5-2006

Object Tracking in Distributed Video Networks Using Multi-Dimentional Signatures

Sabeshan Srinivasan

OBJECT TRACKING IN DISTRIBUTED VIDEO NETWORKS USING MULTI-DIMENSIONAL SIGNATURES

By

Sabeshan Srinivasan

B.E. College of Engineering, Guindy,

Anna University, 2003

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

May, 2006

OBJECT TRACKING IN DISTRIBUTED VIDEO NETWORKS USING MULTI-DIMENSIONAL SIGNATURES

By Sabeshan Srinivasan

Thesis Advisor: Dr. Anthony Stefanidis

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Spatial Information Science and Engineering)
May, 2006

From being an expensive toy in the hands of governmental agencies, computers have evolved a long way from the huge vacuum tube-based machines to today's small but more than thousand times powerful personal computers. Computers have long been investigated as the foundation for an artificial vision system. The computer vision discipline has seen a rapid development over the past few decades from rudimentary motion detection systems to complex model-based object motion analyzing algorithms. Our work is one such improvement over previous algorithms developed for the purpose of object motion analysis in video feeds.

Our work is based on the principle of multi-dimensional object signatures. Object signatures are constructed from individual attributes extracted through video processing. While past work has proceeded on similar lines, the lack of a comprehensive object definition model severely restricts the application of such algorithms to controlled situations. In conditions with varying external factors, such algorithms perform less efficiently due to inherent assumptions of constancy of attribute values. Our approach assumes a variable environment where the attribute values recorded of an object are deemed prone to variability. The variations in the accuracy in object attribute values has been addressed by incorporating weights for each attribute that vary according to local conditions at a sensor location. This ensures that attribute values with higher accuracy can be accorded more credibility in the object matching process. Variations in attribute values (such as surface color of the object) were also addressed by means of applying error corrections such as shadow elimination from the detected object profile. Experiments were conducted to verify our hypothesis. The results established the validity of our approach as higher matching accuracy was obtained with our multi-dimensional approach than with a single-attribute based comparison.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to Dr. Anthony Stefanidis, for his constant guidance, support, encouragement, and patience, and to the other members of my thesis committee, Dr. Peggy Agouris and Dr. Silvia Nittel, for enriching this thesis with their ideas and suggestions. I would also like to thank Dr. Stefanidis and the Graduate School for supporting my studies at the University of Maine all throughout and making my dream of higher studies come true.

I would like to thank the research team of Pete Doucette, Vijay Venkataraman, and others, whose excellent foundational work on motion detection and tracking forms part of my research. I would like to thank all my colleagues in the SIE department, especially, Vijay Venkataraman, Charalambos (Harris) Georgiadis, Caixia Wang, Hariharan Gowrisankar, and Farhan Faisal for their frequent suggestions and ideas. Harris and Vijay were especially helpful in capturing video datasets for the experiments I ran. I would also like to thank the sponsors of this research, the National Science Foundation and the National Geospatial-Intelligence Agency for supporting us so far.

I am very grateful to have a loving and supportive family who have always been with me in all my endeavors and encouraged me to achieve more and more. But for the numerous sacrifices made by my parents, I would not be where I am. I thank all my friends in Maine for making my stay here a memorable and enjoyable one. Finally, I want to thank God for providing me this opportunity. Not many in the world are blessed enough to pursue their dreams successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

Chapter 1

INTRODUCTION

The use of sensor networks is revolutionizing the way that geospatial information is collected and analyzed. In terms of image analysis, traditional satellite- and aerial- based static analysis is now complemented by the use of distributed video sensors to capture and monitor dynamic events (like the movements of cars and people). This evolution towards geosensor networks (Stefanidis and Nittel, 2004) is bringing forward interesting research challenges, mainly relating to information integration and analysis.

Object detection and tracking techniques from motion imagery have improved significantly from being exotic automation techniques and are employed in varied fields like surveillance, human motion analysis, traffic monitoring (Chachich et al., 1996) and human-machine interfaces (Khan and Shah, 2003), security, missile tracking, and rescue operations (Eltoukhy and Salama, 2002). Hardware based object tracking systems enable real-time processing of captured video data. Most approaches to object tracking across a multisensor video network involve comparison of video feeds based one or a few of observed physical attributes of objects. While this is feasible for stationary camera networks and controlled environments, where external errors such as those caused by illumination can be modeled and accounted for in subsequent analyses of data generated from these networks, it is difficult to establish the same for mobile camera networks, such as a battlefield, where network location and topology changes unpredictably. In such scenarios, it would be unrealistic and impracticable to expect a simple error model to work. Worse, it is not possible to establish a noise model as the ambient changes are often sudden and unexpected.

This thesis addresses object tracking in video networks distributed in broader urban environments, and introduces novel modeling and comparison approaches to support persistent object tracking within single and across multiple sensor feeds.

1.1. Background of Thesis

Computer vision systems have existed for many years in several applications. The most fundamental application of an automatic motion detector would of course be a passive device like a detector based on an infrared beam. This worked on the basis of signal loss due to interruption by a moving object in the line of sight of the sensor. Automated detectors have evolved from such a rudimentary past to advanced robotic cameras that consistently track the motion of an object in their field-of-view. Modern object tracking systems comprise diverse state-of-the-art sensors like high-resolution video cameras, onboard GPS units, and mechanized units for panning the camera in response to object motion in its field of view.

Most common applications for motion tracking are localized surveillance and security assessment systems in industrial/commercial facilities. Numerous commercial establishments like shopping malls or security-sensitive locations like oil refineries employ motion monitoring systems that can detect unauthorized intrusion into the premises and issue an alert once prolonged suspicious activity is detected. In addition to such localized applications we have applications like traffic monitoring systems that are network-based, with sensors scattered across a broad geographic area. These applications are more challenging in that their processing needs are much greater than static applications. Robust models of object properties would help in the establishment of a reliable and consistent object labeling mechanism, and the tracking of these objects. Such applications also require that individual sensors are in constant correspondence with one another such that an *ad hoc* area-of-interest can be established where objects are tracked in greater detail or being followed for studying their activity.

Far more complicated are systems that involve a set of mobile sensors monitoring a variable region of interest. In this case, the sensors move in response to the motion of objects in their field of view. Further complication is introduced by the fact that the configuration of the network changes frequently, so for example, sensor S_{11} which was to the right of sensor S_{12} might move to its left in order to continue tracking an object in its view. Besides, sensors in such systems need to co-operate with one another such that the network always keeps track of a set of objects the user is interested in. It is much more difficult to process the video data generated by these systems because of the background motion in the video caused by sensor movement. A typical application is aerial tracking of sensitive locations by means of unmanned aerial vehicles (UAV).

1.2. Statement of objective

The aim of this research and, subsequently, this thesis is to establish a method to enable automatic labeling and linking of objects appearing in video data recorded by distributed non-overlapping video sensors in a geospatial network. The linking of objects will ultimately be useful in the description of the motion path of any object as it moved across the network covered by the video sensors. In addition to defining object trajectories, other useful information about detected objects such as their physical attributes like mean surface color, shape profile and spatio-temporal behavior can be obtained.

In this thesis, we hypothesize that by extending the currently-used limited physical attributes of an object into composite object signatures, we improve our ability to track and link objects in distributed non-overlapping video networks. More specifically, we propose the use of composite object signatures comprising geometric (e.g. size), radiometric (e.g. average surface color), and behavioral properties (e.g. spatiotemporal patterns of movement) of this object. Such attributes are typically considered to be stochastic values, thus we consider accuracy measures assigned to them, to express their participation in dissolving object matching ambiguities. In the content of this thesis, the term *object matching* refers to the identification of the same object in two non-overlapping video feeds, at two distinct instances. This is a critical issue for sensor network, monitoring, and surveillance applications, as it allows us to collect robust and persistent data rather than instantaneous glimpses of a scene.

1.3. Scope of research

In this research, we address the issue of object tracking in a geosensor network of distributed video sensors. As mentioned before, past approaches to linking objects were often based on a single (Saltenis and Jensen, 2002) or limited number of attributes (Eltoukhy and Salama, 2002; Chachich et al, 1996). In (Venkataraman et al, 2004), the authors considered an object indexing and retrieval scheme based on color alone. Another approach deals with indexing objects based on their motion characteristics (Saltenis and

Jensen, 2002). An interesting approach based on Hidden Markov descriptions of positions and traffic flow was presented in (Ch. Jaynes, 2004). However, we propose using several object attributes together for the purpose of object matching in different video streams. Multi-component signatures, comprising geometric, radiometric, and spatio-temporal properties, are defined for each object detected in a video. These signatures are used to establish object correspondences and link trajectory segments across different non-overlapping feeds.

1.4. Intended Audience

This thesis primarily addresses an audience that is related to the domain of geospatial networks, especially one involved in video sensors. This thesis may also be of interest to researchers and scientists involved in the implementation of such sensor networks for applications such as surveillance and monitoring. Image processing-based industries may find the object detection algorithm useful for developing real-time object tracking software systems. The department of homeland security is also a potential stakeholder who could put this research into effective use for monitoring vehicles and troops in locations where manual surveillance is not possible or risky.

1.5. Organization of Thesis

Chapter 2 describes the various approaches developed by other researchers for addressing the issue of motion detection and object-linking. Some motion detection models like the pixel motion, Markov and arithmetic are discussed. A basic outline of the object motion analysis system described later in the thesis is also discussed. This system is based on the principle of multi-component object signature, discussed in later chapters. Further discussion is on sensor as well as network parameters and also upon the environment in which the system is designed to operate, in terms of internal network configuration as well the distribution of motion sensors across a geospatial network.

In Chapter 3, the problem of object tracking is explained in detail in terms of how it is tackled in this research. The core concept of multi-component object signatures is discussed later in the chapter. The problems involved in processing datasets containing additional motion induced by a moving sensor are also discussed in the chapter.

Chapter 4 deals with the important concept of indexing objects based on their parametric signatures. Object indexing is crucial in linking objects correctly across different video feeds. Two possible clustering methods are discussed. We describe the method for linking objects in different video sets by means of various comparison techniques. The chapter also discusses the trajectory linking process which takes place once objects are linked.

Chapter 5 discusses the results of experiments conducted on a few real-world video datasets for both static and mobile sensors. The experimental datasets contain a variety of moving objects of different profiles.

Chapter 6 describes the findings of this research, presents conclusions and sets the tone for future work based on this thesis by suggesting possible directions.

Chapter 2

LITERATURE REVIEW

Spatial data has crossed the threshold from being just an exotic form of data into an everyday necessity in today's networked world. Many applications like facilities management, manufacturing and agriculture make use of some form of spatial data in managing their resources. Already mobile GPS units have made their appearance in the public for aiding activities like hiking, mountaineering, orienteering etc. Spatial data is touching the everyday life of the common as never before. Experts in the domain (Culler et al, 2004) predict the large-scale deployment of dense sensor networks that sense all kinds of phenomena from water quality to industrial equipment health monitoring.

Experimental networks already exist for applications such as mapping glacial movement (Martinez et al, 2004), pattern learning (Stauffer and Grimson, 2000), and radiation detection in an urban environment (Brennan et al, 2004). It is not difficult to imagine that in the near future such spatial networks will have a ubiquitous appeal in most domains. One such domain involves aggregation of data collected from a multitude of video sensors over a network comprising highly mobile targets. In order to make effective use of such networks, theoretical models must be developed which will enable consistent and accurate representations of actual objects being sensed. Our endeavor is a small step in representing data from such a network in a generic manner.

2.1. Motion detection and estimation methods

Motion detection and estimation as a computer vision problem has been investigated since the early 1970s. The earliest techniques were based on the concept of optical flow in video. They worked by relating the image intensity differences at points of motion to actual geometric motion. The majority of motion estimation techniques in vogue today are based upon the original work on the determination of optical flow (Horn and Schunck, 1981) in one way or the other. Most current techniques employ some form of error-minimization/optimization to the classical optical flow equation. Based on the optimization technique employed, motion estimation methods are classified broadly into pure optical flow-based methods, pel-recursive or Bayesian methods.

2.1.1. Optical flow methods

Optical flow is a means for representing object motion, as recorded in a video stream, by correlating the brightness differences, caused by object motion in the video frames, to the motion of the actual object. Optical flow is used in the detection of motion as well as to determine the apparent object velocity in an image sequence. Most optical flow-based methods detect and estimate object motion by approximating the optical flow field using spatio-temporal image intensity gradient measurements. Object profiles are then constructed from the detected motion pixels using templates, region-growing or motion-constraint-based techniques.

An optical flow-based motion detection and estimation technique is presented in (Shin et al., 2005). Points in an image detected as possessing motion are constructed into objects based on feature correspondences using principal component analysis. Objects are assumed to be deformable and hence feature matches are based on shape changes to the object being tracked. Salient features of an object are defined as those that are not generally misclassified. The main objective of (Walker et al., 1998) is to determine the probability of classification of features in a flow field and those with the highest scores are less likely to be misclassified by a motion segmenting algorithm. Knowing the probability of misclassification of an object feature could aid its reclassification in case of an indeterminate match. In (Amer, 2005), an object detection and classification algorithm is presented. The algorithm works based on a two-step feature extraction from the flow field and subsequently classifying the spatial and temporal attributes of objects.

In (Zelek, 2002), the author discusses a technique for computing dense optical flow using posterior probabilities. A simulator is defined for a set of points perceived to be in motion and its future values are based on current motion estimates. By comparing actual flow vectors with the predicted values, the predictor function is refined further. The lack of suitable ground-truth mechanisms for motion data is addressed in (Galvin et al., 1998). The authors propose generation of complex scenes using ray tracing programs in order to verify the accuracy and efficiency of optical flow algorithms. A hierarchical approach to motion estimation is presented in (Bergen et al., 1992). The process consists of establishing a global model to constrain the structure of motion in a sequence of images, a local model to define the motion characteristics of objects and a coarse-fine motion refinement operator that works on a pyramid-collection of datasets. The technique assumes a rigid body model for objects detected in the video sequence. Often, optical flow algorithms fail to detect motion in certain limiting situations like analyzing a localized region where optical flow is not very apparent. This issue is addressed in

(Bergen et al., 1990) where a solution is proposed in the form of a coarse-fine tracker in order to detect motion in special configurations.

The problem of estimating camera motion is approached in (Shakernia et al., 2002) from several panoramic views by means of analyzing optical flow. With the knowledge of the camera's motion, the motion of objects in its field of view can be determined accurately. The advantages of omnidirectional structure-from-motion (SFM), derived using optical flow, over conventional SFM are discussed in (Cheng and Hebert, 2000). The authors argue that the larger field of view of the omnidirectional SFM is more advantageous in certain applications. (Murray and Basu, 1994) discusses the issues related to tracking motion using an active camera. Their approach involves estimating temporal derivatives by means of image subtraction. In (Kate et al., 2004), a motion detection and estimation technique for detecting vehicles using a mobile camera is described. The approach consists of detecting vehicles differently based on their distance. This follows the principle that motion is harder to detect as objects come closer to a camera's focus of contraction.

A new structure-from-motion algorithm based on optical flow is described in (Ohta and Kanatani, 1995) which works by refining the linearized solution using a statistically optimized method. (Camus and Bülthoff, 1995) seeks to optimize a classical optic flow-based motion estimation algorithm by converting a quadratic objective function into a linear one, with acceptable levels of accuracy. An algorithm for computing optical flow in a differential framework is discussed in (Weber and Malik, 1994). Each optical flow vector computed during execution of the algorithm is assigned a reliability value that can be used for motion segmentation purposes. In (Weber and Malik, 1997), the approach is based on the fact that each distinct object in a video has a unique epipolar constraint associated with its motion. This property is used in segmentation of motion rather than detect discontinuities in depth. By exploiting this property in conjunction with a dense optical flow field, detailed motion estimates can be made. This technique offers immunity from the errors caused by mistaking occlusions for depth discontinuities.

In (Thompson, 1998), an algorithm is detailed on estimating optical flow accurately in the presence of depth discontinuities. While most other flow estimation methods suffer from the error caused by boundary ambiguities, the algorithm presented deals with line processes in order to effectively derive optical flow in such a situation. A technique for estimating structure from motion in the case of an uncalibrated camera is presented in (Brooks et al., 1998) from instantaneous optical flow. The calibration data is dependent on the differential epipolar equation for uncalibrated optical flow. The equation works by relating optical flow to the internal parameters of the camera. In (Chang et al., 1997), the authors describe a Bayesian framework that combines optical flow estimation and motion segmentation. The technique produces a piecewise-smooth motion vector field, which results in simpler and more efficient algorithms.

2.1.2. Pel-recursive methods

Pel-recursive methods work on a predictor-corrector approach. The motion estimated at each pixel is updated by a correction term which is the motion estimate obtained from the previous frame. The correction term is applied iteratively over each pixel in a frame recursively. A smoothness constraint is introduced in order to reduce the effects of changing illumination in causing errors in the estimated motion field by minimizing the displaced frame difference.

Instead of computing optical flow directly, the authors in (Pless et al., 2000) present a solution for estimating background motion by finding out the spatiotemporal image intensity gradient. This ensures that independently moving small objects are detected correctly. Motion detection and estimation is performed using spatiotemporal gradients in (Cohen and Medioni, 1998). Object matching and subsequently, trajectory building is achieved by means of object templates formed using a dynamic temporal coherence of the object over a number of frames. A pure shape-based approach is used in (Lipton et al., 1998) in order to detect objects and match them to templates. Moving objects are classified into one of a few groups based on their similarity to a group.

In (Torr, 1996; Torr et al., 2001; Jepson et al., 2002), an improved Estimation-Maximization technique is presented where objects are treated as closed shapes with uniform motion properties within the boundary. This results in a constant likelihood measure for motion estimation for all pixels within an object's boundary. The authors in (Estrela et al., 2004) describe a technique called spatial adaptation for improving accuracy of motion estimation using an Expectation-Maximization technique. The fundamental idea is to make use of templates or "masks" to refine object boundaries in non-homogeneous image frames. (Soatto et al., 1993) presents a recursive method of determining the structure of a scene (including any moving objects) and the camera motion parameters using an Extended Kalman Filter (EKF). An EKF is a modification of a basic Kalman filter so that it can be applied to non-linear data. In (Irani, 1999), flow-fields in a video are modeled as a low-dimensional subspace. Instead of imposing spatial or temporal smoothness constraints, global model constraints are used so that lack of sufficient local information does not introduce noise in the estimates. A recursive method for estimation of ego-motion and camera calibration is presented in (Soatto and Perona, 1994). Knowing camera calibration parameters, object motion in a video can be estimated accurately.

