Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices

Jihoon Ryoo[1], Kiwon Yun[1], Dimitris Samaras[1], Samir R. Das[1], Gregory Zelinsky[2]
Department of Computer Science[1]
Department of Psychology[2]
Stony Brook University
Stony Brook, New York 11794, USA

ABSTRACT

Humans see only a tiny region at the center of their visual field with the highest visual acuity, a behavior known as *foveation*. Visual acuity reduces drastically towards the visual periphery. 'Foveated' video coding/compression techniques exploit this non-uniformity to gain significant efficiency by compressing more in the periphery and less in the center. We propose a practical and scalable method to use such a technique for video streaming service over the Internet. The essential idea is to use a commodity webcam on the user side to provide real-time gaze feedback to the server with the server sending appropriately coded video to the client player. We develop a multi-resolution video coding approach that is scalable in that it is possible to pre-code the video in a small number of copies for a given set of resolutions. The coding approach is designed to match the error performance of an eye tracker built using commodity webcams. We demonstrate that the technique is energy efficient and thus usable in mobile devices. We develop a methodology for performance evaluation of such a system when network budgets may vary and video quality may fluctuate. Finally, we present a comprehensive user study that demonstrates a bandwidth reduction of a factor of 2 for the same user satisfaction.

CCS Concepts

•**Information systems** → **Multimedia streaming;**

Keywords

Foveation, video streaming

1. INTRODUCTION

Various industry analysts [32, 27] report that over half (and projected to be over 80 percent in near future) of the Internet traffic during evening peak hours is from streaming video services, such as Netflix and Youtube. Specifically, the so-called 'cord-cutters' contribute significantly to this consuming over 100 hours of video per month per household, on average. The cord cutters' proportion is increasing fast. The average quality/resolutions of available videos are also improving fast. However, there is hardly enough available bandwidth. Roughly, the average bandwidth available per household in many developed countries is barely enough to stream only two HD quality videos concurrently [33, 24]. While content providers are intent on making available higher-resolution 4K videos for streaming, and display prices are falling fast, no ISP currently can sustain the bandwidth needed for such videos at scale [30]. This is even at only a slow frame rate (30 fps) and with the most aggressive compression.

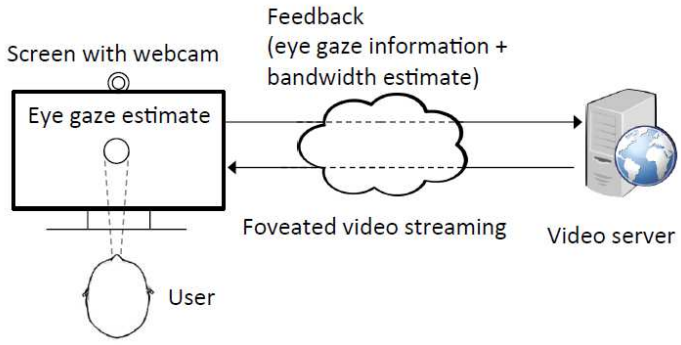


Figure 1: Overview of the proposed foveated video streaming system.

Similar bandwidth concerns apply to mobile platforms, albeit at a different scale. More and more media is now consumed on mobile devices and often outside the home/work networks. Both cellular providers and ISPs employ different rate plans and data caps, making significant media consumption very expensive for the end user. They are also widely understood to employ various forms of traffic shaping and differentiations to reduce stress on their networks (e.g., [7]) that in turn affects end users' quality of experience.

We propose to alleviate this bandwidth crunch by developing a new, variable resolution video streaming service that compresses the video based on where a person is looking with their central or *foveal* vision – a behavior known as *foveation*. The human visual system (HVS) samples information very non-uniformly; sampling is very dense at the center of our visual field (fovea) but drops of roughly quadratically with distance from the center. This decrease in sampling rate explains why human vision is blurred in the visual periphery – we notice this by keeping our gaze fixed straight ahead and

MMSys'16, May 10-13, 2016, Klagenfurt, Austria

DOI: http://dx.doi.org/10.1145/2910017.2910592

trying to read something out of the corner of the eye.

Our idea is to compress video so as to roughly match this sampling limitation of human vision, under the assumption that high-resolution video falling on the low-resolution peripheral retinas of viewers will be wasted. While a number of techniques for achieving a similar compression by exploiting the aforementioned property of the human visual system do exist (see, e.g., [39, 2, 12, 29]), existing methods of video streaming have failed to exploit these advances. The key reason is that it is usually unknown where a person's gaze will be 'pointing' at any given time while watching a video. This generally requires a continuous feedback of the gaze information to the video streaming server so that the server can deliver an appropriately compressed video taking into account the gaze information and available bandwidth estimate. See Figure 1. Determining gaze position typically requires a separate 'eye tracker' that is not a commodity device. A more realistic and scalable solution is needed for wide-spread adoption of foveated video compression for video streaming over the Internet.

With this backdrop we make three contributions in this paper:

1. We develop a multi-resolution video coding approach. The approach is 'scalable' in that only a limited number of copies of the video at different resolutions need to be stored at the server (Section 3.1). The coding approach is designed to match the error performance of an eye tracker built using a commodity webcam (Sections 3.2, 3.3).

2. We demonstrate that the technique is energy efficient and thus usable in mobile devices (Section 3.4).

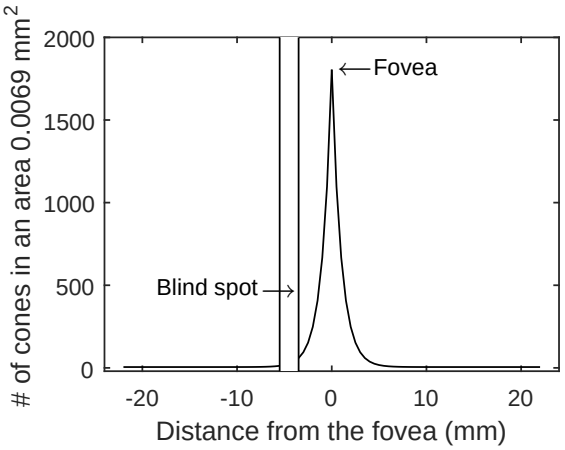
3. We develop a methodology for performance evaluation of such a system. We use this methodology to perform a comprehensive user study (Section 4). The user study shows that significant bandwidth savings are possible by adopting such foveated video streaming without degrading perceptual video quality.

