Foveation-Based Video Coding

by

Mahsa Mohammadkhani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

We designed independent perceptual video compression that can work with all the available video coding standards. We used *fovea* as the Human Visual System (HVS) properties in this thesis.

In this thesis, first, we describe and compare two available foveation techniques used for images. After finding the pros and cons of the techniques, we improve one of the single-fovea techniques for video compression so that it preserves the quality of videos better compared to conventional video coding. Finally, we compare our technique with four foveation-based objective metrics for nine different video contents and different parameters, and in order to improve the evaluation assessment, we did two subjective tests to hear the users opinion about the video results.

To my beloved parents, Nader, and Sima,
and my beloved siblinings
Yalda, and Pouya.

Acknowledgements

First of all, I am most thankful to God, the one who helped me through all the difficulties of this work and the time in Edmonton, when I was far from home and my family.

I would like to thank my supervisors, Prof. Anup Basu, and Dr. Irene Cheng, for all the help they offered me for completing this thesis.

I specially thank my best friend, Bernardo Ávila Pires, for his great help to approach research problems and all of the feedback for this thesis. Also, I would like to thank all of my friends in Edmonton and in Iran who helped me with their encouragement in the completion of this thesis.

I am truly grateful my parents and siblings, who supported me from far away and helped me through all moments, happy and sad.

Contents

1 Introduction **1**
- 1.1 Human Visual Perception Mechanisms . . . 2
  - 1.1.1 Contrast Sensitivity . . . 2
  - 1.1.2 Masking Effect . . . 2
  - 1.1.3 Fovea . . . 3
  - 1.1.4 Visual Attention . . . 4
  - 1.1.5 Multimodality of Attention . . . 5
- 1.2 Motivation . . . 5
- 1.3 Proposed Method Outline . . . 6
- 1.4 Thesis Contributions . . . 6
- 1.5 Thesis Outline . . . 8

2 Literature Review **9**
- 2.1 Literature Review . . . 9
  - 2.1.1 Geometric Methods . . . 9
  - 2.1.2 Filtering-Based Methods . . . 11
  - 2.1.3 Multi-resolution Methods . . . 12
  - 2.1.4 Wavelet-based Methods . . . 14
  - 2.1.5 Other Usage of Foveation . . . 15
- 2.2 Overview of HEVC . . . 15
  - 2.2.1 Partition . . . 16
  - 2.2.2 Predict . . . 16
  - 2.2.3 Transform and Quantization . . . 17
  - 2.2.4 Entropy Coding . . . 17
  - 2.2.5 Parallel Processing . . . 17
- 2.3 Selection of the Fixation Points . . . 17

3 Foveation-Based Video Coding Method **19**
- 3.1 Chapter Overview . . . 19
- 3.2 Foveation Compression . . . 19
  - 3.2.1 Foveation Background . . . 20
- 3.3 Foveated-Based Video Encoder . . . 21
  - 3.3.1 CVR Transform . . . 21
  - 3.3.2 Multiple Scaling Factors . . . 23
- 3.4 Foveated-Based Video Decoder . . . 26
  - 3.4.1 CVR Re-transform . . . 27
  - 3.4.2 Multiple Scaling Factors . . . 31

3.5 Comparison of CVR and Multiple Scaling Factors 31
3.6 An Extension for a More General Transform Function (CVR) 31
3.7 Comparison of The Proposed method and Existing Foveation Methods 34

4 Experiments **38**
4.1 Objective Tests 38
4.1.1 Foveation With Fixed Bit rate 43
4.2 Subjective Tests 45
4.2.1 Selecting The Threshold of α and c 46
4.2.2 Comparison Between The Proposed and Conventional Compression Methods 49

5 Conclusion and Future Work **62**

Bibliography **64**

A Additional Experimental Results **69**
A.1 ROI PSNR Results 69
A.2 FWSNR, FPSNR, and FSSIM Results 110

List of Tables

4.1 Information of Video Sequences 43
4.2 Scores for subjective test 46
4.3 Subjective comparison between conventional (Conv.) and the proposed fovea (Fov.) $\alpha = 0.02$ for both foveated videos. (First series.) . 52
4.4 Subjective comparison between conventional (Conv.) and the proposed fovea (Fov.) $\alpha = 0.02$ for both foveated videos. (Second series) 53
4.5 Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate $= 200$ Kbps. 54
4.6 Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate $= 300$ Kbps. 55
4.7 Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate $= 512$ Kbps. 56
4.8 Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression with bit rate $= 768$ Kbps . 57
4.9 Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate $= 1000$ Kbps. 58
4.10 Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 200 Kbps. 59
4.11 Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 300 Kbps. 59
4.12 Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 512 Kbps. 60
4.13 Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 768 Kbps. 60
4.14 Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 1000 Kbps. 61

List of Figures

1.1 Inner look of the human eye structure (Source: Wang and Bovik (2005)) 3
1.2 Illustration of density of retinal cell and photoreceptors with respect to retinal eccentricity (Source: Wang and Bovik (2005)) 4
1.3 Proposed method diagram 7

2.1 Original image (left) and transformed image to curvilinear coordinates (right) (Source: Wang and Bovik (2005)) 10
2.2 Look-up table for the logmap superpixel transformation. This logmap has 16 spokes (S) and 20 rings (R) which are used for indexing (S,R). (Source: Wallace et al. (1994)) 11
2.3 Multi-resolution pyramid model (Source: Geisler and Perry (1998)) . 13
2.4 Block diagram of HEVC (Source: Richardson (2013)) 15
2.5 Eye tracking (Source: Umesh Rajashekar and Bovik (2001)) 18

3.1 Block diagram of the "Foveation-based video coding method (Proposed method)" 20
3.2 The resulting reconstructed images for different values for α and c. The image is the frame #133 of *Vidyo4* sequence and the bit rate is 1 Mbps. 24
3.2 The resulting reconstructed images for different values for α and c. The fixation point of this image is on the person's nose. The image is the frame #133 of *Vidyo4* sequence and the bit rate is 1 Mbps. 25
3.3 Four regions of the frame according to the fixation point, $(x'_f, y'_f))$. Each line connecting (x'_f, y'_y) to a corner of the image defines an angle θ with respect to the positive side of the X-axis. The pratition is given by the regions between these lines. E.g. , the region between θ_1 and θ_2 is the top region of the frame. 26
3.4 The resulting reconstructed images for different interpolation methods. $\alpha = 0.8$ and $c = 0.5$. The fixation point of this image is on the car. The image is the frame#163 of *ChinaSpeed* sequence. The bit rate is 1 Mbps. The blocking artifacts are visible in the peripheral part of the image 3.4a, however, in the 3.4b and 3.4c the peripheral are smooth. (The rest of Figure 3.4 is continued on the next page.) 28
3.4 (Continued. The two images of this part are on the previous page.)The resulting reconstructed images for different interpolation method. $\alpha = 0.8$ and $c = 0.5$. The fixation point of this image is on the car. The image is the frame#163 of *ChinaSpeed* sequence. The bit rate is 1 Mbps. 29

3.5 CVR method Results. $\alpha = 0.32$ and $c = 0.5$. As you can see in Figure 3.5a, α causes the neighboring pixels around the fixation point to expand. The fixation point is on the person's nose. The bit rate is 1 Mbps. The image is the first frame of *Vidyo4* sequence. 30
3.6 Comparison of method Basu and Wiebe (1998) and our proposed method. 3.6b is the frame #1 from Kimono sequence. As you can see in Figure 3.6a, the residual image shows the shifted result in the reconstruction frame. However, no shifting result is caused by our proposed method. 32
3.7 The MSF method results 33
3.8 The CVR method results with four different α values. $\alpha_{x_l} = 0.02$, $\alpha_{x_r} = 0.16$, $\alpha_{y_t} = 0.64$, and $\alpha_{y_b} = 0.04$ (*Vidyo4* sequence, frame# 60) . 35
3.9 The CVR method results with four different α values. $\alpha_{x_l} = 0.16$, $\alpha_{x_r} = 0.16$, $\alpha_{y_t} = 0.32$, and $\alpha_{y_b} = 0.04$ (*Tennis* sequence, frame# 210) . 36

4.1 A viewing geometry. The pramether e is eccentricy (degrees), and v is the distance to the image (computed according to image width) (Source: Wang and Bovik (2005)) 39
4.2 Cut-off frequency in a 1024×720 image. 40
4.3 Contrast Sensitivity Function (CSF) (Source: Matkovic (1997)) 41
4.4 Subjective Tests' guideline of ROIs 47
4.5 First test interface 48
4.6 Second test interface 50
4.7 Chart of Table 4.5 54
4.8 Chart of Table 4.6 55
4.9 Chart of Table 4.7 56
4.10 Chart of Table 4.8 57
4.11 Chart of Table 4.9 58

Chapter 1

Introduction

Nowadays, the demand for multimedia content is growing tremendously. The recent advances in video communication technologies provide the opportunity to produce a huge volume of video content every day for many different purposes, such as entertainment and video conferencing. Due to large original uncompressed video signals, video must be compressed efficiently for transmission and/or storage.

Video compression has been studied for several years to preserve the video quality while maximizing the compression ratio. Traditional video compression standards such as MPEG-1 ISO/IEC (1993), MPEG-2 ISO/IEC (1994), MPEG-4 ISO/IEC (1999), H.263 ITU-T (1995), H.264/Advanced Video Coding (AVC) JVT (2003), and High Efficiency Video Coding (HEVC) (also known as H.265, Sullivan et al. (2012)), the most recent video coding standard, try to achieve compression by removing *spatial and temporal (statistical) redundancies* in the videos. However, in order to achieve better coding efficiency we need to consider *perceptual redundancies* as well as statistical ones. The characteristics of the Human Visual System (HVS) can be helpful for extracting the perceptual aspects of a video and use them in compression to increase the compression ratio without significant, noticeable quality degradation. Therefore, by incorporating limitations of the human eye to video compression, a technique known as *perceptual video compression*, one can maximize the perceived quality, in contrast to the commonly used quality metrics such as the Peak Signal-to-Noise Ratio (PSNR) or the Mean Square Error (MSE) (Lee and Ebrahimi, 2012).

The HVS properties have been studied in many areas such as biology and neuroscience; however, there are still difficulties to understand them. In addition, people want to use different types of video content in different applications so designing perceptual video coding methods that work well in different situations is also a challenge (Lee and Ebrahimi, 2012).

In perceptual video compression, there are three major subfields. The first studies how to define the perceptual model to detect the most or least important regions in terms of perceptual quality in video sequences. The second studies how to employ the perceptual model into encoding standards, and the third studies how to have accurate quality assessment based on HVS characteristics and not just ordinary quality metrics that, according to Lee and Ebrahimi (2012), only consider the distortions of the signal. This work is related to the second subfield and we

employed existing methods to perform the rest of the tasks related to the other subfields.

In the following, we will describe different concepts of human visual perception mechanisms that have been used in perceptual video compression. Then, we will list our contributions.

1.1 Human Visual Perception Mechanisms

In survey of Lee and Ebrahimi (2012) five different concepts have been introduced as human visual perception characteristics relevant to video compression, namely contrast sensitivity, masking effects, fovea, visual attention, and multimodality of attention. In the following sections, we briefly describe each of them according to the work of Lee and Ebrahimi (2012) and other studies mentioned in each section.

1.1.1 Contrast Sensitivity

Contrast is one of the factors that affect human attention from the beginning of visual processing when the HVS converts luminance into contrast. Therefore, the parts of an image with high contrast compared to their neighboring areas are more likely to be seen (Osberger and Maeder, 1998). Contrast Sensitivity (CS) is the sensitivity of the HVS, and its reciprocal is called CS threshold.

The spatial and temporal frequencies of stimuli in visual signals have different effects on the acuity of the HVS, e.g. the HVS cannot distinguish parts of the signal with a frequency below the CS threshold (Lee and Ebrahimi, 2012). Changes in the luminance, such as different patterns or textures with different orientation, are perceived as spatial frequencies. Motion and flicker are in the temporal frequency category. The HVS is very sensitive to frequencies below 30 Hz but this sensitivity degrades fast as this amount goes above 30 Hz (Cox et al., 2008). In image and video compression, researchers attempted to remove the high frequency components from the low ones to achieve a higher compression ratio. In JEPG compression, a transformation such as Discrete Cosine Transform (DCT), a type of Fourier transformation, is applied to the image to separate the low and high frequency components. Then, half of the high frequency bits are removed (resulting in a loss of approximately 5% of the encoded information) (Smith, 1997).

1.1.2 Masking Effect

The visibility of one stimulus (called "*target*") in a scene can be affected by another spatiotemporally neighboring stimulus (called "*the mask*"), and the target's presence can be less perceptible or imperceptible. This effect is called *visual* masking. One type of masking is called *metacontrast*, which happens when there is one fleeting target whose visibility is reduced by the presence of fleeting, temporally succeeding masks in the same spatial location or in its adjacencies (Breitmeyer and Ogmen, 2006; Enns and Di Lollo, 2000). This kind of masking is a form of *backward masking*. On the other hand, if we change the temporal order of the target and mask, we will have a new mask called *paracontrast*, a form of *forward masking*. Another type of visual masking is *masking by light*, in which the visibility of a preceding or

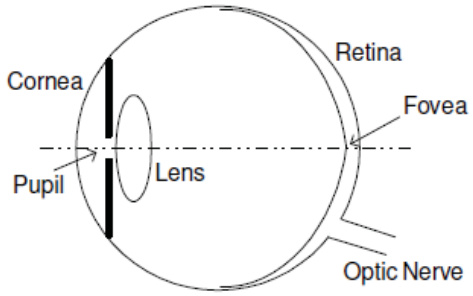


Figure 1.1: Inner look of the human eye structure (Source: Wang and Bovik (2005))

subsequent flashed target stimulus is undetectable due to the presence of a fleeting flash, a uniform illuminated regional mask (Breitmeyer and Ogmen, 2006).

By having a spatially or temporally complex background, human visual ability is also reduced. As an example, fast moving areas or immediate scene changing are difficult for human eyes to track, thus introducing these errors during compression does not affect the perceived video quality (Osberger et al., 1997).

1.1.3 Fovea

Here, we will summarize the *fovea* concept introduced in Wang and Bovik (2005), Chen and Guillemot (2010), Wandell (1995), and Lee and Ebrahimi (2012).

The structure of the human eye is displayed in Figure 1.1 (Wang and Bovik, 2005). The environment's light comes through the pupil and reaches the retina region at the back of the eye, where the photoreceptors of the eye can sense them. The human eye has two different types of photoreceptors: the rods and the cones. The rods are responsible for capturing the visual information of luminance and low-light conditions, while the cones can capture the fast motions and more details in the scene. The fovea is a special circular region with diameter of 1.5 mm at the center of retina. The angle (in degrees) between the straight line from the pupil to the fovea and the other regions in the retina is called *retinal eccentricity* (Wang and Bovik, 2005). Because of the eye's movements, it is possible to intersect this line to any point in sight. This intersection point is called the *fixation point* and is where the viewer's visual attention is fixed. Figure 1.2 from Wang and Bovik (2005) illustrates the density of photoreceptor cells as a function of retinal eccentricity. As can be seen in this figure, in the fovea region (where eccentricity is close to zero degrees) there are many sensor cells, such as the cones and ganglion cells, but there are no rods. This allows the fovea to capture a high-resolution image from a spot of a scene when a person is gazing at it. However, as the distance from the fovea region increases in the retina (i.e. the retinal eccentricity increases), the brain captures less details (or has lower resolution). When the retinal eccentricity is more than 2°, i.e. away from the fixation point, the vision is blurred.

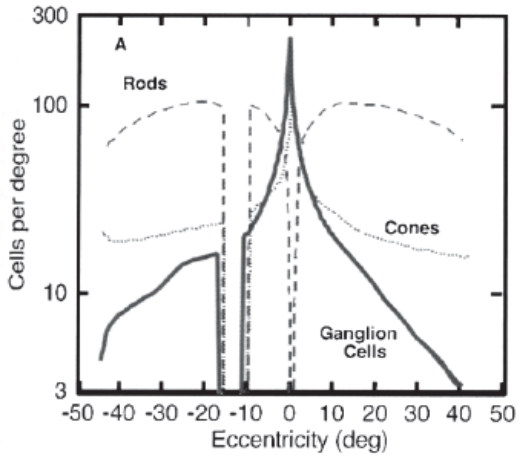


Figure 1.2: Illustration of density of retinal cell and photoreceptors with respect to retinal eccentricity (Source: Wang and Bovik (2005))

1.1.4 Visual Attention

When a human is looking at a scene and is paying attention to examine an object or a person, (by focusing on it,) he/she cannot perceive other things in the object's surroundings very well, even though the peripheral area is visible to him/her. Humans can fail to notice large changes happening in an object or scene (*change blindness*), or even not observe them (*inattentional blindness*) if he/she does not focus attention on them (Simons and Chabris, 1999).

Frintrop et al. (2010, p. 5) define two distinct types of attention:

> directing the focus of attention to a region of interest is associated with eye movements (*overt attention*). However, this is only half of the story. We are also able to attend to peripheral locations of interest without moving our eyes, a phenomenon which is called *covert attention*.

The covert attention is followed by quick eye movement called *saccades* so that the viewer can change his/her focus of attention after he/she finds a Region Of Interest (ROI) in the peripheral area which can be a motion or some familiar objects.

There are different approaches to detect the targets which attract human attention. These approaches use different types of attentional targets: *space-based* (also called *location-based*), *feature-based*, and *object-based* attention. Yantis (2000) provided a thorough overview of the research and studies about these approaches.

Different factors that can drive attention are categorized as *bottom-up* and *top-down*. These factors are defined in Lee and Ebrahimi (2012, p. 686): "Low-level salient features induce bottom-up attention automatically, e.g. abrupt change or prominent appearance of color, shape, motion, orientation, contrast, size, and so on. On the other hand, goal-oriented cognitive control is responsible for top-down attention". Knowledge and expectation are cognitive factors employed to design top-down attention models. As an example, human faces are commonly used as top-down attention factors in video and image applications (Cerf et al., 2007). A

more thorough overview about visual attention can be found in Frintrop et al. (2010) and Lee and Ebrahimi (2012).

1.1.5 Multimodality of Attention

Attention is not solely affected by the HVS. Some studies have proved that hearing can also affect human attention. Mazza et al. (2007) have shown that even though the detailed information about the visual target is given to subjective participants to avoid unwanted distraction from the target, auditory cues affect their attention. This fact is also empirically demonstrated by Tellinghuisen and Nowak (2003).

1.2 Motivation

From time to time, a new video compression standard is introduced with significant differences compared to previous standards, which makes it challenging to adapt existing techniques to the state-of-the-art standard. Therefore, it is important to design a method compatible with the existing compression standards while improving the compression ratio and keeping or increasing the perceptual quality. Another advantage of designing an independent method is that the users do not need to change their encoders; they can simply add the new module as an extension, while the encoder still works with and without the extension video bit streams.

Moreover, as we have said, these methods do not exploit properties of the HVS for compression, and exploring these properties is useful because compression should be employed intelligently so that the users are satisfied with the resulting perceived quality of the coded videos, which is not an easy task to do.

In fact, according to Lee and Ebrahimi (2012, p. 684), "application- and context-dependent quality expectations of users have sometimes prevented researchers from reaching generally applicable perceptual compression techniques." Devices such as mobile phones, personal computers, and hand-held devices have their own limitations. Also, the available bandwidth, the video context, and requested quality for a video bring more challenges to compression. The method should provide a suitable video stream based on the user's demands.

Nowadays, users would prefer to use fast and simple applications rather than complex techniques that require special hardware. The method should also be flexible to all kinds of input, so that the application can be used on various platforms. The goal of this thesis is to address the challenges mentioned above.

In this work, we selected the "fovea" to work with. We believe that when viewers are watching videos, they mostly concentrate on some regions of the video and they do not pay attention to all of the details of the presented frame. Thus, in some situations of limited storage or transmission, foveation can satisfy viewers' expectation of the video quality. With the new video standard, HEVC, some of the existing methods need to be modified and implemented specifically for the HEVC encoder, whereas others are, inherently, incompatible with it. Because after decades of research on image and video compression, DCT has reached the point that it is acting better for quality than wavelet-based coding. Thus, the methods which were introduced based on this fact are not useful for HEVC and future trends. Ac-

cording to Wang and Bovik (2005, p. 3), the major motivation is "high-frequency information redundancy exist on the peripheral regions, thus more efficient image compression can be obtained by removing or reducing such information redundancy." Hence, the required bandwidth is reduced as well. Furthermore, using foveation helps to keep the quality of the attraction regions even in noisy channels. However, this fact depends on how well the used foveation technique tolerates the transmission errors. Moreover, HEVC works well in compressing fixed areas in the video such as the fixed background even in low available bit rates. This is because HEVC encodes the fixed areas with the largest CU size, given to the encoder (see Section 2.2) in the first frame with almost perfect quality. Then, HEVC skips these areas unless those areas change. Hence, HEVC's strength matches with our selected foveation method and they work well together.

1.3 Proposed Method Outline

In our method, we are given a video input and a tool that can provide us with fixation points for each frame. Then we foveate each frame and the foveated video is given as an input to the video compression encoder. After transmission of the compressed video, the foveation parameters and the fixation point(s), the video is decompressed by the video coding decoder. Finally, the video is defoveated according to the transmitted parameters. Figure 1.3 summarizes the design of our proposed method and how it can be combined with the other two subfields (see section 1) to become a complete application.

In this work, as detailed in Section 2.3, we manually selected the fixation points. The video compression standard that we used is HEVC which is detailed in Chapter 3.

The main focus of this work is on improving the existing fovea compression method and empirically studying the impact of the foveation parameters on the compressed video quality.

1.4 Thesis Contributions

In this work, we propose and analyze a fovea-based perceptual video coding method. The idea is to incorporate characteristics of the HVS (foveation) in a standard-independent way, and we provide empirical evidence showing that this strategy does improve compression without significant degradation of the perceived quality compared to the original video. The existing foveation methods used in image and video compression have different advantages and disadvantages which makes it hard to use them for most of the applications. In Section 3.7 more details about the comparison between the proposed method and the existing ones can be found.

This thesis also presents an evaluation of our study in terms of different objective and subjective metrics. This study better informs us about how well or poorly the objective metrics are capturing perceived video degradation. In our subjective tests, users are asked to watch some videos resulting from the proposed and conventional compression method and judge the quality of the videos.

Our method works as a pre-processing step to video encoding and a post-processing step to video decoding. This reduces the input size to the encoder;

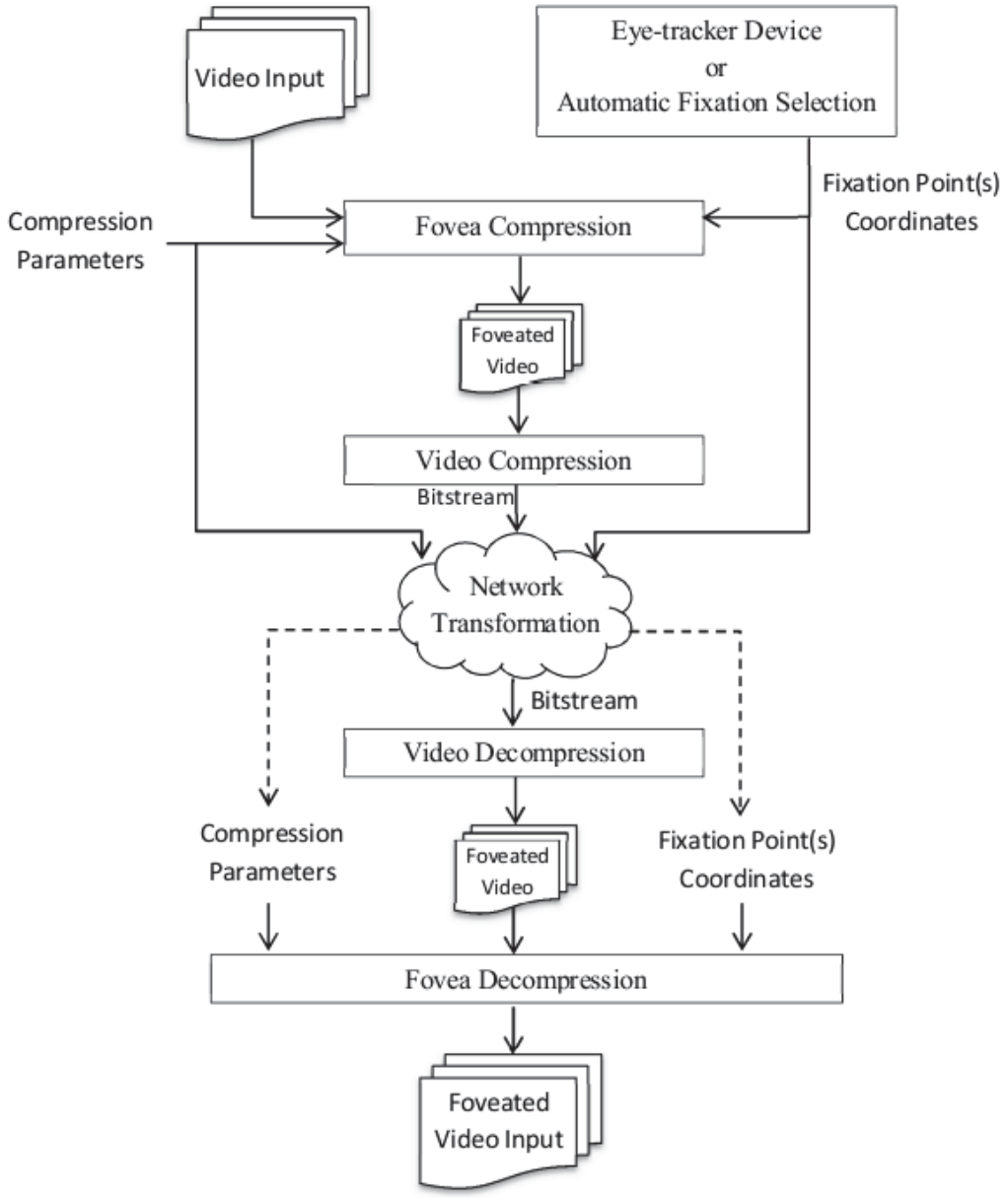


Figure 1.3: Proposed method diagram

therefore, it can reduce the amount of data transmitted to the decoder and the overall computational cost and execution time of encoding/decoding. However, as the quality is more important to the user than the size, we evaluated our method by providing fixed bit rates to evaluate the method's perceptual quality improvement in the ROIs of encoded videos. In addition, this method does not send much unnecessary information (peripheral area) to the encoder to compress; and it completes the peripheral areas in the decoder by interpolation. This part addresses the issue of video compression standards that take a lot of time to encode video data.

Moreover, as an improvement over existing fovea-based methods, our technique produces less visible artifacts and, though standard-independent, videos that are favorable for compression by HEVC, largely regardless of their contents. Our method has a potential to be completely independent from the video's content (this work is just presented for a single fixation point, but it can be easily extended to multiple fixation points). The improved foveation method was selected from an image compression technique which is suitable for use with video compression standards.

The method is independent of how fixation points are selected for each frame of a video, so that it supports both fixed and movable fixation points. Also, quick movements may cause artifacts in HEVC compression, an issue solved by our method. Our experiments show that our method can handle quick movements, moving camera/scene better than HEVC.

1.5 Thesis Outline

This thesis is organized as follows. In Chapter 2, we have a review of previous works on different foveation techniques, a brief introduction about HEVC based on our usage in this thesis, and the review of available techniques for selecting fixation points. In Chapter 3, the improved foveation method is presented. The chapter also describes another foveation method which is compared with the proposed method. Chapter 4 shows the objective and subjective tests on the proposed fovea method with the choice of HEVC as a video compression for 9 different sequences. The results are compared with the state-of-the-art video coding standard (HEVC). Finally, in Chapter 5, we summarize our method and provide some suggestion on how to improve the proposed method and what can be done as future work in foveation video coding.

Chapter 2

Literature Review

2.1 Literature Review

In this thesis, we have used "fovea," one of the HVS features. By applying foveation in a video, one can preserve the quality of the video in the attraction areas (around fixation points) in order to produce higher perceptual quality.

In this chapter, first we describe different foveation techniques according to their functionality, following the survey of Wang and Bovik (2005). These methods are called geometric, filtering-based, multi-resolution, and wavelet-based foveation. Then, we overview the latest video compression standard, HEVC. We conclude this section with a review of the different methods for generating fixation points.

2.1.1 Geometric Methods

The base of this approach is to use the geometry-foveated retinal sampling. In this non-uniform sampling, the image is transformed to the spatially-adaptive coordinate system, which is called "foveation coordinate transform" (Wang and Bovik, 2005). In other words, this foveation transform maps pixels into another image so that regions farther from the *fixation point* (the center of the region where the user is gazing at in the image) become more compressed (In this thesis, the vicinity of the fixation point will be treated as the ROI).

This geometry has been employed in different methods:

- direct transformation of an image to a non-uniform coordinates,
- the Superpixel method,
- non-uniform subsampling.

In the first approach, the foveation transform is applied to an image which is then mapped to new non-uniform coordinates, called *curvilinear coordinates* (Lee et al., 2001). Figure 2.1 shows an example of this transformation. The re-transformation of this transformed image is a foveated image. The downside of this method is that the transformed image grid is non-integer while the pixel locations are integer. In both transform and inverse transform, interpolation and re-sampling are needed. To solve these difficulties, Basu and Wiebe (1998) introduced two techniques, which

Figure 2.1: Original image (left) and transformed image to curvilinear coordinates (right) (Source: Wang and Bovik (2005))

we discuss in detail in Section 3.2.1. These solutions have been studied and improved upon in this thesis in order to extend their use to video compression standards. The improvements increase the quality of compressed videos and take advantage of existing video compression standards to better reconstruct videos. As these methods produce rectangular images instead of curvilinear images, they are compatible with the standard encoders' input.

The second approach, the *superpixel* method (also, referred to as a resolution grid), has been studied by Wallace et al. (1994), Bandera and Scott (1989), Kortum and Geisler (1996), Camacho et al. (1996), and Tsumura et al. (1996). A superpixel consists of some screen pixels with the same assigned value in the compressed image (Kortum and Geisler, 1996). This technique fills the superpixels with the average value of pixels from the original image which are grouped into a superpixel. The retinal sampling is the base for setting the size of each superpixel.

In Wallace et al. (1994) a *logmap* (also called *log polar* or *log spiral*) transform has been introduced as

$$w = \log z + \alpha. \tag{2.1}$$

where z is the original coordinate of a pixel (which is positive), and α is a constant. Using this equation, z will be transformed to a w coordinate. Both z and w are complex numbers. In this foveated sensory approach, first the rectangular image is transformed to an inverse logmap image using a superpixel look-up table (see Figure 2.2) and then by applying (2.1), the logmap image is created. Due to the number and shape of superpixels, applying changes to this look-up table is costly. So this method is unsuitable for moving fixation points (Wang and Bovik, 2005).

In Kortum and Geisler (1996) a square grid is defined based on the retinal sampling foveation for superpixel methods, which is more practical because it is hardware-independent. Each superpixel is filled with the average value of pixels located in the superpixel area in gray scale format. Bandera and Scott (1989) and Camacho et al. (1996) also used a square grid. A multiple-stage superpixel was introduced in Tsumura et al. (1996) where each stage has different block sizes. Therefore, the blockiness artifact is unavoidable in this method.

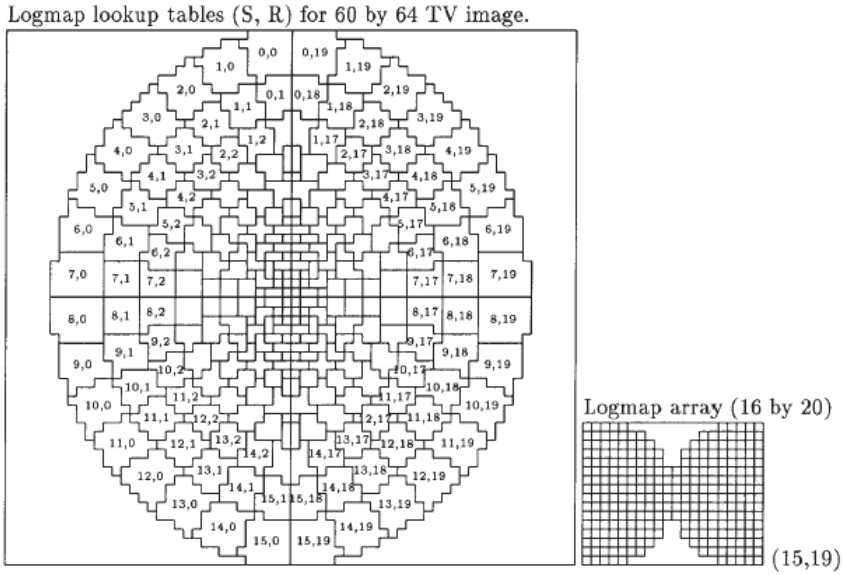


Figure 2.2: Look-up table for the logmap superpixel transformation. This logmap has 16 spokes (S) and 20 rings (R) which are used for indexing (S,R). (Source: Wallace et al. (1994))

The main issue of the superpixel methods is that they produce some discontinuity artifacts in the resulting image, which requires some post processing (such as blending) to fix it. This issue adds more complexity to the method. Moreover, for different fixation points, a new superpixel representation should be constructed from the original image, which increases the computational costs/complexity of the methods.

The third approach is using non-uniform subsampling to take the foveated retinal geometry into account, i.e. the foveated image will be reconstructed by setting the pixel values in the retinal sampling positions (foveating the image). This method was introduced by Kyuel et al. (1999). After reconstructing the subsampled image, which is circular, B-Spline interpolation is used to fill the missing pixels. However, due to the aliasing artifacts introduced, applying a Gaussian filter is necessary, which makes the method more costly. Basu and Wiebe (1998) developed a system based on an idea similar to subsampling for low bandwidth video conferencing. Their method is simpler compared to method of Kyuel et al. (1999), due to employing Cartesian coordinates instead of polar coordinates when subsampling pixels, as well as using simpler interpolation to fill missing pixels after the inverse transform. Also, they have extended their foveated image compression method to MPEG video compression (Basu and Cheng, 2001).

2.1.2 Filtering-Based Methods

Other approaches for foveation, studied by Wang and Bovik (2005), are filtering-based methods that control which parts of the image have higher or lower quality (according to retinal sampling). This control is based on the retinal sampling of the HVS and the method applies a finite number of low-pass or band-pass filters to

the image, with each filter having different frequency effects. Next, the resulting images are merged to form an image with retinal sampling on the fixation point, the foveated image. In the merge process, the low-pass and band-pass filter results should be properly aligned with each other, which is not an easy task. In addition, in designing filters one should consider the domain type (spatial or frequency), as well as the trade-offs between different designs and their effects on the complexity of the implementation. These decisions have a direct impact on the result. By having a low local bandwidth, this method is computationally costly due to repeated filtering, even though it can produce good-quality foveated images.

Sheikh et al. (2003) implemented and compared a filtering-based foveation method with two different transforms; wavelet and DCT. They designed a real-time codec-independent method that must be used as a preprocessing step in the encoder side to provide a foveated image as the input of the encoder. Therefore, the method provides a foveated image for the encoder from the beginning, in spite of geometric methods which require making an intermediate curvilinear image, thus there is no need to apply any changes to the decoder of video compression for retransformation. However, the issue of this design is that the whole foveated image is encoded uniformly, i.e. ,in encoding and decoding time there is no difference between the ROI and peripheral parts. Hence, the resources of encoder and decoder of conventional coding standards (bandwidth, CPU, memory, etc) are assigned equally to different regions of foveated images regardless of their priority, unless the method designed a dependent encoder-decoder based on the foveation model in order to solve this issue.

Sheikh et al. (2003) concluded that their DCT implementation for foveation is less complex than the spatial-domain foveation algorithm, even though spatial-domain foveation compresses more and also produces less blocking artifacts compared to DCT foveation. This is because DCT is applied to small blocks in the image, so the foveated encoder input has blocking artifacts before compression, to which the natural blocking artifacts of common video compression standards are added. Thus, the decoupling of the inner transformation techniques (in codecs) and the filter-based methods results in lower quality results at a higher computational cost (Wang and Bovik, 2005).

Another filter-based foveation video coding has been used in Lee et al. (2001) to foveate the image. They described a novel optimal rate control algorithm according to the fovea retinal properties to maximize the foveal-signal-to-noise ratio (FSNR) (see Section 4.1). This new design helped to have better compression compared with conventional coding due to eliminating high frequencies from peripheral regions (Wang and Bovik, 2005).

2.1.3 Multi-resolution Methods

The multi-resolution method is a combination of the two previous methods. It generates copies of the original image in different sizes, with different scales, using down-sampling or other geometric transformation while applying a filter (such as low-pass filter) to each scaled image. In contrast to the geometric methods, this technique does not need any special indices griding (mapping pixel values to a special grid), blending, interpolation, or superpixels to foveate the image due to uniform down-sampling. As a result, the program runs faster, and indexing is

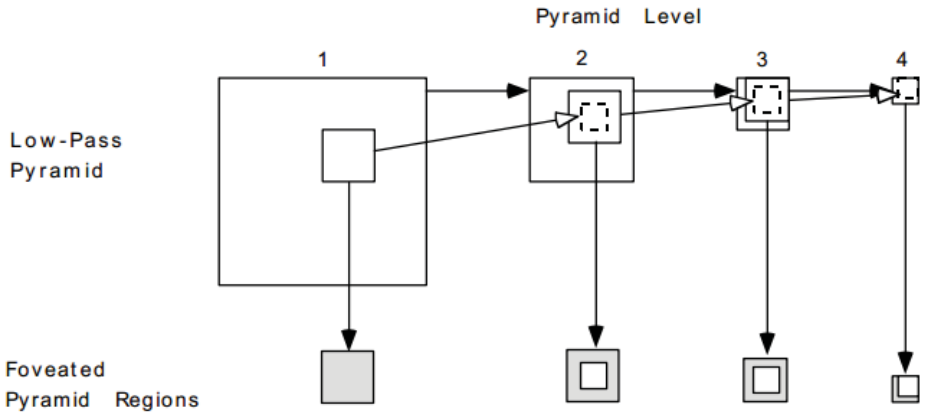


Figure 2.3: Multi-resolution pyramid model (Source: Geisler and Perry (1998))

easier (Wang and Bovik, 2005). But, merging the different scaled images without having artifacts on the borders is not possible. Thus, blending and interpolation have been used as a post processing step in some techniques (Geisler and Perry, 1998). Also, because of the different resulting scaled images, more space and computation are needed per input image, even though the amount of processing drops as the size of the down-sampled image decreases.

Geisler and Perry (1998) proposed a real-time multi-resolution technique in which a Laplacian pyramid, introduced in Burt and Adelson (1983), is used along with down-sampling and regional foveation. Figure 2.3 shows the schematic of this method. In order to find the foveation parts in the image, this technique used a visible contrast threshold function CT satisfying

$$CT(f, e) = CT_0 e^{\left(\alpha \cdot f \frac{e+e_2}{e_2}\right)}, \tag{2.2}$$

where f is a spacial frequency(cycles/degrees), e is the retinal eccentricity (degrees), CT_0 is the minimum contrast threshold, α is a constant for spatial frequency decay, and e_2 is a model parameter (see Section 4.1 for more details about the values of CT_0, α, e_2).