In (Azarbayejani and Pentland, 1995), the authors present an EKF-based technique for extracting structure from a sequence of images. The technique works by modeling motion and structural changes of objects recursively until convergence is reached within a certain number of image frames. A simplified version of the EKF technique is utilized in (Alon and Sclaroff, 2000) where a planarity constraint is imposed on the points in the detected object. Three techniques (batch processing, recursive and bootstrap) for motion estimation using EKF are evaluated in (Hanmandlu et al., 2003). A new EKF-based motion estimation technique is presented in (Tziritas, 1992) that uses a mean square displaced frame difference for better results. A multi-frame approach is followed in (Oliensis, 1997) for extracting structure from object motion in a video. The technique requires some camera calibration in order to generate estimates accurately. Structure and motion parameters of a scene are determined using the Levenberg-Marquardt algorithm in (Szeliski and Kang, 1993) for its faster convergence.

2.1.3. Bayesian methods

Bayesian methods use probabilistic methods to estimate the displacement field as opposed to pel-recursive methods which use gradient-based techniques. Most Bayesian motion estimation methods use posterior probability to determine the motion differences between subsequent frames of a video.

In (Zhou and Tao, 2003), the authors propose a motion modeling system based on the concept of *maximum a posteriori* (MAP) estimation of a Hidden Markov Model (HMM). Foreground and background layers (Wang and Adelson, 1993) are distinguished based on whether or not there is consistent motion detected in them. All background layers are modeled as multivariate Gaussian distributions. The MAP probability is defined based on the prior probabilities of layer order, shapes, motion and appearance and the associated likelihood function. Object state is estimated by means of a multi-step process. The approach presented in the paper presents a means to detect object motion even in the presence of foreground occlusions. A similar approach is followed in (Vasconcelos and Lippman, 1997; MacCormick and Blake, 1998; Tao et al., 2002) where a Gaussian segmentation prior is used as a predictor for estimating object centroids. Instead of processing video data in groups of two or more, the main focus in the paper is on recursively estimating object motion. In (Black and Fleet, 2000), object motion is modeled separately as smooth translational motion (in areas where there is consistent motion) and motion boundaries (in areas where motion properties vary for adjacent pixels; e.g. when there is an occlusion).

The problem of motion estimation is formulated using Bayesian inference in (Cremers and Yuille, 2003; Cremers and Soatto, 2003; Cremers, 2003). Noise in motion estimates are modeled as additive Gaussian noise. Motion is estimated as a piecewise continuous function over time. A moving object could thus be represented as a collection of spatio-temporal regions each of which possesses a consistent motion. Object motion could also be estimated at the pixel-level, as in (Strehl and Aggarwal, 2000). Instead of considering regions with consistent motion, object motion a video frame is segmented by estimating the motion of individual pixels. In (Iannizzotto and Vita, 2002), the authors propose a motion detection system based on a closed-loop system comprising a predictor and a controller that detect object profiles in a video as active contours. Object motion is estimated by first predicting an initial object model for the current frame and then recursively modifying it based on processed information. Object shapes are modeled as radially varying 2D closed-loops which change shape depending on object motion. A similar approach is presented in (Peterfreund, 1999), where a Kalman filter is applied to a spatio-velocity active contour (or velocity snake) in order to be able to detect objects in the presence of occlusions and image clutter. The technique applies well to real-time resolution of object motion from the background layer. A maximum likelihood classification algorithm is used in (Hampapur et al., 2005) to separate moving objects from their background and to classify them. By using a multiscale representation of target objects, multiple views of the same object can be used to monitor its movement in a particular spatial network.

In (Comaniciu et al., 2000), the authors describe a real-time method for tracking non-rigid objects in video recorded by a moving camera. The method is based on mean shift analysis of the color distribution of the target object. Based on a distance-based weighted model, the location and extent of the object in future frames can be outlined. An approach for detecting and tracking objects that change size and shape over time using the mean shift technique is presented in (Collins, 2003). The method incorporates the principles of Lindeberg's theory of blob-scale detection for successfully tracking such objects. (Nummiaro et al., 2002) applies a particle filter based on the mean-shift algorithm to the color distribution of the target object in order to establish a target model in order to track the object in subsequent frames. A feedback-loop-based system is described in (Comaniciu and Ramesh, 2000) where a combination of a mean-shift algorithm with a Kalman filter is used to predict the location of the best possible match of an object in the future frames of a video. The mean shift operator serves for computation of the color distribution metric of the target object and prediction of future distributions. Based on the output of the mean shift algorithm, the Kalman filter tries to predict the spatial location of the next instance of the target object. The predicted value is then used to validate the predicted color distribution of future instances.

2.2. Synopsis

Optical flow methods in general yield a very good estimation of motion in an image sequence by using pixel brightness changes to indirectly measure actual motion. Optical flow, however, does not perform very well in such situations where the ambient illumination varies significantly in a short period of time, e.g., a moving cloud could cause sudden illumination changes in a scene. Our approach is an optical flow-based method as it estimates pixel motion directly from image intensity changes. However, in order to account for illumination changes, we incorporate spectral angles instead of raw color values. Bayesian methods often tend to be computationally expensive and are especially unsuitable for real-time applications. The same drawback is true for pel-recursive systems as well. In our work, we only involve simple subtraction at the pixel level in order to estimate the frame differences. For most real-world applications, the temporal relevance of the results of a motion estimation system is important. Hence it is necessary to avoid complex non-linear computation schemes as they are iterated over every frame of a video stream. For an application that demands timely results, it would thus be advisable to keep computational complexity at a minimum at the frame level without compromising on accuracy.

Chapter 3

OUTLINE OF APPROACH

3.1. Video sensor networks

A video sensor network is defined as a spatial network of video sensors. The location of the sensors could be on any surface – land, air, water or a mix of all three. Terrestrial networks usually consist of static cameras (video sensors) placed at well-known locations. Most terrestrial applications involve tracking of objects in a closed environment, like an industrial complex, military installation or museum building. The main objective in such applications is to have an "observation" mechanism for monitoring the movement of individuals within the environment. Depending on the rules set for the system, individuals could be identified as authorized or unauthorized. Their movements could be compared to established motion patterns in a database connected to the monitoring system and if there is any deviation, alerts could be issued. However, not all terrestrial applications need be static sensor-based. Some traffic applications may require the use of sensors on moving platforms (like trucks) to monitor the traffic situation in an area or to analyze traffic patterns. Most aerial applications belong to the domain of homeland security and defense, with the sensors mounted on Unmanned Aerial Vehicles (UAVs) or other similar vehicles.

In order to extract useful information out of a video network, often it is necessary to know sensor parameters. In most cases, all that is required is the camera coordinates and view angle. If it is necessary to transform the recorded data to a uniform scale for data recorded on all sensors, the cameras could be calibrated by first recording a set of known points before starting to gather data. When camera coordinates are known, it could help in accurately identifying absolute (with respect to network) coordinates of moving objects in the network. Using that information, objects could be more reliably matched across different sensors by placing a check on spatial continuity of observed object locations. The view angle (with respect to a known reference line) of the sensor would also be useful in pinpointing object locations in the network.

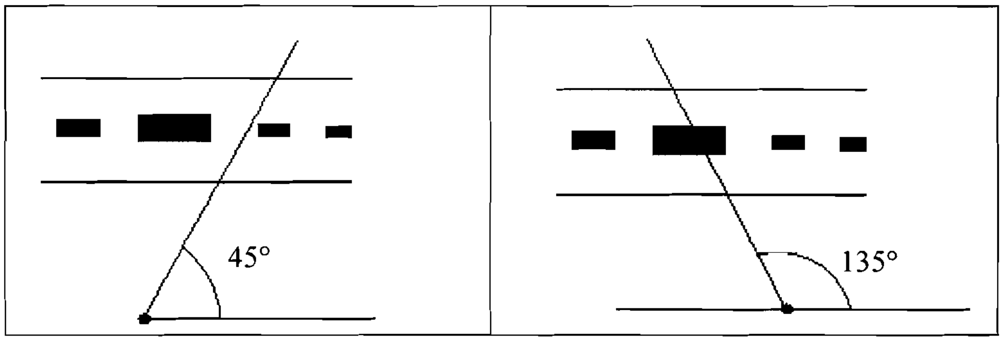


Figure 3.1: Significance of view angle of sensor

With just sensor coordinates at hand, it is not possible to locate objects with reference to network topography. In the Fig. 3.1, let the reference line be taken as East (or to the right of the diagram). If the actual sensor view angles are not known, it is possible to record the objects in the two views as moving opposite directions, even though the directions may be constant in both cases.

Another important parameter is the time at which the data is recorded by the sensor. While it may not strictly be called a sensor parameter, its significance is nevertheless no less. For this information to be useful, all sensors need to be synchronized to the same time. Time-stamped data would be very useful in not only aiding matching objects across sensors but also in building trajectories of moving objects across the network. It is important to know the number of sensors that will be used in constructing the sensor network, as it determines the distribution of the sensors over the network. Further, it might also be necessary to know the spatial and spectral resolutions of each video sensor so that the more precise sensors could be placed in high-activity locations.

In most video sensor network applications, it would be desirable to know the topography of the area where the video network will be setup. This not only is useful in determining the placement of sensors but also in validating the construction of consolidated object trajectories. Depending on the application, sensor placement may differ. In the following figure, a typical video sensor network can be seen. The terrain is represented by the map (in the background) consisting of roads and the location of sensors is indicated by the red dots at various road-junctions on the map.

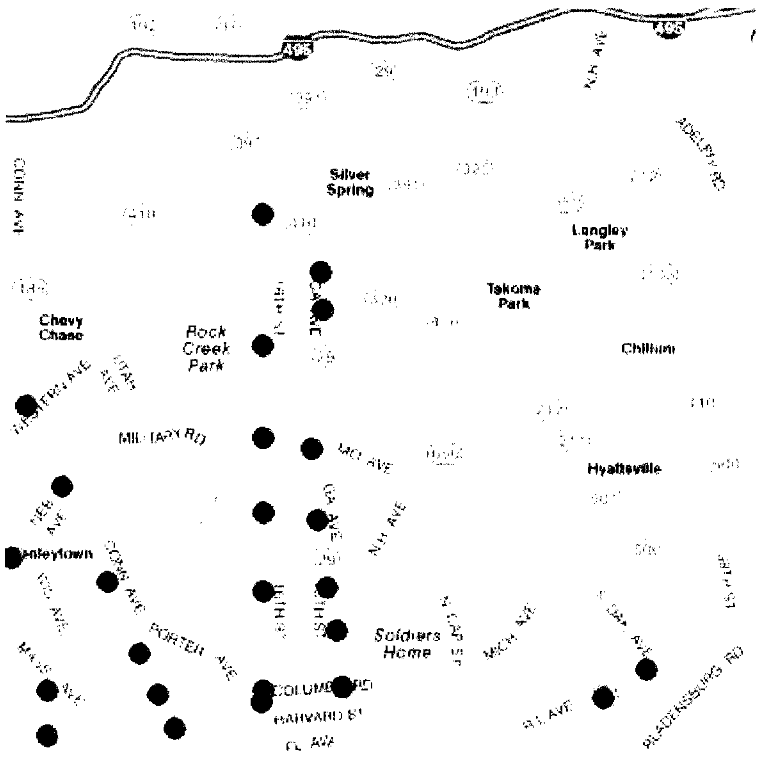


Figure 3.2: Typical video sensor network (courtesy Virginia DoT)

Video sensor networks can be configured in many different ways. Some applications may require a regular placement of sensors in a grid pattern to identify activity in blocks of areas. An example would be a sensitive military installation where it is necessary to monitor the entire complex for any unauthorized intrusion. It would not just be imprudent but also infeasible to expect a single or a few sensors to monitor a large closed area with reasonable accuracy. In such cases, the target area is divided up into a grid composed of regular cells of a fixed size (depending on the needs of the application) and a sensor in every cell. Cells may be defined such that sensors in adjacent cells may have overlapping views. This ensures a continuous yet consistent view of the area being monitored. In other situations, it might be necessary to install sensors in a clustered fashion. This is true of many traffic monitoring applications. In a city, more sensors may be clustered around busy intersections and places where there is a lot of vehicle movement. In residential zones and suburban centers, fewer sensors may be placed to optimize both cost as well as volume of data generated by the network.

While video sensor networks present a wide range of opportunities to collect accurate and temporally continuous information on dynamic phenomena like flowing traffic, the inherent limitations must also be discussed. Videos recorded in outdoor environments are always fraught with all sorts of errors. A source of error in the context of video sensor networks is defined as any entity that either causes data to be recorded with less accuracy than possible or causes extraneous errors in the data. A most frequently occurring source is varying ambient illumination. The implication is that the same moving objects may appear to possess a different surface color in one dataset than in others where they appear too. Another reason for error would be unwanted background motion caused by wind. Trees swaying in the wind could be detected as moving objects too and cause needless system processing. Most errors and noise in recorded video can be removed by using a variety of techniques. Illumination errors could be corrected by normalizing object surface colors while general background noise can be removed by a combination of filters. Nevertheless, most errors can be prevented by placing sensors in locations where there is little chance of ambient factors influencing the recording.

3.2. Analysis of a single video feed

Video processing and analysis is different from conventional image processing in many ways. A video is not only a collection of frames, most of which are related to their neighbors in varying amounts. Individual frames cannot be processed in isolation, as in normal image processing. All processing has to be done keeping in mind the temporal dimension of video. Hence, information derived from frames has to be correlated to construct a temporal sequence of results. This aspect of video processing consumes a lot of computer processing power and also main memory. Therefore, any image processing operations on video have to be thought out carefully, if timely (if not real-time) results are to be obtained. In our approach, we use a combination of frame differencing and morphological image filtering to detect objects.

Human beings can detect moving objects intuitively, whether in real life or in recorded video. The physical process of detection occurs in a sequence of events. The eye acts as a video sensor and collects visual information. The information "seen" by the eye is converted into electrical impulses and transmitted to the brain where it is compared to existing neural patterns. Familiar objects such as cars are "stored" in the brain and are recognizable. Even unidentified objects can be detected by the human brain purely on the basis of difference in motion between the object and its background. Object detection systems work in a similar fashion, even though their operation is rudimentary at its best, when compared to the human visual cognition system. In our approach, we detect objects in a video in a manner very similar to how we perceive motion naturally. The object detection algorithm works with the technique of frame differencing as the central idea.

The general outline of our approach, as it applies to a limiting condition of a single sensor in a geospatial network, is presented in Fig. 3.3 below. After video data is recorded by a sensor, it is pre-processed in order to make it suitable for video analysis. Pre-processing typically involves some form of format (frame dimensions, frame rate or color depth) optimization or general noise elimination (the more specialized form of noise removal, namely, shadow elimination is a later step). Once the raw video data is pre-processed, frames in the video are differenced sequentially and any residue is defined as representing object motion at the corresponding pixel locations. This is the preliminary step in motion detection. After difference images are obtained, only motion from objects must be extracted and the rest (possibly caused by ambient noise or unwanted objects such as tree leaves or pedestrians) can be ignored.

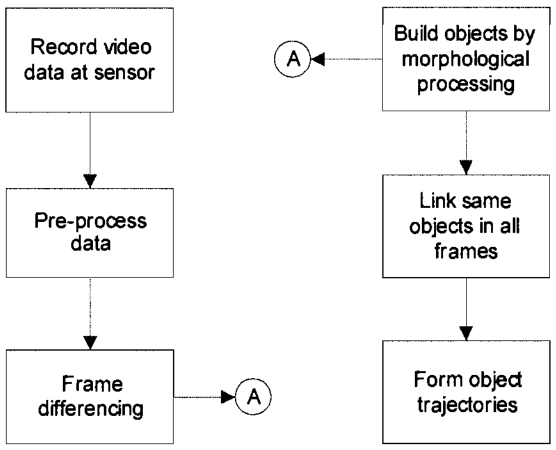

Record video data at sensor
Pre-process data
Frame differencing
A
A
Build objects by morphological processing
Link same objects in all frames
Form object trajectories

Figure 3.3: Analysis of a single video feed

In order to detect which residual pixels are due to object motion, we must define what constitutes an acceptable object, in terms of its expected size. All residual pixels in difference images are outlines of moving objects across two frames. This bit of information is useful in constructing object outlines from these pixels. A pixel lying in a certain spatial neighborhood of other pixels can be associated to the other pixels as a member of the group of pixels which constitute a valid object. A region-growing filter is applied on a difference image to connect all pixels that lie within a predefined neighborhood and label them as an object. In this manner, all objects in a sequence of difference frames are determined and labeled.

Determining objects is only the first step in object motion estimation because the objects in each frame are independent of their instances in other frames. It is necessary to link all objects to their instances so that a temporal sequence of objects can be obtained for the video. In order to link two objects as the same, a more comprehensive means of comparison must be established than the obvious size and shape-based comparison. This is because it is possible to have more than one object with the same size and shape characteristics. Hence, other object attributes must be defined that can effectively link an object in one frame to itself in another. In our approach, we define multiple object attributes in the three domains of spectral, spatial, and spatio-temporal properties. Spectral properties are related to the object surface color. Raw, uncorrected values could give a preliminary idea of the possible matches an object could have in subsequent frames, while normalized values would allow a more accurate pinpointing of the matches. Spatio-temporal properties are crucial only in multiple sensor comparisons. These attributes can be determined from the original video data. Once objects are defined in terms of multiple attributes, an index can be built over the three domains. Indexing is very useful for creating object clusters in the multiple-feed situation. The idea of a multi-component signature forms the core concept in our approach.

Once objects are linked across frames in a video, we need to construct trajectories based on their centroid locations in each frame. The trajectories give the user an idea of the object's motion across the view of the sensor. But its relevance is not limited to that alone; object matching across different sensors can be performed more accurately by validating object matches on the basis of their individual trajectories. Checks can be placed in terms of the maximum distance an object is allowed to travel in between two frames. If the distance between two consecutive links of an object's trajectory in a video feed is beyond the set threshold, the link is severed and the resulting two trajectories are assumed to have been caused by two spatially close objects. A minimum number of links per trajectory can be defined in order to eliminate random noises which might escape preliminary filtering.

3.3. Analysis in a multiple-sensor scenario

Processing a single video feed and extracting objects and constructing their trajectories in space-time is comparatively simpler than it is for multiple feeds. With multiple feeds, there are several factors that change which are usually assumed to be constant for a single feed. While in a single feed the ambient illumination is constant (in most cases), in multiple feeds, it need not be always true. Several factors influence illumination in such a situation. Changing cloud cover could suddenly cause a drop in illumination in one sensor while the rest of the video sensors would still be receiving bright sunlight. Unless calibrated to have a uniform view, multiple sensors will have multiple view parameters. An object may appear twice its size in one sensor than in another due to different focal lengths. Factors such as varying sun orientation may cause shadows in one feed and not in another. View angles could differ too, so this could cause distortion of object shape. Unless sensor parameters are recorded across the network, accurate matching of objects might not be possible. Once parameters are known, objects extracted from each video could be transformed to a uniform frame of reference in the spectral, spatial and spatio-temporal domains.

