Visual Attention-based Small Screen Adaptation for H.264 Videos

by

Abir Mukherjee

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical & Computer Engineering

Waterloo, Ontario, Canada, 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We develop a framework that uses visual attention analysis combined with temporal coherence to detect the attended region from a H.264 video bitstream, and display it on a small screen. A visual attention module based upon Walther and Koch's model gives us the attended region in I-frames. We propose a temporal coherence matching framework that uses the motion information in P-frames to extend the attended region over the H.264 video sequence. Evaluations show encouraging results with over 80% successful detection rate for objects of interest, and 85% respondents claiming satisfactory output.

Acknowledgements

I would like to thank my parents for their love and support and for making me what I am today. I am grateful to my supervisor, Prof. En-hui Yang for being supportive and patient throughout my program, and always mentoring me when the need arose. His persistence was a key to this thesis.

A special thanks to my friends and research mates, especially Xiang Yu, Lin Zheng and Sarvagya Upadhyay for their timely comments and suggestions.

Dedication

To Juthi for brightening up my life;

and to Raiju and Krithika for food, shelter and affection.

Contents

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Encoding Video and the H.264 standard 2

1.2 Need for Video Adaptation 3

1.3 Role of Visual Attention 4

1.4 Document Organization 6

2 Related Literature 7

3 The Display Adaptation Algorithm 11

3.1 Background 12

3.1.1 Visual Attention Model 13

3.2 The Dynamic Small-Screen Adaptation (DSSA) Algorithm 16

3.2.1 Assumptions made in the Algorithm 16

3.2.2 Determining Attended Region in I-frames 17

3.2.3 Determining Attended Region in P-frames: Temporal Coherence . 19

3.2.4 Post-processing and Display 21

4 Experimental Results and Illustrative Examples 24

4.1 Experimental Setup 24

4.1.1 Simplifications adopted for ease of implementation 25
4.2 Simulation Results 26
4.2.1 Benchmark test 26
4.2.2 DSSA Algorithm results 27

5 Conclusion **31**
5.1 Applications 31
5.2 Limitations 32
5.3 Future Work 32

References **34**

List of Tables

4.1 Estimating attended region: Benchmark 26
4.2 Estimating attended region: DSSA 27

List of Figures

1.1 Demonstration of ideal system behaviour: (a) Original frame at 320x240 resolution (b) Cropped out RoI of size 160x120 (c) Resized frame at 160x120 resolution 6

3.1 General architecture of the saliency-based visual attention model (adapted from [16]) 14

4.1 Frame 42 and Frame 80 of the *Coastguard* sequence shows the DSSA switching from following the small boat to the large coastguard boat. Cropped frames are scaled to twice their height and width. 29

4.2 Frame 72 and Frame 132 of the *Mobile* sequence shows the DSSA following the engine and red ball initially, but distracted by the background later. Cropped frames are scaled to twice their height and width. 30

Chapter 1

Introduction

The world is making rapid strides in communication, and this is changing the way we interact with one another. Digital media is at the forefront of this ongoing change. Multimedia has become a part of our daily lives in more ways than we can imagine. People communicate not only through text-based emails but also through audio and video messages and clips. With videos becoming increasingly popular with every passing day, there is a need to make this media more robust and accessible to a variety of users across different platforms.

Multimedia applications are becoming more diverse and are shared over communication infrastructure comprising of different underlying networks and protocols. Hence we need to inter-network multimedia communications over heterogeneous networks. In a network where end users connect to a video source through links of different capacities, the source usually adjusts the bandwidth for the compressed video to meet the available capacity on the most stringent link. In addition to this, end users often use different devices such as desktops, cellular phones, handheld computers etc. for video communication. Since most handheld devices have limited computing and display capabilities, the high quality video encoded earlier may have to be converted into one of lower quality for display on such devices. Furthermore, as the number of coding standards such as MPEG-2, MPEG-4, VC-1, H.261, H.264 etc. increases, there is a growing need to convert between videos coded through different standards. Video transcoding provides techniques to solve these problems [1, 41]. The present work has been developed to process videos encoded with the H.264 video standard. We take a closer look at this in the next section.

1.1 Encoding Video and the H.264 standard

Broadly speaking there are two approaches to video coding, viz. block-based and object-based. In the block-based approach, each frame is divided into a number of blocks, and motion estimation and compensation performed at the block level. These blocks are usually of a fixed size, though variable size blocks might also be used, as they can give an improved match for the currently selected region of the frame. In actual practice, a frame is divided into macro-blocks (MBs) and these are subdivided into blocks. In case the encoder uses different block sizes, there is a restriction on the permissible sizes - this reduces computational complexity, but at the expense of video quality. All major coding standards like MPEG-1, MPEG-2 and H.264 use the block-based coding approach. The MPEG-4 standard goes beyond the block-based coding approach by introducing the concept of a video object layer (VOL), to allow object-based video encoding [32]. In object-based video encoding, the objects of interest in the video are marked at the video source prior to encoding. Since the encoder knows the region that comprises of the object, it ensures that this region is encoded for the best possible quality. To the best of our knowledge, this scheme is unique to MPEG-4 and, as such, is incompatible with other video coding standards. Object-based coding makes intuitive sense, since in everyday life humans don't see scenes as blocks but rather in terms of the objects that make up the scene. In every scene there are objects in the foreground that are in focus while the remaining objects constitute the background. Some of these objects can be identified in images through algorithms that find regions of interest (RoI). Such algorithms typically use the brightness, colour and texture information of the scene along with the connectivity of the region to select the RoI. However, marking objects on the video frames requires selecting key frames or index frames in the video and marking the object or feature on each of them. This is most accurate when done by a person, but that is an extremely time-consuming process. Even otherwise, this process is computationally intensive.

Video and image processing based on objects of interest and RoI are nearly always done in the spatial (or pixel) domain. Spatial domain video processing is computationally intensive as it deals with a large amount of data. A more efficient approach is to design a system to work with compressed (transform domain) video. Since videos extensively deal with moving objects, motion is a useful criterion to identify relevant objects. Object tracking or segmentation in video finds applications in video surveillance, video indexing etc. The algorithms that operate in the transform domain utilize two features of a MB, viz. motion vector (MV) and transform coefficients. MVs are obtained through motion compensation between current frame and its reference frame(s) on a block-by-block (or MB-by-MB)

basis. An MV gives the offset of the current block to the matching block in the reference frame. It gives information about the temporal correlation between the two frames. On the other hand, the transform coefficients contain the image information. The content of the transform coefficients differ depending on the type of block that is encoded. A block may be inter-coded or intra-coded. An inter-coded block is one that is predicted from a reference frame. The transform coefficients of these blocks contain residues of the motion compensation. On the other hand, intra-coded blocks are predicted either from other blocks in the current frame or else encoded as-is. The transform coefficients of intra-coded blocks carry the transformed signal of the original image. Therefore, these block transform coefficients can be used to reconstruct the DC image.

However, the H.264 coding standard (alternately called AVC: Advanced Video Coding, or H.264/AVC) employs several new coding tools and provides a different video format, which makes working on the compressed video a challenging task [32, 35]. While the earlier MPEG standards employed a discrete cosine transform (DCT), H.264 uses a transform that is similar but uses only integer operations. For convenience, we refer to this transform as DCT in the rest of this document. Furthermore, very little literature is available about work done on H.264 compressed video analysis. In H.264, the intra-coded block is spatially intra-predicted from its neighbouring pixels. So, the DCT coefficients carry spatial prediction residue information. H.264 also supports variable block-size motion compensation. A MB may be partitioned into several blocks and have several MVs with varying block size. This is in contrast to the MPEG standard, which has a regular block size. H.264 has been steadily gaining popularity and coming into widespread use. It is now the codec of choice for applications ranging from television broadcast to mobile videos. Now that we have seen what video encoding is about and got an idea of the H.264 video standard, let's take a look at why we need to adapt videos and how it can be done.

1.2 Need for Video Adaptation

As we noted at the beginning of this chapter, contemporary viewers watch videos on different kinds of devices, some of which have lower processing and display capabilities. Today's handheld devices provide the option to watch video, but some details might be lost due to the smaller screen size. Let's take a soccer game for instance. The soccer ball that is clearly visible on a regular size television screen may be rendered so small in the display of the handheld device that it is invisible to the human eye. This happens because the video stream available to the handheld is of lower resolution than the one on television. Owing

to their limited computing capability, the handheld devices usually have a downsampled version of the original video stream. Downsampling is also required when video has to be streamed over a network with constrained bandwidth. Since, downsampling is not selective about particular regions of the video, the output stream is uniformly degraded. However, from the viewer's perspective, if the relatively small soccer ball - the object of interest - is downsampled, there is a great loss in video quality.

In order to overcome this drawback, a video clip needs to be adapted for display on such handheld devices. A simple method for adaptation would be to show only the most relevant part of the original video on the screen. In other words, a video clip can be cropped around the RoI so that it fits the handheld device's smaller screen. Unfortunately, cropping algorithms do not account for object motion. Hence, the cropped region needs to be adjusted in such a manner that it always includes the object. [3, 31, 47] propose techniques for automatic RoI determination in videos. However, cropping and tracking algorithms work wholly in the spatial domain and thus require large processing power for computations.

Thus, there is a need to develop a framework by which videos can be intelligently adapted for viewing on small form factor devices. With H.264 being slated as the future standard for all video and the rising popularity of portable devices, such a scheme will enable users to watch videos on their devices without severe quality degradation. A key requirement for such visual content adaptation is that the output fits human perception. This requires that we identify regions that receive maximum 'visual attention'. The concept of visual attention is covered in the next section.

