Perceptual Compression for Video Storage and Processing Systems

Amrita Mazumdar
University of Washington

Brandon Haynes
University of Washington

Magda Balazinska
University of Washington

Luis Ceze
University of Washington

Alvin Cheung
University of California, Berkeley

Mark Oskin
University of Washington

ABSTRACT

Compressed videos constitute 70% of Internet traffic, and video upload growth rates far outpace compute and storage improvement trends. Past work in leveraging perceptual cues like saliency, i.e., regions where viewers focus their perceptual attention, reduces compressed video size while maintaining perceptual quality, but requires significant changes to video codecs and ignores the data management of this perceptual information.

In this paper, we propose Vignette, a compression technique and storage manager for perception-based video compression in the cloud. Vignette complements off-the-shelf compression software and hardware codec implementations. Vignette's compression technique uses a neural network to predict saliency information used during transcoding, and its storage manager integrates perceptual information into the video storage system. Our results demonstrate the benefit of embedding information about the human visual system into the architecture of cloud video storage systems.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Information systems → Multimedia information systems**; *Storage management*.

KEYWORDS

Storage management, video compression, video streaming

ACM Reference Format:
Amrita Mazumdar, Brandon Haynes, Magda Balazinska, Luis Ceze, Alvin Cheung, and Mark Oskin. 2019. Perceptual Compression for Video Storage and Processing Systems. In *ACM Symposium on Cloud Computing (SoCC '19), November 20–23, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357223.3362725

1 INTRODUCTION

Compressed videos constitute 70% of Internet traffic and are stored in hundreds of combinations of codecs, qualities, and bitrates [2, 11]. Video upload growth rates far outpace compute performance and storage production today, and this trend is predicted to accelerate [12, 17, 43]. New domains of video production—e.g., panoramic (360°), stereoscopic, and light field video for virtual reality (VR)—demand higher frame rates and resolutions, as well as increased dynamic range. Further, the prevalence of mobile devices with high-resolution cameras makes it increasingly easy for humans to capture and share video.

For decades, video codecs have exploited how humans see the world, for example, by devoting increased dynamic range to spatial features (low frequency) or colors (green) we are more likely to observe. One such perceptual cue, *saliency*, describes where in a video frame a user focuses their perceptual attention. As video resolutions grow, e.g., 360° video and 8K VR displays, the salient regions of a video shrink to smaller proportion of the video frame [57]. Video encoders can leverage saliency by concentrating bits in more perceptually interesting visual areas. Prior work in saliency-enabled encoders, however, focus only on achieving bitrate reduction or quality improvement at the cost of complicated, non-portable prototypes designed for a single codec implementation [22, 24, 40, 45]. In this work, we address the challenges of storing and integrating this perceptual data into cloud video storage and processing systems.

Large-scale video systems generally fall into two classes: entertainment streaming, and social media video services; saliency-based compression can provide benefits to both. For entertainment services, which maintain small numbers of videos to be streamed at many resolutions and bitrates, saliency-based compression reduces the storage cost of maintaining many bitrates and resolution scales of these videos. For social media services distributing a vast video library from many users, it reduces outbound network bandwidth. For both types of services, a system enabled to incorporate saliency prediction can improve video compression, for instance, as an initially viral video decreases in popularity, or to reduce bandwidth while streaming video to a 360° video player.

In this paper, we describe Vignette, a cloud video storage system that leverages perceptual information to reduce video sizes and bitrates. Vignette is designed to serve as a backend for large-scale video services, such as content delivery systems or social media applications. Vignette has two components: a compression scheme, *Vignette Compression*, and a storage manager, *Vignette Storage*, as shown in Figure 1. Vignette Compression leverages a new saliency-based compression algorithm to achieve up to 95% lower bitrates while minimally reducing quality. Vignette Storage uses a simple API to trigger saliency-based compression when needed, allowing applications to trade off between faster traditional compression and

SoCC '19, November 20–23, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6973-2/19/11…$15.00
https://doi.org/10.1145/3357223.3362725

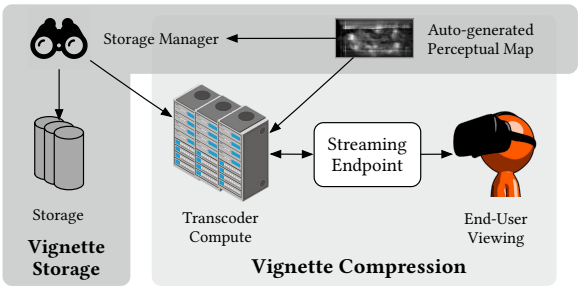

Figure 1: High-level architecture of Vignette. Vignette provides two features: Vignette Compression, a perceptual compression algorithm, and Vignette Storage, a storage manager for perceptually compressed videos. Integrating perceptual information with the storage manager reduces network bandwidth and storage costs.