In our approach, we first established a flow for the processing of a video dataset obtained from a single video sensor. This work was initiated as an internal research problem addressed by the Digital Image Processing and Analysis (DIPA) group at the University of Maine and culminated in a paper outlining its application for car tracking (Venkataraman et al., 2004). Our approach continues from there to vastly improve the process of object motion analysis and afford it a more rigorous generalized model in the form of the multi-dimensional object signatures and object signature indexing. We developed a more general motion detection and analysis process which involves multiple sensors in a geospatial network. As seen in the following figure, the multi-sensor scenario builds on heavily over the simpler single-sensor case. After object trajectories are formed in each video stream, object signatures (attributes) are measured in all the defined domains (spectral, geometric, and spatio-temporal) and multi-component signatures are formed for each object in each video dataset. An object index can then be constructed for each video dataset that orders the objects in a particular order in either a single domain or successively in the three domains based on a set hierarchy.

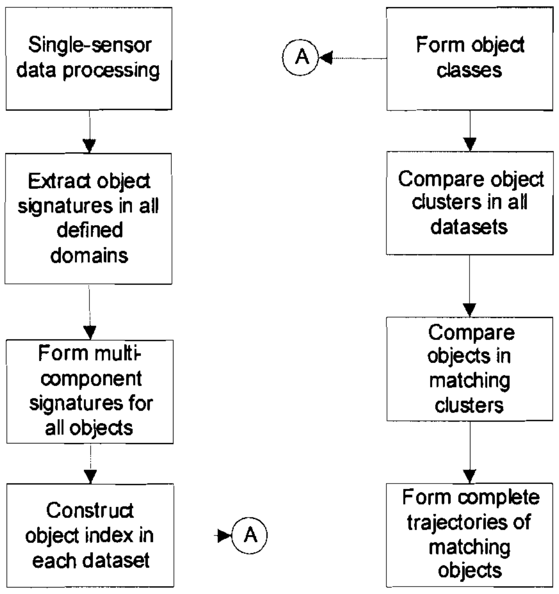

Single-sensor data processing
Extract object signatures in all defined domains
Form multi-component signatures for all objects
Construct object index in each dataset
A
A
Form object classes
Compare object clusters in all datasets
Compare objects in matching clusters
Form complete trajectories of matching objects

Figure 3.4: Analysis of multiple video feeds

Following the establishment of an object index, we proceed to form object classes. Why this step is necessary before direct object comparison can be seen from the following reasoning. With multiple videos, it is necessary to avoid a one-to-one object comparison for all videos. If, for example, there are ten videos covering a geospatial network and each video contains around 1,000 objects. A one-to-one comparison would cost 1,000 raised to the power ten processor cycles! Instead, objects in each video can be clustered into several classes in each domain. While comparing datasets, only the relevant classes of objects need be compared. We can avoid situations like comparing a blue car in one video vs. a red car in another. Restricting comparisons to only relevant groups of objects would result in faster results. Once objects are matched, comprehensive trajectories could be constructed by linking those obtained for the object in question in each video. It is here that network topology comes to play. If we know the position of each sensor in the network and also the orientation with respect to a reference line in the network, we can determine the universal direction of movement of objects. Once the direction is determined, the trajectories can be linked. Chronological information from each sensor can be used to verify the linking. If the sensors in a network are mobile, it adds further processing complexity. Videos have to be processed in order to eliminate background motion in mobile videos. This could place additional overhead on processing and results obtained may not be temporally relevant.

In practice, most video networks have more than one sensor. Motion detection and tracking is most effective when there are multiple sensors monitoring in parallel, for the simple reason that a single sensor cannot track multiple objects at the same time.

Chapter 4

OBJECT TRACKING IN A SINGLE FEED

4.1. Static feeds

Stationary video sensors capture static feeds. In such video data, the sensor captures a fixed portion of the view and the objects moving in the view pass by from one end of the video to another. Most terrestrial video sensors capture static feeds, unless they are designed to track objects in real-time by rotating across the field of view.

After the video has been captured, the process of object detection and motion analysis occurs as a three-step algorithm. In the first step, a certain amount of pre-processing is performed to obtain results in optimum processing times. In the next step, object detection and identification of object attributes as well as motion characteristics takes place. After all objects in the video have been identified frame-by-frame, each object is linked to itself in all frames of the video. The complete trajectories of all objects in the video are then constructed from the individual centroid locations in all the frames they appear. Finally, in the third step, all objects are indexed and clustered into different classes. However, the third step is only important for object tracking in multiple feeds where comparison of objects across feeds takes place. In the following figure, the algorithms and software code for the first column and parts of the second column were conceptualized and developed by the Digital Image Processing and Analysis (DIPA) group at the University of Maine. The concepts of shadow elimination and multi-dimensional indexing for object comparison, as shown in the third column, were developed during the course of our research.

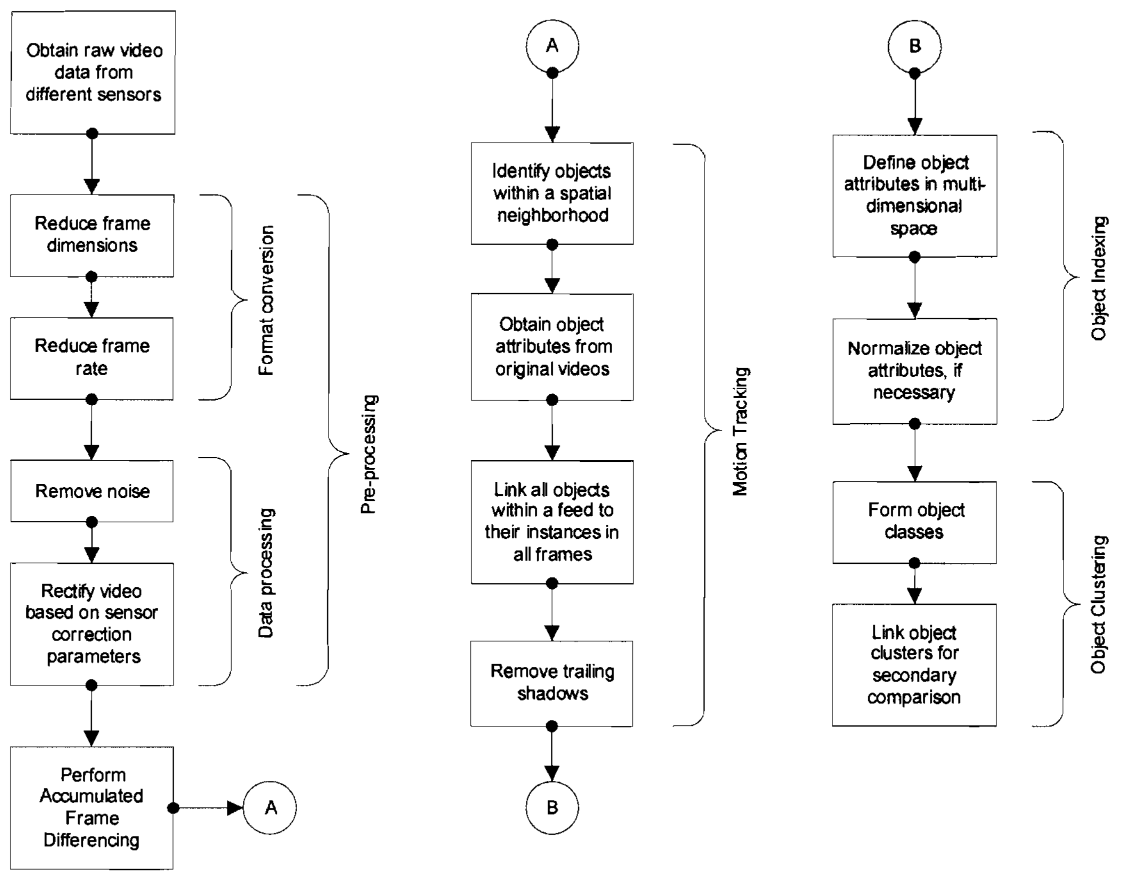

A
B
Obtain raw video data from different sensors
Reduce frame dimensions
Reduce frame rate
Format conversion
Remove noise
Rectify video based on sensor correction parameters
Data processing
Pre-processing
Perform Accumulated Frame Differencing
A
Identify objects within a spatial neighborhood
Obtain object attributes from original videos
Link all objects within a feed to their instances in all frames
Remove trailing shadows
Motion Tracking
B
Define object attributes in multi-dimensional space
Normalize object attributes, if necessary
Object Indexing
Form object classes
Link object clusters for secondary comparison
Object Clustering

Figure 4.1: Flow diagram describing the object motion analysis algorithm

4.1.1. Pre-processing of data

Pre-processing is an important step that must be executed before the object detection process. For a variety of reasons, primarily in the direction of improved response time from the object detection system, we pre-process the raw video data obtained from the sensor. In traditional video encoding, which follows a similar approach of motion analysis and representation but for different reasons, pre-processing has been shown (Agazi et al., 1995) to be an important step in video processing in order to improve system performance. On the same lines, it is also important in our application because the bulk of video processing in our work is very similar to a video encoding application.

4.1.1.1. Change spatial resolution and frame rate

Commercial video sensors capture video at a uniformly high quality with a fixed frame rate. Commonly, data captured from such sensors need to be scaled down in situations where a high level of detail is unnecessary, e.g., a situation where the sensor is close enough to the moving objects in order to allow usage of lower than normal resolutions for motion analysis. What constitutes as acceptable in a certain situation is dependent on the application as also the field configuration of the sensors. In many cases, sensors do not allow for multiple options in resolution. To optimize the video processing in terms of frame dimensions, the end-user needs to specify the format size they expect to be the optimum for their application. It is important to set a format size such that the motion detection algorithm does not incorrectly label the smallest object in the video as background noise.

It might also be advisable to alter the frame rate of the source video so that only those frames that are essential need to be retained. This is very useful in situations where there is not enough movement in the captured scene to warrant processing of the entire set of frames in the original dataset. For example, in a traffic monitoring setup, it is not necessary to capture data at the normal rate during peak hours as the traffic will be slow-moving. If captured at full frame rate, a lot of processing cycles will be wasted in processing redundant frames, whenever objects move very slowly or even stop.

4.1.1.2. Projection of data based on scale factor, view angle etc.

Data captured by a sensor at an angle to the scene needs to be transformed to an orthographic view or some uniform frame of reference so that data captured by different sensors can be compared reliably for geometric properties of the moving objects in those feeds. Using correspondences in a video frame to another system of coordinates (map with local coordinate system), we can transform all record object positions and geometry to a uniform frame of reference. Transformation also needs to take into account any scale changes due to sensor view angle. If there is a known scale factor for the data captured, it needs to be incorporated in the process of comparing objects in different video datasets. Otherwise, geometric size-based comparison will be unreliable. Transformation is usually only necessary in multi-sensor object tracking. It might be useful in single-feed situations where either illumination, view angle or view scale change during the process of video capture.

4.1.2. Preliminary noise removal

Random noise in captured video has to be removed so that it is not detected as a moving object (or group of objects) by the algorithm. While data is being captured, individual frames could be passed through a noise filter. However, it must be noted that excessive filtering could blur the edges of objects or alter their geometric profiles.

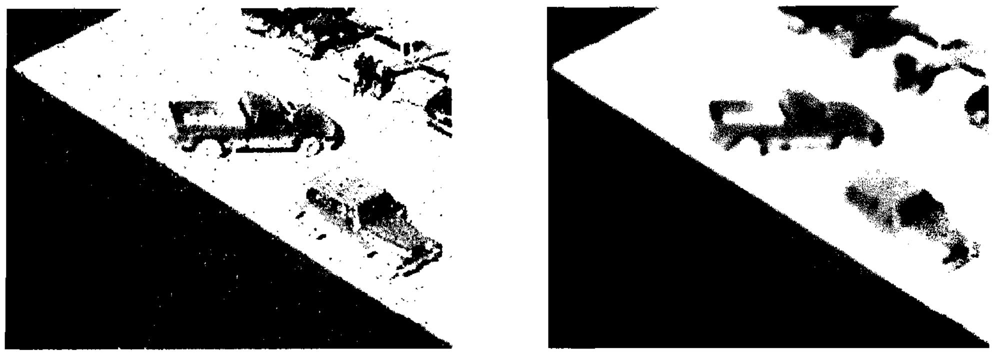

(a) (b)

Figure 4.2: (a) original noisy frame; (b) noiseless output after applying median filter

In the above figure, the image on the left shows a frame from a video containing random salt-and-pepper noise. In order to remove the noise, we applied a median filter. While the filter removed the noise, it also blurred the edges of the objects. This could present difficulties during the process of object matching.

4.1.3. Motion detection by means of frame differencing

The core process involved in motion detection is called Accumulated Frame Differencing (AFD). AFD is an iterative process that operates over sequences of frames in a video stream. It determines motion on the basis of change in pixel color by subtracting or 'differencing' a sequence of frames. In practice, pairs of frames are differenced to obtain differenced images and these images are subjected to a threshold to filter out unwanted objects and any random noise. PID is the difference in color values for the same pixel (x,y) in the scene grid over time. PID is computed between the reference and all other frames in the video stream as follows:

$$PID = R(x,y) - f(x,y,t_k) \tag{4.1}$$

where $R(x,y)$ is the reference image and $f(x,y,t_k)$ is the frame at time t_k. Usually, the reference is the first frame. The Differencing Mode (DM) can be "positive", "negative" or "absolute". The three modes differ in the manner in which the pixel intensity difference (PID) is compared with a pre-defined gray level threshold (GLT). GLT is the minimum gray level change that must occur for an object to be detected as moving across two frames. This value is scene-dependent and is empirical. A new value for each pixel in the currently processed frame is obtained depending on the presence/absence of motion in the neighborhood of each pixel. The new value is binary and is allocated as follows:

$$I_k(x,y) = \begin{cases} I_{k-1}(x,y)+1 \\ I_{k-1}(x,y) \end{cases} \tag{4.2}$$

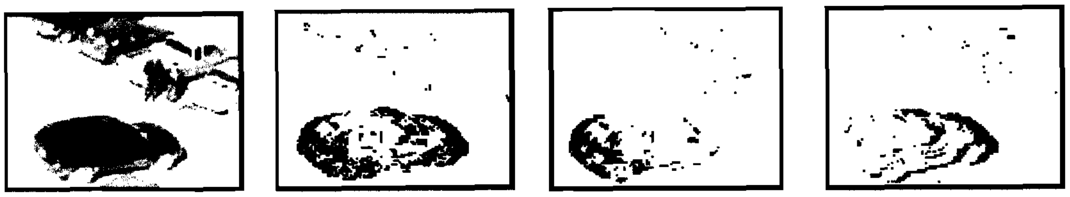
where : $I_k(x,y)$ is the pixel value at location (x,y) of the current frame and its value is dependent on its value in the previous frame $I_{k-1}(x,y)$. Pixel values are either incremented by one or left unchanged from the value in the previous frame depending on the differencing mode. In the "positive" AFD mode, the differenced image is incremented by one at the pixel location (x,y), if PID is greater than GLT. In "negative" mode, the increment is by one if PID is less than $-$(GLT). In the "absolute" mode, the increment is made if the absolute value of PID is greater than GLT. The three modes are described in mathematical form below:

$$Abs_k(x,y) = \begin{cases} Abs_{k-1}(x,y)+1 & \text{if } |\text{PID}| > \text{GLT} \\ Abs_{k-1}(x,y) & \text{otherwise} \end{cases} \tag{4.3}$$

$$Pos_k(x,y) = \begin{cases} Pos_{k-1}(x,y)+1 & \text{if PID} > \text{GLT} \\ Pos_{k-1}(x,y) & \text{otherwise} \end{cases} \tag{4.4}$$

$$Neg_k(x,y) = \begin{cases} Neg_{k-1}(x,y)+1 & \text{if PID} < -(\text{GLT}) \\ Neg_{k-1}(x,y) & \text{otherwise} \end{cases} \tag{4.5}$$

The following figures illustrate the three modes of differencing:



(a) (b) (c) (d)

Figure 4.3: (a) Original frame; vehicle moving in southwesterly direction (b) Absolute AFD output (c) Positive AFD output (d) Negative AFD output

As illustrated by the above figure, an absolute AFD guarantees complete outline of the object when the video is subjected to the AFD process, whereas positive and negative AFD preserve only the leading and trailing edges (with respect to direction of motion) of the object's shape profile. In addition to the mode, two other parameters influence AFD. The first, frame accumulation rate (FAR), determined iteratively, is the number of frames over which the difference from the reference frame will be accumulated. The second, accumulation threshold (AT), is the minimum number of frames across which the object must exhibit change in order to be detected as moving. This parameter helps in removing periodic noise. Its value was selected such that noise was removed but slow moving objects were preserved.

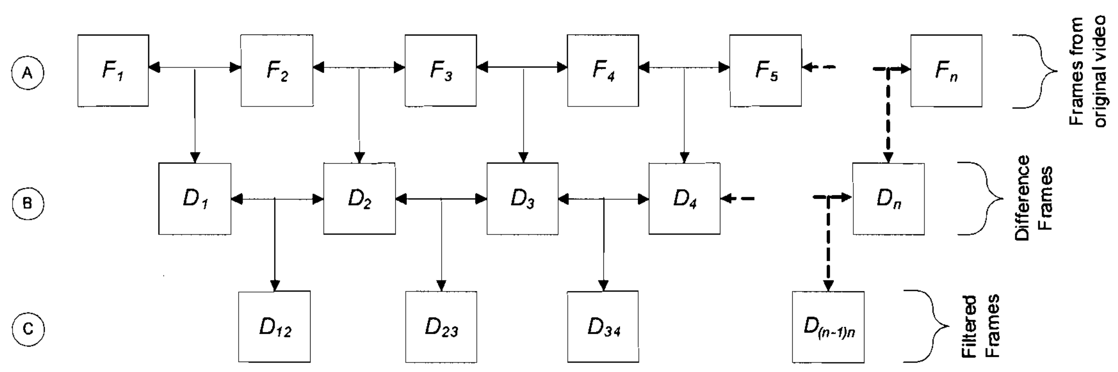


Figure 4.4: Flow diagram describing frame differencing algorithm

In step A, all the frames are differenced in pairs, in accordance to the [eq. on AFD]. For example, frames F_1 and F_2 are differenced and the resulting frame is compared with the GLT. Based on the individual pixel values in D_1, they are assigned either "0" or "1". Then, the next pair, F_2 and F_3 is compared and the difference frame D_2 is generated. Likewise, all frames until F_n are subjected to this process. So, for n frames in the original video, (n-1) difference frames are obtained. In step B, the difference frames are summed up according to the value of AT. For example, if AT = 3, frames are added together in threes and the resulting frame is compared to the AT. In the above diagram, the value of AT is two and so frames are added in pairs. All values less than that of AT are assigned “0” and the rest of the values are retained. In step C, all the filtered frames are grouped together to form the output of the motion detection process. At the end of the process, there are (n-AT) frames in the output video.