Perry and Geisler (2002) used a similar technique with different steps to create foveated images. First, they down-sampled the input image using the same Laplacian pyramid as in Burt and Adelson (1983) and applied a Gaussian Filter to blur each scale. There were six stages for most applications. Then, each image is up-sampled and interpolated in order to be blended with the image in the previous (larger) scale according to an attention map centered at the fixation point (also known as *cut-off frequency* or *resolution map*). This algorithm is able to produce high-quality foveated video which is good for special applications such as videos with text. However, creating each foveated image has significant space and computational costs, due to the different pyramid levels.

In You et al. (2014) a perceptual-based foveal imaging model is presented. This model has an improved attention model generated by combining different existing attention models. Then based on this composite model, they predict the fixation points. The foveation model proposed by Geisler and Perry (1998) is used in or-

der to reduce the resolution in the peripheral areas based on the visual attentional information described by their new attention model.

Chen and Guillemot (2010) present a foveation model combined with the spatio-temporal just-noticeable-distortion (STJND) model for the H.264/AVC video coder. This paper extracted the visibility threshold of each pixel in the image according to the HVS sensitivity to luminance contrast and spatio-temporal masking effects. Then the foveated JND model uses the relationship between visibility and eccentricity to provide the information for *macro-block* (MB) (see Section 2.2.1) quantization adjustment. They keep the MB distortion lower than the noticeable distortion threshold by imposing a constraint in the rate-distortion optimization. By applying this method, the perceptual quality of the compressed video can be improved. This work can be used in interactive video communication such as video games and eye-tracking applications. Chen and Guillemot (2010) do not mention the complexity of their algorithm or whether it is useful for any handheld devices. This method needs to be modified in case one wants to apply it to HEVC, because HEVC uses a new partitioning scheme instead of MB (see Section 2.2).

2.1.4 Wavelet-based Methods

The wavelet transform has been used in many image processing and video compression standards. This transform has been studied for many years to see if it can be a good replacement for DCT in image and video compression. However, this approach was not successful due to some practical issues of the wavelet transform (Garrett-Glaser, 2010). As an example, Inter-frame (see Section 2.2) wavelet-based methods were suggested during the development of MPEG *scalable coding* in 2004. Scalability is a solution in some coding standards that can choose how much data to send to the user based on his/her preferences, application, and network condition. For this solution, some subset bit streams are created beside the main video which contain lower spatial or temporal resolution or lower quality video. Each subset can be decoded beside the main stream and adds more quality to the final reconstructed video. By dropping some of these subset bit streams, scalable coding can handle different support for various bit rates (Schwarz et al., 2007). Even though the wavelet schemes are more flexible for scalable coding and it benefits from large-scale redundancies in the image, its visual quality is significantly lower due to blurriness introduced by the wavelet transform (Garrett-Glaser, 2010; Hang et al., 2010). For this reason, unlike the previous standards such as H.264/AVC, HEVC does not use wavelet transform (see Section 2.2 for more details).

Several works investigate foveation techniques in wavelet-based video compression (Lu et al., 2001), (Wang and Bovik, 2001), (Wang et al., 2001), and (Chang and Yap, 1997). Wang and Bovik (2001) introduced a wavelet-based foveation method in which a foveated weighting model tries to use HVS properties to remove high-frequency redundancies from peripheral regions. This embedded foveation technique controls the order of the generated bitstream according to the available bit rate, the HVS properties and values of the wavelet coefficients. This employed foveation technique is the same as in Geisler and Perry (1998) (2.2). This method has been improved to scalable video coding in Wang et al. (2003).

These foveated wavelet-based techniques cannot be applied to the new video

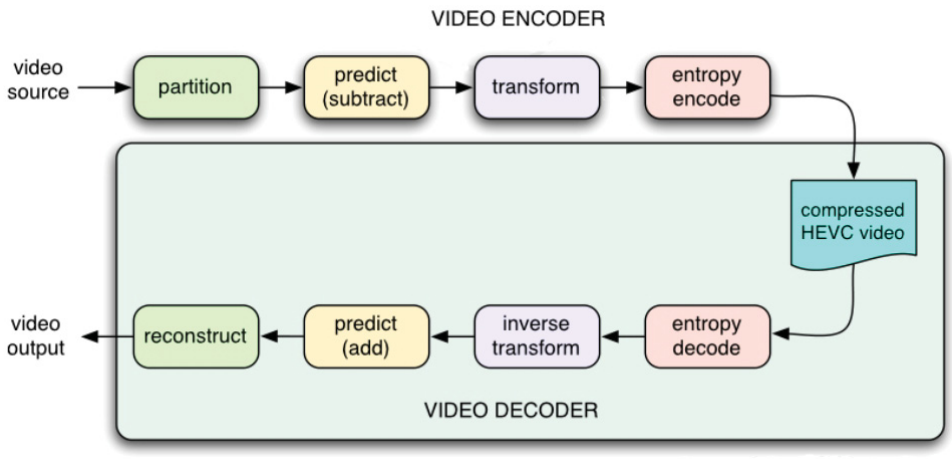


Figure 2.4: Block diagram of HEVC (Source: Richardson (2013))

compression standard, HEVC. Being dependent on one specific video compression standard makes it hard to remain up up-to-date with the state-of-the-art compression standard.

2.1.5 Other Usage of Foveation

In Sanchez et al. (2004) a prioritized ROI Coding method in JPEG2000 image compression has been introduced which considers the concept of foveation for each ROI. These transmitted packets are prioritized by applying Gaussian distribution based on the ROI location, retina sampling and the available bandwidth. Therefore, the resulting images are foveated, and by having more bandwidth more data can be added to the peripheral area in order to increase the quality.

2.2 Overview of HEVC

HEVC is the state-of-the-art video compression standard first released in January 2013. In this video standard, the goal is to increase compression while preserving the video quality. HEVC can generate videos with quality similar to those compressed using H.264/AVC, (the previous video compression standard) while compressing more. HEVC has the potential of achieving 50% bit rate reduction. However, till now they could only achieve up to 35.4% depending on the video content. Also, if HEVC and H.264/AVC have the same available (fixed) bit rate, HEVC produces better quality results. Ohm et al. (2012) present a detailed comparison of HEVC and other video coding standards, including the previous state-of-the-art, H.264/AVC. Sullivan et al. (2012) present an overview of HEVC, which we summarize as follows. Also, Figure 2.4 demonstrates the current scope of HEVC.

2.2.1 Partition

In video compression, the input frame is split into smaller blocks for further processing such as prediction or transformation. MB, a group of blocks, is used in the previous video coding standards such as H.264/AVC. MB typically consists of 16×16 pixels of luma components and two corresponding 8×8 chroma components, which are broken down to smaller blocks.

In HEVC, the video input frames split into smaller units, called *Coding Tree Units* (CTUs) of size $M \times M$ where $M = 16$, 32, or 64. This size can be set at the encoder side according to the application needs and limitations. Also, the size can be larger than traditional MB (16×16, 32×32, or 64×64). Each CTU consists of a luma coding tree block (CTB) with the size $M \times M$ along with two $M/2 \times M/2$ CTBs for chroma components and associated necessary syntax which is require for processing the CTU. Each CTU can be partitioned into smaller units, called coding units (CU). Each of these CUs has one $N \times N$ luma coding block and two corresponding $N/2 \times N/2$ chroma CBs with associated syntax, where N is between 8 and M (Ohm et al., 2012). However, a problem may arise if the width or height of the input video is not an integer multiple of the specified chroma subsampling. The only chroma sampling which is supported in the current HEVC design (HM Version 11.0 3 Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP and 11 (2012)), is $4 : 2 : 0$ (Sullivan et al., 2012).

2.2.2 Predict

In HEVC, like the previous video coding standards, there are two different prediction methods: *Intra prediction* and *Inter prediction*. These predictions are used for reconstructing a frame from the same frame only (Intra prediction) or one or more previous or forward frames (Inter prediction). The frames that only use the Intra prediction are called *Intra frames* (I-frames); those frames that only use Inter prediction with previous frames as references, are called *Predicted frames* (P-frames), and those which use Inter prediction with both previous and forward frames to achieve the highest compression are called *Bi-predictive frames* (B-frames). Video encoders compress video data based on the given structure of the encoding pattern, i.e. the sequence of the type of frames (I-, P-, or B-frames) is included in the structure. Each frame is encoded based on the chosen frame type, and next in the encoder, each frame is reconstructed the same way as the decoder procedures. Then, by subtracting the original and reconstructed frame, the residual values are computed for transform and quantization parts.

For prediction, each CU is partitioned to one or more *Prediction Unit* (PU). In Intra prediction, each PU is coded by using 3 modes: 1) the DC prediction (setting a mean value of decoded PUs boundaries to the left and top of the current PU, for the current PU); 2) planar prediction (filling the current PU with an average value of pixels in the decoded PUs on the top and left side of it; 3) directional prediction with 33 different directions (extrapolating from neighboring pixels).

In Inter prediction, the motion compensation component is responsible for finding the best match from the reference frame(s) and calculating the motion vector(s). In HEVC, *fractional sample interpolation* is applied to all of the reference frames so that their resolutions are increase and be ready for more accurate inter prediction.

This interpolation calculates new values between each of the two pairs of immediate neighboring pixels to increase the frame resolution.

2.2.3 Transform and Quantization

The residual values calculated with the prediction components, are required to be transformed and quantized so that they can be sent to the decoder side. For this purpose, HEVC uses transform units (TUs). Each CU in the frame breaks to four or more TUs. Each TU uses integer bases functions similar to DCT. In addition, a transform derived from discrete sine transform is alternatively used for 4×4 luma transform blocks (Sullivan et al., 2012).

2.2.4 Entropy Coding

Context Adaptive Binary Arithmetic Coding (CABAC) is an entropy coder which is selected for HEVC in order to encode the header data and the resulting coefficients from transform and quantization components. The header data consists of information for partitioning the frame, the prediction modes and motion vectors. CABAC is able to preserve the quality of the video better than the other entropy coders.

2.2.5 Parallel Processing

One of the other goals of HEVC is to increase the processing time by employing parallel processing. For this purpose, HEVC introduced three different concepts: *slice*, *tile*, and *Wavelet Parallel Processing (WPP)*. The frame can be broken down to different sizes of slices. Each slice consists of different CTUs and it is independent from other slices, i.e. . each slice can be encoded and decoded independently. Tiles are a new concept defined by HEVC which is an independently decodable parts of the frame which is rectangular. The last concept is WPP which is introduced by HEVC for processing each independent slice or tile with different threads in parallel.

2.3 Selection of the Fixation Points

For each frame, we need to have a fixation point so that we can define the ROI, whose position depends on the video content. In this thesis, in order to be sure of the accuracy of the proposed method, we are not selecting the fixation points automatically, but this selection is an important part of fovea-based encoding, so this section is included to cover all the necessary parts. There are many ways of selecting the ROI, and in the following, we mention some existing techniques, based on the overview by Itti (2004).

For example, eye-tracking devices to record the eye position of the human observer can be efficient for this selection on noninteractive video compression. The observers will often not notice degradation of the video signal as long as they are gazing at the tracked part by the tracking devices due to the natural degradation caused by their visual system (Kortum and Geisler, 1996; Lee et al., 1999). The eye-tracking foveation system was developed and tested in the University of

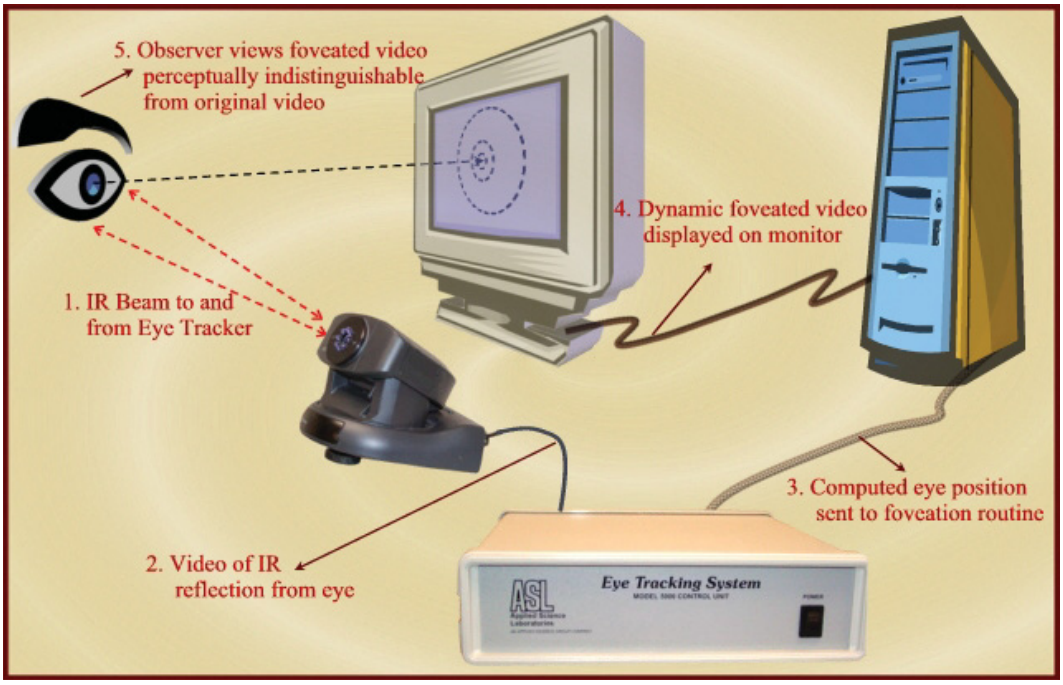


Figure 2.5: Eye tracking (Source: Umesh Rajashekar and Bovik (2001))

Texas (Umesh Rajashekar and Bovik, 2001) and a filter-based foveation method (see Section 2.1.2) was used to foveate videos (Figure 2.5). The fixation points were detected in real-time from the user's eyes gaze. The user could not see the degradation in peripheral areas due to the real-time foveation in the video which works as if the uncompressed video is being watched. Nevertheless, we may not have eye-tracking devices or the video stream may be shown to more than one person. Tracking and using all the viewer's eye positions should be used for finding possible ROIs (Stelmach et al., 1991). There are some techniques which use the HVS properties for selecting important parts of an image. Osberger and Maeder (1998) employed some low-level factors such as contrast, color (which are perceived as spectral frequencies (Cox et al., 2008)), motion, region size and the object shape). Also, Marichal et al. (1996) used the fuzzy logic system (Cox, 1994) to take an image semantic into account while finding the important image regions and Chen and Guillemot (2010) used skin color detection for this selection. Another automatic method is neurobiological attention model of Itti (2004) which is useful for detecting the ROI in most videos regardless of their content. In this model, a saliency map will be computed to show the perceptible visual areas in the image by processing the image into multiscale low-level feature maps (color, flicker, intensity, orientation, and motion) and then summing the result of these maps to the unique scalar saliency map. The aforementioned methods are compared in Itti (2004).

Chapter 3

Foveation-Based Video Coding Method

3.1 Chapter Overview

In this chapter we describe our proposed foveated-based video coding method. In our method, for each frame in the video input, we first apply the foveation compression, which we explain in Section 3.2. Next, the resulting foveated video will be given to the encoder of a video compression standard (in our case it is HEVC). The next step is when the data want to be decoded and displayed. In the decoder side, we apply the reverse procedure of the encoder. It means that first the compressed data is decompressed by the video standard decoder and then foveation decompression is applied to the output of the video compression to obtain the reconstructed video. Figure 3.1, shows a diagram of the proposed method.

In the following sections, we will first describe two foveation methods (Section 3.3 and 3.4). Also, we will describe the improvements which we add to the selected method (Section 3.3.1). Then, in Section 3.5, we will explain why we continued our work only on one of the two methods. Moreover, we provide an extension to this technique to support asymmetric objects' shapes with only single fovea. We conclude the chapter by summarizing the differences between our method and the related work.

3.2 Foveation Compression

Our foveation compression method is improving the Variable Resolution (VR) image transform, which has been studied by Basu and Wiebe (1998), Basu et al. (1993), and Basu and Licardie (1993). In our work, we first considered two foveation methods: Cartesian Variable Resolution (CVR) (Section 3.3.1) and Multiple Scaling Factors (MSF) (Section 3.3.2). After implementing and comparing the results of these methods and finding the strengths and weaknesses of them, we continued our work by improving the CVR method due to its simplicity and speed. The MSF method is more complex and produces few artifact lines in the resulting image (Figure 3.7b), so, we did not investigate it deeper.

In this thesis, we will present an improved foveated method for the video coding. The proposed foveated method can improve video quality both objectively

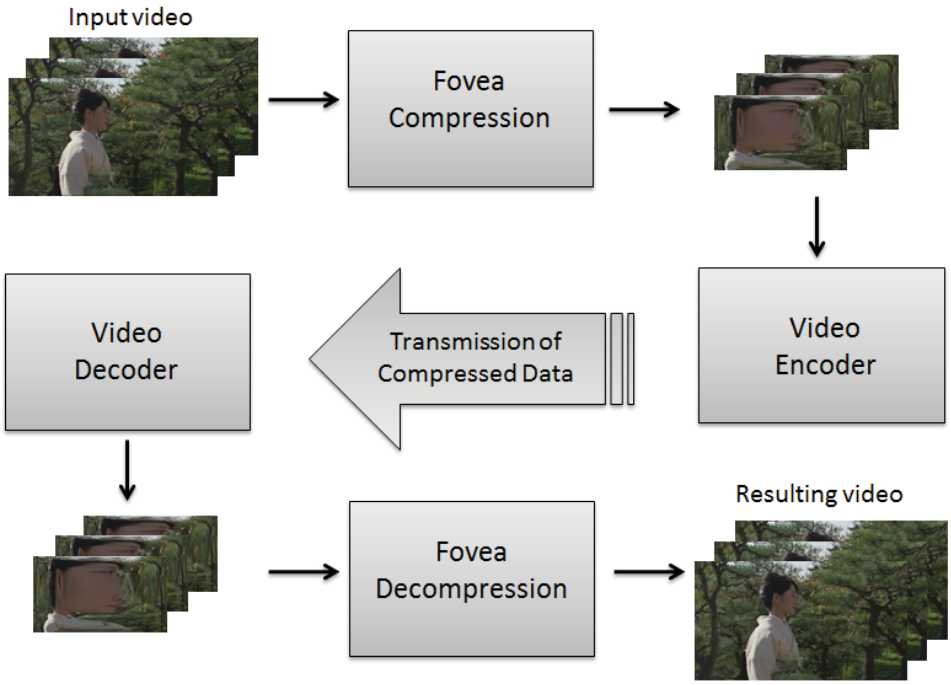


Figure 3.1: Block diagram of the "Foveation-based video coding method (Proposed method)"

and subjectively by removing the blocking artifacts in the peripheral area and the shifted effect in the resulting image effect due to the integer sampling employed by Basu and Wiebe (1998). Moreover, this foveation method provides rectangular images for each frame which are supported by the video compression standards inputs. This method does not need any filter for blurring; e.g. automation model of Itti (2004) uses the Gaussian filter to blur the low priority areas and send the whole frame to the encoder which has lot of redundancies in the peripheral area. Itti (2004) mentions this issue causes the appearance of new objects from peripheral parts while the fovea regions move, or if the blurred area changes due to changes in peripheral regions, the conventional encoders, designed to detect all the changes in the frame uniformly, assign bits for these detections, even though the fovea region is fixed. In this work, we decrease the size of the video in order to reduce the low priority regions' size, while emphasizing the attentional regions. Additionally, the peripheral area is still visible for conventional encoders, i.e. the encoders can encode similar regions in frames more efficiently than some blurred regions which are not similar.

3.2.1 Foveation Background

The transformation proposed by Basu and Wiebe (1998) uses the geometry of the foveated retinal sampling to spatially transform the coordinate system (Wang and Bovik, 2005). By applying the VR transform, sampled pixels will be mapped from one polar coordinate system (r,θ) to another (v, θ). The resulting image may have smaller size compared to the original one, which means that several pixels from the

original image may map to the same pixel position in the resulting image. In the compressed image, the VR transform controls how far we want to put each pixel from the fixation point. The VR transform is defined as

$$v = \ln(r \cdot \alpha + 1) \cdot s, \tag{3.1}$$

where r and v are the positions of the corresponding pixel in the original polar coordinates and the transformed coordinates, respectively. The parameter α is used to control how much we want to foveate the image; i.e. a high value of α will foveate the image more; it means that the foveated region has better quality while the peripheral region has lower resolution. For having a less foveated result, the α value should be set to the smaller amount so that the resolution of the foveated and peripheral region are close to each other. The parameter s is a scaling factor and can affect the whole compression ratio over the image. s is defined as

$$s = \frac{v_{\max}}{\ln(r_{\max} \cdot \alpha + 1)}. \tag{3.2}$$

where $v_{\max}$ and $r_{\max}$ are the maximum value of v and r, respectively. The scale factor value obtained from (3.2) guarantees that (3.1) is satisfied for $v_{\max}$ and $r_{\max}$.

The inverse of the transform of (3.1) is

$$r = \frac{e^{\frac{v}{s}} - 1}{\alpha}. \tag{3.3}$$

3.3 Foveated-Based Video Encoder

As discussed in Basu and Wiebe (1998) and Wang and Bovik (2005), the result of the VR transform is not rectangular and it maps the integer indices to the non-integer ones. Basu and Wiebe (1998) suggested two methods to overcome this. The first method is called Cartesian Variable Resolution (CVR) which uses Cartesian coordinates in formulae (3.1) to (3.2). This transformation is significantly simpler in terms of computational complexity. The second method calculates a scaling factor for each angle θ in polar coordinates, with their individual maximum distances, using (3.2). We call this method Multiple Scaling Factors (MSF). The result of the MSF transform will be rectangular. We describe both of these methods in the following sections.

After providing the foveated video sequence, the sequence will be compressed by the HEVC and H.264/AVC encoder. As this transformation can reduce the size of the picture frames, the high execution time of the HEVC and H.264/AVC encoder and decoder are significantly reduced.

3.3.1 CVR Transform

The CVR transform can control the size of the foveated compressed frame by using a compression ratio parameter. The transformation corresponds to a linear scaling of the image area while preserving its aspect ratio. The new dimensions for the

compressed image are calculated as follows:

$$w' = \text{round}\left(w \cdot \frac{\sqrt{1-c}}{u}\right) \cdot u, \tag{3.4}$$

$$h' = \text{round}\left(h \cdot \frac{\sqrt{1-c}}{u}\right) \cdot u, \tag{3.5}$$

where w', and h' define width and height of the compressed picture frame and w and h are the width and height of the original picture frame, respectively. Parameter c is the compression ratio. It can be set to 0 to have no size changed. As its value becomes closer to 1, the size of the picture frame will be smaller and so we will have more data loss. The factor u must be set according to the video coding standard that one wants to use. The "$\text{round}\,()$" function is used to produce the integer results so that (after multiplying its output with u,) w' and h' are multiples of u. This is a requirement of some video compression standards. In HEVC, e.g. , we have $u = 4$, because w' and h' have to be integer multiples of the minimum CU size so that the compressed output can be given as an input to the HEVC encoder (see Section 2.2). Using the *conformance mode*, one of the HEVC encoder's parameters, is another way of satisfying this myltilicity requirement. This parameter specifies the cropping and padding values corresponding to the width and height of the input video so that the frame dimensions are multiples of the minimum CU size.

Figure 3.2 shows reconstructed frames for different values of c. As this figure shows, by having a larger c the blur of the peripheral area will be increased due to data loss. But this blur does not only depend on the parameter c. Later in this section, another parameter, α, will be introduced.

In this work, we select the fixation points in the original picture frame and manually tracked them in the video sequence to be able to evaluate the accuracy of our method independently from the ROI selection method. (For automatic selection of fixation points see section 2.3.)

In the first step, the CVR compression transformation maps the pixels of the original picture to the compressed one. We define the coordinate origin of the picture frame on the upper left side. For a given picture frame with the fixation point at (x_f, y_f) position, we define a fixation point of the compressed picture as

$$x'_f = \text{round}\left(\frac{w'}{w} \cdot x_f\right), \tag{3.6}$$

$$y'_f = \text{round}\left(\frac{h'}{h} \cdot y_f\right). \tag{3.7}$$

where x'_f and y'_f are the positions of the fixation point in the compressed image.

By having w' and h', the compressed picture boundaries can be defined. Each value of a pixel (x', y') in the compressed frame will have the value of a pixel from the original frame. The corresponding (x, y) in the original picture for each (x', y') is

$$x = \sigma_{dx'} \cdot \frac{\exp\left(\frac{|x'-x'_f|}{s_X}\right) - 1}{\alpha} + x'_f, \qquad \sigma_{dx'} = \text{sign}(x' - x'_f), \tag{3.8}$$

$$y = \sigma_{dy'} \cdot \frac{\exp\left(\frac{|y'-y'_f|}{s_Y}\right) - 1}{\alpha} + y'_f, \qquad \sigma_{dy'} = \text{sign}(y' - y'_f). \tag{3.9}$$

s_X and s_Y are the scaling factors in X and Y axes, respectively. We describe them later on. The parameter α (see Section 3.2.1) controls how the neighboring pixels around the fixation point are expanded. Higher values of α cause more expansion, which blurs the reconstructed image in the peripheral area, i.e. close to the fixation point a pixel gets "stretched" into various pixels, whereas the periphery pixels get "squeezed" into single pixels.

Figure 3.2 shows the effect of α and c on the reconstructed image. The size of the compressed foveated image for each pair of (α, c) can be different according to the value of c. The α parameter controls the size of the ROI and affects the amount of blur in the peripheral. The fixation point of Figure 3.2 is on the person's nose.

As in the VR model, scaling factors are used to control the overall amount of compression in the picture. They are defined so that the pixels with a maximum distance from fixation point map to the compressed picture edges. However, by using (3.8) and (3.9), the non-integer values resulting from the exponential function have to be placed to the valid integer pixel range indices.

The scaling factor, s_X, satisfies

$$s_X = \begin{cases} \frac{x'_f}{\ln(\alpha \cdot x_f + 1)} & \text{if} \quad x' \leq x'_f \\ \\ \frac{w' - x'_f}{\ln(\alpha \cdot (w - x_f) + 1)} & \text{if} \quad x' > x'_f \end{cases} \tag{3.10}$$

Similarly, s_Y satisfies

$$s_Y = \begin{cases} \frac{y'_f}{\ln(\alpha \cdot y_f + 1)} & \text{if} \quad y' \leq y'_f \\ \\ \frac{h' - y'_f}{\ln(\alpha \cdot (h - y_f) + 1)} & \text{if} \quad y' > y'_f \end{cases} \tag{3.11}$$

As we can see from the above equations, s_X depends on whether x' is on the left or right side of the fixation point in the X orientation. Likewise, s_Y of y' depends on whether y' is on the top or bottom part of the fixation point in Y orientation. Hence, each frame will be divided into four quadrants with the origin at the fixation point (x'_f, y'_f). When the fixation point is at the center of the frame, we have just one value for each of s_X and s_Y, due to symmetry.

(3.10) and (3.11) ensure that (3.4) and (3.5) are satisfied by (3.8) and (3.9) when $x = 0$, $x = w$, and $y = 0$, $y = h$.

A resulting foveated frame from the CVR method can be found in Figure 3.5a. Also, Figure 3.5 shows the foveated compressed image and its corresponding reconstructed image from the CVR method.

3.3.2 Multiple Scaling Factors

In the Multiple Scaling Factors transform, we first calculate w', h', x'_f, and y'_f by using (3.4)–(3.7). The way we select (x_f, y_f) is also the same as the CVR method, but the scaling factors are calculated differently. The frame is partitioned into four regions as shown in Figure 3.3 which define the scaling factors. To create these regions, we first connect four lines from each corner of the frame to the fixation

(a) The 1280×720 frame after foveation ($\alpha = 0.8$, $c = 0$), HEVC encoding and reconstruction.

(b) The 1280×720 frame after foveation ($\alpha = 0.8$, $c = 0.5$), HEVC encoding and reconstruction.

Figure 3.2: The resulting reconstructed images for different values for α and c. The image is the frame #133 of *Vidyo4* sequence and the bit rate is 1 Mbps.

(c) The 904×508 frame after foveation ($\alpha = 0.32$, $c = 0.5$), HEVC encoding and reconstruction.

Figure 3.2: The resulting reconstructed images for different values for α and c. The fixation point of this image is on the person's nose. The image is the frame #133 of *Vidyo4* sequence and the bit rate is 1 Mbps.

point, (x'_f, y'_f). We then define a 2D polar coordinate system with the origin at (x'_f, y'_f), then we calculate the angle of each of the four corners counterclockwise ($\theta_i, i \in 1, 2, 3, 4$). These angles can be calculated using the original or the compressed frame. The result will be the same in both cases. Afterwards, the scaling factors will be computed as

$$s_t = \frac{\|(x' - x'_f,\ y'_f)\|}{\ln(\alpha \cdot \|(x - x_f,\ y_f)\| + 1)}, \qquad \text{if} \quad \theta_1 \leq \theta < \theta_2 \tag{3.12}$$

$$s_b = \frac{\|(x' - x'_f,\ h' - y'_f)\|}{\ln(\alpha \cdot \|(x - x_f,\ h - y_f)\| + 1)}, \qquad \text{if} \quad \theta_2 \leq \theta < \theta_3 \tag{3.13}$$

$$s_l = \frac{\|(x'_f,\ y'_f - y')\|}{\ln(\alpha \cdot \|(x_f,\ y_f - y)\| + 1)}, \qquad \text{if} \quad \theta_3 \leq \theta < \theta_4 \tag{3.14}$$

$$s_r = \frac{\|(x'_f - w',\ y'_f - y')\|}{\ln(\alpha \cdot \|(x_f - w,\ y_f - y)\| + 1)}, \qquad \text{if} \quad \theta_4 \leq \theta < \theta_1 \tag{3.15}$$

where s_t, s_b, s_l and s_r are the scaling factors for the top, bottom, left and right regions respectively, and $\|\cdot\|$ is the euclidean norm. Analogously to the CVR transform, each value of a pixel (r', θ') in the compressed frame will have the value of the pixel from the original frame. The transformation does not change θ, so $\theta' = \theta$, while the corresponding r in the original frame can be computed directly from (x', y') as

$$r = \frac{\exp\left(\frac{\|(x' - x'_f,\ y' - y'_f)\|}{s_k} - 1\right)}{\alpha}, \quad k \in \{t, b, l, r\}. \tag{3.16}$$

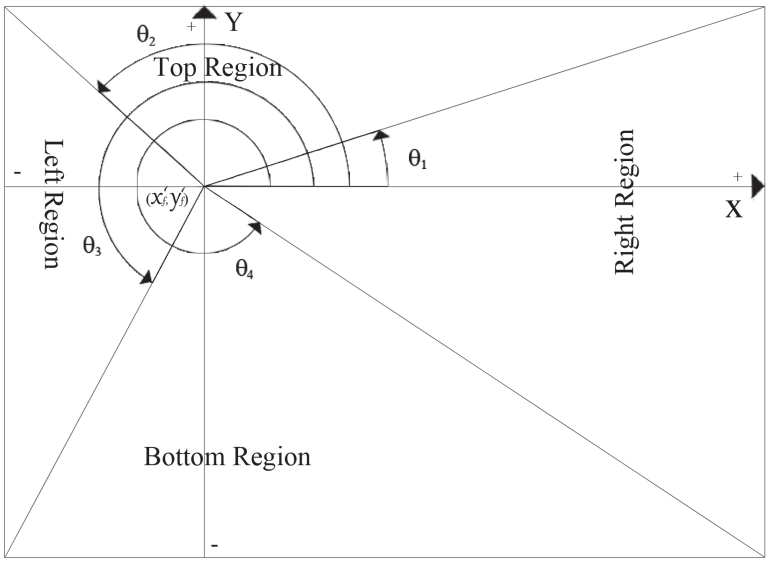


Figure 3.3: Four regions of the frame according to the fixation point, (x'_f, y'_f)). Each line connecting (x'_f, y'_y) to a corner of the image defines an angle θ with respect to the positive side of the X-axis. The pratition is given by the regions between these lines. E.g. , the region between θ_1 and θ_2 is the top region of the frame.

The scaling factor s_k is chosen based on θ and (3.12)–(3.15). Then, we need to convert (r, θ) to Cartesian coordinates, that is

$$x = \text{round}\left(x_f + r \cdot \cos\theta\right), \tag{3.17}$$

$$y = \text{round}\left(y_f - r \cdot \sin\theta\right), \tag{3.18}$$

where we have used that

$$\theta = \arctan \frac{y'_f - y'}{x' - x'_f}. \tag{3.19}$$

From the above, we see that the value of (x', y') in the foveated frame will be the value of (x, y) in the original frame. In case x or y are out of the range of the original frame, cropping will be needed. After foveation, the video will be given to the video encoder.

3.4 Foveated-Based Video Decoder

By receiving the compressed data, the video decoder reconstructs the foveated video sequence. Next, each picture frame is transformed back to its original size. The process of re-transformation is simple. The encoder only needs to send α, c, u, and (x_f, y_f) along with the compressed video data. If the position of the fixation point is different in each frame, we also need to send the frame's corresponding (x_f, y_f) to the decoder.

3.4.1 CVR Re-transform

For this method, s_X and s_Y can be recalculated on the decoder using (3.10) and (3.11). We need to map each pixel from the foveated compressed frame to the defoveated frame of size w and h. (x', y') in the foveated image will correspond to a pixel (x, y) in the defoveated image, calculated according to (3.8) and (3.9). This reverse mapping may leave some gaps (missing pixels) between pixels in the reconstructed frame. We use interpolation to fill these gaps. The interpolation techniques that we used are nearest-neighbor, linear and piecewise-cubic interpolation. The interpolation method can be chosen according to the needs of the user and his system requirements. The nearest neighbor interpolation is fast and simple compared to the other interpolation techniques. However, the visual result of this interpolation has some blocking artifacts. The linear interpolation requires more computational time than the nearest neighbor interpolation. Also, the piecewise cubic interpolation is more complex and needs more memory and computational time than the linear interpolation. The results of the linear and cubic interpolation have better quality than the nearest neighbor one (Gonzalez and Woods, 1992). Figure 3.4 demonstrates the effects of these three interpolation methods on the reconstructed images.

Comparing CVR Method of Basu and Wiebe (1998) and the Proposed Method

Here we are going to explain the differences between our decompression method and the method of Basu and Wiebe (1998).

In the Basu and Wiebe (1998) method, the reverse CVR transform is the inverse procedure of the transform in the encoder part of the proposed method. Each pixel (x, y) in the reconstructed frame will take the value of the pixel (x', y') in the foveated frame, where (x', y') satisfy

$$x' = \sigma_{dx} \cdot s_X \cdot (\ln(\alpha \cdot |x - x_f|) + 1) + x_f, \qquad \sigma_{dx} = \text{sign}(x - x_f), \tag{3.20}$$

$$y' = \sigma_{dy} \cdot s_Y \cdot (\ln(\alpha \cdot |y - y_f|) + 1) + y_f, \qquad \sigma_{dy} = \text{sign}(y - y_f). \tag{3.21}$$

As we can see in the above equations, each pixel in the decoded foveated frame may map to multiple pixels in the original size decoded frame. Also, the result of the above equations will be non-integer values, so they must be converted to integer values. However, as is mentioned in Wang and Bovik (2005), the geometric methods' difficulty (see Section 2.1.1) is mapping non-integer positions to integer grid locations. In method of Basu and Wiebe (1998), the integer index mapping from (3.20) and (3.21) shifts the reconstructed frame to one side (in our case to the right side) and loses pixels at the edge of the frame. Figure 3.6 demonstrates the residual images for method of Basu and Wiebe (1998). The reason is that the mapping from the original image to the foveated image, which is calculated via (3.8) and (3.9), does not match the result of (3.20) and (3.21). It means that if we have the exact pixel of the original frame in the foveated image, we need to put the pixel back in the same location as it was in the original image. However, with (3.20) and (3.21) this may not happen. This issue will cause a problem in pixel-by-pixel evaluations (e.g. if PSNR is used). With the new technique we applied in the decoder side, this issue is fixed. To the best of our knowledge, this shifting effect of Basu's method has not been noticed before. The shifted reconstructed frame from

(a) The reconstructed frame using nearest neighbor interpolation.

(b) The reconstructed frame using linear interpolation.

Figure 3.4: The resulting reconstructed images for different interpolation methods. $\alpha = 0.8$ and $c = 0.5$. The fixation point of this image is on the car. The image is the frame#163 of *ChinaSpeed* sequence. The bit rate is 1 Mbps. The blocking artifacts are visible in the peripheral part of the image 3.4a, however, in the 3.4b and 3.4c the peripheral are smooth. (The rest of Figure 3.4 is continued on the next page.)

(c) The reconstructed frame using cubic interpolation.

Figure 3.4: (Continued. The two images of this part are on the previous page.)The resulting reconstructed images for different interpolation method. $\alpha = 0.8$ and $c = 0.5$. The fixation point of this image is on the car. The image is the frame#163 of *ChinaSpeed* sequence. The bit rate is 1 Mbps.

(a) Foveated 904×508 image resulting from 1280×720 original image, using the CVR method.

(b) Resulting reconstructed 1280×720 image for foveated image in 3.5a. Linear interpolation is applied for this image.

Figure 3.5: CVR method Results. $\alpha = 0.32$ and $c = 0.5$. As you can see in Figure 3.5a, α causes the neighboring pixels around the fixation point to expand. The fixation point is on the person's nose. The bit rate is 1 Mbps. The image is the first frame of *Vidyo4* sequence.

method of Basu and Wiebe (1998) is visible in Figure 3.6a. However, there is no shifting with our proposed method.

Wang and Bovik (2005) mentioned that interpolation and resampling will be needed for transforming and retransforming in geometric methods. However, in our work we just applied the interpolation for the retransformation part which is less complex. In method of Basu and Wiebe (1998) we need to calculate a table of indices with $w \times h$ to reconstruct each frame. This method will fill the missing pixels with the available pixels in the compressed frame. This creates blocking artifacts in the periphery of the resulting image. In our decoder side, we only calculate a table of indices with size w' and h' for (x', y'), where $w' \leq w$ and $h' \leq h$ and w' and h' depend on the value we have chosen for c. Then we need to fill the missing pixels in the reconstructed image using interpolation. In contrast to other methods such as Basu and Wiebe (1998) and Kyuel et al. (1999), after the interpolation there are no noticeable visual artifacts; hence, no post processing is needed.