2. BACKGROUND AND RELATED WORK

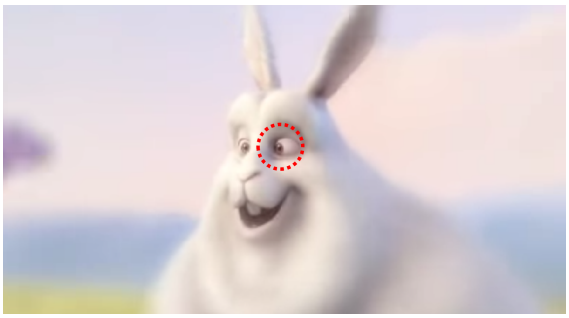
2.1 Concept of Foveation

The light passing through the optics of the human eye projects on the retina and is sampled by the photoreceptor cells – rods and cones. These photoreceptors are non-uniformly distributed over the surface of the retina. See Figure 2. The concentration of cones (the specific type of photoreceptors responsible for chromatic vision in good lighting conditions) is the highest at the center of retina (zero eccentricity) in a very small area– called the *fovea*. The fovea occupies only about 2° of the visual field and is roughly the width of your index fingernail at arm's length. The concentration of cones declines almost quadratically with increasing eccentricity (see Figure 2(a)). This non-uniform sampling gives rise to a very sharp central vision (also called *foveal* vision) and rapid loss of sharpness as one moves away from the fovea.

When a human observer gazes at a point in a real-world image, a variable resolution image is thus transmitted to the brain. The region around the point of fixation (or foveation point) is imaged onto the fovea, sampled with the highest



(a) Density of photoreceptors on retina



(b) Example of foveated compression (gaze focused on eye of the rabbit)

Figure 2: Human visual system (HVS)

density, and perceived by the observer with the highest visual acuity. The sampling density, and thus the visual acuity, decrease dramatically with increasing eccentricity. Despite this non-uniform sampling in the human visual system, traditional imaging techniques use uniformly sampled images in rectangular lattices. This is clearly inefficient. More effective would be to roughly match the level of video compression to the non-uniform sampling introduced by the human visual system.

This observation gave rise to a significant body of work on foveated image processing (see, e.g. [15, 6, 38, 28]) targeted around various applications. This also included image/video compression and efficient communication. Clearly, compression losses are well tolerated in peripheral regions as opposed to near the fixation point. This can be utilized to produce variable rate compression and reduce the bandwidth required to transmit the same image for a similar level of user satisfaction. Similarly, in noisy communication environments, foveation provides a natural way for unequal error protection for different spatial regions of the image. Despite these advances, use of fovated compression techniques for Internet video streaming is far from a reality. This is largely due to scalability reasons. First, the compression must be computed in real time on the server side using the gaze feedback. Second, precise gaze feedback typically needs expensive eye trackers that are not commodity and also inconvenient to set up and use. In this work we address the former issue by designing a multi-resolution precoding ap-

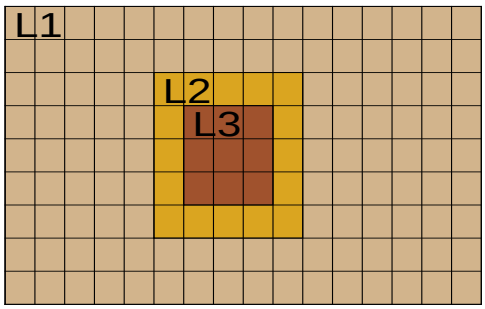


Figure 3: Multi-resolution coding in a 16×9 grid used in the experiments. The regions L_1, L_2 and L_3 are coded in progressively higher resolutions.

proach compatible with modern day video servers. Thus, the need for real time computation is eliminated. We address the latter issue by using commodity webcam-based eye tracking. We discuss eye tracking next.

2.2 Eye Tracking

Eye tracking techniques are capable of providing very good estimates of the point of fixation needed for foveated video processing. Most common eye trackers today use a combination of infra-red (IR) illumination and an IR camera [13, 23]. The basic concept is to use the light source to illuminate the eye causing highly visible reflections, and the camera to capture an image of the eye showing these reflections [36]. The image captured by the camera is then used to identify the reflection of the light source on the cornea (glint) and in the pupil. Geometric calculations based on the relative positions of these images are then used to calculate the gaze direction. Most commercial eye trackers, such as Tobii [36], The Eye Tribe [35], and SMI [34], estimate eye gaze with high accuracy using the above approach. However, these devices are not commodity and are often large and expensive.

Advances in computer vision have now enabled estimation of human gaze direction using images of the eye captured on a webcam-class camera. While the accuracy of these techniques is generally poorer than IR-based trackers, these techniques could be very useful in practice as webcams are commodity and many end-user devices are already equipped with webcams.

Several related webcam-based eye-monitoring techniques have been recently extended to mobile smartphones and tablets [40, 22]. This trend is expected to continue. This generally establishes the potential of using eye tracking on commodity video platforms.

3. DESIGN OF THE FOVEATED STREAMING VIDEO SYSTEM

The basic design of the foveated video streaming system is similar to a conventional adaptive streaming video system [9, 14]. See Figure 1. The client player continuously estimates the available network bandwidth and possibly also available compute capacity on the user device and requests the next chunk of video in the appropriate resolution. The requested resolution is commensurate with the available network bandwidth and compute capacity available for decoding such that the required display frame rate can be maintained. Typically, the video is pre-coded at the server at a set of standard resolutions, as real time encoding could be difficult. The available resolutions are already known to the client at the initial negotiation time and thus the client only asks for one of the available resolutions for each chunk. In the foveated streaming system the basic framework is the same except that the client now feeds back the gaze information periodically to the server.

3.1 Multi-resolution Coding

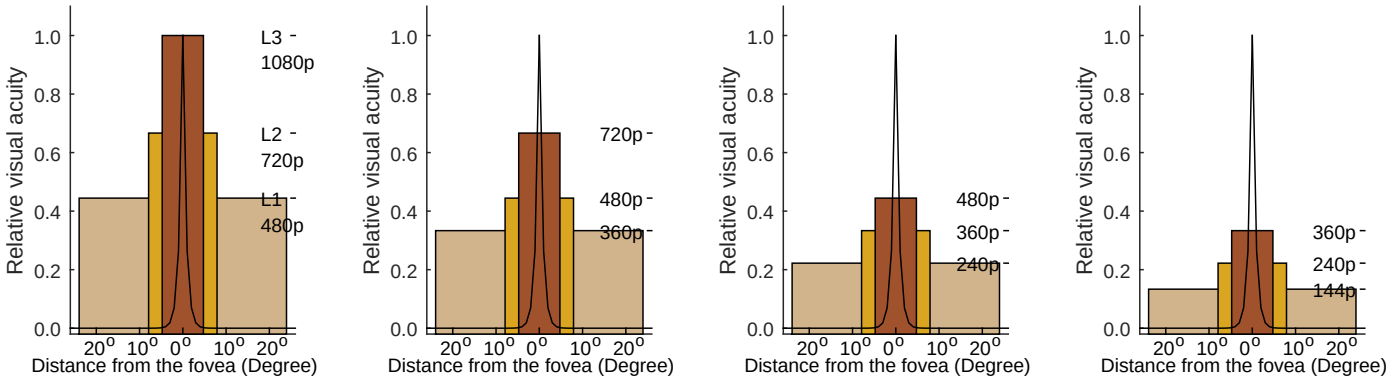
A design decision had to be made about how the video is to be coded on the server side. Foveated image/video coding is a mature topic (see a review article in [39]). However, the existing techniques are dependent on precise gaze feedback (point of fixation) and the coding is dependent on this point. This makes it hard to pre-code videos, as only a limited number of encodings are possible for a given video for scalability reasons.

