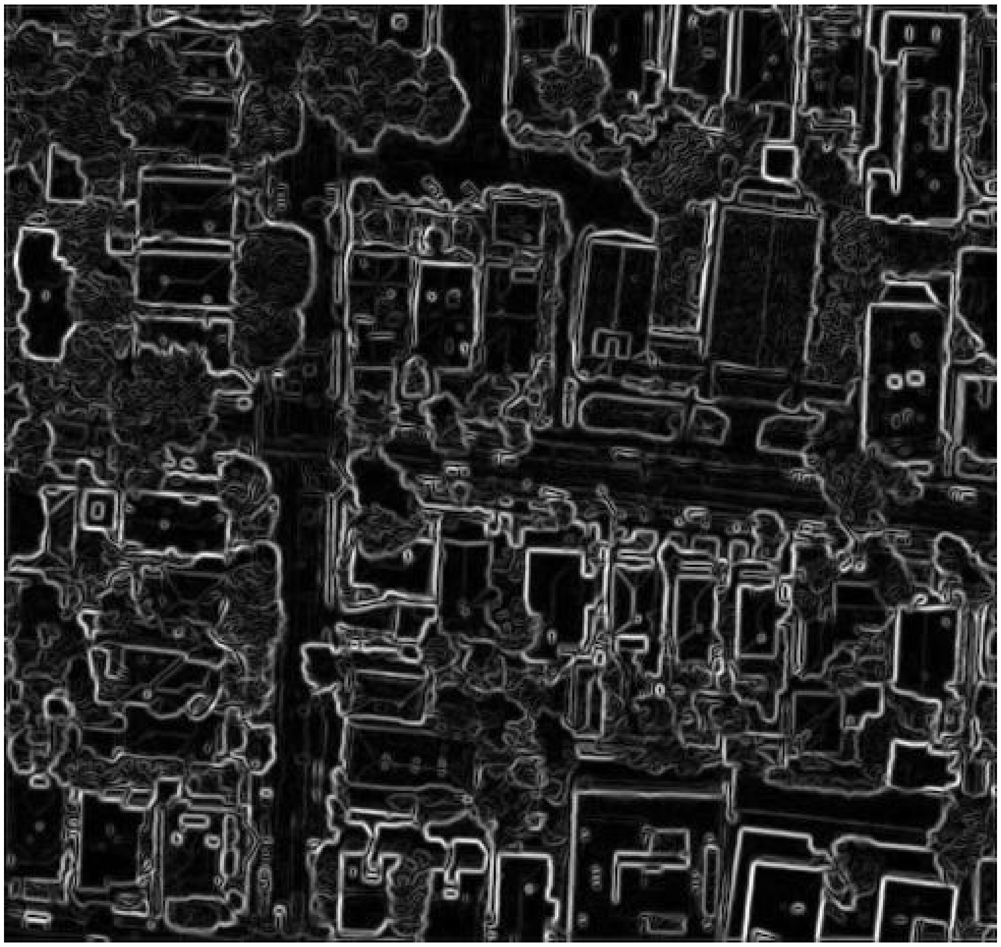


Figure 6.3 Edge image of Img83

The errors are fairly high, but can be justified on the ground that the system is designed to work without any image context information and is assumed to work in cross country also. The non use of road and pavement alignment information though cause this large number of errors, but at the same time proves the efficiency of the system where it has successfully, classified vehicles from other similar objects. The system objective as has been mentioned before also is tom detect the vehicles in cross country terrain, therefore in this image the results can be accepted.

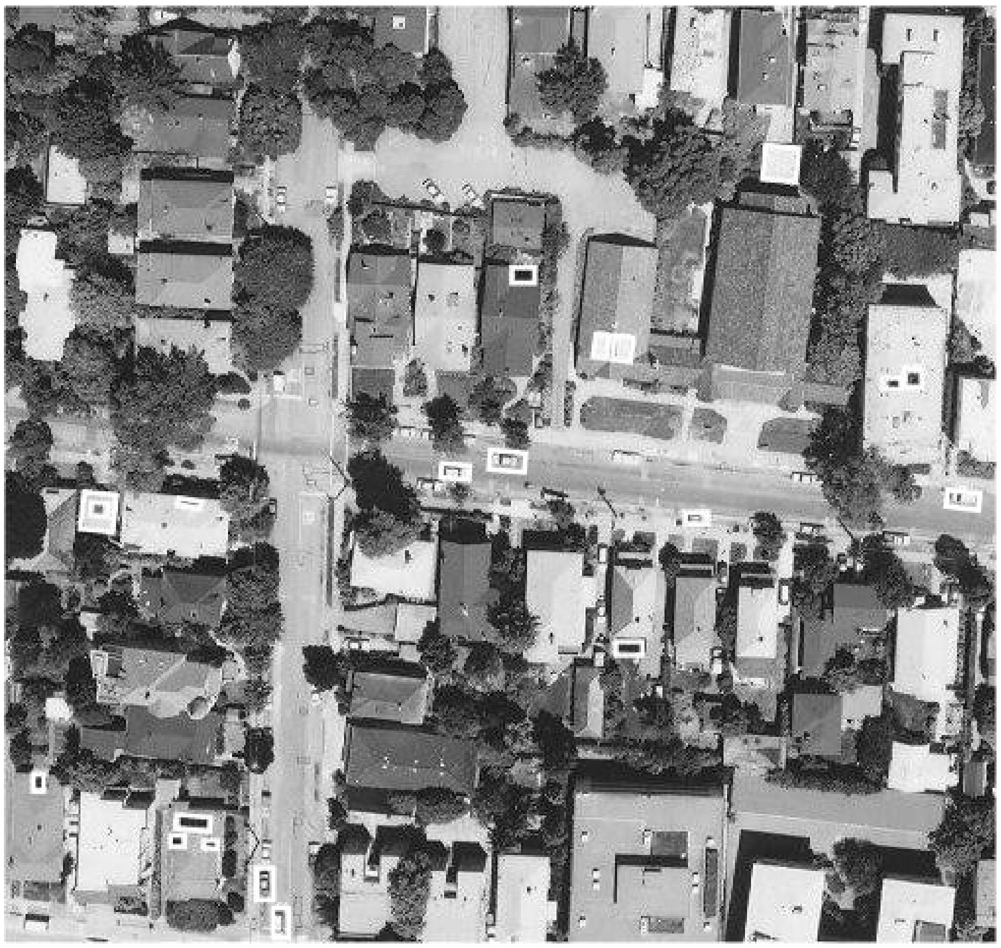


Figure 6.4 Final Output of Img83 with concentration

The use of concentration condition makes the system prone to more errors in an urban area as normally there are places in urban areas where vehicles are mostly dispersed and the concentration of vehicles is not found usually. At places the wind screens of the vehicles divide the edge image of the vehicle in three different parts thus the vehicle is no longer visible as a rectangular entity in the edge image. This phenomenon also gives rise to false negative responses. It is difficult to predict the occurrence of this phenomenon in advance and then incorporate it in the classification as a large number of factors are involved.

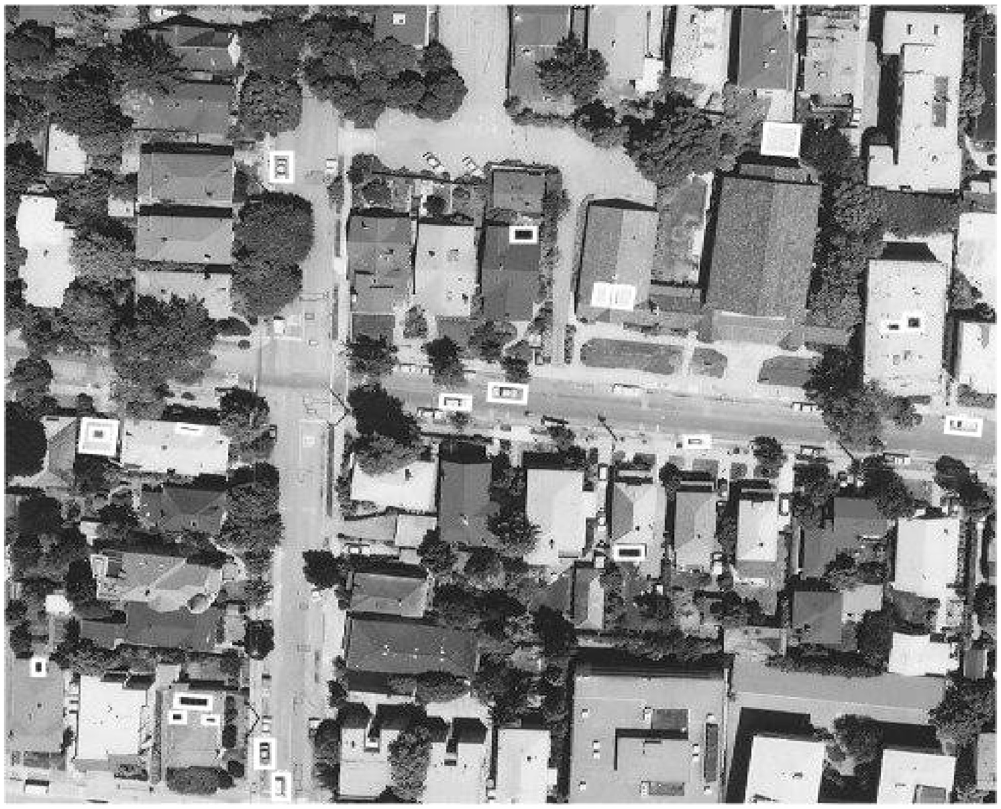


Figure 6.5 Final Output of Img83 without concentration

6.1.3 Test Image v2

This image contains a curved road segment; predominantly it is a dark image because of presence of a large water body. A bridge with five vehicles on it is dividing the image in two halves. Built up area with trees shadowing some vehicles is present. The edge response (Figure 6.6) of the image was good, as despite the presence of the buildings and trees the number of correctly classified vehicles with concentration condition (Figure 6.7) was 4 out of 9 and no false positives. The errors were 5 in the form of false negatives only, which were caused by the use of concentration condition as it prevents detection of isolated vehicles. Without the concentration condition (Figure 6.8) the system was able to detect all the 9 vehicles correctly. 4 false positives were reported as the buildings were giving similar edge profile as the vehicles.

