152554

FOVEA BASED CODING FOR VIDEO STREAMING

by

Çağatay DİKİCİ

B.S. in Electrical and Electronics Engineering, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2004

FOVEA BASED CODING FOR VIDEO STREAMING

APPROVED BY:

Prof. H. Işıl Bozma .....
(Thesis Supervisor)

Prof. Reha Civanlar .....

Prof. Bülent Sankur .....

DATE OF APPROVAL: 16.06.2004

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest sense of gratitude to my supervisor Işıl Bozma for her patient guidance, encouragement and excellent advice throughout this study.

Great thanks to Prof. Reha Civanlar for sharing fruitful ideas about video processing. I would have special thanks to Prof. Bülent Sankur for his advice and valuable assistance in the research.

I am thankful to Sohail Anwer ,my project coordinator at Oksijen Teknoloji A.S., for his endless support during the thesis.

Thanks to Aziz Yücetürk for the discussions on Artificial Intelligence and Neural Networks.

I am thankful to APES team Umut Alp, Hasan Ayaz, Doğan Can, Gökhan Çakıroğlu, Murat Karadeniz, Cengiz Pehlevan, Çağatay Soyer and Murat Tmer, for helping me at the project and sharing experiences and knowledge during the time of study.

This work has been supported by Bogazici University Scientific Research Projects Grant #BAP02S106.

I gratefully acknowledge Doğan Can for his meticulous collection of the APES video database and helping with the experiments.

Last, but not least, thanks to my brother Umit and my parents who made all this possible.

ABSTRACT

FOVEA BASED CODING FOR VIDEO STREAMING

Attentive robots, inspired by human-like vision – are required to have visual systems with fovea-periphery distinction and saccadic motion capability. Thus, each frame in the incoming image sequence has nonuniform sampling and consecutive saccadic images have temporal redundancy. In this thesis, we propose a novel video coding and streaming algorithm for low bandwidth networks that exploits these two features simultaneously. Our experimental results reveal improved video streaming in applications like robotic teleoperation.

Furthermore, we present a complete framework for foveating to the most interesting region of the scene using attention criteria. The construction of this function can vary depending on the set of visual primitives used. In our case, we show the feasibility of using Cartesian and Non-Cartesian filters for the case of human-face videos. Since the algorithm is predicated on the Gaussian-like resolution of human visual system and is extremely simple to integrate with the standard coding schemes, it can also be used in applications such as cellular phones with video.

ÖZET

İLGİ TABANLI VİDEO KODLAMA VE İLETİMİ

İlgi tabanlı robotlar, insana benzer görüden esinlenerek, fovea-çevre ayrımına ve hızlı göz hareketleriyle odaklanma yeteneğine sahip olması gerekmektedir. Bu sonuçla, ardışık imgelerin içinde her yeni gelen çerçeve düzgün olmayan örnekleme, ve ardarda gelen her çerçeve de zamansal artıklık içerir. Bu tez çalışmasında, bu iki belirleyici niteliği kullanarak düşük bant genişliğine sahip ağlar için yeni bir video kodlama ve duraksız iletim algoritması öneriyoruz. Deneysel sonuçlarımız, uzaktan robot erişimi gibi duraksız video iletim uygulamalarındaki iyileştirmeyi ortaya çıkartmaktadır.

Bunun yanında, ilgi kriterini kullanarak, sahnedeki en ilginç bölgeye odaklanabilen ilgi tabanli bir iskelet sistem sunulmaktadir. Bu ilgi tabanlı fonksiyon, kullanılan ilkel görme serisine göre değişiklik gösterebilir. Biz bu çalısmamızda, Kartezyen ve kartezyen olmayan süzgeçlerin insan yüzü bulunması işlemindeki kullanılabilirliğini gösterdik. Algoritmamız insan görü sisteminin Gauss-benzeri çözünürlüğünü kullandığı ve standart kodlama yöntemleri ile çok kolay bir şekilde tümleştirilebildiği için, cep telefonlarından video yayını gibi uygulamalarda da kullanılabilir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF SYMBOLS/ABBREVIATIONS xii
1. INTRODUCTION 1
1.1. Motivation 1
1.2. Literature 2
1.2.1. Video Source Coding 2
1.2.2. Attentive Vision 3
1.2.2.1. Face Detection 4
1.3. Problem Statement 4
1.4. Contributions of the Thesis 4
1.5. Thesis Outline 5
2. VIDEO SOURCE CODING 6
2.1. Spatio-Temporal Processing 6
2.1.1. Fovea Periphery Distinction – Spatial Processing 6
2.1.2. Saccadic Motion – Temporal Processing 7
2.2. Foveation 9
2.2.1. Attention Criteria 10
2.2.2. Biologically Motivated Filters 10
2.2.3. Learning: Neural Nets 11
2.2.4. Learning: Cascaded Adaboost 12
3. SYSTEM DESIGN 14
3.1. APES's Software 15
3.2. Application Software 16
3.2.1. Attention Criteria Learner 17
3.2.2. Attentive Video Producer 18

3.2.3. Online APES Video Broadcasting Tool 19
4. EXPERIMENTS 21
4.1. Video Database 21
4.2. Video Quality Measures 21
4.2.1. Foveal Mean Square Error 21
4.2.2. Foveal Structural Similarity Measure 21
4.3. Attention Criteria 23
4.3.1. Sample Foveas 23
4.3.2. Filter Responses 24
4.4. Attentive Learning 25
4.4.1. Choosing the Filter Subset 25
4.4.2. Determination of System Weights 27
4.4.3. Attention Criterion 27
5. RESULTS AND ANALYSIS 31
5.1. Spatio-Temporal Processing 31
5.1.1. Methodology 31
5.1.2. Results 32
6. CONCLUSION 36
APPENDIX A: VISUAL PRIMITIVES 38
A.1. Cartesian Filters 38
A.2. Non-Cartesian Filters 38
A.3. Filter Responses 39
APPENDIX B: VIDEO STREAMING PROTOCOLS 40
B.1. Real-Time Streaming Protocol (RTSP) 40
B.2. Real-Time Transport Protocol (RTP) 40
APPENDIX C: TECHNICAL NOTES ON SOFTWARE COMPONENTS USED 41
C.1. Matrox Meteor- MIL 7.5 41
C.2. Intel Image Processing Library v2.5 41
C.3. Intel Signal Processing Library v4.5 41
C.4. OpenCV Project 42
C.5. Microsoft DirectX 9.0 42
C.6. Helix Producer 9.1 42

C.7. APES API 42
C.8. Helix Streaming Server 43
APPENDIX D: APES APPLICATIONS USER GUIDE 44
D.1. Compiling and Running the Applications 44
D.1.1. Prerequisites 44
D.2. Using The APES API 45
D.2.1. AttentionInterface 45
D.2.2. BiologicFilters 45
D.2.3. Encoder 45
D.2.4. Pre_processor 46
D.2.5. IPLHelper 46
D.2.6. IOManager 47
D.2.7. SSIM 48
REFERENCES 51

LIST OF FIGURES

Figure 2.1. Visual field and foveation 6

Figure 2.2. Flowchart of the spatial processing with α-blending 8

Figure 2.3. Flowchart of spatial and temporal processing 9

Figure 2.4. Visualization of Cartesian and Non-Cartesian filters 11

Figure 2.5. Neural network structure of the Cartesian and Non-Cartesian visual primitives 12

Figure 2.6. Schematic description of the detection of an Adaboost Cascade . . 13

Figure 3.1. Hardware configuration of the overall system 14

Figure 3.2. Attentive video streaming system flowchart of the APES 15

Figure 3.3. Graphical user interface of the training program 17

Figure 3.4. Graphical user interface of the training program 18

Figure 3.5. Graphical user interface of the attentive video producer application. 19

Figure 3.6. Web interface of the APES 20

Figure 4.1. Visualization of positive training samples, including 100 human face fovea candidates 24

Figure 4.2. Visualization of negative training samples, including 100 non-face fovea candidates 25

Figure 4.3. Visualization of seven selected visual primitives within 50 Cartesian and Non-Cartesian filter set 28

Figure 5.1. Sample frames that are encoded at 25 $kbits/sec$ fix rate using RealOne encoder 32

Figure 5.2. FMSE and FSSIM values of each frame within a video sequence from APES video database. 33

Figure 5.3. Normalized FMSE values of all encoded frames with respect to fovea sizes. 33

Figure 5.4. Normalized FSSIM values of all encoded frames with respect to fovea sizes. 34

LIST OF TABLES

Table 4.1. Mean and standard deviation of the first 25 Filter Responses calculated from face and non-face training samples 26

Table 4.2. Mean and standard deviation of the second 25 Filter Responses calculated from face and non-face training samples 27

Table 4.3. Mean and standard deviation of seven filter responses calculated from face and non-face training samples 28

Table 5.1. Computational overhead statistics for spatial and spatio-temporal pre-processing in milliseconds ($K = 3$) 35

Table D.1. Brief description of important files in Attention Interface library . 46

Table D.2. Brief description of important files and classes in BiologicFilters library 47

Table D.3. Brief description of important file and classes in Encoder library . 48

Table D.4. Brief description of important file and classes in Pre_processor library 48

Table D.5. Brief description of important functions in IPLHelper library . . . 49

Table D.6. Brief description of important files in IOManager library 50

Table D.7. Brief description of important functions in SSIM library 50

LIST OF SYMBOLS/ABBREVIATIONS

c_S^t(x)	Spatial color map at time t
$c_T^t(x)$	Temporal color map at time t
I_c	Candidate fovea
I_f^t	Fovea region at time t
I_p^t	Periphery region at time t
I_v^t	The visual field image at time t
α	Description of α
Ω_m	m^th visual primitive value
μ	Mean
σ	Standard deviation
Λ	Positive constant
APES	Active PErception System
ATM	Asynchronous Transfer Mode
AVI	Audio Video Interleave
DCT	Discrete Cosine Transform
FMSE	Foveal Mean Square Error
FSSIM	Foveal Structure Similarity Metric
JVT	Joint Video Team
MPEG	Moving Picture Experts Group
MV	Motion Vector
QoS	Quality of Service
RGB	Red Green Blue
RTP	Real-time Transport Protocol
RTSP	Real-time Streaming Protocol
YCrCb	Luminance(Y) and chrominance(CrCb)

1. INTRODUCTION

1.1. Motivation

Motivated by biological vision systems, there has been a growing trend to have robots explore their environment and look at objects in an attentive manner – thereby minimizing the amount of collected information and thus reducing the required computation considerably [1, 2, 3]. Remote communication with such a robot is then achieved via real-time video transmission using standard coding schemes. However, such a video has two intrinsic properties which are not exploited in general by the standard compression algorithms:

- *Fovea-periphery distinction:* Unlike traditional cameras, the distribution of receptor cells on the retina is gaussian-like with a small variance, resulting in a loss of resolution as we move away from the optical axis of the eye [4]. The small region of highest acuity around the optical axis is called the fovea, and the rest of the retina is called periphery. Consequently there is varying spatial resolution within each frame.
- *Saccades:* Secondly, as a consequence of this fovea-periphery distinction, saccades - very rapid jumps of optical axis - are used to bring images of chosen objects to fovea where resolution of fine visual detail is at its best [5]. Saccadic eye movements are one of the capabilities of oculomotor system which is involved in controlling the eye movements. Saccadic eye movements require the computation of the relative position of a visual feature of interest with respect to the fovea in order to determine the direction and amplitude of the saccade. Consequently, consecutive frame sequences in the video stream contain much overlapping information.

In this thesis, the goal is to develop a video streaming system that uses these two properties simultaneously– hopefully leading to a better video quality for such video transmissions. In order to achieve this, we focus on the two aspects of the problem:

Spatio-Temporal Processing: Given a foveated video sequence, we present an approach that exploits these two features and applies spatio-temporal processing to the video accordingly before it is encoded. As the fovea size and the video encoding bit-rates may both vary, we need to redefine the video quality metrics that can be used to assess the performance of such a system.

Fovea Determination: Given a non-foveated video sequence, we present a comparative study of two methods that could be used to introduce a foveal-periphery like sampling of each frame.

In order to achieve foveation, an attention criteria needs to be defined. We investigate the case of using Cartesian and Non-Cartesian filters and show that they indeed hold the promise of providing an integrated basis of encoding and recognition.

1.2. Literature

Two different areas were of interest to our work:

1.2.1. Video Source Coding

The application of varying resolution to video coding and streaming is relatively new [6]. As outlined therein, this approach presents several distinct advantages such as guaranteed compression ratios and speed. Foveal and peripheral regions are coded differently in spatial domain, and the priority assignment of the ATM cells are used for transmitting the regions of video frames with varying priorities [6]. However, in the case of a network congestion, peripheral information which attracts relatively lower attention is lost first. However, since the approach depends solely on the 'Quality of Service' (QoS) parameters, it will potentially have problems on best effort systems like internet. Furthermore, the redundancy along the temporal dimension is not utilized at all. An approach that applies both spatial & temporal processing on MPEG2 streams has been presented in [7]. DCT coefficients of the periphery are quantized in order to reduce the length of the bitstream. However, since DCT coefficients are calculated on eight $\times$ eight blocks, the foveal region definition is limited with rectangular shape. In

order to minimize the blocking artifacts between the foveal and the peripheral regions, image pyramids and raised-cosine blending are used in [8]. However, such an approach requires the generated pyramids also to be transmitted through the channel then implied increased bandwidth. Space-variant approaches such as log-polar mapping of original frame can also be applied for foveation [9], but the increased computation for achieving such transformations impede real-time applications. Finally, if methods that take particular coding algorithms like H.263, MPEG-4 and and JVT into account do not offer compression independent solutions.