Instead, we take a more practical approach. Due to the computational and network delays, the feedback to the server about the gaze information is expected to have a lag. Thus, precise real-time gaze information is not likely available. Additionally, the gaze feedback can only be periodic and not continuous. Thus, in between feedback events the gaze estimate is only approximate. (We study this aspect in more detail in Section 3.4.) Given these sources of error, the coding approach we use must be tolerant to errors in the gaze estimate. Based on this observation we take the following approach in order to pre-code videos.

Assume that the screen is split into a rectangular grid – a 16×9 grid is used in all experiments. See Figure 3. The choice of this grid size is somewhat ad hoc. 16×9 is similar to several standard screen sizes. Splitting the grid further (e.g., 32×18 etc.) would result in too many grid cells and affect scalability. Each of the 144 grid cell thus produced is now pre-coded in a standard set of resolutions that are commonly offered on Internet streaming servers. These cells are divided in three sections – L_1, L_2 and L_3, coded in progressively higher resolutions. The sections are organized as shown in Figure 3 with L_3 in the middle, surrounded by L_2 and then L_1. The general idea is to place L_3 such that the estimated fixation point is at the center.

Other important design choices include: (i) the respective sizes of the sections L_i and (ii) the actual resolutions that each of the L_i's should be coded in. These choices are made such that the offered screen resolution follows the central and peripheral visual acuity in the human visual system (Figure 2(a)) as closely as possible. For (i) we choose L_3 as a 3×3 region with L_2 as a unit width annular region around it (Figure 3). The rest of the frame is L_1. For (ii) we use 6 different resolutions for individual grid cells (144p, 240p, 360p, 480p, 720p, 1080p) and choose selected combinations of these in different sections (L_i). Overall, 4 different coding levels are used - $F1$ through $F4$ - with different choices of resolutions for the sections L_i. The actual choices made are shown in Figure 4.

The baseline system chosen for performance bench marking uses 5 different resolutions, from 240p to 1080p. These resolutions are presented uniformly to the entire screen. They are referred to as $B0$ through $B4$. All different choices are



(a) $F1$ (b) $F2$ (c) $F3$ (d) $F4$

Figure 4: Four resolution levels (F_1 through F_4) chosen for the foveated system are shown along with the visual acuity in human visual system. The four levels consume progressively lesser network budget. In each level F_i, the choices of the actual resolution for the sections L_1 through L_3 are made so that they map, in relative terms, as closely as possible to the visual acuity.

again summarized in Table 1 for easy reference.

We now compare the network capacity (bits/sec) needed to transmit the aforementioned resolution levels Bi (Fi). To do this we use the capacity needed for 2K video (i.e., 2048×1080p) as the reference, as if consuming 100% of capacity budget. We then determine the bits/sec capacity needed for all different resolution levels and normalize them against the above reference. The results are summarized in Figure 5. The capacity numbers are determined via actual measurement of bits/sec captured from real network traces. Thus, they capture the actual load on the network including all packet header and protocol related control packet overheads.

This figure demonstrates the network efficiency of the foveated system. For example, with $\approx 50\%$ network budget, the baseline player can play at only 480p, but the foveated player can play 1080p for L_3, 720p for L_2 and 480p for L_1. As evaluations will show later, the latter player provides a much improved user satisfaction without consuming any additional resources.

Summary: In the proposed foveated system, the video is pre-coded on the server side with each of the 144 grid cells pre-coded and stored in 6 base resolutions. Then they are 'stiched' appropriately to form the three sections L_1 through L_3 (Figure 3) at different chosen resolutions, given the resolution level to be played - $F1$ through $F4$ (Figure 4). The positions of sections L_i on the frame is determined from the gaze feedback received from the client. The resolution level is chosen based on the network capacity budget (Table 1). The stitching is actually done on the client side in our experimental system for ease of implementation, but server-side mechanisms are possible.

We will now describe the design of the gaze tracker on the client side.

3.2 Gaze Tracker Design

Low-cost real-time gaze tracking is at the core of foveated video streaming, as the perceived quality of the video depends on real-time knowledge of the viewer's visual attention. In a real-world service (i) the system should have adequate accuracy while using only inexpensive, commodity

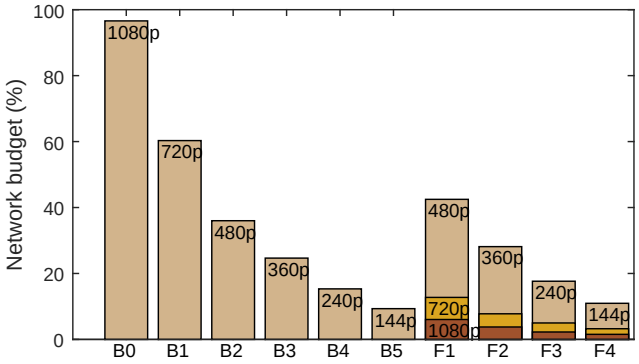


Figure 5: Network capacity used by different resolution levels in baseline and foveated players when normalized against uniform resolution 2K video. For the foveated player, the portions of capacity budget consumed by the different layers L_i are also shown.

hardware; (ii) it should have low computational cost while providing close to real-time service; and (iii) it should be energy efficient overall so that the same general technique can apply to mobile devices. Any off-the-shelf method using a regular webcam allowing for natural head movement (Section 2.2) can be used for our service [20, 40, 44]. High accuracy is not required given the tolerance of our basic design – the highest resolution section L_3 is $\approx 8^\circ$ degrees wide. This is a significant margin as we will see in our evaluations later. We use an open source implementation for our webcam-based gaze tracker [26, 25, 11] and show that it performs adequately to the system needs.