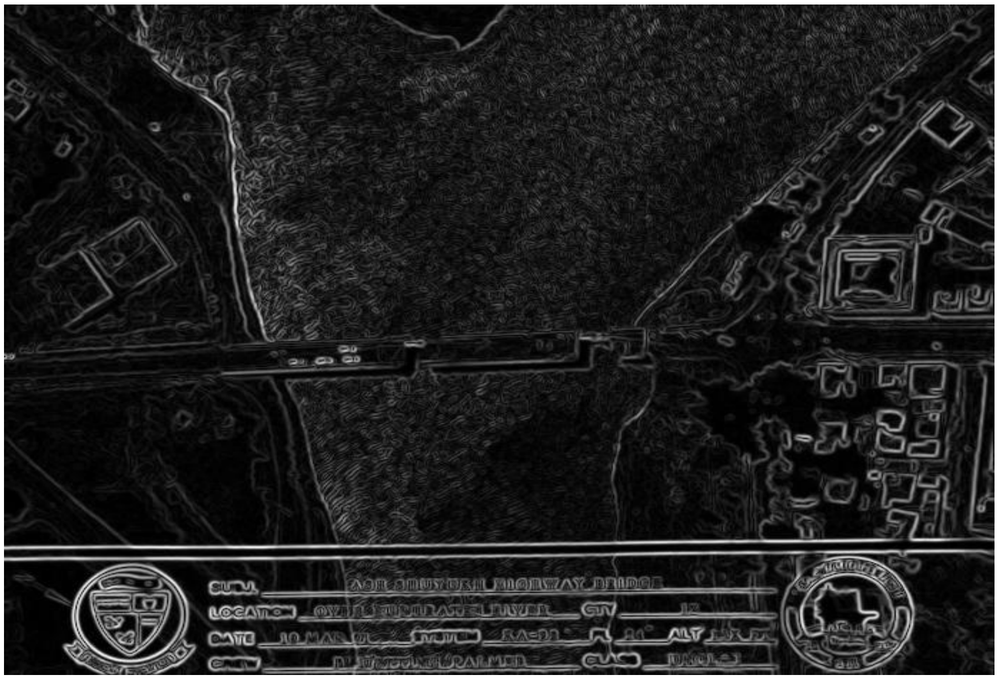


Figure 6.6 Edge image of v2

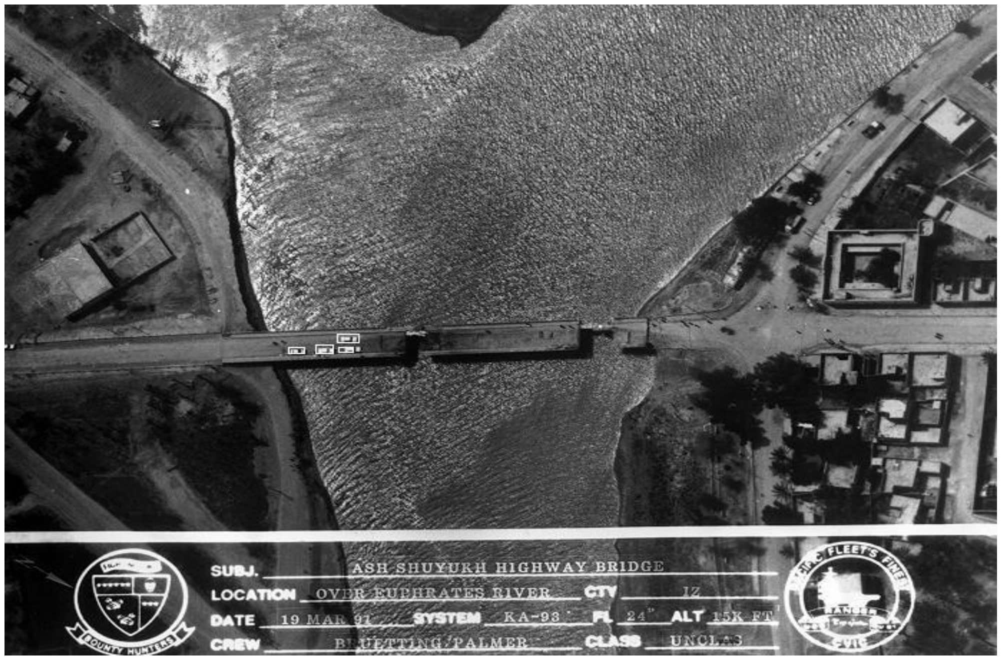

SUBJ. ASH SHUYUKH HIGHWAY BRIDGE
LOCATION OVER EUPHRATES RIVER CTY IZ
DATE 19 MAR 91 SYSTEM KA-93 FL 24" ALT 15K FT
CREW BLUETTING/PALMER CLASS UNCLAS
BOUNTY HUNTERS
PACIFIC FLEET'S FINEST

Figure 6.7 Final Output of v2 with concentration

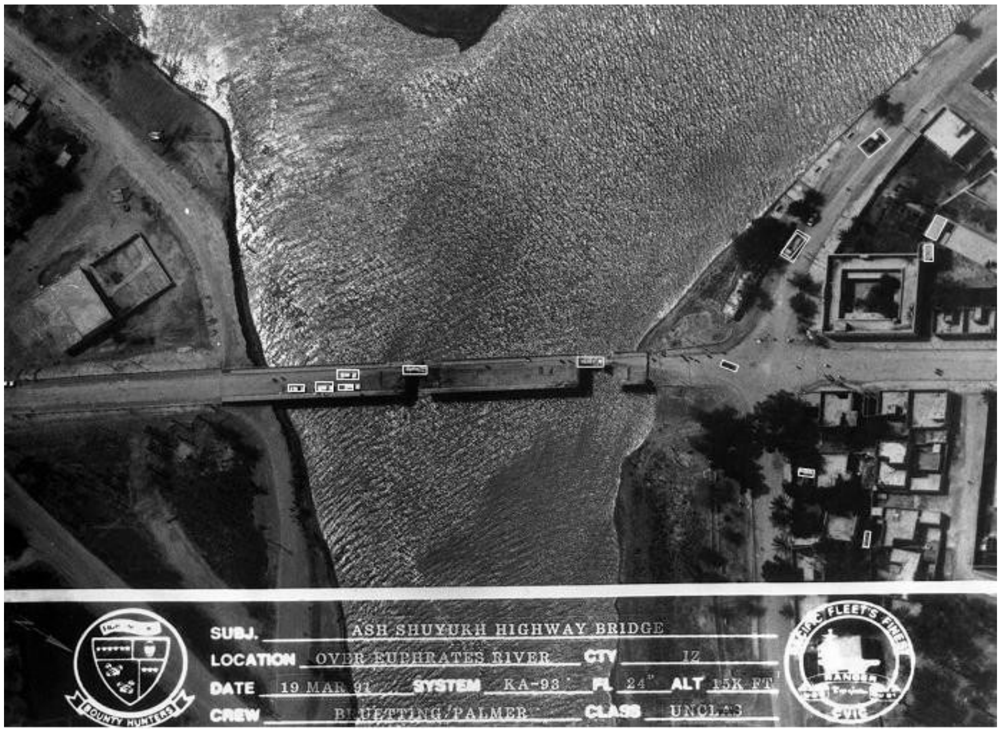

SUBJ. ASH SHUYUKH HIGHWAY BRIDGE
LOCATION OVER EUPHRATES RIVER CTY IZ
DATE 19 MAR 91 SYSTEM KA-93 FL 24" ALT 15K FT
CREW BLUETTING/PALMER CLASS UNCLAS
BOUNTY HUNTERS
FLEET'S FINEST
RANGER

Figure 6.8 Final Output of v2 without concentration

6.1.4 Test Image v10

This image depicted the Gulf war retreat of the Iraqi forces. There were 64 vehicles present many of them burnt and even difficult to classify as vehicles with the naked eye. A substantial portion of the image was dark to the extent of giving no information for the classification. The edge response (Figure 6.9) caused a lot of vehicles to lose their rectangular shape. The results were fair as with concentration condition (Figure 6.10) 23 vehicles were correctly classified. High number of false negatives was caused by partial occlusion of certain vehicles by the shadows and the burnt ground. Furthermore, a large number of vehicles were overturned and piled over each other thus causing the loss of edge information leading to correct detection. This problem can be addressed by the use of intelligent technique to offset the effects of occlusion.

Better results were obtained without the concentration condition (Figure 6.11) with 41 correct classifications.