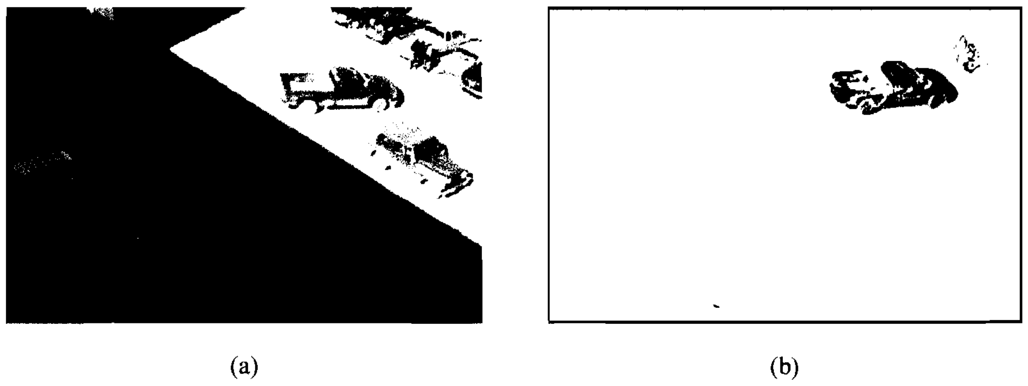


Figure 4.5: (a) original video containing a few moving objects (red truck and two people); (b) raw AFD output of the original frame (black border artificially added to delimit AFD frame)

4.1.4. Shadow removal

To identify an object unambiguously across two video feeds, it is important to isolate a representative object color value that can relate to the object in both video streams. This is only possible if object shadows are eliminated and pure object surface colors are identified. If shadows are not eliminated, they could appear as extensions of the object being detected by the motion tracker algorithm. This could cause errors in determining the exact area and shape of the object while comparing with other objects in other video streams.

The apparent surface color of an object is determined by source illumination, viewing geometry and camera parameters (Raja et al, 1998). Processing color information can be expensive and is typically restricted to the pixels that have been obtained from the morphological operations on the object shape profile. It becomes computationally intensive because of the fact that for the same pixel location, any computation is on three domains (either R,G,B or H,S,I). Color information can be operated upon either in the RGB (Red-Green-Blue) domain or in the HSV (Hue-Saturation-Value) domain depending on the needs of the user. In the RGB domain, obtain color information consistently is not easy since all the three values (R,G,B) change significantly with illumination.

In the HSV domain, the hue value plays a vital part in object determination since it does not vary with intensity changes. This is particularly true when the object is subject to varying illumination levels as under shadows and under bright sunlight. The advantage of using hue values is that objects can be detected reliably in any kind of illumination condition. However, they cannot be obtained directly from a raw video stream. Most video sensors acquire video data in the RGB color domain and therefore, a conversion to HSV would be necessary before any further processing. But this could place a processing overhead on the system. In most situations, illumination is almost constant for a single sensor. While comparing datasets from two different sensors (at different locations), there is a very good chance that they might have been exposed to different levels of illumination. Even in such a case, any sort of a conversion from RGB could be performed after objects have been detected as this could save processing time.

After the objects have been detected as distinct individual groups of pixels in a frame, a mean representative color value must be obtained for the object. However, this cannot be done before removing the shadow detected as part of the object. The object detection process cannot differentiate between an object and its shadow in most cases because the shadow is often continues from the object perimeter without any intervening gap and also "travels" at the same speed as the object. Therefore, the pixels representing the object's shadow must be eliminated; in actuality set to a value or "0" or whatever value was chosen for the background. To go about this process, we define a bounding box enclosing the detected object. Pixel values that are outside the object outline but within the bounding box belong to the surface on which the object traveled, in most cases, a road. A set of empirical values for the hue value of an average road surface was chosen. It was assumed that the hue values of the shadowed region would be very close to the hue values obtained from the road pixels as the shadow is merely a region on the road with an illumination level different from its surrounding areas.

Once the range of hue values corresponding to the shadow was identified, those pixels in the vicinity of an object with hue values falling in the 'shadow range' were eliminated. These pixels were identified by the motion tracking algorithm as part of the object as the shadow moved with the same velocity as the actual object. Once the shadows were eliminated, object hue values obtained were more relevant to the actual object surface color. This helps in linking object trajectories across the two datasets with greater accuracy.

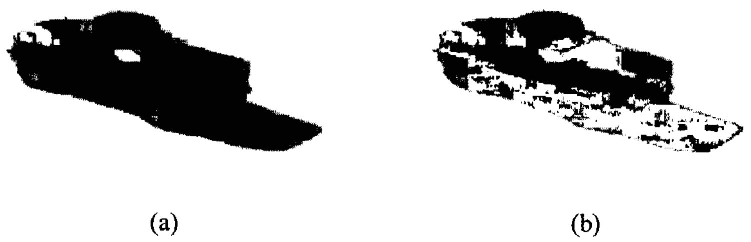


Figure 4.6: Object before and after shadow elimination

As can be seen from the above figures, the shadow removal algorithm has eroded parts of the object as well, because of similar hue values found on the surface of the object. To further increase the accuracy of shadow removal, a simple neighborhood filter could be used to remove shadows as a connected sub-object, rather than removing individual “shadow” pixels purely based on their hue values.

4.1.5. Morphological processing of differenced data

Morphological parameters isolate objects that satisfy a certain geometric criteria. These parameters are essential in detecting just those objects that are needed. The *structuring element* is a bounding box use to establish connection between neighborhood pixels with an object. This element is like a spatial filter that links adjacent object pixels and is the basic object-building mechanism. Depending on the dimensions of the filter, greater accuracy can be obtained in defining the extents of an object. However, this comes with a price as the processing time increases in a geometric progression with each increase in dimension. The *minimum (mOA) and maximum (MOA) object areas* (pixels) constrain the allowable object sizes. These could act as a filter in restricting the objects detected and retained to only relevant classes. For example, if we are interested in extracting only cars from a video stream, then any bicycles and trucks would be unwanted information. The *object compactness (C)* is a shape metric that defines the elliptical shape of the object:

Vector building parameters also control the creation of motion vectors of the filtered objects. The *Maximum link distance* specifies the maximum distance in pixels that the object centroid can change between successive motion frames. This makes sure that two similar but separate objects in the same video are not linked inadvertently. The *Maximum spectral distance* specifies maximum allowable variation in any color metric between successive motion frames. This ensures that two closely-spaced but differently colored objects are not seen as the same by the object detector. In a predominantly color based test, the link distance is set to a large value and the spectral distance becomes the key linking parameter.

4.1.6. Object labeling and parameter definition

The typical scenario for our work consists of a traffic network monitored by multiple video cameras at different locations. The cameras that comprise this geosensor network may be static (e.g. located on buildings or traffic poles) or moving (e.g. attached to moving vehicles). In either case their instantaneous location and orientation is considered available, provided, for example, by GPS sensors attached to them. Vehicles moving within the area of interest monitored by our network pass through the field of view of one or more cameras at an instance. In order to model and link activities within this network we make use of two types of information:

- Network parameters allow us to index feeds (datasets) within our network, while
- Object attributes are used to index objects extracted from individual feeds.

It is important to note here that any two sensors may or may not share a field of view. For the most part, we work under the latter assumption. When there is no continuity (as a mosaic of sensor FOVs) in the area being viewed, it becomes more difficult to label and reliably link objects across feeds. The reason is that there may be some objects that appear in one sensor view and may not in the others. In any case, the object count in one sensor need not be equal to that in any other because the sensors are all assumed to have a disjoint view of the terrain.

4.1.6.1. Network parameters

Global parameters allow us to position different feeds relevant to each other in space and time. Accordingly, they comprise the spatial coordinates of a sensor's field of view, and the corresponding temporal information (timestamp). The coordinates of a sensor's FOV are important in determining the view angle and transformation parameters necessary for normalizing objects in all feeds before comparing them to one another. Timestamping is very important because it acts as a validating piece of information. Similar objects that appear in two disjoint sensor views at the same time could be ruled as the same because it is not possible for an object to be physically present in two different locations.

Sensor-level parameters are intrinsic (i.e. sensor calibration information) and extrinsic (i.e. sensor orientation information). The intrinsic parameters may be fixed or varying. For example, a single sensor may have the capability to function in more than a single band (e.g. visible and thermal), depending on scene illumination conditions. Similarly, the spatial resolution of the sensor may be fixed (in the case of a fixed sensor) or varying (e.g. when a sensor is mounted on-board a moving vehicle). Focal length, on the other hand, may typically be kept fixed. Extrinsic sensor parameters comprise information that allows us to position the sensor in space (e.g. altitude, viewing angle).

In situations where the sensor parameters are not known in advance, or if the sensor is not equipped to report these data, the object transformation parameters could be derived by calibrating the camera against a measured object in its FOV or by recording the positions of a set of known points in its view.

4.1.6.2. Object spectral attributes

The surface color of an object in either the RGB (red, green, and blue) or the HSV (hue, saturation, and value) domain is a fundamental attribute for comparison with other objects. The representative RGB color is derived in a manner similar to that described in (Wei, 2002) but we first eliminate noises that are associated with extracted objects such as shadows. In (Venkataraman et al, 2004) we described a shadow elimination process used in our approach. In order to index color content, we transform color information into spectral angles. The concept of spectral angles derives from the color indexing scheme described in (Stefanidis et al., 2003a). The spectral angles that correspond to the representative color are derived as follows:

$$R = \arccos((\frac{2 \times s \times (s-r)}{b \times g}) - 1) \quad (4.6)$$

$$G = \arccos((\frac{2 \times s \times (s-g)}{r \times b}) - 1) \quad (4.7)$$

$$B = \arccos((\frac{2 \times s \times (s-b)}{r \times g}) - 1) \quad (4.8)$$

where,

$$s = (\frac{r+g+b}{2}) \quad (4.9)$$

Two angles are sufficient to describe the color. The color components (r,g,b) are square roots of the sum of the squares of the other two color coordinates. By color coordinates, we mean the raw coordinates (R^m, G^m, B^m) of the object representative color in the three color axes.

$$r = \sqrt{B^{m^2} + G^{m^2}} \quad (4.10)$$

$$g = \sqrt{R^{m^2} + B^{m^2}} \quad (4.11)$$

$$b = \sqrt{R^{m^2} + G^{m^2}} \quad (4.12)$$

The figure below describes how the three color components (r,g,b) are related to the three spectral angles (R,G,B). In essence, the coordinates of an object's surface color form a triangle in three-dimensional spectral space. Any illumination changes would be reflected as a change of scale of the triangle's dimensions.

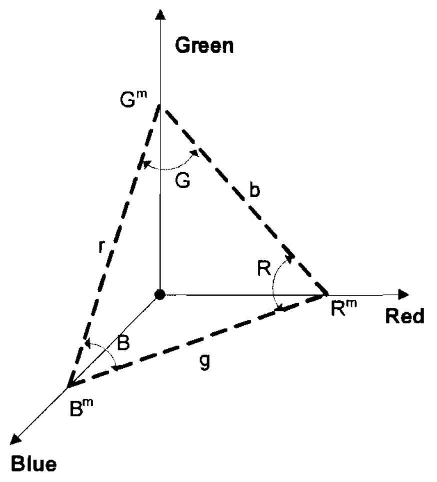


Figure 4.7: Surface color of an object depicted in 3-D RGB colorspace

The advantage of transforming color into spectral angles is in the independence of angles from the effects of illumination. Varying illumination can 'shift' the surface color of an object (as recorded by a sensor) in the colorspace in either direction. Hence, the same object in changing lighting conditions could have several recorded surface colors. The angles derived from each of these colors would still be the same, regardless of the shift that the original surface color underwent in the recorded data.

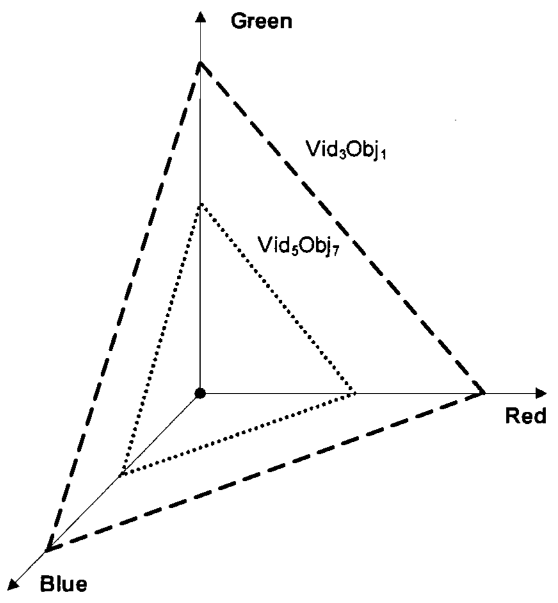


Figure 4.8: Spectral triangles formed by two different surface colors of the same object

In the figure above, there are two objects, Vid$_3$Obj$_1$ (shorthand for Object #1 in Video #3) and Vid$_5$Obj$_7$. From the spectral profiles, we can see that both represent the same object but with similar surface colors. They both have been measured under different incident illuminations and hence, the difference in the profiles. Both their spectral triangles are, however, similar. The scale difference is due to illumination changes. A way of numerically comparing two objects subject to different lighting conditions is to check if the spectral angles derived from their mean surface colors are the same. In reality, there is a slight amount of variation even in the spectral angles. This could be due to errors in computing the mean surface color itself. The errors could arise from inclusion of shadow pixels or pixels from surrounding regions. To accommodate for any unaccountable random errors in computing the mean surface color, a threshold could be defined within which the comparison of two objects could be made.

4.1.6.3. Object spatial attributes

Compactness describes the 'roundness' of an object in comparison to a circle. It is a shape metric that compares an object to a circle which has its perimeter equal to the area of the object.

$$\text{Compactness} = \frac{4\pi a^2}{p^2} \tag{4.13}$$

where a = area of the object, p = perimeter of the object. For a perfectly circular object, the compactness is 1. Compactness could also be seen as the ellipticity of the shape of an object.

Object size is another key attribute which could be used to fix matches. It is approximated by multiplying the size of the object (in pixels) by the scale factor of the corresponding sensor. This can be used for an approximate comparison of two objects if the view scales are known for the sensors in question. Object sizes are also important in discriminating between classes of objects. A motorcyclist could travel at the same speed as a car and a pure color-based or speed-based comparison could incorrectly link them both as the same object. A simple size comparison could add more reliability to the comparison process.

Lastly, geometric information also includes the object's dimensional ratio. Dimensional ratios are the length-to-height, height-to-width and width-to-length ratios of an object. These ratios can be used as an auxiliary metric for object comparison. However, their use in indexing and comparison is only auxiliary due to the low accuracy with which they can be determined.

4.1.6.4. Object behavioral attributes

These parameters describe the spatio-temporal properties of the object like velocity, acceleration or motion pattern. They are part of our model of spatiotemporal helix (Stefanidis et al, 2003b). These properties are useful for identifying objects that behave erratically or for clustering objects that have a certain kind of behavior. Deng and Manjunath define motion parameters as a quantitative measure for object indexing and segmentation in (Deng et al, 1998). Our goal is to utilize the information obtained from spatio-temporal analysis of data in organizing data qualitatively. An example would be a common traffic scenario composed of vehicles of different sizes and motion characteristics such as cars, trucks and buses. Based on expected and observed characteristics, vehicles could be classified as slow moving, fast moving etc. Further, alerts could be raised when vehicles do not meet their defined characteristics. For example, a school bus that speeds frequently.

One spatio-temporal attribute of primary importance is obviously the speed of the object. Speed values usually are dynamic, meaning they change over time and a single representative value can hardly be consistent even within a few dozen frames. However, under the assumption of constant motion (for example, in a traffic scenario) we take the average value for an object's speed vector in a video stream. In most cases, this is what is required in identifying objects across different feeds. Another attribute that bears some relevance is the acceleration. The standard deviations in speed and acceleration could be used in clustering objects. The direction of movement is also an important attribute. It validates the matching of objects in two videos. For example, two objects moving in opposite directions in two video feeds cannot obviously be the same, unless we are certain that in one of the videos, the object has made an about-turn.

In applications where we need to check for any sudden increases in speed or frequent stops, a temporal vector of the object's speed and acceleration would be useful. Further, it could also prove useful in making a secondary classification of objects as periodically halting, fast accelerating and so on.

4.1.7. Intra-feed object linking

Once objects are detected in each frame of a video feed, they need to be linked temporally with their counterparts. This is because objects detected in each frame are not automatically connected to themselves in subsequent frames.

4.2 Mobile feeds

Mobile feeds are captured when there is relative motion between the sensor and its FOV. However, it is the sensor that moves in relation to the ground, whereas the terrain (except the moving objects being imaged) is stationary all the time. This is true in the case of sensors mounted on aerial platforms such as balloons or Unmanned Aerial Vehicles (UAV). Mobile feeds appear as a moving window over a group of objects (also moving). The goal of obtaining most mobile feeds is to track one (or more) moving object(s). While in static feeds, the sensor maintains its static state with respect to the terrain on which the objects are moving, in mobile feeds, the sensor attempts to be stationary with respect to the objects moving in the terrain. The result is that in static feeds, the same object never appears twice in the video (unless it takes a U-turn and goes back in the opposite direction) while in mobile feeds, the same object continues to occupy the FOV of the sensor. Mobile feeds are increasingly used in defense and security applications where it is crucial to keep tracking a group of vehicles (maybe a convoy of vehicles carrying VIPs or a group of enemy tanks in a war) constantly over a period of time.

Processing mobile feeds can be challenging unless special pre-processing is performed first. The first problem, algorithm wise, in processing mobile feeds, is the fact that the background of the video is constantly "on the move". In differencing algorithms like the AFD, the assumption is that of the background being stationary all the time so that a difference between two successive frames would only yield the object pixels as the background pixels in both frames would cancel out. But in mobile feeds, at best, only a large overlap of background between successive frames can be expected. Hence, parts of or the whole background could be detected as one big moving object. What complicates it further is that the objects on the ground also move a little bit with respect to the sensor. In practice, it is not possible for an aerial sensor to maintain perfect synchrony in relative stationariness with the objects moving on the ground. The ground velocity of the sensor could either be more or less than that of the moving objects and so the objects in the mobile feed would move with a small positive or negative velocity. And if a group of objects is tracked by the sensor, they may all not move at the same speed and therefore, it is virtually impossible for the objects to be stationary with respect to the sensor. It is also important to note that when objects change direction of movement, the sensor also changes its view only after a small delay.

4.2.1. Image warping

Under assumptions of simple linear motion, the video could be "warped" such that the background is kept fixed in stretches where there is significant overlap between successive frames. Groups of such frames in a mobile video could be constructed in which the video could be cropped to only show the area that is common to all frames and the objects appear as moving from one end of the frame to another. However, the cropping must not be done at the expense of eliminating a few objects in the extreme frames of the group. In order to warp the frames, common points must be identified and their locations could be used as "tie points" to mosaic sets of frames. Once such sets have been formed, motion trajectories of the moving objects could be formed by connecting the sub-trajectories obtained from all the groups. The only issue with this method is the time consumed in manually selecting ties and selecting extents of groups.