The specific off-the-shelf webcam-based gaze tracker we used is called *GazePointer* [11]. Gazepointer performs in real time (30 frames/sec) and works even with a low-resolution webcam – this makes it widely useful in commodity platforms and also saves energy. It also does not require significant computational load (more on this later).

Baseline Player				Foveated Player				
Res. level	Resolution	Network budget(%)		Res. level	Network budget(%)	L_3	L_2	L_1
B0	1080p	96.6						
B1	720p	60.35	$\longleftrightarrow$	F1	42.48	1080p	720p	480p
B2	480p	36.00	$\longleftrightarrow$	F2	28.14	720p	480p	360p
B3	360p	24.66	$\longleftrightarrow$	F3	17.65	480p	360p	240p
B4	240p	15.33	$\longleftrightarrow$	F4	10.95	360p	240p	144p

Table 1: Resolution levels used in the baseline and foveated players.

For the benefit of the reader, we briefly describe how Gazepointer works. More details are in [5]. Gazepointer follows the conventional model-based approach. First, it detects the position of the face at the first frame [37]. Then, when the face is detected, the characteristic points of the face are identified and a parameterized face mask is automatically mapped on the face. Detection of the characteristic points of the face and their temporal tracking are based on an Active Appearance Model [4] with regard to the 3D position of the camera. The eye corners are detected from the 3D face model, then the precise boundary of the pupil is extracted by a technique similar to [18]. The estimated eye position is computed based on the vector difference between the pupil center and the eye corners.

In order to calculate the fixation point on the screen, a mapping between locations on the screen and an estimated point must be established through an initial calibration procedure. The calibration works by looking at 9 different points on the screen. Re-calibration may be necessary if the head moves significantly. In our experience the system tolerates normal head movements well, when the user is within normal viewing distance and is engaged in the video on screen. In our user experiments re-calibrations were rarely needed.

3.3 Gaze Tracker Evaluation

We evaluate the gaze tracker above for accuracy. For this set of experiments, we use a 1920×1080 resolution 24 inch monitor with a webcam mounted on top. The screen is placed at a reading distance ($\approx 50\,\text{cm}$) from the subject. The subjects are asked to look at 100 points flashed at random locations on the screen and the degree of error is calculated from the point location and the estimated gaze point. We perform two different sets of experiments: (i) with *head fixed* using a chin-support to provide a set of baseline measurements, and (ii) with *head free* that would be the normal operating condition allowing free head motion. Two different webcam resolutions are used – 320×240 and 640×480.

Figure 6(a) shows the accuracy of the gaze tracker in terms of the error CDF (cumulative distribution function). As expected, if the resolution is lower or the head moves freely the performance is somewhat poorer. But overall the performance is quite good – with the 90-th percentile error between $6.3^\circ - 8^\circ$ depending on resolution used when the head moves freely. Figure 6(b) shows the scatterplot of the degree error for all collected data points separated into the X and Y direction. No systematic bias in any specific direction is observed.

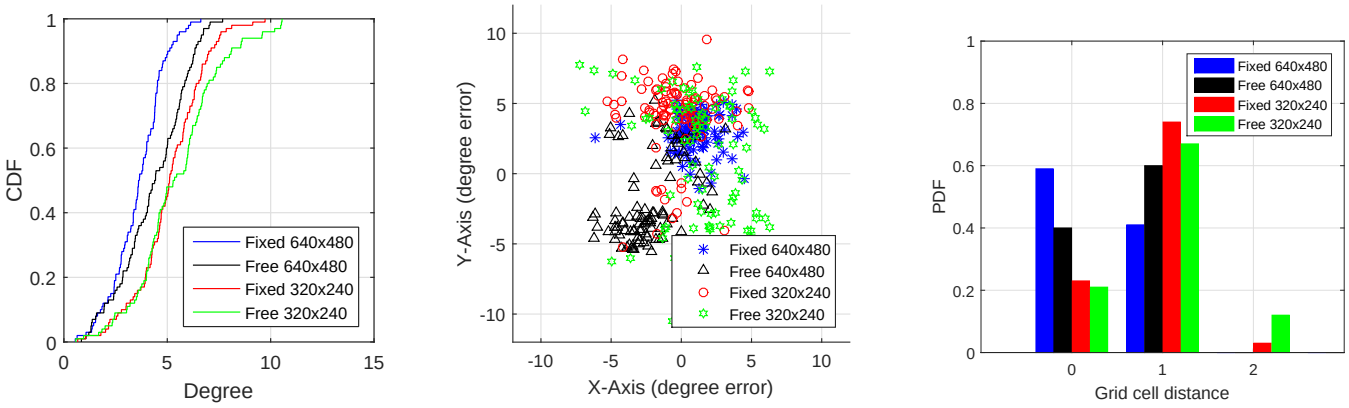
It is important to consider what implications the degree of eye tracker error presented here has for the design choices we made in Section 3.1. To do this, we map each point to the corresponding grid cell (out of 144) and determine the error in terms of the grid cell distance (distance 1 is a neighbor cell when cells are considered 8-connected, etc). The distribution is presented in Figure 6(c), showing that the 90-th percentile error is within distance 1. Thus, given our conservative choice of the size of L_3 (3×3 grid) there is over a 0.9 probability that the user will perceive his/her central vision with the highest offered resolution.

We note here that with improvements in webcam-based gaze tracking and improved camera resolutions, it would be possible to make a much 'tighter' choice for L_3 for even better network performance.

3.4 Discussion

3.4.1 Saliency

Instead of using eye gaze directly to determine visual attention, it may be sufficient to use automatic *saliency detection* as a proxy for gaze position [3, 21, 15, 43]. The idea is to preprocess the video to create a *saliency map* that predicts regions of interest (ROI) that have been shown to correlate with viewer fixations. Various types of saliency have been studied in the computer vision literature, such as saliency maps using low-level image features (e.g., color or intensity) or semantic saliency maps that use higher level features (e.g., face or other meaningful objects) [45]. However, saliency-based techniques have only limited potential in our context. First, the ROI determined via automated techniques may still be quite large in proportion to the screen size. Second, predictions of attention from saliency maps are far from perfect. It is well-known that human attention and fixations reflect the goals of the viewer [41], and these top-down goals are not captured by saliency models. Third, there is often poor agreement in the eye movements of people viewing the same image or video [8, 42], and this variability is also not captured by saliency models. However, saliency detection can still be useful in predicting a few frames ahead, when gaze feedback has substantial lag. It may be possible to learn more personalized, and consequently more accurate, saliency maps from the history of eye movements for a specific user viewing a specific video [31]. If so, perhaps automated saliency detection can be integrated into our foveated video streaming system to further improve performance.



