USER ADAPTIVE TRANSCODING FOR VIDEO TELECONFERENCING

Liangping Ma, Dharm Veer, Wei Chen, Gregory Sternberg, Yuriy A. Reznik, and Ralph A. Neff

InterDigital Communications, Inc., San Diego, CA 92121

ABSTRACT

The human visual system (HVS) cannot perceive spatial frequency components in an image that are above a certain limit, the value of which is affected by factors such as the viewing distance. This has been exploited to improve the video coding efficiency by first filtering out redundant frequency components and then doing conventional video encoding. To facilitate the deployment of this scheme, we propose to implement it in a network entity such as a Multipoint Control Unit (MCU). Specifically, by analyzing the video sent from a client, the MCU infers that client's viewing conditions, which are then used to adapt the encoding of the video destined to that client. The scheme is implemented in a real-world video teleconferencing system. Experimental results show that our approach can result in significant savings in bandwidth without affecting subjective video quality.

Index Terms— viewing conditions, perceptual pre-filter, human vision, contrast sensitivity, user-adaptive video coding, video teleconferencing, MCU.

1. INTRODUCTION

The human visual system (HVS) cannot perceive spatial frequencies in an image that are above a certain limit (or cutoff frequency), which is influenced by viewing conditions such as the viewing distance, ambient luminance and display characteristics. Frequency components above the limit can be removed to reduce the information in an image before the conventional video compression is applied, thereby improving the efficiency of image/video compression.

We use the contrast sensitivity function (CSF) model [1][2][3] to characterize this phenomenon. This model establishes a relationship between the *spatial frequency* (in cycles per degree or cpd) and the contrast sensitivity. The spatial frequency characterizes the oscillation of a sinusoid in an image with respect to the angular span of the sinusoid to the eyes. For a fixed sinusoid, as the viewing distance increases, the angular span decreases and hence the spatial frequency increases. The contrast sensitivity is the inverse of the Michelson contrast, and the higher it is, the lower the contrast is. The contrast sensitivity is determined by the contrast of the image itself, and the viewing conditions such as the ambient luminance and the reflection of the display. Therefore, as the viewing condition changes, the frequency components in an image that are visible to the HVS also changes. Additionally, since the contrast sensitivity may vary significantly from region to region in an image, applying different cutoff frequencies instead of the same one to different regions of an image to filter the image prior to conventional video encoding may dramatically improve the video compression efficiency, as shown in our previous work [4][5]. The filtering is referred to as *perceptual pre-filtering*, and the whole process (filtering together with the conventional video encoding) is referred to as *user adaptive video coding*.

Perceptual pre-filtering has been used to improve the video coding efficiency of streaming video [4][5][6]. It has not been applied to video teleconferencing, which is the focus of our present work. Perceptual pre-filtering can reside either at the client or in the network. To obtain the viewing condition, the client approach involves either the use of various sensors on a mobile device, which may cause privacy concerns, or the installation of sophisticated software on the client devices for face detection, which may be time-consuming or inconvenient. We take a network-based approach, and in particular focus on the Multi-point Control Unit (MCU). The MCU analyzes the incoming video streams to infer the viewing condition in realtime and then performs user adaptive video encoding on the outgoing video streams. Our approach does not require any changes to the clients, and is thus easier to deploy.

The remainder of the paper is organized as follows. We discuss related work in Section 2, present our approach in Section 3, give the implementation details in Section 4, and evaluate the performance in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

In the study of visual perception, spatial frequency is generally measured in cycles per degree. Specifically, let the length of one cycle of the sinusoid be n pixels, the pixel density of the display be ρ pixels per inch, and the viewing distance be a. Then, the spatial frequency is defined as

$$f = \frac{1}{2\arctan(n/(2\rho a))}. \quad (1)$$

In the perceptual pre-filtering process, we eventually need a frequency in cycles per *pixel* rather than cycles per *degree*. The desired frequency is nothing but $1/n$. Thus, the conversion is: f (cycles per degree) $\leftrightarrow$ $1/n$ (cycles per pixel). The contrast sensitivity is defined as

$$C = \frac{I_{\max} + I_{\min}}{I_{\max} - I_{\min}}, \tag{2}$$

where $I_{\max}$ and $I_{\min}$ are the maximum and minimum intensities of a visual signal (e.g., sinusoid) perceived by the human eye, respectively. $I_{\max}$ and $I_{\min}$ depend not only on the luminance value of the sinusoid in an image, but also on the viewing conditions such as the ambient luminance and the reflection property of the display.