1.2.2. Attentive Vision

Attentive robots explore their surroundings in a loop of pre-attention and attention [2]. The aim of the pre-attention stage is to determine the next region of attention. This is achieved through the fovea-periphery mechanism [10]. The distribution of receptor cells on the retina is Gaussian-like with a small variance, resulting in a loss of resolution as we move away from the optical axis of the eye [4]. The fovea is the small region of highest acuity around the optical axis and the rest of the retina is called periphery. A measure of saliency (either bottom-up or top-down) is used to determine the next region of interest. Saccades - very rapid jumps of optical axis - are used to bring images of chosen objects to fovea where resolution of fine visual detail is at its best. In attentive processing, complex processing is applied on the fovea and it has been suggested this response is used in recognition. For example, Galant et al. have investigated the V4 activity of macaque monkeys stimulated with different surface representations and have found the presence of cells selective to both Cartesian (planar texture surfaces) and Non-Cartesian (textured spheres and saddles) stimuli [11]. Previous work on our robot APES has utilized these filters in complex recognition task [12, 13]. The remote access (in particular internet-based) and teleoperation of such a robot requires real-time transmission of thus generated video – which has varying resolution and redundancy. Consequently, video streaming methods that exploit these properties and which can be naturally integrated to these robots become crucial.

1.2.2.1. Face Detection. The face detection and recognition literature is very rich and several comprehensive surveys are presented in [14, 15, 16].

- **Knowledge Based- Methods:** These methods use human knowledge about face. In general, these methods rely on the creation of some basic rules and use of these rules within the frame [17].
- **Feature Invariant Methods:** These methods concentrate on finding the structural features of the face that are independent from pose, lighting conditions and viewpoint. For instance the relationships between the facial features [18], skin color [19] can be classified as feature invariant methods.
- **Template Matching:** Structural patterns of the face are stored to describe the face (hand coded, not learned), and searched within the target image. Deformable face template [20] is an example of this method.
- **Appearance Based Methods:** A set of training images is used to generate templates or models. Then these templates are searched within the target image. Some practical approaches can be listed as eigenfaces [21], neural-nets [22], support vector machines [23], and hidden markov models [24].

1.3. Problem Statement

Suppose that the visual task involves a robot looking at a scene in an attentive manner and a video is generated meanwhile. The objective can be defined as real-time transmission of this data over the Internet so that users can see precisely what the robot is seeing. Moreover, the system should also: i.) allow real-time streaming and ii.) be usable with any particular video compression algorithm.

1.4. Contributions of the Thesis

The major contributions of this thesis are three-fold.

- **Video Source Coding:** In classical video coding like H.263 [25], all the blocks in a frame and between frames are coded with the same priority. However, in

many applications such as attentive robot-video streaming, each frame may have varying resolution and consecutive frames have much similariy. In this thesis, we present a realtime processing algorithm that processes video frames spatio-temporally and hence enables more efficient encoding without any alterations in these standards. Finally, this approach can be used in general in any low-bandwidth video transmission since it matches the fall-off resolution of the human visual system.

- **Foveation Based on Attention Criteria:** Simple attention criteria can be used to define foveal regions for each frame in the sequence. In particular, we use Cartesian and Non-Cartesian filters to construct such criteria and show that these functions indeed hold the promise of introducing fovea-periphery distinction to each frame. The advantage of such an approach is that since these filters are rich enough to be also used in recognition as evidenced by previous work [12], an integrated attentive perception (including pre-attentive, attentive and cognitive stages) and video-streaming framework is obtained.
- **A Complete Real Time System:** A fovea based video streaming system that works real-time is presented.

1.5. Thesis Outline

The outline of the presentation is as follows: In Chapter 2, we first present video source coding based on spatio-temporal processing. We next consider foveation and suggest two alternative approaches that could be utilized – including Haar filters, Cartesian and Non-Cartesian filters. Following, the details of system design of our of our web based real time video streaming tool are presented in Chapter 3. The comprehensive set of experiments are explained in Chapter 4 – along with the introduction of the video Quality Metrics for this purpose. Results of these experiments are discussed in Chapter 5. Finally, the thesis concludes with a brief summary and comments on future work in Chapter 6.

2. VIDEO SOURCE CODING

2.1. Spatio-Temporal Processing

Consider an incoming image sequence. Let I_v^t denote the visual field image at time t. The function $c^t : I_v^t \rightarrow C$ maps each pixel in this region to a value from the color space C. The fovea is represented by $I_f^t \subset I_v^t$. The rest of the visual field is known as the periphery periphery $I_p^t = I_v^t - I_f^t$. The foveal and peripheral regions have different resolutions. While the foveal region is of high resolution, that of the periphery is of much lower resolution. Furthermore, the resolution decreases radially in the outward direction away from the center of the foveal region – known as the fixation point. At each time t, a new fovea is determined. The robot then moves its camera as to fixate on this newly determined fovea. Such camera movements correspond to saccadic eye movements in humans. As a result, an image sequence $\{I_v^0, ..., I_v^t, I_v^{t+1}, ...\}$ is generated.

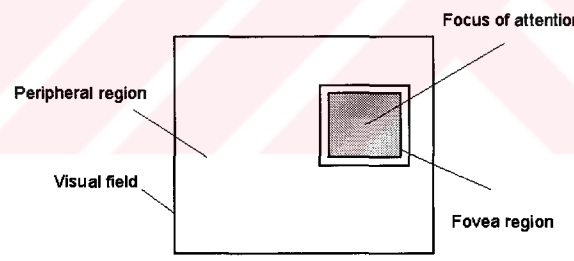

Figure 2.1. Visual field and foveation

2.1.1. Fovea Periphery Distinction – Spatial Processing

In general, most video coding algorithms achieve compression by applying transforms on the original sequence that exploit spatial redundancies such as discrete cosine transform (DCT). However, if each frame is partitioned into foveal and peripheral regions, then processing can vary depending on each region. In the foveal region, the video data is preserved as exactly is. In contrast, in the peripheral region, the video data is made of lower resolution. Let c denote the color map defined on the incoming visual field I_v^t. Consequently, a new color map $\tilde{c}_S^t : I_v^t \rightarrow C$ is defined as:

$$\tilde{c}_S^t(x) = \begin{cases} c^t(x) & \text{if } x \in I_f^t \\ f_S * c^t(x) & \text{if } x \in I_p^t \end{cases} \tag{2.1}$$

where $f_S : I_p^t \rightarrow \tilde{I}_p^t$ is a spatial filter function. The main idea in choosing this filter is that since the peripheral pixels do not attract our attention, the high frequency contained therein is not important and thus can be removed from the data [26]. Consequently, the corresponding image areas can be coded with fewer DCT coefficients. For example, low pass filters such as Gaussian and blurring filters can be utilized.

However, with such a definition, spatial edge artifacts will appear in the reconstructed image after transmission. In order to minimize this, the color map $\tilde{c}_S$ is modified to include α-blending:

$$c_S^t(x) = \alpha^t(x)c^t(x) + (1 - \alpha^t(x))f_S * c^t(x) \tag{2.2}$$

The blending function $\alpha^t : I_v^t \rightarrow [0, 1]$ is time dependent function whose value at time t varies between zero and one. The values of α^t are one or close to one on the fovea and converge towards zero for peripheral pixels as a function of their proximity to the fovea. Consequently, a gradual color transition is introduced in the fovea periphery neighborhood. Furthermore, introducing such a blending enables us to define non-rectangular shaped foveal regions. The flowchart of the spatial processing can be is shown in Figure 2.2.

2.1.2. Saccadic Motion – Temporal Processing

As the camera saccades from the current fovea to the next, an image sequence $\{I_v^0, ..., I_v^t, I_v^{t+1}, ...\}$ is generated. If temporal sampling is fast enough, there is much tem-

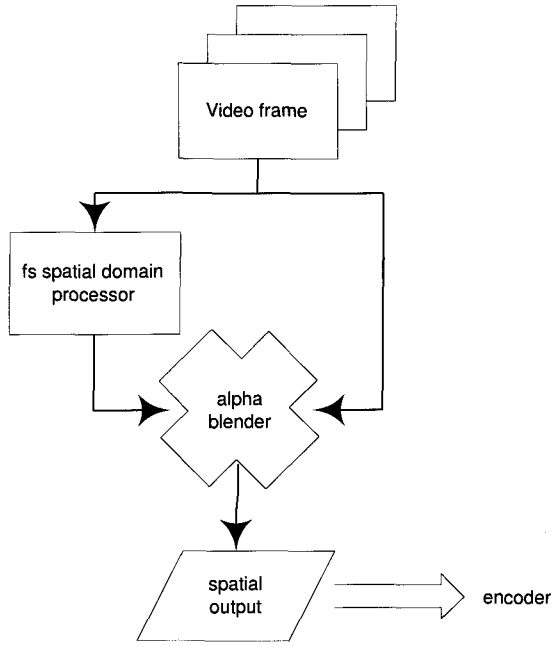

Figure 2.2. Flowchart of the spatial processing with α-blending

poral redundancy between consecutive saccadic image frames. In general, most video coding algorithms also exploit temporal redundancies in their compression schemes. In motion compensation based coding, each frame I_v^{t+1} can be represented as the difference of the current frame and the previous one I_v^t, and thus be coded using motion Vectors (MV). As expected, the efficiency of this type of coding goes up with increased temporal redundancy. In attentive vision, we can increase the temporal redundancy between frames depending on whether foveal or peripheral regions are under consideration. Since the fovea is subject to close scrutiny, all minute detail changes between frames should be preserved and no new temporal redundancy can be introduced. However, this is not the case for periphery. Since changes between frames in the peripheral regions are ignorable to some extent, temporal redundancy can be increased by applying filters across temporal dimension. For example, color maps in the periphery can be updated with every K saccades. In doing so, since the current peripheral region can be estimated from the previous one, the length of the bitstream using MVs is reduced considerably. In this case, the temporal color map $c_T^t : I_v^t \rightarrow C$ is defined as:

$$c_T^t(x) = \begin{cases} c^t(x) & \text{if } x \in I_f^t \\ c_S^{t-t mod K}(x) & \text{if } x \in I_p^t \end{cases} \tag{2.3}$$

If both spatial and temporal redundancies are taken into account, the resulting color map becomes a composition of the two functions c_T^t and c_S^t as $c_T^t \circ c_S^t : I_v^t \rightarrow C$. The flowchart of the combination of spatial and temporal processing can be shown in Figure 2.3.

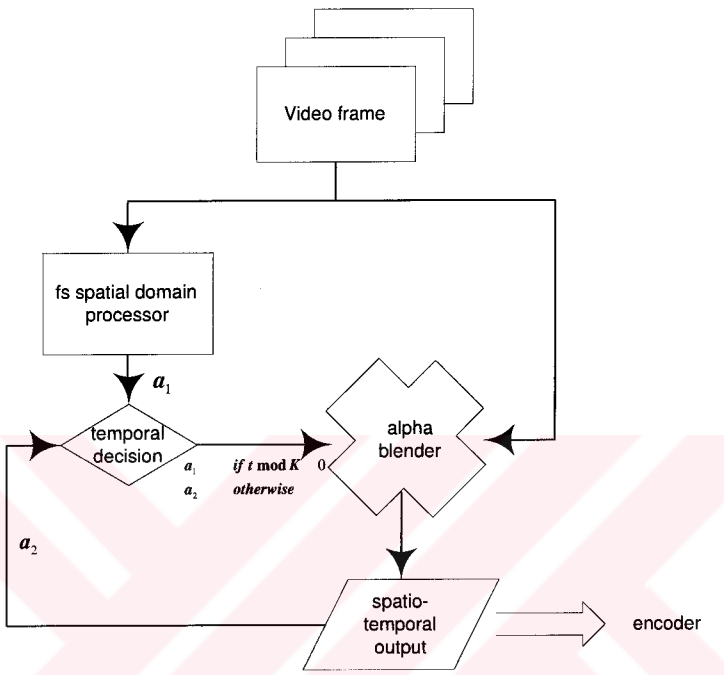

Figure 2.3. Flowchart of spatial and temporal processing

2.2. Foveation

Spatio-temporal coding assumes that each frame is partitioned into foveal and peripheral regions. The next fovea I_f^{t+1} at time $t+1$ is chosen from the set of candidate foveae $C(I_v^t)$ – as determined from the visual field. For each candidate fovea $I_c \in C(I_v^t)$, an attention criteria $a : I_c \rightarrow R^+$ is computed. The candidate fovea that maximizes this measure is then designated to be the next fovea as:

$$I_f^{t+1} = arg \max_{\forall I_c \in C(I_f^t)} a(I_c) \tag{2.4}$$

The attention criteria is a scalar valued function of interest based on the presence of simple features with low computational requirements. This function is determined by the measure of interest on that frame and its definition will vary from task to task depending on the measure of interest such as color, intensity or human facial features.