3.4.2 Multiple Scaling Factors

The decompression process of the MSF method uses s_t, s_b, s_l, and s_r as defined in (3.12)–(3.15) and it is the inverse of the transformation in (3.16). The value of each pixel (x, y) in the reconstructed frame will be the value of the pixel (x', y') in the foveated frame where

$$x' = \text{round}\left(x'_f + r' \cdot \cos(\theta')\right), \tag{3.22}$$

$$y' = \text{round}\left(y'_f - r' \cdot \sin(\theta')\right), \tag{3.23}$$

and $\theta' = \theta$ (see (3.19)) and

$$r' = \ln(\alpha \cdot \|(x - x_f, y_f - y)\| + 1) \cdot s_k, \quad k \in \{t, b, l, r\}. \tag{3.24}$$

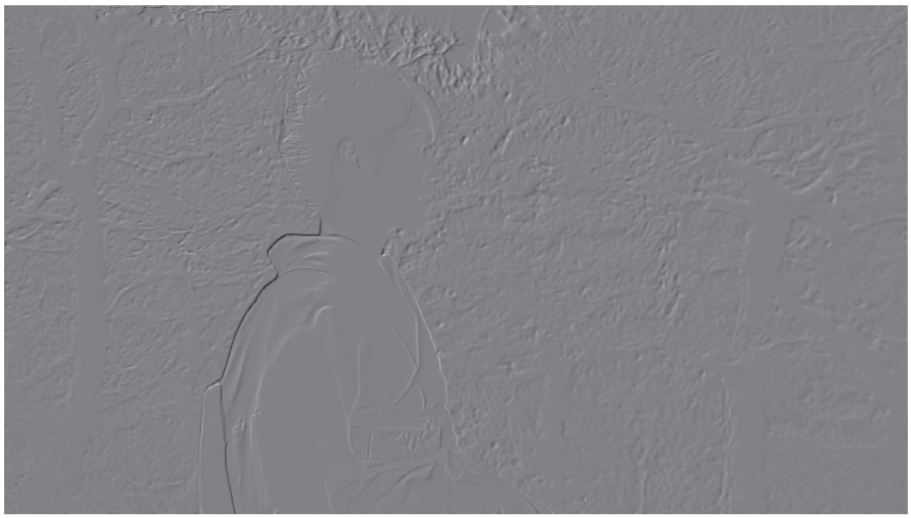
Cropping over the edges of the foveated picture frame may be necessary.

3.5 Comparison of CVR and Multiple Scaling Factors

The CVR method is simpler than the MSF method in terms of computational complexity and time, because in the MSF method we need to calculate scaling factors for all of the pixels separately, while we only use four scaling factors for CVR. The frame foveated by the MSF method can be seen in Figure 3.7a. The discontinuities in the foveated frame (Figure 3.7a) result in some artifacts along the partitioning lines (see Figure 3.3 and Figure 3.7b). Because of this issue, we chose to focus our experiment on the CVR methods.

3.6 An Extension for a More General Transform Function (CVR)

One of the parameters we have used in this work is α, which we can generalize to four different parameters, according to its distance from the four different edges of the frame. With this, we can have arbitrary rectangles as the foveation region, which can be useful for supporting various objects' shapes.



(a) Residual image from original frame and corresponding reconstructed frame resulting from mehtod of Basu and Wiebe (1998).

(b) Original frame. (First frame from Kimono sequence.)

Figure 3.6: Comparison of method Basu and Wiebe (1998) and our proposed method. 3.6b is the frame #1 from Kimono sequence. As you can see in Figure 3.6a, the residual image shows the shifted result in the reconstruction frame. However, no shifting result is caused by our proposed method.

(a) Foveated frame resulting from MSF method

(b) The MSF method reconstruction result with artifact lines

Figure 3.7: The MSF method results

The α values should be selected based on the shape of the object in the ROI or the properties of the ROI. These parameters are defined with the same idea as in scaling factors (see Section 3.3.1). It means that, depending on the position of each pixel in the frame, one of the α values should be selected (α_{x_l}, α_{x_r}, α_{y_t}, and α_{y_b} for the pixels in the left, right, top or below of the fixation point, respectively) for (3.8), and (3.9). Also, the corresponding value of α based on its position (its corresponding edge) should be used in (3.10) and (3.11).

In Figures 3.8 and 3.9, some sample frames are included to illustrate the result of the above generalization. These frames are the result of the foveation technique only; we did not use any video coder to compress them. As you can see in Figure 3.8b, the left side of the fixation point is less foveated compared to the right, and the above side is more foveated compared to the region below the fixation point, due to the choice of α values. This technique is good for an object with asymmetric shape so that ROI encompasses the object. This extension can substitute multiple fixation points on a single object.

3.7 Comparison of The Proposed method and Existing Foveation Methods

The existing foveation methods use different techniques to apply the retina sampling to an image. According to the type of technique used to prodece the foveated image, we can have different effects on the quality of the resulting image and its hardware requirements.

In geometric methods, the image is generally mapped to a curvilinear image (non-integer and non-rectangular), or in some methods it generates a rectangular image by using indexing from the source image to the intermediate foveated one. However, the image grid is integer which causes some loss or blockiness artifacts through mapping by non-integer indices. Therefore, post processing (such as blending, interpolation and Gaussian filtering) is required on the final foveated image. This issue adds more computational and time complexity. In the proposed method we only used the available integer indexing from the rectangular image to the final image and we applied interpolation as a part of the reconstruction process and not post processing, which makes the method less complex.

In filter-based methods, due to usage of multiple low- or band-pass filters, the computational complexity is high. Also, merging and aligning the results of these filters is not easy. In some techniques when the filter-based foveation is applied to the input video of the video coding standard, the overall quality of the resulting videos are lower and it is costly due to applying filtering pre-processing on the input and inner transformation separately. In addition, by providing the final foveated input for the video coders, the whole frame is encoded uniformly regardless of the priority of the regions (ROI or peripheral), which adds more time and resource complexity, even though by having the foveated frames from the beginning no changes need to be applied to the decoder. In our method, we shrink the frames by removing the pixels from the peripheral parts. Therefore, both bit rate and time will be saved in our method which results in having better quality in the resulting videos. Also, the proposed method is independent from the codec, so only an extra extension should be added to both encoder and decoder and it does

(a) Foveated 904×508 frame

(b) Reconstructed 1280×720 frame

Figure 3.8: The CVR method results with four different α values. $\alpha_{x_l} = 0.02$, $\alpha_{x_r} = 0.16$, $\alpha_{y_t} = 0.64$, and $\alpha_{y_b} = 0.04$ (*Vidyo4* sequence, frame# 60)

(a) Foveated 608×340 frame

(b) Reconstructed 1920×1080 frame

Figure 3.9: The CVR method results with four different α values. $\alpha_{x_l} = 0.16$, $\alpha_{x_r} = 0.16$, $\alpha_{y_t} = 0.32$, and $\alpha_{y_b} = 0.04$ (*Tennis* sequence, frame# 210)

not affect the coding and decoding of other video types.

Multiresolution methods are faster and they have easier indexing, but they require merging different intermediate images that induce some artifacts in the boundaries of these images, thus requiring post processing such as blending and interpolation. Furthermore, these methods need more memory space and computation, and some designs are coded-dependent which makes it hard to get aligned with the state-of-the-art codecs. The proposed method is codec-independent.

Wavelet-based methods used for some previous video compression standards used wavelet transform in their design. Even though wavelet-based methods are good for scalable coding, they produce lower quality (blurred) videos. In the new video compression standard, HEVC, wavelet transform is not employed anymore due to its weakness in maintaining the video quality compared to DCT transform. Therefore, the foveation methods used for wavelet-based techniques are not useful anymore in HEVC.

Chapter 4

Experiments

In foveation methods, one of the challenges is finding the fair and best way of evaluating the results. This is because the available objective tests such as PSNR or MSE are common for uniform resolution image and video compression. However, when the compression standard uses perceptual properties, it needs to be evaluated by metrics that consider HVS properties (for more details see Section 1.1) as if a human is looking at the image or the video. Hence, the conventional objective test cannot demonstrate the difference between conventional and perceptual compression (Lee and Ebrahimi, 2012). There are some methods that have been considered for perceptual coding evaluation which have been studied in Lin and Jay Kuo (2011) and Lee and Ebrahimi (2012). In our work, we used the objective and subjective evaluations. In fact, the results of the subjective tests are more reliable and accurate than the objective metrics results due to having humans as observers. The objective test is actually an automatic way of measuring the video quality because checking all the results with human observers is extremely time-consuming.

In this chapter we show our result for the CVR method (See Section 3.3). In our experiment, we used HEVC HM Version 11.0 3 Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP and 11 (2012) with the corresponding low delay B-frame main configuration parameters defined in JCTVC-I1100 Bossen (2012).

4.1 Objective Tests

We designed our objective test scheme as a foveation technique having a fixed bit rate for each video. In the following sections, this experiment is demonstrated.

In this work, we consider four different objective evaluation metrics that are explained in this section. Each metric has been calculated for three different components in each frame, Y, U, and V separately. After all of the frames in a video are considered for each metric, their average value is computed. Then the average of the mean of the three components are computed as the final result. In our experiment we have 180 frames per video.

In some evaluation metrics the *cut-off frequency* (f_c), which has been mentioned in Wang and Bovik (2001), is needed. This frequency is obtained from the contrast sensitivity in 2.2 which is a function of retinal eccentricity. In Geisler and Perry (1998) the best fitting for the constant parameters are $\alpha = 0.106$, $e_2 = 2.3$, and

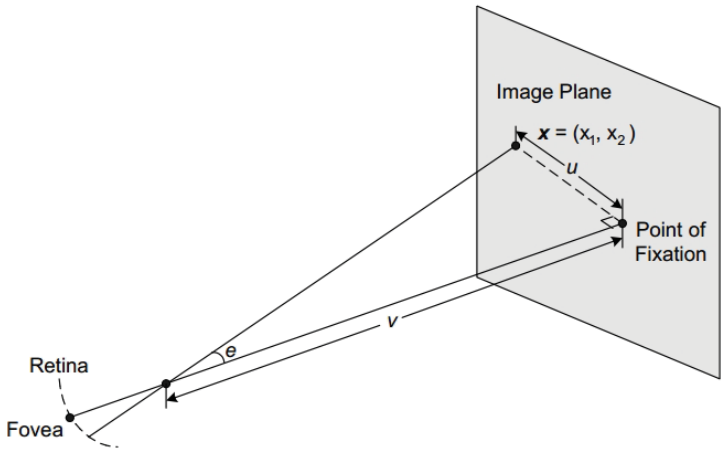


Figure 4.1: A viewing geometry. The pramether e is eccentricy (degrees), and v is the distance to the image (computed according to image width) (Source: Wang and Bovik (2005))

$CT_0 = \frac{1}{64}$. In order to find the cut-off frequency, which is a critical frequency, Geisler and Perry (1998) suggested setting CT to its maximum possible contrast in 2.2, which is 1.0. Then the cut-off frequency will be

$$f_c(e) = \frac{e_2 \cdot \ln\left(\frac{1}{CT_0}\right)}{\alpha \cdot (e + e_2)} \quad (\frac{cycle}{degree}), \tag{4.1}$$

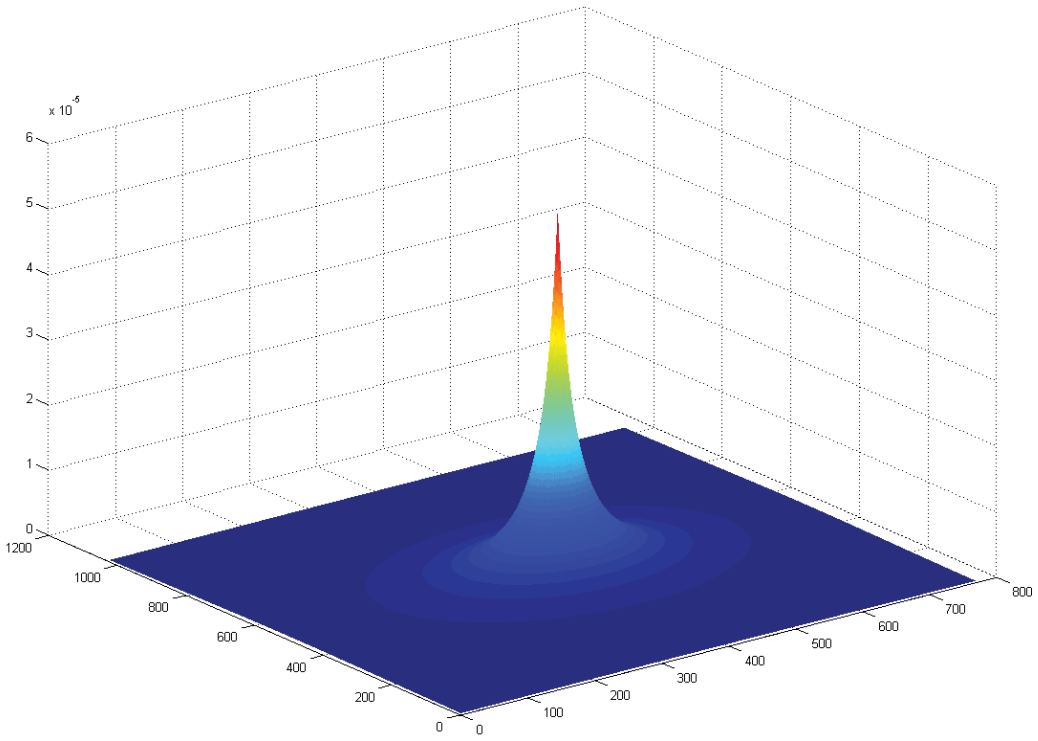
and the eccentricity in 4.1 is

$$e = \frac{180}{\pi} \cdot \frac{s_w}{w \cdot d} \cdot \arctan(\|(x - x_f,\ y - y_f)\|), \tag{4.2}$$

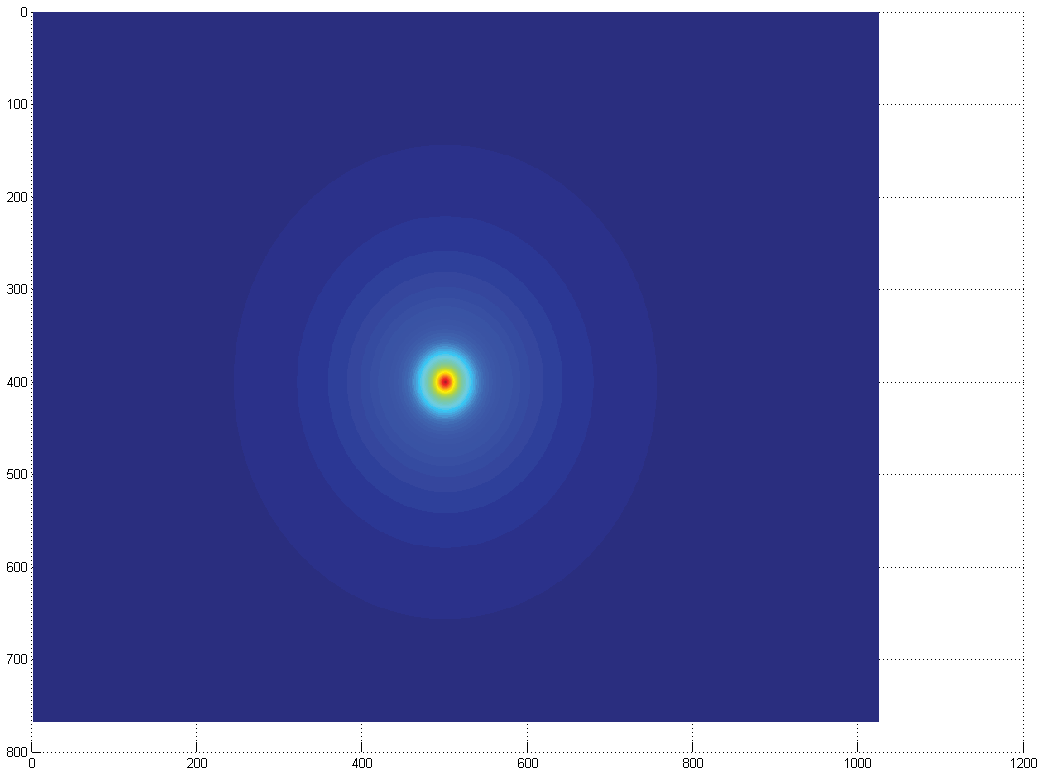
where x and y are the coordinates of each pixel in the image with original size. $\|\cdot\|$ is the euclidean norm. The parameter s_w is the screen width, and d is the distance between the viewer's eyes and the display screen (Wang and Bovik, 2001). For all of the metrics in which they have used f_c, we set $d = 1$ meter and $s_w = 1$ meter. Figure 4.1 demonstrates a viewing geometry.

This cut-off frequency is calculated to simulate the way the human eye can see the image while gazing at one point in the image (fixation point). Figure 4.2 is a sample output for f_c from the side and top view respectively. As you can see, this region is circular and as the eccentricity (distance from fixation point) is increasing, the value of the cut-off frequency is decreasing, i.e. the visual resolution in the human eye is decreasing as well.

1. **ROI PSNR:** The first metric is based on finding the PSNR of the ROI. As in the CVR method, we foveate according to the Cartesian coordinate; therefore, our foveation region is rectangular which can be easily segmented from the rest of the frame. This region is detected by checking the region in the left, right, up and down parts of the image with respect to the fixation point and pick the first missing pixels in x or y direction.



(a) Side view



(b) Top view

Figure 4.2: Cut-off frequency in a 1024×720 image.

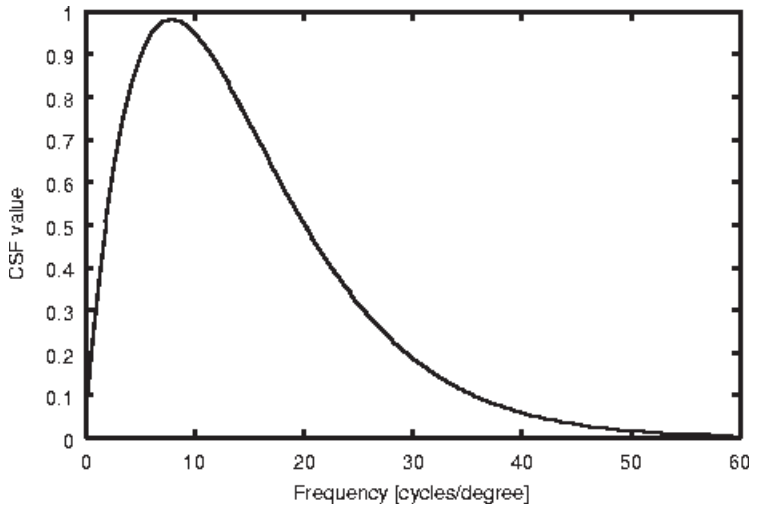


Figure 4.3: Contrast Sensitivity Function (CSF) (Source: Matkovic (1997))

2. **FWSNR (Foveated Weighted Signal-to-Noise Ratio):** This metric has been mentioned in Lee and Bovik (2000), defined as:

$$FWSNR = 10 \log_{10} \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} [p(x,y) * c(x,y)]^2 \cdot f_c^2}{\sum_{x=1}^{w} \sum_{y=1}^{h} [(p(x,y) * p'(x,y)) * c(x,y)]^2 \cdot f_c^2}, \tag{4.3}$$

where the $*$ is a linear convolution and $c(x,y)$ is the contrast sensitivity function (CSF) in the spatial domain:

$$c(x,y) = 2.6 \cdot (0.0192 + 0.114 * f) \cdot e^{-(0.114 \cdot f)^{1.1}}, \tag{4.4}$$

where f is the spatial frequency of the visual stimuli(cycles/degree). This function has a peak at $f = 0.8$ and we do not consider the result after 60 cycles/degree. Figure 4.3 demonstrates CSF behavior (Matkovic, 1997).

3. **FPSNR (Foveated Peak Signal-to-Noise Ratio):** FPSNR is defined in Lee et al. (2001). For this method, first we need to define the foveated mean square error (FMSE):

$$FMSE = \frac{1}{\sum_{x=1}^{w} \sum_{y=1}^{h} [f_c(x,y)]^2} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} [p(x,y) - p'(x,y)]^2 \cdot [f_c(x,y)]^2, \tag{4.5}$$

and, then FPSNR is

$$FPSNR = 10 \cdot \log_{10} \frac{\max[p(x,y)]^2}{FMSE}, \tag{4.6}$$

where for $\max p(x,y)$ we set it to 255.0, for 8-bit representation in our experiments.

4. **FSSIM (Foveated Structural SIMilarity):** The last evaluation technique is called FSSIM, which is mentioned in Wang and Bovik (2006), and compares two images, the original frame, $o(x, y)$, and reconstructed frame $r(x, y)$. This method consists of three parts that we describe next.

   (a) Luminance: In this part, the mean intensity is calculated between two images as follows:

$$\mu_o = \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} o(x, y)^2, \tag{4.7}$$

$$\mu_r = \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} r(x, y)^2, \tag{4.8}$$

   where $0 \leq \mu_o, \mu_r \leq 1$.
   The luminance function is specified as:

$$l(o(x, y), r(x, y)) = \frac{2 \cdot \mu_o \cdot \mu_r + C_1}{\mu_o^2 + \mu_r^2 + C_1}, \tag{4.9}$$

$$C_i = (K_i \cdot L)^2, \quad i \in 1, 2, 3 \tag{4.10}$$

   where $K_i << 1$ and L is the range of pixel value (255 for 8-bit representation). The luminance function finds the similarity of the luminance between two images.

   (b) Contrast: In order to find contrast similarities, we first need to calculate the standard deviation:

$$\sigma_o = \sqrt{\frac{1}{w \cdot h - 1} \sum_{x=1}^{w} \sum_{y=1}^{h} (o(x, y) - \mu_o)^2}, \tag{4.11}$$

$$\sigma_r = \sqrt{\frac{1}{w \cdot h - 1} \sum_{x=1}^{w} \sum_{y=1}^{h} (r(x, y) - \mu_r)^2}, \tag{4.12}$$

   and the contrast function is:

$$c(o(x, y), r(x, y)) = \frac{2 \cdot \sigma_o \cdot \sigma_r + C_2}{\sigma_o^2 + \sigma_r^2 + C_2}, \tag{4.13}$$

   where $0 \leq c(o(x, y), r(x, y)) \leq 1$.

   (c) Structure: This comparison function is calculated as:

$$s(o(x, y), r(x, y)) = \frac{\sigma_{or} + C_3}{\sigma_o \cdot \sigma_r + C_3}, \tag{4.14}$$

   By putting all of these components together, the SSIM index is defined as:

$$SSIM(o(x, y), r(x, y)) = |l(o(x, y), r(x, y))|^{\alpha} \cdot |c(o(x, y), r(x, y))|^{\beta} \cdot |s(o(x, y), r(x, y))|^{\gamma}, \tag{4.15}$$

Table 4.1: Information of Video Sequences

Test Sequence	Frame Rate	Resolution	Content	Frames	Fovea
Cactus	50	1920 × 1080	Indoor moving objects	1-180	Fixed
Kimono	24	1920 × 1080	Moving camera (Face)	1-180	Fixed
Tennis	24	1920 × 1080	Sport (quick actions)	61-240	Movable
Vidyo4	60	1280 × 720	Video conference	1-180	Movable
Johnny	60	1280 × 720	Video conference	1-180	Movable
ChinaSpeed	30	1024 × 768	Car race game	1-180	Fixed
BasketballDrill	50	832 × 480	Sport	2-181	Movable
PartyScene	50	832 × 480	Kids (Zooming)	1-180	Movable
RaceHorses	30	416 × 240	Sport	1-180	Movable

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are for setting the importance of each component. By setting these parameters to 1, (i.e. α, β, and γ), and by setting $C_3 = \frac{C_2}{2}$, the new form of SSIM is:

$$SSIM(o(x,y), r(x,y)) = \frac{(2 \cdot \mu_o \cdot \mu_r + C_1) \cdot (2 \cdot \sigma_{or} + C_2)}{(\mu_o^2 + \mu_r^2 + C_1) \cdot (\sigma_o^2 + \sigma_r^2 + C_2)}. \tag{4.16}$$

By having f_c we can compute FSSIM between the original and reconstructed frame as follows:

$$FSSIM(o,r) = \frac{\sum_{i=1}^{F} f_c(i) \cdot SSIM(o_i(x,y), d_i(x,y))}{\sum_{i=1}^{F} f_c(i)}, \tag{4.17}$$

where F is the number of a frame in a video and $f_c(i)$ are the cut of frequency (f_c) of frame i.

4.1.1 Foveation With Fixed Bit rate

On this experiment, we tested our proposed method with the HEVC coder as an anchor. We set the rate control for both our proposed method and the anchor, with low delay configurations, so that we can set a fixed bit rate for each experiment according to our needs. Fixed bit rate means that there is a limitation for the maximum available bit rate for transmitting through the network. In these experiments we tested nine video sequences, namely *Cactus*, *Kimono*, *Tennis*, *Vidyo4*, *Johnny*, *ChinaSpeed*, *BasketballDrill*, *PartyScene*, and *RaceHorses*. The information of these videos is shown in 4.1. The "Content" in this table explains the content in the video and the "Fovea" column, is for informing whether the fixation points in the whole video are a fixed point in all of the frames (*fixed*) or they are changing from one frame to another (*movable*).

This experiment has been tested for different target bit rates. The target bit rates are 200 Kbps, 300 Kbps, 512 Kbps, 768 Kbps, and 1000 Kbps. The results of ROI

PSNR of sequence *Kimono* for different bit rates are shown in tables 4.5-4.9. The rest of the results can be found in Appendix A.1, Tables A.1-A.8.

Each table refers to a special bit rate. In each of these tables, the result of HEVC and the proposed method, with a different pair of (α, c), are compared. The range of values for α are 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, and 0.90. The different values that are assigned to c are 0.25, 0.50, 0.75, and 0.90.

By looking at these tables of the ROI PSNR results of each sequence, we can see that four different parameters effect the results:

- c: Usually, by increasing c while having fixed value for α, the ROI PSNR increases in the proposed method. It is possible the increment of PSNR is affected by the size of ROI as well, i.e. by having a larger value for c, the ROI size is decreasing and it comes closer to the fixation point which usually has a high PSNR as you can see it out by looking at most of the results (see Tables 4.5-4.9, A.1-A.33, A.3-A.35, A.4-A.36, A.5-A.37, and their corresponding Figures). In some sequences such as *ChinaSpeed*, *BasketballDrill*, and *PartyScene* you can see different results at the lower α values for the higher c; PSNR is lower. One might suggest this is happening because the resolutions of these videos are not as big as other videos and when the parameter c increases, the size of the foveated frame decreases more. However, the small value of α helps to have the biggest ROI in the lower than in the higher values of c. Therefore, more bits will be assigned to the ROI and the video achieves a higher PSNR. As you can see this fact is more obvious in *BasketballDrill* and *PartyScene* than in *Vidyo4* due to their smaller resolution. Finally, we have one sequence, *Tennis*, which does not follow the above description in all of the values of c (i.e. $c = 0.75, 0.90$); however, the subjective tests of the *Tennis* sequence show completely different results.

- α: The parameter α has various effects on the videos based on the fixation points selection modes. If the video has fixed fixation points in all of the frames, the ROI PSNR of the proposed method increases by increments of α and a constant value of c (see Tables 4.5-4.9, A.1-A.33, A.5-A.37 and their corresponding Figures). As an example, in the *ChinaSpeed* table you can find some oscillation in the charts for $c = 0.90$. The reason for this is possibly the smaller ROI in sequences with $c = 0.90$ makes the videos very sensitive to very small changes in the frame and it causes oscillation in ROI PSNR of the foveation method. In case the video has movable fixation points, two different situations can happen by increasing α while c is fixed. One situation is when the position of the fixation points in all of the frames are close to each other and they are following the same object in most of or all the video frames. In this case, the ROI PSNR of the foveation method is generally increasing but you can find oscillation in the way the graph trends (see Tables for *Johnny* and *PartyScene* and their corresponding Figures). The other situation is when there are lots of jumps between the fixation points in neighbouring frames and usually they are following different objects that cause less similarity between the frames. This makes it hard for HEVC to find the best match for frame prediction, especially when the available bit rate is low and HEVC cannot assign more bits in order to preserve the quality of newer

scenes in the video, especially if they are foveated. The rest of the similarities (such as special objects that were followed as attentional regions) go to the peripheral part and appear very small in a way that HEVC cannot reach or detect them due to slice or tile structure of the encoding setting (see Section 2.2). Also, the distance of these areas from the previous frames is large due to the exponential effect of the foveation process (see (3.8) and (3.9)).

- Fixed or movable fovea: The effect of the fixation point mode selection is explained in the previous parameter, α.

- Video content: As this method is for single fovea, the videos which have movable background, or their ROIs, are present in most of the sequential video frames, or their content has very quick movement (motion) in the scene. The foveation technique helps to create a video input that has a somewhat fixed ROI. HEVC is able to encode the foveated video input very efficiently due to the fixed ROI in most of the frames. Also, quick movements cause artifacts in HEVC compression while using the proposed method solves this issue. On the other hand, HEVC can produce better results for fixed background. Therefore, as you can see in the results, *Cactus*, *Kimono*, *Tennis*, *ChinaSpeed*, *PartyScene*, and *RaceHorses* have better results. In the next parameter, we explain more about this fact in HEVC.

- Available fixed bit rate: In general, by increasing the amount of available fixed bit rate, the quality of videos increases both in conventional and foveation compression, and it does not affect the ROI PSNR results if the foveation method has better or worse results than conventional compression. In most of the tables you can see that the foveation results are better (higher) than the conventional ones. The amount of increment is different from one video to another. The general reason is that by giving more bit rate to the video compression standard, the encoder can use more bits in the video and compress less. Therefore, the overall quality gets better at higher bit rates. You can find that in the *BasketballDrill* sequence, we have the lowest increment of PSNR compared to other sequences. Because of the fixed background in the video and the high jumps between the two fixation points between the two adjacent frames, HEVC cannot find very good matches in the prediction part for reconstructing the frame. (For more details, see Section 1.2.)

4.2 Subjective Tests

In this section, we will describe the subjective tests we did based on our final results. Subjective tests can give more reliable results especially because this work is based on the HVS; We have done two different subjective tests: the first for finding the acceptable α and c for each video that has the highest bit rate ($1000Kbs$) based on the subjects' opinions about the quality of the videos. Next, with respect to the acceptable range for α and c, (i.e. from the fair quality up to the highest achievable quality for the videos), we selected our videos for the next subjective test. In our second subjective test, our goal was comparing the video results of conventional compression with our proposed fovea method. We modified an open source

Table 4.2: Scores for subjective test

5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

toolkit, *YUVToolkit* (Zhao, 2011) for our subjective tests. We have used a 1920×1080 TV to show the videos to users. In both tests, the subject user was asked to look at the special region in the video and follow it in order to compensate the absence of eye-tracking device or automatic attentional area selections (See Section 2.3). Figure 4.4 shows the guideline page we have provided for our subjects. In the first subjective test we had twelve subjects, six females and six males. In the second one, we had twelve females and six males. The range of the users' age was from nineteen to forty two.

4.2.1 Selecting The Threshold of α and c

In the first subjective test, we provided different pairs of videos: a result of the foveated method without any video compression (just with the defoveation compression), and the original (reference) video with no compression. In each pair of videos, we shuffled the reference and foveated video and we showed them to the user. The user could decide how many times they wanted to replay the two sequences so as to decide how to score each video in the displayed pair. The scores were in the range of $1 - 5$ (see Table 4.2) labeled by the words in the Table 4.2 instead of numbers. After they scored the first pair, they could select to see the next pair. This test took about $30 - 50$ minutes with 108 different pairs of videos. The user saw all the videos $2m$ away from the TV, even though each video was shown in its original resolution.

We grouped the foveated videos based on their c parameter. In each group, three pairs were shown to the user for comparison different α values (the number of the α values is seven). The first video in each pair was set to the original video, and the next video was selected by a binary search algorithm from the person's scores as follows: we set our α list as

$$l = [0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 0.90] \tag{4.18}$$

where the first element in the list has the smallest effect in the frame, thus it is the most similar video to the original video and one can expect to achieve the lowest difference between the opinion scores (in pair comparison), which is the most satisfying video compared to the rest of videos were achieved by other parameters. On the other hand, $\alpha = 0.90$ has the most effect in the frame so one can expect the highest difference between opinion scores, which is the most dissatisfied video. We assume that the difference between opinion scores is monotonically increasing (but non-linear) in α. In the first pair, the user compared the original video to foveated video with the middle value in list l, which is 0.16. The program calculated the

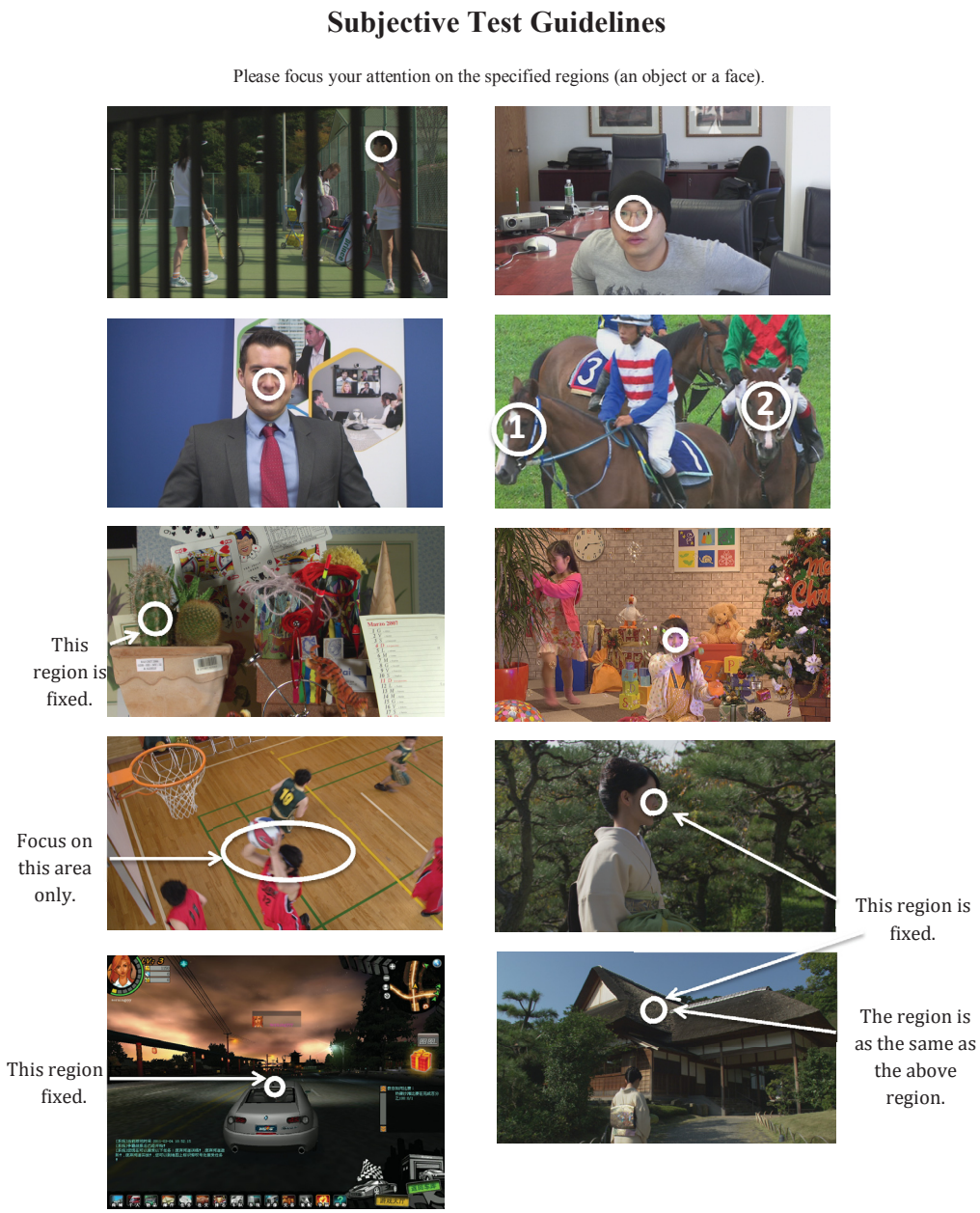


Figure 4.4: Subjective Tests' guideline of ROIs

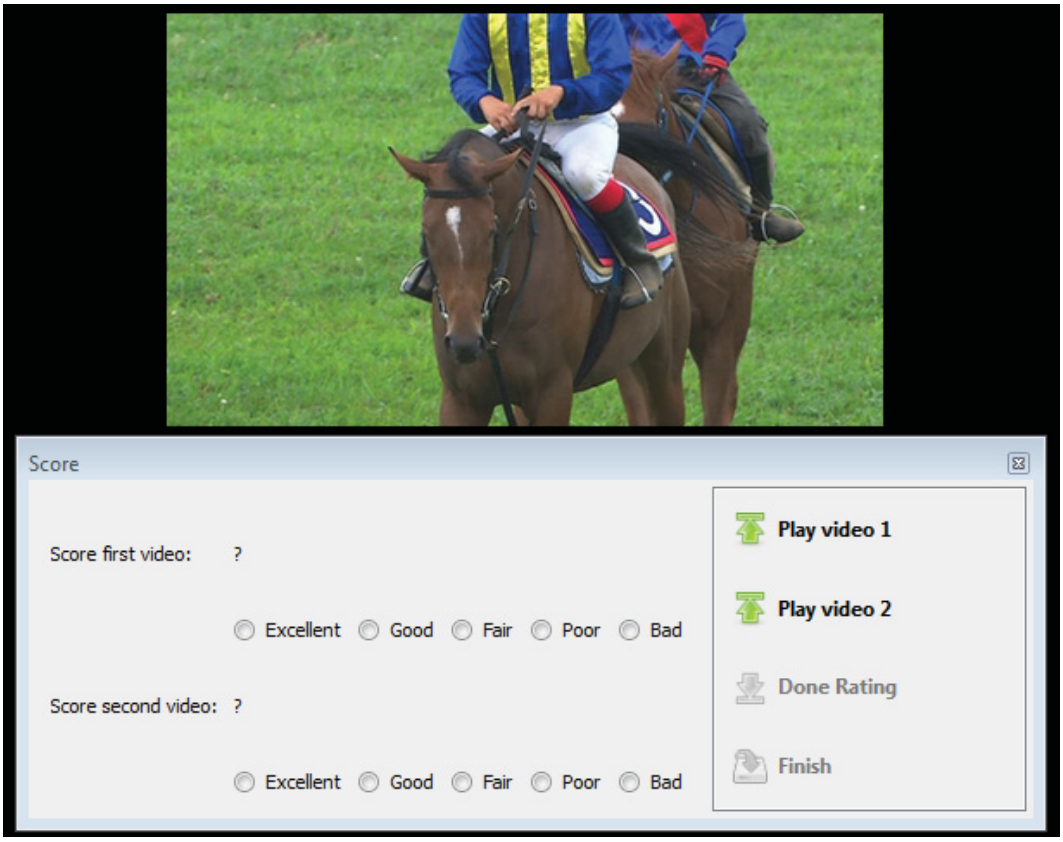


Figure 4.5: First test interface

difference between the given scores for the current pair, and if the difference was larger than a threshold, it would show the next middle parameter in the dissatisfied side of l (the larger α values). Otherwise it would show the middle parameter in the more satisfying side of l (the smaller $alpha$ values). This selection process was repeated until the binary search was concluded. We set our threshold to 2, but one might suggest that a test with lower threshold should be carried out to see if it affects the final results of the test. Figure 4.5 shows the interface of the first test.