The CSF model [1][2][3] characterizes the visibility of a sinusoid as a function of spatial frequency and contrast sensitivity. At a given spatial frequency, there is a contrast sensitivity above which the sinusoid becomes invisible. Pairs of such spatial frequency and the contrast sensitivity form a function, which is called the CSF function, and the function is concave with a peak at a moderate spatial frequency. To get the cutoff frequency, we need to solve for the spatial frequency given the contrast sensitivity, i.e., we need to get the inverse CSF function. Since the CSF function is not monotonic, a monotonic portion of the CSF function is used to obtain the inverse function. Because different regions of an image may have different contrast sensitivity, the cutoff frequency is calculated on a per-region basis. The cutoff frequencies in cpd are then converted to cycles per pixels to low-pass filter different regions of the image before the conventional video encoding is applied. With perceptual pre-filtering, it is reported in [4] that the x264 video encoder can achieve up to 70% improvement in compression efficiency under certain viewing conditions.

Our present work differs from previous studies [7][4][5][6] in several aspects. First, all prior schemes require explicit feedback for communicating the viewing condition from the user to the video encoder or server. In our scheme, the network entity (MCU) estimates the viewing condition by analyzing the passing video in the opposite direction, which is feasible due to the bidirectional nature of video teleconferencing traffic, and thus eliminates the need for such feedback. Second, these prior studies focus on video streaming, whereas our scheme focuses on video teleconferencing, which differs significantly from video streaming in quality of service requirements and the underlying transport protocols. Third, in the prior studies the perceptual pre-filtering is applied to the uncompressed video, while in our scheme it is applied to a video which has undergone compression once at a client. Since the original compression process has not been adapted to viewing conditions, the proposed filtering and transcoding process still leads to bit rate savings. Fourth, unlike prior studies which focus only on video codecs, we design and prototype the complete system based on the latest video teleconferencing platform WebRTC [8] and a real-world MCU platform Licode [9].

3. OUR APPROACH

We first describe the overall architecture of our proposed system, and then discuss specific implementation techniques.

3.1. System Architecture

The system architecture is shown in Fig. 1, where two clients are shown to communicate via video teleconferencing with the assistance of an MCU which resides in the Internet. Each client implements WebRTC [8], an open-source real-time video communication application. The MCU runs Licode [9], an open-source platform that implements some basic functions of a typical MCU. Although Licode manages the call setup, for delivering the video (and audio) content, it serves the function of a router only, without any video (or audio) processing capability. To enable user adaptive video coding in the MCU, we implement additional functions including video decoding, video encoding, video analysis, viewing condition inference, and perceptual pre-filtering. The implementation of WebRTC on the clients is not changed.

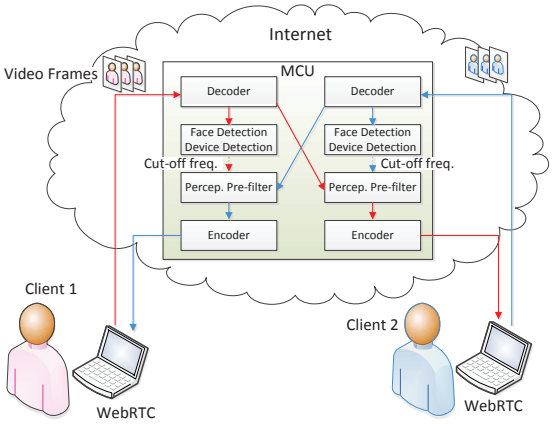

Fig. 1. System architecture with two clients and an MCU.