Vignette's smaller video sizes. The system uses low-overhead metadata, can be easily integrated into existing media storage structures, and remains transparent to standard video applications.

Vignette is *not* a new standalone codec or compression standard. Instead, it extends existing, modern codecs to take advantage of the untapped perceptual compression potential of video content, especially high-resolution video served in VR and entertainment settings. As a result, off-the-shelf software and hardware accelerators can decompress Vignette's perceptually compressed videos with no modifications. We implement Vignette as an extension to LightDB [25], a database management system for video. Our prototype of Vignette demonstrates cost savings to cloud video providers and power savings during mobile video playback.

This paper makes the following contributions:

(1) **Systems support for perceptual video compression.** We propose Vignette, a system for producing and managing perceptually compressed video data. Vignette videos are 80–95% smaller than standard videos, consume 50% less power during playback, and demonstrate little perceived quality loss.
(2) **A forward-compatible perceptual encoding pipeline.** Vignette leverages existing features of modern video codecs to implement perceptual compression, and can be deployed in any video processing system that supports such codecs, such as HEVC or AV1.
(3) **Custom storage for perceptual data.** Vignette's storage manager efficiently stores and manages perceptually compressed videos and is integrated in a modern video processing database system. Vignette Storage supports both a heuristic-guided search for fast perceptual compression and an exhaustive mode to compute an optimal saliency-based compression configuration.

To our knowledge, this is the first work to consider storage management of perceptually-compressed video information. Using predicted saliency as a motivating perceptual cue, we evaluate the limits of perceptual compression in a video storage system with a collection of modern and high-resolution video datasets. Vignette's compression scheme uses a neural network trained to predict content saliency and an off-the-shelf HEVC video encoder to reduce bitrate requirements by 80–95%. Our results show that Vignette can reduce whole-system power dissipation by 50% on a mobile phone during video playback. Quantitative evaluation and user study results validate that these bitrate and power savings come with little perceived loss in video quality.

2 BACKGROUND: PERCEPTUAL COMPRESSION USING SALIENCY MAPS

Saliency is a widely-utilized measure of the perceptual importance of visual information. Saliency data encodes the perceptual importance of information in a video, such as foreground and background or primary and secondary objects. Video codecs already use some perceptual information, like motion and luminance, to improve compression performance [58], but new modes of video viewing (such as with a VR headset) introduce the opportunity to integrate richer cues from the human visual system [36]. In this paper, we use saliency as an example of one such perceptual cue to demonstrate the potential of perceptual compression. This section provides background on saliency, compares methods for generating and encoding saliency information, and introduces the machine learning technique Vignette uses to gather perceptual information about video data. This section also describes *tiles*, the codec feature Vignette uses to compress videos with saliency information.

2.1 Saliency Maps and Detection Algorithms

Saliency-detection algorithms visually highlight potential regions or objects of significance in an image. A saliency map captures likelihood of visual attention in the form of a heatmap, where the map's values correspond to the salience of pixels in the input. In this paper, we visualize saliency prediction maps as grayscale video frames or heatmaps for clarity.

In the past, algorithms could not predict saliency accurately without detailed per-video annotation, such as hand annotation or eye gaze logs. Moreover, the low latency and poor spatial resolution of eye-tracking devices prevented effective deployment of eye-tracker-based saliency prediction [6]. VR headsets, however, allow for efficient deployment of eye tracking, and they have motivated improvements in the performance and accuracy of eye trackers [61]. Recent work in machine learning has produced accurate saliency prediction models using neural networks trained on eye tracker data that mimic the human visual system at levels rivaling human prediction [9], motivating their use for saliency prediction in this work.

To generate saliency maps for this paper, we used the neural network model MLNet [13] running on the machine learning platform Keras with Theano [10, 60]. MLNet is a state-of-the-art saliency prediction neural network, and, when using the publicly-available weights trained on the SALICON [29] dataset, achieves 94% accuracy on the MIT300 saliency benchmark [7]. While the strong performance of MLNet motivates its use in the design of Vignette, our design allows for the replacement of MLNet with any other preferred saliency prediction method, as lower cost or higher accuracy systems are developed.