It should be simple to compute as it is applied many times and need to be very quick if the system is to operate in real-time.

2.2.1. Attention Criteria

The attention criteria a is dependent on the task and should therefore be learned using the set of visual primitives available. Once the task is selected, the construction of a consists of the following stages:

- **Selection of visual primitives:** Suppose there are M different primitives, and let the mth visual primitive be denoted by Ω_m. The value of each visual primitive is obtained via an operator $f_m : I_f^t \rightarrow \Omega_m$ acting on each candidate fovea I_c.
- **Learning:** First, a sample set of foveal images containing both good and bad examples is selected. For example, in face tracking, these are foveal images containing faces or no faces. These filters are then applied to this sample set. Based on statistical properties and the ability to differentiate good foveas from bad, few of these filters are selected. Let us denote this as $M_t << M$ In our case, $M_t = 7$ [1] The attention criteria is then defined as a function of the responses of these filters. Two different approaches are used in order to generate this function
  - ◇ **Biological filters:** Biological filters are used as visual primitives and attention criteria is constructed as a function of the most salient few using either their weighted linear combination or neural-net based learning.
  - ◇ **Haar filters:** Haar filters are used as visual primitives and attention criteria is constructed based on cascaded adaboost learning.

2.2.2. Biologically Motivated Filters

APES uses a biologically motivated set of visual primitives in its attention and cognition stages [12, 13]. These filters consist of Cartesian and Non-Cartesian filters as described originally in [27, 28, 11, 29]. In order to be consistent and integrated with the rest of the system, we also use this set as our basis as shown in Figure 2.4. Note that

[1] Admittedly, a more rigorous approach can be utilized to determine M_t.

$[-1 + 1]$ range is mapped to grayscale in this figure. This set consists of fifty filters – six Cartesian orientations, concentric and radial filters and two hyperbolic filters with five different frequencies. The mathematical formulas of these filters are presented in Appendix A and the interested reader is referred to [12, 13] for further details.

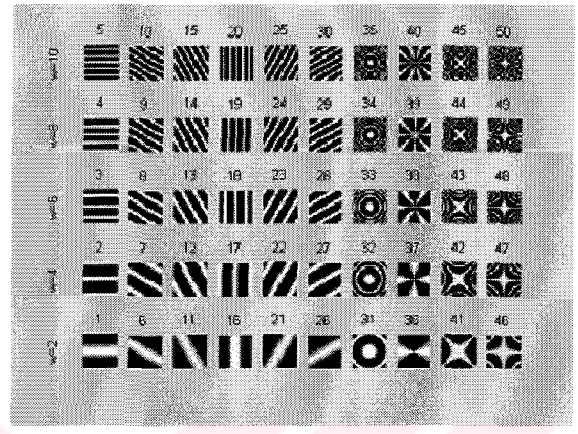

Figure 2.4. Visualization of Cartesian and Non-Cartesian filters

2.2.3. Learning: Neural Nets

The attention criterion a was first constructed using a neural network approach. The reader is referred to [30, 31, 32] for a comprehensive discussion of neural networks. Here, a neural network having M_t inputs is utilized. We used the most differentiate visual primitives while selecting the number t. The details of the selection process can be found in Chapter 4 The selection of the input visual primitives are is The response of each of the M_t selected filters is input respectively. The outputs of the first layer are then fully connected to an intermediate layer consisting of H hidden units with hyperbolic tangent transfer function. Finally the output layer can be trained for the current task (see Figure 2.5). Hence, the output of each neuron within the hidden layer is:

$a_i = \tanh(\sum_{j=0:M_t-1} Wij \times fr_i + b_i)$. Then the output of the system is:

$$a = \tanh(\sum_{j=0:H-1} W_j^{'} * a_j + bo) \tag{2.5}$$

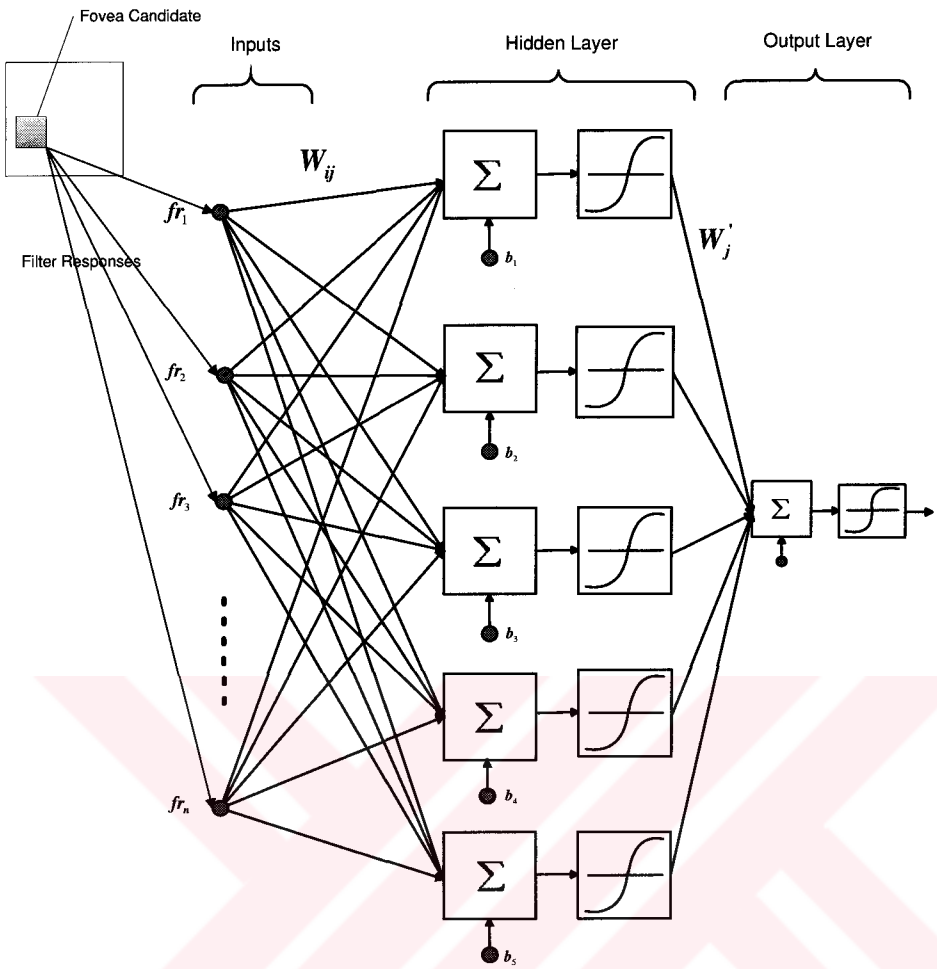

Figure 2.5. Neural network structure of the Cartesian and Non-Cartesian visual primitives

2.2.4. Learning: Cascaded Adaboost

The attention criterion a was also constructed using Cascaded Adaboost Approach [33, 34]. In this learning Haar filters are used as visual primitives. In this approach, several weak hypotheses are created, and the results are combined in order to end up with a final hypotheses which is known as boosting. Moreover, several features are extracted from integral of the image itself. These features are used for the learning mechanism of the system. For each feature, a boosted classifier is trained with a target hitrate and false alarm rate. Afterwards, these boosted classifiers are cascaded within the network such that the strong features are located at early stages of the classifier.

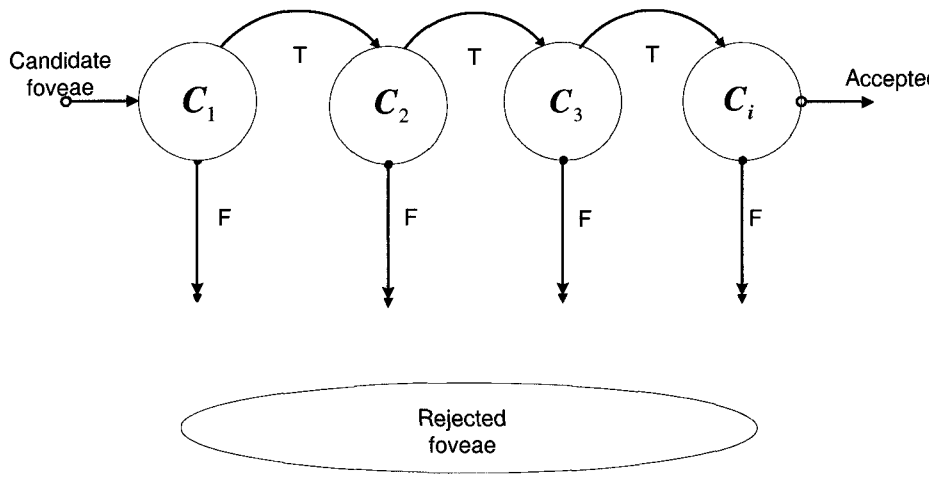

Figure 2.6. Schematic description of the detection of an Adaboost Cascade

From a set of extended Haar-features, Lienhart and Maydt [34] trained a cascade of classifiers, which implements discrete adaboost algorithm. The mathematical formulation of ith classifier is

$$h_i(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_{it} h_{it}(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_{it} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Here, $h_{it}(x)$ is the tth weak learner of the ith classifier. Hence a tree based decision structure is constructed. If the classifier rejects the input in early stages, then the system automatically rejects the candidate without applying the rest of the cascades.

3. SYSTEM DESIGN

Our approach is implemented on video streaming from APES - an attentive robot developed in our laboratory [35, 36]. APES can be remotely controlled and teleoperated and streams its acquired image sequence over the internet. The hardware configuration of the overall system is shown in Figure 3.1. APES grabs its visual surrounding, and after pre-attention and pre-processing functionalities, it encodes the video and push it to a video streaming server. Hence, any registered user can connect and watch the visual field of the APES robot as it explores its current surroundings. One authorized user can also control the APES remotely while watching its captured video in real time. In this setup, Helix^{TM} Project [37] is used as video coding and streaming framework. RTP/UDP/IP is used for transmission of real-time data, and Real Time Session Protocol(RTSP) is used for session initiation and control of the video stream. Furthermore, the remote control of the APES (pan & tilt controls) is sent via TCP/IP for reliable data transmission.

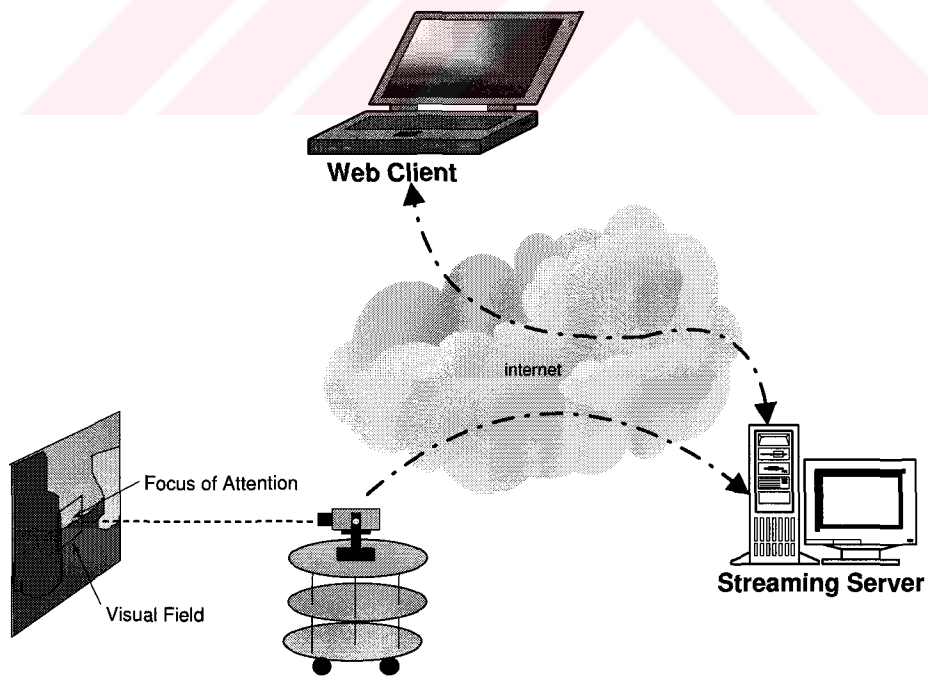

Figure 3.1. Hardware configuration of the overall system

3.1. APES's Software

The block diagram of the attentive video streaming is presented in Figure 3.2. First, real-time video frames are captured in $RGB24$ bit format with size 384×288 at $25 frames/second$. The sequence of the pixel values are aligned as Blue - Red - Green order. The video frames are captured from Matrox Meteor Card, which digitizes the CCD camera outputs and passes to our application. In order to get the raw video data, Matrox Imaging Library's C++ routines are used.