1.3 Role of Visual Attention

The objects which form the focus of attention of a viewer are referred to as Attention Objects (AOs). If a video clip is adapted to a smaller screen in the manner suggested at the end of the previous section, we would want to retain the AOs from the original video sequence. One of the ways to crop a video intelligently is to do it manually, on a frame-by-frame basis. Unfortunately, this method is not feasible owing to the sheer volume of work. For instance, a 5-minute video clip running at a regular 30 frames per second will have 9000 frames! Even if the person skips some frames, finding the object of interest in a soccer game or in a short movie is a daunting task. If we can find a way to identify AOs in a scene, without user intrusion, it will save a lot of time. This is where visual attention-based processing comes in. The human vision system responds more to certain image features and less to others [7, 17]. Humans are known to be sensitive to contrasts

and edges in a scene; they are more sensitive to luminance (brightness) than to colour and they focus more on the centre of the scene than on the surroundings. By modelling a system closely to the human vision system, we can get a good idea of what features in a scene are likely be of most interest to a viewer. In [16], Itti and Koch developed a model closely based on the human vision system to identify AOs in images. They use low-level image features, viz. intensity, colour and orientation to process into feature maps, which are further processed and combined into a saliency map. Regions of the saliency map with high values correspond to attended regions of the image. It has proved to be successful in identifying AOs even in noisy images, and is a promising candidate for such an application.

One limitation of the Itti and Koch model is that it applies to static images but not to videos. In order to apply it to videos, we combine this model with the motion information already present in the encoded video bitstream. Consider a video sequence consisting of I-frames and P-frames. The intra-coded I-frames have no motion information, but they can be decoded without having to resort to prediction from other frames. Consequently, I-frames are good candidate frames in which to search for AOs. The P-frames in the video are inter-coded, which means that they are predicted from I-frames or other P-frames. These frames are predicted through an extensive process of motion estimation and compensation. The motion information present in the P-frames can be used to determine the region where the AO is located in those frames. Once the region containing the AO is identified across the entire video sequence, it is extracted for display on the smaller screen.

To demonstrate the behaviour of this system, let's revisit the example of the soccer game from the previous section. Figure 1.1 below shows the results of resizing and cropping a frame from a soccer game. The original frame has a resolution of 320x240 pixels. This frame (figure 1.1(a)) shows a number of players on the field, but the AOs in the frame are the soccer ball (including the really close players) and the score. We have bounded these with a white rectangle in the figure. When the frame is resized to a quarter of the initial size (160x120 pixels), neither the ball nor the score is legible (figure 1.1(c)). They have been rendered too small to be seen. However, when the region around these AOs is cropped out (160x120 pixels), both the score and the ball can be seen clearly. This is shown in figure 1.1(b).

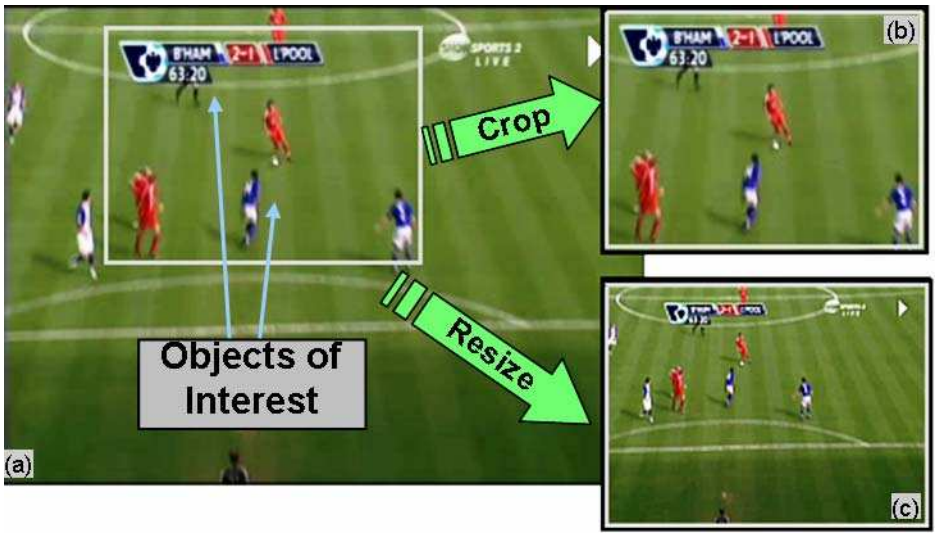

Figure 1.1: Demonstration of ideal system behaviour: (a) Original frame at 320x240 resolution (b) Cropped out RoI of size 160x120 (c) Resized frame at 160x120 resolution

Figure 1.1 demonstrates one example where video adaptation can be successfully applied. Such adapted video can also be re-encoded into smaller video clips that can be transferred over bandwidth constrained networks. We will discuss the design and implementation of this system in the chapters that follow.

1.4 Document Organization

This document details the thesis work and consists of five chapters. The next chapter, Chapter 2, is a survey of existing literature related to the present work. Chapter 3 describes the design of the framework we developed for small screen adaptation using a visual attention model; and in Chapter 4, we present some examples that illustrate the capabilities and limitations of the present work. Finally, we round up the document with conclusions and a discussion on the future directions of this work in Chapter 5.

Chapter 2

Related Literature

The previous chapter gave an overview of H.264 video coding, and established the need for video adaptation in a world of increasingly varied client devices. We also discussed how visual attention analysis is crucial for effective video adaptation. The present chapter discusses existing literature on compressed domain video processing, visual content adaptation and visual attention analysis. We start with work on human vision and perception.

The human vision system (HVS) is a complex network of neurons and light sensitive receptors. Over the years, a lot of research has gone into determining how humans *see* and what attracts their attention to a scene. Human perception first picks the regions of the scene that stimulate the HVS and then interprets the remaining scene. These regions usually correspond to prominent objects in images or action in video sequences. Psychology studies suggest that the HVS perceives external features separately [37] and is sensitive to the difference between the attended region and its neighborhood [10]. The collective results of such research provide us with a set of features [7, 17] that are widely accepted to, so to speak, *grab human attention*. These include colour, orientation, size, motion, lustre and shape to name a few. This has led to work based on the detection of feature contrasts to trigger the HVS [16, 44]. All these literature use visual attention models to determine the attended region(s) in images.

Itti et al proposed one of the earliest works in visual attention detection by utilizing contrasts in color, intensity and orientation of images [16]. They used these low-level features from digital images to create feature contrast maps and further process them into a saliency map. Walther and Koch extended this idea [44] to detect attended regions of any size around the salient points in the saliency map. Milanese demonstrated a similar bottom-up approach for a salient region detection framework [27]. Chen et al [2] used the saliency map generation methods proposed in [16] to determine perceptually important regions in

an image that hold Attention Objects (AOs). After identifying the AOs and the associated region(s) in the image, their branch-and-bound algorithm determines the optimal set of AOs to be included in the final image. The method is shown to be efficient and has provision to identify faces and text and give them priority over other features. The output image size can be altered to fit space constraints such as on a web site or for a thumbnail image. Cheng et al used intensity, color and motion features to determine the region-of-interest (RoI) in a video sequence, based on aesthetic principles [3]. They employed a shot detection algorithm on the source video to form clusters of frames, and then applied saliency-based attention processing to each cluster. Their subjective tests show consistently good results across different kinds of videos. Zhai and Shah utilized a temporal attention model based on point correspondence and a spatial attention model based on color contrasts, and combined them into a spatiotemporal saliency map to detect the attended region [47]. Ma et al also developed a user attention model [23, 24] for video summarization. In [45], Wang et al propose the wavelet-based foveation scalable video coding (FSVC) algorithm that uses a foveation-based HVS model to determine visually important components in the video sequence. They proposed an adaptive frame prediction scheme for encoding and decoding videos that allows good quality in rate scalable video coding systems. Moreover, this flexible scheme can be adapted to different video applications including telemedicine and video communication over heterogeneous networks.

Video object segmentation is an extension of image segmentation to videos, and deals with extraction of RoI's from video. Since our framework pertains to identification and extraction of objects of interest, relevant literature is found in this area as well. In 1997, Yining Deng and B.S. Manjunath proposed a segmentation method called JSEG [9] in which they quantized the colours present in an image into classes and use the class labels to generate a class map for the image. The class map contains colour and texture information which is used to calculate a local parameter, J and this is further saved as a J-image. This J-image is then used for spatial segmentation. A similar approach is applied to videos sequences. Videos are partitioned into shots - sequences of continuous action - and then objects are segmented and tracked across frames. The region tracking feature is embedded in the segmentation algorithm and gives robust results. One of the major limitations of this scheme is over-segmentation due to varying illumination. In [8], they used the colour, texture and motion information from MPEG compressed videos to create an indexing scheme to enable fast retrieval. The fairly simple approach requires partitioning the video into shots as in [9] and then using the colour histogram to get colour information and generate labels. They also use Gabor texture features and developed a novel approach to tracking motion using the motion histogram. These three features are used to classify

videos and for content-based search and retrieval. The system performs well but is limited by the use of global low-level visual features. Localized object feature representation is expected to yield better results.

The foregoing work use source video sequences and images for pixel domain processing. However, visual content is always transmitted in an encoded bitstream. While it is possible to decode and process in the pixel domain, the latter is complex and time-consuming. It is desirable to use the bitstream information to adapt content efficiently. Zeng et al [46] employ a block-based Markov Random Field (MRF) to segment moving objects from the MV field obtained from the compressed bitstream. The method segments moving objects against a stationary background, at real-time processing speeds with over 80% recall and 40% precision. Liu et al [22] proposed a scheme to use watershed filling on a normalized MV field to segment a frame into homogeneous motion regions. This is followed by a binary partition tree (BPT) scheme for the merging process. The system demonstrates over 85% recall and 60% precision for the tested sequences with real-time processing. Both of these systems, however, are susceptible to errors in the presence of shadows or objects moving at nearly the same speed.