(a) Error CDF (b) Error distribution in XY plane (c) Error distribution in terms of grid cell distance

Figure 6: Performance of the webcam-based gaze tracker used in evaluation.

3.4.2 *Handling lag*

The gaze tracker presented in Section 3.2 works at 30 Hz providing an update every 33 ms. On top of this, there is a network delay that could be in the order of 10-1000 ms on the Internet. But Internet latencies are improving fast with the introduction of CDNs by content providers. As a reference, a 2009 study on Google CDNs shows that the 90-th percentile one way delay is about 100ms [17]. To understand how much the eye gaze can move within this time, we analyzed a dynamic eye movement dataset for subjects watching video [21]. From this dataset, we computed the pixel distance that the subject's eye gaze travels within a given interval of time and mapped this pixel distance to the size of the center section L_3. The analysis showed that it took up to an average of $\approx$ 660 ms for the gaze point to move outside of L_3 provided the initial fixation was right at the center of L_3. This bolsters our conviction that lag in the gaze feedback is not likely to be a significant problem except in a network with poor infrastructure. Further, various gaze estimation methods (e.g., [10, 16]) can be used to supplement and overcome possible lags.

3.4.3 *Buffering*

Most streaming systems pre-fetch frames in advance of playing and buffer them at the client. The general goal is to 'ride out' transient network bottlenecks. The amount of buffering is very design specific. Buffering introduces a problem with foveated videos as frames pre-fetched significantly in advance lack gaze feedback and may not be compressed correctly. To alleviate this, we propose to prefetch the entire frame only in the lowest resolution (i.e., L_1 resolution) consistent with the level being played. For example, when the video is being played at level F_2, the entire frame is pre-fetched at resolution 360p. See Table 1. The higher resolution parts of the image for L_3 and L_2 are fetched at approximately real-time based on gaze feedback. Further optimizations can be done by using a person's history of eye movements to pre-fetch parts of $L3$ and $L2$ in advance as well. Given that $L3$ and $L2$ present a small part of the network budget (Figure 5), we expect this to work well in practice.

3.4.4 *Screen size*

While the user experiments reported in this paper are done on relatively large screens (desktop monitor), our work is not fundamentally limited by screen size. This is because changes in viewer fixation still occur even for smaller screens. The fovea occupies only about 2° in the visual field (Sec 2.1), an area smaller than a fingernail at typical smartphone viewing distance. Thus, large regions of a phone screen can still be delivered at a much lower resolution without affecting the viewing experience.

Broadly, the only practical concerns are: (1) the angle that a screen subtends at the viewer's eye at a typical viewing distance and (2) degree-wise accuracy of gaze tracking with respect to this angle. Both the 24 inch and 12 inch screens (studied in the paper) subtend approximately $45° - 50°$ degree angle at the eye. For smaller screens, the viewing angle reduces somewhat falling to about 38° for a 6 inch screen, for example. This is due to the shortening of the viewing distance, which we expect will also improve the gaze tracking accuracy proportionately. Thus, the overall improvement is still expected to be quite similar for small screen devices.

3.4.5 *Number of viewers*

So far we have assumed that there is only a single viewer. Solo video watching is quite common, especially on personal devices. However, in principle our approach can be extended to multiple viewers, although tracking multiple eyes simultaneously does become computationally expensive. Also, if the viewers look at different locations on the screen, the efficiency of the system can diminish as multiple regions would have to be fetched at higher resolution. But several intermediate solutions are possible. For example, our foveated system can be used only when viewer fixations overlap.

3.4.6 *This work*

Our work here does not attempt to design a complete end-to-end solution including handling of real network lag or buffering issues, which are beyond the scope of this paper. Instead, our goal is to highlight the potential for a commodity-based, scalable foveated approach to reduce the network capacity budget and, in so doing, improve user satis-

faction for the same budget. We do this via a comprehensive user study described in the following section.

4. USER STUDY

We describe a comprehensive user study in this section to determine the perceptual quality of the foveated video streaming. The goal is to showcase the potential for significant network budget reduction for the same perceptual quality, or significant improvement of perceptual quality for the same network budget. We first describe the user study set up and then present the results.

4.1 Study Setup

The study was conducted on a lab table top using a 24-inch monitor (1920×1080 resolution, 53.3×29.6 cm physical screen) driven by a general purpose PC (Intel i5, 4GB RAM, Windows 8.1). The subjects were positioned at a comfortable viewing distance (approx 50-60 cm). Thus, the viewing angle was about 50° with each grid cell subtending about 3° at the eye.

4.1.1 Choice of videos and recruitment of subjects

We picked 48 videos of 1080p (1920×1080) resolution from 16 channels of YouTube.[1] From each channel, the 3 most popular 1080p videos were chosen, for a total of 48 videos. Each video was then encoded in multiple resolutions: 1080p (original), 720p, 480p, 360p, 240p, and 144p. Only a 30 sec clip from each video was picked for the actual study.

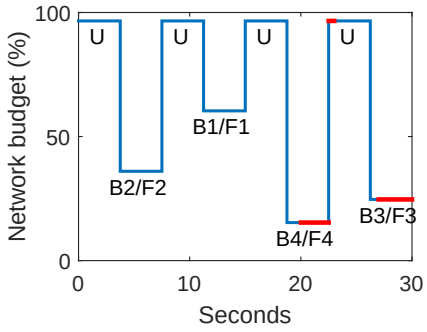
We recruited 16 subjects, half male and half female, with ages ranging from 22-45 years and with varying ethnic backgrounds. None of the subjects were aware of the nature of experiments to be conducted. The subjects were instructed to watch video clips, and to press a single button when they perceived the video to be of poor quality and to release the button when they perceived the video to be of good quality. Button press times were logged for later evaluation. This task was intended to provide a relatively non-intrusive measure of video quality, with longer total button press times indicating a poorer quality viewing experience. Subjects were familarized with the task before the study by having them view a sample video (not part of the data set) at different levels of quality during a practice session. Nevertheless, 3 out of 19 tested subjects were excluded from the study for not following the instructions, leaving the data from only 16 subjects for analysis.

