

# VIDEO ANALYSIS AND COMPRESSION FOR SURVEILLANCE APPLICATIONS

A Dissertation  
Presented to  
The Academic Faculty

By

Sanmati Savadatti-Kamath

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy  
in  
Electrical and Computer Engineering



School of Electrical and Computer Engineering  
Georgia Institute of Technology  
December 2008

Copyright © 2008 by Sanmati Savadatti-Kamath

# VIDEO ANALYSIS AND COMPRESSION FOR SURVEILLANCE APPLICATIONS

Approved by:

Dr. David V. Anderson,  
Committee Chair  
*Professor, School of ECE*  
*Georgia Institute of Technology*

Dr. Patricio Vela  
*Asst. Professor, School of ECE*  
*Georgia Institute of Technology*

Dr. Joel R. Jackson, Advisor  
*Director, Digital Media Labs,*  
*School of ECE*  
*Georgia Institute of Technology*

Dr. David Scott  
*Professor, School of Civil and Environmental*  
*Engineering*  
*Georgia Institute of Technology*

Dr. Russell Mersereau  
*Professor, School of ECE*  
*Georgia Institute of Technology*

Date Approved: November 2008

*To my dear Ananya, Sunil, Sangu, and my parents. And to Nischu, I miss you  
dear brother...*

## ACKNOWLEDGMENTS

First and foremost I would like to thank my parents for everything that they have done for me. I would also like to take this opportunity to thank the committee members for their time and feedback. I would also like to thank my advisor Dr. Jackson for his help and support during the course of my research. Dr. Anderson was always available for any discussion and I thank him for his time.

This work would not be possible without the support and wishes of my large family and many friends. My heartfelt thanks to Sourabh, Shyam, Faik, Walter and Jung-Won for their friendship and support. You guys are the best. Many many thanks to Arti for her wonderful friendship and company during study sessions at Barnes and Nobel stores and libraries. I thank my friend Roma for her unconditional love and support. I would also like to thank Uday, Lalitha and her family for their well wishes.

Neeraj and Corrinne were always there for me and my family with their love, support and guidance. I would also like to thank Jeremiah Golston and Jagadeesh Sankaran from Texas Instruments for their support and wishes during the course of my internships and work at TI.

My sister Sangu and her husband David helped me keep my humor when I was down and for this I owe them. Aayi, mummy, dodamma, papa and others helped me at various times during the course of this work and their love and help will not be forgotten. Thank you all for being their for Ananya, Sunil and me.

Sunil was my rock and I thank him for his faith and unflinching confidence in my abilities.

Last but not the least, I thank my daughter Ananya for her love and patience during the course of this work. My hope is that she reads this research thesis some day and feels inspired to do something bigger and way better.

# TABLE OF CONTENTS

<b>ACKNOWLEDGMENTS</b> . . . . .	iv
<b>LIST OF TABLES</b> . . . . .	vii
<b>LIST OF FIGURES</b> . . . . .	viii
<b>SUMMARY</b> . . . . .	1
<b>CHAPTER 1 INTRODUCTION</b> . . . . .	3
1.1 Previous Work in Vehicle Detection and Surveillance . . . . .	3
1.2 Compression Standards . . . . .	8
1.3 Future Trends in Camera and Video Technologies . . . . .	11
1.4 Scope of this research . . . . .	15
<b>CHAPTER 2 VIDEO ANALYSIS BASED ON COLOR</b> . . . . .	18
2.1 Image Representation . . . . .	18
2.2 The HSI Color Space . . . . .	20
2.3 The HSV Color Space . . . . .	20
2.4 Research Contributions: HSV domain and light-shade discrimination	22
2.5 Direction and direction changes . . . . .	24
2.6 Color Segmentation in the HSI domain . . . . .	30
2.7 Vector Angle Computation on RGB Data . . . . .	31
2.8 Research Contributions: The Vector Angle With Absolute Difference (VA-D) . . . . .	32
2.8.1 Difference Vector Edge Detector . . . . .	33
2.8.2 Saturation and Value based Combination . . . . .	33
2.8.3 Results of VA-D Approach . . . . .	36
2.8.4 Detecting and Connecting Straight Line Edges . . . . .	36
2.8.5 Hough Transform and its Variants . . . . .	40
2.8.6 Research Contributions: Line Connecting Algorithm using the VA-D and Canny Re- sults (LC) . . . . .	42
<b>CHAPTER 3 COLOR DESCRIPTOR</b> . . . . .	45
3.1 Introduction . . . . .	45
3.2 Research Contributions: RGB Vector Angle based Color Histogram Descriptor . . . . .	48
3.2.1 Similarity Measure . . . . .	50
3.2.2 Complexity . . . . .	51
3.2.3 Precision, Recall and Normalized Retrieval Ranks . . . . .	51
3.3 Research Contributions: Modified RGB histogram . . . . .	59

<b>CHAPTER 4</b>	<b>A COMPRESSION SCHEME FOR VIDEO SURVEILLANCE . . . . .</b>	<b>66</b>
4.1	Related Background Work . . . . .	67
4.2	Research Contributions . . . . .	68
4.2.1	Motion JPEG with Differential Encoding (MJPEG-DE) . . . . .	69
4.2.2	MJPEG-DE with Motion Segmentation . . . . .	70
4.2.3	Results of MJPEG-DE with Motion Segmentation . . . . .	72
<b>CHAPTER 5</b>	<b>VIDEO SUMMARIZATION . . . . .</b>	<b>76</b>
5.1	Video Classification Methodologies . . . . .	77
5.2	Research Contributions . . . . .	79
5.2.1	Video Summarization for Surveillance . . . . .	79
5.2.2	Context Header . . . . .	79
5.2.3	Semantic Coder . . . . .	80
<b>CHAPTER 6</b>	<b>CONCLUSION AND FUTURE WORK . . . . .</b>	<b>88</b>
6.1	Future Work . . . . .	90
6.2	Applications of this research . . . . .	91
<b>APPENDIX A</b>	<b>HAUSDORFF DISTANCE MEASURE . . . . .</b>	<b>92</b>
A.0.1	Generalized Hausdorff Distance . . . . .	92
<b>APPENDIX B</b>	<b>CONNECTED COMPONENTS . . . . .</b>	<b>94</b>
<b>REFERENCES</b>	<b>. . . . .</b>	<b>96</b>

## LIST OF TABLES

Table 1	Saturation and Intensity ratios for images given in Figures 6 to 10 .	24
Table 2	Decision (D) for test images . . . . .	34
Table 3	Computations per pixel for RGB to HSI conversion . . . . .	47
Table 4	Computations per pixel for RGB to HSV conversion . . . . .	47
Table 5	Video prioritization based on motion . . . . .	72
Table 6	Compression ratios between MJPEG and MJPEG-DE system . . .	75
Table 7	Speed of execution between H.264 and proposed system . . . . .	75
Table 8	Context Header Example . . . . .	79

## LIST OF FIGURES

Figure 1	Ubiquitous Home Sensors and microphone layout [1] . . . . .	13
Figure 2	Overall processing for Ubiquitous Home . . . . .	14
Figure 3	Overall block diagram of system described in this research . . . . .	16
Figure 4	HSI color space. [2] . . . . .	20
Figure 5	HSV color space as a conical object. [3] . . . . .	21
Figure 6	a) Image of blocks in dark b) Saturation values c) Intensity values .	25
Figure 7	a) Image of blocks in low light b) Saturation values c) Intensity values	26
Figure 8	a) Image of blocks in bright light b) Saturation values c) Intensity values . . . . .	27
Figure 9	a) Image of house in low light b) Saturation values c) Intensity values	28
Figure 10	a) Image of house in bright light b) Saturation values c) Intensity values . . . . .	29
Figure 11	a)'Still1' image b)Figure showing Canny edge detector result c)Figure showing the result of running the VA-Euclidean method described in [4] d)Figure showing result from running the proposed VA-D method.	37
Figure 12	a)'Container4' image [5] b)Figure showing Canny edge detector result c)Figure showing the result of running the VA-Euclidean method described in [4] d)Figure showing result from running the proposed VA-D method. . . . .	38
Figure 13	a)'Container7' image [6] b)Figure showing Canny edge detector result c)Figure showing the result of running the VA-Euclidean method described in [4] d)Figure showing result from running the proposed VA-D method. . . . .	39
Figure 14	a)Broken segments after VA computation b) Segments reconnected after applying LC algorithm to connect lines. . . . .	44
Figure 15	a)Broken segments after VA computation b) Segments reconnected after applying LC algorithm to connect lines. . . . .	44
Figure 16	Cycle count comparison between HSI-SCD and proposed method .	52
Figure 17	Precision values comparison between HSI-SCD and proposed method	53

Figure 18	Recall values comparison between HSI-SCD and proposed method .	54
Figure 19	Retrieval Results for query image 1 . . . . .	55
Figure 20	Retrieval Results for query image 2 . . . . .	56
Figure 21	Retrieval Results for query image 3 . . . . .	57
Figure 22	Retrieval Results for query image 4 . . . . .	58
Figure 23	A sample of database contents . . . . .	62
Figure 24	Results of the modified histogram matching . . . . .	63
Figure 25	Results of the modified histogram matching . . . . .	64
Figure 26	Precision values for descriptor based on vector angle and the modified descriptor based on Sobel edge detection and Vector angle . . . . .	65
Figure 27	MJPEG DE (a) Encoder and (b) Decoder System . . . . .	71
Figure 28	a) Single input frame and b) Motion segmented image . . . . .	73
Figure 29	a) Single input frame and b) Motion segmented image . . . . .	74
Figure 30	Movie Structure [7] . . . . .	82
Figure 31	Flow graph the encoder with video semantic data embedded - Part A	83
Figure 32	Flow graph the encoder with video semantic data embedded - Part B	84
Figure 33	Flow graph for the decoder . . . . .	85
Figure 34	Content Adaptive Encoding (A) and Decoding (B) proposed by ATT Corporation [8] [9] . . . . .	86
Figure 35	Textual Summary after Parsing Context Header at the Decoder . . . . .	87
Figure 36	Block diagram of the complete system addressed in this research . . . . .	89
Figure 37	Figure showing the computation of the Hausdorff Distance. . . . .	93
Figure 38	a)Image with connected regions b) Image with connected components grouped by labels. . . . .	94

## SUMMARY

With technological advances digital video and imaging are becoming more and more relevant. Medical, remote-learning, surveillance, conferencing and home monitoring are just a few applications of these technologies. Along with compression, there is now a need for analysis and extraction of data. During the days of film and early digital cameras the processing and manipulation of data from such cameras was transparent to the end user. This transparency has been decreasing and the industry is moving towards 'smart users' - people who will be enabled to program and manipulate their video and imaging systems. Smart cameras can currently zoom, refocus and adjust lighting by sourcing out current from the camera itself to the headlight [10]. Such cameras are used in the industry for inspection, quality control and even counting objects in jewelry stores and museums, but could eventually allow user defined programmability. However, all this will not happen without interactive software as well as capabilities in the hardware to allow programmability.

In this research, compression, expansion and detail extraction from videos in the surveillance arena are addressed. Here, a video codec is defined that can embed contextual details of a video stream depending on user defined requirements creating a video summary. This codec also carries out motion based segmentation that helps in object detection. Once an object is segmented it is matched against a database using its shape and color information. If the object is not a good match, the user can either add it to the database or consider it an anomaly.

RGB vector angle information is used to generate object descriptors to match objects to a database. This descriptor implicitly incorporates the shape and color information while keeping the size of the database manageable. Color images of objects that are considered 'safe' are taken from various angles and distances (with the same background as that covered by the camera is question) and their RGB vector

angle based descriptors constitute the information contained in the database.

This research is a first step towards building a compression and detection system for specific surveillance applications. While the user has to build and maintain a database, there are no restrictions on the size of the images, zoom and angle requirements, thus, reducing the burden on the end user in creating such a database. This also allows use of different types of cameras and doesn't need a lot of up-front planning on camera location, etc.

# CHAPTER 1

## INTRODUCTION

In this section, past research in object and vehicle detection techniques are addressed. Additionally, an overview of some popular video compression standards is also provided.

### 1.1 Previous Work in Vehicle Detection and Surveillance

Some vehicle and intruder detection techniques are discussed in this section. Most of these systems use both the spatial and temporal details in some form to detect and track objects.

Nakanishi and Ishii introduced a system that extracted moving vehicles based on spatio-temporal information [11]. Here the camera was aimed perpendicular to a straight path and moving vehicles were tracked along a slit which was assumed parallel to the path such that the slit was one of the image's scan lines. The x-axis was assumed to be parallel to this slit. If a vehicle moved at constant velocity, it formed a straight line along the spatio-temporal image. A vehicle moving from left-to-right had a positive slope and a vehicle moving from right-to-left had a negative slope. An intersection between the lines in the spatio-temporal image indicated occlusion and a change in slope indicated changing acceleration. Using the Hough transform, occluded vehicles were extracted.

In this system, the background was extracted based on the background information in the slit provided there was an appropriate inter-vehicle gap along a lane [11]. To extract vehicles, long sequences of subtraction images were superimposed using the velocity information obtained from the spatio-temporal image. Gaussian noise suppression was carried out so as to segment out the contour of each vehicle. To

remove occlusion, the overlapped images of vehicles were removed from each subtraction image. The subtraction image was then shifted along the slit x-axis according to the target vehicle's velocity [11]. This method was also applied to classification based on silhouette analysis. For this, the angular orientation histograms were used. The extended circular image (ECI) was generated from the extracted vehicle image. The ECI has peaks corresponding to the orientation of the silhouette. A nearest neighbor classification was applied to classify the ECI data and recognition was carried out.

Another work targeted the general surveillance problem with multiple objects in motion in different directions [12]. Here, the optical flow was first computed and regions with uniform flow were extracted. Assuming that the objects had constant speeds, a path was predicted for the objects by voting in the spatio-temporal image. Depending on the vote count for a particular path, detection of a moving object was reported.

In [12], it was argued that simple image subtraction for object detection does not work in all cases. For, example, in a video stream with a moving human and trees shaking in the breeze, simple subtraction does not adequately help in segmentation as there are brightness changes in almost all regions of the image due to the moving trees. Hence, here, a method of obtaining the optical flow and splitting and merging regions based on the mean flow was carried out to segment region based on motion. Next, a four-dimensional voting space was created and a path was predicted for every region in for a certain duration of time. Assuming the object was moving along the predicted path, votes for the target were accumulated. When these votes exceeded a predefined threshold, an intruder was reported.

A method which deals with extraction and tracking of the license plates was described in [13]. Here, the camera's image plane was assumed to be parallel to the vehicle's license plate. First, the Sobel filter and the Hough transform were applied for extraction. The Sobel filter was applied for line detection. Then, the edges in the

upper regions of the image were eliminated and the Hough transform was applied to the rest of the image. It was assumed that the remaining lines depict the edges for the vehicle as well as the license plate. Next, the aspect ratio of the license plate was used to extract the rectangular region.

The extracted license plate was then tracked using the Voted Block Matching (VBM) technique for template matching. In VBM, the template is divided into several blocks and correlation matching is applied to each block in the object frame. The 'vote space' is a 2-D region giving the  $\Delta x$  and  $\Delta y$  coordinates along the X and Y direction respectively. The  $(\Delta x, \Delta y)$  coordinate pair, corresponding to the object block that has the highest correlation with a template block, gets the vote. The coordinate pair with maximum votes was considered the matching coordinate pair for a template. In case of occlusion, the votes were scattered. Hence, the correct position could be matched without the influence of occlusion [13].

Some drawbacks of this method were that the extraction was influenced by the color of the license plates as well as the number of vehicles, i.e., tracking operation became cumbersome when multiple vehicles are present.

Another work dealt with automatic segmentation of moving objects for video object plane (VOP) generation [14]. VOPs describe some semantically meaningful content of a video. It was argued here that motion estimation and optical flow techniques are extremely noise sensitive with limited accuracy due to the aperture problem. Change detectors or image differencing methods mark occluded objects as changed while they are in reality unchanged. Hence, the Hausdorff<sup>1</sup> distance measure was applied to binary edge images where the edge pixels were considered the features to be tracked.

In brief, this work did the following- moving objects were initialized by taking frame differences for color or intensity. High values in the difference image indicate

---

<sup>1</sup>See Appendix A for more details on the Hausdorff distance measure.

moving objects. For non-textured objects, only the outline is seen. Thinning was carried out to ensure that the edges were one pixel wide only. The model was updated every frame and hence missing components were picked up eventually. Next, connected components labeling was carried out and components larger than a predefined threshold were assumed to belong to a moving object and were called moving connected components (MCC).

The following mechanism was used to implement the Hausdorff distance for object tracking. It was assumed that the Hausdorff distance is smaller than a threshold  $T$  so that bad matches were avoided. First, the directed Hausdorff distance was computed such that for each image pixel, the distance  $h_k(O, I)$  to the nearest edge pixel was known (It should be noted here that  $h_k(O, I) \leq T$  only are retained). Next, for all translations  $t = (t_x, t_y)$  the object  $O$  was translated by  $t$  and distance transform computed earlier gave the distance between the translated object and the image edge points. These distances were then sorted in ascending order and the  $k^{th}$  value was selected to obtain  $h_{k,t}(O, I)$ , in accordance with the generalized<sup>2</sup> Hausdorff distance. For these translations,  $h_{l,t}(I, O)$  was also computed and to obtain  $H_t(O, I)$ . The smallest Hausdorff distance  $H_t(O, I)$  gave the translation the object has undergone.

In the case of moving background, correspondence vector field was calculated by hierarchical block matching. Global motion is generally due to zooming, panning or translation. Thus, large motion vectors can be attributed to moving objects. Moreover, backgrounds cover large areas compared to the object. Thus, blocks moving differently from the global motion were compared to the correspondence vectors estimated from a affine global motion model. Then, connected blocks with coherent motion were assumed to belong to moving objects if they were different from the global motion [14].

The model update process consisted of two parts, one for slow moving parts and

---

<sup>2</sup>As described in Appendix A

one for rapid moving parts. The object tracking method using Hausdorff distance described above was applied to parts with rigid motion keeping in mind the threshold  $T$ . For parts with nonrigid motion, the MCCs were used. Components that were connected to tracked objects were included in the updated model. These components were assumed to be connected if they were within a specific distance of the MCCs. The combination of the above update methods results in a model that worked for both slow and fast moving components.

Along with the above processes, a counter was maintained for edge pixels that were classified as edges. Edges that were classified more than a threshold value, were eliminated. This avoided erroneous classification of moving objects that had seized to move from being classified as background.

This method resulted in good object tracking while avoiding the drawbacks of motion estimation techniques namely, sensitivity to noise and computational complexity.

Another work utilized adaptive mesh based video object tracking scheme to compute motion trajectories of all node points [15]. Motion and shape information of the VOPs are tracked frame by frame using mesh based algorithms to track the temporal evolution of multiple objects. For indexing, the first appearance of the video object (VO) should be known and its tracked mesh node trajectories. Node motion vectors are transformed to the Hough space. The maximum number of motion vectors corresponding to a Hough bin indicate dominant motion direction for that video object plane (VOP). The Hough angles are further quantized to limit the bin size to eight, namely  $angle \in 0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 325^\circ$ . A separate state of no motion is also included. This direction information is included with camera motion information assigned to each VOP and requires 4 bits.

In the OBSERVER system described in [16], object segmentation was carried out and tracks for the objects were created based on certain criteria like object mask, type, appearance and shape. Two types of events were tracked, unusual behavior

and violation of restricted areas. The first type of event was tracked by classification, and the second one needs the user to mark restricted areas. N-ary tree classifier was constructed by using a set of prerecorded tracks consisting of image attributes. The data was partitioned into equal time slices and for each attribute the mean value was computed. Next, clustering was performed using the Expectation-Maximization algorithm and automatic cluster number detection was carried out.

## 1.2 Compression Standards

Commercial surveillance systems use multiple cameras that transmit over a network, allowing multiple views on a remote screen. They typically use JPEG, MPEG and H.264 compression schemes. One such commercial system advertised in [17] uses changes in motion, temperature, microphone volume and infrared detector to trigger events which then result in ‘alarms’ being sent via emails or as a telephone messages to monitoring centers. Thus, compression is a key requirement of such systems and necessitates technology capable of running such complex codecs. This section gives a brief introduction to some of the compression standards applicable to surveillance.

The International Standards Organization (ISO) has been involved in developing the JPEG and JPEG2000 standards for image compression and Motion-JPEG, MPEG-1, MPEG-2 and the MPEG-4 standards for video compression mainly catering to the needs of the ‘media industry’. The Video Coding Experts Group (VCEG) belonging to the International Telecommunications Union (ITU) has developed the H.261, the H.263 and the H.264 video coding standards targeting real-time, point-to-point or multi-point communications [18].

Joint Photographic Experts Group (JPEG) technology operates in the image transform and is lossy compression. JPEG2000 is their latest effort that applies quantization to sub-band coded image components to achieve higher orders of compression than JPEG and most importantly avoids the blocking effects seen in JPEG

compressed files at lower quality. Motion-JPEG uses JPEG compression techniques on every frame of a video. This method is still quite popular for video conferencing and surveillance applications.