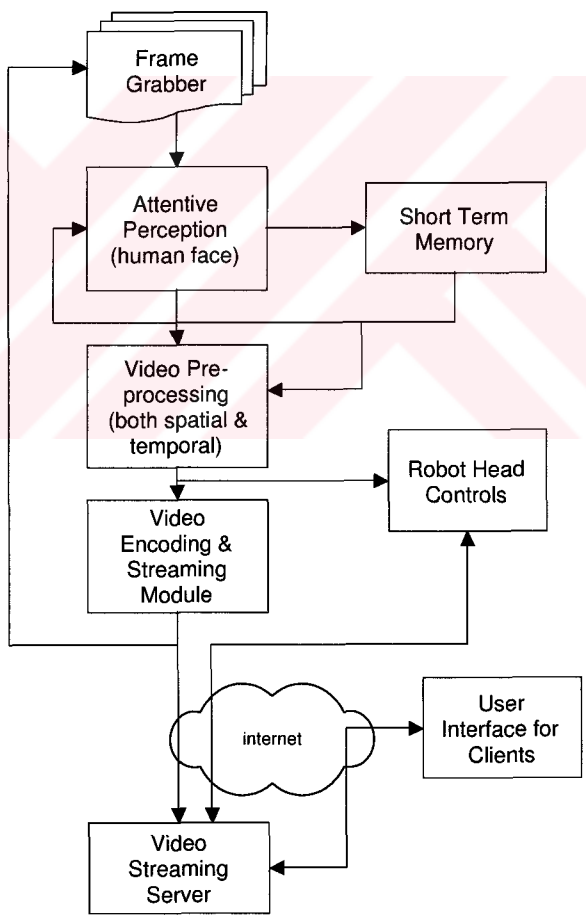

Figure 3.2. Attentive video streaming system flowchart of the APES

Next, the incoming video frame is directed to the attentive perception module. Based on current focus of attention point, a 180×180 visual field I_c is constructed such that the center of visual field corresponds to the center of previous fovea I_f^t. Note that

if the previous focus of attention was located close to the frame borders, the visual field is replaced such that no part of it lies outside of the video frame. Candidate foveae are constructed with an overlapping factor $o_f : 0 < of \leq 1$ and the next fovea is determined based on an attention criteria.

Let us remark that findings on human eye motion[38] have revealed that an eye saccades on average three times per second. This number can go up if there are multiple regions of interest. Accordingly, it is less if there is a dominating region of attention. Similarly, in our system, a foveal region is not computed for each frame. Rather, the saccade rate is held about three frames per second.

In pre-processing, the fovea and the periphery are processed as described in Chapter 2. This frame is then passed to the Helix Video encoder in order to generate a fixed bit-rate video stream. The bit-rate of the system can be configured during the start-up of the application. Following, APES pushes the video stream to a streaming server. which is then responsible for managing and setting up the realtime video streams with the streaming clients. Finally, any web user with RealOne Player installed - can connect to the streaming server and watch the current visual field of the APES robot.

In this thesis, all the codes are written in C++. Because of the time critical processes, some performance primitives of Intel chip-sets are used. We used Matlab for training the Neural Network system. After obtaining the neural-net parameters, the attention criteria was implemented in C++. Additional information about software libraries used in this thesis in Appendix C.

3.2. Application Software

In order to develop a real time video processing and streaming system for internet users, a variety of different applications were designed and implemented:

- Training
- Real time video pre-processing

- Encoding and streaming
- Web interface

The user guides of these applications can be found in Appendix D.

3.2.1. Attention Criteria Learner

In order to construct the attention criteria, VirtualDub which is an open-source video editing tool was modified [39]. VirtualDub is capable of editing pre-recorded AVI video files. The user can access a specific frame data of the video file. Each frame pixel can be retrieved as its 24 bit RGB value. The user can seek also the contents of the video file with a scroll bar, and can mark any desired region of interest fovea. After the marking operation, a message-box asks to the user whether he accepts the selected area or not. If the selected area is accepted, responses of all 50 biological filters are calculated by the system. Moreover these results are saved to an output file. The output file can be used later during the learning stages.

Figure 3.3. Graphical user interface of the training program

In Figure 3.3, a snapshot of the Training Program is shown. If the frame on the main window of the program is you double-clicked, the frame sequence window is created. By using the slider, the desired frame can be searched. If a point within these frames is selected by clicking, a fovea region whose center is on this point is generated

within a message box. If this fovea is accepted, the 50 filter responses are calculated and this region is saved as bitmap for later uses. The flowchart of the training program is shown in Figure 3.4.

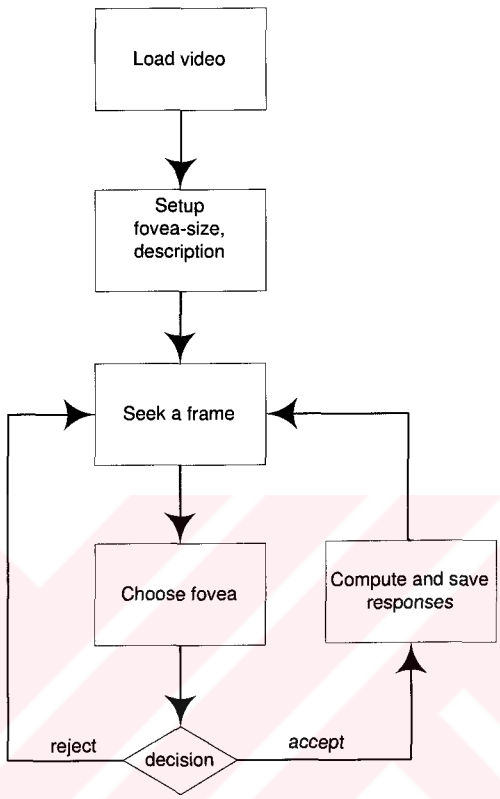

Figure 3.4. Graphical user interface of the training program

3.2.2. Attentive Video Producer

After the training process, the attention criteria need to be tested with random and previously untrained video. For this purpose, an offline video processing and streaming tool was developed. This program has a graphical user interface that allows the loading of any compressed video content including Mpeg4 and Dvix. First, the input video is selected. Next, attention criterion is selected. For now, the attention criteria are "Simple Weight", "Fix Fovea", "Neural-Nets" and "Adaboost". Finally, processing modes (no processing, spatial processing, spatio-temporal processing) are selected. The program is capable of highlighting the border of the fovea. The rest of the program acts like a video player with play, pause and stop options. By using the slider, one can go through the video sequence and search for a particular scene. With the selected configurations, the program processes and displays the video frames in real

time.

Figure 3.5. Graphical user interface of the attentive video producer application.

The user interface of the application is available in Figure 3.5. The user can change the pre-processing mode and attention criteria at run-time. The fovea region can be highlighted by using a checkbox.

3.2.3. Online APES Video Broadcasting Tool

Since the application running on APES should have low memory requirements and be strictly real-time, next an online command line application was developed. The functionalities of the command-line application are very similar to the off-line video processing tool that is described in subsection 3.2.2. The difference stems from the fact that as its name implies, it takes APES's video output as its input. Furthermore, APES uses the selected attention criteria as it is exploring around and generating this video sequence. Finally, it is possible to specify the target bit-rate of the video encoder, and send the encoded video content to a streaming server in real-time. Hence, any web client can connect to the streaming server and watch the APES's visual surrounding, which is identical to what APES perceives.

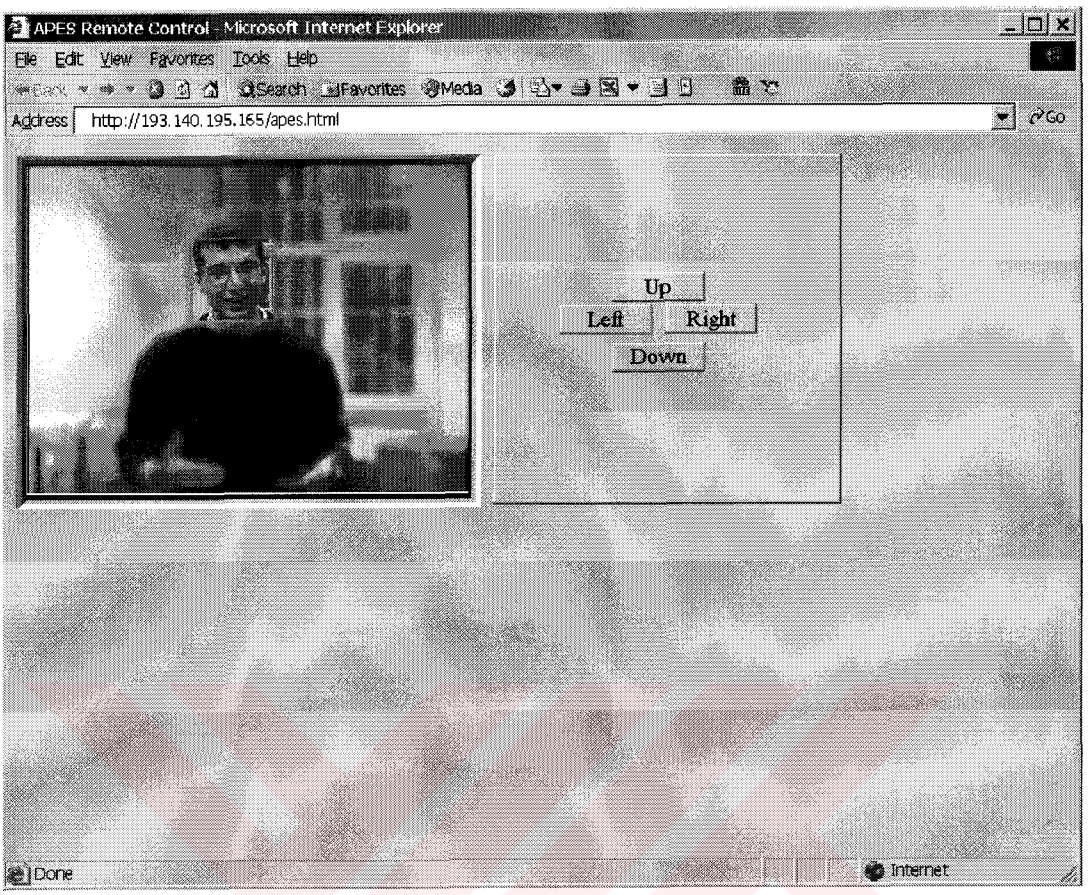

Figure 3.6. Web interface of the APES

A snapshot of the web interface is shown in Figure 3.6. An embedded RealOne player is invoked within the web page, and by using the RTSP url of the robot, the clients can see the the APES's visual field like APES sees. An authenticated user can also control the motor of the APES head, and can navigate the scene with these controls. The four buttons at the right of the figure controls the APES's head movement, and the user is able to pan and tilt the camera. The web user interface of APES is developed by a group from Intelligent Systems Laboratory at Bogazici University [40].

4. EXPERIMENTS

4.1. Video Database

This approach has been tested on a video database generated using APES. APES is made to look at a person with varying poses and proximity. Each video is of 20 seconds long. For each person, nine poses were recorded – including three different frontal, right and left views. During each session, the subject talks and makes random facial expressions. Each of the views are recorded from three different distances – short (two meters), intermediate (four meters), and far (more than four meters) ranges.

4.2. Video Quality Measures

4.2.1. Foveal Mean Square Error

In order to quantify the visual quality of the foveal system, two metrics are used: The first metric is Foveal Mean Square Error(FMSE) which is similar to Mean Square Error (MSE), but is defined only on the fovea[2]

$$FMSE = \frac{1}{\mid I_f^t \mid} \sum_{n=1}^{x \in I_f^t} [c_T^t \circ c_S^t - c^t]^2 \tag{4.1}$$

However, it is well known that MSE value has a clear physical meaning in statistical sense, but it may not always reflect perceived visual quality [41].

4.2.2. Foveal Structural Similarity Measure

As an alternative, Structural Similarity Measure (SSIM) - a metric capturing the amount of structural degradation between two images has been proposed in [41]. In this metric, luminance, contrast, and structural components of the two images are

[2]FMSE is a newly defined quality metric. We leave it to the vision science researchers to check its validity but for our applications, based on visual observations, it seems to be meaningful.

weighted and a quality index is generated. Since there is spatial similarity within the neighborhood of pixels, this structural similarity should also be used while quantifying the image quality. Note that Mean Square Error is independent from the structure and may not be good for the image quality metrics.

The luminance comparison $l(x, y)$ is made by comparing the mean of the images. Here x and y are the images that will be compared. $l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$. Here C_1 is used in order to prevent unstable conditions if the value of $\mu_x^2 + \mu_y^2 = 0$ is very close to zero.

The contrast comparison $c(x, y)$ is made by comparing the standard deviation of two images, as an estimate of the contrast. Then $c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$. Like C_1, C_2 is used in order to prevent unstable conditions if the value of $\sigma_x^2 + \sigma_y^2$ is very close to zero.

The last term is structural comparison $s(x, y)$, which is conducted after luminance subtraction and variance normalization. Then the correlation of $x - \mu_x/\sigma_x$ and $y - \mu_y/\sigma_y$ is used for the structural metric. Geometrically, the correlation coefficient corresponds to the cosine of the angle between the $x - \mu_x$ and $y - \mu_y$ vectors. $s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$.