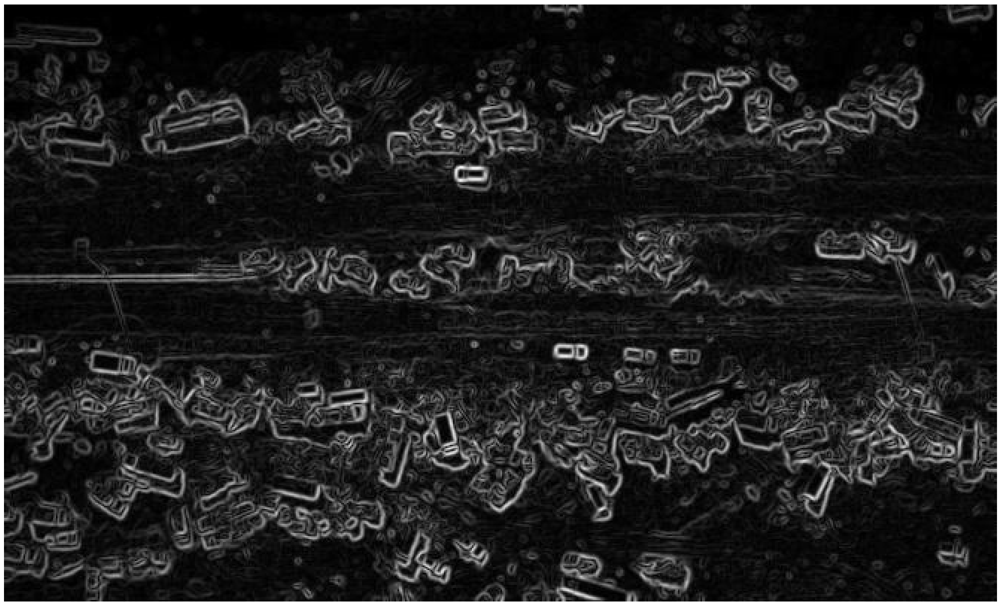


Figure 6.9 Edge image of v10

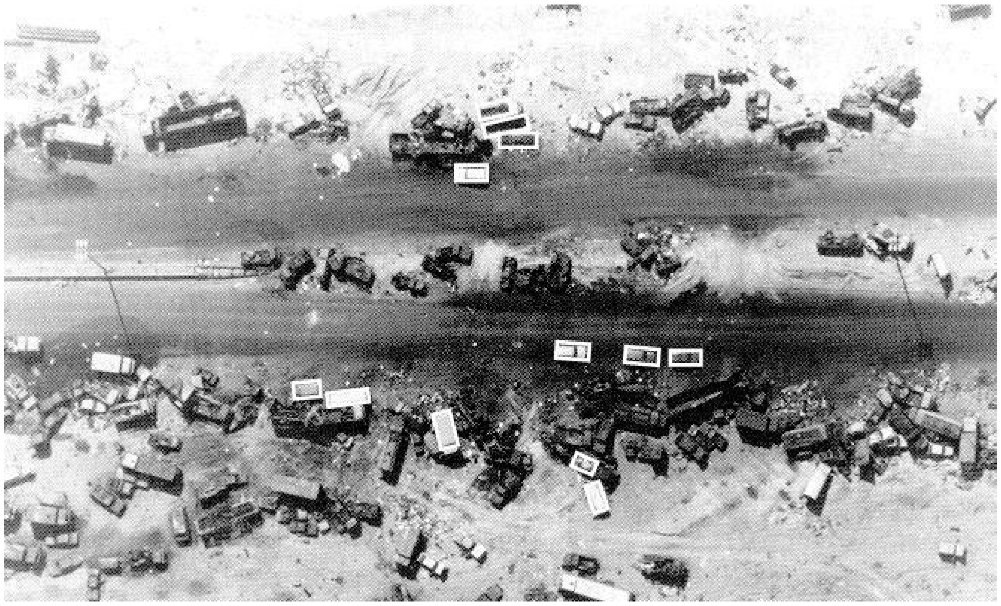


Figure 6.10 Final Output of v10 with concentration

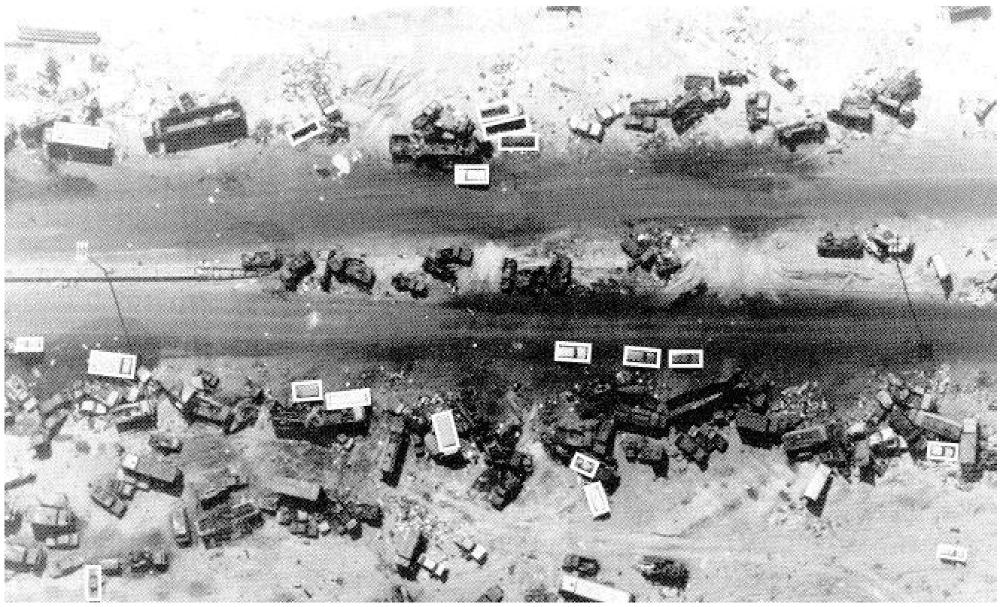


Figure 6.11 Final Output of v10 without concentration

6.1.5 Test Image v18

This image shows various vehicles dispersed in the desert with few barracks. The interesting part is that even visually it is difficult to distinguish between the two. The edge image (Figure 6.12) proves this vulnerability of the system and hence diminishes further the differences. Of the 5 vehicles present in the image 2 were classified correctly with the concentration condition (Figure 6.13) as the dispersal distance between the vehicles was larger than the concentration parameter. Without the concentration condition (Figure 6.14) the results improved to give 4 correct classifications. An interesting observation was again the classification of North marker present on the image as vehicle. This brings forth a conclusion that if somehow the height of the image and subsequently expected size of the vehicles in the image can be provided/calculated by the system, this will result in a substantive enhancement in the performance.