4.1.2 Structuring subjects and videos

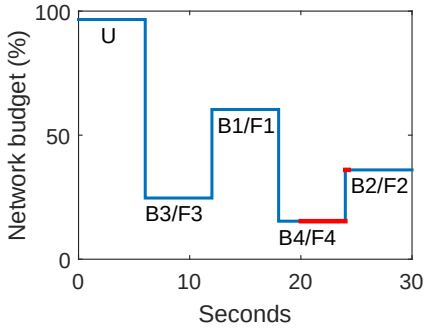
The experiment was counterbalanced such that each subject was randomly paired with another subject for the purpose of evaluation. One subject of the pair watched a random half of the 48 clips in baseline mode and the other half in foveated mode The other subject watched the same sequence of clips - but in the opposite modes. In other words if one watched the 48 clips in the following fashion:

```
CLIP1 - CLIP2 - CLIP3 - CLIP4 - --- - CLIP48
Foveat - Base - Foveat - Base - --- - Base
```

[1]Youtube has 19 channels but 3 of them are derivatives. The 16 chosen channels considered are 1) Music, 2) Comedy, 3) Film and Entertainment, 4) Gaming, 5) Beauty and Fashion, 6) TV, 7) Automotive, 8) Animation, 9) Sports, 10) Tech, 11) Science and Education, 12) Cooking and health, 13) Causes and non-profit, 14) News and Politics, 15) Lifestyle, 16) How-to and DIY.



(a) Sequence of first 24 videos (U at every transition)



(b) Sequence of last 24 videos (U once at start)

Figure 7: Sequencing of video resolutions for the user study. The network budget is from Table 1 and is shown only as a guide. The red segments correspond to a button press indicating poor quality perception from an example trace.

the other would watch the clips in the sequence:

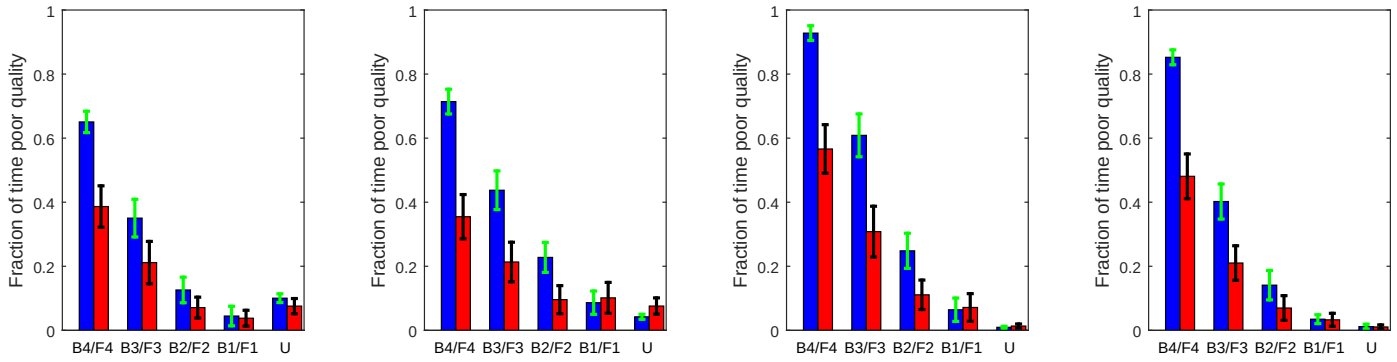
```
CLIP1 - CLIP2 - CLIP3 - CLIP4 - --- - CLIP48
Base - Foveat - Base - Foveat - --- - Foveat
```

We also made sure that the same content category was shown equally in the baseline and foveated modes. This was done because a users' sensitivity to poor video quality may depend on the video content.

4.1.3 Sequencing video resolutions

To measure perceptual video quality we first needed a reference. The reference would ideally be an uncompressed version of the video. We used the 1080p baseline video ($B0$ in Table 1) as a proxy for this ideal, which we denote as U. The idea is to show a segment of the U video either once at the start of the clip, or to periodically show the U video to subjects so as to enable them to recalibrate their expectation of good video quality and to obtain perhaps meaningful feedback about drops in quality when compressed versions are shown. More specifically, for each 30 sec video clip the two sequencing conditions were:

1. *Uncompressed after every transition:* Alternate between uncompressed and compressed several times for each video clip. For example, the 30 sec clip might be played in 8 segments (shown below), with a different resolution used for each 3.75 sec segment (stitched together to make a continuous video stream).



(a) U at every transition (b) U once at start (c) U at every transition (compensated) d) U once at start (compensated)

Figure 8: User study results: perceptual video quality vs. various compression levels in the baseline and foveated players. Error bars indicate standard errors. Note that foveated streaming results in up to a 50% reduction in the time that poor video quality is perceived.

$$U \rightarrow B2 \rightarrow U \rightarrow B1 \rightarrow U \rightarrow B4 \rightarrow U \rightarrow B3$$

This condition models varying resolutions in the compressed versions ($B2, B1$ etc) due to a varying network capacity budget, but U is always shown before the compressed version to recalibrate the subject.

2. *Uncompressed once at start:* Show uncompressed segment only once at the start of a clip, followed by compressed segments varying in resolution. For example, the 30 sec clip might be played in 5 segments (shown below), with each 6 sec long segment having a different resolution.

   $$U \rightarrow B3 \rightarrow B1 \rightarrow B4 \rightarrow B2$$

   Note that in this condition U is shown only once at the start of the clip for subject calibration, followed by segments varying in resolution that might reflect varying network capacity budget.

The two sequencing methods enable different evaluations of the perceptual impact of the varying video resolutions. This is because a subject may perceive the same quality on the same clip differently depending on what was shown immediately prior.

For each subject, the first 24 video clips are shown under condition (1) and the second 24 are shown under condition (2). See Figure 7 for a graphic depiction. Note that each subject watched half of the videos with baseline compression ($B1, B2$, etc.) and other videos with foveated compression ($F1, F2$, etc.), and we ensured that the same sequences were used for both. For example, if a subject watched a video clip as $U \rightarrow B3 \rightarrow B1 \rightarrow B4 \rightarrow B2$, his/her paired viewer watched the same clip as $U \rightarrow F3 \rightarrow F1 \rightarrow F4 \rightarrow F2$. The actual sequence of resolutions, e.g., $F3 \rightarrow F1 \rightarrow F4 \rightarrow F2$, was randomly selected.

Figure 7 also shows button presses from example traces. Note that the user sometimes pressed the button after a slight delay following a drop in quality, and sometimes delayed slightly their release of the button after the quality improved. Our analysis later attempts to correct for this lag in reaction time.