4.2.2. Mosaicking

Instead of mosaicking the frames in a mobile feed manually, they could be overlapped automatically. A feature detection algorithm could be used to identify common background features in two adjacent frames and they could be grouped as long as some predefined overlap threshold is met. For example, a 60% overlap could be set as a possible threshold in binding frames. Anything less than that would automatically form the start of a new group of frames. In the following set of pictures, the frames have more than 60% overlap of background area. Hence, these can form part of a "group".

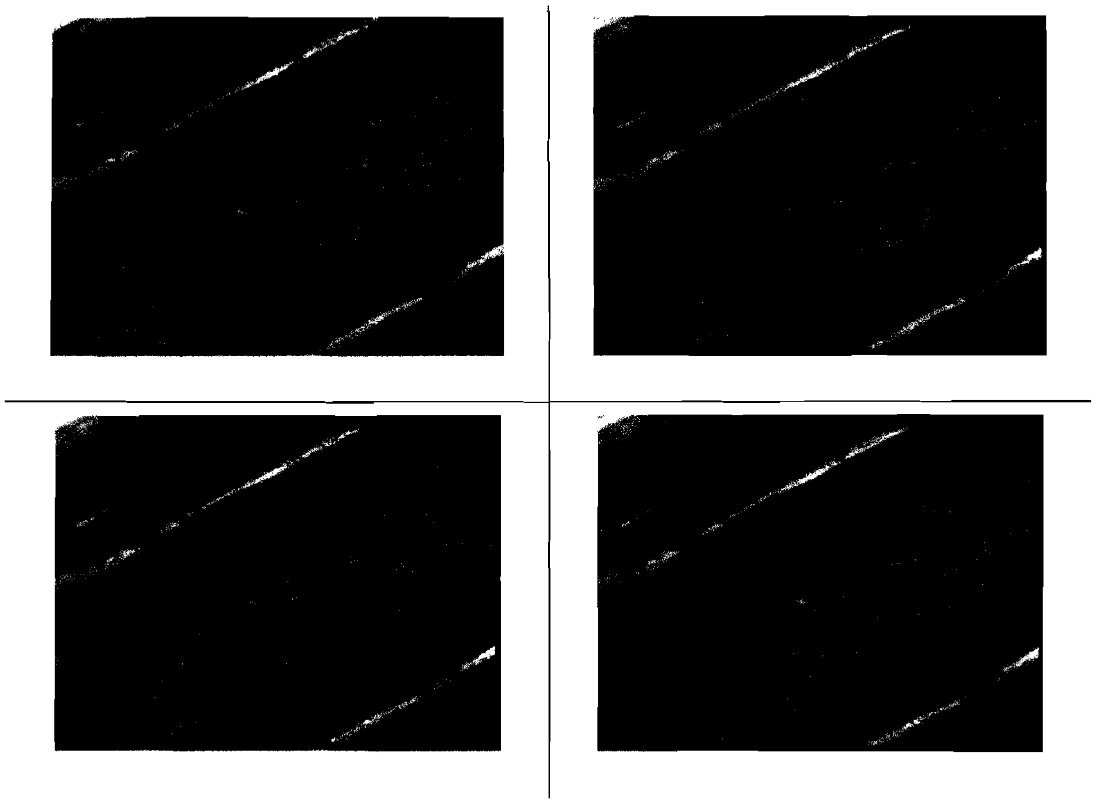


Figure 4.9: Frames extracted from a video feed, captured by a mobile sensor, which shows correspondence in background features

Chapter 5

OBJECT LINKING IN MULTIPLE FEEDS

5.1. Statement of problem

Linking objects across multiple feeds is a far more complex problem than for a single feed. In a single feed, there is continuous coverage of an object's motion across the field-of-view of the sensor. In the case of multiple sensors, more so in the case of disjoint video networks, there could be gaps in the coverage of the terrain by the sensors. While this might seem a trivial issue initially, the potential problems are many. When there are no common areas in the FOVs of two sensors, it is not possible to match object trajectories purely based on feature correspondence in the two FOVs. A certain degree of overlap can aid in the transformation of object profiles to a uniform scale based on the relationship between the two instances, in the two FOVs, of the same background feature. Whereas, with no overlap, camera parameters must be known explicitly in order to normalize object attributes for accurate comparison. Further, since multiple disjoint sensors are located at a considerable distance from one another (for getting distinct views of the terrain), there is a high possibility of local errors due to illumination changes, weather phenomena such as wind, influencing the quality of data recorded by the sensor.

5.2. Proposed approach

In order to effectively tackle the problems posed by installing multiple sensors with non-intersecting views, we propose to establish a system of addressing moving objects using multiple attributes. Using the object attributes extracted from a video dataset (based on the procedure described in Sec. 4.1.6.), we construct a composite object signature that will effectively create a highly-reliable metric for object comparison across multiple video feeds. Object matching will be more reliable using multiple object descriptors rather than a single attribute of varying data quality, e.g. surface color. The following table outlines the object attributes that will be used for building the multi-dimensional signature:

Domain	Attribute	Description	Unit
Spectral	Color.R	R component of object mean surface color	(number)
	Color.G	G component of object mean surface color	(number)
	Color.B	B component of object mean surface color	(number)
	SpAng.R	R component of object spectral angle	radian
	SpAng.G	G component of object spectral angle	radian
	SpAng.B	B component of object spectral angle	radian
Spatial	Area	Area of object profile	pixels
	Compactness	Compactness of the object	(number)
Spatio-temporal	AvSpeed	Average speed of the object	pixels/sec
	AvAccel	Average acceleration of the object	pixels/sec^2
	Dir	Direction of the object motion	radian

Table 5.1: Listing of the object attributes used for building composite object signature

When a single attribute alone is chosen for indexing an object in a video feed, there is very less room for flexibility in case the desired accuracy is not available in that attribute. The proposed multi-component object signatures seeks to solve that issue by affording flexibility to the user in determining which attribute is to be accorded higher precedence based on the quality of the data. For example, if surface color is perceived as the most accurately (in terms of data resolution) descriptive attribute, it could be given a higher weight than other attributes in the object signature. The implication is that the object comparison will be more reliable as the less accurate attributes will receive lower importance for their matching results. Hence, object matches will be much better than with using just one attribute.

Various approaches (Cohen and Medioni, 1998; Lipton et al., 1998; Comaniciu et al., 2000; Pless et al., 2000; Amer, 2005) have been presented in the computer vision discipline on the detection, classification and analysis of object motion involving just one or a few object attributes. Our method differs from others in that we seek to solve the problem of object detection, tracking, and matching by means of a multi-pronged approach that involves establishment of a comprehensive object definition model by means of integrating multiple attributes along with their accuracy weights. Our work proposes a technique for identifying, labeling and analyzing the motion of objects by constructing a descriptive multi-level signature for the moving objects detected in a video stream.

5.3. Object linking

In the initial stages of object detection, we do not know the exact relationships between the objects detected in one frame with those in adjacent frames. While we realize that most, if not all, objects in two consecutive frames are the same; it is not possible for the object detector to automatically link the same objects across frames without extra information on the objects. At this juncture, the detector only knows two attributes of all objects – their size in pixels and the shape (outline) as derived from difference images. This might be enough information in a scenario where all objects in a video feed are of different shapes and sizes. However, in real-world situations, this is rarely the case. For example, in a traffic video, at least a few cars will look alike in several aspects – color, shape, or size or all. In such situations, we need to know the attributes of objects in more than one domain in order to link them reliably within a feed to their counterparts in other frames. We collect information on objects in three domains – spatial, spectral and spatio-temporal. Once all the attributes are known for all objects, we order objects over a possible range of domain-values. While this is not so much to enable intra-feed object linking, indexing assumes great importance while clustering objects in multiple-feed object linking (matching). Grouping indexed objects is easier than trying to cluster an unordered list of objects.

5.3.1. Intra-feed linking

Objects detected in each frame need to be linked to themselves in subsequent frames. Comparison is based on one attribute to speed up process of generating list of objects in a single video stream. Color is a suitable candidate for comparison because of the high level of accuracy with which objects can be labeled. Further, color is not affected by the sampling quality (spatial resolution) of the data too much because it is a mean measure over the object's surface as compared to geometric attributes which depend on the accuracy with which the raster image (frame) represents the actual shape profile of an object in the video. The reason why single-attribute comparison is used is because the ambient variations in illumination are almost negligible for frame-to-frame comparison. Knowing an object's attribute and its location in the frame, it can be compared with other objects in the next frame. The nearest object that has the same or nearly same attribute-value is its image in the next frame. In situations where there is a great deal of similarity of objects in one domain, say color, more attributes from the other domains could be used in linking objects. Using a single domain in such a situation would result in multiple links for one object in the next frame. But such a potentially confounding situation can be resolved by enforcing a proximity parameter that ensures that objects cannot appear at gaps (between two frames) more than the specified threshold. It is for this purpose that we define a vector-building parameter called maximum link distance which defines whether two temporally-adjacent centroid locations represent the same object or not.

5.3.2. Multi-feed linking

Linking (or matching) objects across multiple video feeds is far more complicated than a simple intra-feed linking. Several factors which are usually constant for a single video feed become highly variable for a set of video feeds. The view characteristics of the individual sensor is an important factor which is seldom constant in a video sensor network. Further, local illumination variations may be pronounced with multiple sensors. To tackle the problem of multi-feed linking, we propose the additional steps of object indexing and clustering. Once these processes are complete, actual object comparison can proceed and matches can be obtained.

5.4. Object indexing

Indexing the moving objects detected in a video feed is important for the purposes of clustering. Indexing could be a simple ordering of objects in each domain in a particular order. Such an index could be maintained for each domain. Alternatively, for larger object databases, it could be important to create a dynamic index (based on a B+ tree scheme, for example) so that frequent updates to the database are handled in an elegant manner. However, in our case we consider a simple ordered index of objects as it serves our purpose well. In a real-time application, objects are always added to the database. Any deletions or updates would occur only at a separate level. The application itself would not demand any changes to the data collected over a period of time. However, use of a special indexing mechanism (like the B+ tree) entails the use of a dedicated database system. In our case, we have restricted our choice of an index to an ordered list primarily because including a database system is beyond the scope of this work. However, using a database would be a tremendous boost in the direction of data management. For a real-world application involving moving objects, like traffic monitoring, the number of objects handled in a day would easily run into several thousands. Using a database to index the detected objects would be more appropriate in such a case. In our application, we only consider a limited set of objects that appear in a specific dataset, for our testbed.

5.5. Object clustering

Comparing objects across multiple feeds is not a straightforward procedure. Objects detected in a video possess multiple attributes in three domains. There is added complexity due to some attributes like mean surface color which are multi-valued. The main objective is to take an object in one video feed and match it with its instances in other video feeds. An object may appear in only one video feed in a network of *n* sensors or may appear in all of them. A simple solution to the problem would be to compare the object in question with each and every object in all other feeds. However, this would be a highly inefficient manner of comparison as for every such object in that video feed, the same process has to be repeated. This would place enormous processing overhead on the system. Instead, we propose an intermediate clustering step that will reduce processing significantly. After all the objects in a video stream are linked so that a minimal number of objects in that video are arrived at, we need to cluster the objects into logical groups. This is because when we compare objects in a video stream with those in other video streams, the comparison process can be performed faster by just evaluating similarities of objects belonging to relevant clusters. Note on color: instead of storing direct color values, color shapes can be stored in order to effectively compare the same objects in different video streams but with different apparent color profiles. Ex: (120,120,120) is the same as (240,240,240) due to heavy illumination effects in the second dataset.

After clustering objects in a feed, it would be advisable to "normalize" all attribute values in all video feeds, to a uniform frame of reference. This is because when two clusters from two video feeds are compared, there is every possibility that local effects could have introduced errors in one cluster and hence comparisons would not be totally unbiased. Extreme errors could even cause two actually similar clusters to appear totally different. Further, even in the absence of ambient errors, there is a good possibility of distortion of object profiles as different sensors would have different view angles and scale factors of the scene in front of them. In order to bring all data to a uniform plane, we need to make use of sensor correction parameters such as transformation parameters, view angle, altitude, location etc. This process could also be performed before indexing but normalizing might magnify small errors between frames and intra-feed object linking might not perform accurately. Hence, any adjustments to object attributes should occur during the time when actual objects are being compared so that we do not alter the stored object records.

5.5.1. Clustering based on a dominant attribute

As opposed to single-attribute based linking, single-attribute based clustering does not rely on accuracy to define the choice for the attribute. Whereas, an inaccurately recorded attribute could hamper intra-feed object linking, clustering is basically only an arbitrary grouping of objects in a video stream. The grouping could be based on the variation in that domain seen in the objects in a video. Even though color is the most accurate attribute of an object, it need not be the only suitable candidate for clustering. This is because it is highly likely that in a video stream, the color profiles of the objects are very similar. This could happen if, for example, all the vehicles in a video feed are red or reddish in color. That would leave very less scope for grouping objects into separate classes. It would pose a problem while comparing with other video feeds because it would be necessary to evaluate comparisons with all the objects in the first feed, as they all belong to one class. To avoid such situations, a variability measure could let us decide which domain attribute is well-suited for object clustering. One way could be to check variance of all object attribute values from the mean. If they fell within a certain user-defined threshold, the domain would be unsuitable for clustering purposes. In such cases, we could choose another attribute that varies more than the set threshold. In rare cases where all attributes vary within the preset tolerances, a ranking could help identify the best suitable candidate for clustering.

5.5.2. Clustering based on membership to multiple classes

An alternative to clustering objects based on just one attribute would be to group them in many or all domains so that comparisons between video feeds are more accurate. The disadvantage of relying on one domain for clustering and subsequent inter-feed comparison is that inherent errors (if uncorrected) in one dataset contribute to mismatches with objects in another dataset. In using multiple domains for clustering and comparison, we can be assured that errors in one domain do not alter results. A simple voting system in which the object that has matches with another object in the maximum number of domains could make the comparison more accurate. Further refinements could be brought about by incorporating confidence measures of each domain to introduce a weighted-like selection scheme.

An object in a video feed could thus belong to several clusters, each corresponding to a domain. For example, an object could belong to the classes with mean spectral value of (250, 100, 50), mean size of 400 pixels, mean velocity of 30 pixels/sec. The only apparent drawback in this method is the time taken to cluster objects into many classes as they happen to be in different independent domains. This means computation is essentially repeated for each domain. If the resulting time delay is not a major issue in the application, it could be used to deliver better results. In our approach, we make use of multiple attributes to cluster objects and compare clusters in a hierarchical fashion.

5.6. Object comparison

Once all datasets are clustered, the objects in all feeds are ready for comparison to one another. The motive behind object matching is to establish consolidated trajectories over the entire network for an object. In mobile applications, it is crucial to know a tracked object's location from a holistic, regional view as the network can be reconfigured on-the-fly. Even otherwise, knowing just a fragment of an object's motion path in a network is of little practical use, especially when the network has multiple sensors.

5.6.1. Comparison of object lists

When object clusters are determined in each video feed, the comparison of objects takes place in two steps. First, the clusters in all video feeds are compared to find the closest neighbors of each cluster in each domain. While individual objects are grouped together based on their closeness of attribute values in each of the three domains in order to form object clusters, clusters themselves are in a sense clustered further in this step so that meta-clusters are determined. Meta-clusters can be thought of as clusters of mean domain values across multiple datasets while object clusters are logical groupings in one dataset. Once the list of meta-clusters is drawn up, we compare the objects in the constituent object clusters in a meta-cluster to one another. This ensures that only relevant objects are compared in the process of forming matches. For example, there are three clusters in the spectral domain, namely, red, brown, and black objects in four video feeds. Under a simple comparison, even black objects could be compared to red objects. However, meta-clustering ensures that only red objects in two video feeds are compared to each other.

5.6.1.1. Optimal comparison vs. pairwise comparison

In a multiple-sensor network, it might be enough to compare a few feeds for matching objects and extending matches logically to other feeds. For example, if there are three video sensors in a network, it is enough if we compare $feed_1$ and $feed_2$; and $feed_2$ and $feed_3$. It is not necessary to do the redundant $feed_1$:$feed_3$ comparison. In most cases, such a technique would suffice as the accuracy with which data is recorded in common video sensors is sufficient enough to allow for small amounts of error propagation. However, in applications which require a high level of precision, all pairs of feeds may be compared (except obviously redundant combinations like $feed_1$:$feed_2$ and $feed_2$:$feed_1$) so that there is no error carried over by inferencing match results. While the first technique has the advantage of timeliness of producing results in comparison to the second, it suffers from a potential carry-over of errors from one match to another.

5.6.1.2. Domain-specific comparison vs. weighted comparison

If we consider an atomic comparison step, an object in one video feed is compared to another in another feed. This comparison could be in terms of measuring how close the two objects are in domain-space. We consider three domains (spectral, spatial and spatio-temporal). Within each domain, we have several attributes. For example, spatial domain has attributes like area, compactness and size. When we determine the distance between the two objects in one domain, we find the distances of each sub-domain (attribute) and derive a single distance measure for the domain. If all attributes are assumed to be of equal importance and accuracy, we can normalize them to percent scores and multiply them. We would then get three individual scores for the three domains. In cases where there is not sufficient accuracy in one or more domains, we could choose the domain with the highest accuracy. However, it would be better to use a weighted score while obtaining matches in most cases as the accuracy of other domains is not too low to warrant their exclusion from the comparison process. Depending on its importance, each domain value could be assigned arbitrary weights and consolidated distance measures could be obtained.

5.6.2. Hybrid comparison method

Our approach towards object comparison and matching follows a mix of the techniques mentioned above. Object distance measures are obtained in every domain for all the attributes and normalized. All distance measures in a domain are then consolidated to obtain a single score for all three domains. Instead of assigning random weights to each domain, a hierarchy is followed in deciding if a match is suitable. The following pseudocode gives an idea of how the approach works:

```
For objects ($Obj_1$, $Obj_2$)
Iff dist_measure in $domain_1 <= threshold_1$ and
        Iff dist_measure in $domain_2 <= threshold_2$ and
                Iff dist_measure in $domain_3 <= threshold_3$
Then ($Obj_1 == Obj_2$) is true
```

Where, Obj_1 and Obj_2 are two objects in two distinct video feeds, $threshold_i$ is predefined by the user for $domain_i$, and $dist\_measure_i$ is the distance measure between both objects in $domain_i$.

The advantage of using this approach is that it narrows down the match to only those objects which fulfill the nested condition. In effect, it functions like a three-stage filter, eliminating unwanted matches along the way. The three-step comparison might not be restrictive enough to narrow down to the best match. There might still be multiple matches (though far fewer than obtained using a plain weighted comparison) which can be minimized by choosing the closest match, in terms of the weighted distance. The difference between just a weighted comparison and the hybrid approach is the fact that the filtering of results takes place twice in the latter.