Both clients benefit from user adaptive video coding. For clarity, we describe the process that leads to benefiting Client 1. The MCU first analyzes the video frames (i.e., does face detection on the video frames) sent from Client 1 to Client 2 to infer the viewing distance of Client 1. The MCU also extracts the control signaling (i.e., does device detection) from the data sent from Client 1 to determine the pixel density of the display watched by Client 1. The MCU then determines the contrast sensitivity of each region of a video frame of the video flow in the opposite direction sent by Client 2 and consequently the cutoff frequencies and performs perceptual pre-filtering, followed by conventional video encoding. In theory, the conventional video encoder can use the same quantization

parameter configuration as the one used to encode the video arriving at the MCU. In practice, a target video bit rate can be first calculated via a heuristic formula that maps the incoming video bit rate and the viewing conditions to a target video bit rate for video teleconferencing type of content and then passed to the conventional video encoder. Lastly, the MCU sends the encoded video frame to Client 1.

3.2. Inferring the Viewing Distance

A well known approach to estimating the viewing distance is to estimate the depth information, which can be done by using dedicated sensors such as infra-red and ultrasonic sensors. Exemplary schemes include Microsoft Kinect [10] and those in [11][12]. However, the existence of such dedicated sensors on the clients may not be guaranteed in practice. In addition, these sensors may collect information other than the viewing distance, raising concerns about privacy. Therefore, this approach may not be the best to the wide deployment of user adaptive video coding.

A less known but more attractive approach is to analyze an image without using any custom sensor, as explained in Fig. 2 with a top-down view of the setup. The viewing distance, i.e., the distance between the eyes and the lens, is denoted as s_1. We use the face detection capability of the open source computer vision (OpenCV) [13] library to detect the face and identify the pupils on an image, and then measure the inter-pupil distance d on the image in pixels. The field of view (FOV) or viewing angle of the camera is β. The distance between the lens and the image sensor is s_2. The width of the image is w (in pixels). The real-world inter-pupil distance is D. The viewing angle of the eyes is α. It is easy to get the viewing distance

$$s_1 = \frac{D}{2\tan(\alpha/2)}, \tag{3}$$

where

$$\alpha = 2\arctan\left(\tan\left(\frac{\beta}{2}\right)\frac{d}{w}\right). \tag{4}$$

It is shown that the value of D for most adults varies in the range from 50 to 75mm [14]. A population average 63mm is used in our estimation. The accuracy will improve if D can be calibrated.

A similar but less accurate method is proposed in [15]. The only difference is that the approximations $\tan(x) \approx x$ and $\arctan(x) \approx x$ are used in [15]. As a result, instead of having (4), which is exact, [15] has $\alpha = \beta d/w$. The approximation is inaccurate for large β.

3.3. Determining the pixel density

Without explicit signaling from the clients to the MCU to inform the latter of the pixel density of the display, the MCU has to extract such information from the control messages sent by the clients. The control message we exploit is the HTTP request messages, where the UserAgent field often contains the device information and the operating system information. As an example, Android devices provide detailed information about the device type, and the MCU can look up a device table to find the pixel density and the reflectivity of the display on that device. The device table lists the pixel density and the reflectivity of the display for all major devices and is updated when a new device becomes available in the market.

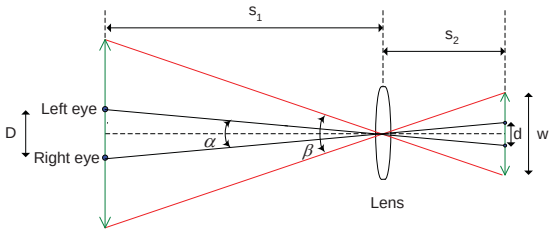

Fig. 2. The calculation of the viewing angle.

3.4. User-adaptive video encoding

Once the viewing distance and the pixel density are determined, we can follow the procedures in [4][5] to perform user-adaptive video encoding. Specifically, the contrast sensitivity is determined for each location in a video frame. The contrast sensitivity takes into account factors such as the contrast of each location in the video frame, the ambient luminance, and the reflectivity of the display. Then a cutoff frequency (in cycles per degree) is determined for each location. The viewing distance and the display pixel density are then used to convert the cutoff frequency from cycles per degree to cycles per pixel. Next, these cutoff frequencies are used to low-pass filter different locations of the video frame, and the output is passed to a conventional video encoder. Finally, the MCU transmits the encoded video frame to the client expecting it.