Due to the binary search procedure, the data we obtained were not complete, i.e. scores were not available for all pairs (c, α) evaluated. We initialized our scores to zero for all of the α values, and set the scores for the appropriate α from the user scores. We then interpolated the scores of each two neighboring α values in order to fill missing scores. The scores were set to 5 for all the values of α larger than the largest α for which user scores were available. From this data, we generated the Differential Mean Opinion Score (DMOS) for each pair (c, α). Then, we obtained in how many videos each pair of (c, α) had a DMOS of at most 0.5, a defined threshold. The results showed that this was the case for $(0.25, 0.02)$, $(0.25, 0.04)$, and $(0.25, 0.08)$ in all of nine videos, and for $(0.5, 0.02)$, $(0.5, 0.04)$, $(0.5, 0.08)$, and $(0.75, 0.02)$ in eight videos. The other pairs attained DMOS within the threshold in at most 6 videos. Therefore, we selected $(0.25, 0.02)$ from the group with the highest number of occurrences and $(0.75, 0.02)$ from the group with the second highest

number of occurrences. We did not expect to have good result with $(0.75, 0.02)$ due to the high value of c; however, the test surprisingly revealed another result, thus we started to use it as one candidate for the next test. In order to reduce the time of second subjective test, only two pairs were selected, as described ahead.

4.2.2 Comparison Between The Proposed and Conventional Compression Methods

From all the 1305 foveated videos we had, we selected the ones with parameters $(0.25, 0.02)$ and $(0.75, 0.02)$. These were the pairs (c, α) deemed most acceptable, from the first subjective test, and they were chosen so that we could have about $30 - 45$ minutes worth of videos for the second subjective test. We prepared these videos to be compared to their corresponding compressed original videos. The selected pairs had the same available bandwidth and the same content. We showed each pair of videos, in an arbitrary order (i.e. the first video can be conventional or fovea and the second one is the other video). Each pair could be replayed several times by the users so that he/she could compare each pair fairly. They could start to vote from the beginning of the test for each pair. In contrast to the first subjective test, we only showed numbers in the range of $1 - 5$ for scores in the test application interface instead of having labels of Table 4.2. The reason is that after performing the first test, we realized that users usually would not commonly choose the first two labels (Excellent and Good) due to their semantic meanings. Also, the videos were compressed with low bit rates, so their quality was not high. In addition, in order to mitigate the effect of different video resolutions, we fixed the ratio of video's width to the user's distance to the TV by asking the user to make his/her distance to the TV proportional to the resolution of the video being shown. We excluded *BasketballDrill* due to its low quality compared to the original and in order to reduce the test time. Figure 4.6 shows the interface of the second test.

Tables 4.3 and 4.4 demonstrate the results of the second subjective tests. We can see that in *Kimono* and *ChinaSpeed* the proposed method have the highest scores in each comparison. The difference between comparison scores in *Kimono* is large. Then, in the *PartyScene*, *Tennis* and *Cactus* the scores of the proposed method are higher than the conventional's, except in some instances with large bit rate and $c = 0.075$. Therefore, our method preserves video quality of big-resolution videos (1920×1080) especially in lower bit rates. This is because the ROI covers the big part of the foveated video input while the peripheral areas are shrunk, so that HEVC assigns more bits to the ROI, and this improves video quality in the ROI. Also, the method is working well when the sequence has moving scenes (such as moving background), moving camera or fast motions such as sports. Because the foveation follows the attentional area and in the video input the attentional area remains in the center of the frame, HEVC can code it efficiently. Also, HEVC is the state-of-the-art video compression standard and it produces better quality compared to the previous coding standards. Hence, our methods produce better results than the previous coding standards, as well.

In *RaceHorses* the conventional compression has better results. Also, in the sequences *Vidyo4* and *Johnny*, which are the video-conferencing sequences, conventional video compression scored higher than our proposed method, although the difference between the scores was not very high. Thus, we may expect our method

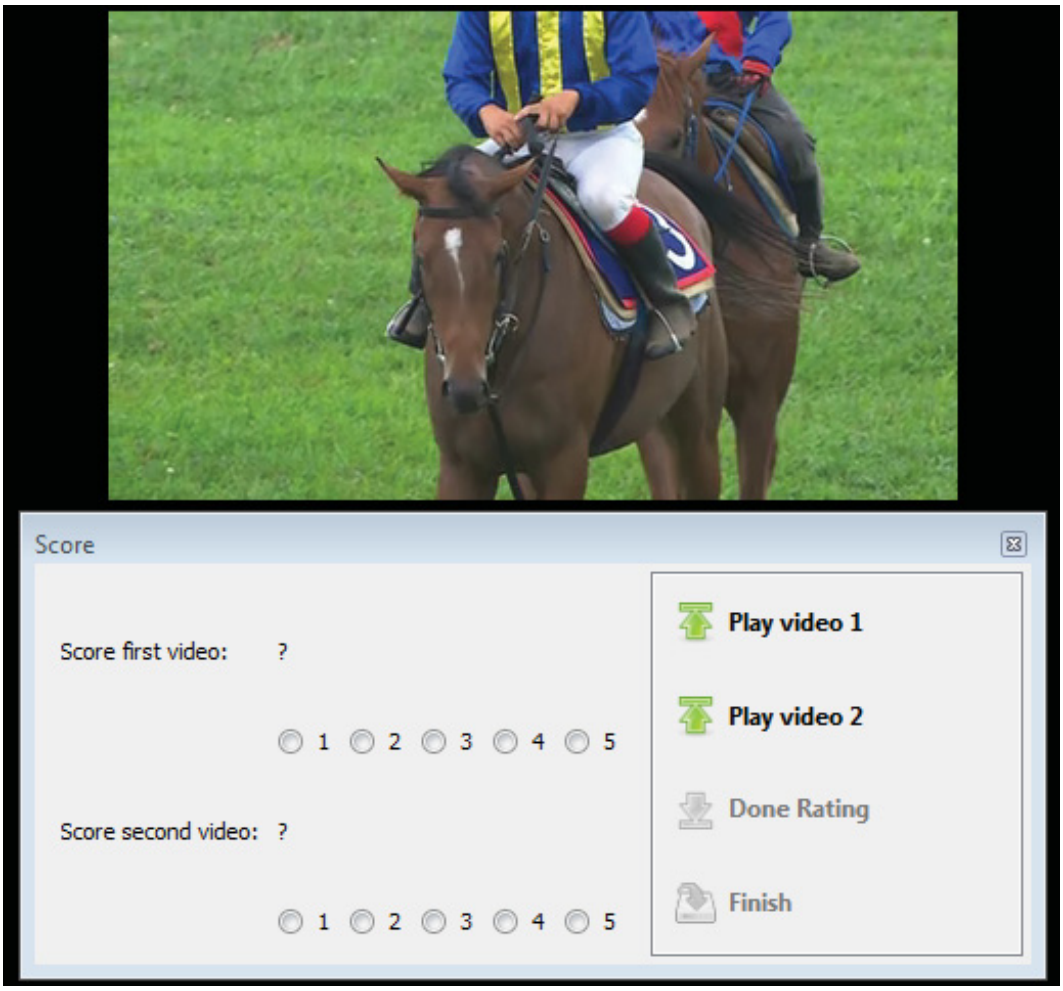


Figure 4.6: Second test interface

to act poorly for video-conferencing and content with fixed camera or background. This evidence refutes the conjecture presented by Basu and Wiebe (1998); Wang et al. (2003), who expected the foveation technique to be particularly good for video conferencing.

Conventional video compression scored higher for these fixed-background videos because HEVC can preserve better quality for fixed background by sending it in the first I-frame in a Group of Pictures, an arranged order of I, P or B frames in a group of successive frames. However, even for this type of content, it is possible that our method may score as well as or better than other previous coding standards due to the low differences in opinion scores.

In the *RaceHourse* and *PartyScene* sequences with $c = 0.25$ have higher scores than $c = 0.75$. This effect is not always true for the rest of the tested sequences. In general, foveated videos with $c = 0.75$ have higher score in lower bit rates. The reason is that higher c leads to a smaller video input to HEVC, which can then assign more bits to the ROI, when the bit rate available is fixed.

We have observed that with a low, fixed bit rate, the video quality was overall poor, but with an increased bit rate the ROI had good quality. However, artifacts in the peripheral area, such as flickering and moving or excessive blur, often disturbed the users.

Video	Bit rate	$c = 0.25$		$c = 0.75$	
		Conv.	Fov.	Conv.	Fov.
RaceHorses	$200K$	3.611	2.278	3.222	1.778
	$300K$	3.722	2.556	3.722	2.0
	$512K$	4.056	3.056	4.278	2.278
	$768K$	4.278	3.333	4.278	2.111
	$1000K$	4.278	3.444	4.222	2.278
PartyScene	$200K$	1.222	2.389	1.333	2.444
	$300K$	1.444	2.944	1.333	2.278
	$512K$	1.944	3.389	2.056	3.056
	$768K$	2.889	3.667	3.222	3.111
	$1000K$	3.5	3.889	3.889	3.167
Tennis	$200K$	1.056	1.556	1.111	2.278
	$300K$	1.778	1.944	1.556	2.5
	$512K$	2.667	2.833	2.333	3.0
	$768K$	3.056	3.444	3.222	3.111
	$1000K$	3.444	3.667	3.5	3.722
Kimono	$200K$	1.111	1.778	1.111	2.333
	$300K$	1.278	2.167	1.167	2.556
	$512K$	1.944	3.056	1.889	3.167
	$768K$	2.389	3.611	2.5	3.444
	$1000K$	2.722	4.0	2.889	3.778

Table 4.3: Subjective comparison between conventional (Conv.) and the proposed fovea (Fov.) $\alpha = 0.02$ for both foveated videos. (First series.)

Video	Bit rate	$c = 0.25$		$c = 0.75$	
		Conv.	Fov.	Conv.	Fov.
Vidyo4	$200K$	2.444	1.889	2.833	2.389
	$300K$	3.556	2.556	3.0	2.778
	$512K$	3.667	3.444	3.889	2.889
	$768K$	4.056	3.5	3.944	3.056
	$1000K$	4.056	3.556	4.0	3.167
Cactus	$200K$	1.056	1.278	1.0	1.722
	$300K$	1.5	1.722	1.167	2.111
	$512K$	2.056	2.389	2.056	2.611
	$768K$	2.611	2.944	2.889	2.889
	$1000K$	3.278	3.5	3.389	3.333
ChinaSpeed	$200K$	1.278	2.333	1.333	2.444
	$300K$	1.778	3.056	1.889	3.056
	$512K$	2.611	3.722	2.333	3.389
	$768K$	3.0	4.0	3.222	4.056
	$1000K$	3.444	4.0	3.722	4.167
Johnny	$200K$	2.5	2.389	2.944	2.444
	$300K$	3.5	2.944	3.222	2.833
	$512K$	3.833	3.389	4.056	3.0
	$768K$	4.222	3.722	4.222	3.222
	$1000K$	4.278	4.0	4.278	3.556

Table 4.4: Subjective comparison between conventional (Conv.) and the proposed fovea (Fov.) $\alpha = 0.02$ for both foveated videos. (Second series)

Table 4.5: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *Kimono*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	36.0945	37.4388	36.2033	38.0806	36.3636	38.9124	36.9805	39.9097
0.04	36.1872	37.7365	36.3301	38.3936	36.3789	39.0938	37.1751	40.2591
0.08	36.2503	37.9510	36.3876	38.5899	36.4093	39.2866	37.1045	40.3115
0.16	36.2235	37.9772	36.3278	38.6103	36.3501	39.3342	37.1727	40.4036
0.32	36.3269	38.3218	36.3774	38.6813	36.2357	39.1610	37.5150	40.6230
0.64	36.3316	38.3045	36.4162	38.9105	36.5860	39.6459	37.8940	41.0091
0.90	36.4017	38.4111	36.3820	38.8733	36.6158	39.7410	38.0534	41.1588

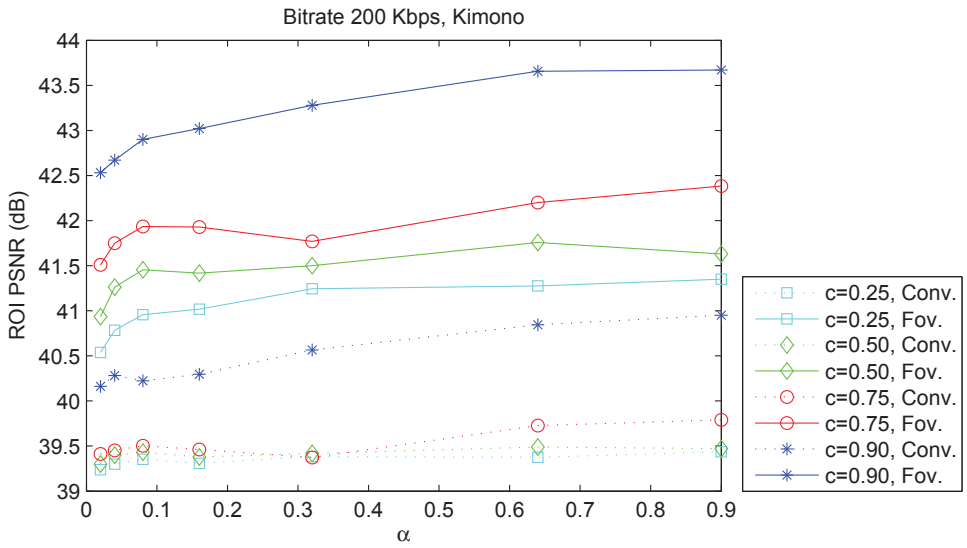


Figure 4.7: Chart of Table 4.5

Table 4.6: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *Kimono*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	36.8457	38.3052	36.9059	38.7452	37.0362	39.5989	37.6885	40.5532
0.04	36.9200	38.5981	37.0289	39.0908	37.0606	39.8080	37.9211	40.7477
0.08	36.9686	38.7439	37.0664	39.3298	37.1178	39.9623	37.8596	40.9220
0.16	36.9286	38.8335	37.0032	39.3564	37.0846	39.9470	37.9486	41.0271
0.32	37.0126	39.0726	37.0567	39.4510	36.9946	39.8352	38.3017	41.1901
0.64	37.0046	39.1113	37.1211	39.6231	37.3655	40.3136	38.6724	41.6047
0.90	37.0752	39.2355	37.0849	39.5617	37.4051	40.4006	38.7979	41.7479

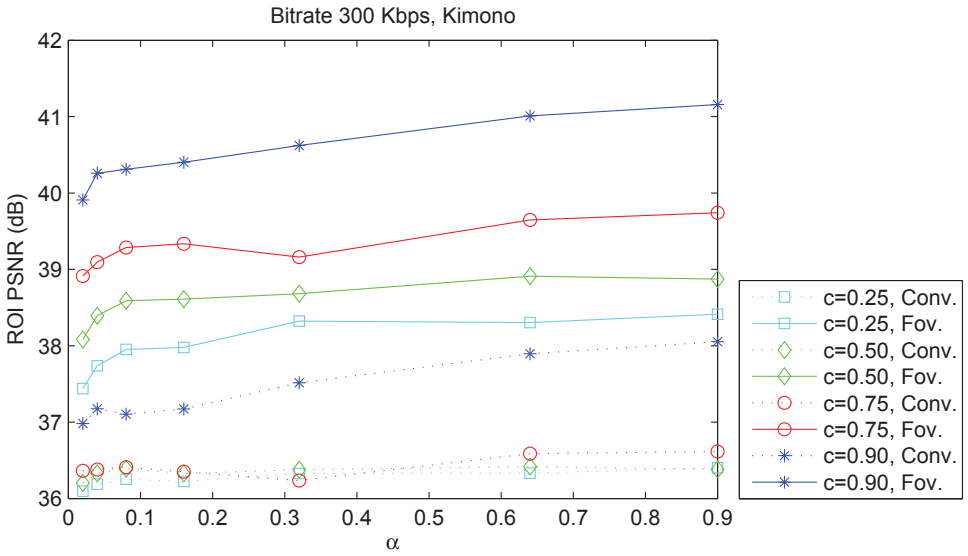


Figure 4.8: Chart of Table 4.6

Table 4.7: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *Kimono*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	37.8396	39.3247	37.8955	39.8222	38.0553	40.3874	38.9001	41.5475
0.04	37.9054	39.6066	38.0111	40.1323	38.0977	40.6155	39.0462	41.7017
0.08	37.9610	39.7631	38.0553	40.3426	38.1529	40.7549	38.9859	41.8966
0.16	37.9149	39.8006	37.9979	40.2857	38.1071	40.7594	39.0560	42.0443
0.32	37.9972	40.0649	38.0629	40.3243	38.0164	40.6545	39.3466	42.2588
0.64	38.0048	40.1080	38.1406	40.5577	38.3922	41.1397	39.6252	42.5973
0.90	38.0742	40.1937	38.1185	40.4845	38.4665	41.2577	39.7492	42.6657

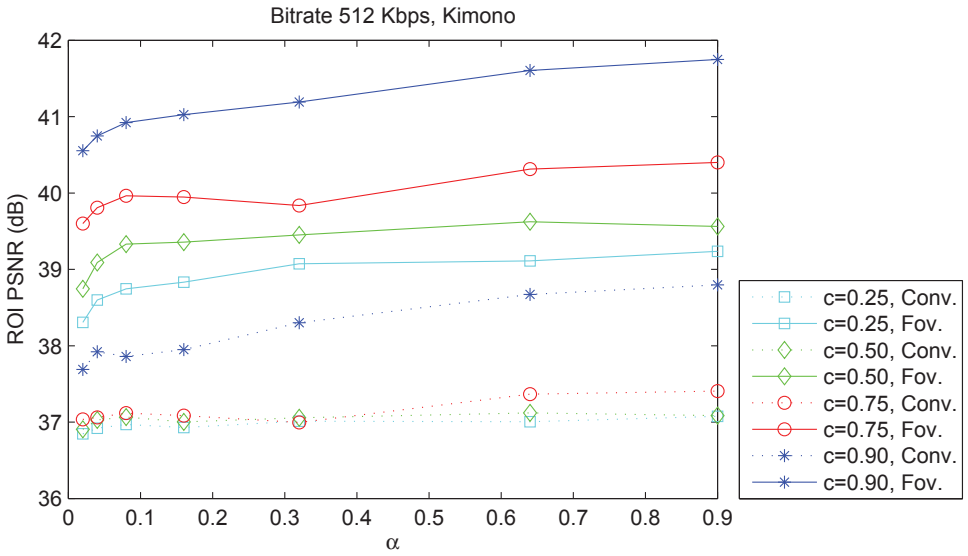


Figure 4.9: Chart of Table 4.7

Table 4.8: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression with bit rate = 768 Kbps

Bit rate 768 Kbps, *Kimono*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	38.6848	40.0884	38.7374	40.4818	38.8186	41.0739	39.5690	42.0485
0.04	38.7466	40.3444	38.8398	40.7914	38.8601	41.2809	39.6794	42.2503
0.08	38.7970	40.5007	38.8748	41.0013	38.9144	41.5059	39.6091	42.4338
0.16	38.7464	40.5580	38.8170	40.9377	38.8709	41.5062	39.6569	42.5194
0.32	38.8225	40.7910	38.8513	41.0093	38.7896	41.3725	39.9255	42.8453
0.64	38.8134	40.8237	38.9183	41.2484	39.1273	41.8130	40.2000	43.2339
0.90	38.8706	40.9198	38.8861	41.1539	39.1747	41.9814	40.3167	43.2423

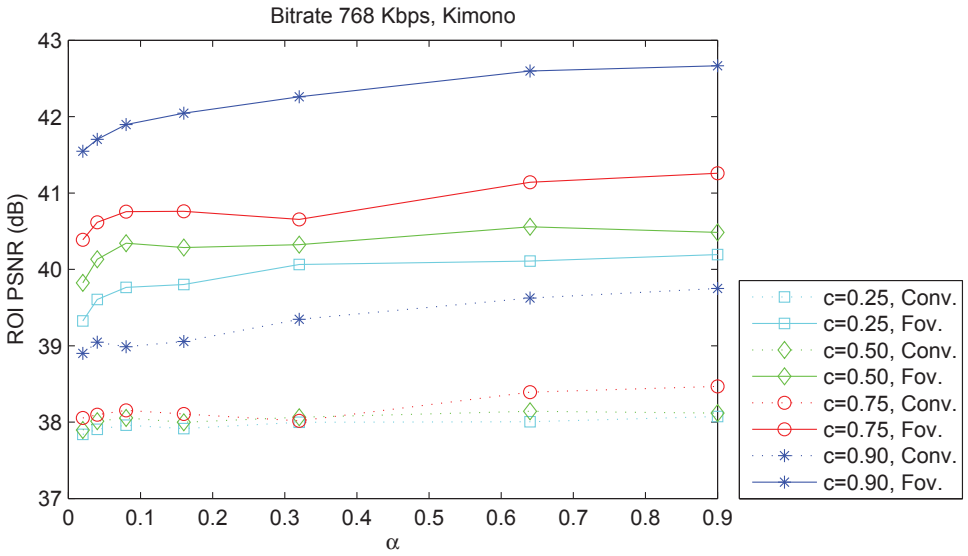


Figure 4.10: Chart of Table 4.8

Table 4.9: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Kimono* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *Kimono*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	39.2374	40.5368	39.3009	40.9350	39.4122	41.5106	40.1623	42.5311
0.04	39.3005	40.7844	39.3993	41.2620	39.4523	41.7500	40.2811	42.6713
0.08	39.3532	40.9563	39.4322	41.4543	39.5019	41.9339	40.2220	42.9006
0.16	39.3087	41.0173	39.3779	41.4180	39.4608	41.9291	40.2951	43.0219
0.32	39.3828	41.2436	39.4185	41.4999	39.3724	41.7687	40.5663	43.2794
0.64	39.3742	41.2765	39.4886	41.7577	39.7273	42.2001	40.8456	43.6570
0.90	39.4352	41.3501	39.4711	41.6296	39.7896	42.3831	40.9502	43.6708

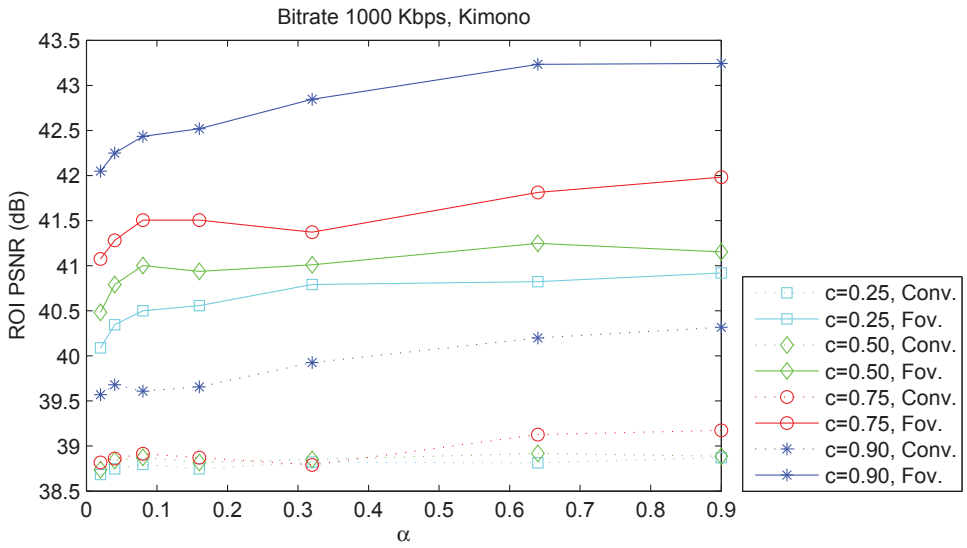


Figure 4.11: Chart of Table 4.9

Table 4.10: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *Kimono*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
	0.50	18.7746	18.4615	18.1590	17.9002	17.6530	17.4953	17.4133	
	0.75	18.6234	18.3196	18.0173	17.7564	17.5361	17.3358	17.2574	
	0.90	18.2134	17.9139	17.6204	17.3190	17.1243	16.9429	16.8185	
FPSNR	0.25	31.9853	31.8102	31.6243	31.4050	31.2651	31.1196	31.0636	
	0.50	32.1345	31.9354	31.7296	31.5013	31.2878	31.1530	31.0826	31.7359
	0.75	32.0590	31.8700	31.6500	31.4207	31.2078	31.0179	30.9670	
	0.90	31.7349	31.5259	31.3058	31.0404	30.8452	30.6522	30.5273	
FSSIM	0.25	0.8968	0.8947	0.8930	0.8913	0.8900	0.8892	0.8887	
	0.50	0.8982	0.8961	0.8942	0.8926	0.8911	0.8901	0.8897	0.9022
	0.75	0.8994	0.8972	0.8952	0.8932	0.8916	0.8904	0.8899	
	0.90	0.8980	0.8958	0.8936	0.8916	0.8900	0.8887	0.8882	

Table 4.11: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *Kimono*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	19.4049	19.0553	18.7311	18.4509	18.2448	18.0876	18.0366	
	0.50	19.3937	19.0689	18.7445	18.4981	18.2269	18.0909	18.0219	20.5574
	0.75	19.2458	18.9259	18.5728	18.2799	17.9892	17.7499	17.6920	
	0.90	18.7293	18.3568	18.0729	17.7482	17.5486	17.3209	17.2396	
FPSNR	0.25	32.7549	32.5547	32.3241	32.1036	31.9254	31.7919	31.7311	
	0.50	32.7200	32.5437	32.3317	32.1227	31.8978	31.7538	31.6758	32.5664
	0.75	32.6573	32.4646	32.2079	31.9607	31.7003	31.4681	31.3978	
	0.90	32.2239	31.9839	31.7480	31.4809	31.2652	31.0415	30.9446	
FSSIM	0.25	0.9038	0.9016	0.8995	0.8975	0.8961	0.8950	0.8946	
	0.50	0.9048	0.9026	0.9004	0.8985	0.8968	0.8955	0.8951	0.9107
	0.75	0.9054	0.9030	0.9006	0.8983	0.8964	0.8949	0.8944	
	0.90	0.9032	0.9006	0.8982	0.8959	0.8941	0.8926	0.8919	

Table 4.12: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *Kimono*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	20.4728	20.0958	19.7314	19.3934	19.1285	18.9628	18.8297	21.8415
	0.50	20.4169	20.0530	19.6699	19.3508	19.0260	18.8538	18.7764	
	0.75	20.0812	19.7097	19.3223	18.9958	18.7320	18.4921	18.4088	
	0.90	19.6276	19.1895	18.8054	18.4433	18.1565	17.8438	17.7534	
FPSNR	0.25	33.6974	33.4891	33.2448	32.9734	32.7721	32.6195	32.5335	33.5957
	0.50	33.6396	33.4451	33.1998	32.9185	32.6518	32.5051	32.4195	
	0.75	33.4257	33.2084	32.9283	32.6530	32.4224	32.2015	32.1043	
	0.90	33.0520	32.7627	32.4639	32.1446	31.8861	31.5913	31.4826	
FSSIM	0.25	0.9137	0.9112	0.9088	0.9064	0.9047	0.9034	0.9027	0.9219
	0.50	0.9140	0.9115	0.9091	0.9067	0.9046	0.9032	0.9026	
	0.75	0.9133	0.9107	0.9080	0.9055	0.9034	0.9016	0.9010	
	0.90	0.9102	0.9071	0.9042	0.9015	0.8991	0.8972	0.8962	

Table 4.13: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *Kimono*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	21.2660	20.8476	20.4518	20.1386	19.8663	19.6210	19.5311	22.9391
	0.50	21.1824	20.8264	20.4072	20.0540	19.7301	19.5069	19.4249	
	0.75	20.9509	20.5203	20.0864	19.7247	19.3898	19.1329	19.0186	
	0.90	20.1839	19.7103	19.2619	18.8576	18.5300	18.1977	18.0586	
FPSNR	0.25	34.3732	34.1470	33.8903	33.6410	33.4260	33.2308	33.1427	34.4221
	0.50	34.3165	34.1212	33.8535	33.5772	33.3174	33.1280	33.0322	
	0.75	34.1378	33.8872	33.6046	33.3026	33.0153	32.7886	32.6793	
	0.90	33.5564	33.2345	32.8859	32.5473	32.2485	31.9498	31.8247	
FSSIM	0.25	0.9210	0.9184	0.9159	0.9134	0.9115	0.9098	0.9092	0.9301
	0.50	0.9208	0.9184	0.9157	0.9131	0.9109	0.9092	0.9085	
	0.75	0.9198	0.9169	0.9140	0.9111	0.9086	0.9066	0.9057	
	0.90	0.9153	0.9118	0.9085	0.9055	0.9028	0.9004	0.8993	

Table 4.14: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Kimono* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *Kimono*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	21.8303	21.3916	20.9834	20.6313	20.3134	20.0975	19.9755	23.6766
	0.50	21.7941	21.3636	20.9619	20.5886	20.2424	19.9657	19.8907	
	0.75	21.4666	21.0084	20.5375	20.1141	19.7275	19.4090	19.2764	
	0.90	20.6087	20.0802	19.5907	19.1280	18.7666	18.3917	18.2476	
FPSNR	0.25	34.8159	34.5798	34.3165	34.0586	33.8285	33.6455	33.5497	34.9722
	0.50	34.7835	34.5613	34.3091	34.0306	33.7617	33.5529	33.4489	
	0.75	34.5619	34.3038	33.9825	33.6436	33.3252	33.0506	32.9375	
	0.90	33.9165	33.5697	33.1904	32.8101	32.4874	32.1530	32.0193	
FSSIM	0.25	0.9254	0.9229	0.9202	0.9177	0.9155	0.9140	0.9131	0.9350
	0.50	0.9254	0.9227	0.9199	0.9173	0.9149	0.9131	0.9123	
	0.75	0.9237	0.9207	0.9175	0.9145	0.9118	0.9096	0.9086	
	0.90	0.9183	0.9146	0.9111	0.9077	0.9048	0.9021	0.9009	

Chapter 5

Conclusion and Future Work

In this thesis, we designed a standard-independent perceptual foveation-based video coding method by improving a foveation-based image processing technique. Also, we have added an extension that allows ROIs with arbitrary rectangular shape. With this one may preserve the quality of various objects' shapes in different video content, while using only one fixation point per object. According to our results and comparisons with HEVC as our conventional video coding standard, we conclude that this method is useful for video contents that have fast motion, the fixation point is mostly following one object in most of the frame, or the background is moving (having movable camera or background). Using fovea in the videos helps to have better quality in the regions which attract more attention to them. This is going to be useful in most video applications such as surveillance videos (especially if it is in a very windy place when camera is moving or most of the objects in the scene are moving), sports, and typical movies. This design is only developed and tested for a single fixation point in the video but it is easily expendable to multiple fixation points, which one might suggest that can improve both perceptual quality and be useful for types of video content with multiple attention areas.

The first step in perceptual foveation-based video compression is detecting the fixation points. In this study, we detected them manually; however, one can use the automatic selection of these points (see Section 2.3) which makes the system completely automatic, one of the essential things in most of video applications. The other part of perceptual compression is the quality assessment, for which we cannot use the conventional PSNR or MSE metrics due to their uniform evaluation on the whole frame. Therefore, we need to have proper evaluation metrics that evaluate the attentional regions (or ROIs) different than the peripheral areas. In this study, we summarized and used four different objective metrics, generally uncommon to find all used in one work. However, objective tests do not always match the perceptual quality truly perceived by humans. As you can see in Tables 4.3 4.4, users vote for better quality of videos in which the objective tests set lower quality values. Thus, subjective test is essential in the perceptual work, but they still have to design better evaluation metrics that can correctly judge among results because it is time consuming for subjective evaluation of all single results and in some cases it is very expensive, too. Another future possibility for evaluating the proposed method is as follows; we have unlimited available bit rate for

each video; and then by getting the used bit rate for the foveated videos, we need to set the used bit rate as the target bit rate (fixed bit rate) for the conventional video compression and compare their quality as well.

The proposed method is independent from the video content. However, it does not produce very good quality for all types of videos. Therefore, as a future trend, one can find out how the foveation method can work completely independent from the content or how one can design a system that can automatically evaluate whether the video content is useful for the foveation method or not. In addition, if an automatic detection of the parameters α, and c can be found, it can maximize the achieved quality.

If real-time performance is a requirement, one may process each row in parallel in order to encode and decode in real-time.

One of the weakness of this method is that in the foveation process, we have repeated pixels in the ROI area. Future work can avoid these redundancies by presenting a new foveated model that can preserve the quality of the ROI. One might suggest that it can improve the compression size of data as well because the proposed method cannot compress the data more than the state-of-the-art video coding standard.

Another lesson we learned from our subjective tests is that users are tolerant to static blur and smoothness in peripheral areas; however, if flicker or moving blur happens, even in the peripheral areas, it is easily noticed by human eyes. Therefore, in future work one should investigate how to decrease these artifacts in the peripheral areas to increase the users' satisfaction with the video quality.

One final future possibility is if the same fix of the shifting effect in CVR re-transform is applied to the MSF re-transform method, one can produce better images with MSF even though the method may not be a very good match for the video coding standards. Moreover, the Basu and Wiebe (1998) foveation method has been used in other applications (Basu and Cheng, 2001) (Sanchez et al., 2004) and (Basu et al., 2002). By applying the proposed method to the previous applications, one can improve the quality of the results achieved from these applications.

Bibliography

3 Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP and 11, I. J. S. W. (2012). Hm 11.0 reference software.

Bandera, C. and Scott, P. (1989). Foveal machine vision systems. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 596–599 vol.2.

Basu, A. and Cheng, I. (2001). Qos based video delivery with foveation. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 986–989 vol.1.

Basu, A., Cheng, I., and Pan, Y. (2002). Foveated online 3d visualization. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 944–947 vol.3.

Basu, A. and Licardie, S. (1993). Modeling fish-eye lenses. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 1822–1828 vol.3.

Basu, A., Sullivan, A., and Wiebe, K. (1993). Variable resolution teleconferencing. In *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, pages 170–175 vol.4.

Basu, A. and Wiebe, K. (1998). Enhancing videoconferencing using spatially varying sensing. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(2):137–148.

Bossen, F. (2012). *Common HM test condtions and software reference configurations*. JCTVC-K1100, 11th Meeting of Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Geneva.

Breitmeyer, B. and Ogmen, H. (2006). *Visual Masking: Time Slices Through Conscious and Unconscious Vision*. Opss Series. OUP Oxford.

Burt, P. and Adelson, E. (1983). The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540.

Camacho, P., Arrebola, F., and Sandoval, F. (1996). Shifted fovea multiresolution geometries. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 307–310 vol.1.

Cerf, M., Harel, J., Einhaeuser, W., and Koch, C. (2007). Predicting human gaze using low-level saliency combined with face detection. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 241–248.

Chang, E.-C. and Yap, C. K. (1997). A wavelet approach to foveating images. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, SCG '97, pages 397–399, New York, NY, USA. ACM.

Chen, Z. and Guillemot, C. (2010). Perceptually-friendly h.264/avc video coding based on foveated just-noticeable-distortion model. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(6):806–819.

Cox, E. (1994). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*. Academic Press Professional, Inc., San Diego, CA, USA.

Cox, I., Miller, M., Bloom, J., Fridrich, J., and Kalker, T. (2008). *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition.

Enns, J. T. and Di Lollo, V. (2000). What's new in visual masking? *Trends in Cognitive Sciences*, 4(9):345–352.

Frintrop, S., Rome, E., and Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. *ACM Trans. Appl. Percept.*, 7(1):6:1–6:39.

Garrett-Glaser, J. (2010). The problems with wavelets.

Geisler, W. S. and Perry, J. S. (1998). A real-time foveated multiresolution system for low-bandwidth video communication. In *in Proc. SPIE*, pages 294–305.

Gonzalez, R. C. and Woods, R. E. (1992). *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.

Hang, H.-M., Peng, W.-H., Chan, C.-H., and Chen, C.-C. (2010). Towards the next video standard: High efficiency video coding. In *Proceedings of the Second APSIPA Annual Summit and Conference*, pages 609–618, Biopolis, Singapore.

ISO/IEC (1993). Iso/iec 11172-1:1993 – information technology – coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s – part 1: Systems. Padrão.

ISO/IEC (1994). Generic coding of Moving pictures and associated audio information - Part 2: Video. *ITU-T Recommendation H.262-ISO/IEC 13818-2, MPEG-2*.

ISO/IEC (1999). Coding of audio-visual objects - Part 2: Visual. *ISO/IEC 14496-2, MPGEG-4 Visual Version 1*.

Itti, L. (2004). Automatic foveation for video compression using a neurobiological model of visual attention. *Image Processing, IEEE Transactions on*, 13(10):1304–1318.

ITU-T (1995). Video coding for low bitrate communication. *ITU-T Draft Recommendation H.263 Version 1*.

JVT (2003). Advanced video coding for generic audiovisual services. *ITU-T Recommendation International Standard of Joint Video Specification, ITU-T Rec. H.264/ISO/IEC 14496-10 AVC, Version 1, JVT-G50*.

Kortum, P. T. and Geisler, W. S. (1996). Implementation of a foveated image-coding system for bandwidth reduction of video images. In Rgowitz, B. and Allebach, J., editors, *Proc. SPIE*, volume 2657, pages 350–360.

Kyuel, T., Geisler, W. S., and Ghosh, J. (1999). Retinally reconstructed images: digital images having a resolution match with the human eye. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 29(2):235–243.

Lee, J.-S. and Ebrahimi, T. (2012). Perceptual video compression: A survey. *J. Sel. Topics Signal Processing*, 6(6):684–697.

Lee, S. and Bovik, A. (2000). Foveated video image analysis and compression gain measurements. In *Image Analysis and Interpretation, 2000. Proceedings. 4th IEEE Southwest Symposium*, pages 63–67.

Lee, S., Bovik, A. C., and Kim, Y. Y. (1999). Low delay foveated visual communications over wireless channels. In *ICIP (3)*, pages 90–94.

Lee, S., Pattichis, M., and Bovik, A. (2001). Foveated video compression with optimal rate control. *Image Processing, IEEE Transactions on*, 10(7):977–992.

Lin, W. and Jay Kuo, C. C. (2011). Perceptual visual quality metrics: A survey. *J. Vis. Comun. Image Represent.*, 22(4):297–312.

Lu, L., Wang, Z., Bovik, A. C., and Kouloheris, J. (2001). Adaptive frame prediction for foveation scalable video coding. In *IEEE Inter. Conference on Multimedia and Expo*.

Marichal, X., Delmot, T., Vleeschouwer, C. D., Warscotte, V., and Macq, B. (1996). Automatic detection of interest areas of an image or of a sequence of images. In *ICIP (3)*, pages 371–374. IEEE.

Matkovic, K. (1997). *Tone Mapping Techniques and Color Image Difference in Global Illumination*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

Mazza, V., Turatto, M., Rossi, M., and Umiltà, C. (2007). How automatic are audiovisual links in exogenous spatial attention? *Neuropsychologia*, 45(3):514 – 522. Advances in Multisensory Processes.

Ohm, J., Sullivan, G., Schwarz, H., Tan, T. K., and Wiegand, T. (2012). Comparison of the coding efficiency of video coding standards x2014;including high efficiency video coding (hevc). *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1669–1684.

Osberger, W., Hammond, S., and Bergmann, N. (1997). An mpeg encoder incorporating perceptually based quantisation. In *TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, volume 2, pages 731–734 vol.2.

Osberger, W. and Maeder, A. (1998). Automatic identification of perceptually important regions in an image. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 701–704 vol.1.