The Movie Picture Experts Group (MPEG) standards take into consideration the motion details and background redundancies between frames in a video. There are three main types of pictures or frames: I-frames, P-frames and B-frames. Each frame of a video is encoded as I, P or B type to produce a coded picture. I frames are intra-coded without any motion compensated prediction. The P-frames are motion compensation predicted (also known as inter-coded) using a reference frame. P-frames are forward predicted and may be used as references for future predictions. The B-frames are inter-coded using two reference frames, either the P and/or I frames before and after the current B frames. Because of this, the MPEG standards result in highly complex systems while providing low bit-rates. MPEG-1 was intended for compression for storage and playback on CD-ROMS. The more popular standard, MPEG-2 was designed for digital television and has the capability of efficiently supporting larger frame sizes (typically 720 x 576 or 720 x 480 pixels for ITU-R 601 resolution) and coding of interlaced videos. It has the added flexibility of separately coding the two fields that make up an interlaced video frame. MPEG-4 was developed for complete video frames and moved towards object based coding rather than just a series of rectangular frames. Background and foreground are separately coded. Various possibilities result from such coding like scene reuse, coding of foreground and background with different qualities etc.

The underlying idea for MPEG-4 is to divide a video scene into various video objects (VO). A video object plane (VOP) is a 'snapshot' of a video object at any given instance of time. A VOP is equivalent to a frame in MPEG-1 and MPEG-2. In MPEG-4 a frame can be an I-frame or a P-frame. MPEG-4 uses transform coding for I-frames and motion estimation and detection for P-frames. There are

some additional features that set MPEG-4 apart from its predecessors. These are listed below:

- Shape Coding: Shape coding is required to specify boundaries of video objects. The shape information is either in binary or 8-bit gray scale. Pixels lying inside a VOP are opaque, whereas pixels lying outside a VOP are transparent.
  - If all pixels are transparent, no shape or texture information needs to be coded.
  - If all pixels are opaque, then no shape information needs to be coded. Texture information, however, is coded.
  - If both opaque and transparent pixels exist, then the block crosses the boundary of the VOP. The shape values are coded using DPCM and texture information is coded using texture coding described below.
- Texture Coding: Pixels within a VOP are coded as texture. Transform coding is used here, with an additional step to predict the quantized DCT coefficients from previous VOPs, for better compression. Macroblocks lying on the boundaries of VOPs have both the transparent (lying outside the VOP) and opaque pixels. In inter coded VOP texture information is motion compensated and hence the transparent pixels are set to zero. In intra coded VOPs transparent pixels are computed by extrapolating pixel values along the VOP boundaries.
- Sprite Coding: VOPs present for a long time or the entire duration of a video are called Sprites. For some videos, the background can be a sprite if it is unchanging for the entire duration of the video. If there is camera movement, then warping is used to create sprites larger than the visible scene [18].

The H.261 codec<sup>3</sup> typically operates at bit rates of 64-384 kbps. The 4:2:0 Y:Cr:Cb

---

<sup>3</sup>A complete compression system which includes both the coder and the decoder is referred to as a codec.

format is used where there are four luminance blocks and two chrominance blocks each 8 x 8 in size. This coder introduced the concept of the loop filter. The loop filter is typically used before motion compensation and smooths the reference picture leading to better prediction. Each picture is intra or inter coded as an I or P frame respectively. Two frame sizes are supported namely the CIF (355 x 288 pixels) and the QCIF (176 x 144 pixels). This codec is low in complexity, but has suffers from low compression and poor video quality below 100 kbps.

The H.263 codec operates at bit rates lower than 20 kbps and is mainly used for internet based applications. The H.263 provided a larger number of frame size options. It also has 19 optional modes on top of the baseline mode. Picture coding is done using intra and inter coding and video data is processed in the 4:2:0 Y:Cb:Cr format. This method achieves better compression than H.261 due to features like half-pixel motion vectors and redesigned Variable Length Code tables [18].

The H.264 codec aimed at significantly increasing the coding efficiency while reducing the number of options that were provided in the H.263 codec. It has most of the features of its predecessors like intra and inter frame coding, loop deblocking, sub-pixel motion vectors and the content based adaptive binary arithmetic coding. It also utilizes the B-frames for applications not sensitive to transmission delays.

### **1.3 Future Trends in Camera and Video Technologies**

The compression schemes discussed in the previous chapter are geared towards exploiting the physical or formative<sup>4</sup> characteristics of a video stream rather than its content. This makes most of these codecs good for compression, but incapable of utilizing or conveying semantic<sup>5</sup> details pertaining to the video. This is commonly referred to as the ‘semantic gap’.

The Human Computer Interaction (HCI) field has been addressing this semantic

---

<sup>4</sup>Pertaining to formation or development of a video content.

<sup>5</sup>The study of meaning of video content.

gap issue from some years now. There have been studies on homes and offices that are completely wired to detect and record the subtlest changes. Such systems would be useful in monitoring homes of the elderly and home surveillance applications. But most of their approaches and solutions involve sensors, microphones, wired floors etc. to collect semantic information [19, 20, 1]. Figure 1 gives an idea of such home setup and Figure 2 shows the processing system.

Such systems are financially feasible in research, corporate or formal health monitoring situations. But, technological advances in memories, high speed, general purpose processors along with broadband networking have enabled penetration of multimedia technologies into peoples' homes. For such users, low cost is one important criteria. Moreover, programmability coupled with ease of use is an important factor. Installing microphones, sensors and their setup can be daunting enough, not to mention analyzing their output to glean useful information.

Cell phones and digital cameras have put processing in the hands of individuals like never before. This has lead to the concept of smart cameras. Current smart cameras have the ability to adjust to zoom and light changes depending on the environment. But, the idea of allowing users to manipulate the software that goes into a camera and customize it for their own goals is now not seeming far-fetched. Whatever features get incorporated in the future, such systems still need to be low cost, low power consuming and come in manageable sizes. Low cost does not mean fewer capabilities though, and users would still expect such systems to be able to accomplish high speed processing, color conversions, compression and other image and video processing like jitter reduction, automatic face detection, color and edge enhancements and noise removal. Thus, the aim of such systems should be to accomplish the required results with algorithms or hardware units that are as computationally constrained as possible [21].

In the security area, smart videos are now able to indicate abnormal situations to

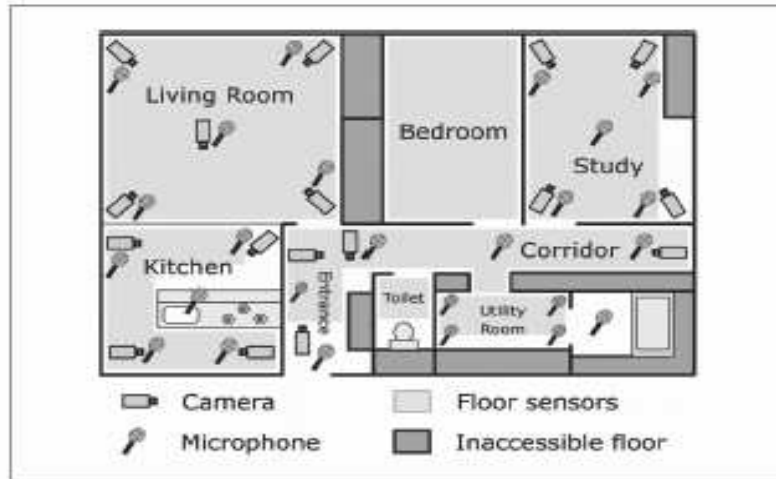


Figure 1. Ubiquitous Home Sensors and microphone layout [1]

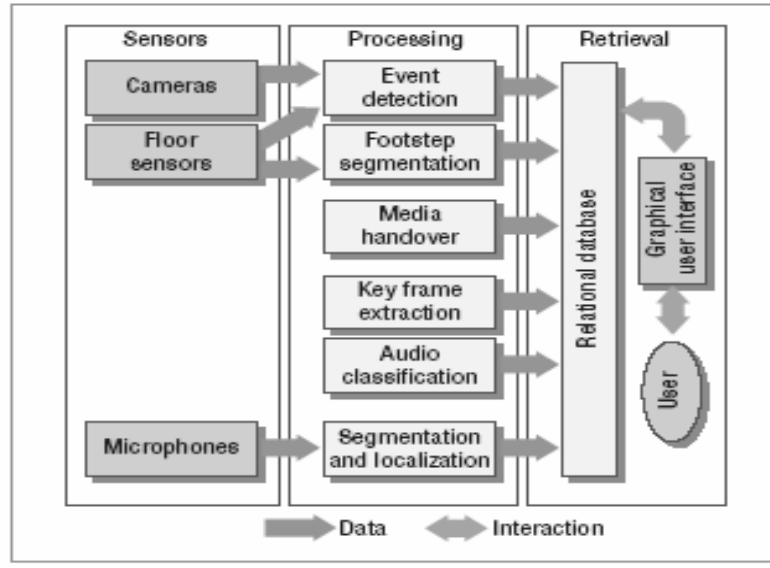


Figure 2. Overall processing for Ubiquitous Home

human operators by analyzing human movement, vehicle movement, counting items and objects etc. The 2008 Beijing Olympic games are a prime example of such surveillance where smart videos have been heavily deployed [22].

Thus future trends include processor reuse and systems on chip (SOC) solutions, that allow end-user programmability and mobile vision applications.

## 1.4 Scope of this research

In the work presented in this document, a complete surveillance scheme is presented with object detection, segmentation and matching and video compression as shown in Figure 3. A method to embed semantic details from the video pertaining to surveillance, giving a summary of the video is also presented. The goal has also been to provide a way for users to create and update a database of objects they feel are pertinent to the surveillance. In order to do this, the detector has to be robust enough to not have specific requirements on the images needed to create the database.

As motion is a key factor in surveillance, motion segmentation is used to detect objects, color information from the segmented object is used to match it to an existing database. These are discussed in detail in Chapters 2 and 3. Chapter 4 gives details on a compression scheme based on separately coding static background and foreground motion using motion JPEG and differential encoding. This method accomplishes both the motion segmentation (used in the object detection described in Chapter 3) and compression. Finally, a method of embedding the analysis information with any video codec to form a textual summary without increasing the compressed bitstream size by a large amount is given in Chapter 5.

In summary, the contributions of this work are as listed below:

- Change detection based on color.
- RGB vector angle based descriptor for object detection.

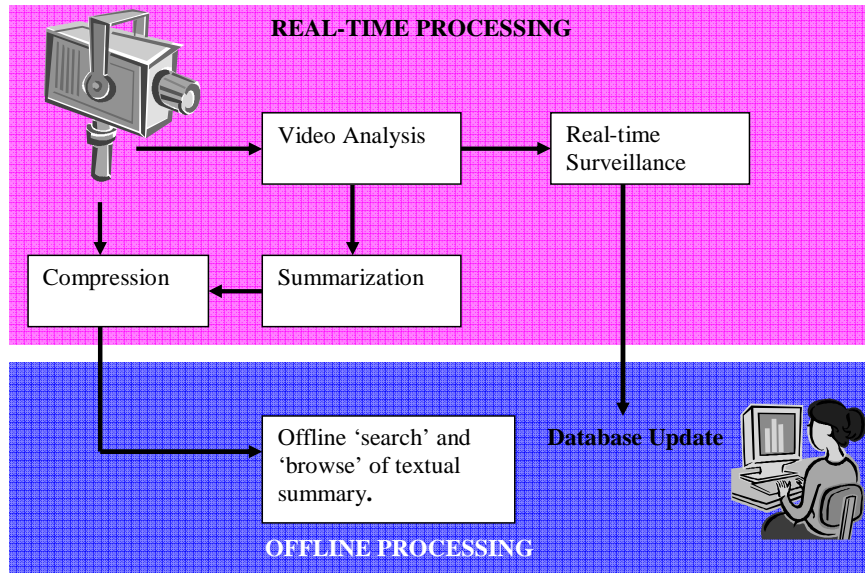


Figure 3. Overall block diagram of system described in this research

- Compression and summarization techniques for semantic coding.
- Capabilities for user interaction in creating and updating an object database.

## CHAPTER 2

### VIDEO ANALYSIS BASED ON COLOR

Image analysis is a crucial first step for object recognition, image registration, compression etc. Such analysis includes, but is not confined to object segmentation, edge detection, motion segmentation at a formative level, and light/shade difference detection, object identification, motion direction tracking and behavior interpretation at a semantic level.

In this chapter, various color space conversions are discussed and video analysis is carried out to compute attributes like light changes, motion, direction and color edge information. In surveillance these attributes help in obtaining semantic information.

#### **2.1 Image Representation**

Images can be represented in various ways. The most common form of representation is the NTSC RGB where three channels are used to depict red, blue and green color information respectively. The YUV representation gives the luminance and the chrominance information and is typically used for encoding television pictures. HSI and the HSV representations imitate the human visual system where hue represents the color information and saturation gives the degree of purity of hue. Intensity  $I$  or the Value  $V$  gives the brightness level. Typical conversion factors between the RGB representation and various representations are given in Equations 1-10 given in Section 2.3. Equations 8, 9 and 10 give the computations involved per pixel for converting from the RGB to HSI representation while Equations 6, 7 and 10 give the computations involved per pixel in converting from RGB to HSV domain. It is seen from the conversion Equation 1 to 3, conversion from RGB to YUV involves 9 multiplications and 6 addition/subtraction operations per pixel.

On the other hand, converting the RGB image to the HSV or the HSI domain

involves even more computations. For computing the hue values, obtaining the MIN and MAX takes 4 compares, at least 2 additional comparisons are required to decide on the exact hue equation to apply, 3 addition/subtraction operations and 1 multiplication and 1 division are also required for each pixel. For computing the saturation, 2 comparisons along with 2 additions and 1 division per pixel are required. The number of additions can be reduced by one by pre-computing (MAX - MIN) and using it for both the hue and saturation calculations. As can be seen from Equations 4 to 10, computing the HSV is less computationally complex than computing the HSI data.

Other color schemes include the  $L^*u^*v^*$  and the Munsell color space. The  $L^*u^*v^*$  color space originated from the  $L^*a^*b^*$  color space developed by the C.I.E. The Munsell space is based on human perception of color and defines a perceptually uniform color space. The  $L^*a^*b^*$  space tries to reduce the computational complexity of the Munsell space.  $L^*$  approximates the luminance component,  $a^*$  correlates with the red-green components and  $b^*$  correlates with the yellow-blue components.  $L^*u^*v^*$  evolved from the  $L^*a^*b^*$  space [23]. An analysis of color histograms by Gargi et. al. in [23] concluded that the  $L^*u^*v^*$  space worked best for video indexing. However, the conversion from RGB to  $L^*u^*v^*$  is very complex.

Segmentation techniques on gray scale images typically exploit intensity discontinuities and use difference measures on adjacent pixels. In color images, additional color information available can be utilized. The Hue, Saturation and Intensity (HSI) and the Hue, Saturation and Value (HSV) models closely resemble the human visual systems perception of color. Both these models employ the polar co-ordinate space. The Red, Green and Blue (RGB) model is based on physical interpretation of color. Various other models like the YCbCr (Y is the luminance and Cb and Cr are the chrominance values) and the CMY models are also popular.

The HSI and HSV color spaces are of interest because the hue values are independent of intensity or luminance. This has the advantage of intensity variations

due to illumination being ignored. However, this also has the disadvantage of being insensitive to differences between objects and their shadows when the shadow has the same color as the object.

## 2.2 The HSI Color Space

In the HSI space (also called the Hue, Saturation and Lightness - HSL space), a maximum value for I indicates the color white. The brightest color has an intensity value of exactly half the maximum [24]. Figure 4 shows the pictorial depiction of this model.

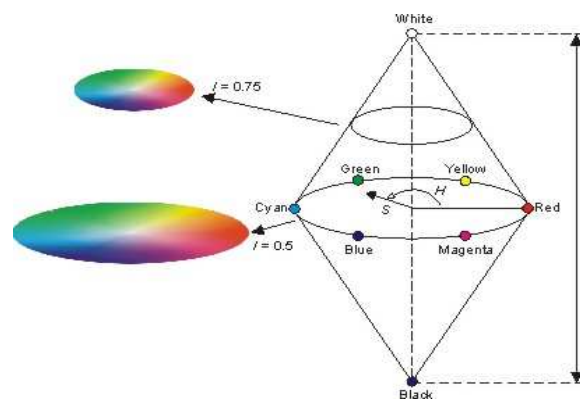


Figure 4. HSI color space. [2]

## 2.3 The HSV Color Space

In the HSV color space, hue stands for the color which is represented by a circular wheel and varies between  $0 - 360^\circ$ , as shown in Figure 5. Saturation and value are depicted as a triangle. The horizontal axis of the triangle gives the saturation (S) and the vertical axis gives the value (V). Both saturation and value range from  $0 - 100\%$  [25]. A maximum value for V indicates that the color is at its brightest and zero indicates the color black.

The saturation values indicate the degree of purity of hue. This means that the higher the saturation, the more relevant the hue information and low saturation

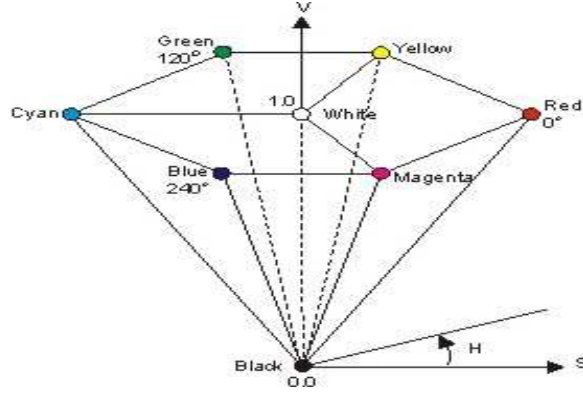


Figure 5. HSV color space as a conical object. [3]

values indicate gray scale pixels. When saturation is low, color information is low and intensity becomes more relevant [26].

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \quad (1)$$

$$U = -0.147 \times R - 0.289 \times G + 0.436 \times B \quad (2)$$

$$V = 0.615 \times R - 0.515 \times G - 0.100 \times B \quad (3)$$

$$MAX = \max(R, G, B) \quad (4)$$

$$MIN = \min(R, G, B) \quad (5)$$

$$Value_{hsv} = MAX \quad (6)$$

$$Saturation_{hsv} = \begin{cases} 0 & \text{if } (V \equiv 0) \\ \frac{MAX-MIN}{MAX} & \text{if } (V > 0) \end{cases} \quad (7)$$

$$Intensity_{hsi} = \frac{MAX + MIN}{2} \quad (8)$$

$$Saturation_{hsi} = \begin{cases} 0 & if(I \equiv 0) \\ \frac{MAX-MIN}{MAX+MIN} & if(0 < I \leq 0.5) \\ \frac{MAX-MIN}{2-MAX+MIN} & if(I > 0.5) \end{cases} \quad (9)$$

$$Hue_{hsi, hsv} = \begin{cases} \text{undefined} & if(MAX \equiv MIN) \\ 60 + \frac{G-B}{MAX-MIN} + 0 & if(MAX \equiv R)and(G \geq B) \\ 60 + \frac{G-B}{MAX-MIN} + 360 & if(MAX \equiv R)and(G < B) \\ 60 + \frac{B-R}{MAX-MIN} + 120 & if(MAX \equiv G) \\ 60 + \frac{R-G}{MAX-MIN} + 240 & if(MAX \equiv B) \end{cases} \quad (10)$$

## 2.4 Research Contributions: HSV domain and light-shade discrimination

Saturation and Value pixels are useful in differentiating between light and shade details. Saturation is a good indicator of color and low saturation might indicate low light conditions. However, even in ambient light, there is a chance of saturation retaining high values due to color content in an image. In such cases, Intensity gives a better way for discriminating between bright and low light.

Images can be classified as having three types for light conditions in terms of Saturation and Value: images with low intensity, high intensity and good visibility. The ones which are shot in the dark or low light conditions usually have low intensity levels. Usually images with good visibility are the ones that have a uniformly distributed saturation histogram or at least peaks that fall in both the low and high saturation bins.

Figures 6 to 10 give the saturation and values for different images. It can be seen that both saturation and intensity can be used to determine light conditions.

Following is a simple algorithm for quick determination of light conditions in images.

```
/* A simple algorithm to determine Saturation and Value histogram */
low_sat_cnt = 0;
high_sat_cnt = 0;
low_value_cnt = 0;
high_value_cnt = 0;
for ( row = 0; row < height; row++ )
{
    for( col = 0; col < width; col++ )
    {
        /* Here we compute the total low and high saturation (S) pixels.*/
        if ( S(i,j) < 0.1)
            high_sat_cnt = high_sat_cnt + 1;
        else if ( S(i,j) >= 0.4)
            high_sat_cnt = high_sat_cnt + 1;
        /* Here we compute the total low and high value (V) pixels.*/
        if ( V(i,j) < 0.5)
            low_value_cnt = low_value_cnt + 1;
    }
}
/* Obtain the high saturation and value counts by subtracting the respective low
counts. */
total = width * height;
low_sat_cnt = total - high_sat_cnt;
high_value_cnt = total - low_value_cnt;
sat_ratio = low_sat_cnt/high_sat_cnt;
value_ratio = low_value_cnt/high_value_cnt;
```

**Table 1. Saturation and Intensity ratios for images given in Figures 6 to 10**

Image Name	Saturation ratio	Intensity Ratio
Image of blocks in the dark	8.48	1013.0
Image of blocks in low light	0.50	69.84
Image of blocks in bright light	0.45	0.27
Image of house in low light	1.64	17.23
Image of house in bright light	0.90	1.25

If *sat\_ratio* is less than zero, then we can assume that the saturation in the image is quite good which leads to the conclusion that there is good hue content in the image.