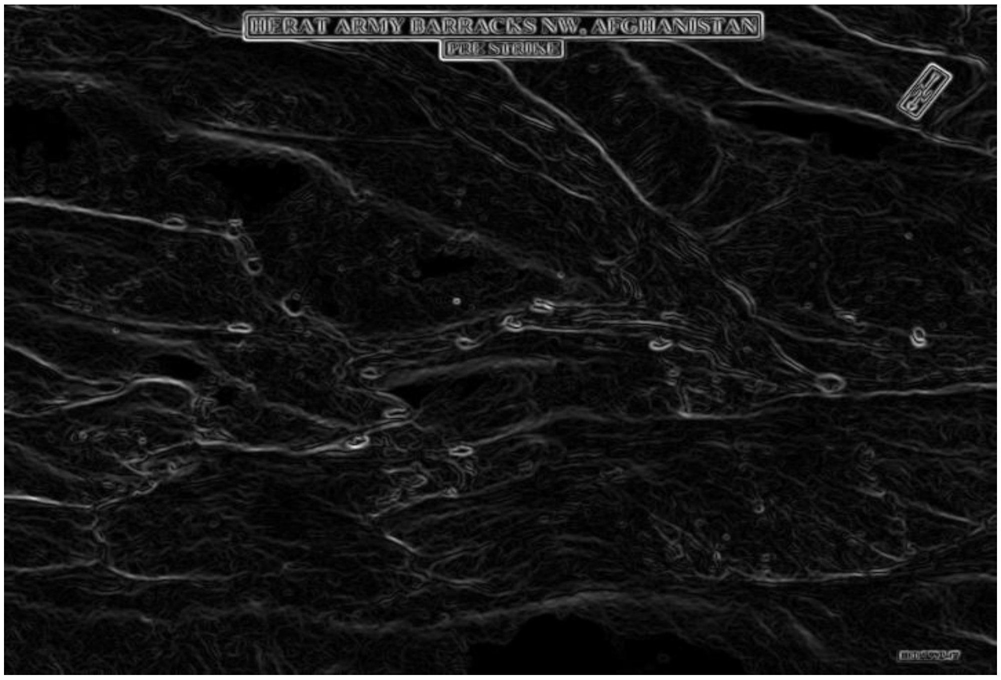


Figure 6.12 Edge image of v18

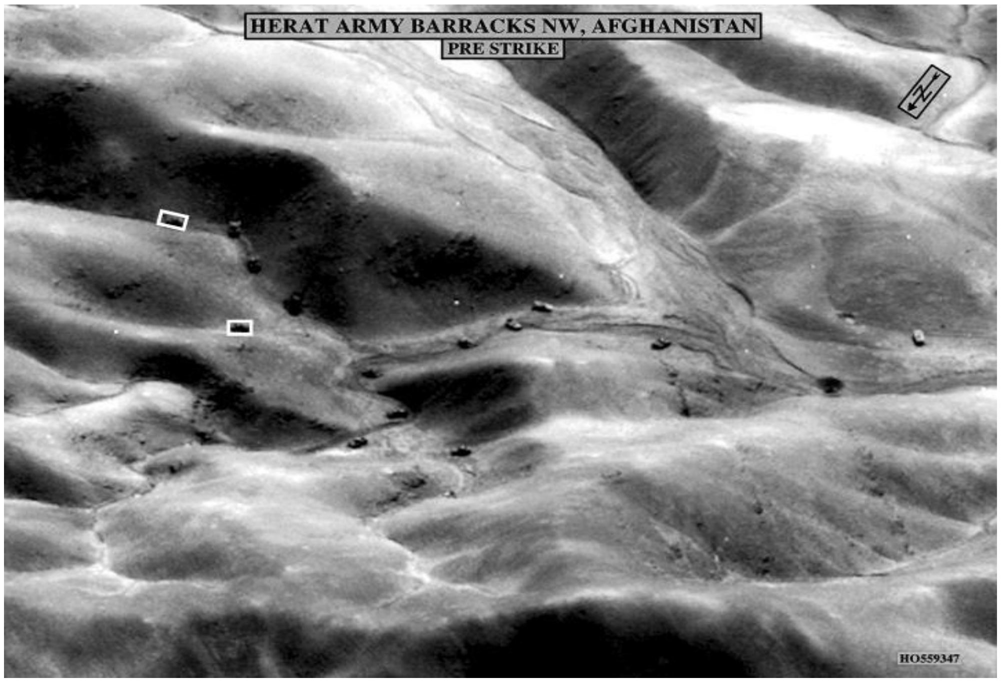


Figure 6.13 Final Output of v18 with concentration

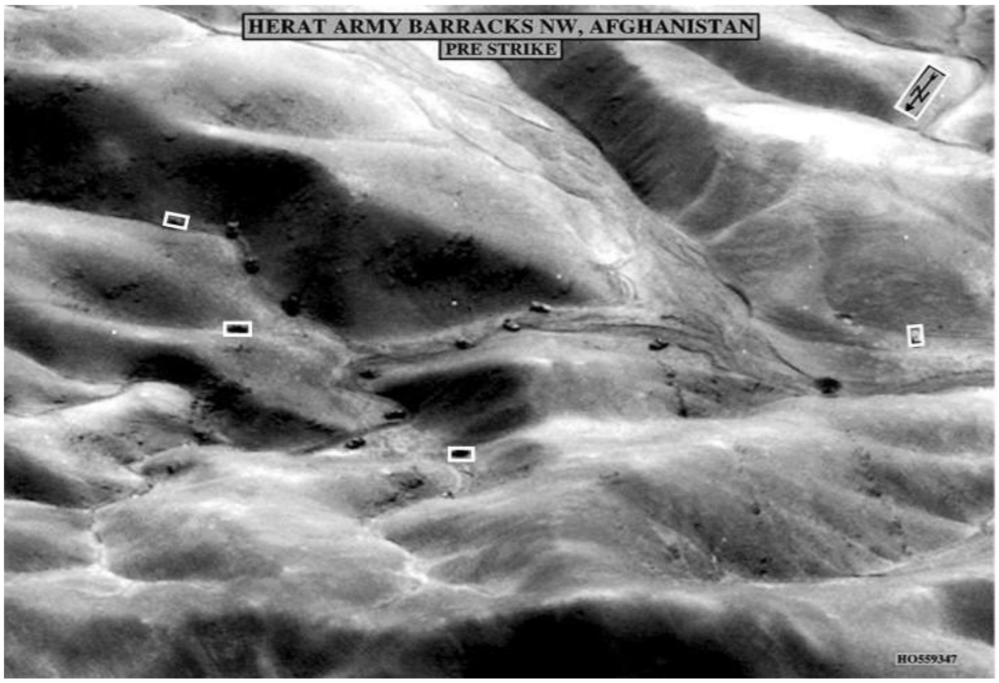


Figure 6.14 Final Output of v18 without concentration

6.1.6 Test Image v22

In this image all vehicles are on the road. Few buildings with rectangular profiles are also present. Some vehicles are merging with the road texture. Vehicles are of varying sizes and are fairly dispersed**.** There are 9 vehicles in the image. The edge image (Figure 6.15) presents another challenge to the system in the form of shadows of the vehicles, which distort the information and thus cause a misclassification. With concentration condition (Figure 6.16) three vehicles were detected successfully and without concentration condition (Figure 6.17) this number improved to 5. At the same time the number of false classifications remained constant thus the number of errors remained static.
The problem of shadows can be over come by applying an operator which can offset the effect of shadows in the image. It is possible if the information about the position of sun or the time of the image exposure is known. This again

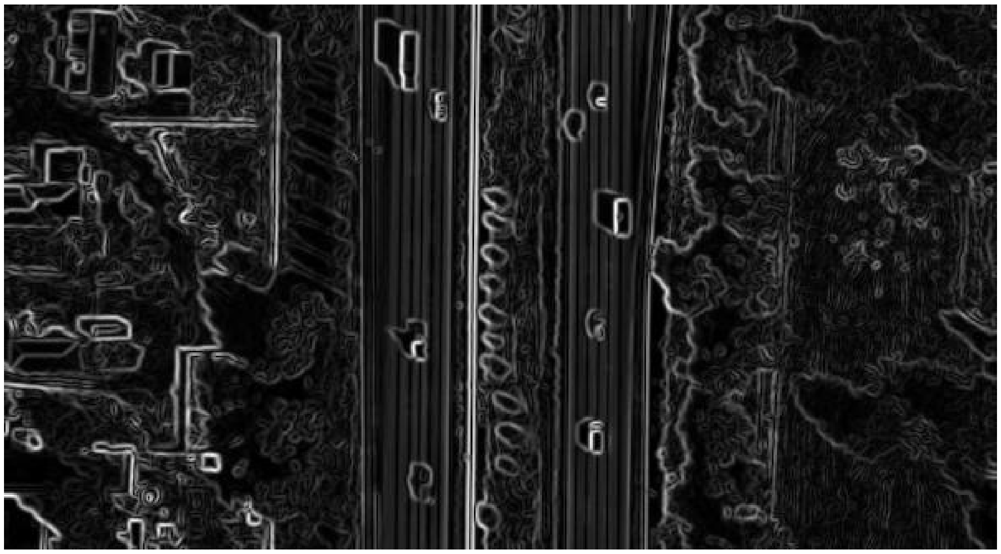


Figure 6.15 Edge image of v22

implies that the image data set should be specialized one and available on demand. Alternatively a method to extract the above information from the image be devised, which is out of scope of the project.

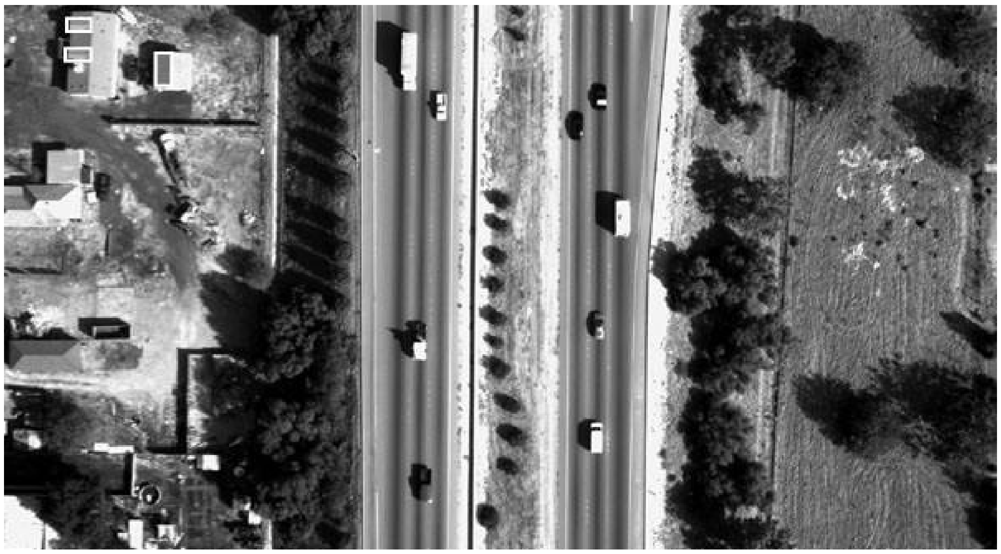


Figure 6.16 Final Output of v22 with concentration

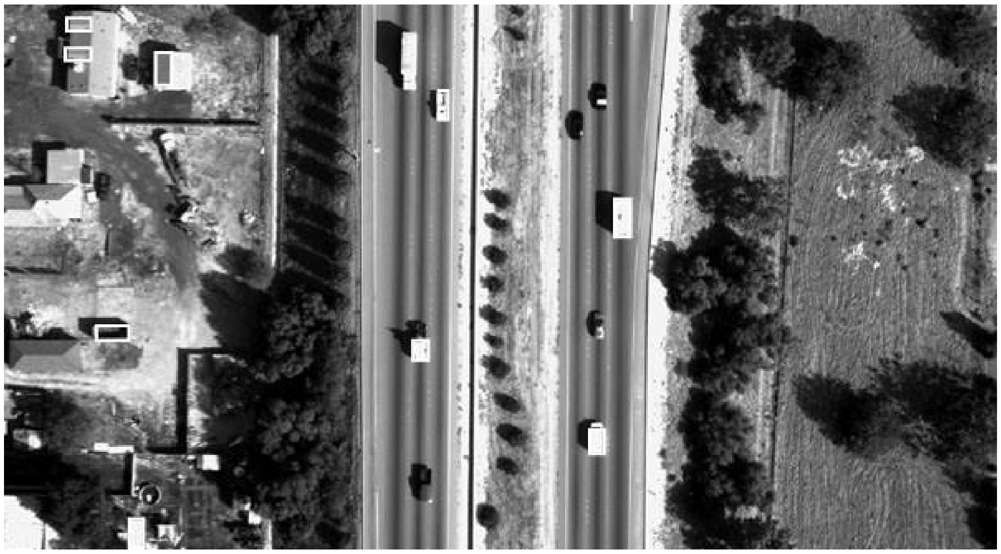


Figure 6.17 Final Output of v22 without concentration

6.1.7 Test Image v23

This image shows a road crossing. It's a typical city square, with buildings and trees all around. Good contrast exists between vehicles and pavement, thus helping in detection. Presence of rectangular shadows again poses a challenge to differentiate between the false indicators and the correct classification. The vehicles are fairly dispersed. The edge image (Figure 6.18) shows that the edges are providing positive information for the detection and have filtered out the unnecessary information. Out of 4 vehicles the system detected two vehicles with concentration condition applied (Figure 6.19), here the cause of the misdetection was the distance between the vehicles, as without the concentration applied (Figure 6.20) the results improved to detection of all the four vehicles, though again presence of a shadow having a darker shade and rectangular shape caused a false positive to occur. As mentioned earlier this can be rectified by the use of an intelligent technique like Bayesian or Neural Networks.