Saliency detection is an important attentional mechanism and is largely determined by what our senses perceive. This was used for motion detection and tracking in [36], where Tian and Hampapur detect salient motion for video surveillance in three steps: first, the two-dimensional optical flow of the image is computed using the Lucas-Kanade method. Second, a temporal filter is applied to the difference images to filter out noise - which includes regions not moving in the same direction over a group of eleven frames. Lastly these are combined and a region growing algorithm gives the region of interest. This system is able to detect motion against a complex moving background and can be used for real-time surveillance operations. In [29], Sonia Mota et al presented a perception-based moving object segmentation scheme that uses Reichardt motion detectors to characterize the motion in the scene. This results in a noisy saliency map, which is further processed by a neural structure to select independent moving objects in the scene by picking a cluster of pixels moving coherently, with approximately the same velocity. In case of more than one moving object, the system only works when the relative speed(s) between the moving objects is large.

In the present work, we use the attended region detection proposed by Walther and Koch [44] and combine it with a homogeneous motion region detection algorithm to identify the salient moving feature(s) in H.264 videos. This framework is built into the H.264 decoder JM (joint model) version 13.2. The decoder finds the best attended region that matches the

specified output frame size from the input H.264 video sequence and extracts it for display. The next section covers the problem formulation and describes the proposed algorithm.

Chapter 3

The Display Adaptation Algorithm

So far, we have established the necessity of a video adaptation system that operates on compressed domain video bitstreams. We also saw some related literature on adaptation and visual attention based processing. The HVS can not identify objects below some critical resolution. However, there seems to be no fixed point for this critical resolution: it varies with the type of object we view. For instance, humans can distinguish a face from other features relatively easily, and can also identify the person from his/her facial features at quite a low resolution. At the same time, they can not identify the letters of the alphabet with equal ease - humans need a relatively larger resolution to read text.

Let's revisit the example from Section 1.2. Consider a person watching a soccer game on a big screen television. The soccer ball and the player(s) around it are usually in focus and located near the centre of the screen. Along with this, a substantial part of the field is in view and there are other players at different parts of the field. In addition to the scene on the field, the current score and play time is displayed in a text box at the top of the screen. If this video is resized to fit a small screen, the soccer ball may be represented by a small dot and the text may be rendered illegible at low resolution. Figure 1.1 shows a frame from such a soccer video.

The original frame is 320x240 pixels in size (fig 1.1 (a)). When this is scaled down to 160x120 pixels, the soccer ball is no longer clearly visible. The scoreline at the top is not readable either (fig 1.1 (c)).

One can think of a few methods to avoid this issue. One option could be to decide a minimum permissible resolution beforehand. If calculations indicate that the resized video will to be smaller than the minimum permissible size, it can be cropped around the edges. The drawback in this method is that it assumes *a priori* knowledge of the video content,

that allows us to select such a resolution. In a practical scenario such information is unavailable, and an alternative would be to crop the video to retain the most relevant section. As we see in Figure 1.1 (b), when the original frame is cropped to 160x120 pixels, their original resolution is preserved, and both the soccer ball and the scoreline are seen as clearly as in the original frame. However, it is not straightforward to crop a video clip in such a manner that only the relevant sections are preserved. To the best of our knowledge, cropping algorithms take the crop offsets for the left, right, top and bottom *of the video* and apply it uniformly to *all* the frames in the video clip. The drawback to using fixed crop offsets is that the soccer ball may not be within the cropped window in every frame. In such a case, the result is a video clip in which the soccer ball goes in and out of the scene in successive frames. People will experience discomfort when viewing such poor quality video. At this point, it is evident that variable crop offsets will yield better results - the problem is to determine these crop offsets.

3.1 Background

The preceding account is a problem of video adaptation. The video clip must be cropped in such a manner that the visually interesting features are retained in the output. The question of visually interesting features has intrigued humans for a long time. [7, 17] describe a number of studies on vision and visual attention done over the years that try to answer this question. The results identified certain features that stimulate vision and capture visual attention, some of which include lustre, colour, shape and size, texture and orientation and motion.

As we discussed in Section 2, visual attention analysis helps us determine the attended regions in visual content. These often correspond to high contrast objects and/or action sequences in videos. The adapted output video sequence should retain these attended regions. Thus our problem can be framed as follows:

Problem statement: How do we determine the attended region in a given compressed and encoded H.264 video sequence, and adapt it to a given display with low complexity?

We developed the display adaptation algorithm to address this problem. The rest of this chapter describes the various steps in the the algorithm. The first step toward solving this problem is to find attended objects in the input video. The following section describes the visual attention model which help us locate attended objects in the video.

3.1.1 Visual Attention Model

The display adaptation algorithm is built around a saliency-based computational model for visual attention. We know that visual perception is an inherently active and selective process by which people attend to a subset of the available information for further processing. Visual saliency is a broad term that refers to the idea that certain parts of a scene are more discriminating or distinctive than others and may create some form of significant visual arousal within the early stages of the HVS. Cognitive psychology and computer vision provide numerous approaches for building visual saliency models [27], and research on visual saliency typically follows one of two approaches: the bottom-up or stimulus-driven approach, and the top-down or task-dependent approach. In our case, for visualizing a scene without a specific task in mind, we focused on the bottom-up, stimulus-driven approach in this work. The Walther [44] and Itti implementation [16] of the biologically inspired saliency-based model of bottom-up attention proposed in [20] provide a framework for extracting features and forming saliency maps.

Shown in Figure 3.1 is the general architecture of the visual attention model. In this procedure, first a multi-scale representation of the original input image is obtained by using dyadic Gaussian pyramids. Feature extraction is accomplished through a set of linear center-surround operations that simulate visual receptive fields as the difference between fine and coarse scales. The across-scale difference between two maps is obtained by interpolation to the finer scale followed by point-by-point subtraction. The extracted feature maps are first normalized, and then across-scale combined into conspicuity maps for the corresponding feature. Finally, the conspicuity maps are merged into a saliency map, S, by linear combination.

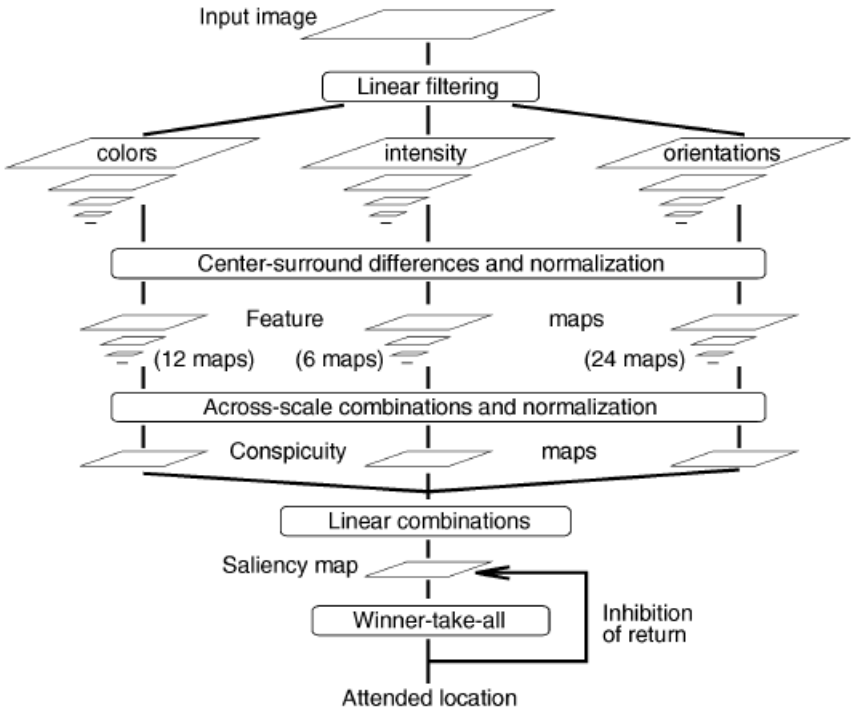

Figure 3.1: General architecture of the saliency-based visual attention model (adapted from [16])

The first step is to extract the the luminance channel Y, of the decoded I-frame. This constitutes the input image to the visual attention model $I(0)$. Next we obtain a multi-scale representation of$I(0)$. We use eight spatial scales, $\sigma \in [0...7]$ which result in eight filtered images, with image reduction from 1:1 at scale $\sigma = 0$ down to 1:128 at scale $\sigma = 7$. These filtered images are stored as a Gaussian luminance pyramid $I(\sigma)$. We perform linear center-surround operations on $I(\sigma)$ to compute feature maps, which is implemented as the difference between fine and coarse scales. The centre is a pixel at scale $c \in \{2, 3\}$ while the surround is the corresponding pixel at scale $s = c + \delta$, with $\delta \in \{3, 4\}$. The across-scale difference between two maps is obtained by interpolation to the finer scale followed by point-by-point subtraction. This operation is denoted "$\ominus$" below.