If on the other hand, *sat\_ratio* is greater than zero, then the number of pixels that are low in saturation is large and the hue information contained in the image might be low.

A *value\_ratio* of less than one indicates a large number of high intensity pixels, which means good lighting conditions.

A *value\_ratio* of greater than one indicates a large number of low intensity pixels, which might mean a dark image.

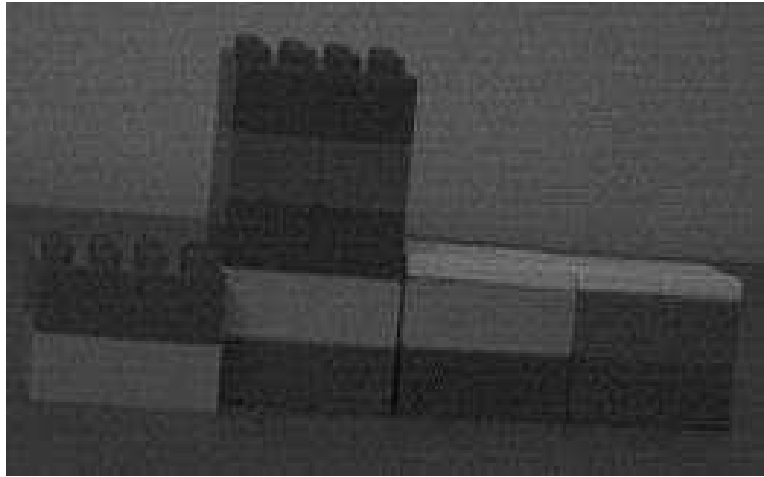
If  $sat\_ratio \approx value\_ratio$ , then this means that there is a good balance of intensity and color in the image, making it ideal for segmentation and detection.

A number of images in different light conditions were analyzed, and it was noted that when  $sat\_ratio > 0$  (low light condition), then the chance of  $value\_ratio > 0$  is also quite high.

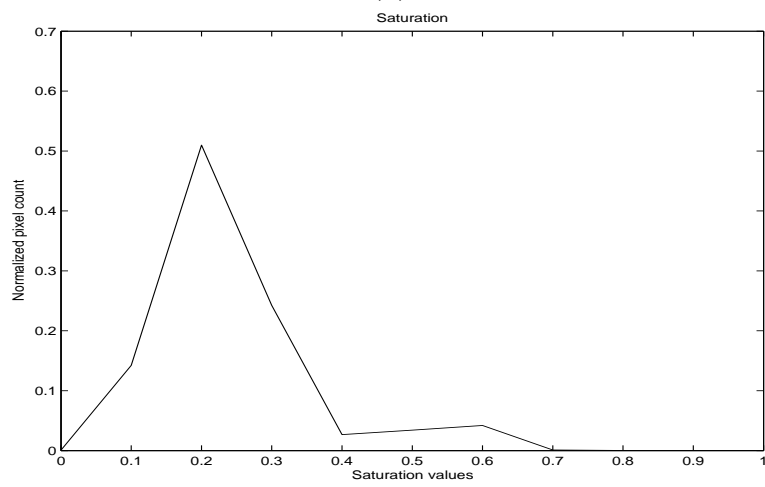
Table 1 gives the intensity and saturation ratios for the images in the figures 6 to 10. Thus, these simple computations give a good judgement on lighting conditions.

## 2.5 Direction and direction changes

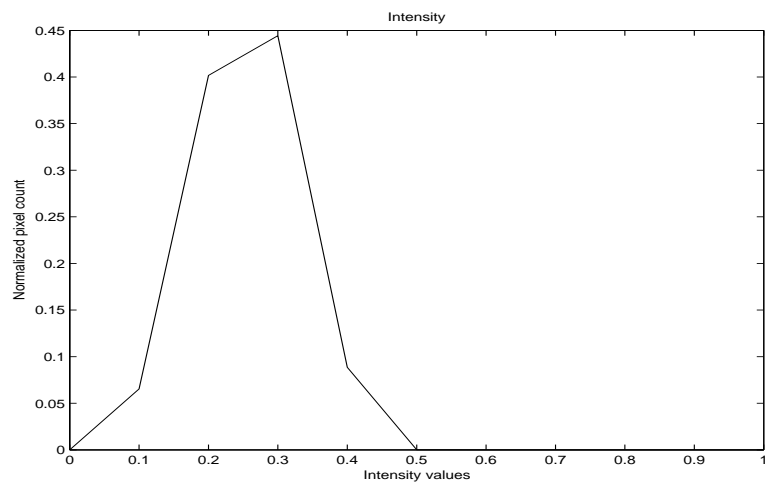
The motion estimation (ME) process gives motion vectors, namely the displacement in the *X* and *Y* directions. Using these vectors, the direction of motion can be



(a)

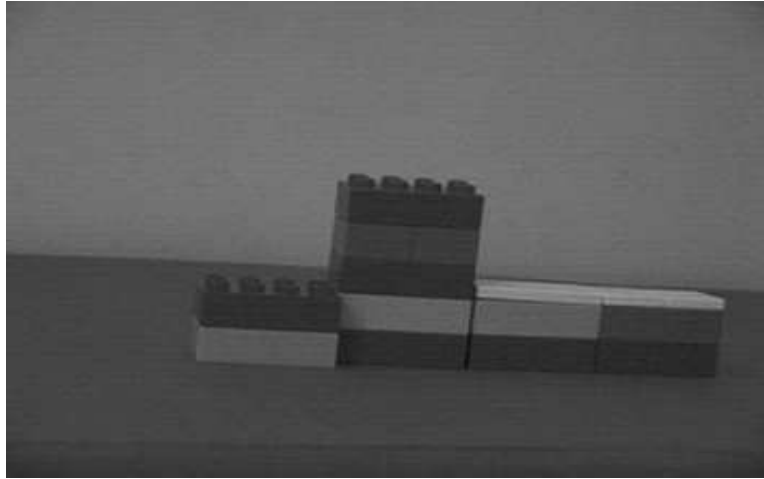


(b)

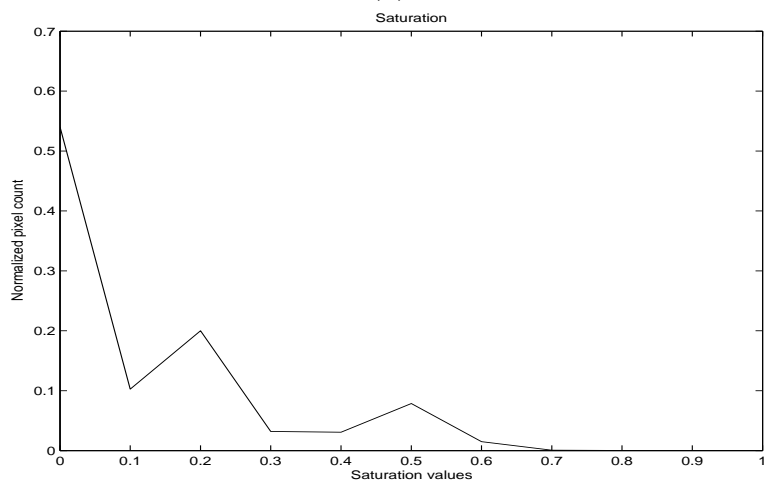


(c)

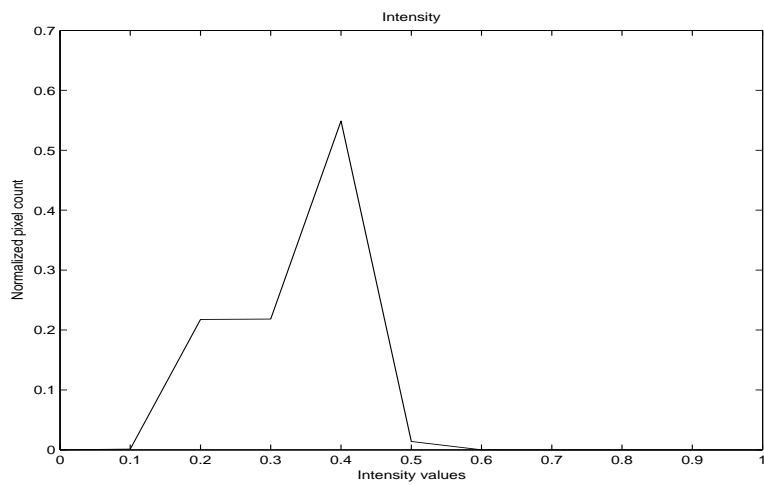
Figure 6. a) Image of blocks in dark b) Saturation values c) Intensity values



(a)

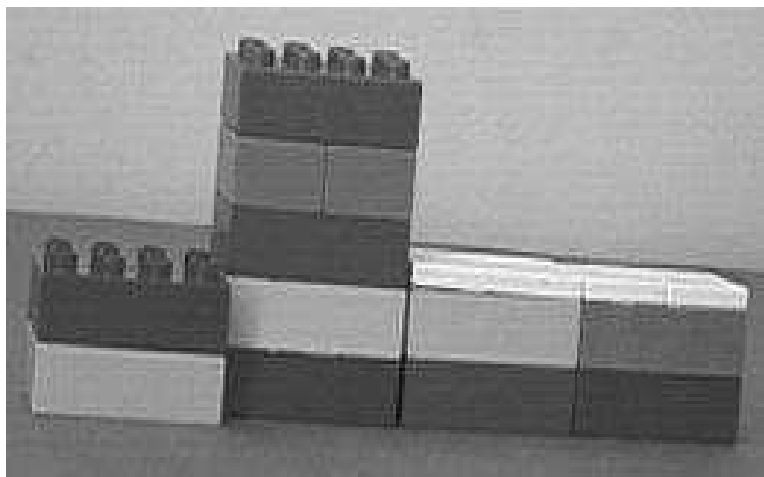


(b)

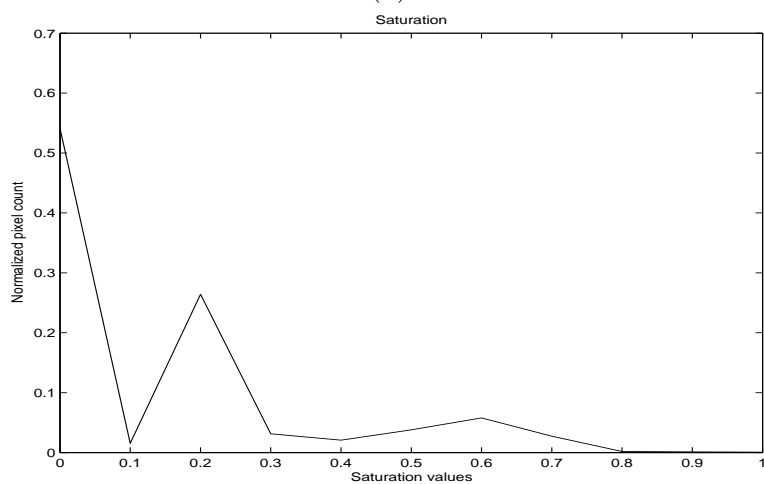


(c)

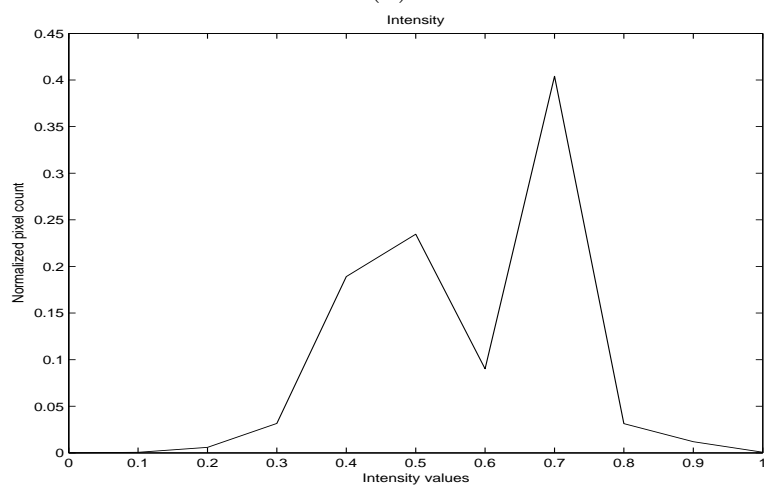
Figure 7. a) Image of blocks in low light b) Saturation values c) Intensity values



(a)



(b)

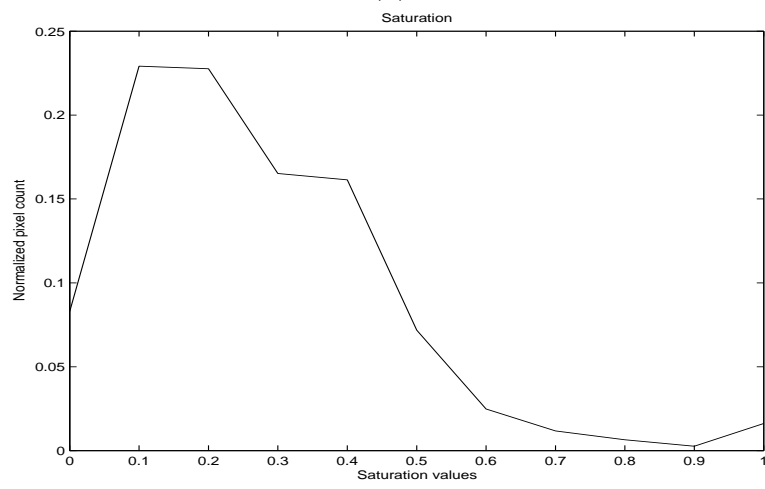


(c)

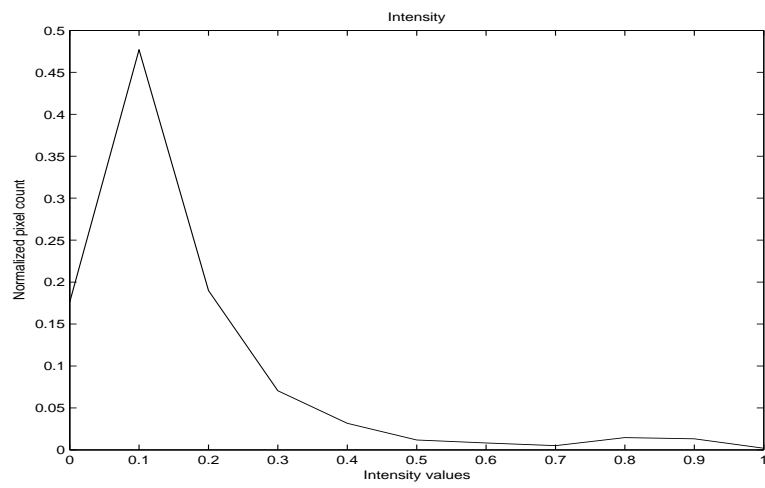
Figure 8. a) Image of blocks in bright light b) Saturation values c) Intensity values



(a)



(b)

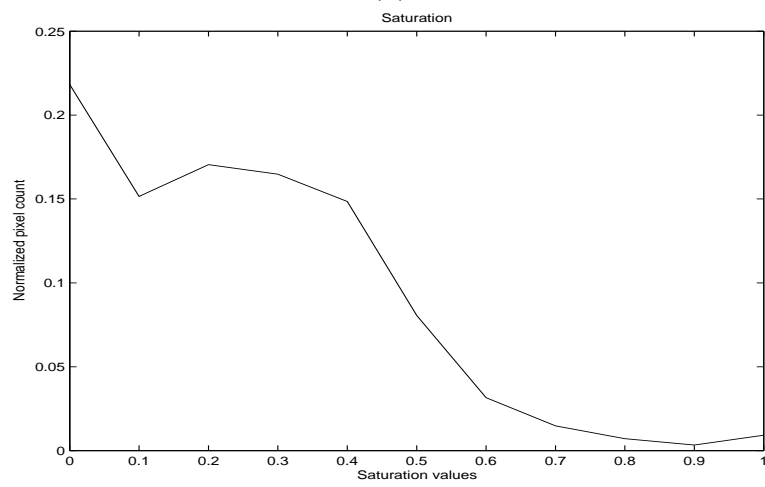


(c)

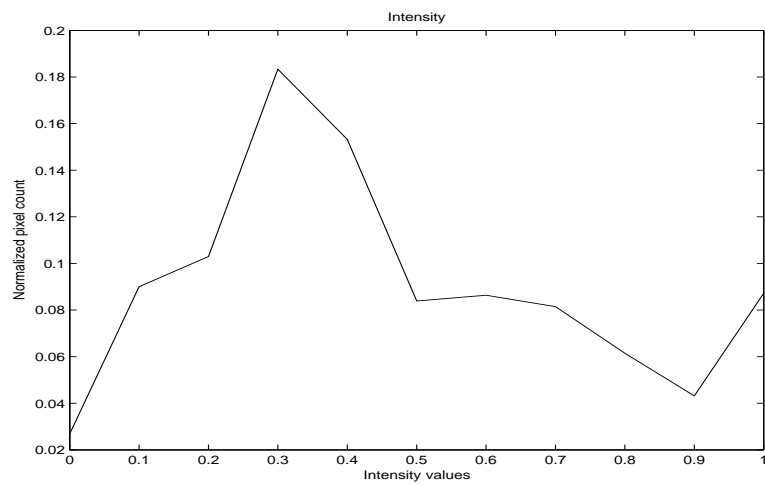
Figure 9. a) Image of house in low light b) Saturation values c) Intensity values



(a)



(b)



(c)

Figure 10. a) Image of house in bright light b) Saturation values c) Intensity values

easily determined based on the sign value. Similarly, any change in motion can be determined by changes in the vector magnitudes or signs.

## 2.6 Color Segmentation in the HSI domain

Utilization of the HSI domain for color segmentation has been fairly popular. Hue gives color information. One of the earliest works obtained hue differences and used them to develop an edge detector jointly using HSI values was by Carron and Lambert [26]. A sigmoid function was used on all pixels of the saturation image to quantize the pixel values, and use this mask to decide on hue relevance as shown in Equation 11.

$$\alpha(s) = \frac{1}{1 + e^{-slope(S-offset)}} \quad (11)$$

Where *slope* is the slope and *offset* is the midpoint at the transition of the sigmoid function. If both the adjacent pixels are highly saturated, Equation 12 is used to obtain the distance measure. This is then multiplied with the hue difference to obtain a hue gradient as given in Equation 13. The gradient is used with different combinations of the saturation and intensity gradients to obtain color gradient operators.

$$\rho(s_1, s_2) = \sqrt{\alpha(s_1) \cdot \alpha(s_2)} \quad (12)$$

$$\Delta H(H_1, H_2) = \rho(s_1, s_2) \cdot \delta H(H_1, H_2) \quad (13)$$

Zhang and Wang used the K-means clustering on hue and intensity for segmentation [27]. Here both the hue and intensity are clustered using Euclidean distance measures and a fuzzy membership function is used to further cluster and segment the image.

## 2.7 Vector Angle Computation on RGB Data

Dony and Wesolkowski utilized the vector angle between the RGB components instead of using the Euclidean measure for edge detection [28]. As the RGB to HSI conversion involves use of cosines, it is computationally expensive. Using the vector angle of the RGB data avoids this conversion. Moreover, the vector angle depicts the hue and saturation information well, though it is insensitive to intensity changes [4]. The vector angle computation on the RGB data is as given in Equation 14.

$$\sin\theta = \sqrt{(1 - \cos^2\theta)} = \sqrt{1 - \left(\frac{\vec{v}_{rgb1}^T \cdot \vec{v}_{rgb2}}{\|\vec{v}_{rgb1}\| \|\vec{v}_{rgb2}\|}\right)^2} \quad (14)$$

where subscripts *rgb1* and *rgb2* indicate RGB values of adjacent pixels. Sine of the angle is used instead of the cosine due to the convention of strong edges (with angular values close to  $90^\circ$ ) being represented by 1 and non-edges (with smaller angles) by 0. Thus Equation 13 gets modified to Equation 15.

$$\Delta H(H_1, H_2) = \rho(s_1, s_2) \cdot \sin\theta \quad (15)$$

Using the vector angle results in intensity invariant segmentation where hue segmentation is better than that obtained by segmenting in the HSI domain. Wesolkowski and Jernigan used both vector angle and Euclidean distance measures as described in [4]. Here, the vector angle was used with large values of  $\rho(s_1, s_2)$  and the Euclidean measure was used with small values of  $\rho(s_1, s_2)$ . This method utilizes pixel saturation strengths to compute the vector angle distance measure when both pixels have high saturation, and Euclidean distance measure otherwise. This method will be referred to as the VA-Euclidean method in this document. It was concluded in [4] that the vector angle approach is best suited for applications like robot vision. The VA-Euclidean method fails to segment properly images with low hue spread. For such images the hue takes prominence and most of the edges are detected due by

the Euclidean difference. This results in noisy segmentation as seen in Figures 12c and 13c.

## **2.8 Research Contributions: The Vector Angle With Absolute Difference (VA-D)**

At a high level, the objective of the proposed research is as follows: 1) Segmentation of objects during surveillance 2) Object identification based on some descriptor.