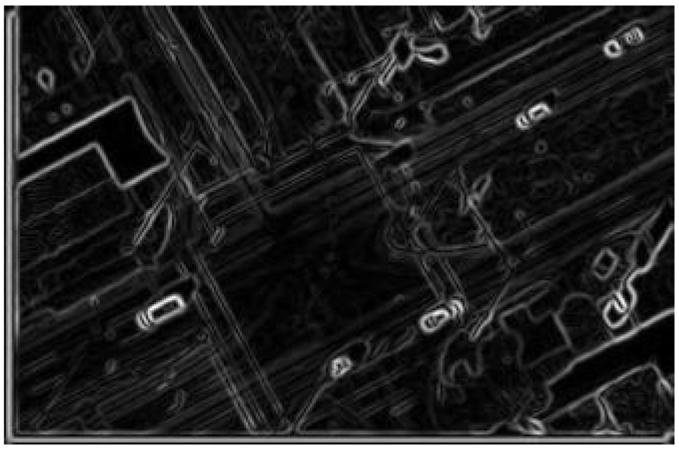


Figure 6.18 Edge image of v23

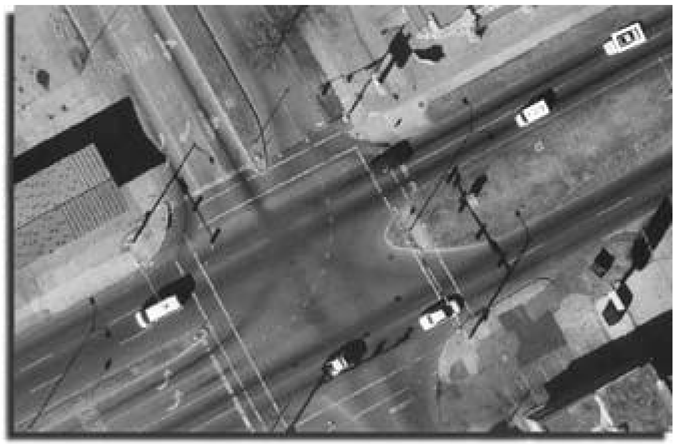


Figure 6.19 Final Output of v23 with concentration

Out of 4 vehicles the system detected two vehicles with concentration condition applied, here the cause of the misdetection was the distance between the vehicles, as without the concentration applied (Figure 6.20) the results improved to detection of all the four vehicles, though again presence of a shadow having a darker shade and rectangular shape caused a false positive to occur. As mentioned earlier this can be rectified by the use of an intelligent technique like Bayesian or Neural Networks.

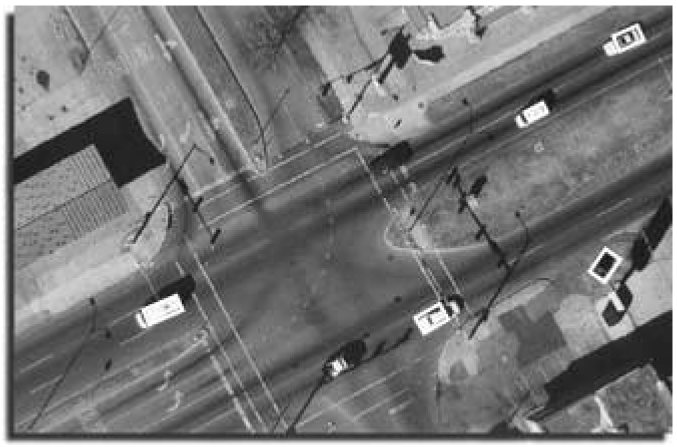


Figure 6.20 Final Output of v23 without concentration

6.1.8 Test Image v24

This image is a low resolution image with lot of noise and low contrast. There is a concentration of tanks, but the edge image (Figure 6.21) shows that almost all the fighting vehicles are not giving a good rectangular edge response and hence the misclassification is caused. Out of 21 vehicles only two vehicles were correctly classified with concentration condition (Figure 6.22) and without the concentration condition (Figure 6.23) the result improved to four vehicles. This image provided an insight into the problems to be faced when classifying tanks as they present a different edge shape in images with low resolution. This implies that a different geometric shape be used to identify and detect the tanks. This makes the system adaptable to detect other objects like bunkers, fortifications and buildings also after making appropriate changes. The problem of lighting conditions is also highlighted by this image, as the low contrast has made the detection more difficult. The results improved with the concentration condition removed and 5 vehicles were correctly classified. There were no false positives as no other object was giving out edges close to that of the vehicles.

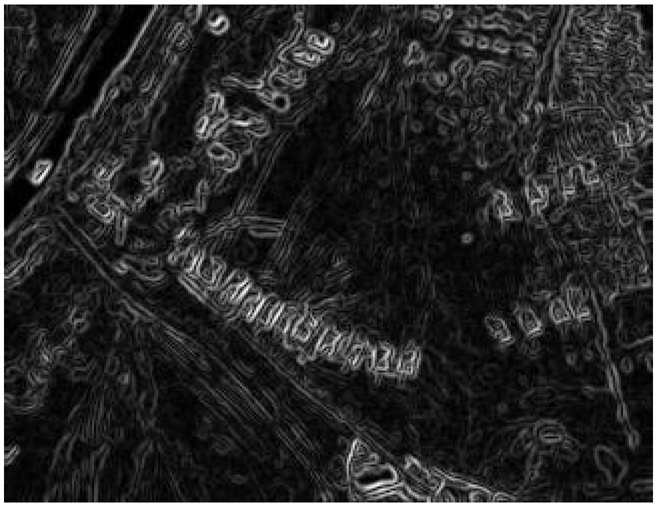


Figure 6.21 Edge image of v24

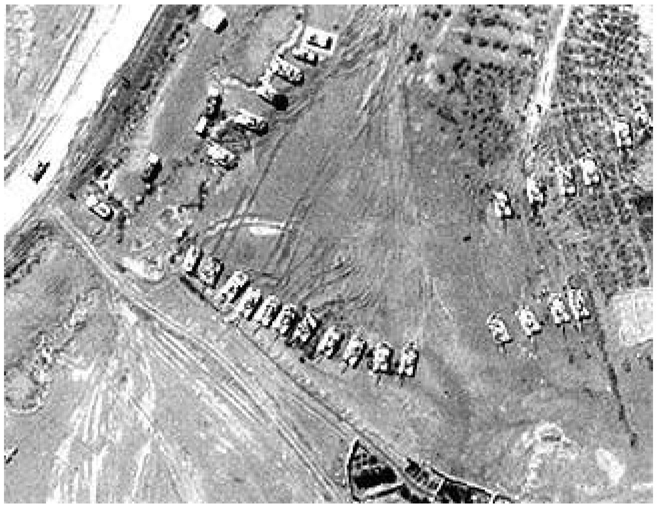


Figure 6.22 Final Output of v24 with concentration

The errors are fairly high, but can be justified on the ground that the system is designed to work without any image context information and is assumed to work in cross country also.

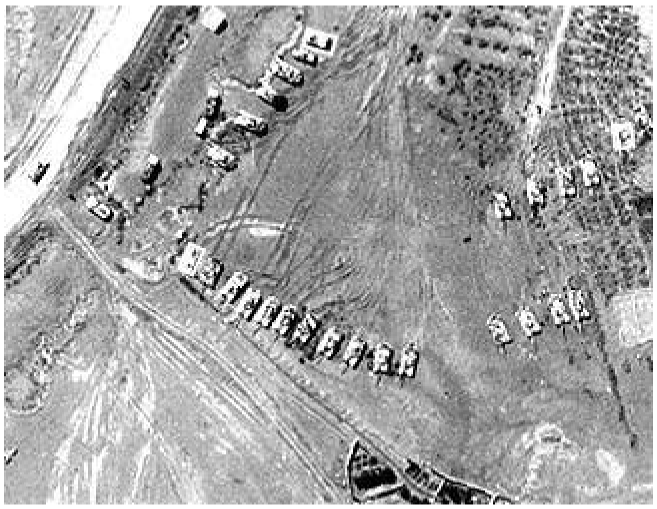


Figure 6.23 Final Output of v24 without concentration

6.1.9 Test Image v25

A vertical aerial photo of a parking lot with 163 vehicles.. There are vehicles of a large variety of shades. Certain vehicles have been occluded by other vehicles or their shadows. Large no of vehicles, evenly dispersed are present. The edge image (Figure 6.24) is an ideal image as it shows the edges of most of the vehicles distinctly.

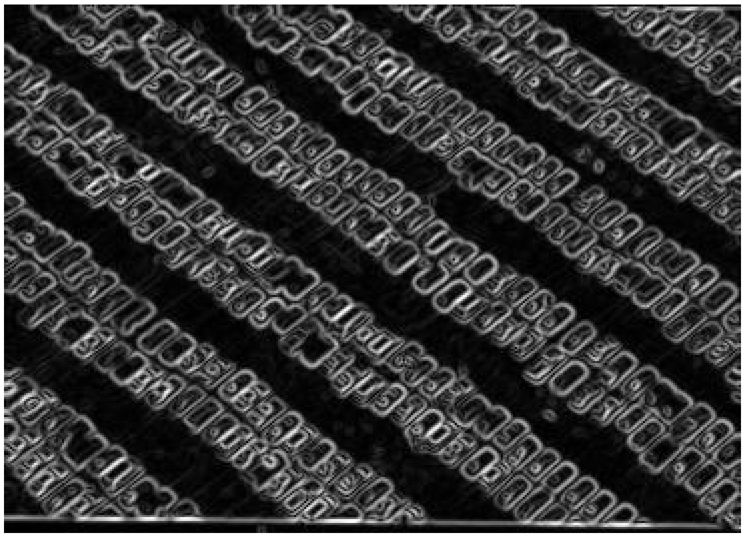


Figure 6.24 Edge image of v25

It was edge image of v25 because of good resolution and contrast in the image. The system detected 130 vehicles with (Figure 6.25) and without the concentration condition (Figure 6.26) as the all the vehicles were closely concentrated.