The combination of these three measures is

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.2}$$

However, since the focus is on the fovea, we use a modified version Foveal Structural Similarity Measure (FSSIM) which is defined only on the fovea:

$$FSSIM = \frac{(2\mu_{c_T^t \circ c_S^t}\mu_{c^t} + C_1)(2\sigma_{(c_T^t \circ c_S^t)c^t} + C_2)}{(\mu_{c_T^t \circ c_S^t}^2 + \mu_{c^t}^2 + C_1)(\sigma_{c_T^t \circ c_S^t}^2 + \sigma_{c^t}^2 + C_2)} \tag{4.3}$$

μ, and σ are the mean and variance of the respective images within the fovea. C_1

and C_2 are used in order to prevent unstable conditions if the values $\mu^2_{c^t_T \circ c^t_S} + \mu^2_{c^t}$ or $\sigma^2_{c^t_T \circ c^t_S} + \sigma^2_{c^t}$ are very close to zero. The values of C_1 and C_2 are selected as $C_1 = 0.01$, $C_2 = 0.03$ respectively. Note that $FSSIM \leq 1$ with equality holding if and only if the source and the target images are identical.

Finally, note that FMSE value indicates the mean square error of the fovea region. If the FMSE value increases, so does the quantity of error increase. On the other hand, FSSIM value indicates the similarity measure of the fovea region by using HVS properties. If the reference frame and the input frame are same, FSSIM value is equal to one - means similar -, otherwise it goes to zero.

4.3. Attention Criteria

4.3.1. Sample Foveas

Attention criteria is constructed based on a sample foveal set – containing both target and non-target objects, faces in our case. VirtualDub has been modified so that the user can manually mark the desired foveal region of each frame in a selected video sequence and the filter responses are automatically generated and stored. The user can specify the following parameters:

- **Frame index:** The frame number in the video sequence.
- **Foveal size:** The target fovea can be selected in any length and width.
- **Foveal center:** The coordinates of the foveal center are specified as a function of the position of the left-bottom corner of the foveal region.
- **Task keyword:** A task keyword is used index all the files containing the calculated filter responses. For example, this phrase may be "face","non-face". For visuality and later use, the candidate foveae that are generated during the learning phase are also stored as bitmap images.
- **Scaling factor:** In order to search the desired object in multiple scales, a gaussian pyramid of the frame is constructed. The scaling factor is the decimation factor of the Gaussian pyramids.

Figure 4.1- 4.2 show positive and negative sample foveas respectively.

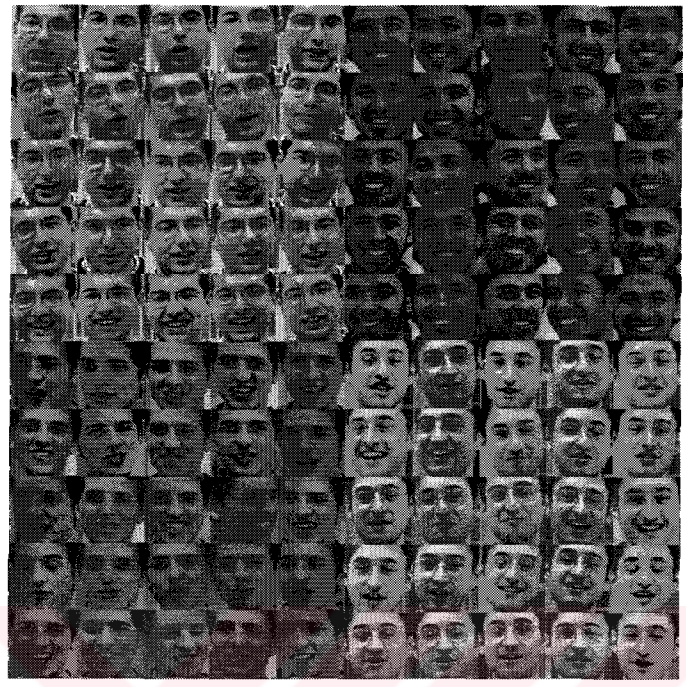

Figure 4.1. Visualization of positive training samples, including 100 human face fovea candidates

4.3.2. Filter Responses

Although foveal size may vary, they are resized to 40×40 before applying the biological filters with the aim of introducing normalization for the learning stage. In order to determine the most salient filters, first all the filter responses are computed for all the sample set and stored using the "Training Software". Average and standard deviation of each filter for both positive and negative samples are then calculated. The filters having well-separated responses for good and bad samples and small variations are designated as salient filters. In our experiments, these turn out to be filters as shown in Table 4.3.

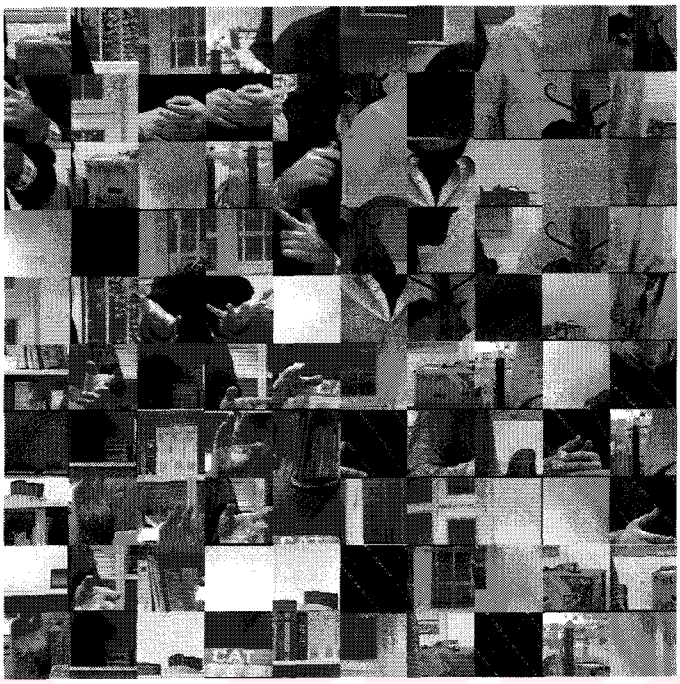

Figure 4.2. Visualization of negative training samples, including 100 non-face fovea candidates

4.4. Attentive Learning

4.4.1. Choosing the Filter Subset

The filter responses o_c^t are calculated and stored by using the training application. In Figures 4.1 and 4.2, you can see some positive and negative fovea candidates respectively. The mean and standard deviation of each filter response is calculated by grouping the responses as positives and negatives. Hence for each filter M, I have the mean and standard deviation of positive faces, and non-faces. Then I observed that seven filter statistics out of fifty filter have their face and non-face means are separated and relatively small variances. So the decision of the input visual primitives are based on the normalized distance of the mean face and nonface results with respect to their standard deviations.

In Table 4.3, you can see the indexes of the selected filters and their corresponding mean and standard deviations. Figure 4.3 contains the visualization of seven selected visual primitives within 50 Cartesian and Non-Cartesian filters.

Table 4.1. Mean and standard deviation of the first 25 Filter Responses calculated from face and non-face training samples

Filter index	μ Face resp.	σ Face resp.	μ Non-Face resp.	σ Non-Face resp.
1	1317.9	297.86	859.38	677
2	585.29	151.82	536.93	479.85
3	447.32	151.9	402.93	364.6
4	325.79	83.526	312.46	282.7
5	271.94	64.432	237.14	190.16
6	1035.8	228.61	704.67	554.1
7	489.88	129.42	274.82	200.8
8	265.59	74.948	168.21	110.94
9	160.65	35.761	98.766	66.382
10	98.835	22.763	71.658	42.185
11	909.59	260.4	767.58	504.28
12	433.42	118.87	260.04	181.17
13	243.04	68.73	153.25	89.423
14	139.39	40.929	94.295	56.479
15	90.27	26.827	67.275	39.154
16	1027.8	322.34	949.64	689.76
17	467.51	170.28	463.76	331.45
18	323.28	105.94	334.08	266.56
19	241.46	74.927	253.75	205.68
20	198.58	63.003	206.02	166.65
21	1004.6	264.64	730.79	464.63
22	488.94	127.66	265.64	192.47
23	268.93	63.167	154.45	93.228
24	151.93	40.784	98.294	60.527
25	96.477	25.486	70.127	41.533

Then, we compose the subset of filter responses $\Omega' = [\Omega'_1\ \Omega'_7\ \Omega'_9\ \Omega'_{12}\ \Omega'_{23}\ \Omega'_{27}\ \Omega'_{46}]$ where Ω'_i is the normalized amount of deviation that the ith filter response from the mean face response. $\Omega'_i = \frac{\Omega_i - \mu_{\Omega_i(face)}}{\sigma_{\Omega_i(face)}}$

Table 4.2. Mean and standard deviation of the second 25 Filter Responses calculated from face and non-face training samples

Filter index	μ Face resp.	σ Face resp.	μ Non-Face resp.	σ Non-Face resp.
26	1055.3	266	679.45	459.79
27	508.38	118.84	291.55	196.26
28	257.13	73.216	183.27	117.41
29	154.54	31.104	110.66	80.984
30	98.774	23.327	77.278	49.945
31	1372.6	389.64	956.41	586.3
32	702.34	191.55	508.48	283.24
33	517.06	162.35	379.95	217.07
34	548.04	151.18	462.92	253.28
35	862.59	240.12	840.56	457.3
36	3177.3	919.45	2183.8	1443.5
37	2247.9	663.13	1562.9	997.02
38	1101	325.34	978	563
39	888.99	276.14	871.42	532.34
40	671.84	212.98	673.76	403.4
41	1946.2	604.68	1318.4	824.08
42	1049.6	304.03	748.47	461.78
43	682.83	203.87	503.95	299.04
44	557.97	170.35	521.36	288.15
45	747.25	196.69	747.49	410.81
46	2272.6	630.94	1450.6	943.98
47	1097.2	267.43	800.05	528.71
48	737.78	197.8	594.04	382.24
49	1195.8	336.7	843.76	549.06
50	821.4	249.79	731.75	468.56

4.4.2. Determination of System Weights

4.4.3. Attention Criterion

The first attention function is a simple weighted linear combination of the response of these salient filters as $a(I_f^c) = \sum_{i=1}^{M_t} \alpha_i \times f_m(I_f^c)$. This attention can be

Table 4.3. Mean and standard deviation of seven filter responses calculated from face and non-face training samples

Filter index	μ Face resp.	σ Face resp.	μ Non-Face resp.	σ Non-Face resp.
1	1317.94	297.85	859.38	677.00
7	489.88	129.42	274.82	200.80
9	160.65	35.76	98.76	66.38
12	433.42	118.83	260.04	181.17
23	268.93	63.17	154.44	93.23
27	508.37	118.84	291.55	196.26
46	2272.64	630.94	1450.64	943.98

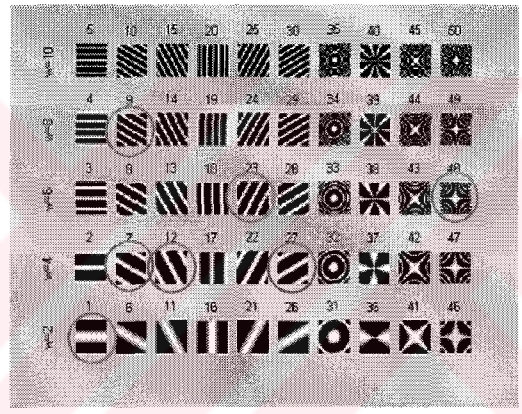

Figure 4.3. Visualization of seven selected visual primitives within 50 Cartesian and Non-Cartesian filter set

considered as a weak classifier.

The second attention criteria is generated using a neural network as explained in Section 2.2.3. For this purpose, a simple feed-forward neural network model with seven inputs, one hidden layer with 15 perceptrons, and one output layer with one perceptron is used. The decision of a face or non-face can be easily made by looking the output of the system. In all of the perceptrons at hidden layer and output layer, a bias term is used and a hyperbolic tangent transfer function. Here are the input weights, biases, and layer weights of the system:

$$W = \begin{bmatrix} -4.77 & 13.78 & -27.19 & 25.32 & -4.85 & -0.30 & -65.66 \\ -2.15 & 0.53 & 0.21 & -0.12 & 1.60 & 0.62 & -0.05 \\ 1.28 & 0.82 & -0.26 & 0.96 & 0.35 & 0.67 & -0.83 \\ 4.78 & 30.71 & 0.31 & -5.69 & -2.33 & -3.07 & 14.91 \\ -1.21 & -0.84 & -0.33 & 4.35 & -0.14 & -0.76 & 0.53 \\ -0.77 & 0.57 & 1.69 & 0.80 & -0.47 & 0.97 & 0.80 \\ 0.36 & -0.29 & -0.26 & -0.76 & 0.91 & -0.59 & -0.65 \\ 1.96 & -2.32 & -0.97 & -0.67 & 0.16 & 3.92 & -6.50 \\ -0.48 & -0.09 & -0.50 & -1.21 & 0.98 & 1.25 & -1.11 \\ -0.07 & 0.82 & 0.22 & -0.11 & -0.74 & 0.57 & -0.29 \\ -0.03 & -0.69 & -0.06 & -0.61 & 0.26 & -0.79 & 0.18 \\ -4.04 & -0.68 & -5.02 & 4.33 & -0.82 & 1.78 & -11.72 \\ -5.89 & 1.91 & 0.48 & -5.30 & 0.45 & -5.10 & 0.77 \\ 0.45 & -1.25 & -16.15 & 7.43 & 0.22 & 4.83 & -0.14 \\ 0.78 & -0.36 & 0.99 & -1.26 & -2.15 & 2.03 & -2.18 \end{bmatrix} \tag{4.4}$$

System parameters are trained using resilient back-propagation algorithm with supervised learning, which means the weights are adjusted such that given the desired output of the inputs, the system minimizes the output error. The network converges to 10^{-3} error rate after training for 702 epochs. W corresponds to the input weight matrix of the neural net. W_{ij} indicates the weight of jth input and the ith perceptron of the hidden layer.

$$b = \begin{bmatrix} 21.85 \\ -1.81 \\ -5.77 \\ -13.53 \\ 1.49 \\ -2.91 \\ 1.36 \\ 1.16 \\ 1.82 \\ -0.17 \\ 2.53 \\ 7.43 \\ 5.63 \\ 0.46 \\ 5.46 \end{bmatrix} \quad b_o = -0.034 \quad W' = \begin{bmatrix} 0.45 \\ -12.97 \\ -13.25 \\ -5.08 \\ 5.87 \\ -6.73 \\ 4.36 \\ -9.33 \\ 6.83 \\ 8.39 \\ -13.58 \\ 6.21 \\ 6.93 \end{bmatrix} \tag{4.5}$$

Moreover all perceptrons of the hidden layer has a biased term given in matrix b, where b_i is the bias of ith perceptron. Similarly, the 15 weights W' and the bias b_o of the output layer is also calculated.