Further enumeration of the above objectives gives us: 1) Segmentation based on motion and tracking motion 2) Object descriptor based on shape and color for matching between frames. In addition to the above, we need a method to detect scene changes.

Most similarity based (grouping region by growing) and discontinuity based (Canny, Laplacian and Difference Operator) detectors detect all edges indiscriminately. For this application, however, a detector is needed that detects what the user of the system considers important. This detector should capture straight line edges with relatively saturated colors and in case of low contrast, enhance the contrast before detection. This system should also segment out objects that satisfy the above criteria (namely have straight line edges and color) but do not belong to a certain shape. Thus, a combination of similarity and discontinuity based detection is required. First, pixels have to be defined as belonging to either containers or background based on some definition of similarity and then the discontinuities in these segments have to be used for separating between objects of interest.

Chen et al. argue that intensity edges indicate three types of factors: 1) Edge height 2) Edge slope and 3) Noise and that among these three factors, edge height is the closest to human perception of edges [29]. Considering this, quantized saturation values are used to segment the hue. Pixels with high saturation values (which give good color information) are considered similar to each other and grouped together.

The vector angle approach, (Equations 14 and 15) is adopted for further discrimination between the hues. In the next step, pixels with high intensity values are grouped together. The absolute difference between the intensity values is used for discriminating between intensity values. Preliminary results using joint saturation and intensity values resulted in good segmentation while leaving the background out.

### 2.8.1 Difference Vector Edge Detector

The Difference Vector Edge Detector (Equation 16) also used in the VA-Euclidean method, is adapted to the vector angle and the absolute difference measures. This edge detector obtains the maximum difference between pixels across the center pixel in the diagonal, the horizontal and the vertical directions.

$$d = \max(\vec{p}_{i-1,j} - \vec{p}_{i+1,j}, \vec{p}_{i,j-1} - \vec{p}_{i,j+1}, \vec{p}_{i-1,j-1} - \vec{p}_{i+1,j+1}, \vec{p}_{i-1,j+1} - \vec{p}_{i+1,j-1}) \quad (16)$$

where,  $p_{i,j}$  is the pixel in the  $i^{th}$  row and  $j^{th}$  column and is the center pixel.  $p_{(i-1,j)}, p_{(i+1,j)}, p_{(i,j-1)}, p_{(i,j+1)}$  are the pixels in the horizontal and vertical directions and  $p_{(i-1,j-1)}, p_{(i+1,j+1)}, p_{(i-1,j+1)}$  and  $p_{(i+1,j-1)}$  are pixels across the diagonal from the center pixel.

### 2.8.2 Saturation and Value based Combination

As hue is considered more important than intensity for segmenting man-made objects in an image, only highly saturated pixels are considered ( $\alpha(s) > 0.5$ ) initially. Thus vector angle is not computed for all pixels. The sigmoid function (Equation 11) is applied to the saturation image in advance and for function values larger than a threshold, the vector angle is computed. The segmented (vector angle) result is converted to binary by applying a threshold for corner detection. Result from using this approach is shown in Section 2.8.3, Figure 11d.

However, there are images with narrow hue spread and not too many saturated pixels. For pixels with low saturation, Euclidean distance measure has been used

**Table 2. Decision (D) for test images**

Image	Hue Vari- ance	Hue Standard Deviation	Percent Pixels Saturated	Decision Measure D
Still1	0.0275	0.1657	0.26	0.043
Container2	0.0097	0.0983	0.03	<b>0.003</b>
Container3	0.0008	0.0291	0.32	<b>0.009</b>
Container4	0.0181	0.1341	0.17	<b>0.022</b>

in the VA-Euclidean method given in [4]. However, the Euclidean distance measure tends to be overly sensitive to noise. Hence, a modification to the vector angle method is proposed.

A decision measure  $D$  is developed using the hue variance  $H_{var}$  and the percent of pixels  $P_s$  that are highly saturated as shown in Equation 17. Table 2 gives the  $D$  value for some test images. A pixel is considered to be highly saturated if it is greater than 0.5.

$$D = H_{var} \cdot P_s \quad (17)$$

The values in bold indicate the images for which the saturation is low and hue is not well spread. For such images, the sigmoid function is applied to the value image also to obtain  $\alpha(v)$ . These sigmoid functions for saturation and value are checked for all pixels. If saturation is low and value is very high, the Difference Vector Edge Detector is applied using absolute differences between  $\alpha(v)$ , else the pixel is set to zero.

Thus, for all pixels,

- IF  $D > Threshold$ ,
  - IF  $\rho(s1, s2) > \kappa_s$  apply the Vector Angle version of the Difference Vector Edge Detector.
- ELSE IF  $D < Threshold$ ,

- IF  $\rho(s1, s2) > \kappa_s$  apply the Vector Angle version of the Difference Vector Edge Detector.
- IF  $\rho(v1, v2) > \kappa_v$  apply the Difference Vector Edge Detector using absolute differences between  $\alpha(v)$  for respective pixels.
- Set all remaining pixels to zero.

More formally, for the case of ( $D < Threshold$ ) the segmentation can be a function  $f_{VAD}$  of the form  $f_{VAD}(\rho(s), \rho(v))$  where  $f_{VAD}$  must fulfill the following conditions:

1. If  $\rho(s1, s2) > \kappa_s$ , then Vector Angle should contribute to the function.
2. If  $\rho(v1, v2) > \kappa_v$ , then only the absolute distance between  $\alpha(v1)$  and  $\alpha(v2)$  should contribute towards the function.

The following function (Equation 18) is modeled to achieve the above conditions.

$$f_{VAD} = \begin{cases} e^{(\rho(s1, s2) - \kappa_s)} \cdot \rho(s1, s2) \cdot \sin(\theta) & \text{if } D > \tau \\ e^{(\rho(s1, s2) - \kappa_s)} \cdot \rho(s1, s2) \cdot \sin(\theta) + e^{\epsilon(\rho(v1, v2) - \kappa_v)} \cdot |\alpha(v1), \alpha(v2)| & \text{if } D < \tau \end{cases} \quad (18)$$

Where  $\tau$  is a threshold set to 0.3 empirically based on different test images (see Table 2).  $\kappa_s$  is set to 0.5 in order to include pixels with good saturation and  $\kappa_v$  is set to 0.8 to including pixels with good intensity.  $\epsilon$  is empirically set to 4, to keep to a minimum the contribution of the absolute difference measure when saturation is good.

This method has the advantage of giving priority to pixels with good hues and including edges that might be present in areas of high luminance. Edges that do not meet any of these criteria are ignored. Some results based on this algorithm are given in Figures 12d and 13d in Section 2.8.3.

### 2.8.3 Results of VA-D Approach

It can be concluded that along with pixels with good hues, there is also a need for considering pixels with good intensity. However, when the Euclidean distance is applied to *all* pixels that have low saturation, a noisy segmentation can result, especially for images with low contrast. Instead, if a simple absolute difference on the intensity values is obtained only for pixels that have very low saturation a less noisy segmentation is achieved. This also has the advantage that image parts with good color content are still given a higher priority during segmentation.

Figures 11b, 11c, and 11d, show the results of Canny edge detection, the VA-Euclidean and the results of the proposed VA-D method. In Figure 11a, the objects of interest are the three containers. The Canny detector detects almost all edges, but additional processing would be needed to eliminate unnecessary objects. For example, in Figure 13b, the antenna shows up in the Canny result, but is segmented out in the VA-D result shown in Figure 13d. The VA-Euclidean method gives some additional edges and really works well only for images with good color details. This is further demonstrated in Figure 12c and 13c where the results of the VA-Euclidean method are not very useful.

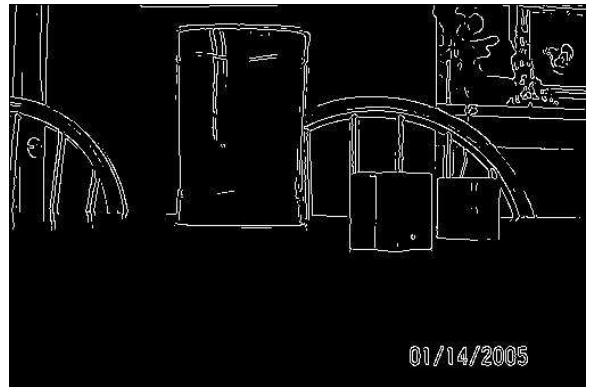
### 2.8.4 Detecting and Connecting Straight Line Edges

As the target scenario for the proposed video data analysis primarily pertains to man-made structures, straight line edges are given importance. Some of these edges are broken in the previous segmentation process. Additional processing is needed to reconnect such edge components. A brief description of connected components and their labeling is given in Appendix B.

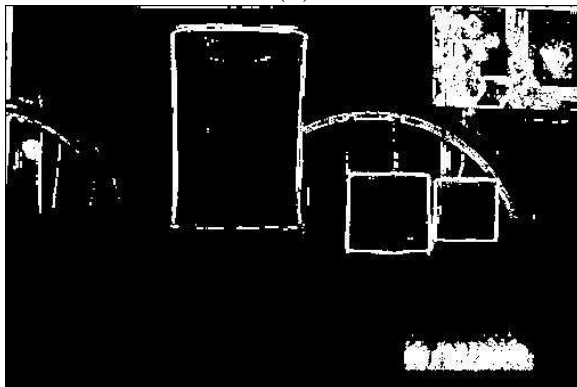
One particular approach of line segment detection in edge images was by grouping together elementary line segments (ELS). These were obtained by linking edge pixels and then approximating them to line segments. In [30], the ELSs were further grouped together to obtain *base lines*, and ELSs near base lines were grouped



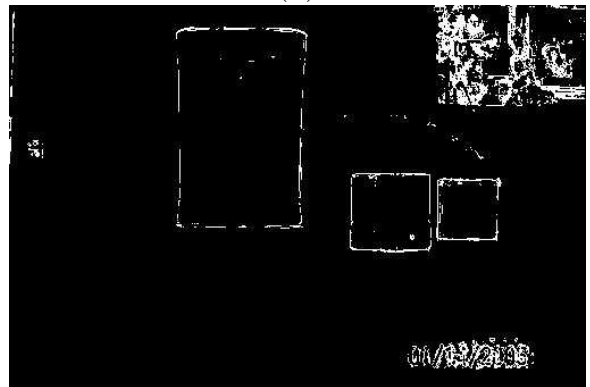
(a)



(b)



(c)

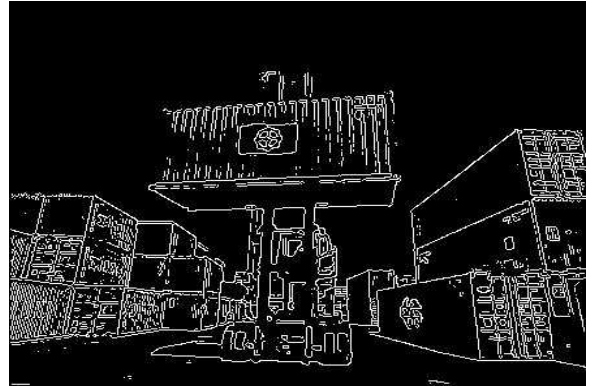


(d)

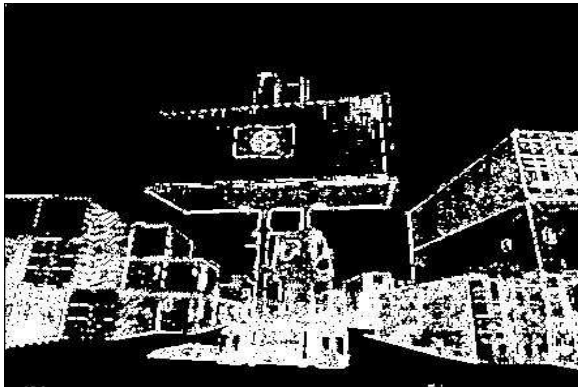
Figure 11. a)'Still1' image b)Figure showing Canny edge detector result c)Figure showing the result of running the VA-Euclidean method described in [4] d)Figure showing result from running the proposed VA-D method.



(a)



(b)



(c)

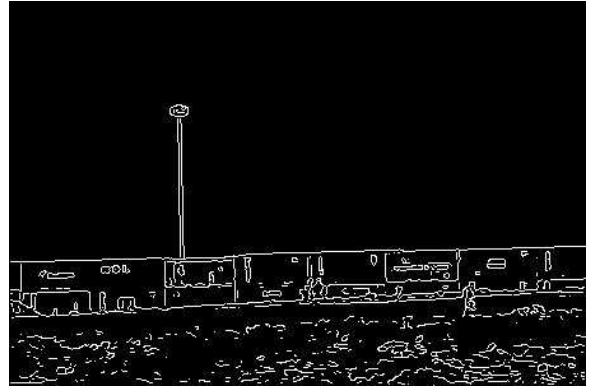


(d)

Figure 12. a)'Container4' image [5] b)Figure showing Canny edge detector result c)Figure showing the result of running the VA-Euclidean method described in [4] d)Figure showing result from running the proposed VA-D method.



(a)



(b)



(c)



(d)

Figure 13. a)'Container7' image [6] b)Figure showing Canny edge detector result  
c)Figure showing the result of running the VA-Euclidean method described in [4]  
d)Figure showing result from running the proposed VA-D method.

again. Finally, the redundant and un-grouped line segments were filtered out by using some criteria [30]. Here, the orientation difference and the lateral distance between ELSs were used for grouping. If two or more line segments shared an ELS, then the longer line segment was favored over the rest. This helped in removing redundant line segments by eliminating segments whose ELSs were shared by more dominant segments.

Another method used the small eigenvalue analysis for line detection [31]. The method scanned the input edge image from top-left corner to bottom-right corner with a moving mask of  $k \times k$  size. The small eigenvalue of the covariance matrix was computed provided the number of edge pixels covered by the mask was more than one, otherwise, the small eigenvalue was set to a high value to eliminate that particular edge from being considered. If the computed small eigenvalue was less than a threshold  $t$ , then all the pixels in the edge image were set to high. This method reported robustness to image transformations like scaling and rotation.

In [32], connected regions are found and depending on the shape and orientation of the connected regions a voting kernel for Hough transform was generated. In many applications, prevalent orientations in the image are required. For such applications, the contributions of the quantized rectangular cells are added corresponding to certain values of  $\theta$ . This gives the prevalent orientation.

### 2.8.5 Hough Transform and its Variants

Hough proposed a method for detecting straight lines in images using the slope-intercept form for lines given in Equation 19.

$$y = slope \cdot x + intercept \tag{19}$$

Rewriting Equation 19 in the form shown in Equation 20, we can say that for a range of slope values, intercept value can be obtained for all pixels in  $[x, y]$ .

$$intercept = y - slope \cdot x \quad (20)$$

To reduce the computational burden, Hough proposed creating accumulator cells by quantizing the slope and intercept values and group the pixels based on these values. The cells which have a maximum number of pixels with the slope and intercept values were given higher priority and indicate collinearity.

However, the slope-intercept form of straight line (Equation 20) suffers from unbounded slope and intercept values. An alternate parametrization for overcoming this problem by using the normal form of straight line equation (Equation 21) was proposed in [33].

$$x \cdot \cos\theta + y \cdot \sin\theta = r \quad (21)$$

Here, the perpendicular distance from the origin  $r$  is computed for angle  $\theta$  values ranging from  $+90$  to  $-90$ . All collinear pixels will have the same  $r$  and values or can be approximated to be within specific ranges of  $[r, \theta]$  [34].

Some of the advantages of the Hough transform are as follows [33]:

1. Detects lines irrespective of missing parts.
2. Insensitivity to noise and non-straight line structures.
3. Detects partially occluded parts of objects.

Some of its disadvantages are as follows [33]:

1. Computationally time consuming.
2. Detects collinear lines irrespective of contiguity.
3. Detects meaningless lines due to quantization of the parameter space.

Regardless of these disadvantages, the Hough transform is a very popular method for line and edge detection. Adaptive, combinational and hierarchical methods have been analyzed and are all variants of the Hough transform, developed for reducing computation time. These reduce the computational complexity of detecting local peaks, but have to be applied to the entire image. To reduce this computational burden, probabilistic methods that make use of polling mechanisms to randomly select few edges have also been put forth [31].

### **2.8.6 Research Contributions:**

#### **Line Connecting Algorithm using the VA-D and Canny Results (LC)**

Most line detection techniques using Hough try to detect correct edges, while ignoring edges due to noise. In [35], connectivity was described in a very strict sense and any gap in a line segment was given due consideration and two different line segments were assumed to exist. In [36], computational burden was further reduced by using the connectivity property of the line. Here, once a end pixel of a line is picked, the following pixels are picked in an increasing (or decreasing) order of the coordinates. Once the row is broken, search for further pixels of that line is abandoned. The pixels belonging to one identified line are also excluded from future line searches.

Obtaining the threshold of the vector angle image results in an image with some broken edges. Hough transform is an effective way to obtain connected components. However, it is computationally expensive and may end up connecting pixels that are not actually part of an edge. The variants of the Hough transform described above, namely [35] and [36], will give inaccurate results as they consider broken segments distinct from each other and not belonging to the same edge. This would give inaccurate results on our analysis.

Another method is based on the likelihood principle of connectivity and thickness for line detection [37]. Short and thick noise segments are sometimes ignored in noisy

images. This method aims at using the likelihood to obtain a weight which is then added to the quantized cells after obtaining the Hough transformed result.

Here, we propose a method using both the Canny and the results of the vector angle (VA-D) method along with the Hough transform to reconnect such broken lines without adding back un-necessary edges.

The proposed algorithm is as follows:

- Create a new empty image R.
- Obtain the binary vector angle image (call it VA).
- Apply the Canny detector on the saturation (S) image (Call it L).
- If  $(VA(i, j) > 0)$  Obtain  $(r, \theta)$  using Hough transform. Call this data  $VA_{r,\theta}$ .
- If  $(L(i, j) > 0)$ , Obtain  $(r, \theta)$  using Hough transform. Call this data  $L_{r,\theta}$ .
- $\forall \theta$  values in  $VA_{r,\theta}$ , if  $(\theta_L == \theta_{VA})$  AND  $(r_L \leq r_{VA} + range)$  AND  $(r_L > r_{VA} - range)$  if set to 1 the respective pixels from L which have  $(r_L, \theta_L)$  values in the final result R.

All lines in the Canny result which are within a *range* and have the same slope  $\theta$  as the vector angle result are considered. Using the vector angle image in combination with the Canny detected image helps in running the transform only for relevant pixels thus reducing computation and connecting only the broken line segments which have an association with an edge.

#### *2.8.6.1 Results of the Proposed Line Connecting Algorithm*

The results from the VA-D system in conjunction with the LC algorithm can be further used for container detection and recognition. Figure 14 shows the broken segments after the segmentation process and the connected lines using the LC algorithm. In Figure 15, the broken segments are reconnected while the edge due to variations

in the intensities (seen in the canny image on the bigger container) and the chairs in the background are left out.



Figure 14. a) Broken segments after VA computation b) Segments reconnected after applying LC algorithm to connect lines.

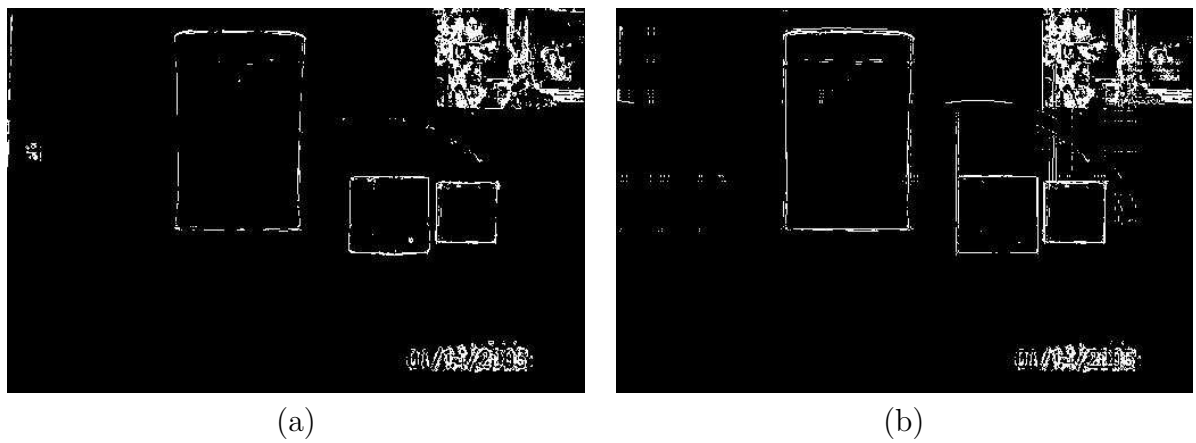


Figure 15. a) Broken segments after VA computation b) Segments reconnected after applying LC algorithm to connect lines.

## CHAPTER 3

### COLOR DESCRIPTOR

During object detection, once segmentation is carried out, the object has to be matched to pre-existing descriptors in an effort to detect and track the object. The onus on such a system is to do this efficiently in real time while keeping computations and size of the descriptors as small as possible.

This section discusses various object descriptors and describes a new method for object detection based on HSV color space and vector angle computation discussed in the previous chapter.

#### **3.1 Introduction**

Matching images in an existing database based on some low level semantics is called ‘Image Retrieval’. Shapes of objects in images, texture and color information are examples of some of the features that are used for retrieval [38]. ‘Video indexing’ is a term used to index key frames in videos to enable search and retrieval. Temporal segmentation is used to identify key frames in videos, which are then used during retrieval [23].