Perry, J. S. and Geisler, W. S. (2002). Gaze-contingent real-time simulation of arbitrary visual fields. In *In Human Vision and Electronic Imaging, SPIE Proceedings*, pages 57–69.

Richardson, I. (2013). Hevc: An introduction to high efficiency video coding [summary version].

Sanchez, V., Basu, A., and Mandal, M. (2004). Prioritized region of interest coding in jpeg2000. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(9):1149–1155.

Schwarz, H., Marpe, D., and Wiegand, T. (2007). Overview of the scalable video coding extension of the h.264/avc standard. In *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY IN CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, pages 1103–1120.

Sheikh, H. R., Evans, B. L., and Bovik, A. C. (2003). Real-time foveation techniques for low bit rate video coding. *Real-Time Imaging*, 9(1):27–40.

Simons, D. J. and Chabris, C. F. (1999). Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception*, 28:1059–1074.

Smith, S. W. (1997). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA.

Stelmach, L. B., Tam, W. J., and Hearty, P. J. (1991). Static and dynamic spatial resolution in image coding: an investigation of eye movements.

Sullivan, G., Ohm, J., Han, W.-J., Wiegand, T., and Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668.

Tellinghuisen, D. and Nowak, E. (2003). The inability to ignore auditory distractors as a function of visual task perceptual load. *Perception and Psychophysics*, 65(5):817–828.

Tsumura, N., Endo, C., Haneishi, H., and Miyake, Y. (1996). Image compression and decompression based on gazing area.

Umesh Rajashekar, Z. W. and Bovik, A. C. (2001). Real-time foveation: An eye tracker-driven imaging system.

Wallace, R., Ong, P.-W., Bederson, B., and Schwartz, E. (1994). Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90.

Wandell, B. A. (1995). *Foundations of Vision*. Sinauer Associates, Inc.

Wang, Z. and Bovik, A. (2001). Embedded foveation image coding. *Image Processing, IEEE Transactions on*, 10(10):1397–1410.

Wang, Z. and Bovik, A. C. (2005). Foveated image and video coding. In Wu, H. R. and Rao, K. R., editors, *Digital Video Image Quality and Perceptual Coding(Signal Processing and Communications)*, chapter 14, pages 431–457. CRC Press, Inc., Boca Raton, FL, USA.

Wang, Z. and Bovik, A. C. (2006). *Modern Image Quality Assessment*. Synthesis Lectures on Image, Video, and Multimedia Processing. Morgan & Claypool Publishers.

Wang, Z., Lu, L., and Bovik, A. (2001). Rate scalable video coding using a foveation-based human visual system model. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 3, pages 1785–1788 vol.3.

Wang, Z., Lu, L., and Bovik, A. C. (2003). Foveation scalable video coding with automatic fixation selection. *IEEE Trans. Image Processing*, 12:243–254.

Yantis, S. (2000). Goal-directed and stimulus-driven determinants of attentional control.

You, J., Ebrahimi, T., and Perkis, A. (2014). Attention driven foveated video quality assessment. *Image Processing, IEEE Transactions on*, 23(1):200–213.

Zhao, D. Y. (2011). Yuvtoolkit.

Appendix A

Additional Experimental Results

A.1 ROI PSNR Results

Table A.1: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Cactus* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *Cactus*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	30.6361	32.1208	30.5868	32.6654	30.2118	33.4031	30.1093	33.5403
0.04	30.5773	32.2234	30.5771	32.8403	30.3964	33.7545	30.2748	34.0701
0.08	30.6707	32.4420	30.5832	33.0796	30.3944	33.9552	30.4025	34.5045
0.16	30.6079	32.6471	30.4966	33.3196	30.3954	34.0659	30.5367	34.7665
0.32	30.5712	32.7232	30.4637	33.3196	30.4757	34.2395	30.7752	35.2653
0.64	30.4479	32.7957	30.4963	33.5741	30.4193	34.4667	30.7364	35.3232
0.90	30.5126	32.9362	30.4273	33.6465	30.5366	34.7096	30.8500	35.3370

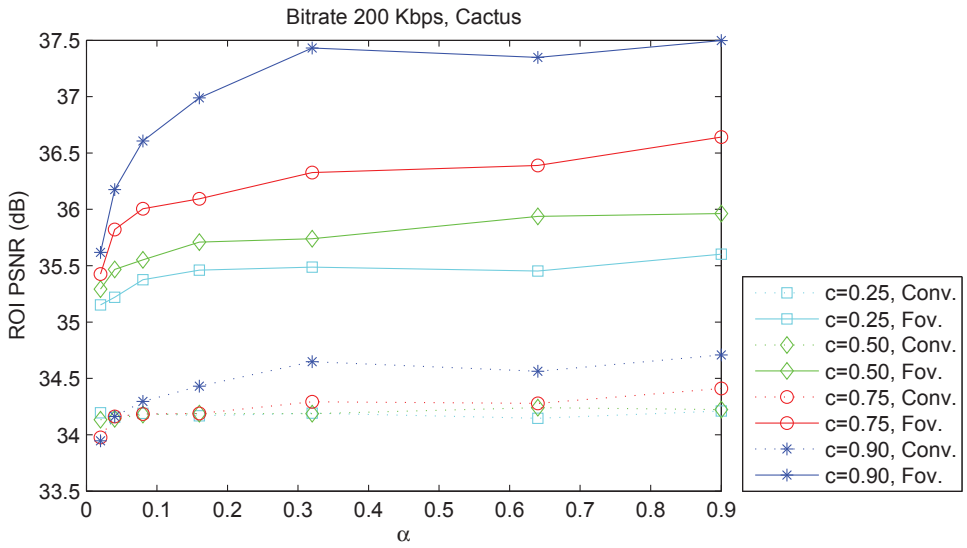


Figure A.1: Chart of Table A.1

Table A.2: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Tennis* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *Tennis*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	24.0795	24.0287	24.3493	24.2234	24.5434	24.2963	25.2899	24.3293
0.04	24.2631	24.2057	24.4186	24.2589	24.6107	24.2946	24.4935	24.0661
0.08	24.3797	24.2861	24.4919	24.2838	24.6503	24.2972	24.4928	24.1137
0.16	24.4374	24.3382	24.5758	24.3627	24.6451	24.2829	24.5899	24.2270
0.32	24.5087	24.3468	24.6519	24.3801	24.6244	24.2141	24.8071	24.3031
0.64	24.5850	24.4011	24.6655	24.3552	24.6277	24.2391	25.0904	24.5322
0.90	24.6204	24.4245	24.6862	24.3761	24.6142	24.2153	25.4387	24.4836

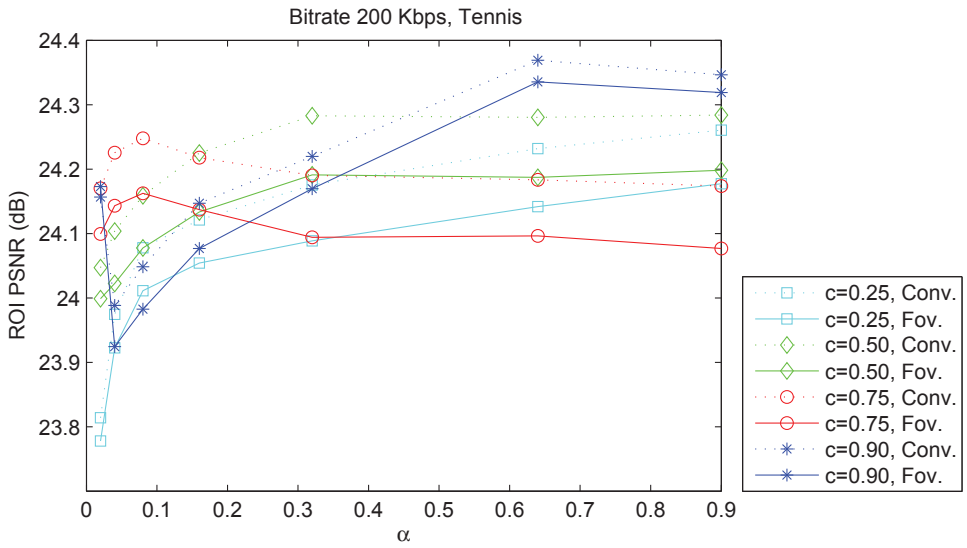


Figure A.2: Chart of Table A.2

Table A.3: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Vidyo4* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *Vidyo4*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	39.0295	38.7590	38.4496	38.6021	38.5517	39.2229	39.3032	40.4728
0.04	38.8170	38.6798	38.3420	38.5988	38.3393	39.2233	38.3241	39.8168
0.08	38.5539	38.5554	38.2159	38.5926	38.3035	39.4770	38.1592	39.8989
0.16	38.3562	38.2057	38.1942	38.5298	38.2222	39.4113	37.9703	39.6780
0.32	38.1900	38.1419	38.2131	38.7014	38.1782	39.4900	37.8210	39.5829
0.64	38.1929	38.2029	38.2105	38.6748	38.1103	39.3971	37.7478	39.4210
0.90	38.2094	38.1138	38.1846	38.7908	38.0799	39.4662	37.9612	39.3835

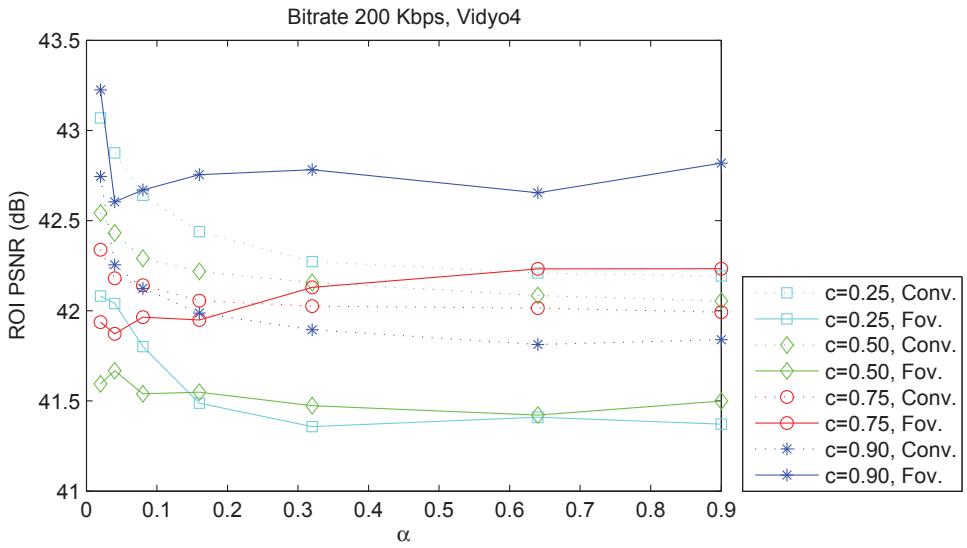


Figure A.3: Chart of Table A.3

Table A.4: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Johnny* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *Johnny*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	39.4802	37.4329	39.0793	37.9662	39.5510	39.5532	40.0893	40.4806
0.04	39.2631	37.5122	39.0276	37.8719	39.5882	39.7737	40.3461	41.1342
0.08	39.1358	37.3761	39.0141	38.1354	39.6526	40.0885	40.3947	41.2014
0.16	38.9757	37.2323	39.1175	38.3361	39.6811	40.0980	40.4490	41.3570
0.32	38.9963	37.2938	39.3318	38.5495	39.7379	39.9690	40.4823	41.4897
0.64	39.1324	37.6207	39.6067	39.2051	40.0232	40.2188	40.5914	41.3364
0.90	39.2684	37.8353	39.6855	39.2699	40.0252	40.1718	40.6802	41.3667

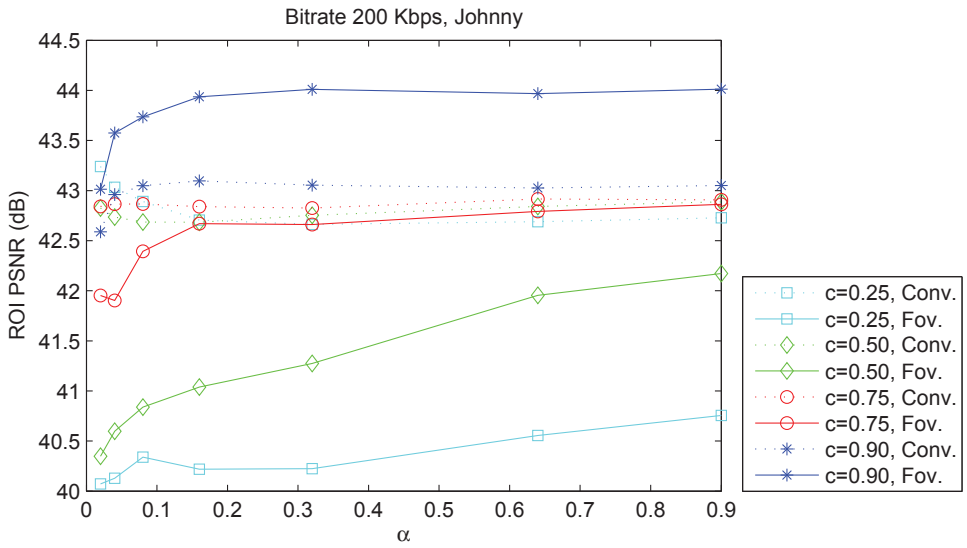


Figure A.4: Chart of Table A.4

Table A.5: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *ChinaSpeed* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *ChinaSpeed*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	32.3387	32.8535	33.7562	34.9894	35.5709	37.2267	38.0730	41.6903
0.04	32.3347	33.3370	33.5695	35.0320	35.2022	37.3332	35.5049	38.7465
0.08	32.7667	33.5910	33.8237	35.0913	35.3337	37.7819	35.7234	39.2940
0.16	33.4740	34.7643	34.8855	36.9403	35.2786	38.6065	35.3542	39.5567
0.32	34.1301	35.7764	34.9988	37.2023	35.6984	39.3232	35.4033	39.1463
0.64	34.8493	36.9278	34.9936	37.5196	35.5124	39.2360	35.4821	40.3929
0.90	34.7373	36.8771	35.2987	38.0887	35.5919	39.3800	35.7614	40.8356

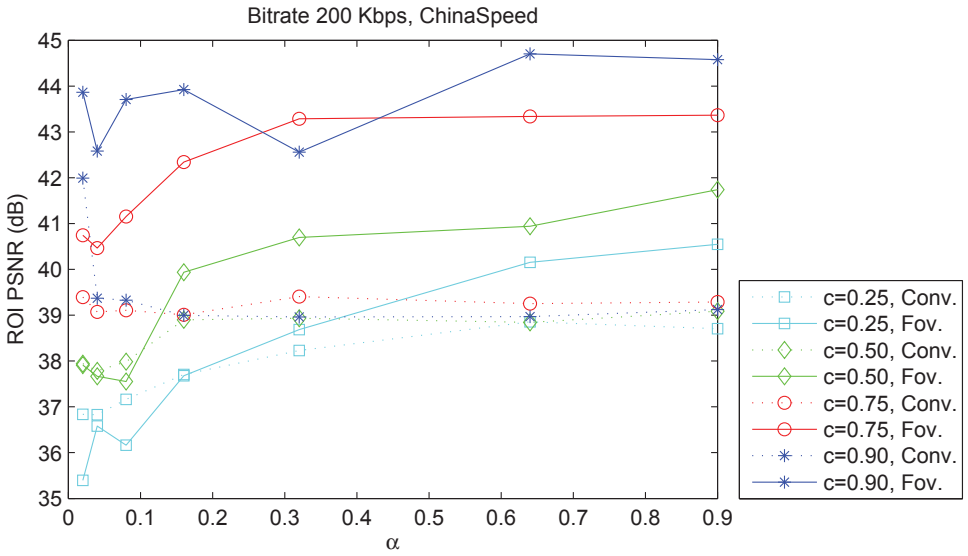


Figure A.5: Chart of Table A.5

Table A.6: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *BasketballDrill* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *BasketballDrill*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	28.0192	27.9087	27.8648	27.8168	27.7901	28.0224	32.2088	32.7002
0.04	27.8869	27.7677	27.6448	27.5377	27.2911	27.3144	30.1664	30.6607
0.08	27.7229	27.5891	27.5005	27.3517	27.1725	27.1604	29.4281	29.7417
0.16	27.5766	27.4249	27.3835	27.2016	27.1837	27.1689	29.4406	29.4865
0.32	27.4998	27.3536	27.2662	27.1332	27.3534	27.3703	30.4705	30.6057
0.64	27.4008	27.2444	27.2262	27.1012	27.5345	27.5221	32.1086	32.6030
0.90	27.3322	27.1804	27.1790	27.0799	27.7538	27.7786	33.6288	33.0218

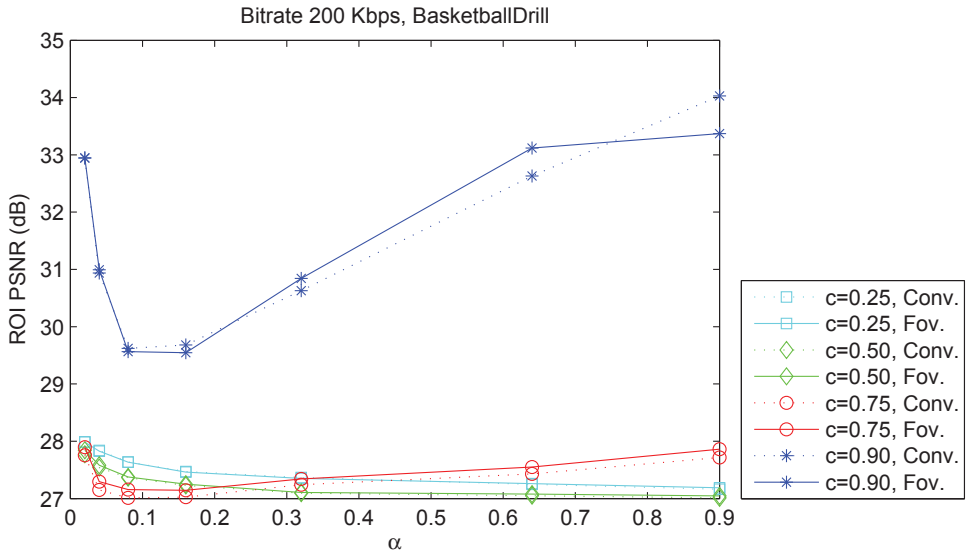


Figure A.6: Chart of Table A.6

Table A.7: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *PartyScene* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *PartyScene*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	29.9127	31.2042	29.8474	31.6367	28.9892	31.2220	26.8859	28.9502
0.04	29.8759	31.4656	29.7830	31.8347	29.2911	31.8456	28.6059	31.1675
0.08	29.7944	31.7532	29.6580	32.1797	29.2177	32.2725	28.5116	31.6171
0.16	29.7450	31.9668	29.5884	32.3307	29.0740	32.3748	28.9002	32.4304
0.32	29.6447	32.0321	29.4962	32.4441	28.9659	32.4860	29.1398	33.0148
0.64	29.5940	32.1134	29.4041	32.5172	28.9795	32.5621	29.1406	33.0053
0.90	29.5694	32.2705	29.3288	32.4574	28.9778	32.5328	29.5107	32.5402

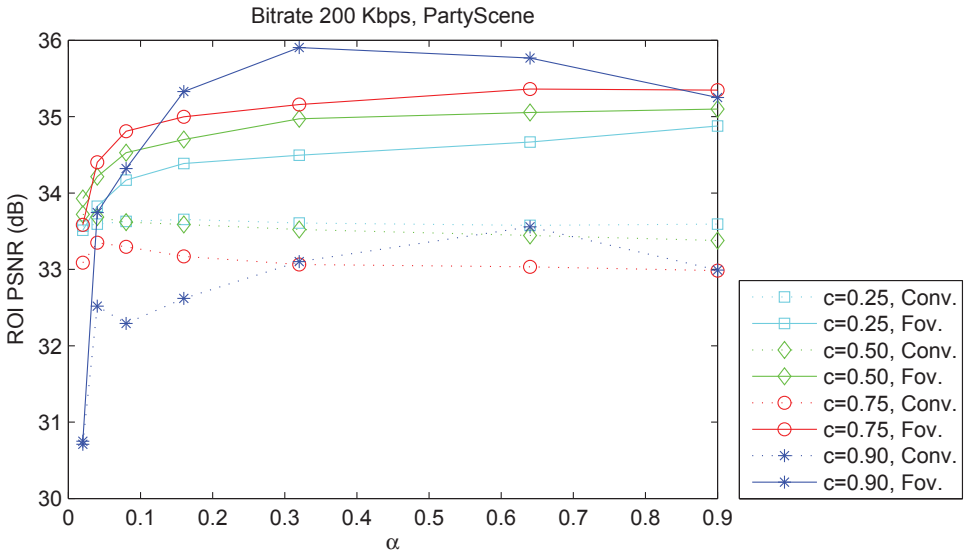


Figure A.7: Chart of Table A.7

Table A.8: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *RaceHorses* with bit rate = 200 Kbps.

Bit rate 200 Kbps, *RaceHorses*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	34.6334	33.9156	35.4099	34.8703	38.1011	38.3571	45.6402	44.1567
0.04	34.6530	34.0816	35.3841	35.0042	36.6773	36.5726	41.6596	43.1171
0.08	34.8647	34.3704	35.5923	35.4403	36.4289	36.7805	39.4726	39.1776
0.16	35.1739	34.8065	35.6427	35.5467	36.3656	36.3897	39.1490	39.1319
0.32	35.3773	35.1425	35.7418	35.9153	36.4522	36.9019	39.8910	40.9656
0.64	35.6279	35.4532	35.9686	36.0123	36.3848	36.9994	42.4999	41.8043
0.90	35.7292	35.6682	36.0833	36.1385	36.8725	37.2829	43.4339	43.7989

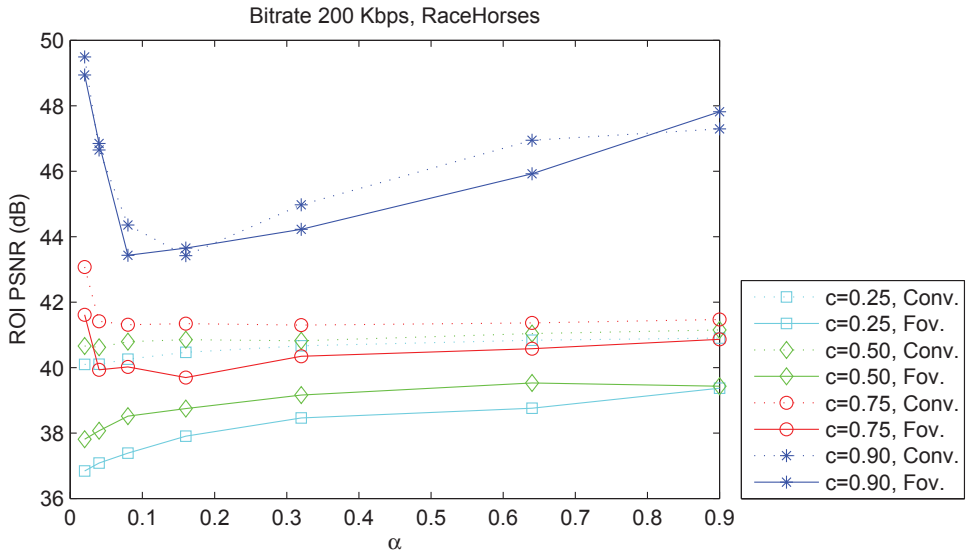


Figure A.8: Chart of Table A.8

Table A.9: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Cactus* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *Cactus*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	31.7964	32.9322	31.7199	33.3946	31.5203	33.9602	31.5860	34.0633
0.04	31.7239	33.0694	31.7074	33.6680	31.7065	34.3258	31.6797	34.5049
0.08	31.7895	33.2717	31.7212	33.8681	31.7288	34.4839	31.7665	34.8897
0.16	31.7289	33.4628	31.7720	34.0790	31.7160	34.6554	31.8891	35.3095
0.32	31.7186	33.5412	31.7582	34.0859	31.8123	34.8069	32.1061	35.7390
0.64	31.6595	33.5632	31.8048	34.3010	31.7714	34.9789	32.0265	35.6486
0.90	31.7877	33.7560	31.7635	34.3446	31.9012	35.2059	32.1213	35.7833

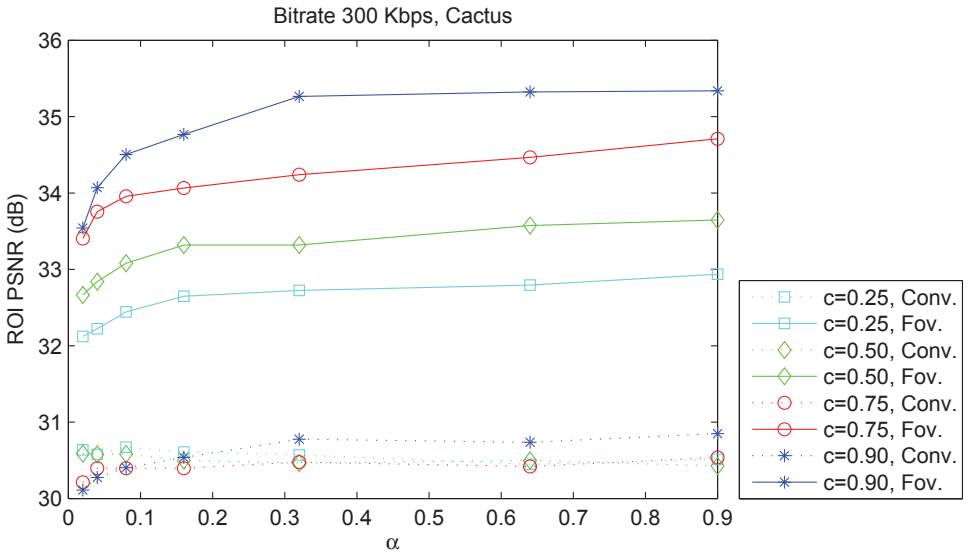


Figure A.9: Chart of Table A.9

Table A.10: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Tennis* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *Tennis*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	23.9736	23.9088	24.2372	24.1045	24.3959	24.1847	24.7533	24.2227
0.04	24.1560	24.0721	24.3077	24.1696	24.4617	24.2468	24.2464	24.0162
0.08	24.2692	24.1440	24.3701	24.2008	24.4928	24.2484	24.2893	24.0545
0.16	24.3263	24.1834	24.4405	24.2613	24.4716	24.2149	24.3998	24.1813
0.32	24.3881	24.2515	24.5118	24.3129	24.4461	24.1662	24.4953	24.2543
0.64	24.4492	24.2912	24.5182	24.3058	24.4389	24.1724	24.7798	24.4635
0.90	24.4834	24.3044	24.5242	24.3185	24.4292	24.1630	24.8836	24.4557

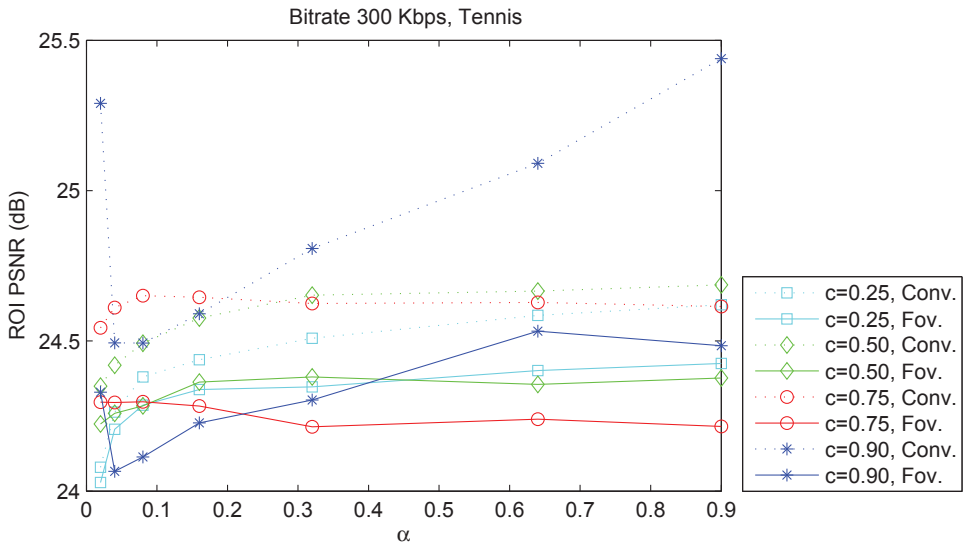


Figure A.10: Chart of Table A.10

Table A.11: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Vidyo4* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *Vidyo4*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	40.2259	39.8424	39.6314	39.4019	39.7371	39.9647	40.2518	40.8950
0.04	40.0057	39.6922	39.5264	39.4305	39.5101	39.9902	39.7643	40.2955
0.08	39.7433	39.4319	39.4169	39.3589	39.4858	40.1440	39.6134	40.5112
0.16	39.5381	39.2271	39.4147	39.3933	39.4303	40.1679	39.4485	40.3641
0.32	39.4021	39.0401	39.4016	39.4776	39.4342	40.3292	39.3249	40.2871
0.64	39.4107	39.0987	39.3719	39.3191	39.4472	40.2964	39.2597	40.1071
0.90	39.4144	39.0986	39.3612	39.5106	39.4384	40.0855	39.3049	40.2047

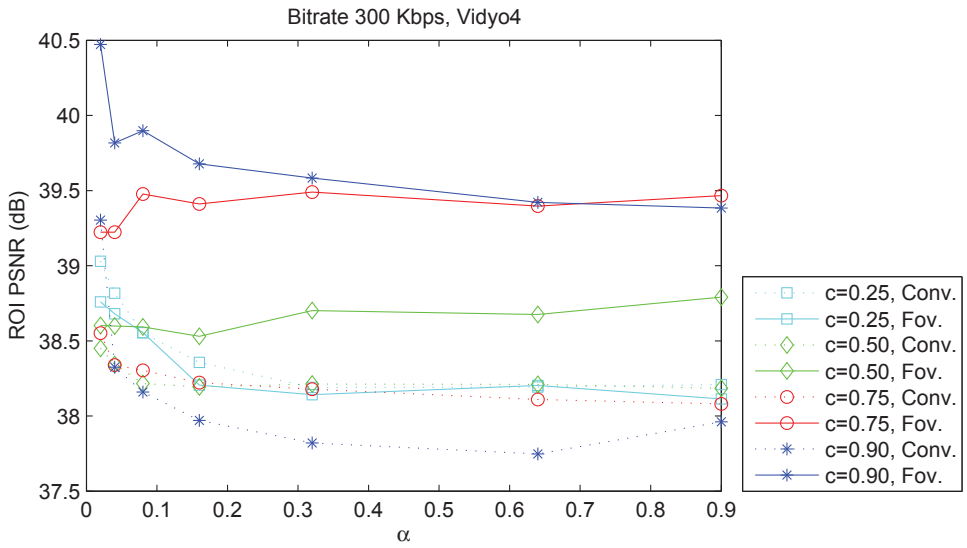


Figure A.11: Chart of Table A.11

Table A.12: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Johnny* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *Johnny*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	40.7208	38.2978	40.3268	38.6360	40.6178	40.1244	40.9374	40.9758
0.04	40.4881	38.3662	40.2553	38.7886	40.6315	40.2754	41.3208	41.6968
0.08	40.3658	38.3403	40.2315	38.9336	40.6771	40.6677	41.3759	41.9082
0.16	40.2162	38.2685	40.2726	39.1681	40.7108	40.9302	41.4602	42.0849
0.32	40.2159	38.2164	40.4023	39.3222	40.7558	40.7340	41.4894	42.0718
0.64	40.2797	38.5291	40.6141	39.9902	40.9755	40.8359	41.5175	42.1128
0.90	40.3516	38.7659	40.6976	40.1503	40.9813	40.8092	41.5404	42.1460

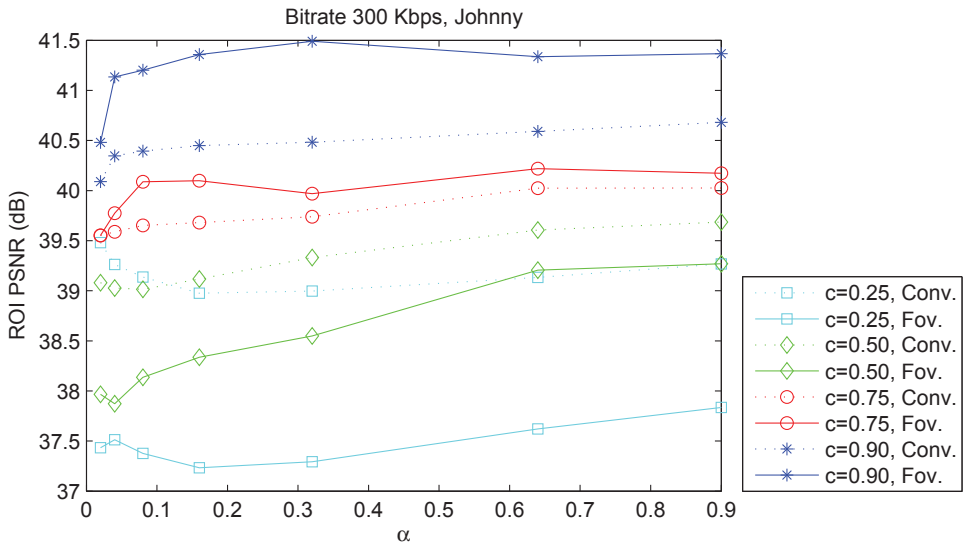


Figure A.12: Chart of Table A.12

Table A.13: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *ChinaSpeed* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *ChinaSpeed*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	33.3226	33.5067	34.7130	35.5742	36.5913	38.1063	39.2296	41.7145
0.04	33.3084	34.0924	34.5253	35.3931	36.2003	38.0078	36.6685	40.0098
0.08	33.7193	34.1876	34.7894	35.7446	36.3170	38.5586	36.7767	40.4060
0.16	34.4096	35.4382	35.8855	37.6292	36.2789	39.3541	36.3654	40.6734
0.32	35.0990	36.4661	36.0031	37.8045	36.7432	40.1220	36.3300	40.1090
0.64	35.8463	37.6126	35.9853	38.2069	36.5761	40.2112	36.3728	41.4148
0.90	35.7099	37.7334	36.2856	38.9309	36.6505	40.3698	36.6100	41.6606

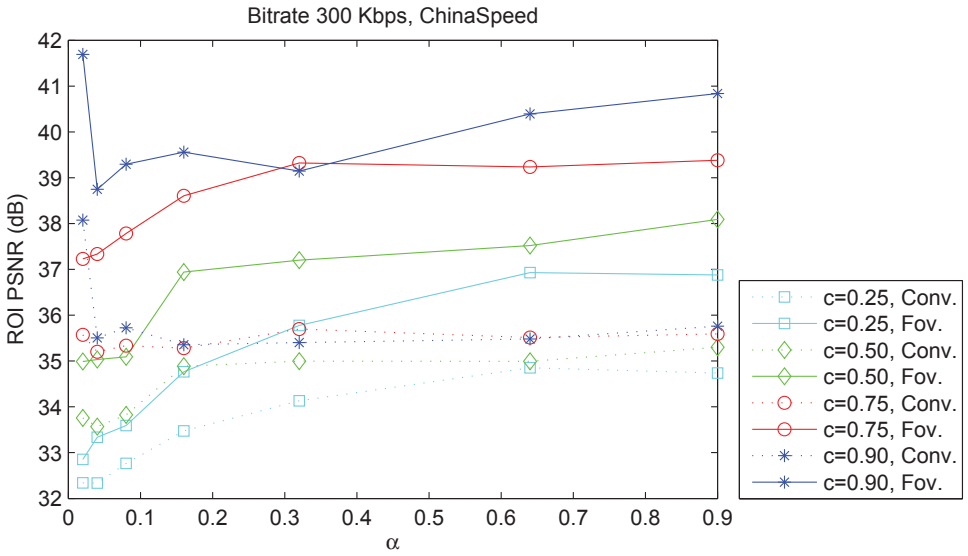


Figure A.13: Chart of Table A.13

Table A.14: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *BasketballDrill* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *BasketballDrill*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	27.9818	27.9461	27.8075	27.8041	27.8065	27.9007	32.4097	32.3260
0.04	27.8411	27.7733	27.5854	27.5687	27.2343	27.3183	30.6035	31.1721
0.08	27.6598	27.6393	27.4327	27.3926	27.1137	27.1652	29.4279	29.4264
0.16	27.5158	27.4659	27.3146	27.2526	27.1294	27.1412	29.6497	29.5305
0.32	27.4322	27.3514	27.1938	27.1359	27.3192	27.3469	30.4696	30.8145
0.64	27.3321	27.2681	27.1521	27.1534	27.5015	27.5115	32.7833	33.1905
0.90	27.2624	27.1824	27.1182	27.0829	27.8091	27.7922	33.1100	32.7784

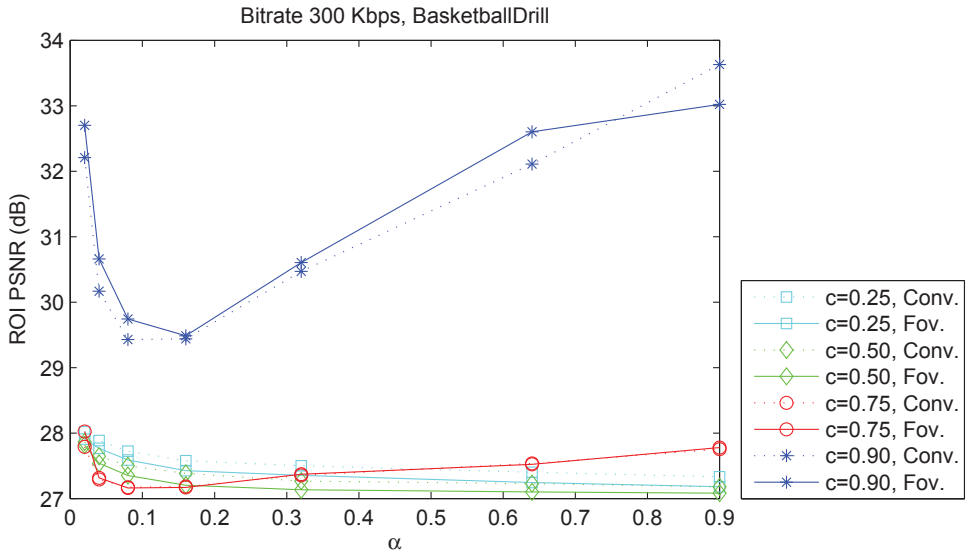


Figure A.14: Chart of Table A.14

Table A.15: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *PartyScene* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *PartyScene*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	30.8528	31.8376	30.8784	32.0915	30.1527	31.6256	28.1157	29.4451
0.04	30.8365	32.1327	30.8366	32.4460	30.4371	32.3834	29.9004	31.5911
0.08	30.7968	32.3897	30.7635	32.6952	30.3859	32.7733	29.4147	32.1618
0.16	30.7940	32.6644	30.7116	32.8170	30.2303	32.9339	29.8737	33.0375
0.32	30.7434	32.6838	30.6007	33.0695	30.1194	33.0693	30.4870	33.5043
0.64	30.7038	32.7921	30.5031	32.9470	30.1039	33.1955	30.9834	33.9124
0.90	30.6962	33.0237	30.4505	33.0661	30.0712	33.1948	30.9736	33.0292

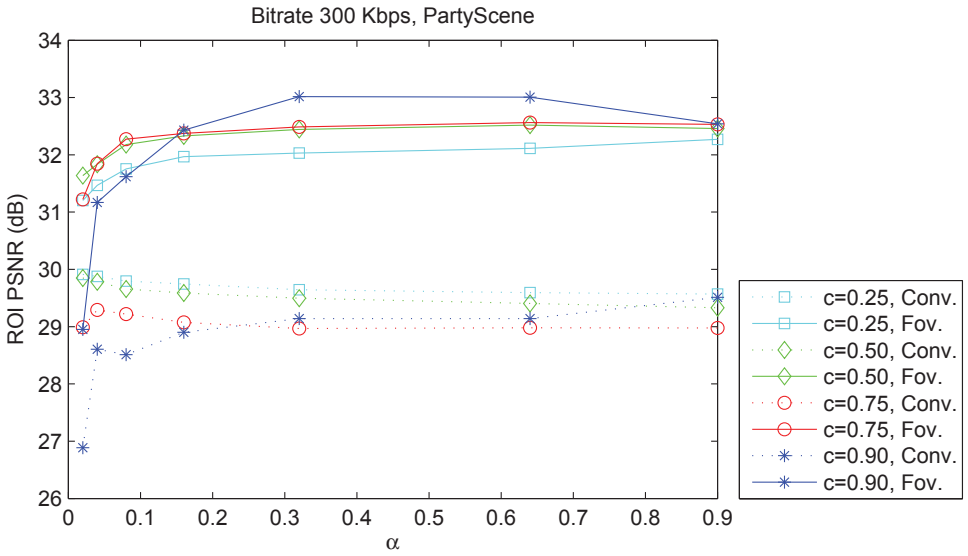


Figure A.15: Chart of Table A.15

Table A.16: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *RaceHorses* with bit rate = 300 Kbps.