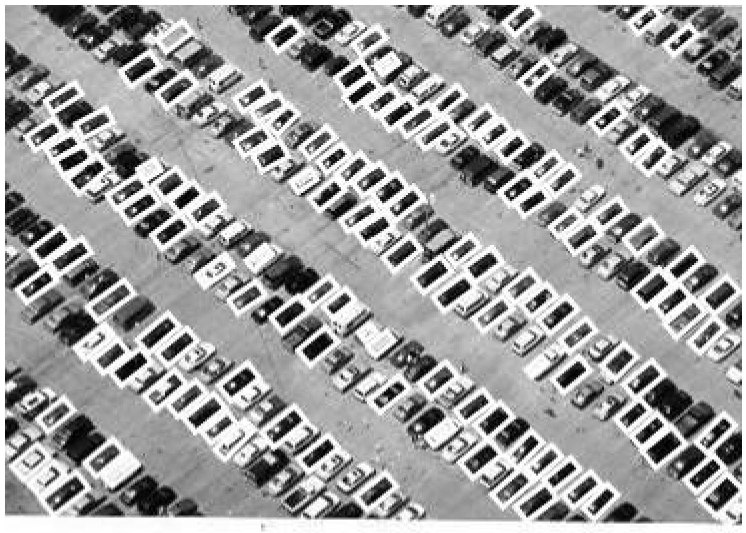


Figure 6.25 Final Output of v25 with concentration

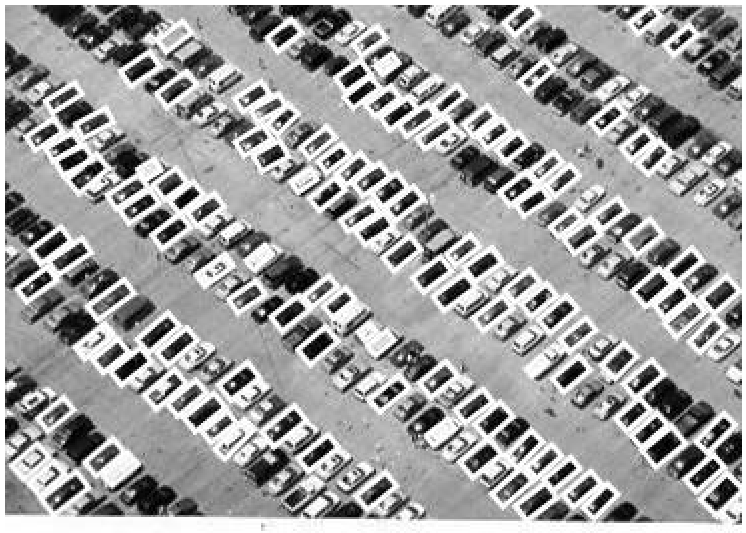


Figure 6.26 Final Output of v25 without concentration

6.1.10 Test Image v26

Multi tiered road structures are shown in this image. The image is characterized by the presence of fewer vehicles, widely dispersed. Different shades of vehicles are visible. Back ground is dark and has good contrast with road texture. The lane markings are creating a problem for the system as visible in the edge image (Figure 6.27). There are 8 vehicles present in the image. The system was able to classify 2 vehicles correctly with the concentration condition (Figure 6.28) applied as the vehicles were a large distance apart. The results improved with the concentration condition removed (Figure 6.29) and 5 vehicles were correctly classified. There were no false positives as no other object was giving out edges close to that of the vehicles.

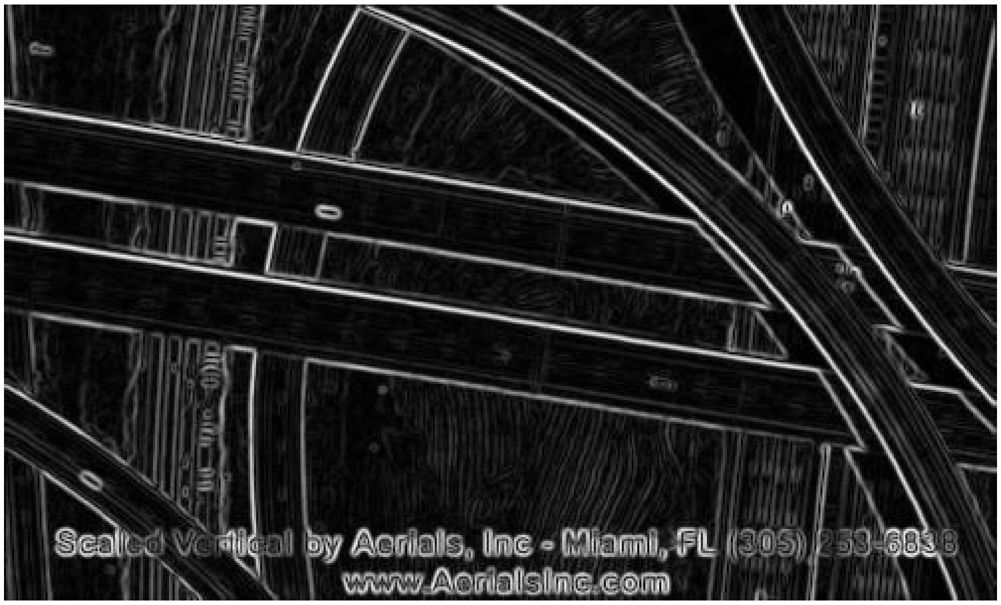


Figure 6.27 Edge image of v26

Here another important conclusion comes to fore, that the system is capable of discounting the presence of large rectangular structures as are visible in this edge image. This is due to the limit imposed on the permissible size of a vehicle. This can be a disadvantage also in cases where the image is taken

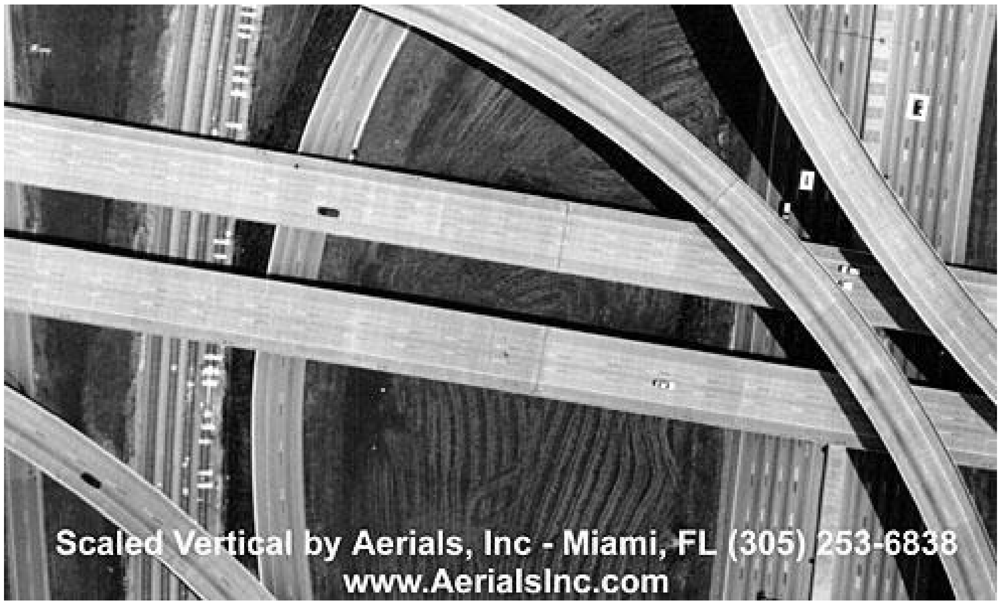


Figure 6.28 Final Output of v26 with concentration

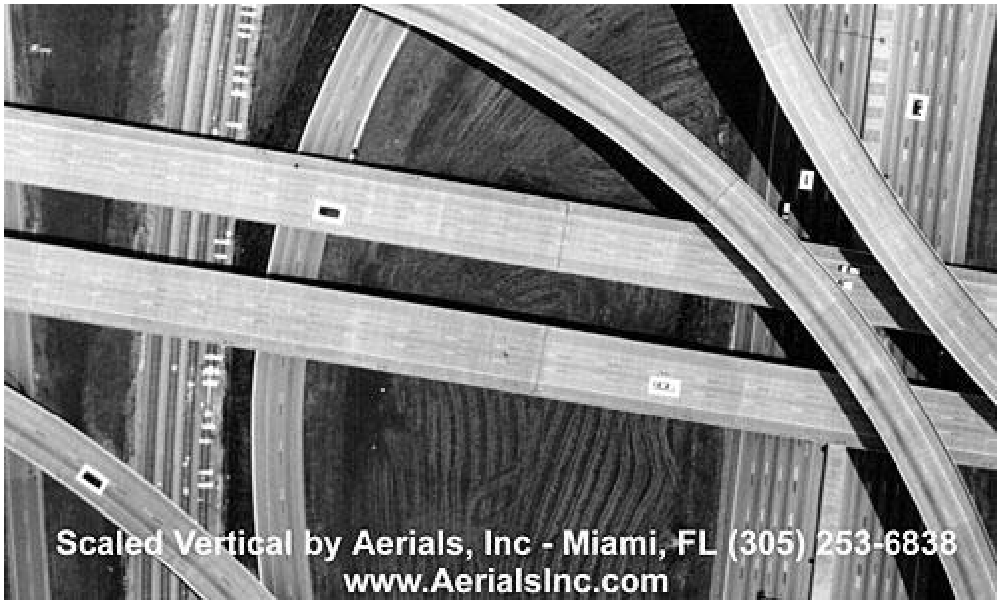


Figure 6.29 Final Output of v26 without concentration

from less height. In that case the vehicle will give out a larger rectangular profile and hence will also be neglected by the system, as the large structures in this image.