Color also gives spatial information irrespective of view angle and is independent of image resolution to a large extent. Color can have its disadvantages though, some of which are the large memory requirements to store color information and additional processing overhead for conversion from one color scheme to another. These disadvantages can be additional burdens on a real-time system. Earlier retrieval methods depended on human supplied textual descriptors for indexing. However, these searches depend on user supplied data, and this can be hard to create for large databases. Moreover, searches based on textual descriptors can have their disadvantages if there is a slight mismatch in the language used for the search and the descriptor itself [38].

One of the simplest implementation of color histograms is by quantizing the RGB or HSV values to generate the histograms for the images being matched. The joint probabilities of the intensities for the three color channels form the histogram [39]. A simple intersection measure is then used to obtain retrieval rates. This has shown to be an efficient method of retrieval [40], [41]. However, such histogram produce false positive due to loss of spatial information in the histograms. To counter this various methods utilizing spatial information have been addressed [38], [42], [43], [44], [45], [46].

In the work proposed by Smith and Chang [38], automatic extraction of color and texture information is carried out by using binary set information of color and texture to index images in a database. Here, color sets are composed of quantized color spaces; region labeling is carried out by median filtering on quantized HSV data which gives the color feature. Similarly, texture feature is extracted by applying sub-band filtering on the images and getting a binary representation of the energy distributions which is then used to reconstruct an image with labeled textures.

Ooi et. al. [42], proposed a method where single color clusters are created over homogenous color regions of an image and image comparison is done by matching clusters in image space. The main challenge in this setup is to determine the number of clusters needed for optimal retrieval. A color coherence vector (CCV) is defined in [43], where every pixel is classified as coherent or not depending on whether or not it is part of a larger similar colored region. This method showed improvements over color histograms in both execution time and effectiveness. Color correlograms are defined in [44], where the  $k^{th}$  entry of a row  $\langle i, j \rangle$  gives the probability of finding a pixel of color  $j$  at a distance of  $k$  pixels from another pixel of color  $i$ . Though this method was found robust to significant changes in appearance of images, it is computationally intensive for practical purposes [47]. Another method classified pixels as border or interior based on color difference between a center pixel and its 4 neighbors

**Table 3. Computations per pixel for RGB to HSI conversion**

Computations	Mul/Div	Add/Sub	Compares
<b>Hue</b>	2	3	6
<b>Saturation</b>	1	1	2
<b>Intensity</b>	1	1	0
<b>Total</b>	4	5	8

**Table 4. Computations per pixel for RGB to HSV conversion**

Computations	Mul/Div	Add/Sub	Compares
<b>Hue</b>	2	3	6
<b>Saturation</b>	1	0	1
<b>Value</b>	0	0	0
<b>Total</b>	3	3	7

in a 3 x 3 neighborhood. Once the classification is done, separate histograms are generated for the pixels classified as border and those classified as interior pixels [45]. A log based distance measure is then used to compare the histograms of two different images. A later work, by Lee et. al. [46] used the vector angle distance measure between center pixels and adjacent pixels to classify pixels as smooth and edge pixels for their descriptor. Two dimensional histograms are created based on this information where quantized Hue (H), Saturation (S) and Intensity (I) values are used to construct the bins and update the histogram. The retrieval results from their method were compared with those from color histogram based methods as well as color correlogram method and the hybrid graph representation. This method reported better recall, precision and better average normalized modified retrieval ranks than the other methods. The method was also found robust to spatial changes including views from different angles and camera zooms.

In section 2.1 of Chapter 2, conversion equations from RGB to HSV and HSI were discussed. Table 3 and table 4 give the number of computations involved in these conversions. It can be seen that these computations can quickly add up.

The color descriptor described in [46], utilizes a 2-D histogram where the x-axis represented the color of the center pixel  $p_c$ , and the y-axis represented the color of the neighboring pixel making the largest vector angle  $p_n$ . In this method, the

relevancy of a bin is considered and relevant bins are set to 1 and irrelevant bins to 0 to create a binary matrix. As this still does not reduce the size of the matrix, each row of the matrix is further converted to an equivalent decimal number to reduce the dimensionality of the histogram to 1-D [46]. The Hue, Saturation and Intensity values are quantized to result in 96 bins in the X and Y directions for the 2-D histogram. Although, the final histogram is one dimensional, there is a large intermediate memory footprint ( $96 \times 96 = 9216$  bins) initially, as well as additional processing overhead to make a decision on relevancy and binary to decimal conversion in order to accomplish the dimensionality reduction. Moreover, this method used the HSI values to construct the histograms and for every  $p_c$  and  $p_n$  of the image, the respective H, S and I values would have to be computed to update the histogram. These are potential disadvantages in real-time analysis when large numbers of images are involved. This method referred to as a spatial color descriptor will be referred to as HSI-SCD method in this document.

### **3.2 Research Contributions: RGB Vector Angle based Color Histogram Descriptor**

The proposed analysis operates on a  $3 \times 3$  neighborhood. The vector angle between a center pixel and its eight neighbors is computed and the color of the pixel  $p_n$ , making the largest vector angle with the center pixel  $p_c$ , is noted. If the largest vector angle is less than a threshold  $\tau(0.045)$ , then the center pixel is classified as smooth and the histogram for this pixel is referred to as a smooth color histogram. For pixels where the largest vector angle is greater than the threshold, the center pixel is considered to lie along an edge; and the histogram is referred to as the color adjacency histogram. Pseudo code for this method is given in the pseudo code below. Thus, a color adjacency descriptor and a smooth color descriptor are generated that consists of two 1-D histograms each. The histograms have 125 bins each representing

5 levels of quantized R, G and B values. Quantization levels lesser than five were seen to be less effective in representing the images accurately.

- Initialize  $C_{edge}$ , the count for pixels classified as belonging to an edge; and  $C_{smooth}$ , the count for pixels classified as smooth, to 0.
- Obtain the vector angle between a center pixel and 8 nearest neighbors for query image.
- For the center and adjacent pixel pair having the largest vector angle (VA), obtain the quantized RGB values of pixel pair.
- If  $VA > \tau$ , then classify the pixel pair as edge pixels and update 1-D color adjacency histogram for the center pixel. Increment  $C_{edge}$  by 1.
- If  $VA \leq \tau$ , then classify the pixel pair as smooth color pixels and update 1-D smooth color histogram for the center pixel. Increment  $C_{smooth}$  by 1.
- Normalize the histograms based on the  $C_{edge}$  and  $C_{smooth}$  counts.
- Repeat steps 1-6 for test image to be compared to query image.
- Compute similarity measure between histograms for query and test image. Larger the similarity value higher the rank for the matched image and better the match.

In the proposed method, only the RGB values are used in constructing the histograms, so the above issues are not of concern. However, there is a slight increase in the number of histogram bins. To avoid dealing with 2-D arrays, we create separate 1-D color histograms for the center and the adjacent pixels of 125 bins each. In the HSI-SCD method the total number of histogram bins after converting to decimal representation was 96, in the proposed method the bin count is 250. Separate counts for

pixels that classify as smooth and as edges are maintained and are used to normalize both the smooth and the adjacency histograms in order to make them independent of image size.

### 3.2.1 Similarity Measure

For the smooth color histogram Swain's histogram intersection measure is used. Bins are classified as relevant if a bin has at least 10% of the total pixels classified as smooth. A relevant bin is set to 1, otherwise 0. For two images 'a' and 'b' being compared,  $N_{ab}$  gives a count of bins which have the same bin number and are also relevant.  $N_a$  gives the count of bins that are relevant for image 'a';  $N_b$  gives the count of bins that are relevant for image 'b'. The intersection measure for the smooth histogram is given in Equation 22.

$$S_s = \frac{N_{ab}}{N_a + N_b - N_{ab}} \quad (22)$$

For the adjacency histogram a simple absolute differencing is carried out as shown in Equation 23. In the case of two completely non-intersecting frames, the factor of 2 ensures that the difference  $d_{ab}$  is always less than or equal to 1.0.

$$d_{ab} = \frac{1}{2} \sum abs(h_a[i] - h_b[i]) \quad (23)$$

The similarity measure for the adjacency histogram is given as Equation 24.

$$S_{adj} = 1 - d_{ab} \quad (24)$$

The overall similarity metric is computed as a combination of the measures obtained from Equations 22 and 24 and is given in Equation 25, where  $\alpha$  is 0.5 and  $\beta$  is 0.5.

$$S = \alpha \cdot S_s + \beta \cdot S_{adj} \quad (25)$$

### 3.2.2 Complexity

For testing the system a database of 1500 images was used that consisted of a variety of images ranging from sceneries to buildings and vehicles. It was argued earlier that computing the HSI data can be an overhead in the HSI-SCD method. Figure 16 gives the cycle count over various images of different sizes for both HSI-SCD and the proposed methods. It can be seen that on an average, the HSI-SCD method consumes about 1000 more cycles per image. Even for a very small database of 500 images, this can be approximately 500,000 extra cycles. It should be noted that the proposed method has an advantage over the HSI-SCD method as the image sizes get larger. For smaller images, comparing the larger histogram sizes generated in the proposed method can take up almost the same number of cycles as computing the HSI data.

### 3.2.3 Precision, Recall and Normalized Retrieval Ranks

For every query image the relevant or correct retrieved images are given by  $N_c$  the false alarms by  $N_f$  and the missed relevant images by  $N_m$ . Precision is the ratio of correct images  $N_c$  and the first  $M$  retrieved images and is given by Equation 26. Recall is defined in Equation 27.

$$precision = \frac{N_c}{N_c + N_f} \quad (26)$$

$$where, M = (N_c + N_f)$$

$$recall = \frac{N_c}{N_c + N_m} \quad (27)$$

Figures 17 and 18 give the precision and recall values for various queries. It can be

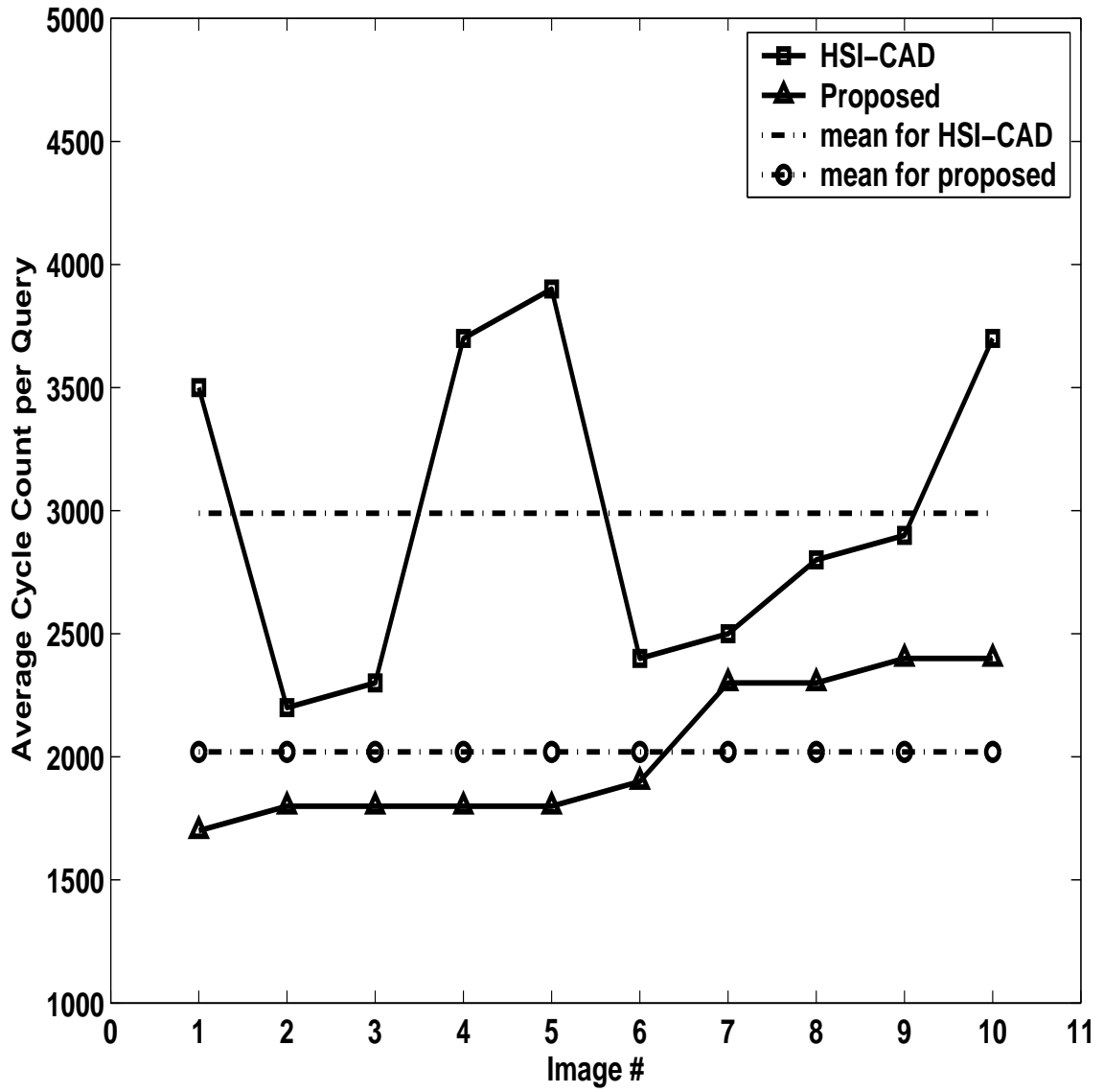


Figure 16. Cycle count comparison between HSI-SCD and proposed method

seen that the proposed method performs similar or better than the HSI-SCD method in all cases.

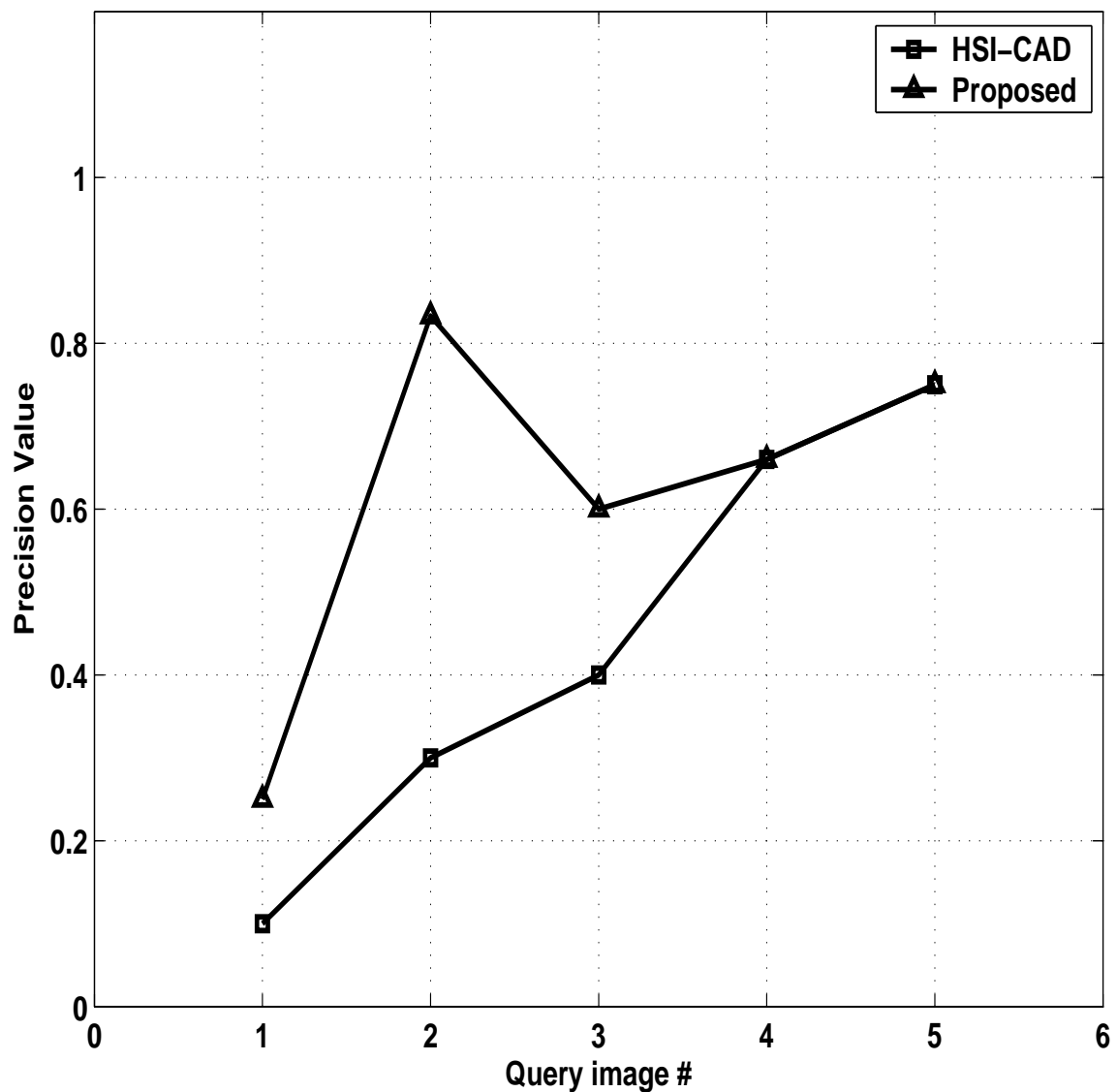


Figure 17. Precision values comparison between HSI-SCD and proposed method

Figures 19, 20, 21 and 22 give some results on retrieval on the entire database of 1500 images for random queries. The results clearly show that the method proposed in 3.2 gives better results than the HSI-SCD method and retrieves images that are closer in content and description to the query image.

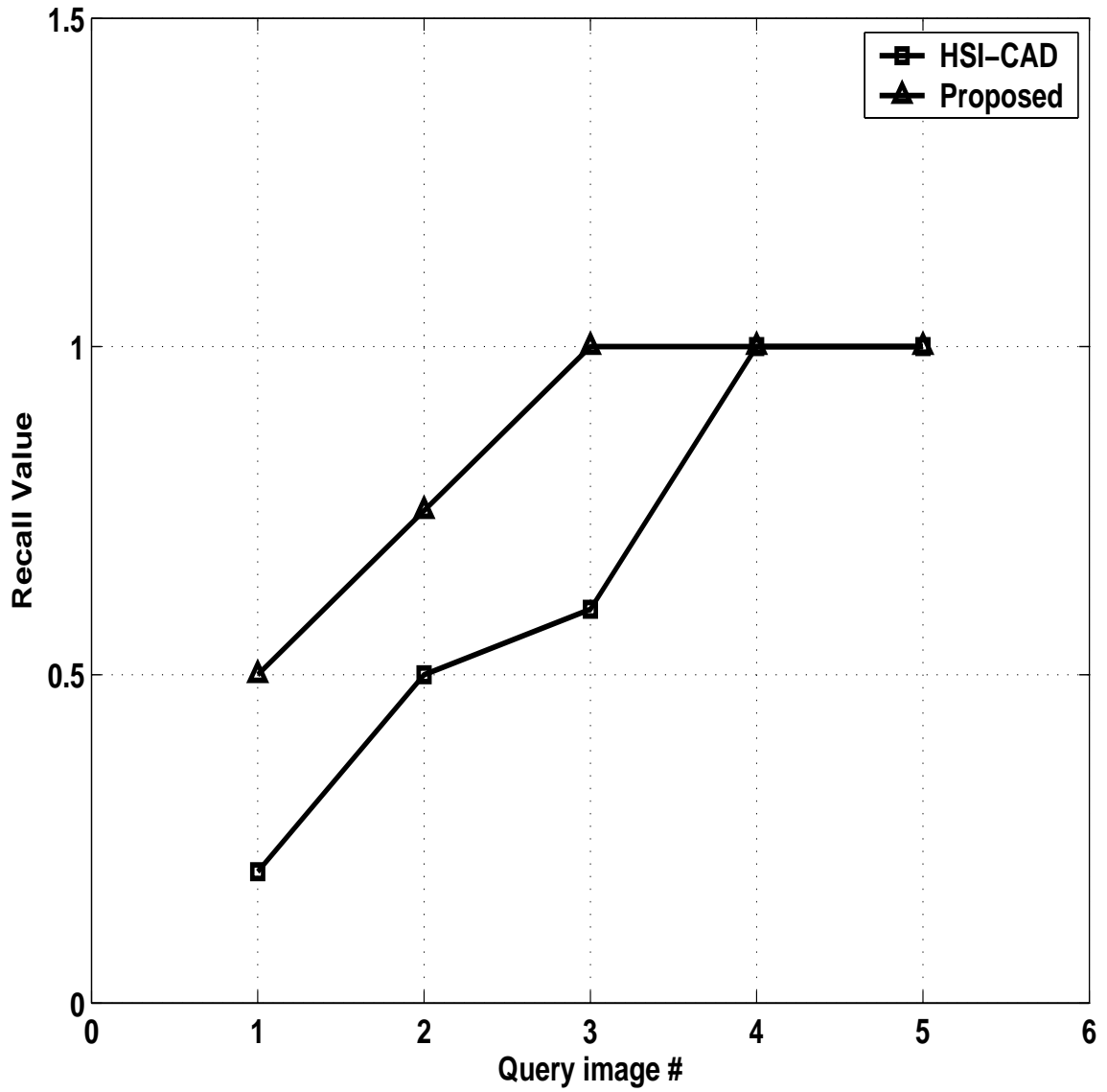
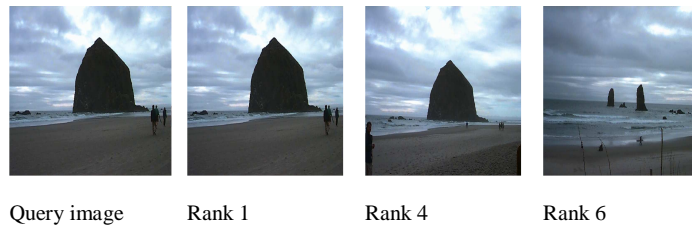


Figure 18. Recall values comparison between HSI-SCD and proposed method



**Retrieval Results for proposed color descriptor**



**Retrieval Results for HSI-SCD color descriptor**

**Figure 19. Retrieval Results for query image 1**



Query image

Rank 1

Rank 2

Rank 4

**Retrieval Results for proposed color descriptor**



Query image

Rank 1

Rank 2

**Retrieval Results for HSI-SCD color descriptor**

**Figure 20. Retrieval Results for query image 2**



Query image

Rank 1

Rank 2

**Retrieval Results for proposed color descriptor**



Query image

Rank 1

Rank 2

Rank 3

**Retrieval Results for HSI-SCD color descriptor**

**Figure 21. Retrieval Results for query image 3**



Query image

Rank 1

Rank 2

Rank 3



Rank 4



Rank 5

**Retrieval Results for proposed color descriptor**



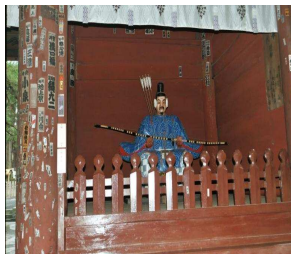
Query image



Rank 1



Rank 2



Rank 3



Rank 4

**Retrieval Results for HSI-SCD color descriptor**

**Figure 22. Retrieval Results for query image 4**

### 3.3 Research Contributions: Modified RGB histogram

One of the drawbacks of the RGB histogram based descriptor described in section 3.2 is that it relies heavily on the vector angle measure for histogram creation. But in low light conditions this becomes a drawback. The second drawback is that for the work discussed in the research, the object matching is between segmented objects from surveillance videos and images in an existing database. The image size, pan, zoom and background and light conditions of the segmented images generally do not match the images in the database exactly. Lastly, although the RGB histogram based descriptor computed solely using the vector angle measure gives good matches, it does not result in very good discrimination between true positives and false positives. In other words, the similarity values between true positives and false positives are very close. In very large databases, or even images with similar color contents, this can lead to many mismatches. These characteristics make the RGB or even the HSI-SCD descriptors not very robust to such object matching. To get around this problem, the histogram method was modified by applying an edge detector to the images and then creating the smooth and edge histograms. The pseudo code for generating this descriptor is given in the following pseudo code.