2.2 Systems Support for Perceptual Video Compression

Prior work investigated many techniques for including predicted saliency information in video compression, but these techniques required significant changes to video codecs. For instance, some maintain full-resolution, per-frame saliency maps to use as additional input [45], while others compute saliency prediction on-the-fly at high computational cost [22] or solve complex optimization problems to allocate video bits [40].

Rapid advances in both deep learning and video compression, however, resulted in these integrated prediction-and-compression codecs being quickly outmoded by the quality of both standard video compression and saliency prediction techniques. Most critically for codecs, the saliency-enabled encoders lacked many latency and quality optimizations of more recent codec releases and did not guarantee functionality on already-existing hardware accelerators on GPUs and mobile devices.

This paper takes a different approach, proposing a system that supports software extensions for perceptual compression without modifying the video codec. Instead of designing a new codec, we propose shifting the burden from a single codec to the data management infrastructure, where decisions about hardware resources, encoding optimization, and metadata management already occur.

2.3 Tiled Video Encoding

Vignette uses tiles to implement perceptual compression. Tiling a video divides a single video stream into independent regions that are encoded as separate decodable streams [49]. Encoders can code tiles at separate qualities or bitrates, and decoders can decode tiles in parallel. Tiles are simple to express using standard encoding libraries, like `FFmpeg` [5] and are supported by many video codecs. Restricting our implementation to native tiling features introduces some loss of detail compared to designing a custom encoder. Standard encoders only support rectangular tiles and cannot leverage motion across tiles during encoding process. Using only native features, however, guarantees that our compression scheme is compatible with any modern codec that implements tiling, like HEVC [58] or AV1 [18]. As video standards and codec efficiency improve, using general codec features to perform encoding and manage storage ensures that perceptual information remains useful.

3 VIGNETTE SYSTEM OVERVIEW

We designed Vignette to be easily deployed in existing video storage systems and transparent to video applications that do not require perceptual information. Figure 1 shows how Vignette can be deployed on a video storage system, with Vignette Compression used during the transcoding pipeline and Vignette Storage managing the integration of perceptual information with video data.

3.1 Vignette Compression

Vignette Compression uses native features found in modern video codecs. Our implementation of Vignette Compression produces videos that work out-of-the-box with any system that supports HEVC [58], including hardware accelerators. Vignette Compression perceptually compresses videos by enumerating configurations of video tiles and saliency-quality correspondences to maximize quality while minimizing video size. The algorithm has three high-level steps: generate a perceptual data map (e.g., saliency prediction map) for a given video file (§4.1), determine the optimal number of rows and columns, or a "tile configuration", to spatially partition the video into (§4.2), and select a mapping of saliency values to encoder qualities for each tile (§4.3).

3.2 Vignette Storage

Vignette Storage manages perceptual information as simple metadata embedded within videos or maintained in the storage system. This reduces storage complexity for data management and ensures Vignette data is transparent to saliency-unaware video applications such as VLC or Optasia [44]. The storage manager supports the following features: low-overhead perceptual metadata transmitted alongside video content, without impeding the functionality of applications that choose not to use it (§5.2), storage management policies to trigger one-time perceptual compression during "open loop" mode, and a heuristic-based search for faster perceptual compression (§5.4).

4 VIGNETTE PERCEPTUAL COMPRESSION DESIGN

Vignette Compression uses off-the-shelf video codec features to encode perceptual information and improve coding efficiency. Our technique takes a video as input, generates a per-frame perceptual map for the video, and aggregates the per-frame maps into a single video saliency prediction map.[1] Vignette Compression then transcodes the input video with a tiled encoding, where the quality of each tile corresponds to the saliency of the same tile in the video's saliency prediction map. It uses only the native features of the HEVC [58] codec to ensure compatibility with other video libraries. Whenever possible, it overestimates saliency to minimize the potential of degrading video quality in areas of interest.

4.1 Automatically Generating Saliency Maps

Vignette Compression uses MLNet [13] to automatically generate a corresponding saliency map for a video input. Figure 2 shows the saliency map generated for a video frame and how the generated maps capture the visual importance of a given video frame. The process requires decoding the video and processing each frame through the neural network to produce output saliency maps. Vignette Compression accumulates the per-frame saliency maps into a single map by collecting the maximum saliency for each pixel in the frame across the video file. These aggregated saliency values produce a single saliency map of importance across the video. This method uses more compute time than only generating saliency maps for keyframes or at a fixed timestep, but it more generously accommodates motion and viewpoint changes during a scene.