5. RESULTS AND ANALYSIS

5.1. Spatio-Temporal Processing

5.1.1. Methodology

In order to evaluate the performance of the algorithm, an extensive statistical comparative study was conducted. First, the APES Video Database was created by making the robot look at scenes consisting primarily of a person – talking and mimicking at three different distances (long, intermediate, short distances) and three different poses (left, right, frontal views)[42]. Each incoming video was recorded in 384 × 288 resolution, 20 second long, $25 frames/sec$ RGB video in uncompressed AVI format without any preprocessing. Next, the videos in this database were subjected to the following preprocessing:

- Twelve video sequences are selected randomly from the APES Database - with four of long, intermediate and short distance category respectively.
- Next, for each video sequence, the foveal area is determined after visual examination of the video sequence and ensuring that the fovea overlaps with the image area containing the person's face. For each category, two different fovea sizes are considered:
  - For *long-distance* sequences (more than four meters), these are taken to be 100 × 100 and 130 × 130 pixels;
  - For *intermediate distance* sequences (four meters), they are 130 × 130 and 160 × 160 pixels;
  - For *short distance* sequences (two meters), they are taken to be 160 × 160 190 × 190 pixels respectively.
- Each sequence is first only spatially processed - using the two different fovea sizes. In spatial processing, five×five box blur filter is selected as spatial filter function f_S. For α-blending the transition width is chosen as five.
- Each sequence is next spatio-temporally processed for the two different fovea

sizes. K is selected as three in this process.

- The original raw video and the two preprocessed videos are then are then encoded with Real-Media [37] codec with bit-rates $25k/sec$ and $35k/sec$. A sample frame is shown in Figure 5.1.

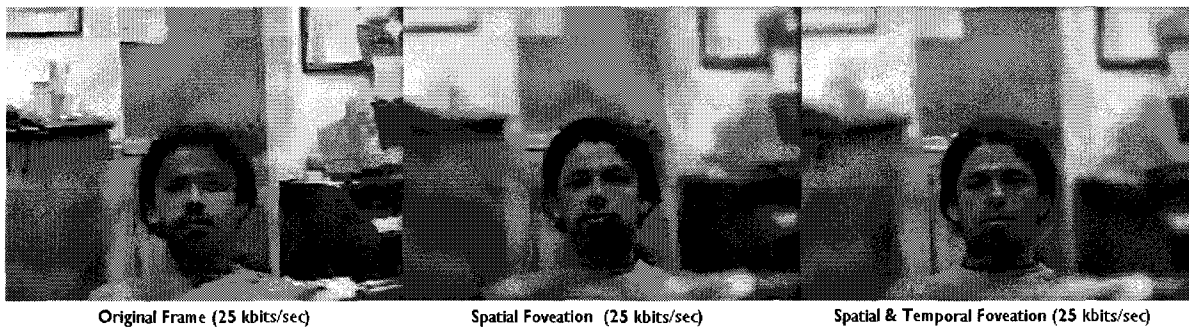

Figure 5.1. Sample frames that are encoded at 25 $kbits/sec$ fix rate using RealOne encoder

In Figure 5.1, the video frames from left to right are no preprocessing ,spatial only and spatio-temporal preprocessing frames respectively. In spatial and spatiotemporal encoding, 130×130 fovea is used on face regions.

5.1.2. Results

In analysis part, all the encoded video frames are compared with the original input raw frames. The comparison is performed as follows: First, for each encoded frame, MSE,FMSE,SSIM,FSSIM values are calculated. Since the number of encoded frames and the number of input raw frames may vary because of the encoding process, minimum FMSE values within five frame neighborhood of input raw frames are selected as reference frames. For each sequence, the first 450 frame statistics are stored.

Figure 5.2 presents FMSE and FSSIM values of a video sequence with a 130×130 sized fovea with classical coding, with spatial and spatio-temporal coding. The sequence is encoded at 35 $kbits/sec$. As expected, compared to classical coding, spatial and spatio-temporal coding improve the FMSE and FSSIM values considerably.

Next, statistical metrics are gathered for each processed sequence. FSSIM and

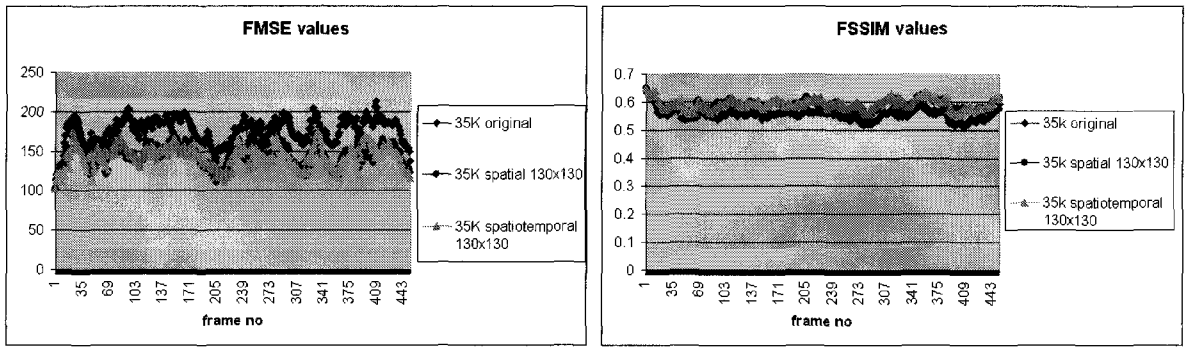

Figure 5.2. FMSE and FSSIM values of each frame within a video sequence from APES video database.

FMSE values are first normalized by dividing each frame's FSSIM and FMSE values by their originally coded frames FSSIM and FMSE values respectively and then averaged over the video. Finally, average normalized FSSIM and normalized FMSE values for each fovea size and encoding bit-rate is computed. For each group, the mean, minimum and maximum values are calculated – using the 2700 frames in each group. Figure 5.3 presents the FMSE results.

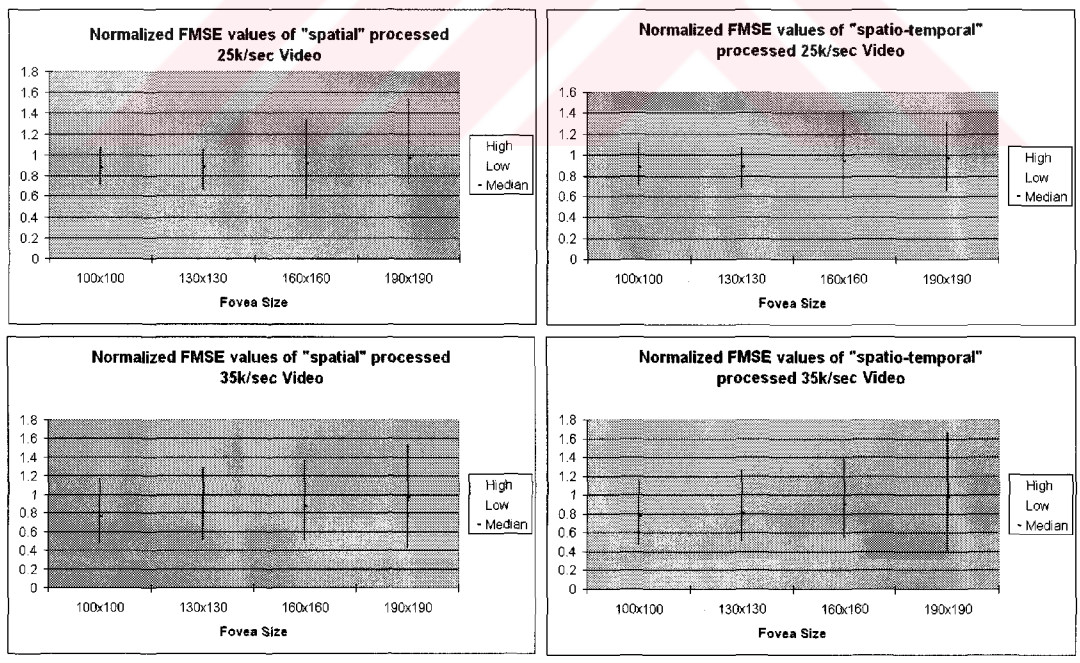

Figure 5.3. Normalized FMSE values of all encoded frames with respect to fovea sizes.

In Figure 5.3, the normalized FMSE values of encoded frames are grouped with respect to the encoding bit rate and the pre-processing mode. In each graph, the minimum, maximum and the median of the normalized FMSE values for each fovea

size is drawn. The first two graphs on the top of the figure show the spatial and spatio-temporal results at 25 $kbits/sec$ respectively. The graphs at the bottom contain the results for 35 $kbits/sec$.

As you can see from the top-left graph in Figure 5.3, we end up with 15 per cent FMSE improvement for 100×100 and 130×130 fovea sizes. As we increase the fovea size, the video quality of the fovea converges to the originally coded fovea quality (see the bottom-left graph in Figure 5.3). Interestingly, if we look at 35 $kbits/sec$ performances, the FMSE improvements can increase up to 22 per cent. However, there is not much added performance between just spatial and spatio-temporal coding schemes – possibly due to the temporal filtering selected for our particular application. The performance should improve with a more appropriately selected filter.

Similarly, the FSSIM values are shown in Figure 5.4. Averaged FSSIM values are greater than one, which is consistent with the FMSE outputs.

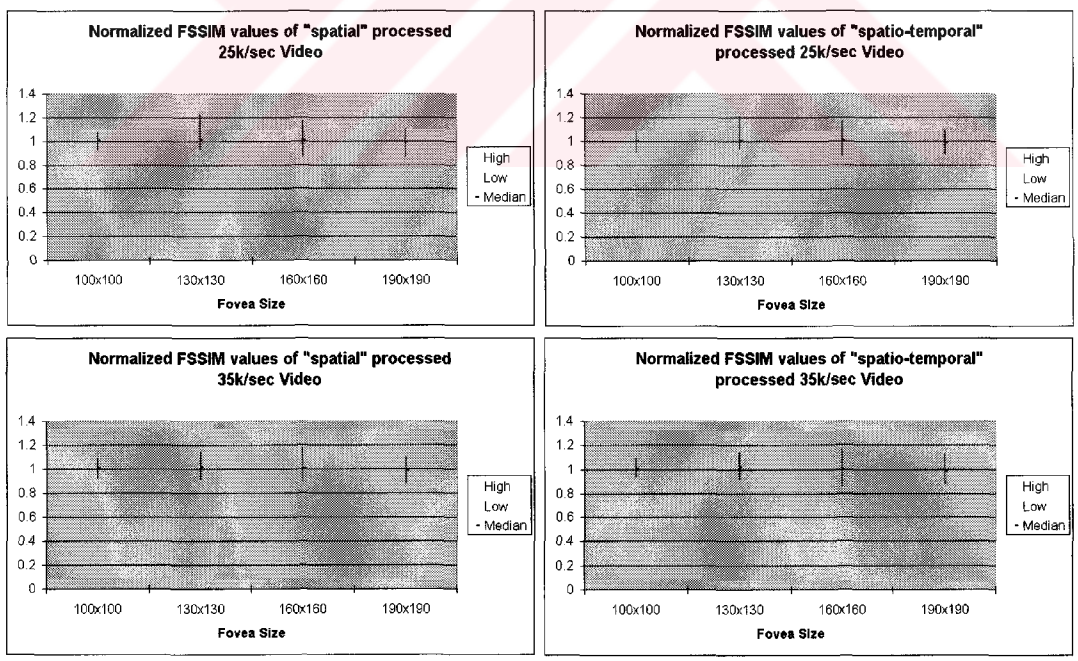

Figure 5.4. Normalized FSSIM values of all encoded frames with respect to fovea sizes.