6.2 Efficiency Analysis Through Data Set

For efficiency analysis of the algorithms, various parameters of the results obtained from the implementation on the data set were recorded according to output with and without concentration conditions respectively in the Table 6.1 and Table 6.2.

Image Label	Total vehicles	Detected	False +ve	False -ve	Correct Detection	Errors	Time Taken (ms)
ATR	1	1	0	0	1	0	469
Img83	20	19	13	14	6	27	3906
v2	9	4	0	5	4	5	5766
v10	64	14	0	50	14	50	4016
v22	9	8	3	4	5	7	3047
v23	4	2	0	2	2	2	1172
v25	163	130	0	33	130	33	1500
v26	8	2	0	6	2	6	2328

Table 6.1 Results with Concentration Conditions

Image Label	Total vehicles	Detected	False +ve	False -ve	Correct Detection	Errors	Time Taken (ms)
ATR	1	1	0	0	1	0	469
Img83	20	19	13	14	6	27	3735
v2	9	13	4	0	9	4	5609
v10	64	23	0	41	23	41	3857
v22	9	9	4	4	5	8	3015
v23	4	5	1	0	4	1	1141
v25	163	130	0	33	130	33	1515
v26	8	5	0	3	5	3	2297

Table 6.2 Results without Concentration Conditions

The results show that there are instances when the system is giving false positive responses. After analyzing the image characteristics of the images where there is a large number of false positive responses like Test image Img83 Figure 6.3, it comes to notice that where there is a large built up area with rectangular profile buildings the system gives a lot of false positive responses. It can be justified on the basis of assumptions that state that the system is designed for detection of vehicles in cross country terrain. When we consider the false negative responses, mostly the shadows of surrounding images and occlusion caused by other vehicles are responsible for this. In Test image v10 (Figure 6.9), there are maximum false negative responses which are caused by presence of deformed and overturned vehicles and burnt out ground which has caused the most of false negative responses. Points which can be derived from the analysis of results: There is a need to integrate some measure of intelligent classification preferably through Bayesian network to eliminate the false positive responses if the system is to work efficiently in built up area too. Vehicle shadow can be used as a very important cue for detection; it will help in eliminating the false negative responses. As the number of vehicles increases the error rate tends to become constant Figure 6.30 shows this that the error rate has become stagnant after a certain threshold. It proves that the system is working efficiently on detection of clusters of vehicles. Further specific military images are needed to verify the efficacy of the system on military vehicles. The Error analysis (Figure 6.30) shows that the percentage of errors drop considerably with the increase in the number of vehicles in the image, this validates the efficiency of the system based on the underlying assumption of concentration detection. Figure 6.31 shows the time vs. size relationship between the image processing parameters. As the image size increases the memory and the time required for its processing also increases. The memory plays an important role in the processing of any image by the system. As the size of the image increases more than a threshold then the available memory proves deficient to process it and hence the system returns an out of memory exception thrown by the Java virtual machine.

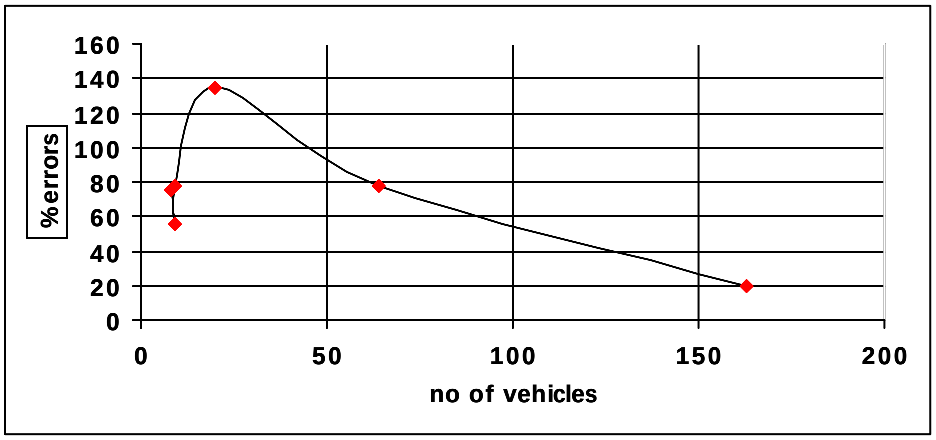


Figure 6.30 Error Analysis

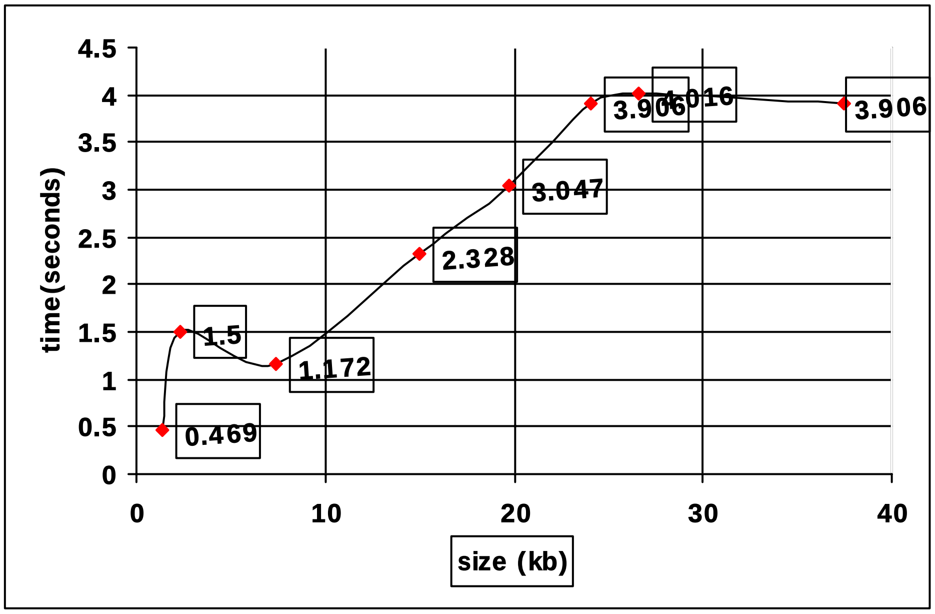


Figure 6.31 Performance Analysis

Conclusion

This project was an attempt to indigenously develop a system for automatic analysis of aerial images. It is by no means a complete system, but still it provides a foundation stone for further research and development in the field. It has a lot of room for improvement especially in the form of intelligent classification of vehicles. A larger and more specific image data set is needed to further test and refine the system. A more specific data set, keeping the project assumptions in focus is necessary to further validate the system. The findings have proved the efficacy of the approach i.e. use of geometric shape with no context information about the background features to detect objects in aerial images. Most important finding is the need for an intelligent post analysis processing module. It will help in reducing the large number of false positives which are increasing the error rate. Keeping the past research in view and the requirements of the project, the most suitable intelligent solution seems to be the Bayesian networks.

The analysis highlights another area: shadow information, which is presently causing false detections/misdetections. If the lighting directions, orientation of the sun is known then the same information can be used to augment the information contained in the geometric model used for detection. The best approach will be to estimate the shadow effect on the model, based on the sun orientation information and the incorporate this information in the model to help in the detection process. The detection of fighting vehicles like tanks is another area where the system can be improved. The different geometric model adapted to the requirement will make the detection of fighting vehicles more efficient and effective. In the present approach back ground information available from the image is discounted completely, this information can be helpful, if used in conjunction with the geometric model. The presence of one vehicle in an area where there are no linear features (roads, highways) should increase the probability of finding other vehicles in surrounding cross country terrain. The distinction between soft vehicles and fighting vehicles will also become possible if the intelligent technique is incorporated. The present state of the project provides an optimized geometric information based detection paradigm to be used by the intelligent plat form.

Future work

The project also identified two major directions for future research. Two methods presented in this study require a background image. Research is required for suitable methods for generating the background image. Research will also be required for developing a standard set of post-processing operations. These operations are performed on the segmented image to sieve out spurious responses and cluster appropriate pixel groups into potential vehicles. The standard set of operations will consist of a set of criteria that will be applied in a pre-determined sequence to generate the final vehicle counts. Similar approaches can also be useful for other object detection and recognition tasks. Intelligent post processing in the form of Bayesian networks or alternatively neural networks is highly desirable to improve the detection rate. Another area to focus on is the migration from Java to C to have access to more efficiency in terms of processing power and time. The incorporation of shadow and back ground information in the post processing steps can be helpful in enhancing the performance.

LIST OF REFERENCES

[1] Taylor, J. I., Photogrammetric determination of traffic flow parameters, Ph.D. Dissertation, The Ohio State University, Columbus, Ohio, 1965.

[2] Agin, G.J, Knowledge based detection and classification of vehicles and other objects in aerial images, DARPA 79, pp. 66-71.

[3] Moon, H., R. Chellapa, and A. Rosenfeld, Performance Analysis of a simple vehicle detection algorithm, Image and Vision Computing, vol. 20, no. 1, pp. 1-13, 2002.

[4] Rajagopalan, A. N., Phillipe Burlina and Rama Chellapa, Higher Order Statistical Learning for Vehicle Detection in Images, Proceedings of the Seventh International Conference on Computer Vision, vol. 2, pp. 1204-1209, 1999

[5] Zhao, T. and R. Nevatia, Car Detection in low-resolution aerial images, IEEE Proc. of Int. Conf. on Computer Vision, 2001.