Luminance (intensity) contrast is detected by neurons in the HVS that are sensitive either to bright centers on dark surrounds or to dark centers on bright surrounds [10]. The responses computed through center-surround differences are stored as a set of four feature maps, $L_I(c, s)$, with $c \in \{2, 3\}$ and $s = c + \delta, \delta \in \{3, 4\}$:

$$L_I(c, s) = |I(c) \ominus I(s)| \tag{3.1}$$

Studies [7] have shown that Gabor filters closely approximate the impulse response of orientation-sensitive neurons in the HVS. Local orientation maps, $O(\sigma;\theta)$ at different scales σ, and orientations θ, are obtained by convolution the corresponding level of the intensity pyramid, $I(\sigma)$ with Gabor filters. Similar to Eq. 3.1, the orientation contrast maps are obtained by computing the across-scale difference of the local orientation maps with $c \in \{2,3\}$ and $s = c + \delta,\ \delta \in \{3,4\}$ and $\theta \in \{0°, 45°, 90°, 135°\}$. These are stored as a set of sixteen maps $L_\theta(c,s)$:

$$L_\theta(c,s) = |O(c;\theta) \ominus O(s;\theta)| \tag{3.2}$$

Thus we obtain four feature maps for intensity and sixteen feature maps for orientation. Our total of twenty maps is about half of the forty-two feature maps generated in [44]. This reduction is achieved by neglecting the colour features and using fewer scales in the multiscale pyramid.

The feature maps obtained in Eqs. 3.1 and 3.2 have different global maxima. Hence they are normalized before further processing. The normalization operator, $\mathcal{N}(\cdot)$ is a non-linear iterative function that simulates local competition between neighbouring salient locations [15]. This is implemented through a convolution with a difference of Gaussian filter followed by rectification. For our simulations, we used two iterations of the operator. Increasing the number of iterations did not produce any perceptibly different results.

The feature maps, Eqs. 3.1 and 3.2 are first summed over the centre-surround combination using across-scale addition $\oplus$, and these sums are normalized thereafter:

$$\bar{L}_l = \mathcal{N}\left(\bigoplus_{c=2}^{3}\bigoplus_{s=c+3}^{c+4} L_l(c,s)\right),\ l \in \{I,\theta\} \tag{3.3}$$

The conspicuity map for intensity is the same as $\bar{L}_I$ obtained in Eq. 3.3 above. For the orientation feature, we have obtained four normalized orientation maps corresponding to the four orientations θ. These maps are once again summed and normalized to yield the orientation conspicuity map:

$$\begin{aligned} C_I &= \bar{L}_I \\ C_O &= \mathcal{N}\left(\sum_\theta \bar{L}_\theta\right), \quad \theta \in \{0°, 45°, 90°, 135°\} \end{aligned} \tag{3.4}$$

All conspicuity maps are now combined into one saliency map:

$$S = \frac{1}{2}\left(C_I + C_O\right) \tag{3.5}$$

In our framework, we have used the Walther-Koch visual attention model to determine the attended region in decoded I-frames, with a few differences:

- We consider only luminance and orientation contrasts for the saliency map, and supplement it with the motion information to determine the attended region. Since color contributes less to the overall visual saliency, whereas motion contributes highly [7], we assume that this offsets any loss incurred through neglecting color contrasts.
- We use less scales for feature contrast computation. In our implementation, the center is at scale $c \in \{2, 3\}$ and the surround is at scale $s = c + \delta,\ \delta \in \{3, 4\}$. There is no perceptible difference in the resulting output, and this reduces the computation time.
- We determine the attended region following a morphological closing operation as opposed to using a neural network [44, 15]. This reduces system complexity with no perceptible difference in the output.

3.2 The Dynamic Small-Screen Adaptation (DSSA) Algorithm

The Dynamic Small-Screen Adaptation (DSSA) algorithm follows two separate approaches to determine the attended region in I-frames and P-frames. It uses the visual attention model (Section 3.1.1) to locate the attended region in the I-frame. We augment this attended region with motion information from the bitstream, to detect a coherent attended region in a P-frame. The region centroids are then passed through a smoothing filter to obtain the best trajectory for the attended regions, which are subsequently used to crop the frames. These processes are outlined in the Algorithm steps listed below:

3.2.1 Assumptions made in the Algorithm

We made the following assumptions in our algorithm:

1. *Motion saliency is significantly greater than colour saliency:* The visual attention model described above determines attended regions from static scenes. Extending this model to video brings the temporal dimension into play. This induces us to account for another attention feature, viz. motion. The study of attention features indicates that the HVS is most responsive to motion and least responsive to colour. Since we are working with videos, we expect the attended object to be part of an action sequence, and as a consequence, motion will contribute significantly to the overall saliency. We assume that in such a case the contribution of colour saliency can be neglected.

2. *15 frames for each Group of Pictures (GoPs):* We assumed that the attended object moves consistently in a particular direction for 15 frames. This translates to half a second of uniform motion in a video encoded with 30 frames per second (fps), and is thus, a reasonable assumption. Regions of the video that do not obey this restriction are likely to be noise. All the input videos considered in the implementation have a IPPP GoP length of fifteen frames, which consists of a reference I-frame followed by fourteen P-frames.

3. *Attended object present in I-frame:* An implicit assumption in our algorithm is that the attended object is present in the I-frame. As we describe in later sections, motion information from the P-frame supplements the knowledge of the attended region in the I-frame while determining the attended region in the current P-frame. One might think that this simplifying assumption may adversely affect performance. For instance, if the attended object does not appear in the first I-frame, the entire GoP might be focussed on the wrong attended region. However, the following I-frame should have the attended object. This information is then used to update the attended region selected in future frames. Thereafter the smoothing filter ensures that the early attended regions are updated accordingly.

The following section describes how we determine the attended region in an I-frame using the saliency map.

3.2.2 Determining Attended Region in I-frames

The I-frames in compressed video contain all spatial domain information of the frame. Hence, we can use them to determine the spatial visual saliency in the video clip. In our

implementation, we have used each I-frame as the reference frame for the following P-frames in the GoP. Every time an I-frame is read from the video sequence, a visual saliency map is generated for it.

In our implementation, the saliency map is at a scale 4. In other words, it has one-quarter the height and one-quarter the width of the original image. The reason we chose this is that the motion vector resolution for H.264 is at the most 4x4, so the motion vector map also has one-quarter the height and width of the original frame size. This allows us to translate the centroids for temporal coherence (Section 3.2.3) without the need for scaling. Furthermore, smaller maps save computation time and memory.

The saliency map obtained from the visual attention model contains a number of competing attended regions. To locate the fixation region, we partition the saliency map S into coherent regions. First, all pixels in S smaller than 97% of the global maximum, are set to zero.

$$S(i,j) = \begin{cases} S(i,j), & if\ S(i,j) \geq 0.97 \max\{S\} \\ 0, & otherwise \end{cases} \tag{3.6}$$

Then we perform a morphological closing operation. This removes isolated noise regions that are not candidates for the attended region, and leaves fewer pixels to process. We define the Spatial Attention Value (SAV) of each region as the sum of the constituent pixels. Accordingly, the region R^* that maximizes the SAV is chosen as the attended region:

$$SAV(R^*) = \arg\max_{R} \left\{ \sum_{(i,j)\epsilon R} S(i,j) \right\} \tag{3.7}$$

The attended region R^* is saved for checking temporal coherence in future P-frames, while its centroid C_{t^*} is stored for cropping. The centroid is calculated as

$$C_{t^*} = \frac{\sum_{\mathbf{p}\in R^*} \mathbf{p} \cdot S(\mathbf{p})}{\sum_{\mathbf{p}\in R^*} S(\mathbf{p})} \tag{3.8}$$

where $\mathbf{p}$ denotes any pixel location (i,j), and $S(\mathbf{p})$ is the value of the saliency map S at $\mathbf{p}$.

This gives us the attended region in the I-frame and its corresponding centroid. The next frame in the video clip is a P-frame. As we know, a P-frame contains motion information

and prediction residues. We use the motion information together with the knowledge of the attended region R^*, to determine the attended region in the P-frame. This is covered in the following section.

3.2.3 Determining Attended Region in P-frames: Temporal Coherence

When the processing a P-frame, the algorithm uses the motion vectors in the bitstream to create a motion map, which is then processed further to yield the attended region for the current P-frame. The blocks in a P-frame are predicted from other reference blocks. For each such block or macroblock, the bitstream stores the difference of the actual value from the prediction in the residue, and the vector offset of the prediction block from the current block is stored in the motion vector.

The H.264 standard has provision for multiple reference frames and variable block sizes. This presents a challenge since a frame can be divided into an arbitrary number of macroblocks and blocks with corresponding motion vectors. Hence one motion vector may be represent a 16x16 macroblock, a 16x8 block, an 8x16 block and so on, all the way down to a 4x4 block. Two 16x16 macroblocks within a frame may be subdivided differently and have different number of motion vectors. Furthermore, neighbouring blocks may even be predicted from different reference frames in the sequence. While these features make H.264 a powerful coding standard, they also increase complexity for processing. To overcome any issues that may arise from these variations, we consider a motion vector for each 4x4 block in the current frame and normalize it to account for different reference frames.

As we mentioned above, we consider a motion vector for each 4x4 block in the frame. In case there is a single motion vector for, say a 16x16 block, it is split into sixteen 4x4 blocks, each having the same motion vector. These motion vectors are then normalized by dividing it by the distance between the current P-frame (or B-frame) block and its associated reference frame block. This accounts for the difference in motion vector magnitudes due to the use of multiple reference frames. If the original frame is assumed to be of size $4w \,\mathrm{x}\, 4h$, the normalized motion vector field can be visualized as a two-channel map of size $w \,\mathrm{x}\, h$. The two channels correspond to the horizontal and vertical components of the motion vector, respectively.

$$\mathrm{MVF}(w, \mathbf{p}|t) = \mathbf{mv}(w, \mathbf{p}|t).\frac{|t - t^*|}{|t - t_{ref}(\mathbf{p})|} \tag{3.9}$$

Here MVF is the motion vector field, **mv** is the motion vector read from the H.264 bitstream, t is the index of the current frame, t_{ref} is the index of the reference frame for the current block $\mathbf{p} \equiv (i, j)$, t^* is the index of the previous I-frame, and $w \in \{x, y\}$ denotes the motion vector component.