We also computed the required processing overhead for the spatial and spatial-temporal coding schemes in order to check its suitability for real-time applications (on

Table 5.1. Computational overhead statistics for spatial and spatio-temporal pre-processing in milliseconds ($K = 3$)

Fovea size	Overhead (msec) for spatial coding	Overhead (msec) for spatial-temporal coding (K=3)
100 × 100	9.39	5.3
130 × 130	9.64	5.29
160 × 160	9.47	5.48
190 × 190	9.48	5.55

a Pentium II/1000 Mhz with 224 MB RAM). Each fovea size was considered seperately using randomly selected 200 frames. The results are as shown in Table 5.1.2. First of all, it is observed that the worst is about 10msec – which is quite acceptable with the frame rate of 25 frames/sec. As expected, the overhead is reduced considerably with the spatial-temporal coding.

6. CONCLUSION

In this thesis, we present a fovea based coding scheme for video streaming through low bandwidth networks that exploits two important aspects of human vision: fovea-periphery distinction and saccadic motion. Thus, each frame in the acquired image sequence has nonuniform sampling and consecutive saccadic images have temporal redundancy. Such a coding scheme is suitable for applications such as video broadcasting from attentive robot or cellular phones with video where the perceiver fixates on objects in a continual manner.

Our experimental results shows that compared to classical coding, spatial and spatio-temporal coding improve the transmission quality. Foveae with size 100×100 and 130×130 give better video quality at 25 $kbits/sec$ and 35 $kbits/sec$ encoding rate. Hence in pre-attention stage, we should take the resultant fovea size into account for a better quality within fovea region.

There is not a considerable video quality improvement between spatial and spatio-temporal processing, however when we compare the two approach with respect to the computational overhead, spatial processing requires a computational overhead twice more than the spatio-temporal processing. This overhead can be critical in real-time applications.

Next, the issue of foveation is studied. Foveation is based on an attention criteria - a mathematical function encoding the salient features in the incoming image. We consider visual primitives based on Cartesian and Non-Cartesian filters. Although these filters have been utilized in previous work for recognition purposes, they are used here for the first time in the pre-attention stage for foveation. Our preliminary studies indicate even with extremely limited learning, it is possible to have a 20 percent miss rate and about 20 percent false alarm rate. Since the experiments are done with a small set of training phase, we can easily increase the performance by creating a large training samples and readjusting the learning parameters. Hence, this work reveals

that these set of visual primitives can also be used in both pre-attention and attention stages.

For our future work, we will focus on improving the performance of the pre-attention stage. We will also work on methods for increasing temporal redundancy through careful generation of the saccadic movements. A recognition module can be added to the system in order to recognize the focus of attention subject, whom it is interacted with.

APPENDIX A: VISUAL PRIMITIVES

In this section, the construction of visual primitives is reviewed briefly. The reader is referred to [12] for all the details.

A.1. Cartesian Filters

The Cartesian filters $f_{cw} : SO(1) \times SO(1) \rightarrow [-1, 1]$ can be formulated in Equation A.1. Please note that $SO(1) =^{\triangle} [-\frac{\pi}{2}, \frac{\pi}{2}]$.

$$f_{cw}(x, y) = \cos(\omega \times (\alpha \times x + \beta \times y)) \tag{A.1}$$

In Equation A.1, $\alpha = \sin((c-1) \times \Pi/\Lambda)$, and $\beta = \cos((c-1) \times \Pi/\Lambda)$. The parameters c, and w are orientation, and frequency of the sinusoid respectively. By choosing $c = 1, ..., \Lambda$, and $w \in \{k \times \delta w \mid k = 1, ..., K\}$, the Cartesian filters look similar to those in [11]. In this work, we choose $\Lambda = 6$, $\delta w = 2$, and $K = 5$.

A.2. Non-Cartesian Filters

Non-Cartesian filters [11] are also a function of sinusoids, but the arguments of the sinusoid have nonlinear component. The Non-Cartesian filters can be grouped as concentric, polar, and hyperbolic filters. The concentric filters $f_{7w} : SO(1) \times SO(1) \rightarrow [-1, 1]$ can be modeled as $f_{7w}(x, y) = \cos(\omega \times (x^2 + y^2))$. By varying $w \in \{k \times \delta w \mid k = 1, ..., K\}$, we generate circular filters with different frequencies.

The polar filters $f_{7w} : SO(1) \times SO(1) \rightarrow [-1, 1]$ is defined as $f_{8w}(x, y) = \cos(\omega \times \arctan(y/x))$. Again by varying the frequency parameter $w \in \{k \times \delta w \mid k = 1, ..., K\}$, a set of circular filters are modeled.

In the final step, we generate two different forms of hyperbolic filters. First of

the hyperbolic filters $f_{9w}(x, y) = \cos(\omega \times \arctan(y/x))$ is $f_{9w}(x, y) = \cos(\omega \times (y^2 + x^2))$. The second set of hyperbolic filters $f_{10w}(x, y) = \cos(\omega \times \arctan(y/x))$ is obtained by rotating the $f_{9w}(x, y)$ function around the origin by θ degrees, which yields $f_{10w}(x, y) = f_{9w}(\cos(\theta) \times x + \sin(\theta) \times y, -\sin(\theta) \times x + \cos(\theta) \times y)$. By choosing $\theta = \pi/4$ and $w \in \{k \times \delta w \mid k = 1, ..., K\}$, we obtain another set of hyperbolic filters.

A.3. Filter Responses

The response of each filter is computed based on $2-D$ convolution results of the candidate fovea I_c^t and the filter kernel f_m. According to the experiments of Bozma et al. [12], the response of each filter Ω_m for $m = 1, ..., M$ – where $m = (c-1) \times K + k$, $w = k\delta w$, and $M = (c-1) \times K$ – can be calculated by using the filter output of f_{cw}. Experimentally, the best choice of Ω_m for Cartesian filters $f_{cw} \in F$, $c = 1, .., 6$ can be calculated by discarding the mean intensity level of the filter outputs. In our case, we define Ω_m for $m = 1, .., 30$ as the standard deviation of the convolution result $\Omega_m = stdev(f_{cw} \star I_c^t)$. Again, for Non-Cartesian filters (Circular, Polar, Hyperbolic, and Rotated Hyperbolic),experimentally the best choice of Ω_m is found as the greatest magnitude of the response. In our case, it can be formulated as $\Omega_m = max(f_{cw \star I_c^t})$, where $m = 31, ..., 50$.

APPENDIX B: VIDEO STREAMING PROTOCOLS

B.1. Real-Time Streaming Protocol (RTSP)

RTSP is a standards based streaming media protocol endorsed by the Internet Engineering Task Force (http://ietf.org), and it is defined in RFC:2326. The basic responsibility of RTSP is controlling and setting up the streaming media delivery. It can be on top of TCP or UDP. It is used between the streaming client and the media server. Default port for RTSP is 554. There are several messages like DESCRIBE, SETUP, PLAY, PAUSE, and TEARDOWN. Streaming client can query the properties of a content(URL,) using DESCRIBE message. The server responses the DESCRIBE message by sending a Session Description Protocol called SDP, which contains the port ranges, audio and video encodings and so on. Then SETUP message is used for initiating the data connections. Since the multimedia data is transported in a separate channel, the ports and transport protocol are negotiated in this step. Then client can send PLAY and PAUSE messages until the connection is closed with the TEARDOWN message.

B.2. Real-Time Transport Protocol (RTP)

RTP specifies a packet structure for packets carrying audio and video data, which is described in RFC:1889. It runs on top of UDP, so there is no guarantee of receiving RTP packets in sequence order, or recovery of packet loss, however since there is no need to channel setup procedure, and redelivery of all lost packets in TCP, it is better to use RTP for real time multimedia applications. Due to the nature of UDP datagrams, the application is responsible of handling packet loss, end to end jitter, delay jitter by using play-out buffer and some error-resilience methods.

APPENDIX C: TECHNICAL NOTES ON SOFTWARE COMPONENTS USED

C.1. Matrox Meteor- MIL 7.5

Matrox Meteor is a video capture card for CCD cameras. We used it's capture mechanism and A/D conversion. AVIExport module of MIL is used for the creating the Video Database. There is no other processing method used on that card. For more info, http://www.matrox.com/imaging/products/meteor2/- home.cfm and http://www.matrox.com/imaging/products/mil/home.cfm.

C.2. Intel Image Processing Library v2.5

The Intel Image Processing Library provides a set of highly optimized C functions that implement image processing functions on Intel architecture processors. In particular, functions responsible for holding the image data, changing the region of interest, performing 2D filters on this region, transformation of color space from RGB to YUV, YUV to RGB, selection of channel of interest. For more information, the homepage of IPL can be found at http://developer.intel.com/software/products/perflib/ipl/.

C.3. Intel Signal Processing Library v4.5

The Intel Signal Processing Library provides a set of highly optimized C functions that implement typical signal processing operations on Intel Architecture processors. It is the lowest level API for Signal Processing. We used several data conversion routines of this API. Moreover Intel Image Processing Library is also built on Intel SPL. For more information, please refer to http://developer.intel.com/software/products /perflib/spl/.

C.4. OpenCV Project

OpenCV means Intel Open Source Computer Vision Library. It is a collection of C functions and few C++ classes that implement some popular algorithms of Image Processing and Computer Vision. In this thesis, a routine that frames a region of interest region border with constant colors was utilized. Secondly, HaarObjectDetection module was used as a benchmark in the foveation part of the thesis. These routines are available from http://www.sourceforge.net/projects/opencvlibrary.

C.5. Microsoft DirectX 9.0

Microsoft DirectX is a set of low-level application programming interfaces (APIs) for creating games and other high-performance multimedia applications. It includes support for two-dimensional (2-D) and three-dimensional (3-D) graphics, sound effects and music, input devices, and networked applications such as multiplayer games. In the thesis work, DirectShow API which performs high-quality video and audio playback or capture is used during the application of processing or fovea determination on frames captured by APES, USB web-cameras, or any stored AVI files (it also supports Divx and Xvid video codecs).

C.6. Helix Producer 9.1

Helix Producer is used as video coding of APES robot. It is also an open source project and it uses Real Codec for encoding. We modified the producer such that we can feed it with our preprocessed video frames that is captured from Matrox Frame Grabber. More information can be found at https://helix-producer.helixcommunity.org/

C.7. APES API

High level C++ API that I developed by using all the tools above. This is the core API for the Vision and Video Streaming. Furthermore it contains the GUI API's.

C.8. Helix Streaming Server

An open source on demand and live streaming server that can stream video and audio to Real Clients. The clients could be run on PCs, handhelds, and Mobile phones. The server is running on Linux machine. It uses 554 port for RTSP protocol, UDP port range for RTP protocol, 8080 port for communication with the video source APES. More information can be found at https://helix-server.helixcommunity.org/.

APPENDIX D: APES APPLICATIONS USER GUIDE

D.1. Compiling and Running the Applications

The following software packages are required: Visual C++ 6.0 SP5 + Processor Pack and Microsoft Macro Assembler (MASM) 6.15. The Processor Pack is a free download from Microsoft's website, although it needs to be patched to SP5 first. It also includes MASM (ml.exe) as well. The latest version of the software can be accessed by a Concurrent Version System(CVS) Client for checking out the source codes from http://www.isl.boun.edu.tr . Most popular CVS client WinCVS can be downloaded free from http://ww.wincvs.org.

D.1.1. Prerequisites

Several libraries need to be installed before compiling the source code.

- **Matrox Imaging Library(MIL) 7.5:** MIL is needed for compiling the frame grabber module of Matrox Meteor capture-card. The frame grabber code only works with Matrox-Meteor Capture card, so if you do not run the application in real-time, its not mandatory to install MIL 7.5.
- **Intel Image Processing Library(IPL) v2.5** IPL v2.5 is used for low level image processing performance routines. The IPLHelper module uses this library. You can download IPL v2.5 from http://developer.intel.com/software/products/perflib/ipl/.
- **Intel Signal Processing Library(SPL) v4.5** SPL v4.5 is used for low level signal processing performance routines. The IPLHelper module uses this library. SPL v4.5 can be downloaded from http://developer.intel.com/software/products/perflib/spl/.
- **Microsoft DirectX 9.0** Microsoft DirectX 9.0 or later needs to be installed in order to compile the IOManager module. DirectX-SDK can be used for opening and seeking offline video files. It can be downloaded from DirectX SDK from

Microsoft's website.

D.2. Using The APES API

APES.dsw workspace file includes seven static library projects. The name of these projects are:

- AttentionInterface.dsp
- BiologicFilters.dsp
- Encoder.dsp
- Pre_processor.dsp
- IPLHelper.dsp
- IOManager.dsp
- SSIM.dsp

The contents of these projects are explained in the sequel.

D.2.1. AttentionInterface

AttentionInterface library implements the pre-attentive and attentive stages. The list of important files and their basic functionalities is shown in Table D.1.