4. IMPLEMENTATION

Our scheme is implemented in Licode (version 0.1.0) [9], an open source MCU platform designed for WebRTC. Licode sets up video teleconferencing sessions, and routes the media and control traffic among endpoints (clients). The delivery of the media follows a publishing/subscribing process: each client publishes its own video, which is sent to the MCU, and the other clients get the video by subscribing to the video via the MCU. As mentioned in Section 3, since using user adaptive video coding requires video decoding and encoding, functions which the original Licode does not provide, we integrate the VP8 video codec with Licode, where VP8 is used in WebRTC.

As described in Section 3, the MCU needs both the inter-pupil distance d and the pixel density of the display in order to infer the viewing distance. To get d, we write a face detection module which makes use of the OpenCV library APIs. The pixel density of the display is obtained as follows. When a client connects to the MCU, it automatically downloads a Javascript program, which inspects the UserAgent field of the outgoing HTTP request messages to extract the device type information, which is then sent across the network to the MCU for table lookup. The table is locally maintained on the MCU, and it lists device types along with the respective display pixel densities.

The enriched Licode runs on a Linux Ubuntu 12.04 computer which serves as the MCU. The clients are Chrome web browsers, each running on a MacBook laptop. To capture the effect of Internet latency, we add artificial delays on each client using the Linux *tc* utility.

5. PERFORMANCE EVALUATION

The one-way delay from each client to the MCU is set to 50ms, which results in an RTT of 200 ms between the two clients. In the experiment, there are two subjects, each looking at a laptop, which is connected via a simulated network to the MCU.

We look at the video bit rate savings resulted from user adaptive transcoding. In the experiment, the viewing distance is fixed at 25 inches. Figure 3(a) shows the arrival bit rate (blue solid line) and the departure bit rate (red dashed line) as functions of time for the case where user adaptive transcoding is used in the MCU. Let the average arrival bit rate be r_a, and the average departure bit rate be r_d. The bit rate savings, i.e., $\eta := (r_a - r_d)/r_a$, is 36.7%. As comparison, we also plot the bit rates in Fig. 3(b) for the case where user adaptive transcoding is not used, with the average arrival bit rate denoted as R_a and the average departure bit rate as R_d. The reduction in the bit rates $(R_a - R_d)/R_a$ is only 2.7%. Since the contents of the videos are similar in the two cases, we can compare the two departure bit rates. The reduction in the average departure bit rates, i.e., $(R_d - r_d)/R_d$, is 26.9%. We do the experiment 10 times, and the average bit rate savings is 25%. The more severe fluctuation in the arrival video bit rate in Fig. 3(a) is due to the impact of not implementing Google Congestion Control (GCC) [8] in the MCU, and the difference in the arrival video bit rate between Fig. 3(a) and Fig. 3(b) will go away if GCC is implemented in the MCU.

To confirm that our scheme does not lead to significant degradation in subjective video quality as predicted by the theory, we carry out subjective testing. There are 20 subjects, and we use 3 viewing distances: 20 inches, 25 inches and 30 inches. We ask each subject to evaluate the quality of the video captured and delivered across the network in real time during a video teleconferencing, and the evaluation method is a 5-grade scale where 1 is for bad, 2 for poor, 3 for fair, 4 for good, and 5 for excellent. The results are shown in Fig. 4. We see that with 95% confidence interval, the use of user adaptive video coding does not result in degradation in subjective video quality.