The pixels in the motion vector field MVF can take any value, both positive and negative, within the search range used by the H.264 encoder. Moreover, since the motion vectors are usually interpolated to quarter pixel accuracy, the number of possible values increases four-fold. This gives quite a large range of values in the motion map. The normalized motion vector field MVF consists of a multitude of values, indicating the background and moving objects in the foreground. Such moving objects usually constitute action sequences, that are of user interest. A group of blocks with identical motion is likely to be an object. To find regions with homogeneous motion, we separate the range of available values into classes and give each class a unique label. In our implementation, we use seven classes - three each for positive and negative components and one for the zero component. The zero-valued component denotes a stationary background, which is conveniently ignored from consideration for the attended region. The motion vector field MVF is thus transformed into a M-map of class-labels.

$$\mathrm{M}\left(w, \mathbf{p}|t\right) = Q\left(\mathrm{MVF}\left(w, \mathbf{p}|t\right)\right) = m_i, \tag{3.10}$$

where the range of motion vector values $[mv_i, mv_{i+1}]$ fall in class m_i.

Here $Q\left(\cdot\right)$ is a classifying operator: it acts on each element of MVF to determine which class the element falls in, and assigns a class label to the corresponding location in the M-map. Thereafter, a morphological closing operations removes noise from the M-map. The M-map is now partitioned into homogeneous motion regions, $R_k,\ k = 1, 2, 3, ...K$. Although there are only six labels for motion, there may be any number of homogeneous motion regions in a frame. The regions R_k are created in a way that they are mutually exclusive, $\mathrm{M}\left(w|t\right) = \bigcup_k R_k$. In case there is an overlap between two regions, the contentious portion goes to the region that exhibits faster motion.

As a convention, we chose R_0, the region with $k = 0$ to be the stationary background. Thus, we obtain a set of regions $R_k,\ k = 1, 2, 3, ...K$ which are contenders for the attended region in the P-frame. The attended region must be coherent across frames in the video clip. Hence, the attended region must closely match the region R^* we obtained from the I-frame in the preceding section. The attended region R_t is the one whose centroid is the closest to C_{t^*} after accounting for motion.

$$D\left(R_t\right) = \arg\min_{R_k} \parallel C_k + \mathbf{mv}(C_k) - C_{t^*} \parallel, k = 1, 2, 3, ...K \tag{3.11}$$

Here C_k is the centroid of the region R, $\mathbf{mv}(C_k)$ is the motion vector at location C_k and C_{t^*} is the centroid of R^*(Eq. 3.8).

The computation for the centroid, C_k is similar to that shown in Eq. 3.8 for C^*. If we let the pixel, $\mathbf{p} \in R_k$ denote the location (i, j) in $\mathbf{mv}_\mathrm{t}$, the centroid is:

$$C_k = \frac{\sum\limits_{\mathbf{p} \in R_k} \mathbf{p} \cdot \mathbf{mv}_\mathrm{t}\left(\mathbf{p}\right)}{\sum\limits_{\mathbf{p} \in R_k} \mathbf{mv}_\mathrm{t}\left(\mathbf{p}\right)} \tag{3.12}$$

The region R_t is the best temporally coherent match to the visually attended region, R^*, as per the DSSA algorithm. The centroid of the region, C_t is stored for cropping, and the process of attended region detection continues until all the frames in the video clip are exhausted. Thereafter, the frames are cropped and sent to the display.

3.2.4 Post-processing and Display

The sequence of centroids $\{C_t\}_{t=0}^{t=T_{max}}$ obtained through the above process is jittery, since they are computed based only on the reference frame, and ignore other inter-frame correlation. The jitter in frame transitions results in a poor viewing experience. It can be minimized by smoothing the centroid sequence. For our implementation, we assume that the centroids follow a second order polynomial trajectory over time. This is given by

$$\begin{aligned} y_t &= a_1 t^2 + a_2 t + a_3 \\ x_t &= b_1 y_t + b_2 t + b_3 \end{aligned} \tag{3.13}$$

where $C_t \equiv (x_t, y_t)$ is the centroid for frame $\mathcal{F}_t$.

This smoothed centroid sequence is used to crop the frames before they are sent for display. The H.264 decoder buffers the decoded frames before writing them out. This allows us to smooth the entire centroid sequence before writing out the frames. If the buffer size is limited, a viable alternative would be to use a different smoothing filter, such as a moving average filter. Only a few frames may be in the buffer at a time for processing. In our implementation, we obtained both types of outputs. For both the cases, we used equation 3.13 on the centroid sequence. For the moving average filter, we used a sequence of 30

centroids at a time, and updated the set with new values for every iteration. The output subjective results are comparable for this filter. A better choice for such a scheme might be to use an adaptive filter such as a recursive least squares (RLS) filter [14]. The advantage of this is that using a forgetting function and an the initial set of inputs, the computational complexity can be reduced to first order. This would provide for better performance.

The output display size is fixed to $w_{disp} \mathrm{x}\, h_{disp}$. Knowing this and the centroid, $C_t \equiv (x_t, y_t)$ we can easily determine the crop offsets for the frame $\mathcal{F}_t$ to be given by: left-top $\equiv \left(x_t - \frac{w_{disp}}{2}, y_t - \frac{h_{disp}}{2}\right)$ and right-bottom $\equiv \left(x_t + \frac{w_{disp}}{2} - 1, y_t + \frac{h_{disp}}{2} - 1\right)$.
The frames are now cropped so that the output frames retain only the cropped rectangle. These cropped frames are then displayed on the screen.

Summary Before we end this chapter, here's a summary of the DSSA algorithm:

1. `Read input frame from bitsream`
2. `If the current frame is an I_frame:`
   (a) `Decode the bitsream to get frame,` $\mathcal{F}_t$.
   (b) `Compute visual saliency map,` $S = \mathrm{SaliencyMap}\left(\mathcal{F}_t\right)$
   (c) `The attended region` R^* `is the one with highest Spatial Attention Value, SAV (eq. 3.7):`

$$SAV\left(R^*\right) = \arg\max_{R} \left\{ \sum_{(i,j)\epsilon R} S\left(i,j\right) \right\}$$

   (d) `Store` $C_{t^*} = centroid\left(R^*\right)$.
3. `If the current frame is a P_frame:`
   (a) `Generate M-map from the motion vectors` $\mathbf{mv}_t$ `(eqs. 3.9 and 3.10):`

$$\mathrm{M}\left(\mathbf{p}\right) = Q\left(\mathrm{mv}\left(w, \mathbf{p}\right) \cdot \frac{|t - t^*|}{|t - t_{ref}\left(\mathbf{p}\right)|}\right),$$

   `where` $w \in \{x, y\}$ `and` $Q\left(\cdot\right)$ `is a classification operator.`
   (b) `Partition` $\mathrm{M}\left(w|t\right)$ `into non-overlapping homogeneous motion regions,` R_k.

(c) Find the attended region $R_t \in \{R_k\}_{k=1}^{k=K}$ whose centroid best matches the region R^* (eq. 3.11):

$$D\left(R_t\right) = \arg\min_{R_k} \parallel C_k + \mathbf{mv}(C_k) - C_{t^*} \parallel, k = 1, 2, 3, ...K$$

(d) Store $C_t = centroid\left(R_t\right)$.

4. Repeat steps 1 through 3 until all the frames in the video sequence are processed.

5. Smooth the centroid sequence, $\{C_t\}_{t=0}^{t=T_{max}}$

6. Crop frames $\{\mathcal{F}_t\}_{t=0}^{t=T_{max}}$ to a rectangle of size $w_{disp}\,\mathrm{x}\,h_{disp}$, centred at $\{C_t\}_{t=0}^{t=T_{max}}$. Output cropped frames.

In the next chapter, we describe the experiment to evaluate system performance and discuss the observed results.

Chapter 4

Experimental Results and Illustrative Examples

In the earlier chapters, we discussed the need for video adaptation, and described our Dynamic Small-Screen Adaptation (DSSA) algorithm. This display adaptation algorithm is based upon the saliency-based visual attention model proposed by Walther and Koch in [44]. Whereas Walther and Koch built upon the earlier Itti and Koch [16] model to find attended regions in images, the DSSA algorithm extends this to videos, but focuses on it's application to video adaptation. As a result, we are concerned with detecting one coherent attended region in the video sequence.

This chapter covers our experimental setup, the benchmark tests and the results we observed. We also include some illustrative examples which highlight the capabilities and limitations of the display adaptation algorithm. The following section covers the experimental setup.

4.1 Experimental Setup

All the experiments were run on a notebook computer, with an AMD Turion64 1.6 GHz processor and 768 MB of RAM. We used a 32-bit Windows XP operating system and Visual C++ for programming. We implemented our algorithm on this system and integrated it into the H.264 codec JM version 13.2. The DSSA algorithm is part of the H.264 decoder. We executed it to generate a set of test video sequences. The input video sequences were either CIF (352x288) videos or SIF (352x240) videos. All input sequences were in the YUV 4:2:0 colour format.

Since assessing the effectiveness of a visual detection scheme is a subjective task, manual evaluation is inevitable. To evaluate the performance of our scheme, we invited 14 respondents and showed them three video clips for each sequence: the original (input), benchmark and DSSA output video sequences. The respondents were asked to rate each of these test sequences on a scale of 1 (Poor) to 5 (Good). We normalized the benchmark and DSSA ratings with respect to the average input video ratings. The column *Satisfied Respondents* in the Tables 4.1 and 4.2 below lists the proportion of users who rated the corresponding video 3 or higher. As part of the evaluation, we also asked the volunteers to identify the object-of-interest in the video sequence, and counted its occurrence in the input and output sequences. The ratio of the object's occurrence in the output sequence to that in the input sequence is listed under the heading *Detection Rate*. Further, we measured the time taken to adapt the input video sequence, and listed the *Adaptation Time* as the time taken per frame.