- Initialize  $C_{edge}$ , the count for pixels classified as an edge pixel, and  $C_{smooth}$ , the count for pixels classified as a smooth pixel, to 0.
- Obtain the vector angle (VA) between a center pixel and 8 nearest neighbors for query image.
- If  $VA > \tau$ , then classify the center pixel as an edge pixel and update the 1-D edge histogram. Increment  $C_{edge}$  by 1.
- Obtain an edge image by using any edge detector. In this case, the Sobel edge detector was used. Call this edge image  $E_i$ .

- If a pixel does not belong to an edge in the image  $E_i$ , then classify the center pixel as smooth and update the 1-D smooth histogram. Increment  $C_{smooth}$  by 1.
- Normalize the histograms based on the  $C_{edge}$  and  $C_{smooth}$  counts.
- Repeat steps 1-6 for test image to be compared to query image.
- Compute similarity measure between histograms for query and test image. Again larger similarity values indicate better matches.

This modification implicitly incorporates the shape and color information of the images and thereby remarkably improving the results. Some segmented query objects and the respective match results are given in the following discussion.

The database of images used for testing the descriptor contained around 60 images with different vehicles taken from different angles and positions. Some of these images are given in Figure 23. The images in the database can be converted to their equivalent histograms making the database even more compact, namely one array of 432 values (216 each for smooth and edge values), assuming a RGB quantization value of 6. As the histograms are normalized, this means 432 bytes per image are enough to represent each image in the database.

Some of the assumptions of such object matching is that the cameras are fixed and always focused on a certain area and that the objects that are observed make regular and frequent appearances. Thus, a database can be built and gradually expanded if newer objects are detected and cannot be matched correctly to the database. If a new object cannot be matched the system raises an alarm following which an action can be taken of whether to consider it a security breach or add the object as a new addition to the database. The advantage of this descriptor is that the a naive user can add images to the database and attributes such as image size, angle from which image is taken or zoom effects are not a factor in making additions to the database.

Some of the results of using this descriptor on matching some segmented vehicles to the above mentioned database are given in Figures 24 and 25.

The precision of the modified RGB descriptor was overall better than that of the original RGB descriptor that used only the vector angle to discriminate between smooth and edge pixels. Figure 26 shows the graph of precision of retrieved results.



Figure 23. A sample of database contents

**MOTION  
SEGMENTED  
OBJECT BEING  
QUERIED**



**MATCH RANK 1**



**0.53**

**Positive Match**

**MATCH RANK 2**



**0.52**

**Positive Match**

**MATCH RANK 3**



**0.49**

**Positive Match**



**0.658**

**Positive Match**



**0.58**

**Positive Match**



**0.566**

**Positive Match**



**No database entry,  
hence false  
positives**



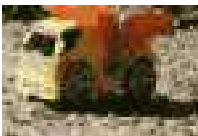
**0.544**



**0.54**



**0.52**



**0.675**

**Positive Match**



**0.671**

**Positive Match**



**0.590**

**Positive Match**

**Figure 24. Results of the modified histogram matching**

**MOTION  
SEGMENTED  
OBJECT BEING  
QUERIED**



No database entry,  
hence false  
positives

**MATCH RANK 1**



**0.52**

**MATCH RANK 2**



**0.51**

**MATCH RANK 3**



**0.498**



**0.91**

**Positive Match**



**0.8601**

**False Positive**



**0.854**

**Positive Match**



**0.739**

**Positive Match**



**0.66**

**Positive Match**



**0.646**

**False Positive**



**0.77**

**Positive Match**



**0.73**

**Positive Match**



**0.66**

**Positive Match**



**0.77**

**Positive Match**



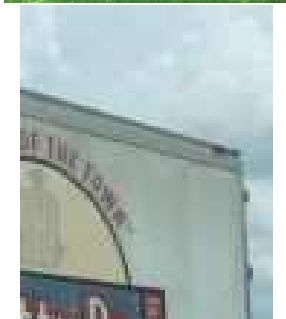
**0.76**

**Positive Match**



**0.76**

**Positive Match**



**0.595**

**Positive Match**



**0.587**

**Positive Match**



**0.50**

**False Positive**

Figure 25. Results of the modified histogram matching

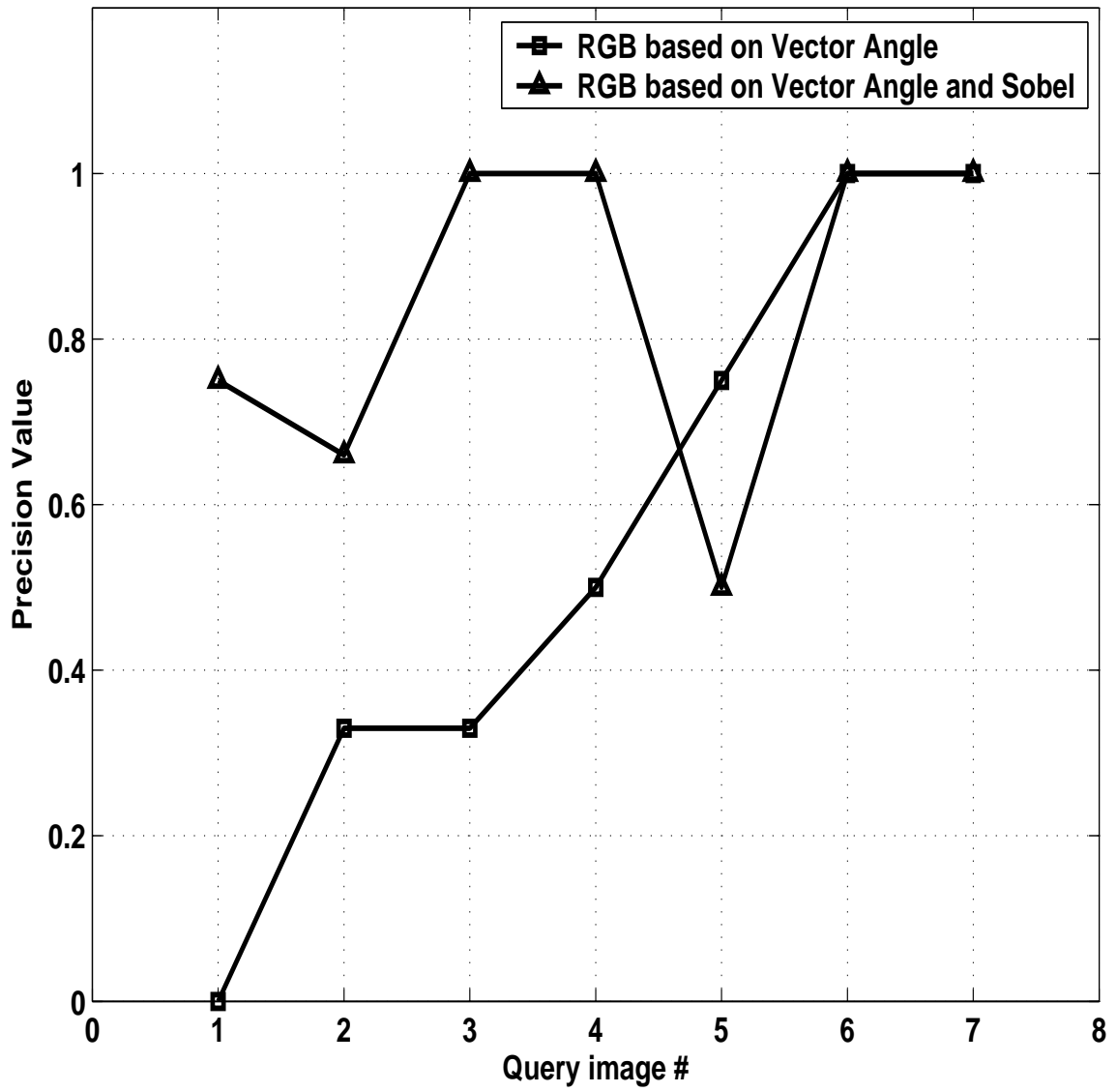


Figure 26. Precision values for descriptor based on vector angle and the modified descriptor based on Sobel edge detection and Vector angle

## CHAPTER 4

# A COMPRESSION SCHEME FOR VIDEO SURVEILLANCE

In this section, compression and motion segmentation are discussed and a method of separately coding foreground and static background is proposed for videos with little motion content.

Most surveillance usually takes place in environments where objects are static for extended durations of time. Thus, there is redundancy in terms of very little movement per frame and unchanging backgrounds. Moreover, there are multiple video cameras viewing the same scene at the same time. So, apart from multiple locations, multiple videos can originate from the same location.

The Movie Picture Experts Group (MPEG) standards take into consideration the motion details and background redundancies between frames in a video. Because of this, the MPEG standards result in highly complex systems while providing low bit-rates. MPEG schemes however, are applicable in general to all types of videos and do not make an effort to exploit features of the environment in which a video occurs. Especially in cases where there are multiple videos in the same environment, there is scope for exploiting redundancies in backgrounds than just motion between frames of the same video.

This chapter consists of two sections. The first section gives some past research done in this area, and the next section discusses contributions of this research work.

## 4.1 Related Background Work

There have been attempts in the past to exploit the fact that the background in video conference setup has little or no information to convey. A method where mean absolute error (MAE) is computed between corresponding blocks in consecutive motion-predicted (P) frames and a decision to code the block is made if the MAE is above a certain threshold is discussed in [48]. This method is one of the earlier attempts to reduce bit rate using region based coding.

Another scheme [49], where dynamic sub-window skipping (DSWS) is carried out depending on motion activity computations on sub-windows. The bits saved in skipping are used to enhance the regions of interest in the non-skipped windows. A low complexity scheme is also proposed to compose and trace the unavailable motion vectors of the skipped sub-windows. Multiple JVT H.263 encoded video streams are variable length decoded and transcoded into lower data rates. Here two measures are used:

- The mean accumulated magnitude of motion vectors of the  $m^{th}$  sub-window

$$SM_m^{MV} = \frac{1}{N} \sum_{n=1}^N (|MV_{m,n}^x| + |MV_{m,n}^y|) \quad (28)$$

where  $N$  is the number of MBs in a sub-window and  $(MV_{m,n}^x, MV_{m,n}^y)$  is the motion vector associated with the  $n^{th}$  MB of the  $m^{th}$  sub-window.

- The mean of accumulated absolute difference (MAAD) of the sub-window is defined as

$$MAAD_m = \frac{1}{N} \sum_{n=1}^N \sum_{x,y \in MB_{m,n}} (|f_m(x,y) - f_m^{prev}(x + MV_{m,n}^x + MV_{m,n}^y)|) \quad (29)$$

The mean accumulated magnitude of motion vectors of a sub-window can be used as a good indication of its motion activity [49]. A sub-window is classified as active if the sum (Equation 28) is larger than a predetermined threshold, say  $TH_{MV}$ .

An inactive window is skipped if its associated MAAD value defined in Equation 29 is below a threshold, say  $TH_{MAAD}$ . Thus, the MAAD value acts as a constrain determining the extent of sub-window skipping. It helps in preventing accumulation of error caused by steady but slow motions.

The advantages of the methods discussed above is that they use the fact that the background is usually unchanging and static in certain situations and use it to increase video compression and even enhance relevant segments of the images.

A related work is discussed by Pons et. al. in [50], where one reference frame is used to ‘scale’ down the DCT coefficients of a group of pictures using differential encoding. This method called ‘Scaled MJPEG’, however, leads to large variations in the difference images than the proposed MJPEG-DE method and needs additional analysis for updating the reference frames.

## 4.2 Research Contributions

In surveillance, activity can be low most of the time but once it does happen, it can be quite important. Described here is a compression scheme that uses Motion JPEG with differential encoding to provide fast compression with low complexity. This process segments out the objects that are in motion and further tries to match them to a scalable database of images. This matching is done using the shape and color information and is described in the next chapter.

This scheme also builds a codec that can embed critical information of the video stream in a ‘context header’ which can be parsed for obtaining a quick summary of the stream. The proposed codec also allows for decoding starting from intermittent locations of the bitstream making it easier to skip unwanted parts of the stream at the user end.

### 4.2.1 Motion JPEG with Differential Encoding (MJPEG-DE)

MJPEG for video coding still has some appeal due to its simplicity and quality. In video conferences and remote surveillance, due to the real-time nature of such applications, low latency in processing and transmission is also essential. An attempt is being made to develop a system specifically designed for video conference and meeting applications. This paper discusses aspects of this system and provides some results obtained.

As part of the preliminary research for developing such a system a method for extending MJPEG to incorporate differential encoding in the DCT domain was developed [51]. In this system, called the Motion JPEG with Differential Encoding (MJPEG-DE) each frame was transformed and quantized in a way similar to JPEG coding. However, only the difference between the coefficients of the current and the previously reconstructed frame was encoded.

The key difference between [50] and the MJPEG-DE method is that, in the latter, the difference between the DCT coefficients current frame and the previous frame is encoded instead of a previous reference frame as shown in Equation 30. Hence, error propagation through the system is reduced. This also reduces the dynamic range of the bits that are to be coded per frame (assuming more similarity between two adjacent frames than a reference frame and a frame some distance apart) as compared to using one initial frame as reference without a major perceived loss in quality.

$$\delta K_t(n) = K_t(n) - K_{t-1}(n) \quad (30)$$

where,  $K_t(n)$  is the  $n^{th}$  DCT coefficient of the frame at time instant  $t$ . At time  $t=0$ ,  $K_{t-1}$  is initialized to zero.

At the decoder, the encoded difference is added to the DCT coefficients of the

current frame to reconstruct the next frame as shown in Equation 31.

$$\hat{K}_t(n) = \delta K_t(n) + \hat{K}_{t-1}(n) \quad (31)$$

where,  $\hat{K}_{t-1}$  is initialized to zero at time  $t=0$ .

Error frames are not obtained or encoded as no large error propagation is anticipated. However, periodic refreshes are carried out to make sure changes in scenes do not large differences. The frequency of the refresh frames is based on the nature of the video stream and these frames are inserted at periodic intervals. For a video conference stream, it is expected that very few refresh frames will be required for the duration of the stream.

During encoding, to carry out a frame refresh, the  $K_{t-1}$  frame is initialized to zero and  $K_t$  is encoded in its entirety as a JPEG frame. Though this increases the bit-rate, it also ensures better quality of the encoded stream.

#### 4.2.2 MJPEG-DE with Motion Segmentation

Part of the proposed system consists of implementing motion segmentation as a front end to the MJPEG-DE system. Motion segmentation was carried out by carrying out block-matching on the frames. The mean square error (MSE) given in Equation 32 was used as a measure to find a matching block using the full search method between the current frame and a reference frame.

$$MSE(k) = \sum_{n=0}^N [I_t(n) - I_0(n)]^2 \quad (32)$$

where  $I_t$  is the image frame at time  $t$ .  $k$  is the block number that is being matched on the reference frame and typically goes from 0 to 8 in a full search scenario centered around block number  $k=4$ .

Typically, the block with the least MSE would be considered a match. However, minor variations in the background values between frames creates a false illusion of motion. To make the segmentation more robust and not include unnecessary

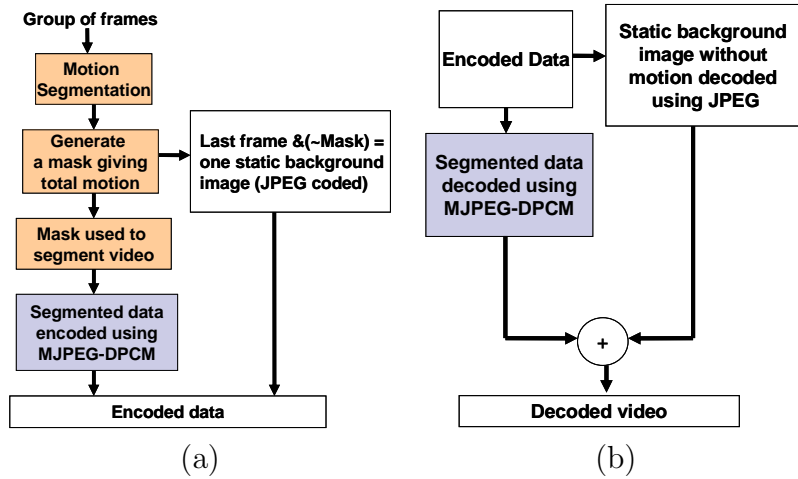


Figure 27. MJPEG DE (a) Encoder and (b) Decoder System

background information, a threshold value was used. Only when the MSE was above the threshold, it was considered that motion had occurred. The threshold value was obtained by approximation and varies depending on the video content.

Based on the above segmentation a mask image was created. Each time a block was considered to be in motion, all the pixels in the block were set. This process of block matching was then carried out for all the frames. Blocks in the mask image once 'set' remain set.

Once all the frames are processed, the mask image is used to recreate the video stream consisting of only the image parts with motion. The background is obtained by subtracting the original frame from the motion segmented frame.

The newly segmented video stream is then MJPEG-DE encoded and the background and mask images are JPEG encoded as shown in Figure 27a. Thus, the result of encoding result in two parts:

1. MJPEG-DE encoded motion segments of the individual frames of the video.
2. JPEG encoded background image for a group of related frames.

During reconstruction, the MJPEG-DE data is decoded, and the background image is added back to all the frames to obtain the decoded video stream. The block

**Table 5. Video prioritization based on motion**

Set of video streams	No. of frames	Priorities obtained
Foreman, Container, News	30	1, 3, 2
Doorview, Viewfromoppositedoor, Centerview	30	3, 1, 2

diagram for the decoder is as shown in Figure 27b. The consistency of backgrounds in certain video applications makes this a viable concept for compression. The idea was applied to video streams in a meeting scenario as well as a port scenario and better compression ratios were obtained over the MJPEG-DE solution. The end result is that only parts of the frames that are actually in motion get coded, while the parts that are static are coded once and added back during decoding. As long as the contents of the frames do not vary much and the camera movement is kept minimal, good compression can be obtained. In case of varying content, small groups of frames can be made into sequences and coded.

#### 4.2.3 Results of MJPEG-DE with Motion Segmentation

The results applying the above mentioned method are given in Figures 28-29. Figure 28a, shows a frame from the input stream. Figure 28b, gives the segmented image that was generated. Similarly, Figures 29a, shows a frame from another video. The motion segmented result is in Figure 29b.