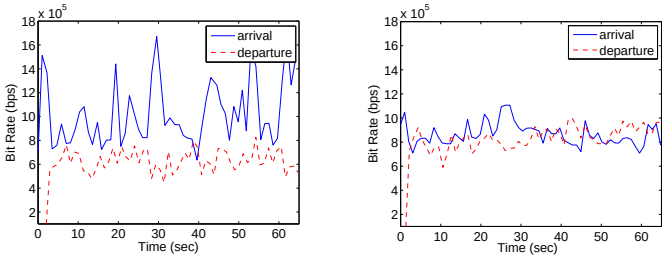

Fig. 3. The arrival video bit rate (blue solid line) v.s. the departure video bit rate (red dashed line), for the cases of user adaptive transcoding (a) being turned on, and (b) being turned off.

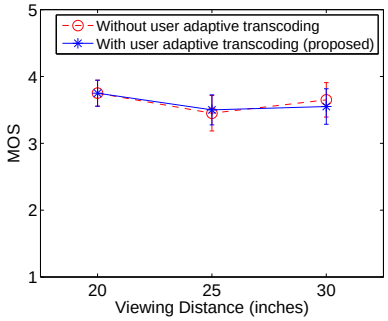

Fig. 4. Subjective testing scores for the case where user adaptive transcoding is used (blue stars) and the case where user adaptive transcoding is not used (red circles). The error bars stand for 95% confidence intervals.

6. CONCLUSION

We propose a network-based transcoding scheme for video conferencing to improve the video coding efficiency using perceptual pre-filtering. By analyzing the video sent from a client, the MCU infers that client's viewing conditions, which are then used to adapt the encoding of the video destined to the client. The scheme is implemented in a real-world video teleconferencing system. Experimental results show that our approach can significantly save the bandwidth without affecting subjective video quality.

Acknowledgement: The authors would like to thank Dr. Rahul Vanam, Dr. Louis Kerofsky, Dr. Ariela Zeira and Dr. Robert A. DiFazio of InterDigital for insightful discussions.

7. REFERENCES

[1] F. W. Campbell and J. G. Robson, "Application of fourier analysis to the visibility of gratings," *J. of Physiology*, vol. 197, no. 3, pp. 551–566, Aug. 1968.

[2] J. Movshon and L. Kiorpes, "Analysis of the development of spatial contrast sensitivity in monkey and human infants," *J. Opt. Soc. Am. A*, vol. 5, no. 12, pp. 2166–2172, Dec. 1988.

[3] P. Barten, *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality*, SPIE Press, 1999.

[4] Rahul Vanam and Yuriy A. Reznik, "Perceptual preprocessing filter for user-adaptive coding and delivery of visual information," in *Picture Coding Symposium (PCS)*, 2013, pp. 426–429.

[5] Rahul Vanam, Louis Kerofsky, and Yuriy A. Reznik, "Perceptual pre-processing filter for adaptive video on demand content delivery," in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2537–2541.

[6] W. Chen, L. Ma, G. Sternberg, Y. Reznik, and C.-C. Shen, "User-aware dash over wi-fi," in *International Conference on Computing, Networking and Communications (ICNC)*, 2015.

[7] Y. A. Reznik *et al.*, "User-adaptivemobile video streaming," in *Visual Communications and Image Processing (VCIP)*, 2012.

[8] WebRTC, "http://www.webrtc.org/," Available Online, 2014.

[9] Licode, "http://lynckia.com/licode/," Available Online, 2014.

[10] K. Khoshelham, "Accuracy analysis of kinect depth data," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 5, pp. 133–138, 2010.

[11] C. Harrison and A. K. Dey, "Lean and zoom: proximity-aware user interface and content magnification," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 507–510.

[12] C. Dickie, R. Vertegaal, C. Sohn, and D. Cheng, "Eyelook: using attention to facilitate mobile media consumption," in *ACM Symposium on User Interface Software and Technology*, 2005, pp. 103–106.

[13] OpenCV, "http://opencv.org/," Available Online, 2014.

[14] N. A. Dodgson, "Variation and extrema of human interpupillary distance," in *Proc. SPIE*, 2004, vol. 5291, pp. 36–46.

[15] J. Dostal, P. O. Kristensson, and A. Quigley, "Estimating and using absolute and relative viewing distance in interactive systems," *Pervasive and Mobile Computing 10 (2014) 173186*, vol. 10, pp. 173186, Feb. 2014, Available online July 2012.