4.1.1 Simplifications adopted for ease of implementation

1. *No B-frames in encoded video*: We assume that the input H.264 video sequence has a IPPP frame structure. This makes the implementation easier, and allows us to check the performance of the algorithm. B-frames are bi-predictive frames, and can use reference frames both from the past and the future in terms of display order. Thus in the algorithm we presented in the previous chapter, B-frames can have a negative value of $t - t_{ref}$ whereas P-frames will always have a positive value for this term, since for P-frames $t > t_{ref}$ always. However, we used the absolute value $|t - t_{ref}|$ in our algorithm, so it can work with B-frames as well.

2. *QCIF video output*: We fixed the default output display size in our algorithm to the QCIF format, 176x144. This was done simply to reduce the number of variable parameters. The system can be easily modified to take this parameter as input.

3. *Some encoder parameters*: All videos are encoded in `FRExt High Profile, Level 2`. Other parameters worth noting include `IntraIDRPeriod=15, IDRPeriod=15, QPISlice=28, QPPSlice=28, SearchRange=32`.

We discuss the simulation results in the next section.

4.2 Simulation Results

This section tabulates the results of subjective assessment on the benchmark output and the DSSA output videos. The output video ratings were normalized with respect to the corresponding input video ratings in order to offset any difference in video quality, perceived by the respondents.

4.2.1 Benchmark test

Our benchmark was the visual attention model applied independently to each decoded frame. This is a logical choice for our benchmark since the DSSA algorithm extends the attended region detection from images to videos, and in doing so, it makes use of the additional motion information present in the video bitstream. The benchmark is created by running the visual attention module separately on each decoded frame of the H.264 bitstream. This gives us the attended region in each frame. The region centroids are then passed through the smoothing filter, after which the frames are cropped and displayed. No motion information is used. Table 4.1 lists the performance of the benchmark.

Table 4.1: Estimating attended region: Benchmark

Sequence	**Attended objects**	**Object detected in #Frames**	**Object present in #Frames**	**Detection Rate (%)**	**Satisfied Respondents (%)**	**Adaptation Time (s/frame)**
Coastguard	Small private boat	90	91	98.90	75	5.035
	Large coastguard boat	50	59	84.75		
Dravid	Ball and bat	10	17	58.82	85.71	4.923
Football	Player#82 (blue jersey)	115	115	100	87.5	4.687
	Scuffle	87	120	72.50		
Irene	Hands	433	539	80.33	100	4.736
Mobile	Ball and engine	103	138	74.64	62.5	4.641
Pingpong	Racquet and ball	31	67	46.27	62.5	4.632
Tempete	Yellow flower	122	259	47.10	85.71	4.764

4.2.2 DSSA Algorithm results

Table 4.2: Estimating attended region: DSSA

Sequence	Attended objects	Object detected in #Frames	Object present in #Frames	Detection Rate (%)	Satisfied Respondents (%)	Adaptation Time (s/frame)
Coastguard	Small private boat	91	91	100	75	0.556
	Large coastguard boat	59	59	100		
Dravid	Ball and Bat	17	17	100	85.71	0.700
Football	Player#82 (blue jersey)	115	115	100	75	0.484
	Scuffle	100	120	83.33		
Irene	Hands	496	539	92.02	100	0.452
Mobile	Ball and engine	121	138	87.68	71.4	0.521
Pingpong	Racquet and ball	34	67	50.47	75	0.492
Tempete	Yellow flower	225	259	86.87	100	0.499

As shown in Table 4.2, the framework determines the appropriate attended object from a video sequence in over 80% of the frames. The system performs better for sequences in which the attended object is larger and is moving uniformly. The dynamic nature of the system is demonstrated in the *Coastguard* sequence, where the camera pans first from left to right following the private boat, and then pans from right to left tracking the coastguard boat. The system is able to follow both the boats appropriately and handles the change in direction with ease. Figure 4.1 demonstrates this feature.

When competing objects are available, the system may occasionally choose distractors over the attended object. This occurs in the *Pingpong* sequence where the bright red racquet, the player's arm and the smaller but fast moving ball are all competing for salience. The system is not always able to fit all of them in the output frame. As a result, the *Pingpong* sequence only has a 50% detection rate.

In the *Mobile* sequence (Fig. 4.2), the system follows the train engine and the ball initially but is later distracted by the scrolling background and has to decide between competing salient objects. As a result of this, it misses the engine in some frames. However, the overall result is still fairly satisfactory.

In [3] Cheng et al obtained intensity, color and motion feature maps from source video sequences. They also processed the video for camera motion. To evaluate their scheme,

they showed 15 video clips marked with the estimated RoI to 10 observers, who rated the clips as Good, Acceptable or Failed. The results indicate over 95% of the respondents feel comfortable (Good or Acceptable) with the determined RoI. Zhai and Shah used color histograms and motion contrast based on planar motion between frames to generate a spatiotemporal saliency map [47]. They processed source video sequences to determine the attended region. In a test similar to above, they showed their video clips with marked attended regions to 5 assessors, who once again rated the clips as Good, Acceptable or Failed. Their results show that over 90% of the respondents were satisfied with the detected region. Neither literature has any other measure for evaluation.

Unlike [3] and [47], our framework uses the encoded compressed video bitstream as input, and the motion vectors form our source of motion information. We also produce a cropped output which is likely to introduce some distortion, leading to reduced video quality. It is to be noted that even with 100% accurate attended region detection, the cropped output video sequence may not be pleasant to view. Our results show that over 85% of the respondents were satisfied with the output video sequence. Also the attended region was satisfactorily identified with a detection rate of over 80%. Another point of contrast is that the DSSA is a causal scheme. While [3] utilizes a continuous video shot for RoI determination, [47] uses successive frames to generate the temporal and spatial saliency maps. DSSA processes and determines the attended region in I-frames followed by that in the successive P-frames, until the next I-frame is encountered. This causal nature of DSSA makes it suitable for application in a transcoder.

One limitation of the system is that the P-frame attended region detection depends on the success of the I-frame region detection. In case there is a detection failure in the I-frame, the following 14 P-frames in the GoP will also have an erroneous attended region selected. However, the system should recover and produce satisfactory detection for the next I-frame. Since the centroids of all the frames are processed together in the smoothing filter, the error in one GoP is rectified to some extent. Despite the rectification, the Detection Rate is expected to drop in such a case.

Overall, the DSSA framework shows promising results, and is a good candidate for real-world video adaptation applications. We present our concluding remarks and a few problems for future research in the following chapter.

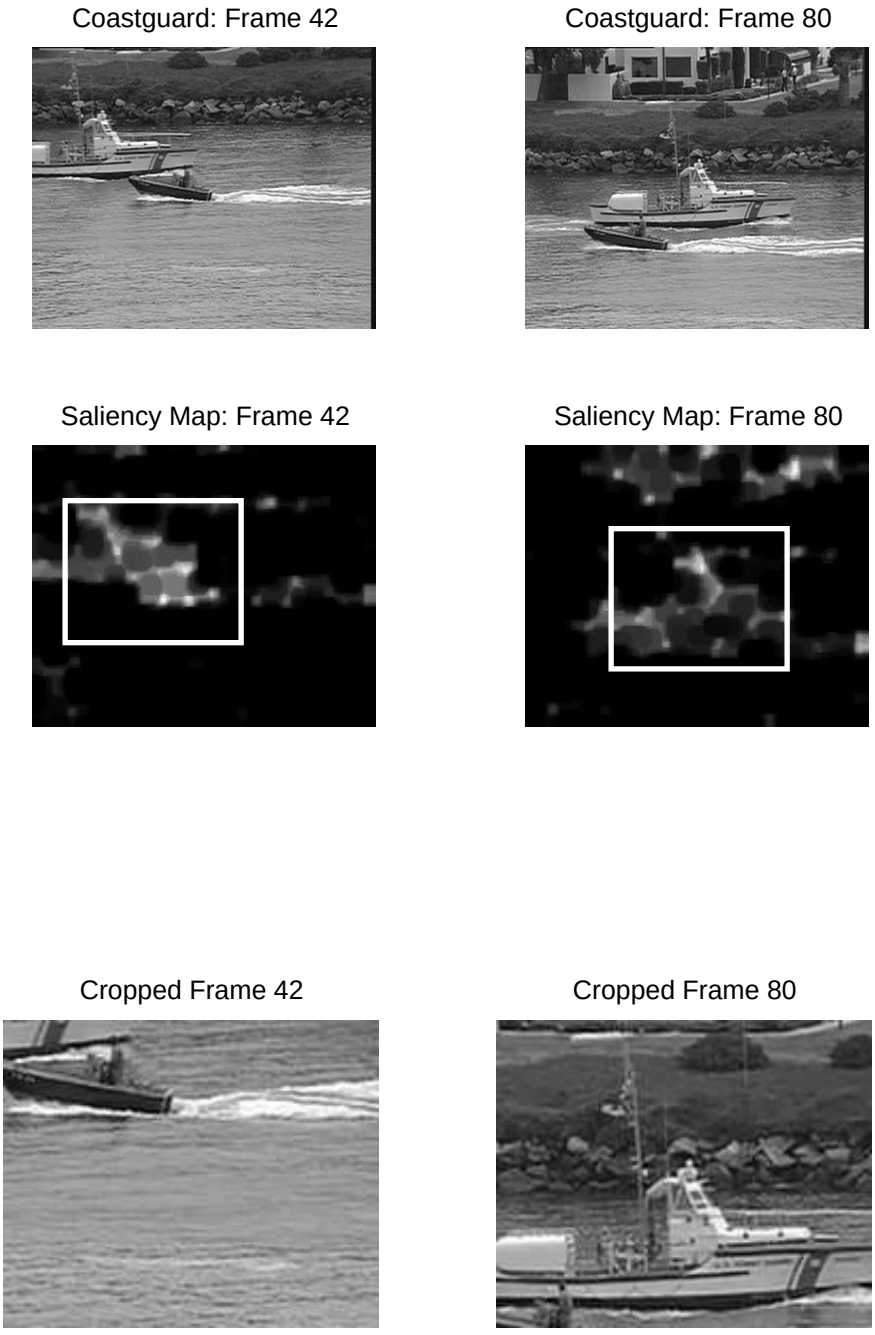

Figure 4.1: Frame 42 and Frame 80 of the *Coastguard* sequence shows the DSSA switching from following the small boat to the large coastguard boat. Cropped frames are scaled to twice their height and width.