Some of the results from implementing MJPEG-DE scheme are shown in Tables 5, 6 and 7. JPEG compression used was lossy and maintained at a quality factor of 75 for all the tests. The compression ratios between plain MJPEG and MJPEG-DE are given in Table 6. Compression ratios in the order of two to three times are obtained using this method. Due to its simplicity, this method is executes much faster than the H.264 algorithm.

This method is a simple but effective starting point for developing a low-complexity system for applications with static backgrounds and low foreground motion.



Figure 28. a) Single input frame and b) Motion segmented image



Figure 29. a) Single input frame and b) Motion segmented image

**Table 6. Compression ratios between MJPEG and MJPEG-DE system**

<b>File Name</b>	<b>Compression Ratio MJPEG/MJPEG-DE</b>
Yellow car	1.45
Yellow truck	1.45
Blue car	1.43
Meeting	5.5
Container	3.6
Silent	2.8

**Table 7. Speed of execution between H.264 and proposed system**

<b>File Name</b>	<b>Speed of execution ( H.264/MJPEG-DE)</b>
Meeting	37.7
Container	38.7

## CHAPTER 5

### VIDEO SUMMARIZATION

In this chapter, a method for embedding the result of analysis of the video in chapters 2 and 3 into a compressed video stream is provided. Such embedded information acts like a ‘snapshot’ or ‘summary’ of the video giving its highlights and can be very useful for video ‘browsing’ or ‘search’ purposes.

Video summarization is a field which attempts to extract semantic details from videos and create quick and easy to navigate summaries of the same. An important consideration in such a classification is the input domain. *”A narrow domain has a limited variability in all relevant aspects of its appearance”* [52]. A narrow domain makes it easier to define and incorporate semantic classification. for example, medical, surveillance and industrial fields can have specific and controlled video footage that can be considered narrow domains. Broadcast video, on the other hand has a wide domain. *”A wide domain has an unlimited and unpredictable variability in its appearance even for the same semantic meaning”* [52]. Earliest classification tasks were based on specific printed text. But with the advent of the internet and the world wide web, content based querying and retrieval have gained in importance. Such systems aid in retrieval on the world wide web, digital libraries and databases, video education, browsing highlights of meetings, video conferences and video on demand.

The goal of such systems, as defined in [53], is to reduce the ‘semantic gap’. *”The semantic gap is the lack of coincidence between the information that one can extract from the ”multimedia material” and the interpretation that the same data have for a user in a given situation”* [52]. Roach et. al. [53] further qualifies this as a *”lack of coincidence between the formative and cognitive information”*. Formative information is the shape, color, pattern information that makes up the video, cognitive information is the ‘knowledge’ that the viewer interprets from the formative information.

Most video summarization techniques use images or key shots for summarizing the video. This gives a visual description of the video content. In a summarization method based on shot importance [54], video streams are segmented into shots and combined based on similarity measures. Measure of importance are generated depending on normalized weights of the segments and the rarity of each shot. These shots are then combined to form a summary of the video. Larger shots are given more importance and shorter shots get less importance. Thresholding is also used to keep the number of important shots to a desirable level. Divakaran et. al. describe a motion and color based summarization scheme in [55]. In this method, it is argued that large motion translates to large change in color information. Depending on motion, video segments are differentiated into simple and complex streams. A single frame is enough to summarize the simple motion segments. On the other hand, segments with large motion are further clustered and summarized based generating RGB histograms for key frames and comparing them with each other.

In [56], dangerous situations like abandoned object detection and certain predefined human events are considered. In this work, slow environmental changes are absorbed as part of the background, while untouched or abandoned objects are considered part of a foreground until they are classified and detected. This method works well when few moving objects are present at the scene, with an increase in moving objects, the chances of occlusion of abandoned object increase. Shape features and classification by a neural network is carried out to discriminate among different causes of change. With low, medium and high complexity the system worked correctly with a rate of 95%, 75% and 33%.

## 5.1 Video Classification Methodologies

1. Video Summarization: Summarization involves extracting ‘shots’ for videos. A ‘shot’ is defined as one important run of the camera to expose a series of

frames [57]. Summarization aims at compressing the video in a cognitive sense rather than a formative sense [53]. Video summarization usually depicts the results in a graphical format. Video skimming creates a shorter version of the video and helps in getting a quick overview of the video.

2. Video Indexing: Video segments are classified and based on semantic details and indexed or labeled and each new video is assigned to one of many pre-defined classes.
3. Retrieval by Example: In this method the system is given examples on what is being looked for. Color, shape and other such details are used to retrieve images or videos based on the queried examples.

In a related work described in [7], movie content indexing and summarization was carried out. As movies classification covers a very wide domain, the summarization described here deals with identifying conversations, exciting and emotional scenarios so far. Audio and visual information is used for summarizing. Moreover, some movie making techniques like the 180 degree rule (camera placement during conversations between two characters so as not to confuse audience), editing pace during exciting scenes, music for emotional scenes etc. are also taken into account. They showed good recall rate for all three areas of dialogue, exciting events and emotional scenarios. Precision was however slightly low for exciting events. This can be attributed to semantic gap between a classifier and a director's handling of exciting events, namely, editing pace, camera movements and music. A block diagram giving movie structure is given in Figure 30. As seen in the block diagram, a movie consists of scenes which are further composed of events. For example, a bank robbery can be a scene, consisting of events such as excited conversations, car chases, car crashes etc. which are further summarized into shots.

**Table 8. Context Header Example**

<b>Bit 7</b>	<b>Bit 6</b>	<b>Bit 5</b>	<b>Bit 4</b>	<b>Bit 3</b>	<b>Bit 2-0</b>
0: Not a key frame	0: No change in light	0: No motion	0: Single object	0: No change in direction	Reserved
1: Key frame	1: Change in light	1: Motion present	1: Multiple objects	1: Change in direction	Reserved

## 5.2 Research Contributions

### 5.2.1 Video Summarization for Surveillance

Key events in surveillance would encompass intrusion, sudden motion activity, multiple objects in motion and their identification, light changes, direction changes in moving objects, zoom and pan scenarios. In this work, additional changes to the codec described above were made to enable embedding details of the video stream into the compressed bitstream. A ‘context header’ allows us to embed relevant information on motion, direction, lighting changes etc. in the video stream. The encoder also writes out a special set of ‘marker’ bits to the motion vector array that allows the decoder to decode from any user specified I frame in the encoded bitstream. This helps in avoiding tedious searches for certain events. The decoder provides a textual summary of the video stream, allowing a user to quickly find relevant information.

### 5.2.2 Context Header

An example of a context header is given in Table 8. This context header includes most of the relevant information pertaining to a surveillance scenario, while adding minimal overhead to the video stream, which is about 1 byte/frame. But, when dealing with a large number of videos, it could be very useful for retrieving useful information.

### 5.2.3 Semantic Coder

Figures 31 and 32 give the flow chart of the encoder developed for this research. It embeds details in the context header while analyzing a video stream during compression. Light, direction, motion are some of the parameters that are captured and returned to the header. As decoding and context header parsing are more passive (non real-time) in nature, the method described in Chapter 3, to compare the motion segmented object against a database of images and the indicate best three matches is used to convey important activity to the user in real-time.

The decoder is shown in Figure 33. ‘Marker’ bits in the motion vectors allows the decoder to decode from intermittent locations in the bitstream.

In a related work by Puri and others at ATT [8], [9], an idea for segmenting a video stream into coherent or summarized segments is put forth. These segments are then encoded using different encoders that are specifically designed to exploit the semantics of these segments. The segments carry header information regarding the type of encoder used and this header information is used by the decoder at the top level to decide which specific decoder to apply. A flow graph of this system is given in Figure 34. While, this idea seems quite appealing at first, given some thought, some major flaws or shortcomings come to mind. First, is the design implementation of specific encoders and decoders which will need domain experts and software programmers, which makes the task non-trivial. Second, is the complexity of the entire system and third is its real-time execution.

In the method proposed in this research, the underlying encoder and decoder do not change, especially because it is assumed that if the compression is acceptable, then sending extra details on the stream will allow a user to make informed decisions fast and in real-time. Any additional changes are incremental and do not disturb the core working of the codec, only add extra analysis at the top level.

Figure 35 shows the result after parsing the encoded context header. In this case,

only details on which frames were P and I respectively was captured during encoding. Figure 35A gives the result of parsing the context header and decoding all the frames in the video. The decoder is also able to decode from any P frame that is specified by the user and this is depicted in Figure 35B and 35C. If the requested starting frame for a particular decode is not a key frame, then the most recently decoded key frame is used to decode as shown in 35B.

As becomes obvious from the above description and results, this research combines summarization and compression to give details on the semantics of the video and allow for compression based on these semantics.



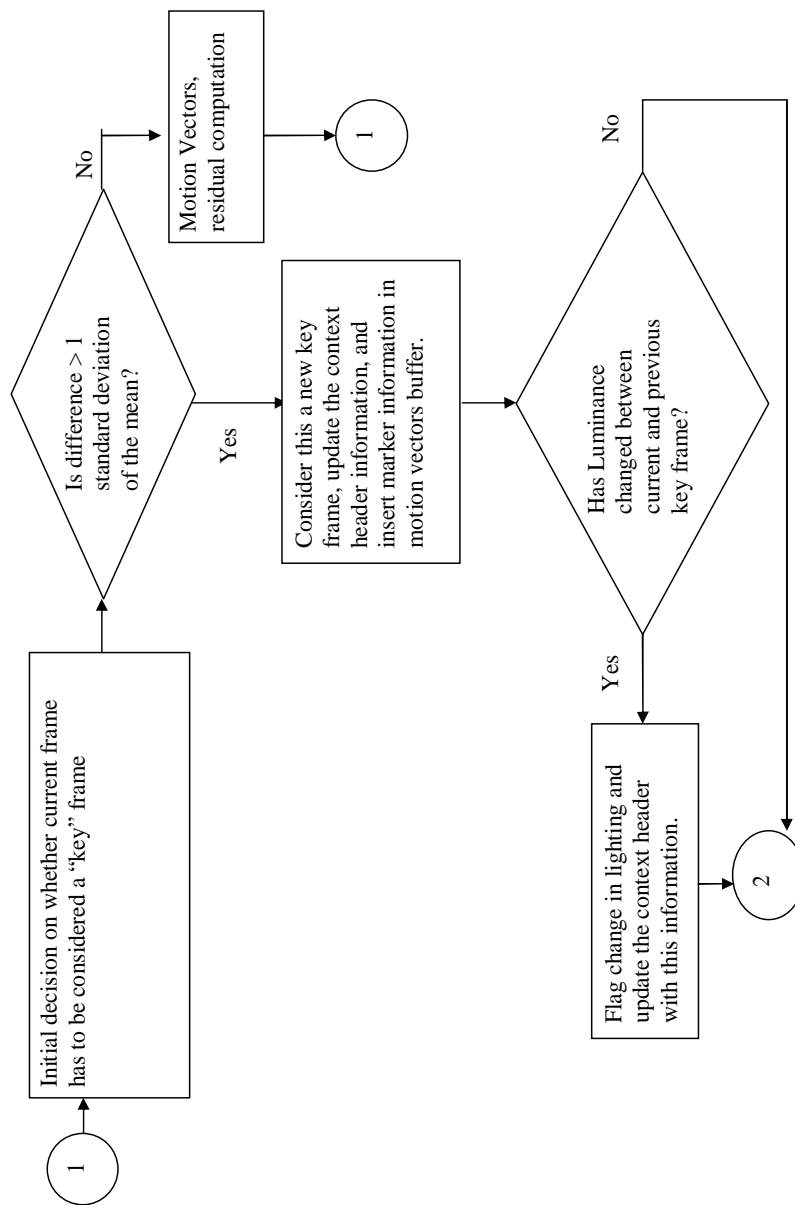


Figure 31. Flow graph the encoder with video semantic data embedded - Part A

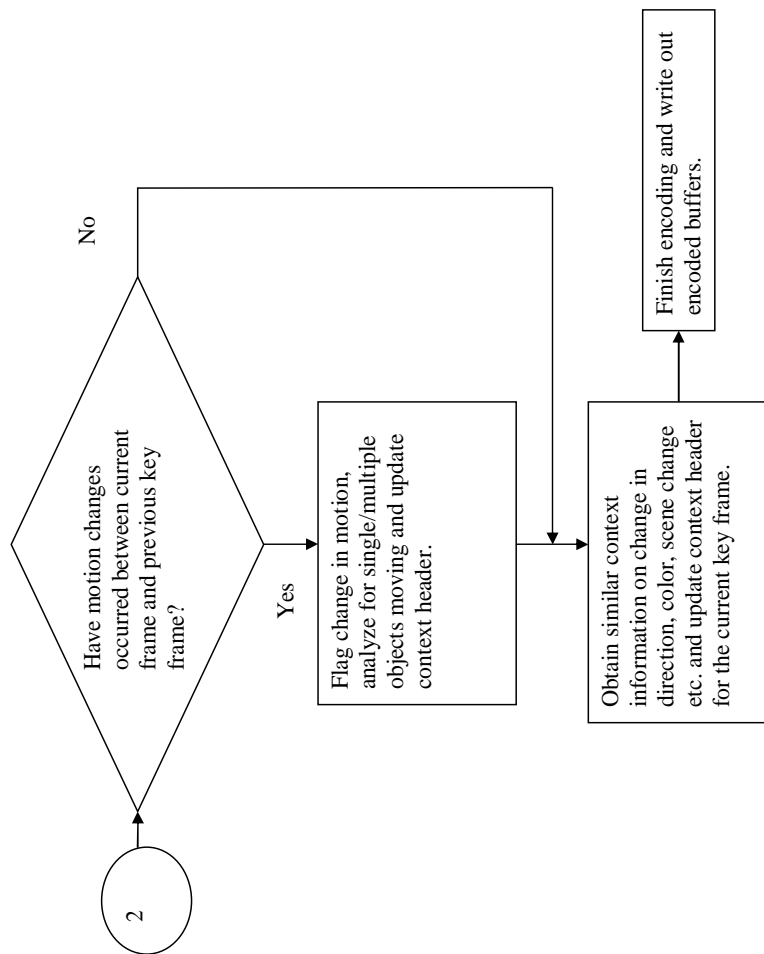
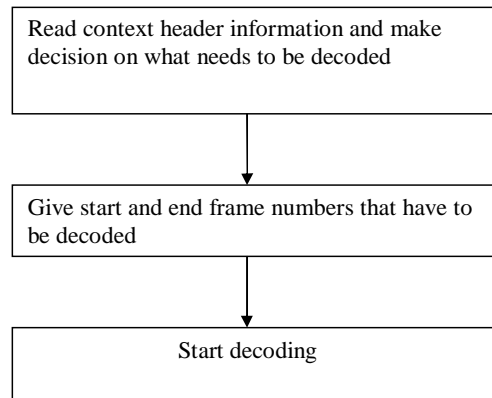
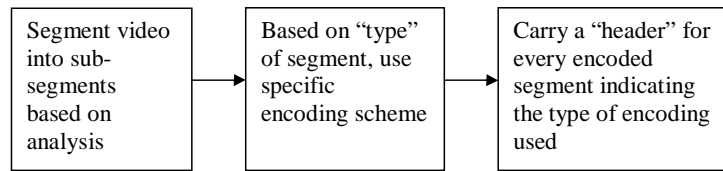


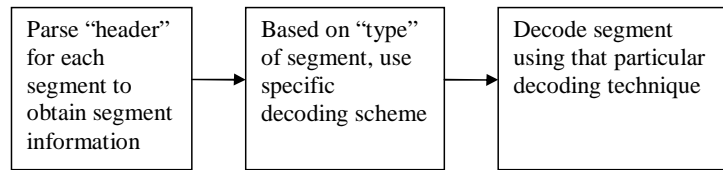
Figure 32. Flow graph the encoder with video semantic data embedded - Part B



**Figure 33. Flow graph for the decoder**



**A**



**B**

**Figure 34. Content Adaptive Encoding (A) and Decoding (B) proposed by ATT Corporation [8] [9]**

DECODING ALL FRAMES	DECODING FRAMES 42-46	DECODING FRAMES 43-46
(frm_no, frm_data)	(frm_no, frm_data)	(frm_no, frm_data)
( 0, 80)	( 0, 80)	( 0, 80)
( 1, 0)	( 1, 0)	( 1, 0)
( 2, 0)	( 2, 0)	( 2, 0)
( 3, 0)	( 3, 0)	( 3, 0)
( 4, 0)	( 4, 0)	( 4, 0)
( 5, 0)	( 5, 0)	( 5, 0)
( 6, 0)	( 6, 0)	( 6, 0)
( 7, 0)	( 7, 0)	( 7, 0)
( 8, 0)	( 8, 0)	( 8, 0)
( 9, 0)	( 9, 0)	( 9, 0)
(10, 0)	(10, 0)	(10, 0)
(11, 0)	(11, 0)	(11, 0)
(12, 0)	(12, 0)	(12, 0)
(13, 0)	(13, 0)	(13, 0)
(14, 0)	(14, 0)	(14, 0)
(15, 0)	(15, 0)	(15, 0)
(16, 0)	(16, 0)	(16, 0)
(17, 0)	(17, 0)	(17, 0)
(18, 0)	(18, 0)	(18, 0)
(19, 0)	(19, 0)	(19, 0)
(20, 0)	(20, 0)	(20, 0)
(21, 0)	(21, 0)	(21, 0)
(22, 0)	(22, 0)	(22, 0)
(23, 0)	(23, 0)	(23, 0)
(24, 0)	(24, 0)	(24, 0)
(25, 0)	(25, 0)	(25, 0)
(26, 0)	(26, 0)	(26, 0)
(27, 0)	(27, 0)	(27, 0)
(28, 0)	(28, 0)	(28, 0)
<b>(29, 0)</b> ← <i>Indicates frame 29 is a P frame</i>	(29, 0)	(29, 0)
(30, 0)	(30, 0)	(30, 0)
(31, 0)	(31, 0)	(31, 0)
(32, 0)	(32, 0)	(32, 0)
(33, 0)	(33, 0)	(33, 0)
(34, 0)	(34, 0)	(34, 0)
(35, 0)	(35, 0)	(35, 0)
(36, 0)	(36, 0)	(36, 0)
(37, 0)	(37, 0)	(37, 0)
(38, 0)	(38, 0)	(38, 0)
(39, 0)	(39, 0)	(39, 0)
(40, 0)	(40, 0)	(40, 0)
(41, 0)	(41, 0)	(41, 0)
(42, 0)	(42, 0)	(42, 0)
<b>(43, 80)</b> ← <i>Indicates that frame 43 was coded as a key frame (I frame)</i>	(43, 80)	(43, 80)
(44, 0)	(44, 0)	(44, 0)
(45, 0)	(45, 0)	(45, 0)
(46, 80)	(46, 80)	(46, 80)
(47, 80)	(47, 80)	(47, 80)
(48, 80)	(48, 80)	(48, 80)
(49, 80)	(49, 80)	(49, 80)
Start = 0	<b>Start = 0</b> ← <i>As 42 is not a key frame, decoder defaults to previous key frame, in this case frame 0</i>	Start = 1
End = 4	End = 2	End = 2
Total = 48	Total = 46	<b>Total = 3</b> ← <i>Decoding frames 43 to 46 means only 3 frames were decoded</i>
<b>A</b>	<b>B</b>	<b>C</b>

Figure 35. Textual Summary after Parsing Context Header at the Decoder

## CHAPTER 6

### CONCLUSION AND FUTURE WORK

In conclusion, increased user interaction for a system translates to customizable software and hardware. Providing video analysis tools along with a semantic codec and a configurable database for object matching, allows for such interaction with minimal knowledge of the system by the user. The goal of this research has been to add the semantics of the video into the compression technique thereby automating summarization and event detection for surveillance videos. A secondary goal has been to keep such a system low in complexity thereby enabling real-time processing and minimal storage requirements. Figure 36 gives a diagram of the entire system with details on what gets processed in real-time and what is processed offline.

Although the Human-Computer Interaction field has been trying to extract semantic information from everyday scenarios, their solutions are not cost effective. By exploiting audio and video information, most semantic details can be obtained without extensive wiring of homes and buildings. By tailoring a video processing system to its environment, a lot of relevant information can be extracted without adding to the cost. This will keep the system simple and also allow user interaction due to low complexity. But, there is the initial human involvement on deciding what the relevant features are and which objects are of interest to be tracked.