[6] Zhao, T. and R. Nevatia, Car Detection in low-resolution aerial images, IEEE Proc. of Int. Conf. on Computer Vision, 2001.

[7] Sastry, C. V. S., Extraction of vehicle information from 1-m resolution imagery, Masters' Thesis, The Ohio State University, Columbus, Ohio, 2000.

[8] Nagao, M., and T. Matsuyama, A structural analysis of complex aerial photographs, Plenum Press, New York.

[9] McCord, M. R., P. K. Goel and C. J. Merry, Traffic Monitoring Using Satellite and Ground Data: Preparation for Feasibility Tests and an Operational System, Final Report to The Ohio Department of Transportation, The Ohio State University, Research Foundation, Columbus OH, 2000.

[10] Shapiro, L. G., and G. C. Stockman, Computer Vision, Prentice-Hall, 2001.

[11] Lee, S-S. , S-J. Horng, and H-R. Tsai, Entropy Thresholding and Its Parallel Algorithm on the Reconfigurable Array of processors with Wider Bus Networks, IEEE Transactions on Image Processing, Vol. 8, No. 9, 1999.

[12] Chang, C. I., K. Chen, J. Wang, and M. L. G. Althouse, "A Relative entropy-based approach to image thresholding," Pattern Recognition, vol.120, pp. 215–227, 1993.

[13] Beghdadi, A., A.L. N´egrate, and P.V. Lesegno, "Entropic thresholding using a block source model," Comput. Models Image Processing, vol. 57, pp. 197–205, 1995.

[14] Lin C. and R. Nevatia, Building Detection and Description from Single Intensity Images, Computer Vision and Image Understanding, vol. 72, No. 2, pp. 101-121, 1998.
[15] Cheng, H. D., Y. H. Chen, and X. H. Jiang, Thresholding Using Two-Dimensional Histogram and Fuzzy Entropy Principle, IEEE Transactions on Image Processing, vol. 9, no. 4, 2000.
[16] Yanowitz, S. D. and A. M. Bruckstein, A New Method for Image Segmentation, Computer Vision, Graphics and Image Processing, no. 46, pp. 82-95, 1989.
[17] Sarkar, S., and K. L. Boyer, Quantitative measures of change based on feature organization: eigen-values and eigen-vectors, Computer Vision and Image Understanding, Vol. 71, No. 1, pp. 110-136, 1998.
[18] Roux, M., and D. M. McKeown, Feature matching for building extraction from multiple views, IEEE Computer Vision and Pattern Recognition Conference, pp. 204- 206, 1994.

Appendix 1

1. User Manual

1.1 About Image/J

Image/J is a program which purpose is to enable picture processing. There are several types of software aimed to manipulate images. Some programs are made to change an image by adding different features to it. Those programs are widely used by designers and other artists. Image/J on the other hand is a program that can by described as a tool used in scientific areas, e.g. medicine. Researchers want to measure the size of different parts and organs to make various conclusions. Image/J allows them to analyze their images in such a way. This is an example of Image/J's range of uses but it can be used in many other science fields e.g. physics. Image/J can display, edit, analyze, process, save and print 8-bit, 16-bit and 32-bit images and can create Stacks - series of images presented in one window. It can calculate pixels, distance and angles, area and pixel value. Furthermore, it supports many standard image processing functions, including contrast enhancement, density profiling, smoothing, sharpening, edge detection, and other filtering functions.

Besides the obvious advantages with Image/J as an image processing software there is one great concept behind its development worth mentioning. Image/J is open source and many people participate in the development. Image/J was designed with an open architecture that provides extensibility via Java plugins. The plugins can be developed using Image/J's built in editor and a Java compiler, which makes it possible to write code that solves almost any image processing or analysis problem.

1.2. Installing ImageJ (Windows operating system)

Copy the ImageJ folder to disc and transfer it to the C drive of computer. Open the ImageJ folder in the C drive and copy the shortcut (microscope with arrow) to your desktop. Double click on this shortcut to run ImageJ. If you are running ImageJ from some location other than the C drive, double click on the 'red apple' icon in the ImageJ folder to launch the program.

1.3. Basic Operations.

An ImageJ window will appear on the desktop; do not enlarge this window. Select File,. Open from the menu to open a stored image file. Selection Tools**:** The first four buttons on the toolbar are area selection tools; they allow you to surround an area on the image with a rectangle, oval, polygon or a freehand shape. After selection, these specific areas may be altered, analyzed, copied, etc. using the menu commands. Notice that the status bar, below the toolbar, gives the location of the selection (xxx, yyy) and its dimensions in pixel .Line Tools**:** The next three buttons are line tools that create straight, segmented or freehand lines. Again note that information is displayed on the status bar as the line is drawn. Double-click on the line button to alter the line width. Ctrl+D or Edit Draw makes the line permanent. Crosshair Tool**:** The crosshair tool allows you to mark locations on the image; with each click the coordinates of the pixel (xxx, yyy) and brightness (0-255) are recorded in the data window. Color images will have three brightness readings displayed on the status bar, one each for the red, green and blue channels, however only one brightness value will be printed in the data window. Wand Tool**:** This tool automatically finds the edge of an object and traces its shape. It works best with high contrast images (see Thresholding, next page). Place the wand to the left of an edge; click and the algorithm will search to the right for an edge. It will then trace along the edge of the object until it returns to the starting point. Text Tool**:** Double click on this button to select a font and size. A large font size will probably be required for an image from a digital camera. Single click the button, click-drag a text box and type the label. Move the box to the desired location and set the text in place with Ctrl+D or with Edit Draw**.** Magnifying Glass: Left-click on the image to magnify; right-click to reduce the image size. Scrolling Tool: This allows you to move the image if the picture is larger than the window. Color Picker: This tool sets the foreground drawing color or text color by "picking up" colors from images with the eyedropper. Colors also may be picked up from the *Colors* window by double-clicking the color picker button. Alt-click in the Colors window to change the background color. The icon for this tool (eye dropper) shows the current foreground color while the frame around it shows the background color.

1.4 Image Processing

The next is a very small sampling of processing techniques that are possible with ImageJ. See the ImageJ and NIH Image websites for more information. ImageJ is best used for image analysis; I use it in conjunction with more powerful photo editors such as Adobe Photoshop. You may want to open a spreadsheet so that data can be efficiently 'cut & pasted' during image analysis. Also, it is a very good idea to make a backup copy of your image before doing any processing. Undo. Edit Undo reverses the preceding action. Only one back step is possible. Revert. File Revert should revert to the original saved image. Cropping. Surround the area with the rectangular selection tool followed by Image Crop. Clear Outside. Make a perimeter with a selection tool followed by Edit Clear Outside. The technique is useful for clearing extraneous objects near an item of interest. Edit Clear clears inside of the perimeter. Enhancing Brightness and Contrast. Image.Adjust Brightness/Contrast; click 'auto' or set manually Removing Noise. Process Noise Despeckle or try Process Filters Median Rotating an Image. Image Rotate and select type of rotation Converting to Grayscale. Image Type 8-bit converts the image to 256 shades (8-bit) of gray. In this scale 0 = pure black and 255 = pure white…. a grayscale reading of 128 would be a medium gray. Thresholding (Binary Contrast Enhancement). This is commonly used when detecting edges, counting particles or measuring areas. A grayscale image is converted to binary (a.k.a. halftone or black & white) by defining a grayscale cutoff point. Grayscale values below the cutoff become black and those above become white. The procedure: First convert the image to 8-bit grayscale (see above). Process Binary Threshold creates a 'thresholded' binary image. A less automated procedure involves: Image Adjust Threshold; use the slider to adjust the threshold. The red areas will become the black portions in the binary image. Click 'Apply' to complete the conversion. 'Brightness slicing' is a similar procedure that uses both upper and lower thresholds. Measuring and Counting Objects (also see accompanying handout with ImageJ examples) Measuring Distance Between Points. Using the straight-line tool draw a line between two points. The status line will show the angle (from horizontal) and the length in pixels. Use the next step to set the scale: Setting Measurement Scale. Draw a line between two points of known distance such as a ruler on the photograph. Go to Analyze Set Scale. In the Set Scale window the length of the line, in pixels, should be displayed. Type the known distance and units of measure in the appropriate boxes and click OK. Measurements will now be shown using these settings. If the pixel: length relationship is known from a previous measurement you may directly type this information in the Set Scale window. Check 'global' to apply this scale to other frames. Measuring Area. Surround an area with a perimeter. This can be done with an area selection tool, the wand (in high contrast images) or with Analyze Particles (see below). Go to Analyze □Measure; the data window will show the area and pixel brightness values for the object in the perimeter. Use Analyze . Set Measurements to select additional parameters (such as perimeter length) to be displayed. Counting Particles. As described above convert the image to 8-bit grayscale and then 'threshold' the image. Go to. Analyze Particles, type the upper and lower limits for the particle size and toggle 'show outlines'. Click on OK and each counted particle will be outlined and numbered in a new widow (numbers may be very small). The data window contains measurements for each particle. Saving Files. Images from digital cameras are usually saved as JPEG files. JPEG is a type of memory compression that results in the loss of some data. A JPEG image degrades each time it is opened, edited and resaved. It is best to save a file in a 'lossless' format such as a TIFF during the editing process. Pressing "S" on the keyboard brings up the Save as TIFF window. As you save a file, confirm that the extension '.tif' has been added to the filename. Printing. Should you encounter printing problems, save the processed image and print with a photo editor. Saved images also may be inserted into MS Word for printing.

Appendix 2

Assumptions/Constraints

Following are a few assumptions or constraints of our project.

a. Required resolution of the aerial image should be 25 cm.
b. Only vertical/nearly vertical images are considered.
c. No context information about the image is available.
d. No image registration/GIS support.
e. No information about camera height.
f. Focus on detecting convoys/vehicle concentrations.
g. A single image is used
h. 8-bit grey scale images are used.