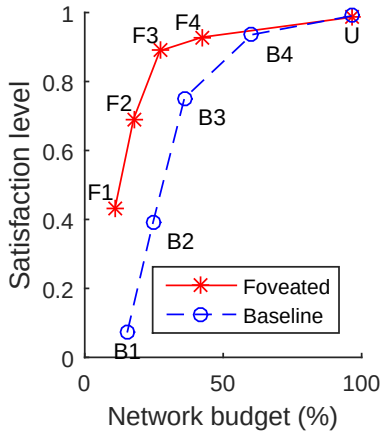
4.2 User Study Results

The results are summarized in Figure 8, which show the *fraction of time the subjects perceived poor video quality* for baseline and foveated compression conditions. The error bars indicate standard error. In the first two plots (Figures 8(a) and (b)), the total duration of button press times are used to calculate the fraction of time poor video quality was reported. The second two plots (Figures 8(c) and (d)) compensate for user reaction time lag (discussed in the previous subsection) as follows. If a button press is initiated in response to a given level of resolution, that entire clip segment shown at that resolution is counted as a poor viewing experience. Also, if the button is pressed for the entire duration of a segment (i.e., the button is pressed during an earlier segment and is released during a later segment), such segments are also counted as a poor viewing experience. No other time intervals are counted as indicating a poor viewing experience (e.g., when a button press ends but does not start in a segment) in this lag-corrected estimate.[2] Note that after this compensation, indications of a poor perception of U almost disappear, suggesting that the compensation method worked well.

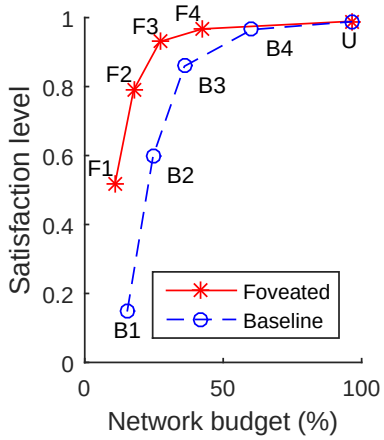
Compression levels with similar network budgets (see Table 1) are grouped together in the plots for ease of reading. For example, $B4$ and $F4$ are grouped together. Recall that in each of these cases they have similar (but not identical) network budgets, and the budget for the foveated version (e.g., $F4$) is always slighty lower (Table 1). Still, the foveated versions almost always result in a significantly better user experience. Looking at Figure 8(d), which portrays the most realistic evaluation with compensation applied, note that the fraction of poor quality perceptions roughly halves with foveated compression.

In Figure 9 the same data are plotted differently after normalizing for the quality of perception. The plots show *satisfaction level*, defined as '1 − fraction of time the sub-

[2]Thus, if the subject presses the button sometime after the start of a 'bad' interval and releases it after the start of a 'good' interval, the entire bad interval is counted as a poor viewing experience and no part of the good interval is counted as poor.



(a) U at every transition



(b) U once at start

Figure 9: Satisfaction level vs. network budget for baseline and foveated compressions.

ject perceives poor video quality,' as a function of network budget. Note that for the same satisfaction level the network budget is roughly halved for the foveated version. As expected, the difference between the two gets smaller with increasing budget, indicating that constrained networks will benefit more from foveated video streaming. Qualitatively, there is no significant difference between the two cases 'U at every transition' or 'U once at start.'

5. ENERGY MEASUREMENTS FOR MOBILE USE

Our goal is to make foveated video streaming widely available across a range of devices. Since a large and growing amount of video is consumed using untethered mobile devices, we want to ensure that our design is energy efficient when used on a mobile device. The concern is that our system requires that the webcam be continuously on, and that there is some additional computational load imposed by the gaze tracking.

We used a Microsoft Surface Pro3 tablet with 4GB RAM running Windows 8.1 to perform the energy measurements. Software-based measurements of the remaining battery level were obtained via the system-provided API. Four major

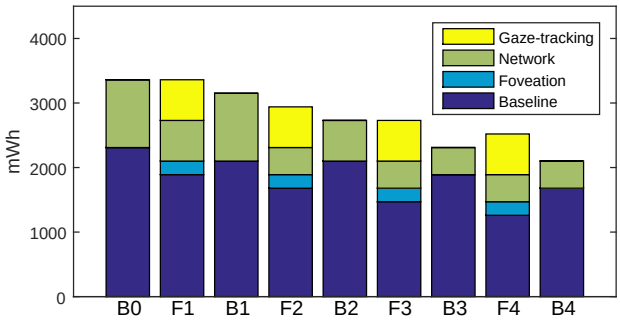


Figure 10: Energy expended in running the foveated streaming client at different resolution levels on a tablet over WiFi.

components of the client player software were systematically assessed by turning on each component incrementally and measuring the total energy expenditure for running a 24 min long video (sequence of all the clips used in the user study). An average of 5 runs are reported. The components studied were:

1. *Baseline:* Basic functionality such as display and computations for the streaming service, excluding the contribution of the network components; the network is disabled and video is stored on local flash.
2. *Foveation:* Additional video decoding computation needed for foveated compression, such as stitching the sectioned multi-resolution images.
3. *Network:* All network components while using 802.11n to connect to an external server.
4. *Gaze tracking:* Webcam and gaze tracking software.

The results are shown in Figure 10. Note that the baseline energy costs are slightly higher for baseline (Bi) playing, as a high resolution video must be decoded for the entire screen. The network energy costs are roughly similar to the network budget. The energy cost for gaze tracking is non-negligible – about 20-25% of the total. We believe that the use of lower power cameras, such as those being proposed for continuous vision applications on mobile devices [19], or low power IR cameras, such as the ones used in Amazon Fire phone [1], would reduce this component of the energy cost. Computational optimizations internal to the gaze tracker are also possible. Similar optimization is also possible in the foveation component, but this was not attempted in the current implementation. Nevertheless, even with the current, unoptimized system, the energy results are encouraging: compression levels producing similar user satisfaction levels (e.g., $B2$ and $F1$, $B3$ and $F2$, $B4$ and $F3$, etc.) incur very similar energy costs. See the user study in Section 4 for the actual satisfaction levels.

6. CONCLUSIONS

This work introduces a conceptual framework for a foveated video streaming service for Internet video servers. The goal is to exploit commodity webcams commonly available in many devices to develop a scalable system that takes eye

gaze information from the user and uses it to transfer different parts of the video frame from the server at varying resolutions. The hope is that such a service will alleviate the bandwidth crunch in today's Internet – much of it arising from streaming videos. We see the contribution of this paper being the description of this conceptual framework, with its focus on a commodity-based scalable solution, and not the delivery of a system designed and optimized for specific hardware settings. For much of our study the basic tools that we used were off-the-shelf, such as the webcam-based eye gaze tracking.

This paper also develops a robust methodology for conducting user studies aimed at comparing video quality when network budgets vary and video resolutions fluctuate. Using this methodology, we conducted a comprehensive user study that showed a factor of 2 bandwidth reduction while keeping the same user satisfaction. This is promising.