Bit rate 300 Kbps, *RaceHorses*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	35.7691	34.6084	36.5561	35.5988	39.6629	39.5179	44.9040	47.1029
0.04	35.8008	34.7825	36.5066	35.8733	37.7353	37.4426	42.9250	44.4445
0.08	36.0057	35.0908	36.6340	36.2702	37.4248	37.3058	40.4028	41.0936
0.16	36.2606	35.5679	36.7166	36.2671	37.3314	37.5346	39.7043	40.3298
0.32	36.4922	36.0173	36.7533	36.5522	37.7166	37.8956	41.3700	41.3029
0.64	36.7130	36.2406	36.9977	36.9000	37.5912	37.5584	42.8749	43.2882
0.90	36.9872	36.5410	37.0573	36.8108	38.0592	37.7508	43.2285	44.6658

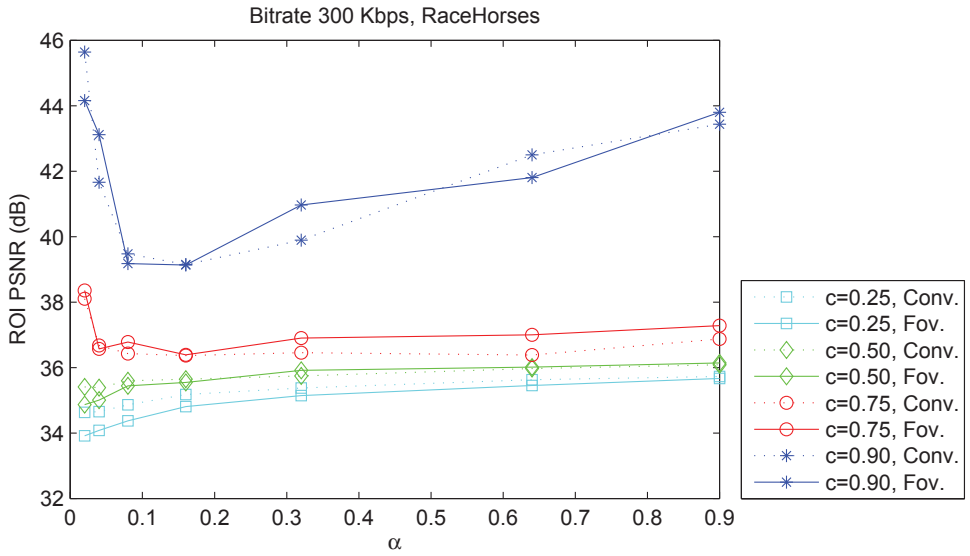


Figure A.16: Chart of Table A.16

Table A.17: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Cactus* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *Cactus*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	32.9851	34.0677	32.9042	34.3461	32.7579	34.6411	32.6683	34.7390
0.04	32.9243	34.1727	32.9136	34.5630	32.9186	34.9943	32.8481	35.2687
0.08	32.9718	34.3587	32.9593	34.7155	32.9335	35.1130	32.9720	35.5982
0.16	32.9415	34.4800	32.9714	34.9125	32.9306	35.2709	33.0540	36.0002
0.32	32.9727	34.5630	32.9580	34.8730	33.0010	35.4704	33.2067	36.5139
0.64	32.9133	34.5280	33.0109	35.0747	32.9687	35.5509	33.1392	36.3705
0.90	32.9927	34.6751	32.9737	35.1277	33.0618	35.7987	33.2031	36.5536

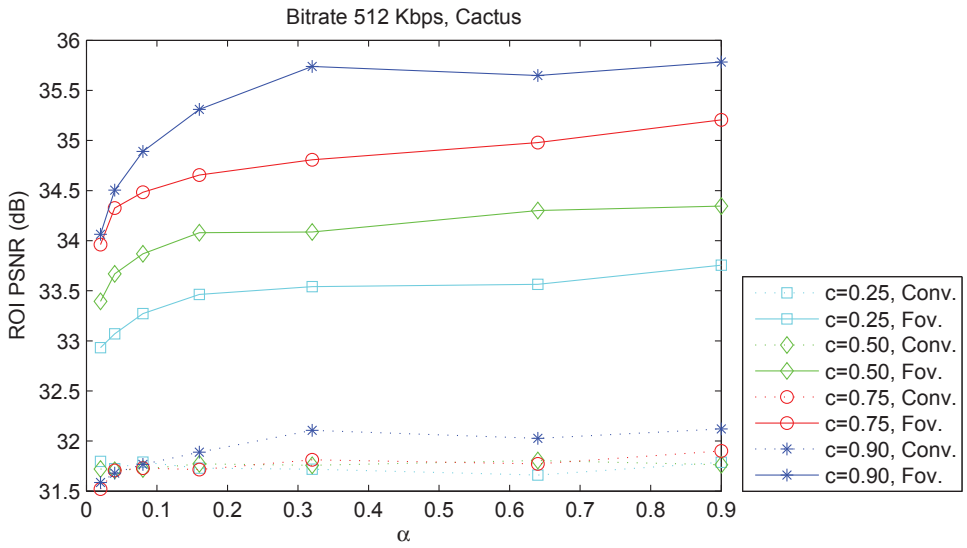


Figure A.17: Chart of Table A.17

Table A.18: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Tennis* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *Tennis*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	23.8801	23.8253	24.1328	24.0331	24.2603	24.1468	24.2565	24.2049
0.04	24.0530	23.9759	24.1939	24.0760	24.3179	24.1957	24.0735	23.9692
0.08	24.1625	24.0632	24.2508	24.1267	24.3380	24.2102	24.1300	24.0343
0.16	24.2107	24.1043	24.3154	24.1717	24.3080	24.1716	24.2250	24.1454
0.32	24.2672	24.1481	24.3734	24.2194	24.2752	24.1295	24.3156	24.2129
0.64	24.3229	24.1942	24.3675	24.2243	24.2636	24.1223	24.4975	24.3894
0.90	24.3505	24.2215	24.3726	24.2323	24.2552	24.1233	24.7078	24.4124

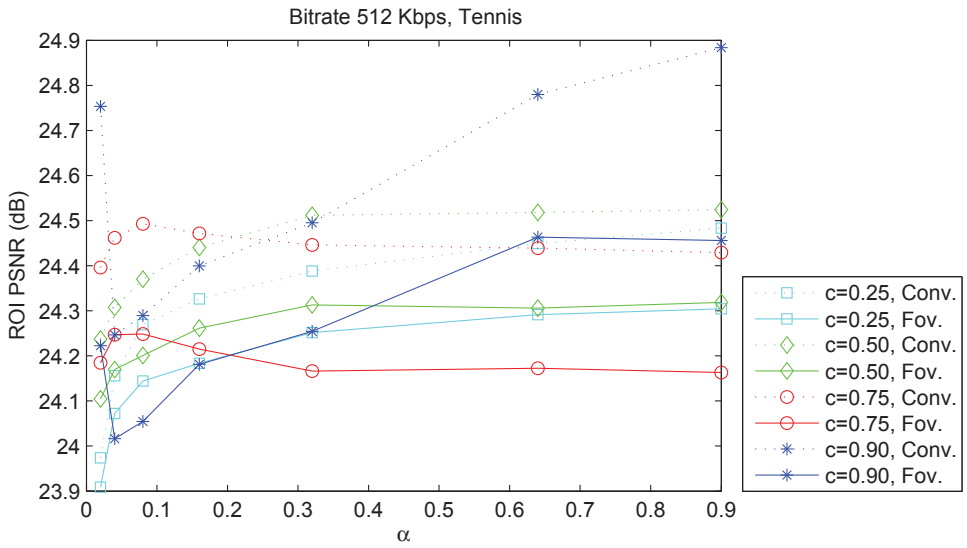


Figure A.18: Chart of Table A.18

Table A.19: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Vidyo4* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *Vidyo4*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	41.5661	40.8417	40.9715	40.3888	40.9092	40.7748	41.2739	42.1370
0.04	41.3519	40.7024	40.8615	40.4474	40.7025	40.7934	40.7634	41.4249
0.08	41.0836	40.5264	40.7298	40.3781	40.6608	40.9661	40.6250	41.5307
0.16	40.8742	40.3383	40.6711	40.4247	40.5724	40.9094	40.4924	41.4891
0.32	40.7103	40.1822	40.6307	40.4092	40.5314	40.9639	40.3770	41.4439
0.64	40.6609	40.1638	40.5783	40.3521	40.5134	40.9678	40.3011	41.3449
0.90	40.6536	40.2117	40.5548	40.4283	40.5047	41.0674	40.4291	41.2951

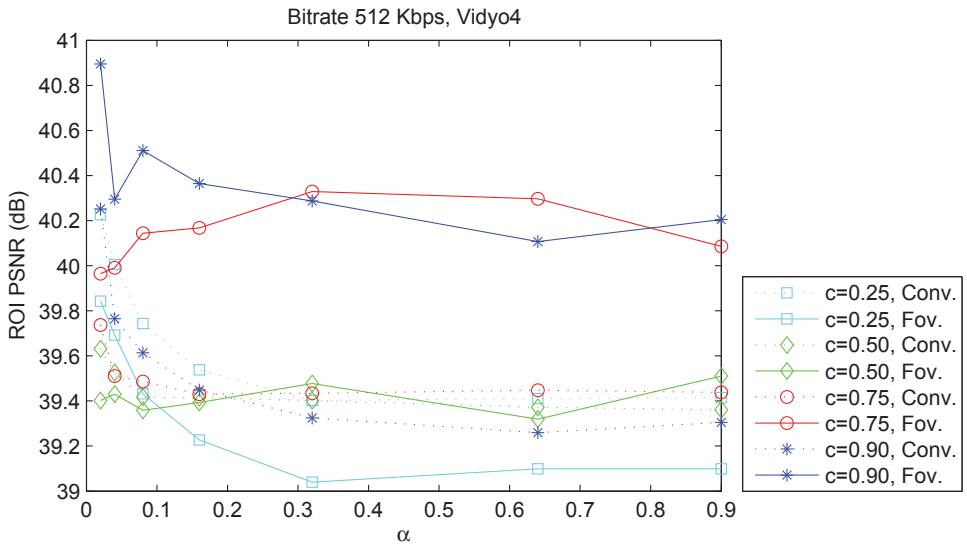


Figure A.19: Chart of Table A.19

Table A.20: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Johnny* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *Johnny*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	41.8923	39.1150	41.5092	39.4921	41.6757	40.9684	41.6889	42.0241
0.04	41.6691	39.2287	41.4329	39.5425	41.6969	41.1070	42.0744	42.4348
0.08	41.5515	39.2409	41.3915	39.8539	41.7207	41.4561	42.1334	42.7649
0.16	41.3934	39.1424	41.4123	40.0947	41.7039	41.7392	42.1835	42.8793
0.32	41.3729	39.2006	41.5330	40.2737	41.7045	41.6893	42.1874	42.9709
0.64	41.4199	39.4580	41.6881	40.9009	41.8581	41.8277	42.1883	42.9277
0.90	41.4881	39.6983	41.7419	41.0219	41.8497	41.7896	42.2272	42.8410

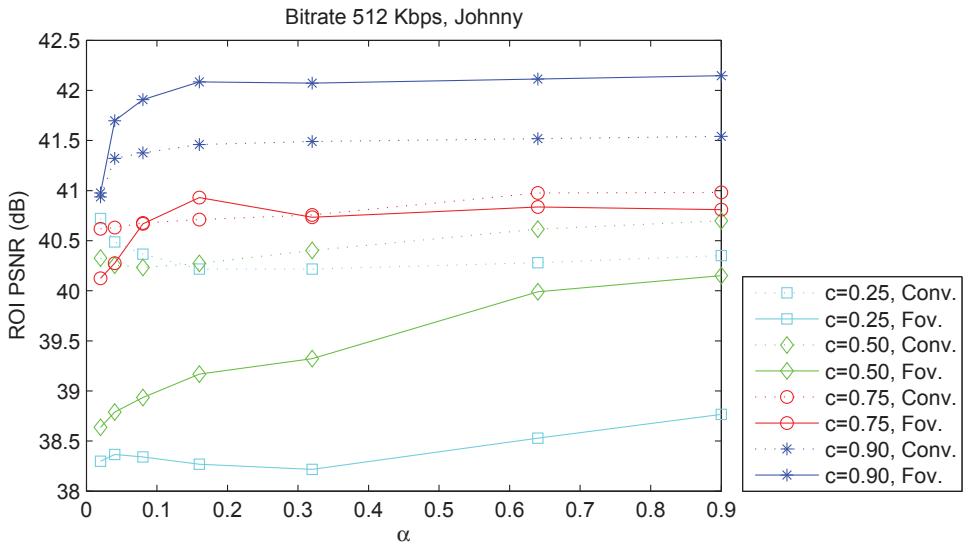


Figure A.20: Chart of Table A.20

Table A.21: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *ChinaSpeed* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *ChinaSpeed*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	34.5716	34.2148	35.8109	36.5930	37.3972	39.0973	40.0376	42.9848
0.04	34.5476	35.0617	35.6306	36.4667	37.0654	39.1128	37.4251	40.9949
0.08	34.9317	35.0416	35.8405	36.4879	37.1493	39.7984	37.4655	42.1313
0.16	35.5439	36.3485	36.8083	38.4905	37.1396	40.5583	37.1195	42.2003
0.32	36.1306	37.4301	36.8789	39.0009	37.4817	41.5279	37.0842	41.3099
0.64	36.7678	38.7577	36.8648	39.5443	37.3628	41.5672	37.1337	42.8571
0.90	36.6367	38.8476	37.1223	40.2496	37.3810	41.5884	37.3628	43.0355

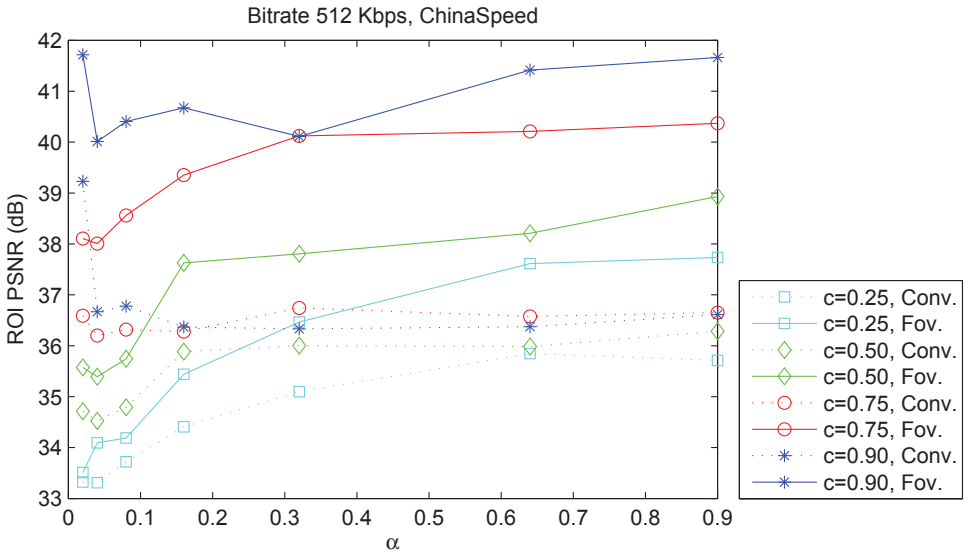


Figure A.21: Chart of Table A.21

Table A.22: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *BasketballDrill* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *BasketballDrill*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	27.9964	27.9553	27.8044	27.8213	27.7017	27.9152	32.7288	32.7967
0.04	27.8553	27.8273	27.5734	27.5784	27.1955	27.3592	30.8070	30.6583
0.08	27.6635	27.6448	27.4029	27.3835	27.0583	27.1460	29.5294	29.5115
0.16	27.5062	27.4768	27.2771	27.2601	27.0663	27.1529	29.4182	29.6014
0.32	27.4085	27.3599	27.1450	27.1343	27.2543	27.3584	30.8504	30.8161
0.64	27.2954	27.2788	27.1022	27.1025	27.4154	27.5351	33.0005	32.4517
0.90	27.2135	27.1916	27.0624	27.0388	27.6671	27.8182	34.2021	33.7787

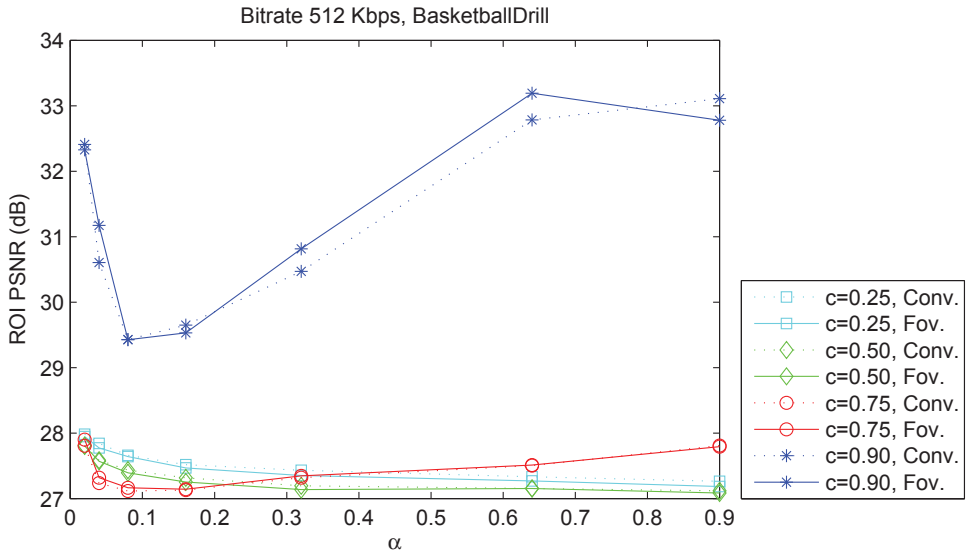


Figure A.22: Chart of Table A.22

Table A.23: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *PartyScene* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *PartyScene*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	31.9333	32.6917	31.9471	32.9123	31.0156	32.4401	29.3143	29.9415
0.04	31.9153	32.9840	31.8910	33.1742	31.3557	33.1896	30.3298	32.5459
0.08	31.8763	33.2738	31.7843	33.4549	31.2915	33.7484	30.1382	33.0004
0.16	31.8462	33.5250	31.7083	33.6466	31.1342	33.8515	30.5422	33.9123
0.32	31.7781	33.4840	31.6031	33.8320	30.9846	33.9564	30.9083	34.2379
0.64	31.6975	33.5608	31.4850	33.8878	30.9165	34.1524	30.9473	34.0564
0.90	31.6934	33.8726	31.4128	33.9935	30.8597	34.1787	31.1698	33.8413

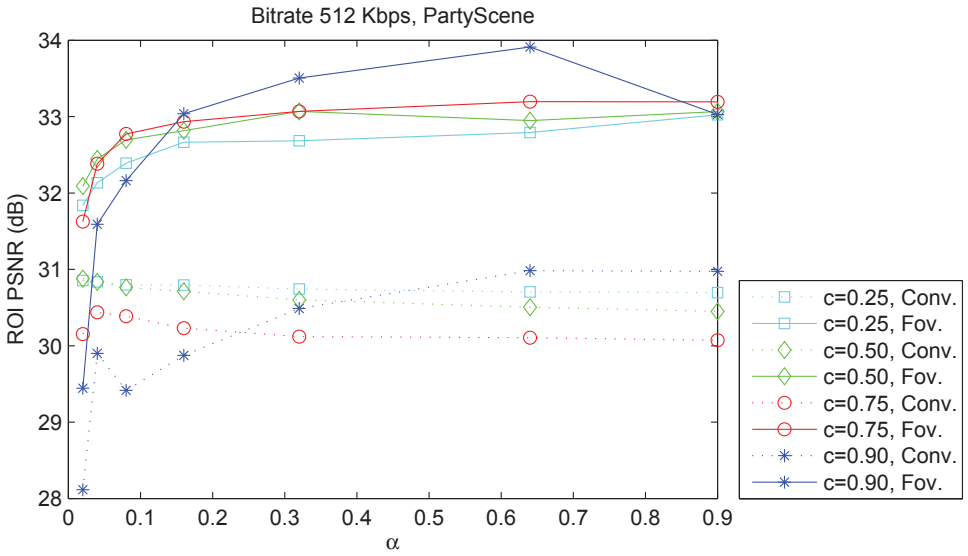


Figure A.23: Chart of Table A.23

Table A.24: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *RaceHorses* with bit rate = 512 Kbps.

Bit rate 512 Kbps, *RaceHorses*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	37.5027	35.5823	38.1012	36.5913	40.9441	40.0456	46.3654	46.9069
0.04	37.4884	35.8076	38.0695	36.9023	39.1932	38.2729	43.4974	44.9072
0.08	37.6513	36.1149	38.2818	37.2499	39.0850	38.4580	42.0788	42.1050
0.16	37.9868	36.5555	38.2635	37.2873	38.8495	38.3802	41.4530	41.0670
0.32	38.1350	36.9568	38.4496	37.6706	39.6390	38.9211	42.6181	43.8624
0.64	38.3257	37.3765	38.8185	37.8849	38.9617	38.9487	43.9092	45.1658
0.90	38.9239	38.0159	38.8092	37.8452	39.6483	39.1477	44.9270	44.8468

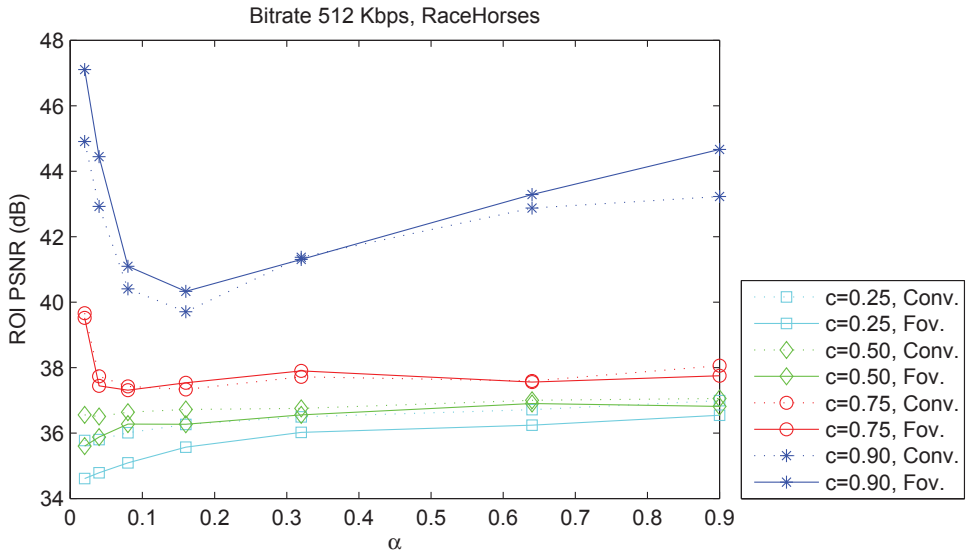


Figure A.24: Chart of Table A.24

Table A.25: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Cactus* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *Cactus*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	33.7750	34.7722	33.6991	34.9325	33.6061	35.0756	33.6277	35.2131
0.04	33.7274	34.8501	33.7177	35.1141	33.7783	35.3975	33.8092	35.5843
0.08	33.7635	35.0050	33.7610	35.2468	33.8112	35.6179	33.9216	36.0872
0.16	33.7476	35.1193	33.7953	35.4457	33.8132	35.7273	34.0252	36.4271
0.32	33.7835	35.1318	33.7949	35.3833	33.9133	35.9266	34.2263	36.9248
0.64	33.7380	35.0723	33.8623	35.6129	33.8944	36.0830	34.1648	36.8780
0.90	33.8167	35.2297	33.8420	35.6361	34.0133	36.3353	34.2842	37.0947

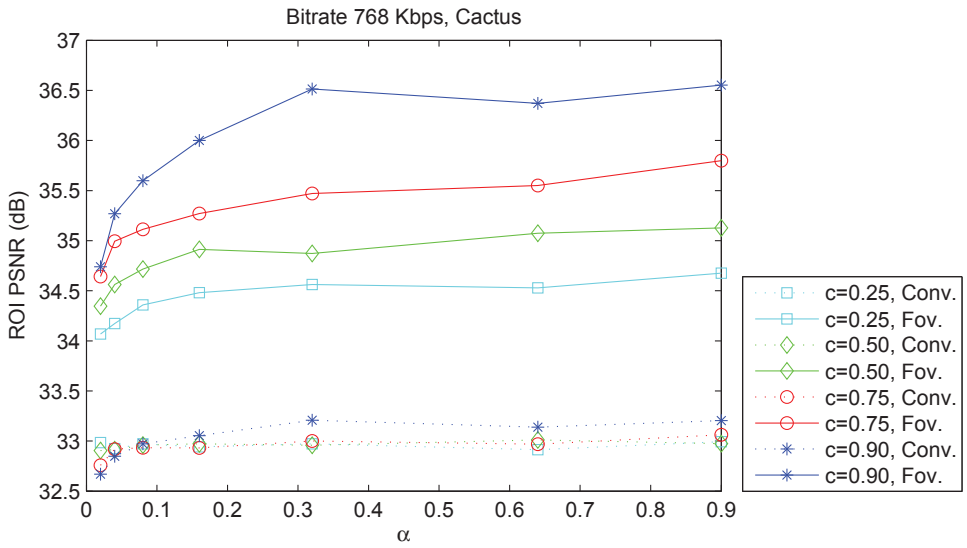


Figure A.25: Chart of Table A.25

Table A.26: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Tennis* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *Tennis*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	23.8384	23.7981	24.0790	24.0093	24.2182	24.1280	24.2602	24.1412
0.04	24.0034	23.9429	24.1416	24.0505	24.2718	24.1568	24.0434	23.9369
0.08	24.1091	24.0255	24.1998	24.0977	24.2941	24.1862	24.0934	24.0009
0.16	24.1591	24.0578	24.2677	24.1635	24.2653	24.1461	24.2137	24.1024
0.32	24.2167	24.1107	24.3238	24.2104	24.2380	24.1124	24.2797	24.1760
0.64	24.2740	24.1522	24.3221	24.2008	24.2318	24.1016	24.4551	24.3404
0.90	24.3010	24.1893	24.3279	24.1985	24.2238	24.1111	24.5069	24.3760

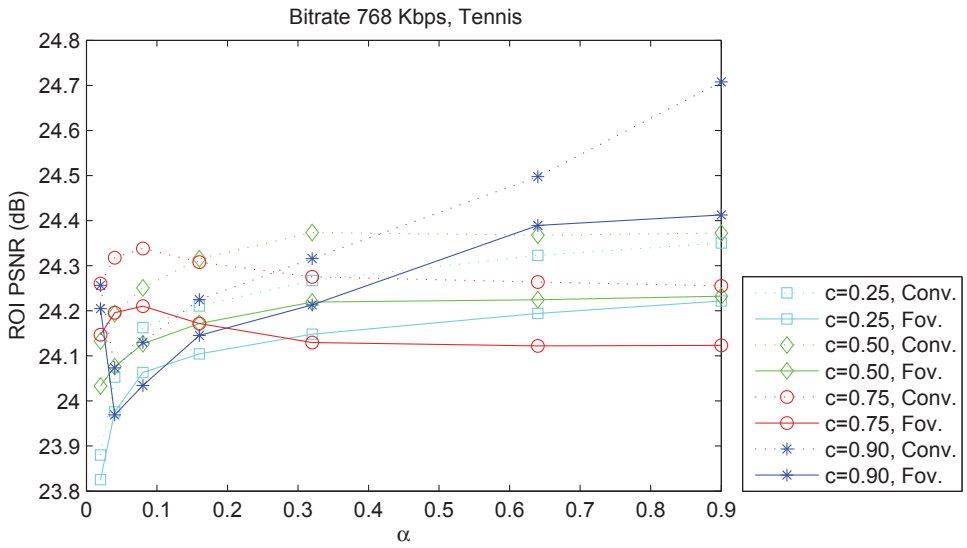


Figure A.26: Chart of Table A.26

Table A.27: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Vidyo4* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *Vidyo4*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	42.5091	41.5941	41.9561	41.2122	41.8231	41.5896	42.1886	42.5507
0.04	42.3070	41.5025	41.8481	41.2589	41.6550	41.5864	41.6377	42.0347
0.08	42.0570	41.2083	41.7165	41.1568	41.6174	41.6786	41.4887	42.1516
0.16	41.8564	40.9386	41.6637	41.1041	41.5313	41.6386	41.3118	42.1548
0.32	41.6987	40.8171	41.6219	41.0395	41.4856	41.7683	41.1605	42.2600
0.64	41.6605	40.8607	41.5580	41.0580	41.4350	41.7664	41.1233	42.1485
0.90	41.6537	40.8331	41.5359	41.1303	41.4037	41.9337	41.1346	42.0505

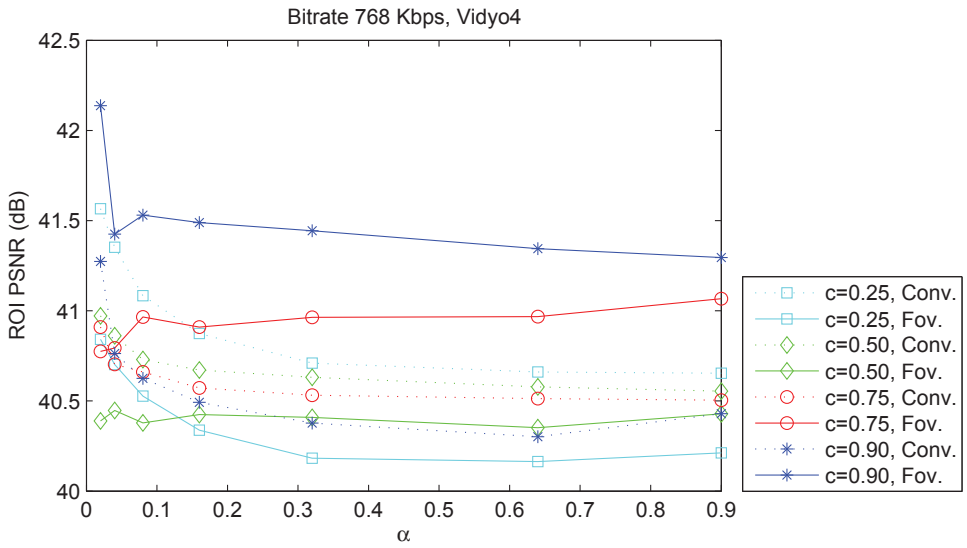


Figure A.27: Chart of Table A.27

Table A.28: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Johnny* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *Johnny*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	42.7137	39.6995	42.3139	40.0281	42.4353	41.5430	42.2866	42.4640
0.04	42.4875	39.7793	42.2247	40.2336	42.4461	41.6792	42.6592	43.2018
0.08	42.3634	39.9830	42.1927	40.4765	42.4535	42.0628	42.7175	43.3560
0.16	42.1935	39.8533	42.2034	40.6452	42.4207	42.2869	42.7705	43.5361
0.32	42.1600	39.8431	42.3120	40.8639	42.3887	42.2562	42.7619	43.5003
0.64	42.2109	40.1635	42.4167	41.5707	42.5128	42.3743	42.7655	43.5200
0.90	42.2765	40.3715	42.4658	41.7763	42.5059	42.4388	42.7864	43.6444

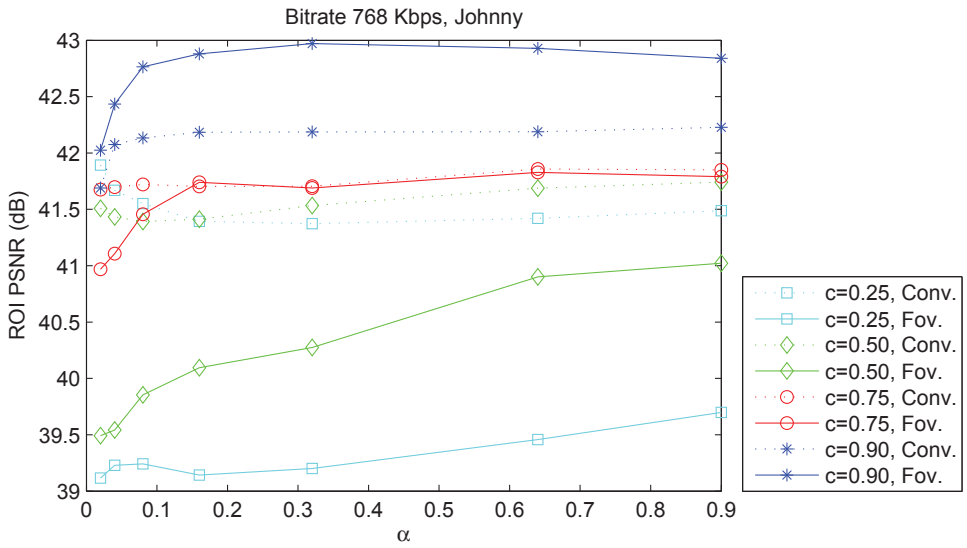


Figure A.28: Chart of Table A.28

Table A.29: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *ChinaSpeed* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *ChinaSpeed*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	36.0159	34.8927	37.1462	37.4703	38.6168	40.1117	41.0726	43.5030
0.04	36.0058	36.0196	37.0015	37.2337	38.2780	39.9702	38.6146	41.8704
0.08	36.3640	35.7091	37.2026	37.1025	38.3238	40.5620	38.7331	42.9283
0.16	36.9227	37.0705	38.0803	39.2843	38.2958	41.6438	38.4067	43.2429
0.32	37.4512	38.1296	38.1441	39.9770	38.7210	42.6249	38.3633	42.0921
0.64	38.0390	39.5444	38.0670	40.4314	38.5953	42.8240	38.3261	44.0301
0.90	37.9138	39.7829	38.3006	41.0829	38.6515	42.7974	38.5493	44.0256

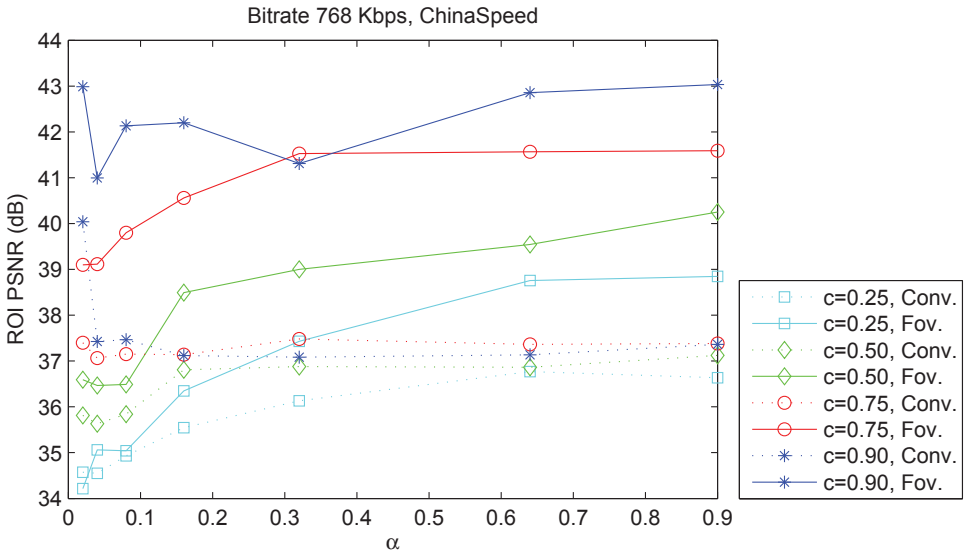


Figure A.29: Chart of Table A.29

Table A.30: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *BasketballDrill* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *BasketballDrill*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	27.9938	27.9655	27.7940	27.8352	27.7242	27.9397	32.6957	33.1810
0.04	27.8459	27.8238	27.5625	27.5791	27.2018	27.3264	30.7675	30.6266
0.08	27.6502	27.6376	27.3899	27.3998	27.0637	27.1613	29.4577	29.7013
0.16	27.4954	27.4674	27.2616	27.2624	27.0826	27.1412	29.4792	29.7194
0.32	27.3945	27.3592	27.1344	27.1210	27.2922	27.3530	30.7118	31.1341
0.64	27.2831	27.2626	27.0958	27.1142	27.4555	27.5380	32.7335	32.3831
0.90	27.2006	27.1950	27.0606	27.0610	27.7356	27.8267	34.2101	33.8961

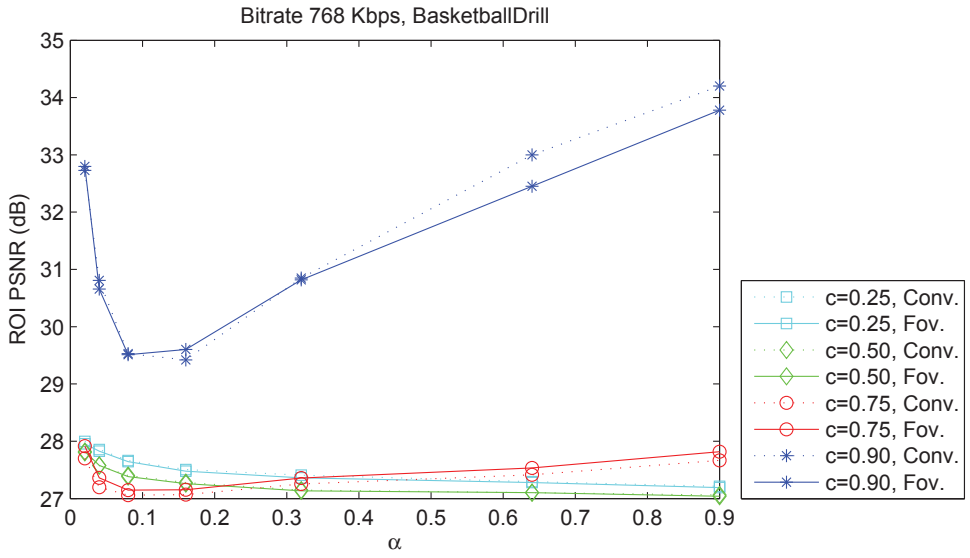


Figure A.30: Chart of Table A.30

Table A.31: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *PartyScene* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *PartyScene*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	32.9217	33.2028	33.0243	33.5121	32.2899	33.1184	30.0919	30.3373
0.04	32.9409	33.5100	32.9921	33.7509	32.5971	33.8016	31.6152	33.4436
0.08	32.9528	33.8231	32.9164	34.0605	32.5260	34.3153	31.4495	33.8499
0.16	32.9587	34.0386	32.8897	34.2502	32.3760	34.5548	31.7882	34.8079
0.32	32.9131	34.0751	32.8117	34.4611	32.2584	34.6010	32.0473	35.2599
0.64	32.8834	34.2067	32.7074	34.6215	32.2072	34.8621	32.4330	35.1932
0.90	32.8930	34.4792	32.6212	34.5633	32.1691	34.8646	32.3703	34.6629

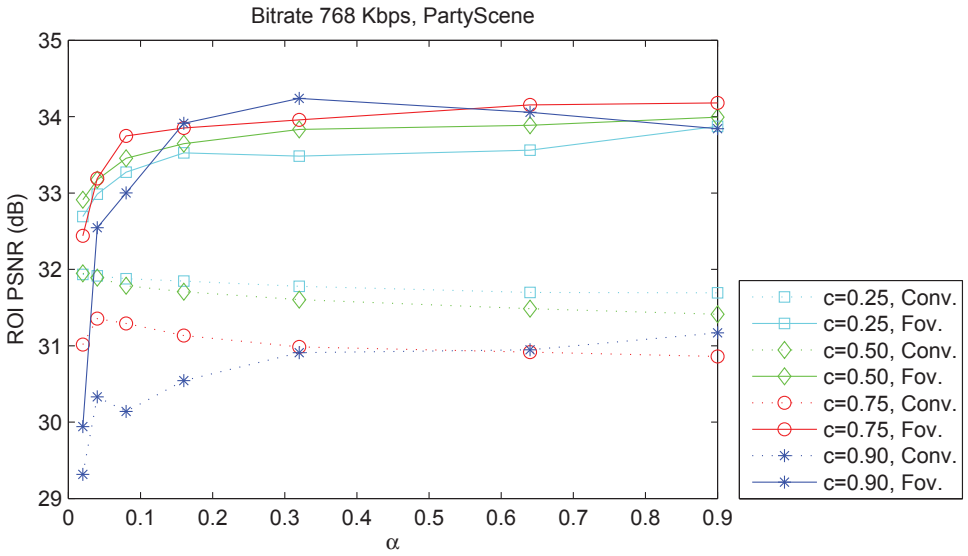


Figure A.31: Chart of Table A.31

Table A.32: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *RaceHorses* with bit rate = 768 Kbps.