5.6.3. Trajectory linking

Building a complete trajectory of an object's journey across a network basically consists of linking individual sub-trajectories obtained from video feeds recorded by different video sensors across the network. When we have enough information on object matches (or correspondences) across different video feeds (essentially, different sections of the network), a complete trajectory may be formed by linking sub-trajectories in a certain sequence. Linking sub-trajectories in a spatial sequence may seem to be a possible way of achieving it, but it must be noted that at times, objects may travel in complex loops across a network. For example, an object may travel in one direction for some distance, turn into a side-road at some angle and after some time may even come back by the same road it entered the network. A spatial sequence may not be able to capture the complexities of the object's path unless predictive models are in place.

Using timestamps as a key linking mechanism, we can connect sub-trajectories reliably without falling prey to simplistic assumptions about an object's possible route across the network. Trajectory linking using just timestamps is only a rough means to estimate an object's complete motion path across a network. This is especially true when sensors are sparsely distributed across the network and there is no overlap between adjacent sensors. If commonly used routes in a network are known, they could be used to predict an object's possible path in a blind spot (a location between two spatially-disjoint sensors).

Chapter 6

EXPERIMENTS

6.1. Experiments with data from a single sensor

In the course of establishing our approach for detecting moving objects in data recorded by multiple sensors, we tested our motion detection and tracking algorithms on videos obtained from a single sensor. The aim was to establish the efficacy of the algorithms in detecting objects correctly based on specified parameters and also to extract object attributes from the video.

6.1.1. Data collection

Color video was captured using digital video camcorders at 720×480 pixels and a frame rate of 29.97 fps. The camcorder was focused in such a way that as large as possible a view of the road below was available. This was to ensure that a significant portion of the object's trajectory (including turns) was captured as opposed to a closer but incomplete view of the object's path. This input video was reduced to 320×240 at 8 fps for testing low-resolution analysis and 720×480 at 3 fps for high-resolution analysis. High frame rates are not desirable given the nature of intense computations. The digital camcorders were set atop a three-storied building 50 feet tall. The sensor (camcorder) focused on an exit road. At the time of recording data, the sensor parameters such as location, focal length, etc. were not known. The goal of the experiments was to merely establish an object detection and tracking system upon which support for multi-sensor input could be built.

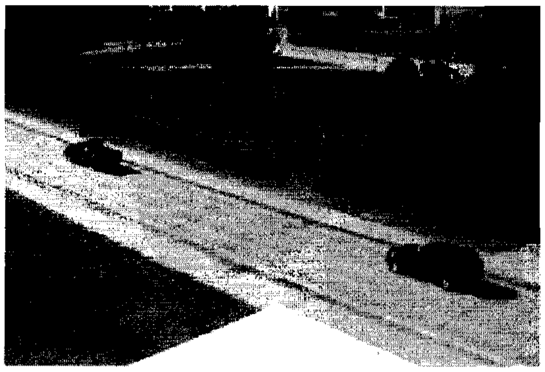


Figure 6.1: Viewing field of the sensor

The algorithms accept preset values for parameters that are derived iteratively. Parameter values are applied consistently across the video scene and act as global constraints, establishing a knowledge base for a potential distributed video sensor network.

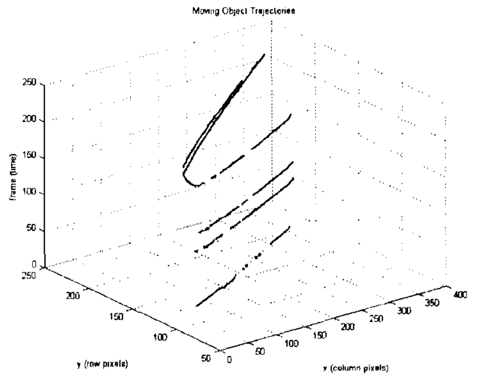
6.1.2. Processing

The apparent surface color of an object is determined by source illumination, viewing geometry and camera parameters. Processing color information can be expensive and is typically restricted to the pixels that have been obtained from the morphological operations on the object shape profile. Color information can be operated upon either in the RGB (Red-Green-Blue) domain or in the HSI (Hue-Saturation-Intensity) domain. In the RGB domain, isolating color information is not easy since all the three values (R,G,B) change significantly with illumination. In the HSI domain, the Hue value plays a vital part in object determination since it does not vary with intensity changes. This is particularly true when the object is subject to varying illumination levels as under shadows and under bright sunlight. Hence we attempt to track objects based on proximity in Hue values after shadows are eliminated. In order to get a Hue signature, the median hue value of pixels in an object blob is used.

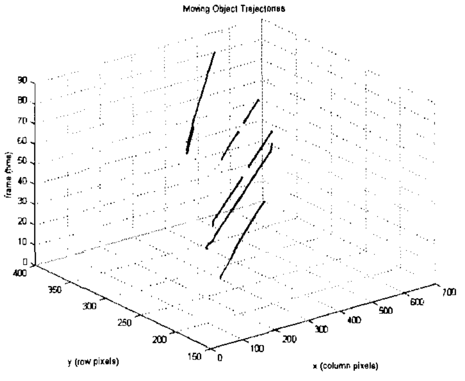
A rectangular region enclosing a detected object was first defined. Pixel values that were outside the object outline but within the bounding box belong to the surface on which the object traveled, in this case, the road. The frequency distribution of the hue values of these pixels was studied and a range of hue values which represented the road surface was chosen. It was assumed that the hue values of the shadowed region would be very close to the hue values obtained from the road pixels as the shadow is merely a region on the road with an illumination level different from its surrounding areas. The basis for this assumption is the premise that hue value of a surface does not vary with varying incident illumination. Once the range of hue values corresponding to the shadow was identified, those pixels in the vicinity of an object with hue values falling in the 'shadow range' were eliminated. These pixels were identified by the motion tracking algorithm as part of the object as the shadow moved with the same velocity as the actual object. Once the shadows were eliminated, object hue values obtained were more relevant to the actual object surface color. This helped in linking object trajectories across the two datasets with greater accuracy.

6.1.3. Results

In the analysis of the video, we used the mean surface color of the object as a comparison metric for linking objects across frames. Trajectories of each object were formed based on the recorded centroid location of the object in consecutive frames. Trajectory formation was subject to the vector-building parameters. Hence, in frames where the objects stopped, their centroid locations were undetectable (due to absence of motion), and this caused their trajectories to fragment. The motion trajectories for the objects in in the scene analyzed based on different resolutions are shown in Fig. 6.2 below. The trajectories are plotted on the X-Y plane with time along the vertical axis. Shadow filtering was performed only on the high resolution analysis due to high loss of object surface pixels when performing the filtration.



(a) 320*240; 8fps



(b) 720*480; 3fps

Figure 6.2: Motion vectors: Low- (a) and High- (b) resolutions

Resolution	**Size**	**FR**	**C**	**mOA**	**MOA**
Low	320*240	8 fps	0	200	10000
High	720*480	3 fps	0	4000	10000

Table 6.1: Parameter values for scenes 1 and 2

In both the output figures (Fig. 6.2), we see that the trajectories of individual objects are represented by independent lines over the entire duration of the video. The first figure (Fig. 6.2a) incorrectly links two separate objects as one and hence the "hairpin" trajectory. It can clearly be seen that even an object making an about turn would have the portion of the trajectory where it travels in the opposite direction, still pointing in the increasing direction of the Z-axis (frames, as a representative of time).

In the actual video, the objects do move linearly and hence the first output is wrong as it connects two objects because they appear to have similar properties. The same error does not occur in the high-resolution data output. High-resolution data offers better accuracy whilst entailing significantly longer processing times than lower resolution data. In this example, the ratio between higher- and lower- resolution data is 4.5 ((720*480)/ (360*240)). This means that the processing time is 4.5 times longer in the case of the high-resolution data for the same duration of video. This can be somewhat mitigated by reducing frame rate (by a factor of 2.67 (8/3) in our case). However, reducing frame rate can cause linking errors in a high-volume dataset. Hence, a proper balance has to be struck by studying the potential characteristics of a geospatial network. Further, in this case two different objects were merged in the first output because of using only a selection of all possible parameters (color). A more complete description of the object's properties could help avoid linking errors.

6.2. Experiments with synthetic data

In order to demonstrate our clustering-based object matching approach, we conducted an experiment based on synthetically-generated data. We generated the datasets to act as object profiles extracted from real-world video data in a multi-sensor-monitored traffic network. The aim of the experiment was to generate a large number of objects (>500) per dataset in order to simulate a real-world traffic network with a high density of moving vehicles (e.g., a highway).

6.2.1. Data generation

For dataset generation we used actual signatures of five distinct objects captured with real cameras in our university campus (see Table 6.2). Their radiometric and geometric properties (color, area and compactness are reported here) were extracted and served as seed values to generate our synthetic datasets. These seed values were subjected to the application of random errors (using a pre-defined standard deviation) in order to generate multiple object instances within five classes that correspond to the seed objects. Then, datasets were generated comprising three of these classes and an additional random class, with randomly generated properties (see Table 6.3).

Class	**Spectral Value (R, G, B)**	**Area**	**Compactness**
1	104 230 240	2500	0.21
2	128 10 10	4100	0.36
3	22 21 18	5700	0.49
4	235 55 46	9300	0.72
5	255 206 207	7700	0.59

Table 6.2: Reference dataset seed values in three domains

Dataset	Reference classes (200 objects each)				Deviation from reference (%)	Total objects
1	1	4	5	0	1.01	800
2	3	4	1	0	1.02	800
3	2	4	3	0	1.05	800
4	2	1	5	0	0.97	800

Table 6.3: Composition of synthesized datasets: '0' refers to randomly generated data

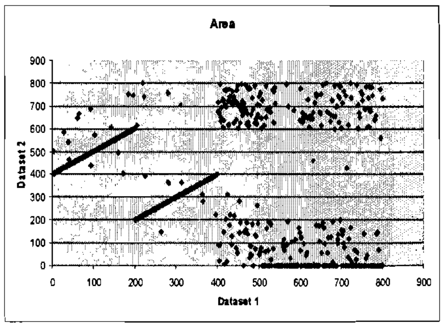
The combination of classes in each datasets is such that there is some overlap between any two datasets, indicating instances where a car moved from the field of view of one sensor onto the field of view of another. Data that are not common to two datasets but present in one of them represent objects that crossed the field of view of one sensor only in our hypothetical network. Randomly generated datasets (represented by '0' in Table 6.3) would be interpreted in such a manner.

6.2.2. Clustering

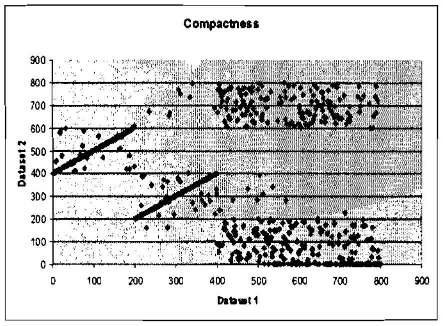
After the artificial datasets were generated, each dataset was subjected to a clustering algorithm that grouped objects together based on similarity in a single attribute. The clustering started with a predefined number of classes. Based on a migrating-means approach, the clusterer grouped objects whose value in an attribute-domain (say, Area) fell within a certain threshold. During the second iteration, the class means were refined by taking the mean of the attribute values of the member objects in each class. The new mean values were then used to reclassify objects. The same process was repeated once more and if the class mean values changed little, the clustering process was stopped. The clustered objects were then compared to those in another dataset based on the closeness of the classes they belonged to.

6.2.3. Results

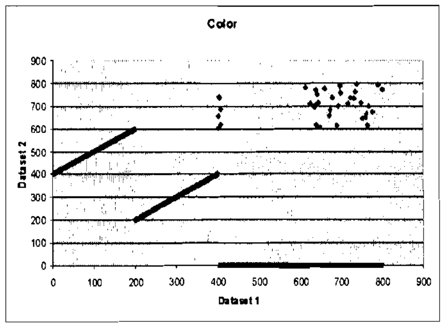
The objects detected in one dataset were matched to those in another by first clustering each dataset. The clustering was based on a simple Euclidean-distance based clusterer that grouped objects into one of a predefined number of classes based on the proximity of the object. The clustering algorithm is based on the ISODATA algorithm for classifying digital images. The clustering algorithm worked by iterating over the datasets more than once in order to first determine initial mean seed values for the clusters and then grouping objects in subsequent iterations, while refining the mean values. The results of the comparison are shown in Fig. 6.3.



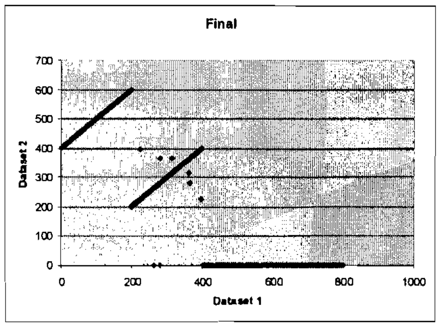
(a) Object matches with area alone



(b) Object matches with compactness



(c) Object matches with color alone



(d) Object matches with all three domains

Figure 6.3: Results of the comparison (a,b,c) separately and (d) in all domains together

Dataset	1	2	3	4
1	-	399	181	286
2	399	-	370	179
3	181	370	-	149
4	286	179	149	-

Table 6.4: Tabulation of object matches in all dataset pairs

In Fig. 6.3, objects in dataset-1 that did not map to any object in dataset-4 lie on the X-axis. The comparison between two objects is based on a simple computation of percent change in the parameters from one of the objects. If it fell within an acceptable threshold, the object was added to a table of 'possible matches'. When all objects in a dataset were compared to the current object in the other, the matching table was sorted. The object that had the least change in comparison to the current object was chosen as its match in the other dataset. Successful matches were eliminated from the pool in order to speed up the process. In a real-time system where newly detected objects are compared to those that already exist in the database, time-consuming comparisons could slow down the system and will give rise to performance issues. On an average, our system took around 36 seconds to compare 1600 objects in the database. After matches were obtained in individual domains, they were consolidated by choosing the match that appeared in more than one domain. In inconclusive cases, the match was determined as 'null'. The system could be extended by assigning levels of confidence to each attribute (color, compactness etc.). In cases where the matches conflict with one another, the match that is in the domain with a greater degree of confidence could be chosen over the rest. As of now, all attributes were assumed to have been measured with the same level of accuracy.

6.3. Experiments with data from multiple sensors

To demonstrate the validity of our approach of multi-component object signatures, we recorded data using eight video sensors with disjoint views of a traffic scene and processed the data to obtain object information in multiple domains. The object signatures were then used to build indexes for each dataset that ordered objects according to the value of their attributes. The objects were then clustered and object comparison proceeded by comparing object clusters first and then comparing objects in the matching clusters.

6.3.1. Data collection

Data was recorded at eight different locations on the University of Maine campus using digital camcorders mounted on tripod. The camcorders recorded data for a period of nearly 45 minutes starting at nearly the same time of the day. The locations and extents of their coverage are shown below in the following figure:

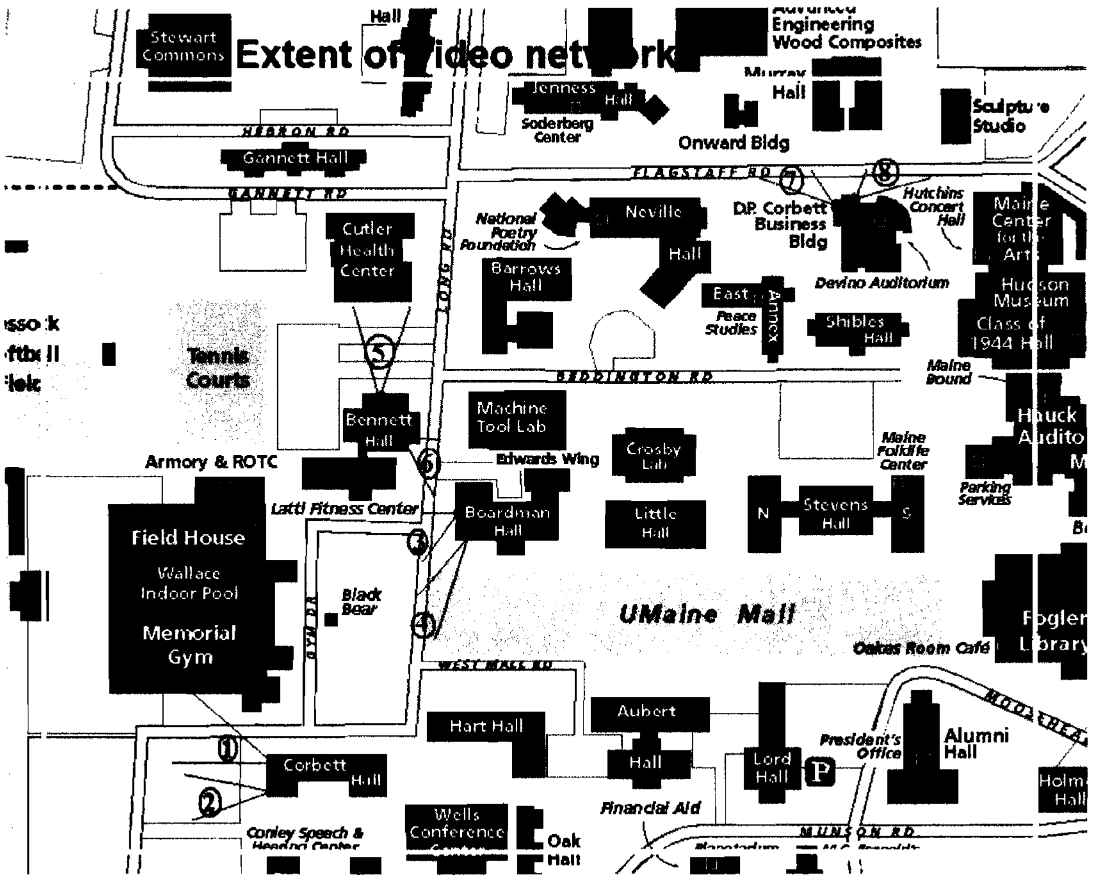


Figure 6.4: Map of the University of Maine campus showing the location of sensors 1-8

The sensors were placed in such a way that they covered the flow of the bulk of traffic on the eastern wing of the university campus. All the sensors had an elevated view of the terrain below as they were placed at window sills of the buildings in which they were located. The illumination changed considerably in certain sensor locations due to changes in weather. The data recorded by each camcorder was of a framesize of 720×480 and 30 fps. When transferred to the computer, the data was reduced to a size of 360×240 and 10 fps as the frequency of vehicles in each video was not too high and also the size of the vehicles in the encoded video was adequate enough for the purposes of our experiment. This conversion of the datasets ensured that processing time was cut down significantly but at very little compromise on the tracking accuracy.