Although outside the scope of the current study, future work will move closer to a completely designed end-to-end system enabling the study of varying network conditions and prefetching and buffering techniques in the context of our foveated video streaming system. We will also attempt to combine automated saliency detection with eye gaze [3, 21, 31] so as to further improve system performance.

7. ACKNOWLEDGMENTS

The work was partially supported by NSF grants IIS-1161876 and CNS-1405740, and the SUBSAMPLE project of the DIGITEO Institute France.

8. REFERENCES

[1] Amazon Fire Phone with Dynamic Perspective. https://developer.amazon.com/public/solutions/devices/fire-phone/docs/understanding-the-dynamic-perspective-ui.

[2] A. Bokani. Empirical evaluation of real-time video foveation. In *Proceedings of the Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, VideoNext, pages 45–46, New York, NY, USA, 2014. ACM.

[3] A. Borji, D. N. Sihite, and L. Itti. Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE Transactions on Image Processing*, 22(1):55–69, 2013.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

[5] S. Deja. Real-time system for eye detection and tracking of computer user using webcam. Master's thesis, A G H University of Science and Technology, 2010.

[6] L. Dell'Osso and R. Daroff. Congenital nystagmus waveforms and foveation strategy. *Documenta Ophthalmologica*, 39(1):155–182, 1975.

[7] M. Dischinger, M. Marcon, S. Guha, P. K. Gummadi, R. Mahajan, and S. Saroiu. Glasnost: Enabling end users to detect traffic differentiation. In *Networked Systems Design and Implementation (NSDI '10)*, pages 405–418, 2010.

[8] M. Dorr, T. Martinetz, K. R. Gegenfurtner, and E. Barth. Variability of eye movements when viewing dynamic natural scenes. *Journal of Vision*, 10(10), 2010.

[9] Dynamic Adaptive Streaming over HTTP. http://en.wikipedia.org/wiki/dynamic-adaptive-streaming-over-http.

[10] Y. Feng, G. Cheung, W. tian Tan, and Y. Ji. Hidden markov model for eye gaze prediction in networked video streaming. In *IEEE International Conference on Multimedia and Expo (ICME '11)*, pages 1–6, July 2011.

[11] GazePointer. http://gazepointer.sourceforge.net/.

[12] W. S. Geisler and J. S. Perry. Real-time foveated multiresolution system for low-bandwidth video communication. In *Electronic Imaging*, pages 294–305. International Society for Optics and Photonics, 1998.

[13] D. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):478–500, March 2010.

[14] HTTP Live Streaming. http://en.wikipedia.org/wiki/HLS.

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

[16] O. Komogortsev. Predictive perceptual compression for real time video communication. In *ACM Multimedia*, pages 220–227, 2004.

[17] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 190–201. ACM, 2009.

[18] D. Li, D. Winfield, and D. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *IEEE CVPR Workshop on Computer Society*, pages 79–79, June 2005.

[19] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th MobiSys*, pages 69–82. ACM, 2013.

[20] F. Lu, Y. Sugano, T. Okabe, and Y. Sato. Adaptive linear regression for appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):2033–2046, Oct 2014.

[21] S. Mathe and C. Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *Computer Vision–ECCV 2012*, pages 842–856. Springer, 2012.

[22] E. Miluzzo, T. Wang, and A. T. Campbell. Eyephone: Activating mobile phones with your eyes. In *Proceedings of the Second ACM SIGCOMM Workshop on MobiHeld*.

[23] C. H. Morimoto and M. R. Mimica. Eye gaze tracking techniques for interactive applications. *Journals on Computer Vision and Image Understanding*, 98(1):4–24, 2005.

[24] OECD. Average and median advertised download

speeds, http://www.oecd.org/sti/broadband/oecdbroadbandportal.htm, Sept 2012.

[25] OpenEyes. http://thirtysixthspan.com/openeyes/software.html.

[26] OpenGazer. https://github.com/tiendan/opengazer.

[27] C. L. Palermino. Online video will account for 80 percent of the world's internet traffic by 2019. Technical report, Digital Trends, 2015.

[28] J. Park, S. Lee, and A. Bovik. 3d visual discomfort prediction: vergence, foveation, and the physiological optics of accommodation. *IEEE Journal of Selected Topics in Signal Processing*, 8(3):415–427, 2014.

[29] J. S. Perry and W. S. Geisler. Gaze-contingent real-time simulation of arbitrary visual fields. In *Electronic Imaging*, pages 57–69. International Society for Optics and Photonics, 2002.

[30] D. Rayburn. The Adoption Of 4K Streaming Will Be Stalled By Bandwidth, Not Hardware & Devices in http://blog.streamingmedia.com/2015/01/4k-streaming-bandwidth-problem.html, January 2015.

[31] D. Rudoy, D. Goldman, E. Shechtman, and L. Zelnik-Manor. Learning video saliency from human gaze using candidate selection. In *Computer Vision and Pattern Recognition, (CVPR '13)*, pages 1147–1154, June 2013.

[32] Sandvine Incorporated ULC. GLOBAL INTERNET PHENOMENA REPORT. Technical report, Sandvine Intelligent Broadband Networks, 2014.

[33] B. Sanou. The World in 2013 ICT Facts and Figures, http://www.itu.int/en/itu-d/statistics/documents/facts/ictfactsfigures2013-e.pdf, 2013.

[34] SMI Sensomotoric Instruments. http://www.smivision.com/.

[35] The Eye Tribe. https://theeyetribe.com/.

[36] Tobii. http://www.tobii.com/en/.

[37] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

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

[39] Z. Wang and A. C. Bovik. *Foveated image and video coding*. Marcel Dekker Series in Signal Processing and Communications, 2005.

[40] E. Wood and A. Bulling. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 207–210, New York, NY, USA, 2014. ACM.

[41] A. L. Yarbus. *Eye Movements and Vision*. Vision Science: Photons to Phenomenology, 1967.

[42] K. Yun, Y. Peng, D. Samaras, G. J. Zelinsky, and T. L. Berg. Exploring the role of gaze behavior and object detection in scene understanding. *Frontiers in Psychology*, 4(917), 2013.

[43] Y. Zhai and M. Shah. Visual attention detection in video sequences using spatiotemporal cues. In *ACM Multimedia*, MULTIMEDIA '06, pages 815–824, New York, NY, USA, 2006. ACM.

[44] Y. Zhang, A. Bulling, and H. Gellersen. Pupil-canthi-ratio: A calibration-free method for tracking horizontal gaze direction. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 129–132, New York, NY, USA, 2014. ACM.

[45] Q. Zhao and C. Koch. Advances in learning visual saliency: From image primitives to semantic contents. In *Neural Computation, Neural Devices, and Neural Prosthesis*, chapter 14, pages 335–360. Springer, 2014.