One concern for these aggregate heatmaps is that pixels may become "saturated". A "saturated" pixel contains its maximum value (255), after which any additional saliency information would not register during aggregation. Computing aggregate saliency maps for long videos would potentially result in many "oversaturated"

[1] For clarity, we will use saliency prediction map and saliency map interchangeably in the remainder of the paper, acknowledging that the maps used in discussion visualize *predicted* saliency and not actual human-annotated saliency.

pixels from scene changes and motion. But video storage systems slice videos into short segments (10-20 seconds) for coding efficiency; as a result, these short-duration aggregate saliency maps can be collected without oversaturating the saliency heatmap.

In comparison to a single video frame, Vignette's aggregated video saliency map can indicate many more salient pixels, especially for videos that have fast-moving salient objects across frames. We considered more motion-tolerant metrics like moving average, but found that, for the domain of video distribution platforms, using the most generous metric of maximum saliency provided the best quality guarantee. In this case, using aggregate video saliency maps with maximum saliency functions as a "worst-case" estimate of salient regions.

4.2 Leveraging Saliency With Tiled Encodings

Once Vignette Compression produces a saliency map for a video, it can perceptually encode videos with the tiling feature in HEVC [58]. To produce saliency-based tiled video encoding, Vignette divides a video segment spatially into tiles and then map each tile to a quality setting. The saliency map's value at each tile determines the tile's quality setting. For simplicity and generality, the tiling patterns used are rectangular tiles with uniform width and height across the video frame. Vignette uses the same tile configuration throughout the entire 10-20 second video segment for coding simplicity. Intuitively, larger tiles have better compression performance, but would allow for less saliency levels to be encoded in the video. Vignette selects the size and number of tiles in a tiling configuration based on either an exhaustive search of all tile configurations or a heuristic-guided search, described in §5.4.

While tiling is simple and provides coding benefits, a given tile configuration can incur overheads from introducing suboptimal encoding boundaries. Tiles are self-contained video units that can be decoded separately. They cannot compress information beyond per-tile boundaries. As a result, information that may be efficiently coded using partial frames in a standard encoding must be repeated if it appears in multiple tiles. A poor tile configuration produces larger videos than a standard encoding pass with no tiling, especially for fast-moving scenes. We investigated this compression inefficiency by encoding our test videos at various tiling configurations with no change in quality (e.g. lossless encoding). We found that the inclusion of tiling with no change in quality incurred ~6-15% overhead, depending on motion in the video sequence and number of tiles used.

Vignette minimizes the penalty of adding tile boundaries in areas that would benefit from being encoded together by exhaustively enumerating all tile configurations. Vignette evaluates across all row-column pairs a video frame allows to find the per-video best tiling configuration. The HEVC standard constrains the minimum size of row and column tiles, which restricts the row-column tile configurations allowed. In practice, we enumerate tile configurations ranging from 2×2 to 10×10, compress the tiles according to their saliency values, and measure the resulting bitrate and video quality achieved. This exhaustive enumeration takes about 30 minutes per 15-second video to find the best tile configuration with our experimental setup.

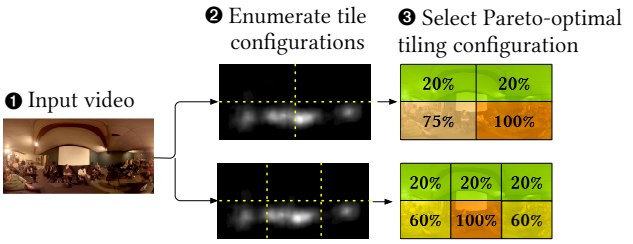

Figure 2: Overview of Vignette Compression algorithm.

4.3 Mapping Saliency to Video Quality

Each HEVC tile is encoded at a single 'quality' or bitrate setting throughout the video stream, requiring Vignette Compression to select per-tile encoding qualities. Vignette deconstructs saliency maps into per-tile parameters by mapping the highest encoding quality to the maximum saliency value in the tile's saliency map. Selecting the video encoding quality that corresponds to a tile's saliency value is less straightforward. To do so, Vignette must determine both how to express video quality during encoding and how saliency should correspond with that quality metric.