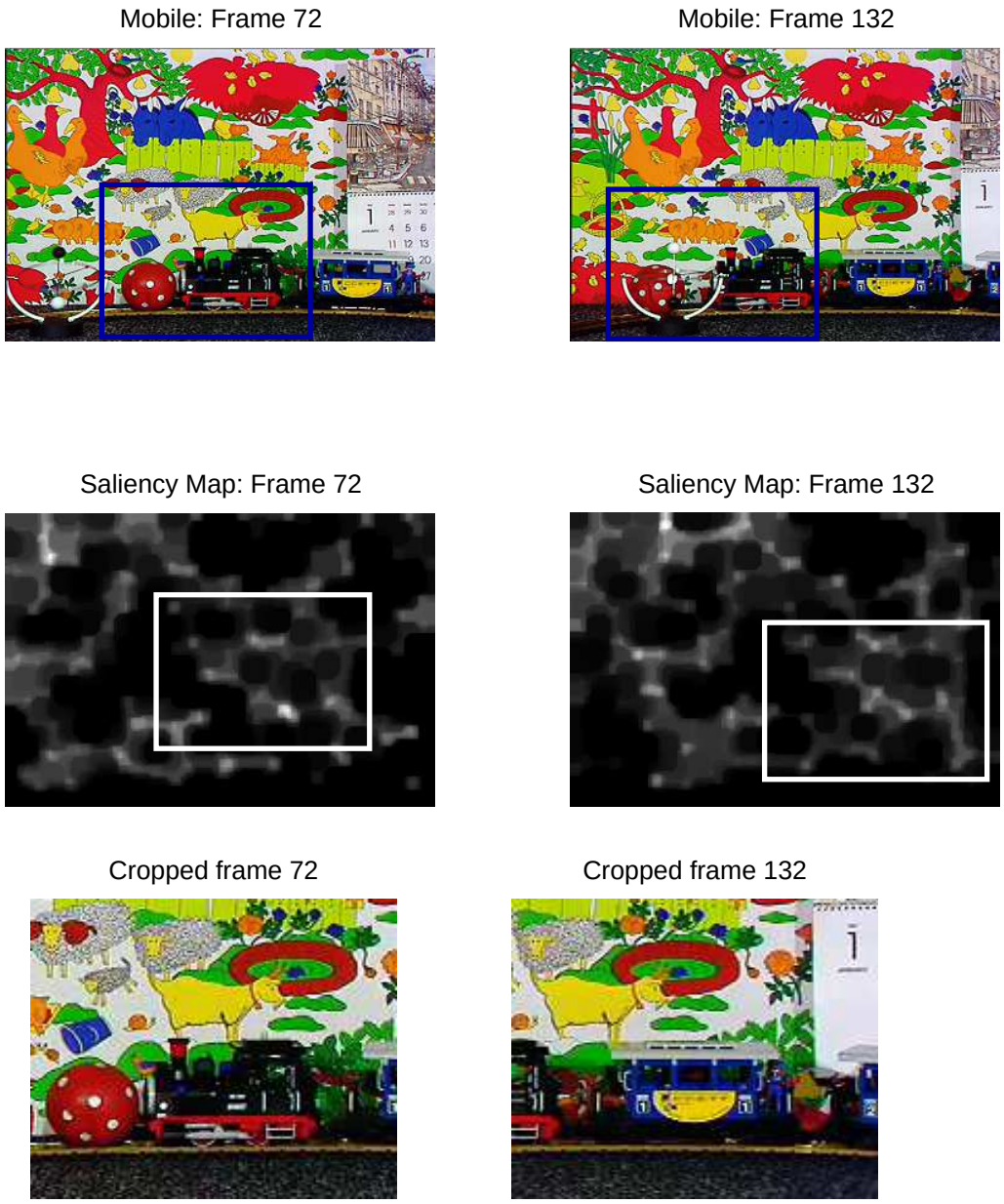

Figure 4.2: Frame 72 and Frame 132 of the *Mobile* sequence shows the DSSA following the engine and red ball initially, but distracted by the background later. Cropped frames are scaled to twice their height and width.

Chapter 5

Conclusion

We developed a framework to determine the attended region in a H.264 video sequence using a bottom-up saliency approach. The attended region is chosen based on the luminance, orientation (texture) and motion features coded in the bitstream. The luminance and orientation information are obtained from the decoded I-frame, and are used to create the saliency map S. We use the motion vectors in P-frames to generate a homogeneous motion region through motion classification in M-maps, and then select the coherent region as the attended region in the P-frame. When the output sequences were shown to a set of respondents, they gave an encouraging response. Such a system has real-world application and great potential for emerging technologies.

5.1 Applications

1. This framework can be directly applied to small form factor devices, such as handheld computers and media players for watching video. Most users aren't happy to see videos on their devices due to the poor resolution of the video on-screen. With a display adaptation algorithm, the viewing experience can improve significantly.

2. DSSA can be applied not only in the devices, but can also be used by content delivery operators at the source to transcode video *before* transmission. For instance, a single source video sequence may be transcoded into a video compatible for digital TV broadcast and another compatible for mobile video players. The DSSA algorithm can adapt each output video according to the desired resolution before they are re-encoded for transmission.

5.2 Limitations

As with every system, the DSSA scheme also has its limitations. Some of these are listed here.

1. We assumed that the attended object is visible in the first I-frame. Attended region detection in P-frames depends on successful detection of the attended region in the previous I-frame. Thus, the DSSA is a causal system. In case the system detects fails to detect the correct region in the I-frame, the remaining P-frames in the GoP will also choose the wrong region. But the system is expected to recover and detect the correct region for the next I-frame. Since the smoothing filter is applied to the entire set of centroids, small errors in the attended region centroids are likely to be rectified. However, larger errors might not be fully compensated. Therefore, we expect a drop in the detection rate.

2. At this time, our scheme lacks any fuzziness in the attended region selection. In other words, we consider only one RoI at a given time for any past or present frame. This limits the coherence tests for the attended region built into the DSSA. It is possible that the most salient RoI for a frame may not be the best choice for the attended region when the entire video sequence is considered. The scheme may be improved by building in some fuzziness, wherein multiple RoIs could be marked on each frame and a selection algorithm later determines the optimal set of RoIs for final output.

3. If the DSSA were to consider fuzziness, we would also have the option of selecting a weighted centroid based on region sizes as well as saliency values. This might account for the occasions when different viewers want to view different objects in the video, by including multiple contending objects in the adapted video sequence.

4. The P-frame attended regions in the DSSA scheme are determined based on the attended region detected in the I-frame. This limits the robustness of the system. Considering the correlations among P-frames would give us more information about the RoI. Furthermore, since motion is a very important attention feature, such a step is likely to provide an improved estimate for the attended region selection.

5.3 Future Work

The current application uses motion information in conjunction with saliency-based attended region location to determine a meaningful video region to display. This work pro-

vides opportunity for further research in a number of areas. These include the following:

1. So far, the P-frame region determination is related only to the reference I-frame. In that sense, it is a memoryless system. The system performance can improve by considering the inter-frame correlation among P-frames. If the knowledge of past attended regions is available, the system can also utilize it to restrict its search for the attended region to a smaller part of the current frame.

2. The benchmark test using the Walther-Koch model [44] is limited in scope since it was designed to evaluate images (rapid static scenes). Since image analysis is different from video analysis, a different benchmark may be considered to account for motion sequences. A possible subjective benchmark test could be to use eye-trackers on human participants, and record their fixations through gaze tracking. Such an evaluation can directly provide us with the selected RoI, which can then be used as a benchmark.

3. The output videos are sometimes jittery since the centroids do not fall on a smooth trajectory. At the time of writing, the authors are not aware of any smoothing filter that can account for such dynamic systems. A filter that can preserve the continuity of the video sequence will greatly improve the viewer satisfaction.

4. The present system takes about 0.5 seconds to process each frame. This is rather slow for real-time applications. The slowest step in the whole system is the Gabor filter stage to determine orientation contrast. Research is needed to study the efficacy of other filters or processes which can be successfully substituted for the Gabor filter. A possible direction for work is to use the existing DCT coefficients for texture filtering. We know that the DCT coefficients contain spatial frequency components of the image. Instead of using a Gabor filter to obtain texture information from the pixel domain, we can use the existing DCT coefficients in the bitstream, such as in [11]. Since the DCT coefficients can be read directly from the bitstream, such a scheme should be fast and would save processing time significantly.

5. Lastly, DSSA provides a smaller output video to an input video. This can be especially helpful for transmission over bandwidth-constrained networks. This system can be further developed to behave as a transcoder, wherein, the existing motion information can be reused, thus saving resources in the costly motion estimation and compensation steps. As we mentioned in Chapter 4, the DSSA is a causal system, which makes it suitable for application in a transcoder. 3G service providers and users will derive immense benefit from such a transcoder.