With regards to compression and summarization, again, simplicity is the key especially for real-time usage. If like described in [8], different coding schemes for different segments of video are used, then the complexity of both the encoder and decoder increases linearly. Moreover, the ability to tailor a system to an environment diminishes as the variability in a codec increases. User interaction also reduces considerably due to complexity. On the other hand in the proposed scheme, one compression scheme is used but summarized in a header and displayed in a textual format during decoding

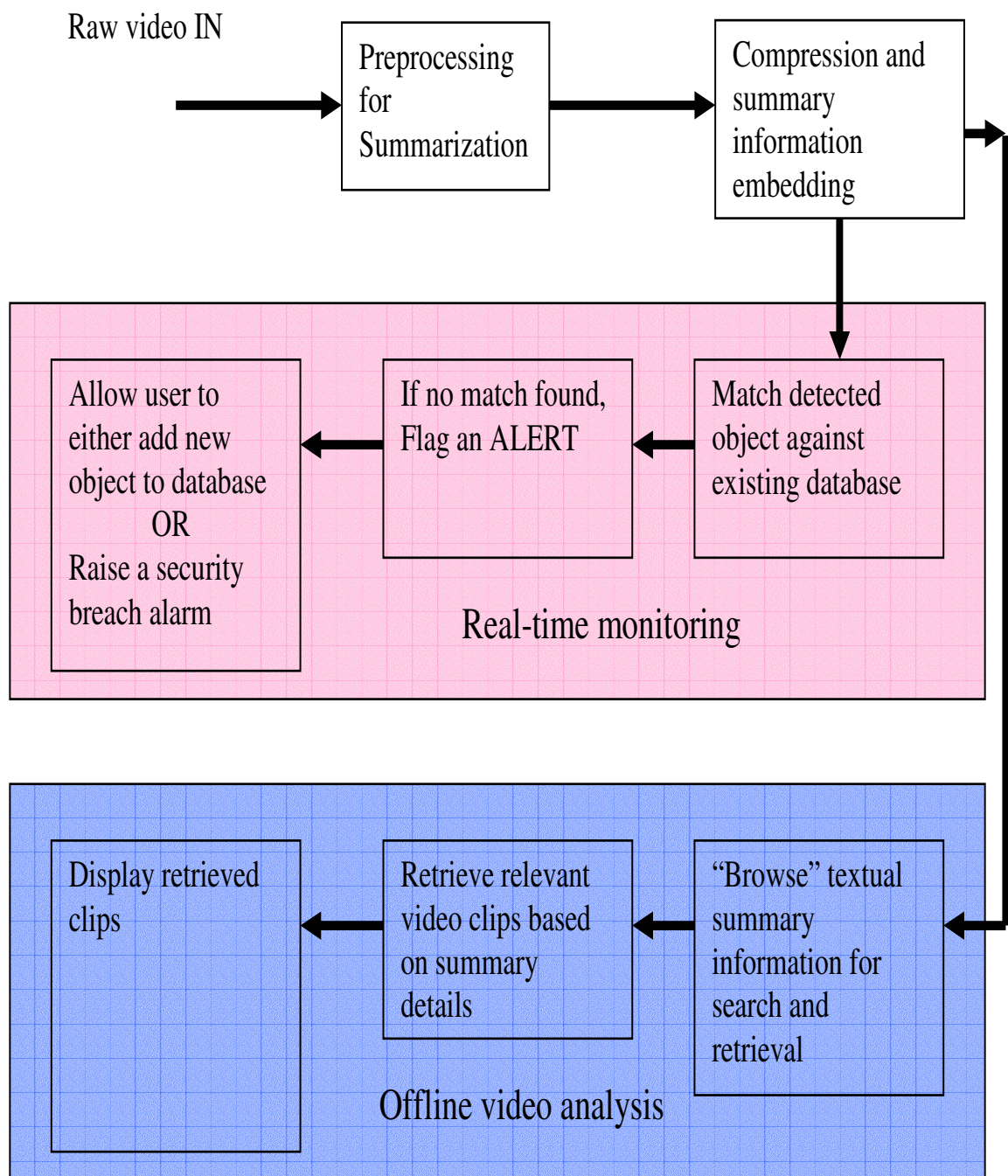


Figure 36. Block diagram of the complete system addressed in this research

making the system practical.

## 6.1 Future Work

Tracking multiple objects in motion is one of the simplest improvements to the current system. The motion vectors can be clustered using the K-means clustering algorithm and different objects isolated. These can again be matched to the database using the RGB histogram created using the Sobel edge detector to decide on whether to update an edge-pixel histogram or a smooth-pixel histogram.

Using audio in conjunction with video can improve event discrimination and classification leading to an extremely sophisticated system. Some work is being done in the area of using audio as well as video in movie database search engines. Typically, any major event in a video where movement is a dominant factor is accompanied by sound. These are features that can be easily detected and categorized as safe or potential danger. Moreover, cameras with night vision or infra-red cameras further help in qualifying the data and are quickly becoming commonplace. Analysis of videos from such cameras and extracting semantic information from them can be fairly involved and interesting.

Another modification would be to store in the database not only the image of an object but one or many clips defining various scenarios in which the object interacts with its surroundings. These scenarios can contain multiple events, which can be an object's regular and known behaviors like speed, movement, sound etc. This gives better insight as it acts as a collection of semantic details related to that object. The success of such a system would depend on a very good understanding of the domain for which it is being built.

Collision and damage detection is an important area that can benefit from such a system. A video surveillance system that can automatically provide such information would be very useful from a security and safety standpoint. Audio and video analysis

can be done for automatic collision detection.

Night surveillance and video analysis using infra red cameras and the information they provide is another interesting area for research and an extension of this work.

## **6.2 Applications of this research**

The medical arena is seeing more use of digital devices and image and video use for remote diagnosis and analysis. A system that provides summarization and object detection in conjunction with compression would be extremely useful in such cases. Such a system can have applications in future applications like remote procedures, augmented medical procedures and remote consultations. The biggest challenge in this area is however is understanding the domain and what can be considered as semantic information for that particular domain.

Remote learning and distant education is another area that can benefit from such a system. This domain is easier to understand and semantic details are easily defined. Such a system would not only help in compression, but allow a user to sift through enormous databases and extract relevant information. Video conferencing is a similar area of application for this work.

A futuristic view of imaging and video devices, points to more and more interaction between device and the user. User interaction is something that is inevitable in such devices with technological advances. In the case of digital still and video cameras such user interaction would mean adding new features, customizing the camera for specific tasks dependent on events and situations (as against simple feature customization that is possible today), essentially making the software on the device scalable. In such devices, a system like that described in this research would be easy to incorporate due to its flexibility, ease of implementation and minimum computational and storage complexities.

## APPENDIX A

### HAUSDORFF DISTANCE MEASURE

The Hausdorff distance is a max-min distance and helps in comparing binary images in part or whole. The edge pixels of the image are considered the features to be compared.

Given two finite point sets  $A = a_1, \dots, a_n$  and  $B = b_1, \dots, b_n$  the Hausdorff distance is defined as

$$H(A, B) = \mathbf{max}(h(A, B), h(B, A)) \quad (33)$$

where,

$$h(A, B) = \mathbf{max}_{a \in A} \mathbf{min}_{b \in B} \|a - b\| \quad (34)$$

and  $\|\cdot\|$  is some underlying norm on the points of A and B [58].

The distance  $h(A, B)$  is called the *directed* Hausdorff distance from A to B. For every point in A, the distance to the nearest neighboring point in B is saved. The maximum of these distances is the directed Hausdorff distance. The Hausdorff distance is the maximum of the two directed Hausdorff distances (namely, from A to B and B to A). Figure 37 gives a pictorial example of this distance for two sets of points A and B. The directed distance from A to B is  $a$  and the directed distance from B to A is  $b$ . The Hausdorff distance is the maximum of the two and is hence  $b$ .

#### A.0.1 Generalized Hausdorff Distance

If one of the points in A or B is an outlier while all other points are very close to each other, the resulting Hausdorff distance is very large and does not represent the true distance. To avoid such errors, the generalized Hausdorff distance was proposed [58]. Here, instead of using the maximum value (Equation 34), the distances are sorted in the ascending order and the  $k^{th}$  value is chosen (Equation 35).

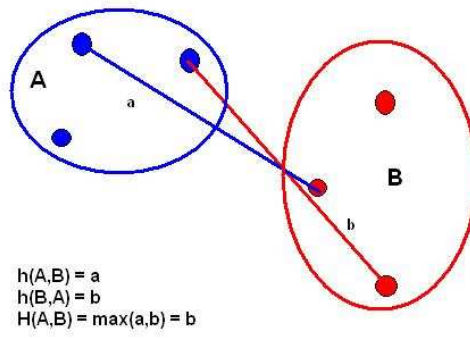


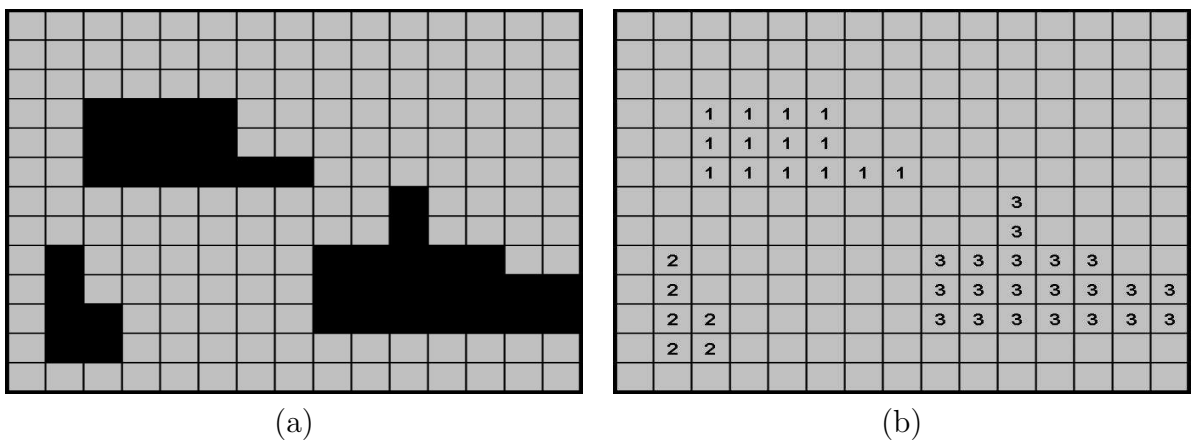
Figure 37. Figure showing the computation of the Hausdorff Distance.

$$h_k(A, B) = \mathbf{k}_{a \in A}^{th}(\mathbf{min}_{b \in B}) \|a - b\| \quad (35)$$

## APPENDIX B

### CONNECTED COMPONENTS

A set of pixels in which each pixel is connected to all other pixels is called a connected component [59]. Consider pixels  $p_1$  and  $p_2$  belonging to a region  $P$ . All pixels in  $P$  are considered similar based on some criteria.  $p_1$  and  $p_2$  are said to be connected if  $p_1 \in P$  and  $p_2 \in P$  and there is a path from  $p_1$  to  $p_2$  consisting entirely of pixels in  $P$  (Figure 38a).



**Figure 38.** a)Image with connected regions b) Image with connected components grouped by labels.

It is assumed that the background pixels have a value of 255 (white) and all the pixels belonging to connected regions have a value of 0. Various methods exist for extracting connected components. Three such methods are described below:

- A recursive method where a pixel previously unlabeled is given a new label. Next, all its neighbors which are not part of a background are labeled. The process is stopped if there are no more unlabeled pixels.
- A sequential algorithm which runs two passes over the image. It works on two rows at a time. The image is scanned left to right. If a pixel is 0, then its upper and left neighbors are checked to see if a label is present. If both have the same

label, it is copied. If both have different labels, the upper neighbors label is copied and both the labels are marked as being equivalent. Otherwise, a new label is assigned to the pixel.

In the next pass, all the labels marked as equivalent are set the same label by scanning the entire image.

- The region boundary method detects pixels that are adjacent to the background. The algorithm scans the image row by row from left to right and for each pixel with a value 0, the 8 neighboring pixels are checked in the clockwise direction until a background pixel is found. If a background pixel is found, then the center pixel retains the 0 value, else it is set to 255. The result of this process is that the outline of the region is obtained.

## REFERENCES

- [1] G. C. de Silva, T. Yamasaki, and K. Aizawa, “An interactive multimedia diary for the home,” *Computer, IEEE Computer Society*.
- [2] “Image titled HSI *color space* downloaded from <http://www.blackice.com/colorspspacehsi.htm>,” Black Ice Software, Inc., 2005.
- [3] “Image titled HSV *color space as a conical object* downloaded from <http://www.blackice.com/colorspspacehsv.htm>,” Black Ice Software, Inc., 2005.
- [4] S. Wesolkowski and E. Jernigan, “Color edge detection in RGB using jointly euclidean distance and vector angle,” *Vision Interface*, pp. 9–16, 1999.
- [5] “Image titled *Container4* downloaded from <http://www.eurocentre.fr/index.php>,” Eurocentre Inc.
- [6] “Image titled *Container7* downloaded from <http://www.arch.columbia.edu/studio/spring2003/up/acra/photo%20gallery/markets%20vendors,%20and%20businesses/index.htm>,” Graduate School of Architecture, Columbia University, New York, NY 10027.
- [7] B. Lehane, N. O’Connor, H. Lee, and A. F. Smeaton, “Indexing of fictional video content for event detection and summarisation,” *EURASIP Journal on Image and Video Processing*, 2007.
- [8] A. Puri and M. Civanlar, “System for content adaptive video decoding,” *United State Patent, US 6970513 B1*, Nov. 2005.
- [9] A. Puri and M. Civanlar, “Method of content adaptive video decoding,” *United States Patent, US 6968006 B1*, Nov. 2005.
- [10] “Ni smart cameras for machine vision,” *National Instruments tutorial at <http://zone.ni.com/devzone/cda/tut/p/id/6582#toc0>*.
- [11] T. Nakanishi and K. Ishii, “Automatic vehicle image extraction based on spatio-temporal image analysis,” *11th IAPR International Conference on Computer Vision and Applications*, vol. 1, pp. 500–504, 1992.
- [12] A. Nagai, Y. Kuno, and Y. Shirai, “Surveillance system based on spatio-temporal information,” *icip*, vol. 1, pp. 593–596, 1996.
- [13] Y. Yanamura, M. Goto, D. Nishiyama, M. Soga, H. Nakatani, and H. Saji, “Extraction and tracking of the license plate using hough transform and voted block matching,” *IEEE Intelligent Vehicles Symposium*, pp. 243–246, 2003.

- [14] T. Meier and K. N. Ngan, "Automatic segmentation of moving objects for video object plane generation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 525–538, 1998.
- [15] B. Günsel, A. M. Tekalp, and P. J. L. van Beek, "Object-based video indexing for virtual studio productions," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 11, pp. 769–774, 1997.
- [16] D. Duque, H. Santos, and P. Cortez, "The observer: An intelligent and automated video surveillance system," *International Conference on Image Analysis and Recognition*, pp. 898–909, 2006.
- [17] "Modern port and harbor security: efficient pedestrian and vehicle access control," in [http://www.mobotix.com/en/4\\_technology/index.php](http://www.mobotix.com/en/4_technology/index.php), MOBOTIX AG, 2005.
- [18] I. E. G. Richardson, "Video codec design - developing image and video compression systems," in *John Wiley and Sons, LTD, West Sussex, England*, 2003.
- [19] C. D. Kidd, R. J. Orr, G. D. Abowd, and et. al., "The aware home: A living laboratory for ubiquitous computing research," *Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99*, 1999.
- [20] G. D. Abowd, C. Atkeson, A. Bobick, and et. al., "Living laboratories: The future computing environments group at the georgia institute of technology," *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, 2000.
- [21] B. Flinchbaugh, "Smart camera technology trends," *Texas Instruments Developers Conference*.
- [22] A. D. Smith, "Smart video will help secure Beijing Olympics," *The Dallas Morning News*.
- [23] U. Gargi and R. Kasturi, "An evaluation of color histogram based methods in video indexing," *First International Workshop on Image Databases and Multi-Media Search*, p. 7582, 1996.
- [24] "Advanced color imaging on the MAC OS," in <http://developer.apple.com/documentation/mac/ACI/ACI-48.html>, Apple Computers Inc., 1996.
- [25] "HSV color space," in [http://en.wikipedia.org/wiki/HSV\\_color\\_space](http://en.wikipedia.org/wiki/HSV_color_space), Wikimedia Foundation Inc., 2005.
- [26] T. Carron and P. Lambert, "Color edge detector using jointly hue, saturation and intensity," *IEEE International Conference on Image Processing*, vol. 3, pp. 977–981, 1994.

- [27] C. Zhang and P. Wang, "A new method of color image segmentation based on intensity and hue clustering," *15th International Conference on Pattern Recognition*, vol. 3, pp. 613–616, 2000.
- [28] R. Dony and S. Wesolkowski, "Edge detection on color images using RGB vector angle," *Engineering Solutions for the Next Millennium. IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 687–692, 1999.
- [29] H.-C. Chen, W.-J. Chien, and S.-J. Wang, "Contrast based color image segmentation," *IEEE Signal Processing Letters*, vol. 11, pp. 641–644, 2004.
- [30] J.-H. Jang and K.-S. Hong, "Fast line segment grouping method for finding globally more favorable line segments," *Pattern Recognition*, pp. 2235–2247, 2002.
- [31] D. S. Guru, B. Shekar, and P. Nagabhushan, "A simple and robust line detection algorithm based on small eigenvalue analysis," *Pattern Recognition*, pp. 1–13, 2004.
- [32] S. J. Perantonis, N. Vassilas, T. Tsenoglou, and K. Seretis, "Robust line detection using weighted region based hough transform," *IEEE Electronics Letters*, vol. 37, pp. 648–650, 1998.
- [33] D. R. O and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, pp. 11–15, 1972.
- [34] R. Gonzales and R. Woods, "Digital image processing," in *Addison-Wesley Publishing Company, Reading, MA*, 1993.
- [35] H. Li, D. Pao, and R. Jayakumar, "Improvements and systolic implementation of the hough transformation for the straight line detection," *Pattern Recognition*, vol. 22, pp. 697–706, 1989.
- [36] S. Yuen, T. Lam, and N. Leung, "Connective hough transform," *Image and Vision Computing*, vol. 11, pp. 295–301, 1993.
- [37] M. C. K. Yang, J.-S. Lee, C.-C. Lien, and C.-L. Huang, "Hough transform modified by line connectivity and line thickness," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 905–910, 1997.
- [38] J. R. Smith and S.-F. Chang, "Automated image retrieval using color and texture," *Technical Report CU/CTR 408-95-14, Columbia University*, 1995.
- [39] S. Jeong, "Histogram-based color image retrieval," *Psych221/EE362 Project Report, Stanford University*, 2001.
- [40] M. Flickner, H. S. Sawhney, J. Ashley, Q. Huang, M. G. B. Dom, J. Hafber, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The qbic system," *IEEE Computer*, 28(9), pp. 23–32, 1995.

- [41] M. Swain and D. Ballard, “Color indexing,” *International Journal of Computer Vision*, 7(1), pp. 11–32, 1991.
- [42] B. C. Ooi, K.-L. Tan, T. S. Chua, and W. Hsu, “Fast image retrieval using color-spatial information,” *International Journal on Very Large Databases*, vol. 7, pp. 115–128, May 1998.
- [43] G. Pass, R. Zabih, and J. Miller, “Comparing images using color coherence vectors,” *ACM International Conference on Multimedia*, pp. 65–73, 1997.
- [44] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, “Spatial color indexing and applications,” *International Journal of Computer Vision*, vol. 35, pp. 115–128, 1999.
- [45] R. O. Stehling, M. A. Nacsimento, and A. X. Falcao, “A compact and efficient image retrieval approach based on border/interior pixel classification,” *Conference on Information and Knowledge Management*, pp. 102–109, 2002.
- [46] H. Y. Lee, H. K. Lee, and Y. H. Ha, “Spatial color descriptor for image retrieval and video segmentation,” *IEEE Transactions on Multimedia*, vol. 5, pp. 358–367, 2003.
- [47] W.-Y. Ma and H. J. Zhang, “Benchmarking of image features for content-based retrieval,” *32nd Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 253–257, 1998.
- [48] H. Chen, P. Wu, Y. Lai, and L. Chen, “A multimedia video conference system: using region base hybrid coding,” *IEEE Transactions on Consumer Electronics*, vol. 42, pp. 781–786, Aug. 1996.
- [49] C. Lin, Y. Chen, and M. Sun, “Dynamic region of interest transcoding for multi-point video conferencing,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 982–999, Oct. 2003.
- [50] J. Pons, M. Malumbres, and R. Garca, “Scaled mjpeg: A symmetric video compression method for low bit-rates,” *IASTED/ACTA Press (AI’2000)*, pp. 302–308, 2000.
- [51] S. Kamath and J. Jackson, “Low-bit rate motion jpeg using differential encoding,” in *Asilomar Conference on Signals, Systems and Computers*, Nov. 2004.
- [52] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content based image retrieval at the end of the early years,” *trpami*, 2000.
- [53] M. Roach, J. Mason, L.-Q. Xu, and F. Stentiford, “Recent trends in video analysis: A taxonomy of video classification problems,” *Proceedings of the International Conference on Internet and Multimedia Systems and Applications*, 2002.

- [54] S. Uchihachi, J. Foote, and L. Wilcox, “Automatic video summarization using a measure of shot importance and frame-packing method,” *United State Patent, US 6535639 B1*, 2003.
- [55] A. Divakaran, K. A. Peker, and H. Sun, “Method for summarizing a video using motion and color descriptors,” *United State Patent, US 6697523 B1*, Feb. 2004.
- [56] G. L. Foresti, L. Marcenaro, and C. Regazzoni, “Automatic detection and indexing of video-event shots for surveillance applications,” *IEEE Transactions on Multimedia*, vol. 4, Dec. 2002.
- [57] D. Bordwell and K. Thompson, “Film art: An introduction,” *McGraw-Hill, New York, NY, USA*, 1997.
- [58] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, “Comparing images using the hausdorff distance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850–863, 1993.
- [59] R. Jain, R. Kasturi, and B. G. Schunck, “Machine vision,” in *McGraw-Hill*, 1995.