Bit rate 768 Kbps, *RaceHorses*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	39.0257	36.3305	39.5899	37.3531	41.9566	41.5990	48.8919	49.3731
0.04	39.0170	36.5578	39.5224	37.5550	40.4670	39.0704	45.3402	45.4287
0.08	39.1747	36.8921	39.7988	38.0843	40.0715	39.7683	43.2951	42.8734
0.16	39.3778	37.3640	39.7186	38.1975	39.9937	39.1742	42.8450	42.9984
0.32	39.6102	37.7664	39.7409	38.5785	40.2039	39.7332	44.0050	43.2195
0.64	39.6603	38.1989	39.8530	38.7549	40.3375	40.2006	46.4501	45.8320
0.90	39.9657	38.5873	39.8458	38.6740	40.0806	40.3869	47.3146	46.4707

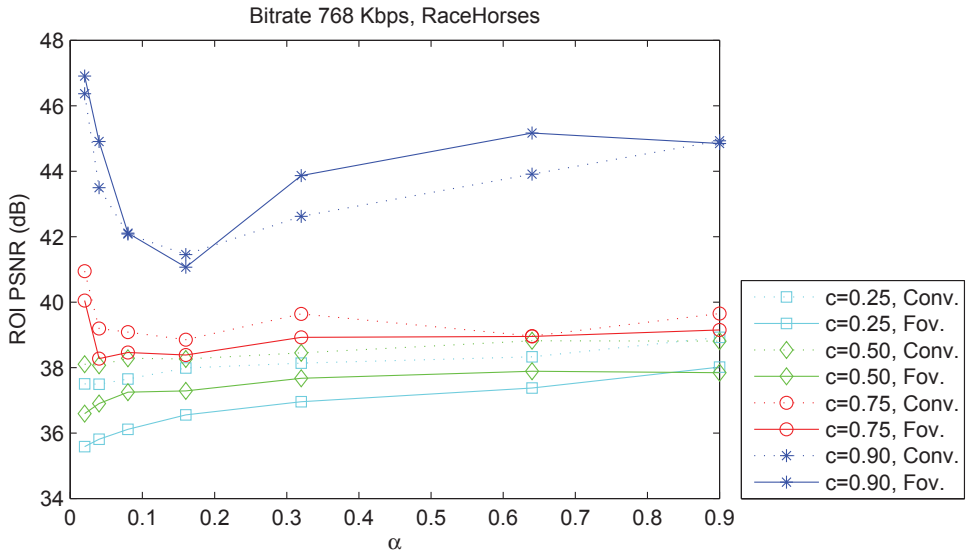


Figure A.32: Chart of Table A.32

Table A.33: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Cactus* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *Cactus*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	34.1943	35.1517	34.1333	35.2928	33.9755	35.4259	33.9460	35.6185
0.04	34.1549	35.2200	34.1412	35.4666	34.1626	35.8218	34.1613	36.1751
0.08	34.1903	35.3762	34.1794	35.5512	34.1837	36.0053	34.2954	36.6066
0.16	34.1686	35.4609	34.1869	35.7099	34.1892	36.0939	34.4316	36.9894
0.32	34.1924	35.4869	34.1865	35.7396	34.2928	36.3262	34.6484	37.4322
0.64	34.1458	35.4532	34.2410	35.9375	34.2789	36.3895	34.5626	37.3480
0.90	34.2106	35.6022	34.2226	35.9632	34.4107	36.6417	34.7082	37.4969

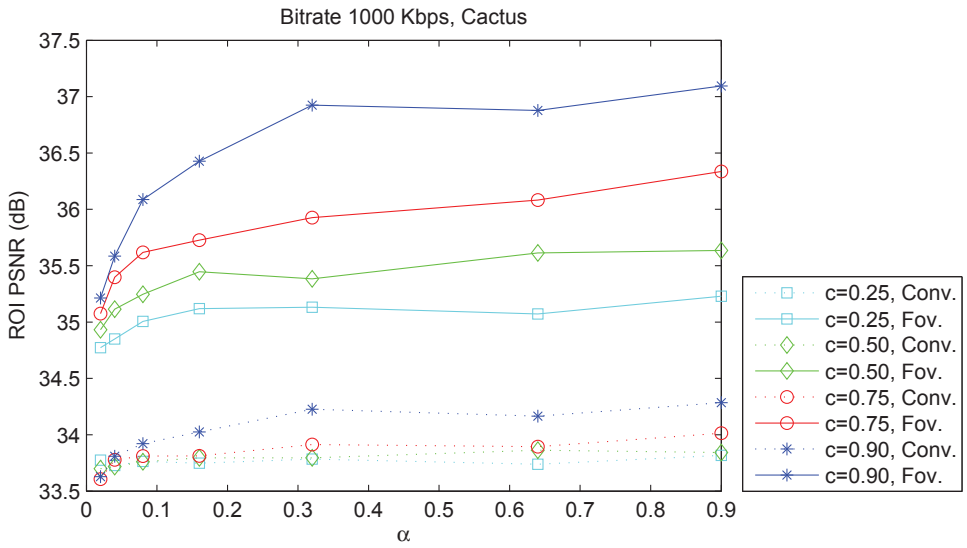


Figure A.33: Chart of Table A.33

Table A.34: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Tennis* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *Tennis*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	23.8138	23.7778	24.0471	23.9987	24.1700	24.0993	24.1567	24.1728
0.04	23.9744	23.9224	24.1038	24.0225	24.2255	24.1430	23.9883	23.9243
0.08	24.0777	24.0111	24.1586	24.0776	24.2478	24.1625	24.0484	23.9824
0.16	24.1210	24.0541	24.2250	24.1335	24.2178	24.1370	24.1470	24.0768
0.32	24.1754	24.0885	24.2827	24.1910	24.1903	24.0942	24.2197	24.1696
0.64	24.2318	24.1416	24.2802	24.1872	24.1832	24.0962	24.3689	24.3353
0.90	24.2605	24.1774	24.2839	24.1981	24.1737	24.0766	24.3463	24.3189

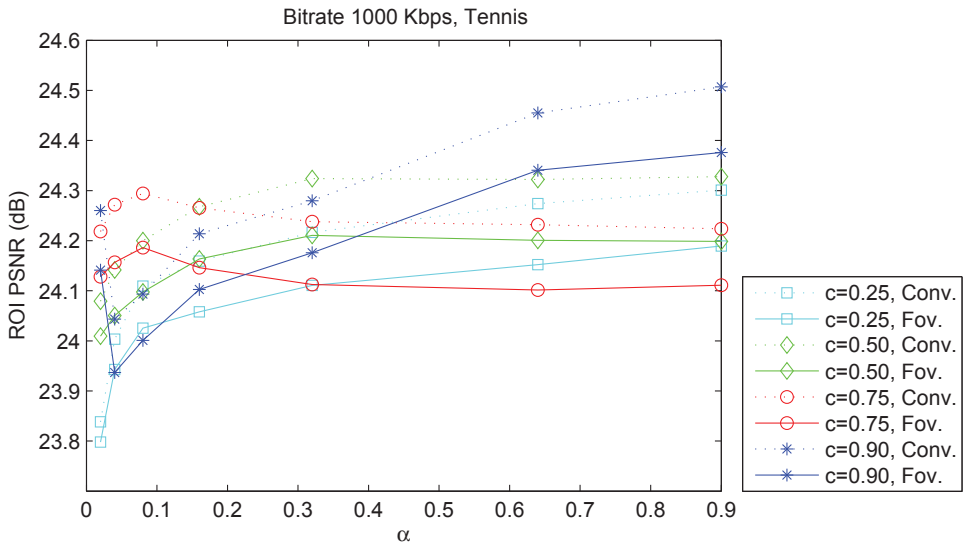


Figure A.34: Chart of Table A.34

Table A.35: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Vidyo4* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *Vidyo4*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	43.0700	42.0815	42.5423	41.5943	42.3388	41.9375	42.7448	43.2246
0.04	42.8758	42.0409	42.4316	41.6671	42.1803	41.8721	42.2545	42.6041
0.08	42.6407	41.8015	42.2914	41.5395	42.1413	41.9650	42.1238	42.6697
0.16	42.4385	41.4873	42.2185	41.5484	42.0564	41.9493	41.9873	42.7549
0.32	42.2731	41.3573	42.1564	41.4738	42.0254	42.1298	41.8952	42.7819
0.64	42.2099	41.4097	42.0848	41.4215	42.0145	42.2323	41.8132	42.6542
0.90	42.1931	41.3717	42.0547	41.4991	41.9926	42.2331	41.8400	42.8186

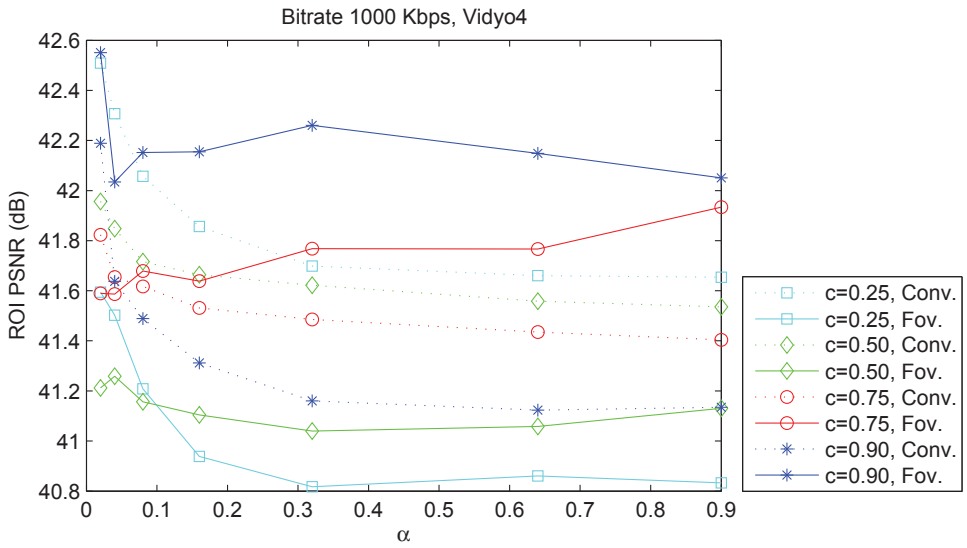


Figure A.35: Chart of Table A.35

Table A.36: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *Johnny* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *Johnny*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	43.2398	40.0721	42.8263	40.3472	42.8409	41.9522	42.5881	43.0133
0.04	43.0332	40.1292	42.7350	40.5985	42.8640	41.9022	42.9599	43.5752
0.08	42.8909	40.3391	42.6854	40.8378	42.8670	42.3941	43.0492	43.7352
0.16	42.7058	40.2192	42.6845	41.0390	42.8405	42.6695	43.0962	43.9357
0.32	42.6629	40.2244	42.7527	41.2737	42.8256	42.6601	43.0533	44.0097
0.64	42.6894	40.5544	42.8413	41.9542	42.9156	42.7919	43.0250	43.9680
0.90	42.7286	40.7548	42.8870	42.1725	42.9077	42.8626	43.0507	44.0115

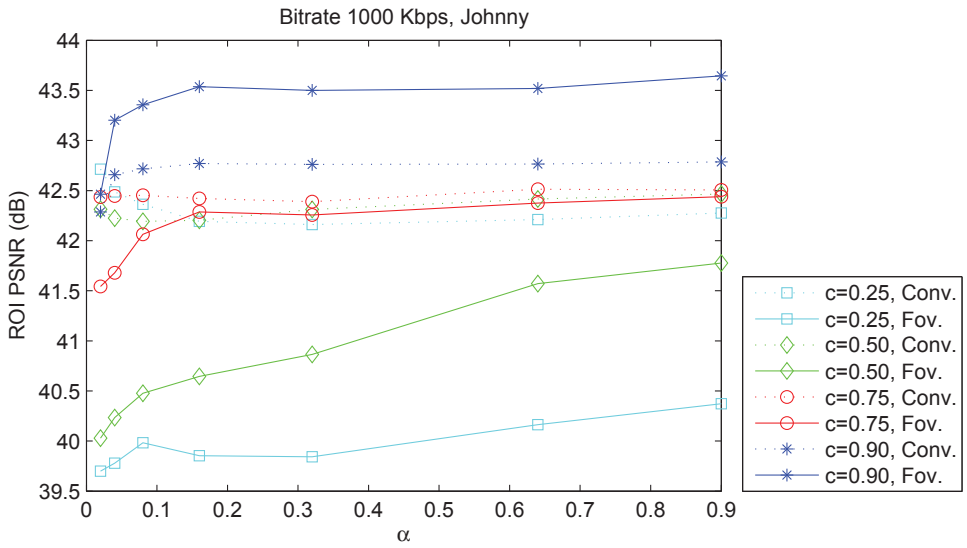


Figure A.36: Chart of Table A.36

Table A.37: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *ChinaSpeed* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *ChinaSpeed*								
c / α	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	36.8344	35.3954	37.9405	37.9116	39.3929	40.7431	41.9931	43.8665
0.04	36.8271	36.5811	37.7863	37.6644	39.0718	40.4660	39.3702	42.5813
0.08	37.1624	36.1625	37.9831	37.5505	39.1066	41.1515	39.3264	43.7070
0.16	37.7054	37.6770	38.8967	39.9385	39.0066	42.3417	38.9969	43.9253
0.32	38.2271	38.6841	38.9338	40.6970	39.4061	43.2879	38.9611	42.5597
0.64	38.8551	40.1537	38.8417	40.9401	39.2520	43.3367	38.9675	44.7035
0.90	38.7043	40.5461	39.0856	41.7393	39.2878	43.3640	39.1228	44.5751

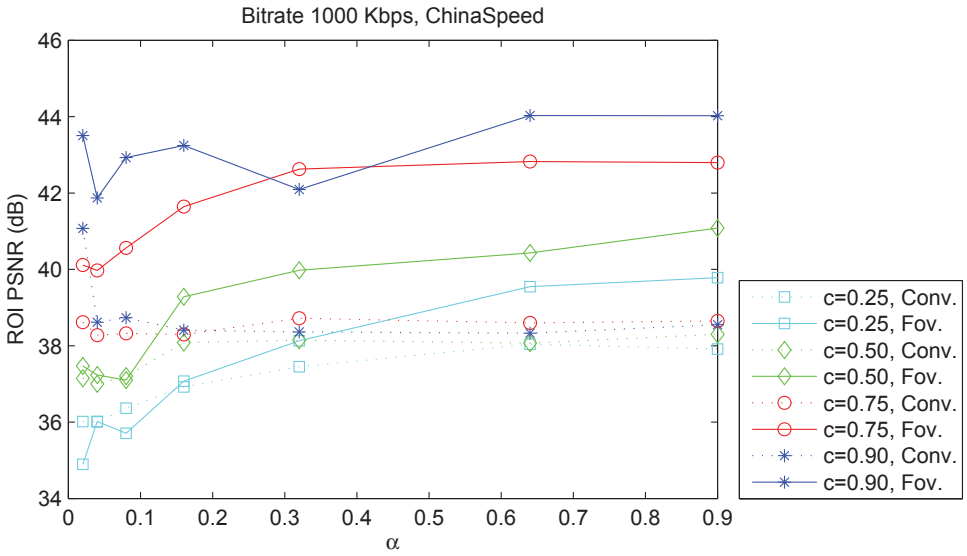


Figure A.37: Chart of Table A.37

Table A.38: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *basketballdrill* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *BasketballDrill*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	27.9895	27.9845	27.7744	27.8433	27.7554	27.8956	32.9429	32.9500
0.04	27.8382	27.8253	27.5397	27.5824	27.1494	27.2942	30.9366	30.9927
0.08	27.6367	27.6342	27.3638	27.3759	27.0111	27.1549	29.6218	29.5630
0.16	27.4719	27.4619	27.2279	27.2490	27.0250	27.1437	29.6811	29.5443
0.32	27.3655	27.3510	27.0932	27.1061	27.2415	27.3430	30.6270	30.8429
0.64	27.2456	27.2606	27.0483	27.0769	27.4312	27.5502	32.6302	33.1191
0.90	27.1623	27.1860	27.0121	27.0454	27.7175	27.8618	34.0275	33.3706

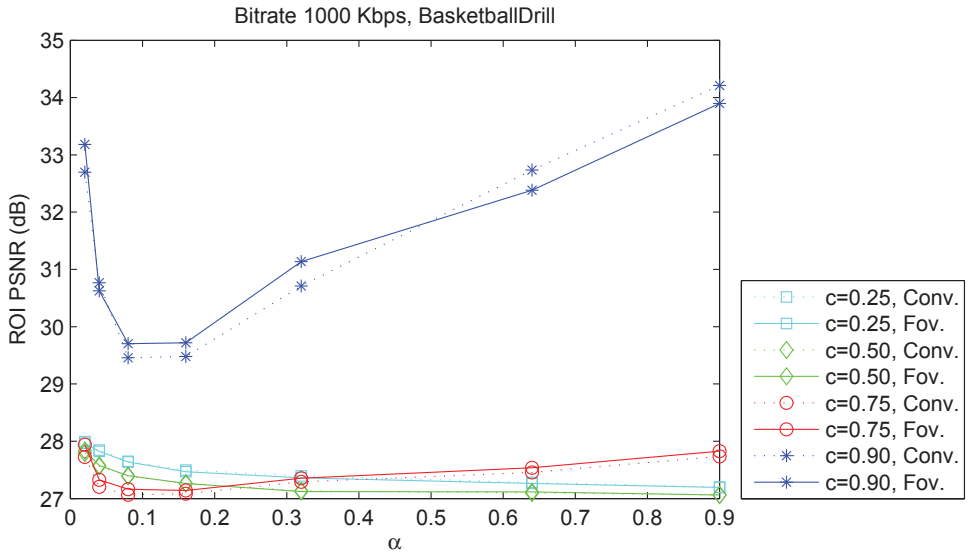


Figure A.38: Chart of Table A.38

Table A.39: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *PartyScene* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *PartyScene*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	33.5678	33.5127	33.7213	33.9282	33.0862	33.5823	30.7093	30.7498
0.04	33.5923	33.8258	33.6851	34.2134	33.3502	34.4037	32.5180	33.7430
0.08	33.6270	34.1686	33.6194	34.5272	33.2946	34.8086	32.2912	34.3199
0.16	33.6536	34.3857	33.5881	34.7007	33.1697	34.9966	32.6208	35.3283
0.32	33.6059	34.4941	33.5223	34.9717	33.0644	35.1588	33.1019	35.9041
0.64	33.5766	34.6681	33.4438	35.0546	33.0336	35.3606	33.5580	35.7666
0.90	33.5932	34.8769	33.3776	35.0984	32.9823	35.3461	32.9910	35.2508

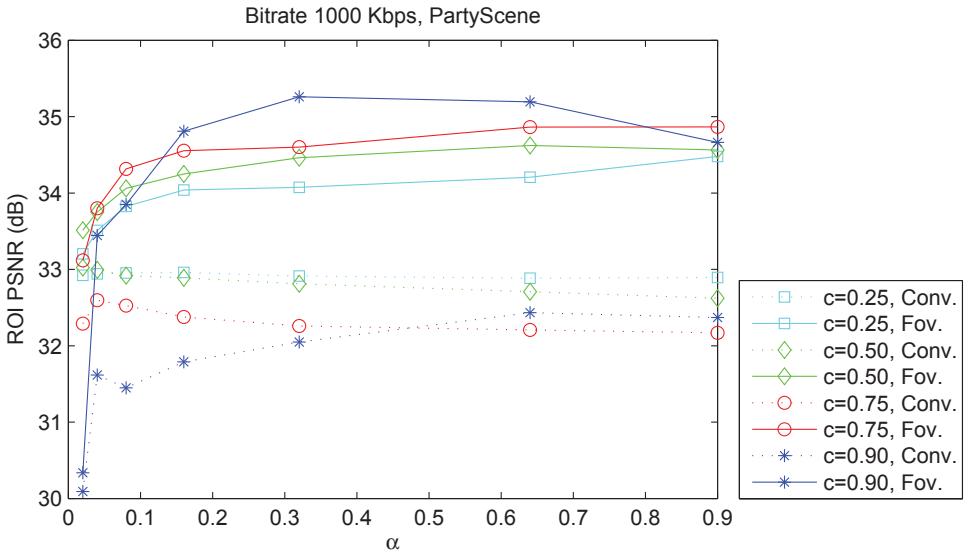


Figure A.39: Chart of Table A.39

Table A.40: Comparison of ROI PSNR between the conventional (Conv.) and the proposed fovea (Fov.) video compression for *RaceHorses* with bit rate = 1000 Kbps.

Bit rate 1000 Kbps, *RaceHorses*								
α \ c	0.25		0.50		0.75		0.90	
	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.	Conv.	Fov.
0.02	40.0973	36.8399	40.6605	37.8088	43.0733	41.6135	49.4929	48.9404
0.04	40.1018	37.0803	40.6177	38.0695	41.4138	39.9285	46.6483	46.8459
0.08	40.2555	37.3869	40.7922	38.5176	41.3081	40.0193	44.3550	43.4297
0.16	40.4653	37.9030	40.8517	38.7454	41.3442	39.6930	43.4194	43.6516
0.32	40.6638	38.4621	40.8188	39.1615	41.3001	40.3451	44.9715	44.2209
0.64	40.8320	38.7576	41.0363	39.5310	41.3609	40.5774	46.9457	45.9189
0.90	40.8981	39.3702	41.1509	39.4274	41.4699	40.8611	47.2911	47.8145

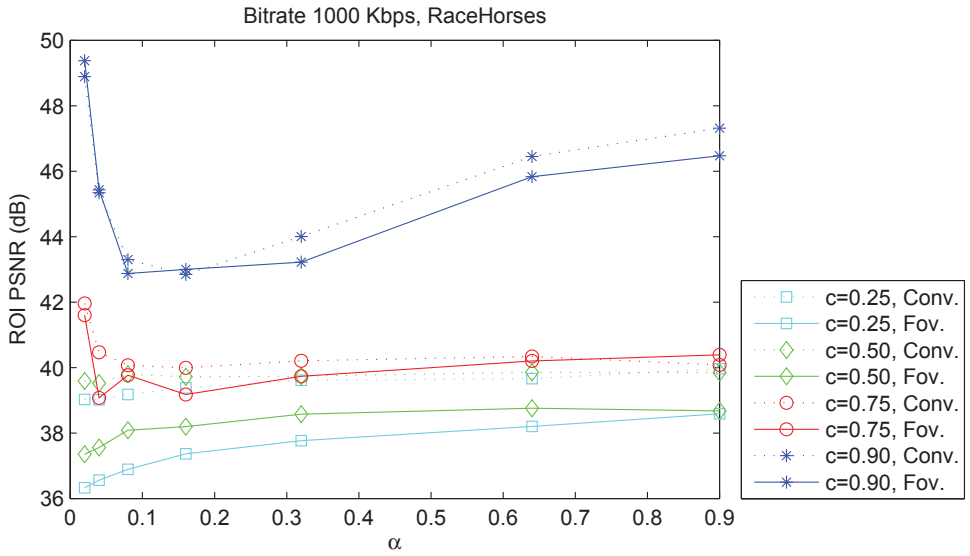


Figure A.40: Chart of Table A.40

Table A.41: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Cactus* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *Cactus*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	29.2810	28.9232	28.4577	28.0661	27.7929	27.5202	27.3816	30.5473
	0.50	29.4138	28.9530	28.4638	28.1495	27.7724	27.5440	27.3925	
	0.75	29.2409	28.7210	28.2920	27.8084	27.4388	27.1245	26.9846	
	0.90	28.2440	27.7039	27.2526	26.9444	26.6348	26.0654	25.8800	
FPSNR	0.25	27.8308	27.6884	27.4389	27.2549	27.0456	26.9016	26.7719	27.3838
	0.50	27.9946	27.7963	27.5436	27.3503	27.1400	26.9702	26.8884	
	0.75	28.0566	27.8533	27.6387	27.3850	27.1148	26.9209	26.8042	
	0.90	27.5633	27.3327	27.1081	26.8518	26.6139	26.3213	26.1670	
FSSIM	0.25	0.8166	0.8126	0.8088	0.8044	0.8018	0.8001	0.7983	0.8346
	0.50	0.8157	0.8110	0.8065	0.8045	0.8011	0.7989	0.7980	
	0.75	0.8117	0.8077	0.8040	0.8007	0.7971	0.7951	0.7941	
	0.90	0.8039	0.7995	0.7962	0.7939	0.7913	0.7876	0.7868	

A.2 FWSNR, FPSNR, and FSSIM Results

Table A.42: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Tennis* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *Tennis*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.8551	10.9275	10.9248	10.9192	10.9699	11.0076	10.9949	10.8064
	0.50	10.8635	10.8702	10.9033	10.9308	10.9441	11.0032	10.9773	
	0.75	10.8607	10.8705	10.9234	10.9455	10.9824	10.9856	11.0156	
	0.90	10.8880	10.8775	10.9290	10.9349	11.0171	11.0090	11.0015	
FPSNR	0.25	20.0090	20.0554	20.0458	20.0576	20.0618	20.0989	20.1021	20.0403
	0.50	20.0067	20.0086	20.0243	20.0391	20.0535	20.0767	20.0800	
	0.75	20.0048	20.0073	20.0264	20.0628	20.0722	20.0874	20.0989	
	0.90	20.0247	20.0335	20.0658	20.0703	20.1166	20.1342	20.1329	
FSSIM	0.25	0.8155	0.8170	0.8179	0.8186	0.8194	0.8206	0.8207	0.8109
	0.50	0.8161	0.8171	0.8183	0.8196	0.8204	0.8212	0.8210	
	0.75	0.8172	0.8184	0.8194	0.8203	0.8210	0.8217	0.8220	
	0.90	0.8185	0.8195	0.8205	0.8214	0.8224	0.8231	0.8232	

Table A.43: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Vidyo4* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *Vidyo4*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	20.8092	20.1019	19.5921	18.9764	18.5033	17.9644	17.8536	24.7125
	0.50	20.6090	20.0258	19.2539	18.7033	18.3151	17.7923	17.5810	
	0.75	19.8222	18.9279	18.3971	17.8428	17.2419	16.8717	16.7875	
	0.90	18.5875	17.8376	17.1175	16.8748	16.3656	16.0931	15.6870	
FPSNR	0.25	34.2836	33.9543	33.6778	33.2102	32.9234	32.4879	32.3676	35.6587
	0.50	34.2291	33.9051	33.5064	33.1380	32.7514	32.3730	32.2281	
	0.75	33.6993	33.2712	32.9529	32.5014	32.1179	31.7693	31.6838	
	0.90	32.7289	32.4091	32.0524	31.7030	31.2543	30.9903	30.7177	
FSSIM	0.25	0.9400	0.9359	0.9326	0.9292	0.9266	0.9233	0.9225	0.9616
	0.50	0.9396	0.9359	0.9322	0.9288	0.9262	0.9235	0.9225	
	0.75	0.9374	0.9333	0.9300	0.9267	0.9235	0.9211	0.9200	
	0.90	0.9310	0.9274	0.9238	0.9208	0.9180	0.9154	0.9144	

Table A.44: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Johnny* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *Johnny*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	22.9225	22.1443	21.5502	20.5170	20.0684	19.4919	19.1894	27.0629
	0.50	22.5456	21.9252	21.1444	20.2986	19.3981	19.1000	18.7163	
	0.75	21.3592	20.5279	19.7001	18.9550	18.1861	17.3803	17.0256	
	0.90	19.2993	18.3150	17.4186	17.0384	15.9324	15.4450	14.9999	
FPSNR	0.25	34.3817	34.0153	33.5529	32.9897	32.5515	32.1526	31.9380	36.5509
	0.50	34.1700	33.8248	33.3354	32.8272	32.2157	31.8648	31.5821	
	0.75	33.2725	32.8927	32.3546	31.8633	31.2780	30.6999	30.4684	
	0.90	31.7588	31.2422	30.7293	30.3393	29.6869	29.1591	28.9155	
FSSIM	0.25	0.9511	0.9473	0.9440	0.9402	0.9371	0.9343	0.9330	0.9685
	0.50	0.9500	0.9462	0.9425	0.9388	0.9351	0.9326	0.9311	
	0.75	0.9455	0.9413	0.9369	0.9338	0.9302	0.9266	0.9249	
	0.90	0.9359	0.9318	0.9281	0.9252	0.9207	0.9174	0.9149	

Table A.45: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *ChinaSpeed* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *ChinaSpeed*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	8.7390	7.8472	6.9772	6.1707	5.4858	4.7846	4.2290	12.7957
	0.50	8.1023	7.1621	6.2182	5.0989	4.3277	3.1598	3.3595	
	0.75	6.8681	5.3073	3.6752	3.1617	2.7564	1.7517	1.5377	
	0.90	3.8362	2.6995	1.5909	1.3561	0.7073	0.2942	0.1730	
FPSNR	0.25	26.7977	26.6542	26.1954	25.9163	25.5613	25.3576	25.1438	29.4732
	0.50	26.5092	26.3054	25.9377	25.4477	25.1300	24.6707	24.7553	
	0.75	25.9939	25.4027	24.9755	24.6504	24.5091	24.1396	23.9402	
	0.90	24.7599	24.4166	23.9484	23.9163	23.6192	23.3266	23.3080	
FSSIM	0.25	0.8685	0.8636	0.8535	0.8470	0.8404	0.8358	0.8341	0.9122
	0.50	0.8636	0.8552	0.8480	0.8390	0.8347	0.8282	0.8280	
	0.75	0.8523	0.8414	0.8357	0.8288	0.8257	0.8186	0.8179	
	0.90	0.8342	0.8283	0.8187	0.8174	0.8114	0.8077	0.8065	

Table A.46: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *BasketballDrill* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *BasketballDrill*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	12.4417	12.2429	11.9840	11.7842	11.5568	11.3877	11.3492	13.5731
	0.50	12.3851	12.1166	11.9221	11.6686	11.4561	11.2712	11.2027	
	0.75	12.1804	11.9260	11.6453	11.4118	11.1755	11.0316	10.9160	
	0.90	11.6737	11.4043	11.1324	10.8309	10.6354	10.4376	10.3433	
FPSNR	0.25	23.0095	22.9197	22.8066	22.7315	22.6422	22.5627	22.5568	23.4919
	0.50	22.9932	22.8734	22.7815	22.6840	22.6030	22.5251	22.4980	
	0.75	22.9505	22.8452	22.7397	22.6416	22.5331	22.4650	22.4331	
	0.90	22.8091	22.6845	22.5702	22.4368	22.3539	22.2345	22.1999	
FSSIM	0.25	0.7662	0.7599	0.7538	0.7491	0.7456	0.7431	0.7419	0.8333
	0.50	0.7640	0.7572	0.7520	0.7468	0.7439	0.7413	0.7405	
	0.75	0.7604	0.7538	0.7482	0.7445	0.7412	0.7399	0.7385	
	0.90	0.7508	0.7460	0.7420	0.7384	0.7362	0.7345	0.7338	

Table A.47: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *PartyScene* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *PartyScene*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.2168	9.5826	8.8667	8.1760	7.6516	7.1382	6.9259	12.2334
	0.50	9.7618	9.0955	8.4183	7.7713	7.1518	6.6753	6.5386	
	0.75	9.0082	8.3496	7.6757	7.0906	6.4982	6.0940	5.8495	
	0.90	7.3769	6.8166	6.2326	5.6448	5.1806	4.8603	4.5049	
FPSNR	0.25	25.5192	25.4078	25.1640	24.8882	24.6096	24.3309	24.2073	25.3605
	0.50	25.2939	25.1562	24.9638	24.6817	24.3687	24.1436	24.0495	
	0.75	24.8717	24.7423	24.5297	24.2886	23.9861	23.7622	23.6066	
	0.90	24.0532	23.9610	23.7560	23.5146	23.2321	22.9719	22.7736	
FSSIM	0.25	0.7101	0.6978	0.6853	0.6753	0.6671	0.6579	0.6553	0.7573
	0.50	0.7019	0.6903	0.6793	0.6697	0.6596	0.6539	0.6503	
	0.75	0.6878	0.6781	0.6679	0.6596	0.6517	0.6465	0.6434	
	0.90	0.6631	0.6578	0.6520	0.6452	0.6399	0.6345	0.6320	

Table A.48: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *RaceHorses* with bit rate = 200 Kbps.