6.3.2. Processing

The eight videos recorded by the sensors were individually processed in order to derive motion information. Each video was processed using the motion detector and motion tracking algorithms. Based on input parameters, a number of objects was extracted, with multiple characteristic-attributes (area, compactness, R, G, B, average speed and acceleration), from each video. The input parameters for the processes of motion detection and tracking were generated from the previous experiments with single feeds. Absolute mode of AFD was used in order to preserve the shape of the detected object. The following table lists the values for the input parameters for the motion detection and tracking algorithms:

	Parameter	Description	Value
Motion detection	keyFrame	Key frame	3
	diff_mode	Differencing mode	'abs'
	accum_rate	Frame accumulation rate	5
	T1	Accumulation threshold	3
	T2	Gray-level threshold	20
Motion tracking	keyFrame	Key frame	3
	SE_size	Structuring element size	3
	minCompact	Minimum object compactness	0
	minArea	Minimum object area (pixels)	50
	maxArea	Maximum object area (pixels)	10000
	maxDist	Maximum allowable distance for connecting trajectory points (pixels)	20
	minLinks	Minimum number of links for a trajectory	5

Table 6.5: Parameters used for object detection and tracking

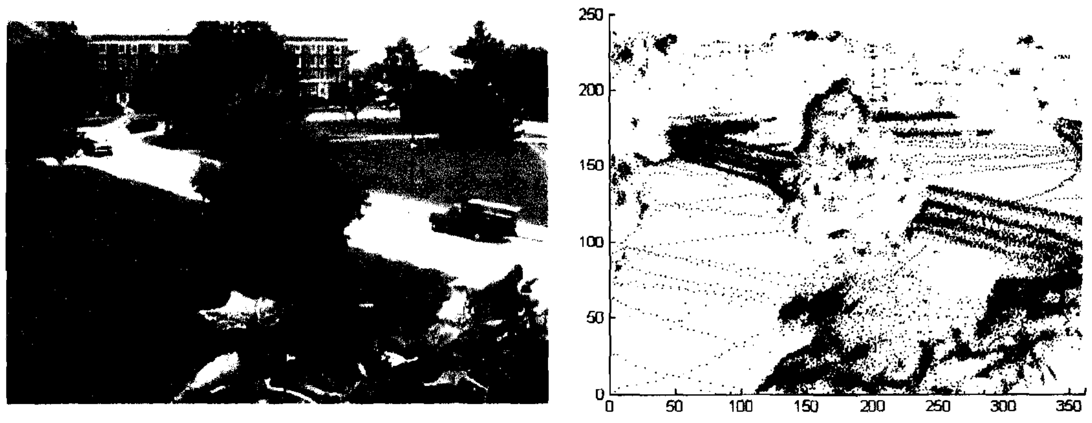


Figure 6.5: (a) Frame from the original video, (b) Scatter plot of object centroids

The above figure (Fig. 6.5a) shows a frame from one of the videos. The adjacent figure (Fig. 6.5b) shows a scatter plot of all the object centroids in the video. The outline of the tree as well as the plant in the edge of the frame is visible in the scatter plot. Due to the windy weather at the time of the video capture, the leaves in the plant and trees moved for considerable length in the video and hence accumulation of frames was not completely successful in eliminating unwanted object points due to the plant movement. Using a higher value for the frame accumulation rate could have resulted in obliteration of some infrequent but valid objects.

After each video was processed, object signatures (area, compactness, mean surface color, average speed, acceleration and direction of motion) in the spectral, geometric and spatio-temporal domains were obtained. In addition to that information, object centroids were obtained for each frame in which the corresponding object appeared. Knowledge of centroid locations helps us to build object trajectories for each sensor view. A sample output is shown below in Fig. 6.6, where three sensor views and the trajectory plot for each sensor are shown.

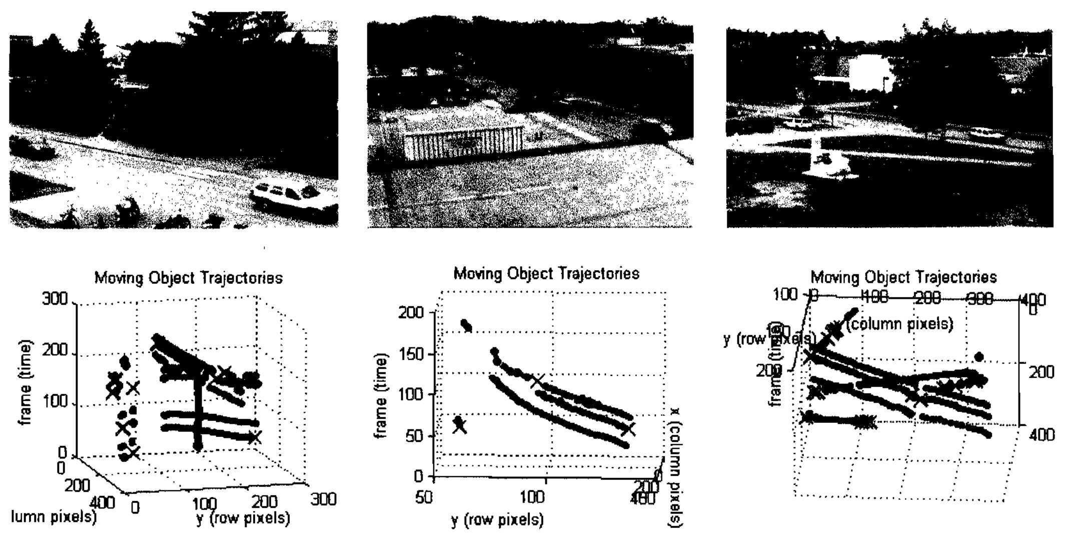


Figure 6.6: (top row) Frames from videos taken at sensors 3,5,7 (bottom row) Trajectory plot for all objects in each video

Object attributes were then used to build indexes for each domain in each video feed. As already described in Chapter 5, indexing was implemented as a simple ordering of objects. Once indexes were created for each dataset, clusters were built from analyzing the indices for similar values. The clustering was based on a K-means algorithm. The software program used for the clustering algorithm was the public domain Fuzzy Clustering and Data Analysis Toolbox written by Balazs Balasko, Janos Abonyi and Balazs Feil, available from the website of Mathworks Inc. An arbitrary number (four) of clusters was chosen to be constant throughout all domains. The following figure shows a clustering of the "Area" attribute of objects detected in the first dataset, with five clusters:

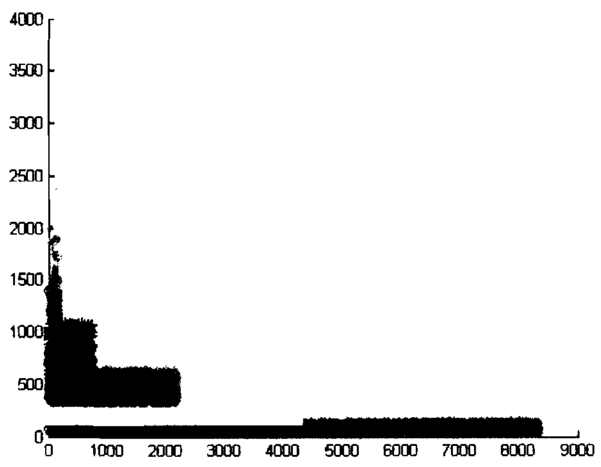


Figure 6.7: Clustering results for the “Area” attribute of the first dataset

Object attribute	Description	Value
Area	Area of the object (pixels)	193.55
Comp	Compactness of the object	0.18075
SpAng.R	R component of spectral triangle	0.96531
SpAng.G	G component of spectral triangle	0.98556
SpAng.B	B component of spectral triangle	1.1907
Color.R	R component of mean surface color	166.45
Color.G	G component of mean surface color	162.86
Color.B	B component of mean surface color	126.45
AvSpeed	Average speed of object (pixels/sec)	0.60337
AvAccel	Average acceleration of object (pixels/sec^2)	0.071995

Table 6.6: A sample object from the first dataset

After clusters are generated in each dataset, we begin the process of comparing individual objects in two datasets. The comparison first starts a comparison between the clusters in each of the two datasets. This step is necessary to ensure that only relevant objects are compared. This step can be likened to comparing hypothetical objects that possess the cluster means as their attribute values. Once a correspondence between the clusters in two datasets is established, object comparison can proceed further. In order to make sure that object comparisons are unbiased by local errors, we need to bring all object attribute values to a uniform scale. In the datasets we have captured, the size of objects is one attribute that varies greatly across datasets. Based on apparent variations between datasets, weights are assigned to each dataset to scale the size values of each object during the process of comparison. During object comparison, only those objects are checked whose clusters are similar. Also, in order to avoid redundant comparisons, only a subset of the possible pairs of datasets is compared, as seen in the following table:

Datasets	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	-	*	*	*	*	*	*	*
2	-	-	*	*	*	*	*	*
3	-	-	-	*	*	*	*	*
4	-	-	-	-	*	*	*	*
5	-	-	-	-	-	*	*	*
6	-	-	-	-	-	-	*	*
7	-	-	-	-	-	-	-	*
8	-	-	-	-	-	-	-	-

Table 6.7: Dataset-pairs marked by "-" are redundant and are not involved in the comparison process. Those marked by "*" were considered.

A single object comparison ($Obj^i_m == Obj^j_n$) follows the following rule-scheme

The following differences are defined:

$$SpAng(Obj^i_m) \sim SpAng(Obj^j_n) \quad (6.1)$$

$$RGB(Obj^i_m) \sim RGB(Obj^j_n) \quad (6.1a)$$

$$Area(Obj^i_m) \sim Area(Obj^j_n) \quad (6.2)$$

$$Comp(Obj^i_m) \sim Comp(Obj^j_n) \quad (6.3)$$

$$AvSpeed(Obj^i_m) \sim AvSpeed(Obj^j_n) \quad (6.4a)$$

$$AvAccel(Obj^i_m) \sim AvAccel(Obj^j_n) \quad (6.4b)$$

A potential match is chosen:

$$\text{iff } (6.1) < threshold_1 \text{ and iff } (6.2) < threshold_2 \text{ and iff } (6.3) < threshold_3 \quad (6.5)$$

If (6.5) is fulfilled, Obj^j_n is chosen as a possible match m_k for Obj^i_m. In order to pick the most relevant match out of a set of k matches, we choose the object for which (6.1a), (6.4a), and (6.4b) are the least. A list of matches is built in this manner for all objects in both datasets. The same process of comparison is repeated for each pair of datasets marked by "*" in Table 6.7.

A sample final match-list is produced below in Table xx. The two videos compared were short (30 sec.) segments of datasets 1 and 3. The table on the left (Table xx.a) shows the results obtained by comparing the objects in both videos solely on the basis of their mean surface color and the one on the right shows the results obtained by subjecting the objects to a multi-stage comparison based on their signatures.

Object in dataset-1	**Object in dataset-3**
1	6
2	7
3	14
4	0
5	12
6	0
7	0
8	0
9	18
10	2
11	0

(a)

Object in dataset-1	**Object in dataset-3**
1	6
2	1
3	14
4	3
5	12
6	0
7	0
8	5
9	18
10	2
11	0

(b)

Table 6.8: (a) Match results obtained from color-based comparison (b) Match results obtained from multi-component signature-based comparison

Matches with a "0" refer to no possible matches. In the first case, the matching accuracy is a low 46%. While a pure color-based comparison usually yields good performance, the reason for the poor match is due to the presence of occlusions (trees) in one of the videos. Normally, using color information and proximity as an auxiliary measure, a fairly reliable matching can be obtained. With occlusions, proximity cannot be used as a means to verify matches as spatial continuity is lost during the period of occlusion. In the case of the multi-component signature matching, the accuracy for the same pair of datasets is 73%. The main inference from this result is that even with occlusions, a multiple attribute-based matching scheme offers more scope for accuracy due to the mere fact that with more information about the objects at hand, a more reliable matching can be performed.

6.3.3. Results

The results of processing the eight datasets confirm the validity of our approach in matching moving objects detected in disjoint video streams. The parameters used for detecting and building objects were chosen so that all valid objects were detected. However, a lot of unwanted objects (caused by background motion) were also detected as a result. The breakup of the object matches between the various pairs of datasets is presented in Table 6.9. The topmost row shows the number of objects detected in each dataset and the leftmost column shows the dataset to which those in the topmost row are compared. The number of object matches indicates only the object correspondences between each dataset, based on multi-component signatures. It does not indicate the percentage of successful matches as it would be virtually impossible to verify each and every match, and if so whether it is the best match. Further, the number of objects common to two datasets may not be the only objects appearing in both. Added to the non-common objects are unwanted objects that might have crept in due to the relaxed value of the frame accumulation rate.

Dataset	**1 (5871)**	**2 (2396)**	**3 (5310)**	**4 (5523)**	**5 (4293)**	**6 (3469)**	**7 (5343)**	**8 (3761)**
1	0	2627	3409	3383	3746	2781	3208	2381
2	0	0	1725	1612	1490	1964	1369	1081
3	0	0	0	1422	1379	1613	1152	1253
4	0	0	0	0	1374	1642	1209	1282
5	0	0	0	0	0	1317	1729	1817
6	0	0	0	0	0	0	1229	1349
7	0	0	0	0	0	0	0	2805

Table 6.9: Object matches across eight datasets

On an average, each dataset contained more than 3,000 objects (including valid objects such as cars and bikes, and unwanted moving objects like pedestrians and background motion) in an approximate duration of 45 minutes. To compare each object in a dataset individually in a pairwise comparison over four attributes with those in another dataset would have resulted in $3{,}000 \times 3{,}000 \times 4 = 36$ million operations. However, by clustering objects into four classes, only the relevant classes are compared. This brought in a significant reduction in processing time. For a real-time system operating with constraints on reporting times, it is essential to eliminate redundant processing cycles. Our clustering-based object motion analysis approach is well-suited for such an application.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1. Summary

In this research, we sought to solve the problem of establishing a semi-autonomous computer-based system for processing and analyzing video streams obtained from multiple video sensors having non-overlapping views of a terrestrial network (region). The problem involved a multitude of parameters and variables. System variables included location of sensors, sensor intrinsic parameters such as sensor view angle, scale factor, altitude. The video capturing system was also not assumed to be robust or, in other words, totally error-free. The design of the system was envisioned to work with error-prone data and yet produce acceptable and timely results. The major outputs of the system included location and description of the moving objects in the different video streams. Derived outputs include trajectories of the object across multiple sensor-views. In the course of the development of the video analysis system, a limiting case was first considered and then generalized to hold good for a complex sensor network scenario. Initially, algorithms were developed for the case of a single video sensor. The aim was to develop techniques to detect motion in a video and construct objects from the detected moving regions.

7.1.1. Algorithms

The fundamental algorithm underlying the video analysis system is the object comparison technique based on multi-dimensional signatures. The preliminary object information is obtained by means of a frame differencing algorithm. The frame differencing algorithm is fairly popular in the computer vision domain for the primary reason that it offers superior accuracy at very little processing cost. Since all video analysis tasks revolve around the central technique of matrix manipulations, it is very desirable to reduce the fundamental complexity of the algorithms as an iterative processing of a video using a complex algorithm could prove to be computationally expensive and also slow in producing results. The basic differencing algorithm was modified to incorporate a "memory" of sorts to eliminate false positives from the list of moving regions. The differenced video data was then processed using pixel-level morphological operators to build objects from moving regions in the video. These two algorithms constituted the backbone of the single-sensor-based tests. When a high level of detection accuracy was achieved, the algorithm was further developed to enable processing of multiple video feeds and match objects detected in each of them.

The core concept behind the multiple-sensor-suitable system is that of multi-component object signatures. Using the attribute information derived for each object from each video, a composite signature is constructed which is used for matching the object to its instances in other video feeds. By means of user-defined weights for rating the quality of the attribute data, the accuracy can be held consistent even in the face of changing ambient conditions that adversely influence the quality of data recorded at each video sensor. Further accuracy in generating results is by bringing all the data values of two comparable objects to the same scale. Often, local factors such as changing illumination could seriously affect the accuracy with which two objects can be matched. This issue was addressed by bringing in the concept of spectral angles (analogous to "spectral signatures" of physical objects in remote sensing terminology) for defining the instrinsic color characteristics of detected objects. Another source of error is trailing shadows of objects caused by the oblique position of the sun in the sky. Shadows cannot often be removed by differencing because they travel with the same velocity as the object and are "attached" to the object. We incorporated a shadow removal algorithm in our analysis system which removed shadow pixels from an object's shape profile by performing a neighborhood search on "blacklisted" hue values.

7.1.2. Evaluation of the algorithms

For the single-sensor-based tests, we captured datasets with objects undergoing simple motion. The objects in question followed predetermined paths. The reason for predefining the motion path was to create a scenario where objects moved back and forth and in opposite directions at different points of time. Besides, objects of different sizes (cars, people etc.) were also involved in the video shoot to test for the efficiency of the motion analysis algorithms in detecting objects of varying sizes. Another added complexity was the deliberate coverage of areas of different background illuminations to check for the robustness of the algorithm. With all these constraints, the algorithms performed very well and produced accurate tracking results. In the multiple-sensor-based tests, we captured video feeds from sensors distributed at different points on the university campus. We made sure that each sensor had a totally disjoint view of the scene as compared to the adjacent sensors. The data capture was started at nearly the same time to ensure common objects in more than one of the captured video feeds. The video feeds were processed independently first to obtain lists of objects in each of them, along with their respective attributes in the three domains – spectral, spatial and spatio-temporal. The generated data was used to build signatures for each object and consequently, an object index for each dataset. The index was then clustered on each attribute to reduce the complexity of matching objects across multiple feeds. The actual object comparison proceeded after cluster matches were obtained. This ensured that redundant processing cycles were avoided for performing a direct one-to-one comparison. The results of the object matching confirm our hypothesis that a multi-dimensional object signature enables in obtaining better accuracy in object detection and matching.

7.2. Conclusions

Several conclusions were drawn during the course of this research on the characteristics of the original problem – object matching across multiple feeds. Firstly, the ambient environment for this problem space is highly variable. Multiple natural factors strongly influence the way in which raw visual data is recorded on the motion of objects on a terrain. Any system that seeks to establish an autonomous monitor should be built keeping this factor in mind. The system has to be designed to withstand the negative effects of the environmental factors as they influence the basic binary accuracy (pass/fail) of the matching process. Other factors, such as background clutter, subtly influence the final matching accuracy. An example where they play a role would be where the trajectories of two similar objects could be "knotted" by the system, due to both of them traveling very close to each other (because of traffic conditions). The system should, thus, be able to discriminate two similar (in terms of color, size, shape, or motion) objects.

Another important conclusion is that an object motion analysis system based on video sensors should be as minimally complex as possible. Video processing is quite different from conventional image processing in the fact that the process is basically iterative (over multiple digital images, or frames). This adds a further level of complexity to any algorithm and basically delays the time by which the results of an image processing can be obtained. Keeping this in mind, a video analysis system should thus be based on a simple, though not at the expense of the final accuracy, algorithm. The fundamental iterative (over all pixels in all frames) algorithm in our system is a subtractive process that requires the least of system resources for its processing and executes fast compared to costly filter-based operations such as edge detection.