References

[1] Ishfaq Ahmad, Xiaohui Wei, Yu Sun and Ya-Qin Zhang, "Video Transcoding: An Overview of Various Techniques and Research Issues," IEEE Trans. Multimedia, vol. 7, pp. 793-804, October 2005

[2] Li-Qun Chen, Xing Xie, Xin Fan, Wei-Ying Ma, Hong-Jian Zhang and Heqin Zhou, "A visual attention model for adapting images on small displays," Multimedia Systems, Springer, vol. 9, pp. 353-364, October 2003

[3] Wen-Huang Cheng, Wei-Ta Chu, Jin-Hau Kuo and Ja-Ling Wu, "Automatic Video Region-of-Interest Determination Based on User Attention Model", ISCAS, vol. 4, pp. 3219-3222, 2005.

[4] Wen-Huang Cheng, Wei-Ta Chu and Ja-Ling Wu, "A Visual Attention Based Region-of-Interest Determination Framework for Video Sequences," IEICE Trans. Information and Systems, vol. E88-D (7), pp. 1578-1586, 2005.

[5] Kyungjoo Cheoi and Yillbyung Lee, "Detecting Perceptually Important Regions in an Image Based on Human Visual Attention Characteristic," LNCS 2396, pp 173-187, 2002.

[6] Lang Congyan, Xu De and Yang Xu, "Perception-Oriented Prominent Region Detection in Video Sequences," Informatica, vol. 29, pp. 253-260, 2005.

[7] Karen K. De Valois (editor), "Seeing: Handbook of Perception and Cognition," 2nd ed., Academic Press, 2000

[8] Yining Deng and B.S. Manjunath, "Content-based Search of Video using Color, Texture, and Motion," Proc. IEEE Int'l Conf. Image Processing, vol. 2, pp. 534-537, 1997

[9] Yining Deng and B.S. Manjunath, "Unsupervised Segmentation of Color-Texture Regions in Images and Video," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, pp. 800-810, August 2001

[10] J. Duncan and G.W. Humphreys, "Visual Search and Stimulus Similarity", Psychological Review, 1989.

[11] Richard E. Frye and Robert S. Ledley, "Texture discrimination using discrete cosine transformation shift-insensitive (DCTSIS) descriptors," Pattern Recognition 33(10), pp. 1585-1598, 2000

[12] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C.H. Anderson, "Overcomplete Steerable Pyramid Filters and Rotation Invariance," Proc. IEEE Computer Vision and Pattern Recognition, pp. 222-228, June 1994.

[13] Junwei Han, King N. Ngan, Mingjing Li and HongJiang Zhang "Towards Unsupervised Attention Object Extraction by Integrating Visual Attention and Object Growing," IEEE Int'l Conf. Image Processing, vol. 2, pp. 941-944, October 2004

[14] Simon Haykin, "Adaptive Filter Theory," Prentice Hall, 2002

[15] Laurent Itti and Cristof Koch, "Feature combination Strategies for Saliency-based Visual Attention Systems," Journal of Electronic Imaging, vol. 10 (1), pp. 161-169, 2001.

[16] Laurent Itti, Christof Koch and Ernst Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, pp. 1254-1259, November 1998

[17] Michael Jenkin and Laurence Harris (editors), "Vision & Attention," Springer-Verlag 2001

[18] T. Kadir and M. Brady, "Scale Saliency: a novel approach to salient feature and scale selection," Intl. Conf. Visual Information Engineering, pp. 25-28, 2003.

[19] C. Kamath, A. Gezahegne, S. Newsam, G. M. Roberts, "Salient Points for Tracking Moving Objects in Video," Proc SPIE: Image and Video Communications and Processing 2005, vol. 5685, pp. 442-453, January 2005.

[20] Christof Koch and S. Ullman, "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry," Human Neurobiology, vol. 4, pp. 219–227, 1985.

[21] Ying Li, Yu-Fei Ma and Hong-Jiang Zhang, "Salient region detection and tracking in video," Proc. Intl. Conf. Multimedia and Expo, vol. 2, pp. 269-272, 2003.

[22] Zhi Liu, Zhaoyang Zhang and Liquan Shen, "Moving Object Segmentation in the H.264 Compressed Domain," SPIE Optical Engineering, vol. 46 (1), pp. 17003, 2007

[23] Yu-Fei Ma, Xian-Sheng Hua, Lie Lu and Hong-Jiang Zhang, "A generic framework of user attention model and its application in video summarization," IEEE Trans. Multimedia, vol. 7 (5), pp. 907-919, October 2005.

[24] Yu-Fei Ma, Lie Lu, Hong-Jiang Zhang and Mingjing Li, "A user attention model for video summarization," Proc. ACM Intl. Multimedia Conf., pp. 533-542, 2002.

[25] Yu-Fei Ma and Hong-Jiang Zhang, "A model of motion attention for video skimming," Proc. IEEE Intl. Conf. Image Processing, vol. 1, pp. 129-132, 2002.

[26] Gloria Menegaz and Guang-Zhong Yang, "Image Perception," Editorial in EURASIP Journal on Advances in Signal Processing, vol. 2007

[27] R. Milanese, "Detecting Salient Regions in an Image: From Biological Evidence to Computer Implementation," PhD thesis, University of Geneva, 1993.

[28] Nicolas Moënne-Loccoz, Eric Bruno and Stéphane Marchand-Maillet, "Knowledge-based Detection of Events in Video Streams from Salient Regions of Activity," Pattern Analysis and Applications, vol. 7(4), pp. 422-429, December 2004.

[29] Sonia Mota, Eduardo Ros, Javier Díaz, Rodrigo Agís and Francisco de Toro, "Bio-inspired Motion-Based Object Segmentation," Lecture Notes in Computer Science, Springer-Verlag, vol. 4141, pp. 196-205, 2006

[30] Sung-Mo Park and Joonwhoan Lee, "Object Tracking in MPEG Compressed Video using Mean-Shift Algorithm," Proc. Joint Conference of the 4th Int'l Conference on Information, Communications and Signal Processing and the 4th Pacific-Rim Conference on Multimedia (ICICS-PCM '03), vol. 2, pp. 748-752, December 2003

[31] Konstantinos Rapantzikos, Nicolas Tsapatsoulis and Yannis Avrithis, "Spatiotemporal Visual Attention Architecture for Video Analysis," IEEE Wkshp. Multimedia Signal Processing, pp. 83-86, 2004

[32] Iain E.G. Richardson, "H.264 and MPEG-4 Video Compression," John Wiley & Sons, 2003

[33] Philip Salembier, "Overview of the MPEG-7 Standard and of Future Challenges for Visual Information Analysis," EURASIP Journal on Applied Signal Processing. Vol. 2002, no. 4, pp. 343-353, 2002

[34] Bongwon Suh, Haibin Ling, Benjamin B. Bederson and David W. Jacobs, "Automatic thumbnail cropping and its effectiveness," Proc. ACM Annual Symposium on User interface software and technology, pp. 95-104, 2003.

[35] Gary J. Sullivan, Pankaj Topiwala and Ajay Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," SPIE Conference on Application of Digital Image Processing XXVII, August 2004

[36] Ying-Li Tian and Arun Hampapur, "Robust Salient Motion Detection with complex Background for Real-time Video Surveillance," Proc. IEEE Workshop Motion and Video Computing, vol. 2, pp. 30-35, 2005

[37] A. Treisman and G. Gelade, "A Feature-Integration Theory of Attention", Cognitive Psychology, 1980.

[38] John K. Tsotsos, Sean M. Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis and Fernando Nuflo, "Modelling Visual Attention via Selective Tuning,"Artificial Intelligence, vol. 78 (1-2), pp. 507-545, Oct 1995

[39] US Patent 6282317 - http://www.patentstorm.us/patents/6282317/fulltext.html

[40] US Patent 7116716 - Systems and methods for generating a motion attention model - http://www.freepatentsonline.com/7116716.html

[41] Anthony Vetro, Charilaos Christopoulos and Huifang Sun, "Video Transcoding Architectures and Techniques: An Overview," IEEE Signal Processing, vol. 20, pp. 18-29, March 2003

[42] Anthony Vetro, Ajay Divakaran, Huifang Sun and Tommy Poon, "Adaptive Transcoding System based on MPEG-7 Meta-data," Proc. IEEE Pacific-Rim Conference on Multimedia, Sydney, Australia, Dec 2000

[43] Anthony Vetro, Huifang Sun and Yao Wang, "Object-Based Transcoding for Adaptable Video Content Delivery," IEEE Trans. Circuits and Systems for Video Technology, vol. 11, pp. 387-401, March 2001

[44] Dirk Walther and Christof Koch, "Modeling Attention to Salient Proto-objects," Neural Networks, 19, pp. 1395-1407, 2006.

[45] Zhou Wang, Ligang Lu and Alan C. Bovik, "Foveation Scalable Video Coding with Automatic Fixation Selection," IEEE Trans. Image Processing, vol. 12, no. 2, February 2003.

[46] Wei Zeng, Jun Du, Wen Gao and Qingming Huang, "Robust Moving Object Segmentation on H.264/AVC Compressed Video using the Block-based MRF model," Real-Time Imaging, vol. 11 (4), pp. 290-299, 2005

[47] Yun Zhai and Mubarak Shah, "Visual Attention Detection in Video Sequences Using Spatiotemporal Cues," ACM international conference on Multimedia, pp. 815-824, 2006.

[48] Shile Zhang, Jianping Fan, Hong Lu and Xiangyang Xue, "Salient Object Detection on Large-Scale Video Data," IEEE Conf. Computer Vision and Pattern Recognition, pp. 1-6, June 2007.