	Bit rate 200 Kbps, *RaceHorses*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	20.3206	19.7380	19.0047	18.3449	17.6169	16.9664	16.7419	23.7336
	0.50	19.8022	19.1398	18.3904	17.6355	16.9111	16.2806	16.0289	
	0.75	18.7566	18.0111	17.2271	16.4361	15.6918	15.0658	14.7914	
	0.90	16.7882	16.0657	15.3055	14.5136	13.8475	13.1954	12.8906	
FPSNR	0.25	28.7608	28.5760	28.2192	27.8539	27.4039	27.0202	26.8461	30.5039
	0.50	28.4085	28.1729	27.8416	27.4148	26.9861	26.5438	26.3624	
	0.75	27.7191	27.4269	27.0808	26.6227	26.1452	25.6924	25.4847	
	0.90	26.3996	26.1485	25.7446	25.2603	24.7889	24.2968	24.0357	
FSSIM	0.25	0.8200	0.8115	0.8011	0.7912	0.7808	0.7723	0.7683	0.8661
	0.50	0.8139	0.8044	0.7942	0.7828	0.7728	0.7639	0.7603	
	0.75	0.8001	0.7898	0.7792	0.7684	0.7584	0.7497	0.7458	
	0.90	0.7736	0.7655	0.7552	0.7449	0.7361	0.7274	0.7229	

Table A.49: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Cactus* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *Cactus*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	30.2375	29.8093	29.4143	29.0086	28.6147	28.2812	28.1306	32.1254
	0.50	30.2991	29.8278	29.3108	28.8967	28.5532	28.2471	28.1146	
	0.75	29.9035	29.3336	28.8161	28.2958	27.8773	27.5583	27.4120	
	0.90	28.6819	28.0874	27.5908	27.2230	26.9051	26.2558	26.1915	
FPSNR	0.25	28.6132	28.4319	28.2386	28.0303	27.7709	27.5878	27.4642	28.6175
	0.50	28.7194	28.5502	28.2925	28.0658	27.8271	27.6025	27.5099	
	0.75	28.5947	28.3793	28.0860	27.8408	27.5347	27.3393	27.2231	
	0.90	27.9443	27.6811	27.4114	27.1588	26.9348	26.5658	26.4835	
FSSIM	0.25	0.8249	0.8205	0.8170	0.8127	0.8091	0.8062	0.8044	0.8464
	0.50	0.8240	0.8195	0.8149	0.8109	0.8072	0.8047	0.8035	
	0.75	0.8189	0.8144	0.8097	0.8052	0.8022	0.7992	0.7976	
	0.90	0.8093	0.8050	0.8006	0.7974	0.7947	0.7905	0.7898	

Table A.50: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Tennis* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *Tennis*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.8217	10.8370	10.8410	10.8679	10.8789	10.8851	10.9071	10.7176
	0.50	10.7989	10.8210	10.8236	10.8615	10.9004	10.8943	10.8991	
	0.75	10.8083	10.8133	10.8484	10.8635	10.8909	10.9342	10.8910	
	0.90	10.8367	10.8244	10.8679	10.8900	10.9015	10.9310	10.9330	
FPSNR	0.25	19.9432	19.9497	19.9538	19.9708	19.9819	19.9944	20.0084	19.9432
	0.50	19.9273	19.9414	19.9479	19.9630	19.9987	20.0020	20.0053	
	0.75	19.9342	19.9396	19.9668	19.9835	19.9964	20.0267	20.0123	
	0.90	19.9700	19.9808	19.9957	20.0227	20.0377	20.0497	20.0641	
FSSIM	0.25	0.8129	0.8141	0.8153	0.8164	0.8170	0.8179	0.8182	0.8070
	0.50	0.8136	0.8148	0.8158	0.8172	0.8180	0.8186	0.8188	
	0.75	0.8144	0.8157	0.8169	0.8177	0.8187	0.8194	0.8197	
	0.90	0.8163	0.8174	0.8185	0.8196	0.8205	0.8213	0.8216	

Table A.51: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Vidyo4* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *Vidyo4*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	21.9072	21.0523	20.4817	19.8659	19.4210	18.7745	18.6733	26.1320
	0.50	21.5177	20.7887	20.0384	19.4130	18.9410	18.4583	18.2291	
	0.75	20.5309	19.7334	18.9904	18.4627	17.7863	17.4354	17.2376	
	0.90	18.9521	18.2907	17.7065	17.1008	16.5770	16.3372	15.9742	
FPSNR	0.25	35.2207	34.8054	34.4664	34.0491	33.7353	33.2891	33.1609	36.8194
	0.50	34.9671	34.6040	34.1458	33.7643	33.4099	33.0229	32.8585	
	0.75	34.3287	33.9704	33.5142	33.1233	32.6953	32.3690	32.1739	
	0.90	33.2123	32.9114	32.5363	32.1091	31.6475	31.3529	31.1261	
FSSIM	0.25	0.9474	0.9434	0.9397	0.9363	0.9333	0.9300	0.9292	0.9673
	0.50	0.9461	0.9419	0.9381	0.9346	0.9319	0.9289	0.9276	
	0.75	0.9428	0.9388	0.9349	0.9313	0.9276	0.9254	0.9241	
	0.90	0.9349	0.9314	0.9275	0.9240	0.9207	0.9183	0.9173	

Table A.52: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Johnny* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *Johnny*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	23.8179	23.0335	22.3637	21.4762	20.7831	20.1658	19.7713	28.5198
	0.50	23.2369	22.5376	21.5872	20.8518	19.8843	19.4411	18.9975	
	0.75	21.9068	20.9102	20.0724	19.3853	18.6425	17.6530	17.5491	
	0.90	19.6061	18.6295	17.6797	17.1647	16.1963	15.5376	15.0571	
FPSNR	0.25	35.1160	34.7770	34.2932	33.7523	33.1621	32.7818	32.5648	37.6398
	0.50	34.7328	34.4329	33.8198	33.3205	32.6853	32.3328	32.0443	
	0.75	33.7178	33.3165	32.7073	32.3031	31.6938	31.0972	30.9050	
	0.90	32.1116	31.6057	31.0062	30.5569	29.9759	29.4332	29.0687	
FSSIM	0.25	0.9560	0.9525	0.9493	0.9451	0.9415	0.9384	0.9372	0.9726
	0.50	0.9545	0.9506	0.9465	0.9429	0.9387	0.9361	0.9343	
	0.75	0.9489	0.9446	0.9403	0.9371	0.9334	0.9295	0.9281	
	0.90	0.9385	0.9343	0.9305	0.9272	0.9226	0.9194	0.9164	

Table A.53: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *ChinaSpeed* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *ChinaSpeed*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	9.3635	8.5471	7.4236	6.6067	5.8833	5.1289	4.5875	14.3470
	0.50	8.5675	7.7621	6.6414	5.4349	4.6255	3.4410	3.6329	
	0.75	7.2487	5.6867	3.9059	3.3685	2.9627	1.9278	1.6343	
	0.90	4.0506	2.9112	1.7169	1.5353	0.8378	0.4397	0.2539	
FPSNR	0.25	27.1272	27.0049	26.4899	26.1725	25.8013	25.5849	25.4007	30.5430
	0.50	26.7360	26.5732	26.1657	25.6441	25.3171	24.8758	24.9323	
	0.75	26.2010	25.5991	25.1334	24.7715	24.6679	24.2734	24.0283	
	0.90	24.8853	24.5437	24.0397	24.0443	23.7292	23.4381	23.3888	
FSSIM	0.25	0.8781	0.8716	0.8608	0.8539	0.8466	0.8429	0.8410	0.9237
	0.50	0.8714	0.8635	0.8550	0.8457	0.8412	0.8346	0.8344	
	0.75	0.8583	0.8472	0.8417	0.8346	0.8323	0.8239	0.8230	
	0.90	0.8401	0.8341	0.8236	0.8228	0.8167	0.8124	0.8109	

Table A.54: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *BasketballDrill* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *BasketballDrill*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	12.7296	12.5268	12.2898	12.1194	11.8918	11.7049	11.6353	13.5827
	0.50	12.5896	12.3710	12.1627	11.9297	11.7168	11.5185	11.4734	
	0.75	12.3730	12.1232	11.8421	11.6356	11.3654	11.1908	11.1049	
	0.90	11.7985	11.5105	11.2332	10.9632	10.6857	10.5035	10.3294	
FPSNR	0.25	23.1123	22.9971	22.9296	22.8462	22.7705	22.7018	22.6797	23.4576
	0.50	23.0630	22.9652	22.8910	22.7922	22.7175	22.6597	22.6141	
	0.75	23.0216	22.9292	22.8142	22.7268	22.6153	22.5502	22.5044	
	0.90	22.8851	22.7494	22.6170	22.5067	22.3868	22.2906	22.2208	
FSSIM	0.25	0.7779	0.7700	0.7629	0.7576	0.7530	0.7490	0.7477	0.8416
	0.50	0.7730	0.7660	0.7598	0.7536	0.7494	0.7460	0.7449	
	0.75	0.7670	0.7600	0.7533	0.7490	0.7450	0.7423	0.7411	
	0.90	0.7553	0.7492	0.7445	0.7401	0.7369	0.7348	0.7337	

Table A.55: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *PartyScene* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *PartyScene*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	11.0755	10.3950	9.6166	8.8665	8.3178	7.7053	7.5471	13.5899
	0.50	10.5468	9.8874	9.1120	8.3332	7.5840	7.1665	7.0861	
	0.75	9.4569	8.7756	8.0734	7.3636	6.8412	6.3293	6.1506	
	0.90	7.6739	7.1221	6.4888	5.8256	5.3184	4.9482	4.6097	
FPSNR	0.25	26.0447	25.9182	25.6496	25.3595	25.0529	24.7503	24.6673	26.2251
	0.50	25.7291	25.6188	25.3622	25.0436	24.7034	24.4589	24.3891	
	0.75	25.1625	25.0514	24.8138	24.5259	24.2601	23.9702	23.8885	
	0.90	24.2820	24.2231	23.9964	23.6965	23.3863	23.1233	22.9266	
FSSIM	0.25	0.7307	0.7180	0.7033	0.6910	0.6803	0.6704	0.6678	0.7842
	0.50	0.7212	0.7080	0.6941	0.6811	0.6706	0.6637	0.6606	
	0.75	0.6996	0.6895	0.6787	0.6691	0.6607	0.6544	0.6520	
	0.90	0.6717	0.6663	0.6596	0.6513	0.6448	0.6391	0.6362	

Table A.56: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *RaceHorses* with bit rate = 300 Kbps.

	Bit rate 300 Kbps, *RaceHorses*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	21.1859	20.5181	19.7400	18.9763	18.2785	17.6012	17.3064	25.3260
	0.50	20.6264	19.8993	19.0856	18.3198	17.5645	16.8089	16.5763	
	0.75	19.3743	18.5790	17.7235	16.8302	16.0179	15.4009	15.0781	
	0.90	17.0924	16.3451	15.5284	14.6979	13.9812	13.2870	12.9715	
FPSNR	0.25	29.3361	29.1330	28.7654	28.3739	27.9243	27.5136	27.3104	31.6158
	0.50	28.9789	28.7041	28.3528	27.9182	27.4682	26.9990	26.8091	
	0.75	28.1368	27.8589	27.4775	26.9630	26.4432	25.9997	25.7754	
	0.90	26.6599	26.3833	25.9740	25.4469	24.9605	24.4294	24.1678	
FSSIM	0.25	0.8371	0.8274	0.8164	0.8058	0.7951	0.7865	0.7816	0.8928
	0.50	0.8301	0.8196	0.8089	0.7975	0.7866	0.7772	0.7735	
	0.75	0.8145	0.8036	0.7922	0.7801	0.7693	0.7602	0.7562	
	0.90	0.7845	0.7753	0.7646	0.7528	0.7433	0.7334	0.7289	

Table A.57: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Cactus* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *Cactus*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	31.6114	31.0703	30.5543	30.0697	29.6112	29.3076	29.1036	34.2012
	0.50	31.4449	30.8391	30.2267	29.7837	29.3456	28.9713	28.7764	
	0.75	30.6859	30.0214	29.3459	28.8219	28.3314	27.9936	27.8039	
	0.90	29.1222	28.4469	27.8904	27.4952	27.1893	26.5317	26.4182	
FPSNR	0.25	29.6769	29.4266	29.1940	28.9294	28.6722	28.4415	28.3088	29.9756
	0.50	29.5962	29.3452	29.0739	28.7875	28.5167	28.2698	28.1516	
	0.75	29.2373	28.9821	28.6220	28.3466	28.0138	27.7590	27.6211	
	0.90	28.3837	28.1070	27.7814	27.5133	27.2784	26.8874	26.7715	
FSSIM	0.25	0.8388	0.8333	0.8290	0.8239	0.8186	0.8166	0.8147	0.8635
	0.50	0.8371	0.8308	0.8247	0.8210	0.8158	0.8122	0.8111	
	0.75	0.8288	0.8233	0.8173	0.8126	0.8083	0.8050	0.8034	
	0.90	0.8160	0.8103	0.8056	0.8020	0.7986	0.7943	0.7936	

Table A.58: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Tennis* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *Tennis*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.7245	10.7567	10.7532	10.7936	10.8028	10.8101	10.8150	10.6501
	0.50	10.7277	10.7464	10.7614	10.7760	10.7941	10.8248	10.8225	
	0.75	10.7289	10.7582	10.7787	10.7895	10.8017	10.8247	10.8146	
	0.90	10.7777	10.7832	10.8023	10.8074	10.8430	10.8436	10.8490	
FPSNR	0.25	19.8508	19.8696	19.8726	19.8995	19.9035	19.9171	19.9162	19.8469
	0.50	19.8612	19.8689	19.8871	19.8920	19.8988	19.9307	19.9255	
	0.75	19.8719	19.8872	19.9051	19.9158	19.9280	19.9406	19.9445	
	0.90	19.9132	19.9242	19.9399	19.9465	19.9695	19.9812	19.9743	
FSSIM	0.25	0.8094	0.8109	0.8119	0.8130	0.8139	0.8145	0.8148	0.8038
	0.50	0.8102	0.8115	0.8127	0.8137	0.8145	0.8155	0.8157	
	0.75	0.8113	0.8127	0.8139	0.8148	0.8158	0.8165	0.8170	
	0.90	0.8136	0.8150	0.8161	0.8172	0.8183	0.8191	0.8194	

Table A.59: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Vidyo4* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *Vidyo4*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	23.1452	22.2339	21.7656	21.0220	20.3139	19.8165	19.5701	27.8691
	0.50	22.6159	21.7979	21.0672	20.3234	19.7224	19.2318	19.0482	
	0.75	21.4805	20.6407	19.9552	19.3245	18.5673	18.0959	17.9823	
	0.90	19.6152	18.9211	18.2216	17.6644	16.9991	16.5065	16.3078	
FPSNR	0.25	36.1536	35.7415	35.4617	35.0149	34.5838	34.1824	34.0480	38.0663
	0.50	35.7992	35.4617	35.0411	34.5735	34.1720	33.7976	33.6563	
	0.75	35.1292	34.7259	34.3546	33.9027	33.4206	33.0187	32.9054	
	0.90	33.9353	33.6106	33.1739	32.7144	32.2031	31.7944	31.5850	
FSSIM	0.25	0.9547	0.9509	0.9475	0.9438	0.9406	0.9375	0.9365	0.9728
	0.50	0.9528	0.9491	0.9451	0.9414	0.9381	0.9351	0.9341	
	0.75	0.9485	0.9445	0.9406	0.9369	0.9330	0.9301	0.9291	
	0.90	0.9392	0.9353	0.9312	0.9275	0.9238	0.9209	0.9197	

Table A.60: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Johnny* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *Johnny*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	24.6776	23.8375	23.1625	22.2529	21.7129	20.8932	20.6360	30.0178
	0.50	24.0787	23.2030	22.3803	21.4925	20.6191	20.0271	19.7234	
	0.75	22.5736	21.7405	20.7077	19.9676	19.1583	18.2746	17.9375	
	0.90	19.9151	18.8958	17.9503	17.3094	16.3065	15.6417	15.0929	
FPSNR	0.25	35.8431	35.4579	34.9889	34.4456	33.9653	33.4714	33.2892	38.7270
	0.50	35.3819	35.0189	34.4667	33.9377	33.3336	32.8967	32.6341	
	0.75	34.3110	33.9337	33.2984	32.8327	32.1835	31.5937	31.3114	
	0.90	32.4609	31.8916	31.3191	30.7868	30.1741	29.5621	29.2220	
FSSIM	0.25	0.9608	0.9576	0.9544	0.9505	0.9475	0.9436	0.9423	0.9764
	0.50	0.9589	0.9554	0.9513	0.9475	0.9434	0.9405	0.9388	
	0.75	0.9530	0.9493	0.9445	0.9411	0.9371	0.9333	0.9315	
	0.90	0.9411	0.9368	0.9328	0.9289	0.9245	0.9210	0.9180	

Table A.61: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *ChinaSpeed* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *ChinaSpeed*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.0850	9.1902	8.1258	7.2174	6.3653	5.5408	4.9868	16.2377
	0.50	9.1688	8.3532	7.1204	5.8462	4.9472	3.7627	3.9290	
	0.75	7.7467	6.0197	4.1661	3.6483	3.1915	2.1249	1.8450	
	0.90	4.3086	3.0984	1.9247	1.6874	0.9620	0.5957	0.4225	
FPSNR	0.25	27.4801	27.3407	26.8069	26.4900	26.0637	25.8283	25.6431	31.8931
	0.50	27.0406	26.8972	26.4163	25.8576	25.5422	25.0928	25.1163	
	0.75	26.4251	25.7977	25.3201	24.9523	24.8329	24.4340	24.1925	
	0.90	25.0514	24.6772	24.1917	24.1824	23.8559	23.5702	23.5190	
FSSIM	0.25	0.8887	0.8816	0.8707	0.8637	0.8558	0.8518	0.8501	0.9376
	0.50	0.8806	0.8729	0.8638	0.8538	0.8498	0.8433	0.8428	
	0.75	0.8668	0.8557	0.8499	0.8428	0.8400	0.8315	0.8303	
	0.90	0.8479	0.8414	0.8303	0.8293	0.8225	0.8181	0.8163	

Table A.62: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *BasketballDrill* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *BasketballDrill*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	12.9304	12.7666	12.5609	12.3917	12.2027	11.9924	11.8996	13.6010
	0.50	12.8176	12.6114	12.4225	12.1894	11.9758	11.7781	11.7143	
	0.75	12.5437	12.2879	11.9972	11.7578	11.5125	11.3011	11.1912	
	0.90	11.8631	11.5655	11.2482	10.9363	10.6732	10.4418	10.3366	
FPSNR	0.25	23.1421	23.0757	23.0013	22.9365	22.8816	22.8198	22.7935	23.4513
	0.50	23.1165	23.0490	22.9712	22.8905	22.8262	22.7600	22.7296	
	0.75	23.0625	22.9896	22.8710	22.7827	22.6957	22.6210	22.5675	
	0.90	22.9215	22.7890	22.6336	22.5122	22.4050	22.2828	22.2363	
FSSIM	0.25	0.7904	0.7820	0.7741	0.7676	0.7618	0.7570	0.7549	0.8544
	0.50	0.7855	0.7767	0.7694	0.7622	0.7569	0.7527	0.7513	
	0.75	0.7762	0.7678	0.7601	0.7544	0.7495	0.7458	0.7442	
	0.90	0.7603	0.7533	0.7471	0.7415	0.7378	0.7348	0.7337	

Table A.63: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *PartyScene* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *PartyScene*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	12.1727	11.3393	10.6238	9.7399	9.1399	8.5025	8.2594	15.5509
	0.50	11.4154	10.6574	9.8117	8.9712	8.3209	7.7762	7.6022	
	0.75	10.1020	9.2886	8.5478	7.8660	7.2099	6.6850	6.5233	
	0.90	8.0611	7.3910	6.7592	6.1147	5.5101	5.0636	4.7913	
FPSNR	0.25	26.7465	26.5530	26.2746	25.9237	25.5956	25.2745	25.1655	27.2984
	0.50	26.3088	26.1403	25.8631	25.5021	25.1893	24.8984	24.8273	
	0.75	25.6087	25.4590	25.2102	24.9277	24.6116	24.3071	24.2160	
	0.90	24.6148	24.4924	24.2582	23.9567	23.6035	23.3043	23.1156	
FSSIM	0.25	0.7565	0.7418	0.7263	0.7119	0.7000	0.6882	0.6847	0.8232
	0.50	0.7417	0.7277	0.7130	0.6993	0.6877	0.6790	0.6759	
	0.75	0.7170	0.7053	0.6932	0.6833	0.6729	0.6652	0.6623	
	0.90	0.6840	0.6763	0.6684	0.6593	0.6516	0.6451	0.6417	

Table A.64: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *RaceHorses* with bit rate = 512 Kbps.

	Bit rate 512 Kbps, *RaceHorses*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	22.3218	21.6110	20.7582	19.8789	19.1351	18.3561	18.0712	27.7867
	0.50	21.5878	20.7865	19.8955	19.0112	18.1688	17.3686	17.0618	
	0.75	20.0654	19.1722	18.2428	17.2520	16.3940	15.6957	15.3452	
	0.90	17.4032	16.5736	15.6725	14.8064	14.0516	13.3343	12.9973	
FPSNR	0.25	30.1125	29.8978	29.4915	29.0387	28.5603	28.1037	27.9193	33.2696
	0.50	29.6290	29.3354	28.9386	28.4738	27.9656	27.4643	27.2512	
	0.75	28.6281	28.3169	27.9090	27.3391	26.8038	26.3049	26.0528	
	0.90	26.9142	26.6175	26.1545	25.6130	25.0853	24.5262	24.2428	
FSSIM	0.25	0.8609	0.8499	0.8381	0.8255	0.8149	0.8048	0.8006	0.9233
	0.50	0.8516	0.8404	0.8286	0.8155	0.8041	0.7938	0.7896	
	0.75	0.8332	0.8214	0.8085	0.7956	0.7836	0.7730	0.7680	
	0.90	0.7972	0.7868	0.7743	0.7615	0.7501	0.7388	0.7334	

Table A.65: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Cactus* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *Cactus*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	32.5023	31.9568	31.3159	30.7704	30.2243	29.7649	29.5690	35.7464
	0.50	32.1602	31.4897	30.7997	30.2617	29.7908	29.4113	29.2031	
	0.75	31.1844	30.4557	29.7604	29.2339	28.6823	28.2742	28.1415	
	0.90	29.3839	28.6460	28.0740	27.6709	27.3286	26.6832	26.5648	
FPSNR	0.25	30.3173	30.0796	29.7765	29.4966	29.1676	28.8895	28.7515	30.8969
	0.50	30.1427	29.8587	29.5651	29.2415	28.9283	28.6726	28.5656	
	0.75	29.6379	29.3421	29.0135	28.7171	28.3656	28.0729	27.9614	
	0.90	28.6640	28.3512	28.0205	27.7427	27.4882	27.1114	26.9703	
FSSIM	0.25	0.8489	0.8436	0.8373	0.8317	0.8267	0.8223	0.8207	0.8773
	0.50	0.8455	0.8387	0.8321	0.8272	0.8219	0.8182	0.8162	
	0.75	0.8364	0.8302	0.8235	0.8187	0.8134	0.8093	0.8078	
	0.90	0.8210	0.8149	0.8095	0.8057	0.8016	0.7975	0.7963	

Table A.66: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Tennis* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *Tennis*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.6874	10.7183	10.7122	10.7382	10.7416	10.7514	10.7613	10.6125
	0.50	10.6889	10.7077	10.7188	10.7228	10.7480	10.7642	10.7585	
	0.75	10.7054	10.7193	10.7302	10.7517	10.7596	10.7886	10.7876	
	0.90	10.7218	10.7421	10.7626	10.7656	10.7853	10.7985	10.8100	
FPSNR	0.25	19.8190	19.8314	19.8360	19.8462	19.8545	19.8710	19.8727	19.8058
	0.50	19.8221	19.8355	19.8452	19.8492	19.8676	19.8793	19.8803	
	0.75	19.8463	19.8506	19.8636	19.8766	19.8882	19.9120	19.9114	
	0.90	19.8623	19.8745	19.8899	19.8985	19.9163	19.9264	19.9355	
FSSIM	0.25	0.8071	0.8084	0.8095	0.8106	0.8116	0.8122	0.8125	0.8013
	0.50	0.8077	0.8090	0.8102	0.8113	0.8124	0.8132	0.8135	
	0.75	0.8093	0.8104	0.8117	0.8128	0.8138	0.8148	0.8153	
	0.90	0.8117	0.8131	0.8144	0.8155	0.8166	0.8175	0.8179	

Table A.67: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Vidyo4* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *Vidyo4*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	23.9915	23.0794	22.2972	21.6392	20.9842	20.4132	20.0587	29.0550
	0.50	23.3871	22.5560	21.7051	20.9358	20.3458	19.6350	19.5442	
	0.75	22.0753	21.2543	20.3811	19.7937	19.0506	18.5207	18.2571	
	0.90	19.9319	19.1603	18.4885	17.8738	17.2069	16.7084	16.5597	
FPSNR	0.25	36.8122	36.4262	35.9644	35.5373	35.1228	34.7206	34.5305	38.9587
	0.50	36.4699	36.0871	35.6339	35.1379	34.7104	34.2702	34.1709	
	0.75	35.7166	35.3402	34.8515	34.4261	33.9219	33.5067	33.3296	
	0.90	34.2884	33.9142	33.5005	33.0346	32.5024	32.0918	31.9328	
FSSIM	0.25	0.9592	0.9556	0.9520	0.9484	0.9453	0.9425	0.9411	0.9761
	0.50	0.9569	0.9532	0.9492	0.9457	0.9421	0.9390	0.9380	
	0.75	0.9519	0.9480	0.9438	0.9400	0.9361	0.9330	0.9317	
	0.90	0.9417	0.9377	0.9335	0.9295	0.9257	0.9225	0.9214	

Table A.68: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Johnny* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *Johnny*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	25.3538	24.4914	23.7410	22.7285	22.0780	21.3932	21.0693	31.0593
	0.50	24.7749	23.8803	23.0268	22.0990	21.0471	20.4887	20.1028	
	0.75	22.9581	22.0290	20.9830	20.2871	19.3903	18.4536	18.0843	
	0.90	20.0676	19.1259	18.0505	17.4778	16.3986	15.6945	15.2027	
FPSNR	0.25	36.3718	35.9591	35.4784	34.8715	34.3575	33.9023	33.6875	39.4635
	0.50	35.9130	35.5082	34.9484	34.4114	33.7226	33.2845	33.0177	
	0.75	34.6441	34.1835	33.5833	33.1143	32.4507	31.8144	31.5279	
	0.90	32.6610	32.1346	31.4943	30.9336	30.3202	29.6826	29.3592	
FSSIM	0.25	0.9641	0.9609	0.9578	0.9539	0.9506	0.9472	0.9458	0.9786
	0.50	0.9622	0.9587	0.9547	0.9508	0.9466	0.9435	0.9417	
	0.75	0.9555	0.9514	0.9467	0.9432	0.9391	0.9352	0.9331	
	0.90	0.9425	0.9385	0.9341	0.9304	0.9257	0.9221	0.9190	

Table A.69: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *ChinaSpeed* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *ChinaSpeed*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.5473	9.6653	8.5080	7.5279	6.6229	5.8138	5.2739	17.8005
	0.50	9.5631	8.6840	7.4715	6.0781	5.2059	3.9631	4.1069	
	0.75	8.0623	6.2423	4.3608	3.8168	3.3601	2.2507	1.9810	
	0.90	4.4568	3.1921	2.0221	1.7731	1.0490	0.6803	0.4597	
FPSNR	0.25	27.7097	27.5965	27.0053	26.6629	26.2207	26.0056	25.8280	33.2895
	0.50	27.2376	27.0736	26.5889	26.0150	25.7002	25.2159	25.2332	
	0.75	26.5902	25.9272	25.4509	25.0645	24.9456	24.5284	24.2892	
	0.90	25.1400	24.7463	24.2697	24.2515	23.9226	23.6338	23.5635	
FSSIM	0.25	0.8953	0.8890	0.8779	0.8707	0.8623	0.8586	0.8569	0.9484
	0.50	0.8871	0.8794	0.8703	0.8602	0.8567	0.8498	0.8493	
	0.75	0.8734	0.8625	0.8562	0.8485	0.8456	0.8368	0.8353	
	0.90	0.8531	0.8462	0.8348	0.8334	0.8261	0.8216	0.8194	

Table A.70: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *BasketballDrill* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *BasketballDrill*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	13.0058	12.8471	12.6842	12.5091	12.3448	12.1588	12.0495	13.5980
	0.50	12.9252	12.7197	12.5323	12.2962	12.0882	11.8547	11.7775	
	0.75	12.5971	12.3331	12.0383	11.7887	11.5293	11.3035	11.1848	
	0.90	11.8656	11.5388	11.2115	10.9245	10.6512	10.4059	10.2905	
FPSNR	0.25	23.1513	23.0861	23.0345	22.9695	22.9213	22.8685	22.8426	23.4474
	0.50	23.1475	23.0660	23.0024	22.9213	22.8606	22.7914	22.7625	
	0.75	23.0621	22.9989	22.8814	22.7869	22.6903	22.6176	22.5696	
	0.90	22.9160	22.7610	22.6237	22.4966	22.3755	22.2527	22.1916	
FSSIM	0.25	0.7990	0.7902	0.7824	0.7751	0.7692	0.7636	0.7612	0.8622
	0.50	0.7933	0.7845	0.7767	0.7689	0.7627	0.7575	0.7557	
	0.75	0.7823	0.7735	0.7648	0.7583	0.7527	0.7484	0.7465	
	0.90	0.7634	0.7554	0.7489	0.7432	0.7386	0.7347	0.7335	

Table A.71: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *PartyScene* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *PartyScene*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	12.8705	12.0686	11.2367	10.3455	9.5685	8.9769	8.7437	16.9844
	0.50	12.0180	11.1761	10.3020	9.4199	8.6745	8.1872	7.9650	
	0.75	10.4700	9.6783	8.8214	8.1279	7.4506	6.8825	6.6198	
	0.90	8.3386	7.5832	6.9045	6.1546	5.5740	5.1117	4.7811	
FPSNR	0.25	27.1636	26.9483	26.6601	26.2904	25.9143	25.6010	25.4916	28.2783
	0.50	26.6893	26.4678	26.1776	25.8308	25.4651	25.2003	25.1050	
	0.75	25.9097	25.7273	25.4450	25.1719	24.8111	24.4910	24.3534	
	0.90	24.8344	24.6745	24.4068	24.0975	23.7288	23.4312	23.2198	
FSSIM	0.25	0.7737	0.7583	0.7427	0.7270	0.7137	0.7018	0.6978	0.8477
	0.50	0.7570	0.7419	0.7266	0.7124	0.6996	0.6906	0.6868	
	0.75	0.7295	0.7167	0.7034	0.6927	0.6814	0.6728	0.6690	
	0.90	0.6930	0.6841	0.6747	0.6647	0.6564	0.6490	0.6455	

Table A.72: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *RaceHorses* with bit rate = 768 Kbps.

	Bit rate 768 Kbps, *RaceHorses*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	23.1038	22.2919	21.3868	20.4310	19.6102	18.8047	18.4524	30.0247
	0.50	22.3056	21.4228	20.4329	19.4983	18.5652	17.6969	17.3733	
	0.75	20.4630	19.4965	18.5029	17.4644	16.5612	15.8124	15.4179	
	0.90	17.5262	16.6474	15.7133	14.8225	14.0480	13.3113	12.9826	
FPSNR	0.25	30.6541	30.4073	29.9684	29.4624	28.9552	28.4703	28.2568	34.7668
	0.50	30.1061	29.7742	29.3520	28.8597	28.3108	27.7607	27.5375	
	0.75	28.9009	28.5864	28.1517	27.5527	26.9765	26.4493	26.1720	
	0.90	27.0292	26.7029	26.2244	25.6575	25.1141	24.5403	24.2595	
FSSIM	0.25	0.8765	0.8657	0.8532	0.8402	0.8286	0.8183	0.8137	0.9428
	0.50	0.8676	0.8556	0.8428	0.8294	0.8170	0.8062	0.8013	
	0.75	0.8456	0.8332	0.8197	0.8054	0.7921	0.7804	0.7746	
	0.90	0.8031	0.7917	0.7783	0.7644	0.7522	0.7399	0.7344	

Table A.73: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Cactus* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *Cactus*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	33.0831	32.3531	31.7148	31.1478	30.5989	30.1409	29.9629	36.6339
	0.50	32.5719	31.8418	31.1907	30.6158	30.0960	29.6866	29.4681	
	0.75	31.4371	30.6867	30.0141	29.4246	28.8729	28.4372	28.2914	
	0.90	29.5398	28.8478	28.2437	27.8072	27.4452	26.7754	26.6461	
FPSNR	0.25	30.6984	30.3839	30.1025	29.7763	29.4873	29.1794	29.0555	31.3625
	0.50	30.4537	30.1526	29.8363	29.5116	29.2023	28.9263	28.7889	
	0.75	29.8718	29.5688	29.2454	28.9257	28.5662	28.2569	28.1529	
	0.90	28.8769	28.5938	28.2162	27.9117	27.6469	27.2375	27.0946	
FSSIM	0.25	0.8555	0.8488	0.8424	0.8367	0.8312	0.8269	0.8251	0.8851
	0.50	0.8507	0.8435	0.8375	0.8321	0.8260	0.8222	0.8201	
	0.75	0.8409	0.8343	0.8275	0.8223	0.8169	0.8125	0.8107	
	0.90	0.8239	0.8182	0.8125	0.8081	0.8040	0.7996	0.7982	

Table A.74: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Tennis* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *Tennis*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.6659	10.6780	10.6890	10.7210	10.7245	10.7258	10.7219	10.5964
	0.50	10.6588	10.6840	10.6914	10.7085	10.7196	10.7396	10.7324	
	0.75	10.6832	10.6880	10.7188	10.7366	10.7205	10.7525	10.7447	
	0.90	10.7072	10.7086	10.7383	10.7329	10.7630	10.7522	10.7555	
FPSNR	0.25	19.7948	19.8007	19.8131	19.8340	19.8371	19.8440	19.8494	19.7782
	0.50	19.8034	19.8109	19.8207	19.8274	19.8443	19.8583	19.8582	
	0.75	19.8175	19.8265	19.8457	19.8549	19.8570	19.8758	19.8741	
	0.90	19.8449	19.8461	19.8630	19.8691	19.8862	19.8811	19.8877	
FSSIM	0.25	0.8056	0.8068	0.8082	0.8093	0.8102	0.8108	0.8111	0.7997
	0.50	0.8062	0.8075	0.8087	0.8099	0.8109	0.8119	0.8122	
	0.75	0.8079	0.8092	0.8105	0.8115	0.8127	0.8137	0.8140	
	0.90	0.8105	0.8119	0.8134	0.8143	0.8155	0.8162	0.8167	

Table A.75: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Vidyo4* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *Vidyo4*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	24.5220	23.5737	22.7291	22.0498	21.2737	20.6991	20.4948	29.8365
	0.50	23.7782	22.9729	22.0305	21.3109	20.6748	19.9836	19.8455	
	0.75	22.2763	21.3802	20.5812	19.9571	19.1656	18.7200	18.4334	
	0.90	20.1110	19.4257	18.6421	18.0424	17.3801	16.8775	16.6882	
FPSNR	0.25	37.2208	36.8364	36.3781	35.9196	35.4498	35.0444	34.9146	39.4885
	0.50	36.8037	36.4366	35.9431	35.5017	35.0593	34.6064	34.4831	
	0.75	35.9431	35.5165	35.0619	34.5860	34.1039	33.7150	33.5156	
	0.90	34.5276	34.1824	33.7458	33.2418	32.7469	32.2798	32.1357	
FSSIM	0.25	0.9616	0.9582	0.9546	0.9511	0.9478	0.9447	0.9437	0.9779
	0.50	0.9591	0.9555	0.9517	0.9481	0.9448	0.9413	0.9402	
	0.75	0.9537	0.9497	0.9455	0.9416	0.9377	0.9346	0.9330	
	0.90	0.9430	0.9391	0.9348	0.9307	0.9267	0.9234	0.9224	

Table A.76: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *Johnny* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *Johnny*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	25.7755	24.9150	24.0459	23.0675	22.4465	21.5699	21.3028	31.6722
	0.50	25.0459	24.1466	23.2654	22.3045	21.3534	20.7137	20.2578	
	0.75	23.1533	22.2112	21.0690	20.3788	19.4229	18.5701	18.1415	
	0.90	20.1722	19.1774	18.1366	17.5141	16.4209	15.7129	15.2156	
FPSNR	0.25	36.7065	36.2936	35.7743	35.1387	34.6358	34.1319	33.9177	39.9163
	0.50	36.1592	35.7475	35.1828	34.6054	33.9882	33.5193	33.2009	
	0.75	34.8290	34.3704	33.7229	33.2071	32.5305	31.9218	31.6263	
	0.90	32.7738	32.2118	31.5969	31.0098	30.3800	29.7379	29.3882	
FSSIM	0.25	0.9660	0.9630	0.9598	0.9558	0.9526	0.9488	0.9476	0.9797
	0.50	0.9639	0.9604	0.9566	0.9526	0.9484	0.9453	0.9434	
	0.75	0.9568	0.9527	0.9479	0.9443	0.9401	0.9362	0.9341	
	0.90	0.9434	0.9391	0.9349	0.9311	0.9261	0.9225	0.9194	

Table A.77: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *ChinaSpeed* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *ChinaSpeed*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	10.7841	9.8842	8.7439	7.7109	6.7864	5.9776	5.4117	18.8494
	0.50	9.8327	8.8863	7.6864	6.2549	5.3676	4.0365	4.2383	
	0.75	8.2100	6.3757	4.4490	3.8664	3.4420	2.2829	2.0665	
	0.90	4.5288	3.2476	2.0218	1.7939	1.0695	0.7061	0.5214	
FPSNR	0.25	27.8452	27.7256	27.1263	26.7666	26.3136	26.1057	25.9212	34.0951
	0.50	27.3588	27.1815	26.6887	26.1177	25.8056	25.2729	25.3101	
	0.75	26.6606	26.0044	25.5101	25.1103	25.0045	24.5719	24.3367	
	0.90	25.1753	24.7728	24.2848	24.2800	23.9453	23.6538	23.5978	
FSSIM	0.25	0.9000	0.8938	0.8827	0.8752	0.8669	0.8631	0.8611	0.9541
	0.50	0.8921	0.8839	0.8751	0.8649	0.8610	0.8538	0.8532	
	0.75	0.8773	0.8665	0.8601	0.8520	0.8490	0.8397	0.8383	
	0.90	0.8558	0.8490	0.8373	0.8358	0.8283	0.8233	0.8213	

Table A.78: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *BasketballDrill* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *BasketballDrill*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	18.6686	18.5196	12.7341	12.5771	12.4073	12.1708	12.1125	13.5965
	0.50	19.3710	19.1616	18.9768	12.3357	12.1009	11.8902	11.8218	
	0.75	21.9988	21.7217	12.0504	11.7913	11.4911	11.2842	11.1631	
	0.90	25.9833	25.6380	11.1819	10.8837	10.6040	10.3767	10.2357	
FPSNR	0.25	24.7012	24.5855	23.0402	22.9824	22.9284	22.8677	22.8506	23.4404
	0.50	24.8951	24.7561	24.6165	22.9300	22.8595	22.7971	22.7774	
	0.75	23.6646	23.4288	22.8805	22.7882	22.6819	22.6064	22.5562	
	0.90	22.0751	21.7675	22.5960	22.4661	22.3457	22.2187	22.1613	
FSSIM	0.25	0.8047	0.7957	0.7871	0.7800	0.7734	0.7669	0.7649	0.8674
	0.50	0.7987	0.7891	0.7812	0.7728	0.7659	0.7606	0.7585	
	0.75	0.7859	0.7766	0.7675	0.7603	0.7539	0.7494	0.7472	
	0.90	0.7650	0.7565	0.7497	0.7436	0.7386	0.7349	0.7334	

Table A.79: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *PartyScene* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *PartyScene*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	13.3182	12.4439	11.5534	10.6838	9.8535	9.2074	8.9493	17.9502
	0.50	12.4549	11.5662	10.6628	9.7426	8.9347	8.3722	8.1484	
	0.75	10.7351	9.8639	8.9519	8.2733	7.5601	6.9787	6.7270	
	0.90	8.4635	7.7484	6.9730	6.2368	5.6254	5.1285	4.8199	
FPSNR	0.25	27.4153	27.2047	26.8663	26.5046	26.1150	25.7948	25.6618	28.9100
	0.50	26.9614	26.7465	26.4360	26.0545	25.6830	25.3832	25.2682	
	0.75	26.1032	25.9131	25.5829	25.2883	24.9447	24.6074	24.4656	
	0.90	24.9652	24.8032	24.5004	24.1837	23.8038	23.4607	23.2590	
FSSIM	0.25	0.7839	0.7688	0.7521	0.7361	0.7221	0.7102	0.7057	0.8618
	0.50	0.7679	0.7525	0.7362	0.7214	0.7077	0.6983	0.6939	
	0.75	0.7376	0.7240	0.7100	0.6984	0.6870	0.6778	0.6737	
	0.90	0.6987	0.6896	0.6795	0.6685	0.6596	0.6513	0.6474	

Table A.80: Comparison of FWSNR, FPSNR, and FSSIM between proposed fovea method and conventional compression for *RaceHorses* with bit rate = 1000 Kbps.

	Bit rate 1000 Kbps, *RaceHorses*								
	c \ α	0.02	0.04	0.08	0.16	0.32	0.64	0.90	Conv.
FWSNR	0.25	23.5874	22.7712	21.7590	20.7759	19.8982	19.0316	18.6568	31.5768
	0.50	22.6885	21.7367	20.7031	19.7181	18.7534	17.8111	17.4629	
	0.75	20.6785	19.6601	18.6314	17.5705	16.6087	15.8357	15.4392	
	0.90	17.5532	16.6547	15.7037	14.8081	14.0257	13.2936	12.9677	
FPSNR	0.25	31.0128	30.7525	30.2608	29.7484	29.1990	28.6778	28.4401	35.8491
	0.50	30.3788	30.0323	29.5759	29.0659	28.4777	27.8931	27.6452	
	0.75	29.0716	28.7257	28.2752	27.6533	27.0503	26.4984	26.2153	
	0.90	27.0487	26.7219	26.2338	25.6640	25.1108	24.5326	24.2498	
FSSIM	0.25	0.8865	0.8757	0.8628	0.8498	0.8378	0.8269	0.8219	0.9533
	0.50	0.8767	0.8646	0.8513	0.8376	0.8246	0.8125	0.8072	
	0.75	0.8523	0.8394	0.8251	0.8104	0.7961	0.7834	0.7769	
	0.90	0.8045	0.7929	0.7790	0.7650	0.7523	0.7396	0.7341	