D.2.2. BiologicFilters

The Cartesian and Non-Cartesian Filters are implemented in this library. The grabbed frames are also encapsulated with an object in this library. Here is a list of important files and classes of BiologicFilters library:

D.2.3. Encoder

The encoder library for encoding the video frames is developed in RealMedia codec format. This encoder was modified such that instead of taking an offline video

Table D.1. Brief description of important files in Attention Interface library

Attention.cpp	An abstract class for pre-attention. By deriving this class, any pre-attention mechanism can be implemented. All derived classes should implement locatenextfovea(..) method.
HaarFaceDetect.cpp	Implementation of Cascaded Adaboost method using Extended Haar Features. It uses the current frame as an input and returns the foveae list, which contains the coordinates of the face candidates.
ProposedFaceDetection.cpp	This file contains the proposed face detection algorithm with trained neural network. There is also an implementation of simple weighting of the filter responses.

file, it can also be input realtime video frames. Hence, spatio-temporally processed frames coming from APES can be encoded. Here are some important parts of the Encoder library:

D.2.4. Pre_processor

Pre_processor library is responsible of processing the raw input frame Spatially or Spatio-Temporally with given parameters. Hence the video quality of the fovea region considerably improved.

D.2.5. IPLHelper

This library contains high level image processing functions. It uses the Intel performance primitives for these functions. Here is some important routines that I developed.

Table D.2. Brief description of important files and classes in BiologicFilters library

ApesFrame.cpp	All raw video frames in RGB24 format are encapsulated with ApesFrameObject. You can define region of interest area for an ApesFrame. Then all the operations on this object is affected only on the defined region of interest. If the ROI is null, then the operations are made on whole frame.
Fovea	Structure that holds the bottom left coordinates, width, and height of focus of attention area.
CandidateFovea.cpp	This class holds the Biologic Filter Responses of each fovea candidate. Note that fovea candidates are created within the visual field I_v of the frame with an overlapping factor of of. A CandidateFovea object is also capable of retrieving the index of maximum candidate fovea response, which will be the next fixation fovea.
FilterKernel.cpp	An abstract class for Biologic filters that contains two-dimensional filter kernels.
CartesianFilter	Implementation of the FilterKernel class. It initializes all Cartesian Filter kernels.
ConcentricFilter	Implementation of the FilterKernel class. It initializes all Concentric Filter kernels.
PolarFilter	Implementation of the FilterKernel class. It initializes all Polar Filter kernels..
HyperbolicFilter	Implementation of the FilterKernel class. It initializes all Hyperbolic Filter kernels..

D.2.6. IOManager

If the AttentionInterface is the brain of the application, then IOManager will be the hearth of the application. Because all low level video frame capture and offline video file access functions are completed using this library. In this table, I will explain the core functionalities of the C++ files.

Table D.3. Brief description of important file and classes in Encoder library

encoder.cpp	Responsible for the encoding job. It contains crucial classes and routines for video encoding and streaming.
CEncoderApp	The input frame dimensions and target output bitrate could be configured via this object. All of the controls of the encoding engine are done via CEncoderApp.
dataDestinationServer	It contains the IP and port number of the streaming server. CEncoderApp uses this object to direct the RTP video stream to the target streaming server.
dataInputVideo	The input video frame formats of the video encoder is stored in dataInputVideo. Frame rate, width, height, and color format are some of the properties of dataInputVideo.

Table D.4. Brief description of important file and classes in Pre_processor library

preprocessor.cpp	This is the C++ file that is responsible for preprocessing the video frames in order to use the available band-with more efficiently.
Spatial	$5x5$ box blur filter is applied to the outside of the fovea region.
SpatioTemporal	In addition to the Spatial coding, the refresh rate of the consecutive periphery region of video frames can be tuned in SpatioTemporal routines.
NoPreProcessor	There is also a possibility to turn off the preprocessing. It is useful if we want to compare the performance of the quality improvements of Spatial and SpatioTemporal coding with the case of no preprocessing outputs.

D.2.7. SSIM

In this library there is a list of routines for calculating the statistical measures and some video quality metrics. Here is some important functions in SSIM library:

Table D.5. Brief description of important functions in IPLHelper library

IPLHelp_BGR_to_YCrCb	It is a color space conversion routine that converts pixel values from BGR space to YCrCb space. The mathematical formula of the conversion is: $Y = 0.3 \times R + 0.6 \times G + 0.1 \times B$ $U = B - Y$ $V = R - Y$ $Cb = 0.5 \times (U + 1)$ $Cr = V/1.6 + 0.5$.
IPLHelp_8U_to_FP	Converts 8 bit unsigned pixel value to 32 bit floating precision value. It is useful when you operate floating point operations on the raw frames.
IPLHELP_calculate_Stdev	This function calculates the standard deviation of the pixel values.
IPLHelp_CopyImage	This copies the source image pixels to destination image. The interesting utility is the copy process is defined only the region of interest areas.
IPLHelp_compute_filter_responses	This method convolves the candidate fovea with the Biologic Filter-bank and computes the filter responses.

Table D.6. Brief description of important files in IOManager library

FrameGrabber.cpp	This is the file that enable us to digitize the CCD camera output and make it publicly available to the applications. I used a callback function for implementing this utility. When a new frame is captured, then it is directly accessible to the applications via a shared memory.
SaveAvi.cpp	This class can save any raw video frame sequence in uncompressed AVI file format. It is useful to save the outputs of attentional sequences for conducting experimental outputs.
DirectXHelper.cpp	This class deals with the low level DirectShow API. By using this class a complete DirectX Graph can be created.
VideoInput.cpp	This class is developed using DirectXHelper routines. You can open and seek any video files with this class. as well as querying the index of a particular video frame. The frame is returned in in 24bit RGB pixel format.

Table D.7. Brief description of important functions in SSIM library

calculateMSE	This function calculates the Mean Square Error and Foveal Mean Square Error of an encoded video frame.
calculatePSNR	This function calculates the Peak Signal to Noise Ratio and Foveal Peak Signal to Noise Ratio of an encoded video frame.
calculateSSIM	And this routine calculates the Structural Similarity Metric and Foveal Structural Similarity metric of an encoded video frame.

REFERENCES

1. Akins, K., (editor), Perception, *Oxford University Press*, pp. 290-316, 1996.

2. Ballard, D. H. and C. M. Brown, "Principles of Animate Vision", *CVIP:Image Understanding*, Vol. 56, July 1992.

3. Itti, L. and C. Koch, "A Saliency-Based Search Mechanism for Overt and Covert Shifts of Visual Attention", *Vision Research*, 2000.

4. Gouras, P. and C. H. Bailey, "The Retina and Phototransduction", *Principles of Neural Science Elsevier*, 1986.

5. Kowler, E., "Eye Movements", In S. M. Kosslyn, D. N. Osherson, (editors), *Visual Cognition*, MIT Press, pp. 215-266, 1995.

6. Basu, A. and K. Wiebe, "Improving Image and Video Transmission Quality over ATM with Foveal Priorization and Priority Dithering", *Pattern Recognition Letters*, Vol. 22, 2001.

7. Reeves, T. H. and J. A. Robinson, "Rate Control of Foveated MPEG Video", *CCECE*, 1997.

8. Geisler, W. S. and J. S. Perry, "A Real-time Foveated Multiresolution System for Low-bandwidth Video Communication", *SPIE Proceedings: Human Vision and Electronic Imaging*, Vol. 3299, pp. 294-305, 1998.

9. Grosso, E., R. Manzotti, G. Sandini and R. Tiso, "A Space-variant Approach to Oculomotor Control", *Proceedings of International Symposium on Computer Vision*, pp. 509-514, 1995.

10. Kowler, E., (editor), "Eye Movements and Their Role in Visual and Cognitive Processes", *Elsevier*, 1990.

11. Gallant, J. L., H. C. Nothdurft and D. C. Van Essen,"Two-Dimensional and Three Dimensional Texture Processing in Visual Cortex of the Macaque Monkey", *In Early Vision and Beyond*, T.V. Papathomas, C. Chubb, A. Gorea and E. Kowler, (editors), MIT Press, pp. 89-98, 1995.

12. Bozma, H. I., G. Cakiroglu and C. Soyer, "Biologically Inspired Cartesian and Non-Cartesian Filters for Attentional Sequences", *Pattern Recognition Letters (SCI)*, Vol. 24/9-10, pp. 1261-1274, June 2003.

13. Bozma, H. I. and G. Cakiroglu, "Dynamic Integration for Scene Recognition Using Complex Attentional Sequences", *Proceedings of Intelligent Autonomous Systems*, Amsterdam, 2004.

14. Hjelmas, E. and B. K. Low, "Face Detection: A Survey", *Computer Vision and Image Understanding*, Vol. 83, pp. 236-274, 2001.

15. Ahuja, N., D. J. Kriegman and M. H. Yang,"Detecting Faces in Images: A Survey", *IEEE Transactions in Pattern Analysis and Machine Intellgence*, Vol. 24/1, pp. 34-58, January 2002.

16. Comaniciu, D. and V. Ramesh, "Robust Detection and Tracking of Human Faces with an Active Camera", *IEEE*, 2000.

17. Huang, T. S. and G. Yang, "Human Face Detection in Complex Background", *Pattern Recognition*, Vol. 27, No. 1, pp. 53-63, 1994.

18. Burl, M.C., T. K. Leung and P. Perona, "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching", *Proc. Fifth IEEE Intl Conf. Computer Vision*, pp. 637-644, 1995.

19. Garcia, C. and G. Tziratis, "Face Detection Using Quantized Skin Color Regions Merging and Wawelet Packet Analysis", *IEEE Transactions on Multimedia*, Vol. 1/3, pp. 264-277, September 1999.

20. Cootes, T. F., A. Lanitis and C. J. Taylor, "An Automatic Face Identification System Using Flexible Appearance Models", *Image and Vision Computing*, Vol. 13, No. 5, pp. 393-401, 1995.

21. Pentland, A. and M. Turk, "Eigenfaces for Recognition", *J. Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86, 1991.

22. Baluja, S., T. Kanade and H. Rowley, "Neural Network-Based Face Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 23-38, 1998.

23. Freund, R., F. Girosi and E. Osuna, "Training Support Vector Machines: An Application to Face Detection", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130-136, 1997.

24. Chaudhuri, S., U. Desai, J. Karlekar, K. Kumar, R. Manivasakan, M. Patil, P. Poonacha, and A. Rajagopalan, "Finding Faces in Photographs", *Proc. Sixth IEEE Intl Conf. Computer Vision*, pp. 640-645, 1998.

25. ITU-T Recommendation H.263, "Video Coding for Low Bit Rate Communication", March 1996.

26. Bozma, H. I., R. Civanlar and C. Dikici, "Fovea Based Real-Time Video Processing and Streaming", *11.st Signal Processing Conference*, June 2003.

27. He, Z. J. and K. Nakayama, "Attention to Surfaces: Beyond a Cartesian Understanding of Focal Attention", *In Early Vision and Beyond*, T.V. Papathomas, C. Chubb, A. Gorea and E. Kowler, (editors), pp. 69-77, 1995.

28. Sagi, D., "The Psychophysics of Texture Segmentation", *In Early Vision and Beyond*, T.V. Papathomas, C. Chubb, A. Gorea and E. Kowler, (editors), pp. 69-77, 1995.

29. Connor, C. E., J. L. Gallant, J. W. Lewis, S. Rakshit and D. C. Van Essen, "Neural

Responses to Polar, Hyperbolic and Cartesian Gratings in Area V4 of the Macaque Monkey", *Journal of Neurophysiology*, Vol. 76, No. 4, pp. 2718-2739, 1996.

30. Fausett, L., "Fundamentals of Neural Networks", *Prentice-Hall*, 1994.

31. Haykin, S., "Neural Networks: A Comprehensive Foundation", *Prentice-Hall*, 1994.

32. Lippman, R. P., "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4-22, 1987.

33. Jones, M. J. and P. Viola, "Rapid Object Detection Using a Boosted Cascade of Simple Features", *IEEE CVPR*, 2001.

34. Lienhart, R. and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection", *ICIP*, 2002.

35. Bozma, H. I., Y. Istefanopulos and C. Soyer, "Apes: Actively Perceiving Robot", *Proceedings of IEEE/RSJ International Conference on Robots and Systems, Lausanne*, October 2002.

36. Bozma, H. I., Y. Istefanopulos and C. Soyer, "Attentional Sequence Based Recognition: Markovian and Evidential Reasoning", *IEEE Transactions on Systems, Man and Cybernatics,* Vol. 33, No. 6, pp. 937-950, 2003.

37. Helix Encoder, *http://www.helixcommunity.org*, 2004.

38. Henderson, J. M. and A. Hollingworth, "Eye Movements During Scene Viewing: An Overview", *Eye Guidance While Reading and While Watching Dynamic Scenes*, pp. 269-295, 1998.

39. VirtualDub, "Open-source Video Processing and Capture Project", *http://virtualdub.sourceforge.net/*, 2004.

40. Alp, U., H. Ayaz, H. I. Bozma, R. Civanlar, C. Dikici, and M. Karadeniz, "Remote

Control of a Robot Over the Internet", *11.st Signal Processing Conference*, June 2003.

41. Bovik, A. C., H. R. Sheikh, E. P. Simoncelli and Z. Whang, "Image Quality Assessment: From Error Measurement to Structural Similarity", *IEEE Transactions on Image Processing*, Vol. 13, No. 1, January 2004.

42. APES Video Database, *http://www.isl.ee.boun.edu.tr/Apes/VideoDatabase.html*, 2004.

43. Takacs, D. and H. Wechsler, "A Dynamic and Multiresolution Model of Visual Attention and Its Application to Facial Landmark Detection", *Computer Vision and Image Understanding*, Vol. 70, No. 1, pp. 63-73, April 1998.