HEVC exposes different modes of controlling quality and bitrate, such as constant bitrate or constant rate factor, with varying levels of effort and efficiency. For evaluation simplicity, Vignette uses a perceptually-controlled version of a target bitrate, where the target bitrate either corresponds to the bitrate of the original video or is specified by the API call. The highest-saliency tiles in the video are assigned the target bitrate, and tiles with lower saliency are assigned lower bitrates, with a minimum bitrate of 10% the original video bitrate. As shown in Figure 2, Vignette Compression encodes a 0-255 saliency map as discrete bitrates corresponding linearly from the minimum to the target bitrate or quality. Because Vignette supports standard codec features, target bitrate could be replaced with a codec's quality control, i.e. constant rate factor, as well.

5 VIGNETTE STORAGE SYSTEM DESIGN

We now describe Vignette's storage manager for maintaining perceptual video information. Vignette Storage uses low overhead metadata to encode perceptual data and a heuristic-guided search to reduce the compute load of generating perceptual transcodings. Vignette Storage's metadata representation reduces full-resolution frames to a small number of bytes, and its heuristic search algorithm reduces the time taken to find an optimal tile configuration by ~30× in our experiments.

5.1 Overview of Vignette Storage

Vignette Storage exposes perceptual video compression to applications by providing three features: (1) transparent perceptual metadata, (2) simple storage management policies, and (3) a search algorithm that reduces transcoding cost. Vignette embeds perceptual metadata as a side channel within the video container. Standard video containers (i.e., mp4) encapsulate saliency information along with video content, so that applications with and without perceptual support can decode Vignette videos. A 360° video player, for example, can initialize videos to be oriented in the direction of a

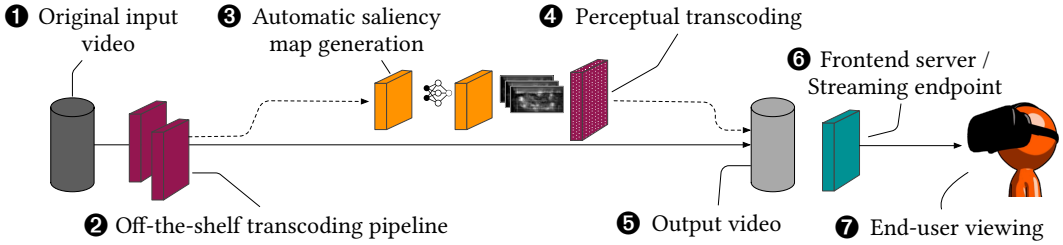

Figure 3: Vignette supports conventionally video transcoding pipelines as well as automatically generating saliency maps for perceptually-aware video video transcoding.

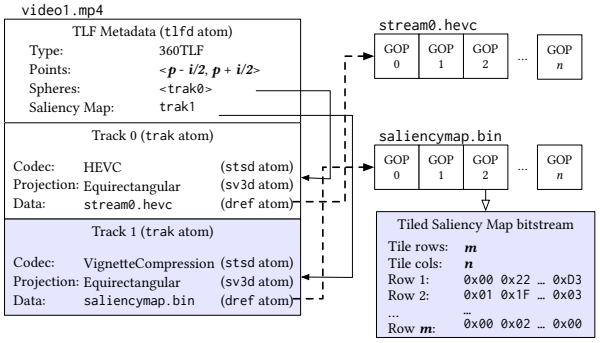

Figure 4: Physical layout of video metadata in LightDB. Vignette-specific features are highlighted.

high-saliency region it decodes from Vignette metadata, but the videos can also be played traditionally in a standard video player like VLC.

Vignette Storage operates like similar large video management services [28, 43, 47]. Upon upload, it chunks videos into segments, typically 6-12 seconds in length. Each video segment consists of one keyframe and an ensuing set of predicted frames. Vignette Storage can perform perceptual compression on a per-video basis, or across the video library when a specified condition is met (e.g., low storage capacity, or video popularity decreasing beneath a threshold).

5.2 Saliency Map Metadata

Video storage systems maintain containers of compressed video data that store relevant video features in metadata. Vignette Storage adopts this approach, and injects a small amount (~100 bytes) of saliency metadata inside each video container. This map is encoded as a bitstring that includes fields for the number of rows and columns used for tiled saliency and the saliency weights for each tile. These bitstrings typically range in size from 8–100 bytes. Figure 4 shows how this metadata is included as a saliency `trak`, similar to other metadata atoms in a video container.