Further, while some time can be sacrificed for extracting object information from a video, the process of matching objects must take the least amount of time possible. In the initial stage, the input is a huge data, basically, a video with multiple frames and each frame with thousands of pixels. After the process of video analysis completes, the output is a mere few data records (of object information) per frame which can be analyzed in much quicker time. An object comparison algorithm cannot waste time on redundant (Obj_1:Obj_2 and Obj_2:Obj_1) and/or invalid comparisons (red-colored object:blue-colored object). For a real-time object detection system, all the above conclusions hold good. As final concluding note, our system is well-suited for a real-time application as it was designed keeping all these factors in mind.

7.3. Deployment scenarios

Our experiments were only constructed to act as "proofs of concept" or validators of our hypothesis, but we envisage that our approach can be successfully implemented to solve a variety of practical real-world applications. One representative application is standard traffic monitoring.

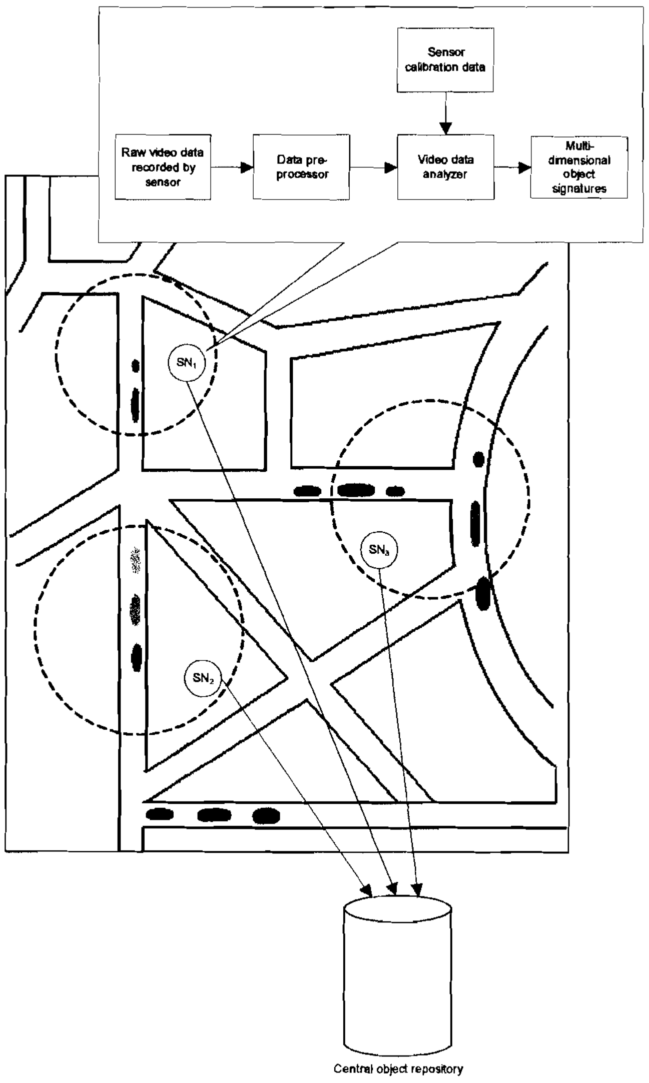

Sensor calibration data
Raw video data recorded by sensor
Data pre-processor
Video data analyzer
Multi-dimensional object signatures
SN1
SN2
SN3
Central object repository

Figure 7.1: Possible configuration of a traffic-monitoring system based on our approach

A typical road traffic network consists of hundreds, if not more, of vehicles of different sizes and shapes. Furthermore, they also vary in their hues and shades, not to mention their vastly different motion characteristics. This makes it an ideal situation for implementing our approach of multi-dimensional object indexing and linking. A real-time system for monitoring the traffic could be constructed on the lines of Fig. 7.1. The network shown in this figure comprises four sensor nodes. Each sensor node (shown as SN_n) is an integrated video analysis system which consists of several distinct components like, video sensor, data pre-processor, video data analyzer, and wireless data transmitter. The video sensor captures raw video data at a preset spatial resolution, frame rate and color depth. This data is sent to a pre-processor to optimize the raw video data for further processing. A pre-processor could be an integrated circuit board specifically designed to downsample video data. The processed video data is then sent to the video data analyzer, which is basically a dedicated computer/circuit board fabricated for the purpose of motion detection, object tracking and indexing. The data analyzer takes in the sensor calibration data as auxiliary input to scale the intermediate object attribute information. The final output of the analyzer is a list of objects as well as their multi-dimensional signatures.

The output is transmitted to a central object repository where the information sent from all the sensors is maintained in a database. The repository, thus, maintains an "index of indices", meaning that it holds a list of object attribute lists which could be used for future object comparisons. The advantage of processing captured video at the sensor node-level is that precious wireless bandwidth need not be wasted in transmitting raw video data to a central processor. Further, it would be impossible to process the video data relays of all sensors at the same time by a central processor. For a real-time application like traffic monitoring, it would be best suited to process data at the node-level and use the central processor only for the purposes of object signature analysis and subsequent comparisons. The object database could be used to build a knowledge base of object classes such as school buses, 4×4 trucks, or bicycles. This could improve the accuracy with which objects can be compared in future comparisons, as knowledge of object classes would enable the system to instantly label an object instead of subjecting it to a standard clustering procedure. Our approach was implemented in a MATLAB® environment which is very fast for the purposes of implementation but is not optimized for commercial deployment due to the interpretation of user code. Using a compiling language like C++ could increase the throughput of the system significantly. Best results would of course be achieved only by a hardware implementation. However, more research needs to be done in the area of formalizing many of the arbitrarily-held parameters in our approach so that the algorithm is more suited to a hardware implementation.

7.4. Future Work

Future research in this domain could proceed in the direction of constructing a real-time system that works in an encapsulated fashion. In our case, the system was designed to act as a concept-machine. Further development could be in increasing system throughput by optimizing the program code. The concept of assigning weights to different object attributes could be studied in a formal manner and a theoretical basis for the same could be established. At present, the weights are user-defined and are dependent on the judgment of the user. By tying down the weights to an intrinsic calibration mechanism, the object comparison could be made more accurate. Another improvement could be in the area of object data normalization. Using GPS location data, camera parameters could be established, using which the normalization could be made more accurate. Another important enhancement could be in the direction of creating a formal object database. This could form the basis for a knowledge base, which ultimately would enable definition of object classes (buses, trucks, cars etc.) with more accuracy.

BIBLIOGRAPHY

V. R. Agazi, G. E. Ford, A. I. El-Fallah, and R. R. Estes, Jr. (1995) Preprocessing for improved performance in image and video coding, *Proceedings of SPIE: Applications of Digital Image Processing XVIII*, Vol. 2564, pp. 22-31

J. Alon and S. Sclaroff (2000) Recursive Estimation of Motion and Planar Structure, *IEEE Conference on Computer Vision and Pattern Recognition*, South Carolina

A. Amer (2005) Voting-Based Simultaneous Tracking of Multiple Video Objects, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 11, pp. 1448-1462

A. Azarbayejani and A. P. Pentland (1995) Recursive Estimation of Motion, Structure, and Focal Length, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 6, pp. 562-575

J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani (1992) Hierarchical Model-Based Motion Estimation, *Proceedings of the Second European Conference on Computer Vision*, pp. 237 - 252

J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg (1990) Computing Two Motions from Three Frames, *International Conference on Computer Vision*, pp. 27-32

M. J. Black and D. J. Fleet (2000) Probabilistic Detection and Tracking of Motion Boundaries, *International Journal of Computer Vision*, 38(3), pp. 231-245

S. M. Brennan, A. M. Mielke, D. C. Torney, and A. B. Maccabe (2004) Radiation detection with distributed sensor networks, *IEEE Spectrum*, pp. 57-59

M. J. Brooks, W. Chojnacki, A. Hengel, and L. Baumela (1998) Robust Techniques for the Estimation of Structure From Motion in the Uncalibrated Case, *European Conference on Computer Vision*, Freiburg, Germany, pp. 281-295

T. Camus and H. H. Bülthoff (1995) Real-Time Optical Flow Extended in Time, Max-Planck Institut für biologische Kybernetik, Technical Report No. 13

M. M. Chang, A. M. Tekalp, and M. I. Sezan (1997) Simultaneous Motion Estimation and Segmentation, *IEEE Transactions on Image Processing*, Vol. 6, No. 9, pp. 1326-1333

P. Chang and Martial Hebert (2000) Omni-directional Structure from Motion, *IEEE Workshop on Omnidirectional Vision (OMNIVIS'00)*, pp. 127-133

I. Cohen and G. Medioni (1998) Detecting and Tracking Moving Objects in Video from an Airborne Observer, *DARPA Image Understanding Workshop*, Monterey

R. T. Collins (2003) Mean-shift Blob Tracking through Scale Space, *IEEE Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin

D. Comaniciu and V. Ramesh (2000) Mean Shift and Optimal Prediction for Efficient Object Tracking, *IEEE International Conference on Image Processing*, Vol. 3, pp. 70-73

D. Comaniciu, V. Ramesh, and P. Meer (2000) Real-Time Tracking of Non-Rigid Objects using Mean Shift, *IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina

D. Cremers (2003) A Variational Framework for Image Segmentation Combining Motion Estimation and Shape Regularization, *IEEE Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin

D. Cremers and A. Yuille (2003) A Generative Model Based Approach to Motion Segmentation, *German Conference on Pattern Recognition*, Magdeburg, Germany

D. Cremers and S. Soatto (2003) Variational Space-Time Motion Segmentation, *International Conference on Computer Vision*, Nice

D. Culler, D. Estrin, and M. Srivastava (2004) Overview of sensor networks, *IEEE Spectrum*, pp. 41-49

A. Chachich, A. Pau, A. Barber, K. Kennedy, E. Olejniczak, J. Hackney, Q. Sun, and E. Mireles (1996) Traffic sensor using a color vision method, Massachusetts Institute of Technology, Center of Transportation Studies, Cambridge, Massachusetts

Y. Deng, D. Mukherjee, and B. S. Manjunath (1998) NeTra-V: Towards an Object-based Video Representation, *Proceedings of SPIE: Storage and Retrieval of Image and Video Databases VI*, pp. 202-213

H. Eltoukhy and K. Salama (2002) Multiple Camera Tracking, Stanford image sensors group, Electrical Engineering Department, Stanford University.

V. Estrela, L. A. Rivera, M. H. S. Bassani (2004) Pel-Recursive Motion Estimation using the Expectation-Maximization Technique and Spatial Adaptation, *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Plzen-Bory, Czech Republic

B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills (1998), Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms, *Proceedings of the Ninth British Machine Vision Conference*

A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, A. Senior, C. Shu, and Y. L. Tian (2005) Smart Video Surveillance, *IEEE Signal Processing Magazine*, pp. 38-51

M. Hanmandlu, S. Vasikarla, and V. K. Madasu (2003) Estimation of Motion from a Sequence of Images using Spherical Projective Geometry, *International Conference on Information Technology: Computers and Communications*, pp. 541-545

B. K. P. Horn and B. Schunck (1981) Determining Optical Flow, *Artificial Intelligence*, Vol. 17, pp. 185-203

G. Iannizzotto and L. Vita (2002) On-line Object Tracking for Colour Video Analysis, *Real-Time Imaging*, 145-155

M. Irani (1999) Multi-frame Optical Flow Estimation using Subspace Constraints, *IEEE International Conference on Computer Vision*, Corfu

Ch. Jaynes (2004) Acquisition of a Predictive Markov Model using Object Tracking and Correspondence in Geospatial Video Surveillance Networks, *GeoSensor Networks*, pp. 149-166

A. D. Jepson, D. J. Fleet, and M. J. Black (2002) A Layered Motion Representation with Occlusion and Compact Spatial Support, *European Conference on Computer Vision (ECCV)*, Copenhagen, Vol. I, pp. 692-706

T. K. Kate, M. B. Leeuwen, S. E. Moro-Ellenberger, B. J. F. Driessen, A. H. G. Versluis, and F. C. A. Groen (2004) Mid-range and distant vehicle detection with a mobile camera, *IEEE Intelligent Vehicles Symposium*, pp. 72-77

S. Khan and M. Shah (2003) Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, pp. 1355-1360

A. J. Lipton, H. Fujiyoshi, and R. S. Patil (1998) Moving Target Classification and Tracking from Real-time Video, *DARPA Image Understanding Workshop*, Monterey

J. MacCormick and A. Blake (1998) A Probabilistic Contour Discriminant for Object Localisation, *Proceedings of the International Conference on Computer Vision*

K. Martinez, J. K. Hart, and R. Ong (2004) Environmental sensor network applications, *IEEE Spectrum*, pp. 50-56

D. Murray and A. Basu (1994) Motion Tracking with an Active Camera, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.. 16, No. 5, pp. 449-459

K. Nummiaro, E. Koller-Meier, and L. Van Gool (2002) A Color-based Particle Filter, *1st International Workshop on Generative-Model-Based Vision*, pp. 53-60

N. Ohta and K. Kanatani (1995) Optimal Structure from Motion Algorithm for Optical Flow, *IEICE Transactions on Information and Systems*, Vol. E78-D, No. 12, pp. 1559-1565

J. Oliensis (1999) A Multi-frame Structure-from-Motion Algorithm under Perspective Projection, *International Journal of Computer Vision*, Vol. 34, pp. 163-192

N. Peterfreund (1999) Robust Tracking of Position and Velocity with Kalman Snakes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 6, pp. 564-569

R. Pless, T. Brodský, and Y. Aloimonos (2000) Detecting Independent Motion: The Statistics of Temporal Continuity, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 768-773

Y. Raja, S. J. McKenna, and S. Gong (1998) Colour Model Selection and Adaptation in Dynamic Scenes, Department of Computer Science, Queen Mary and Westfield College, England; Department of Applied Computing, University of Dundee, Scotland.

S. Saltenis and C. S. Jensen (2002) Indexing of Moving Objects for Location-Based Services, Department of Computer Science, Aalborg University

O. Shakernia, R. Vidal, and S. Sastry () Infinitesimal Motion Estimation from Multiple Cameras, *IEEE Workshop on Vision and Motion Computing*, Orlando, FL, pp. 229-234

J. Shin, S. Kim, S. Kang, S. Lee, J. Paik, B. Abidi, and M. Abidi (2005) Optical Flow-based Real-Time Object Tracking using Non-Prior Training Active Feature Model, *Elsevier Real-Time Imaging*, Vol. 11. pp. 204-218

S. Soatto and P. Perona (1994) Recursive Estimation of Camera Motion from Uncalibrated Image Sequences, *IEEE International Conference on Image Processing*, Austin, pp. 58-62

S. Soatto, P. Perona, R. Frezza, and G. Picci (1993) Recursive Motion and Structure Estimation with Complete Error Characterization, *IEEE Conference on Computer Vision and Pattern Recognition*, New York

C. Stauffer and W. E. L. Grimson (2000) Learning Patterns of Activity using Real-Time Tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 747-757

A. Stefanidis, K. Eickhorst, P. Agouris, and P. Partsinevelos (2003) Modeling and Comparing Change using Spatiotemporal Helixes, *ACM-GIS'03*, ACM Press, pp. 86-93

A. Stefanidis, Ch. Georgiadis, P. Agouris (2003) Registration of Urban Imagery to VR Models Through Radiometric Queries, *Videometrics VII, SPIE Proceedings*, Vol. 5013, pp. 176-185, Santa Clara, CA

A. Stefanidis and S. Nittel (2004) *GeoSensor Networks*, CRC Press, Boca Raton.

A. Strehl and J. K. Aggarwal (2000) A New Bayesian Relaxation Framework for the Estimation and Segmentation of Multiple Motions, *IEEE Southwest Symposium on Image Analysis and Interpretation*, Austin, Texas, USA

R. Szeliski and S. B. Kang (1993) Recovering 3D Shape and Motion from Image Streams using Non-Linear Least Squares, tech. report, Robotics Institute, Carnegie Mellon University

H. Tao, H. S. Sawhney, and Rakesh Kumar (2002) Object Tracking with Bayesian Estimation of Dynamic Layer Representations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, pp. 75-89

W. B. Thompson (1998) Exploiting Discontinuities in Optical Flow, *International Journal of Computer Vision*, pp. 163-173

P. H. S. Torr (1996) Geometric Motion Segmentation and Model Selection, *Philosophical Transactions of the Royal Society A*, pp 1321—1340

P. H. S. Torr, R. Szeliski, and P. Anandan (2001) An Integrated Bayesian Approach to Layer Extraction from Image Sequences, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 3, pp. 297-303

G. Tziritas (1992) A New Pel-Recursive Kalman-based Motion Estimation Method, *European Signal Processing Conference*, pp. 1341-1344

N. Vasconcelos and A. Lippman (1997) Empirical Bayesian EM-based Motion Segmentation, *IEEE Conference in Computer Vision and Pattern Recognition*, San Juan, Puerto Rico

V. Venkataraman, S. Srinivasan, and A. Stefanidis (2004) Object Color Propagation Across Disjoint Camera Feeds, *IEEE International Conference on Image Processing*, Singapore.

K. N. Walker, T. F. Cootes, and C. J. Taylor (1998) Locating Salient Object Features, *Proceedings of the Ninth British Machine Vision Conference*

J. Y. A. Wang and E. H. Adelson (1993) Layered Representation for Motion Analysis, *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, New York, pp. 361-366

J. Weber and J. Malik (1994) Robust Computation of Optical Flow in a Multi-Scale Differential Framework, *International Journal of Computer Vision*, Vol. 2, pp. 5-19

J. Weber and J. Malik (1997) Rigid Body Segmentation and Shape Description from Dense Optical Flow Under Weak Perspective, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, pp. 139-143

J. Wei (2002) Color Object Indexing and Retrieval in Digital Libraries, *IEEE Transactions on Image Processing*, 11(8), pp. 912-922

J. S. Zelek (2002) Bayesian Real-Time Optical Flow, *International Conference on Vision Interface*, Calgary, Canada

Y. Zhou and H. Tao (2003) A Background Layer Model for Object Tracking through Occlusion, *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 1079-1085

BIOGRAPHY OF THE AUTHOR

Sabeshan Srinivasan was born in the lovely coastal city of Vishakhapatnam, Andhra Pradesh state, India on June 11, 1981. He was raised in different cities in India. He spent the first decade of his life in the eastern metropolis of Kolkata, where he received his elementary and middle schooling. His family moved to the southern metro Chennai, where he has stayed since 1992. He received his high school diploma there and went on to pursue his undergraduate degree (Bachelor of Engineering) in Geoinformatics Engineering at the College of Engineering, Guindy (CEG), Anna University. CEG is one of the oldest engineering colleges in the world, having been established in 1794.

After completing his B.E. degree, he arrived in the United States in the fall of 2003 when he started on his Master of Science program in Spatial Information Science and Engineering at the Department of Spatial Information Science and Engineering, the University of Maine, Orono. After receiving his degree, Sabeshan will be joining ESRI Inc. as a software developer to kickstart his career in GIS/Database development. Sabeshan is a candidate for the Master of Science degree in Spatial Information Science and Engineering from the University of Maine in May, 2006.