5.3 Vignette Storage API

The Vignette Storage API is shown in Figure 3. Table 1 shows the programming interface for Vignette, which includes three perception-specific operations: `vignette_transcode()`, `vignette_squeeze()`, and `vignette_update()`. Each API operation ingests a video and some required parameters and outputs a video with any generated perceptual metadata encapsulated in the video container.

The Vignette API is linked into LightDB as a shared library. System developers using Vignette Storage to manage video data can write storage policies or preconditions to execute Vignette Storage functions for a specific video or collection of videos. For instance, a social media service could apply perceptual compression as videos decrease in popularity to reduce storage capacity. A VR video-on-demand service that ingested eye tracking information could apply perceptual compression as new perceptual information is collected for certain videos.

5.3.1 Transcode Functions. Transcode operations express the most basic Vignette Storage function, video transcoding. When a new video is uploaded to the storage system, the storage manager triggers the general-purpose `transcode()` function to transcode the video to any specified bitrates and formats for content delivery. This function takes as input a video and target quality parameter, expressed either by CRF or bitrate, and produces a regularly transcoded video.

The `vignette_transcode()` function is the default saliency-based API call. It takes as input a video and an optional quality or bitrate target, and produces both a video and its corresponding generated saliency metadata. When `vignette_transcode` is triggered, Vignette Storage generates new saliency maps, and then compresses the video according to the target quality expressed.

Vignette Storage's transcode functions use similar signatures, letting the system easily switch between regular and perceptual compression when storage system pressure changes. Including saliency information as a metadata stream included in the video file container makes it transparent to saliency-agnostic applications or commands like `mediainfo` or `ffprobe`.

5.3.2 Quality Modulation Functions. As noted in §4.3, Vignette Compression maps saliency to quality levels for each tile. A call to `vignette_squeeze()` will re-compress a video using a specified, reduced bitrate or quality threshold. It takes in a video, target bitrate, and saliency mapping and produces the newly compressed video. This function only executes transcoding and compression with pre-generated saliency metadata, but does not update or generate new saliency metadata. The `vignette_squeeze()` function will recompress videos from a higher quality mapping to a lower one,

Table 1: Vignette API

Function	Compression Type	Data required
`transcode`	General	`<IN video, IN CRF/target bitrate, OUT video>`
`vignette_transcode`	Perceptual	`<IN video, (IN CRF/target bitrate,) OUT video, OUT saliency metadata>`
`vignette_squeeze`	Perceptual	`<IN video, IN CRF/target bitrate, OUT video>`
`vignette_update`	Perceptual	`<IN video, IN fixation map, OUT video, OUT saliency metadata>`

but it will not transcode low-quality videos to a higher-quality mapping to avoid encoding artifacts. For example, a call to `vignette_squeeze(input.mp4,100k)` transcodes a video previously encoded with saliency at a higher bitrate to a maximum of 100kbps in the most salient regions. By leveraging the saliency metadata attached to videos in Vignette Storage, `vignette_squeeze()` can avoid re-encoding tiles that already are lower than the threshold bitrate. A system can invoke `vignette_squeeze()` before video data is sent to smaller cache or in preparation for distribution to devices with smaller displays.

5.3.3 Functions for Updating Perceptual Maps. Vignette Storage also supports updating saliency map with new information, such as from eye tracking devices. To invoke this mode, Vignette Storage uses the `vignette_update()` function to ingest and re-process videos with new perceptual information. A 2-dimensional eye tracker map can be used in the same way as the saliency map input used in Vignette Compression, or it could be aggregated with the existing saliency map metadata. Similar to how Vignette constructs per-video saliency maps, `vignette_update()` updates the video's saliency map with eye tracker information by executing a weighted average of the original map and the input eye tracker map. The update function takes in a fixation map and generates a new metadata bitstream of saliency information that is attached to the video container. Should a client want to re-encode a video based on the updated saliency metadata, it could call `vignette_squeeze()` after a `vignette_update()` call.

5.4 Heuristic Search for Tiling

Most of Vignette's computation overhead comes from the exhaustive search over tile configurations for a given video. This exhaustive search is typically performed once, upon video upload, but consumes significant processing time. Vignette Storage contributes a lower cost search algorithm that achieves near-optimal results with a ~30× performance improvement, for situations where fast saliency-based transcoding is required. Vignette Storage can switch between the exhaustive search for optimal results or heuristic-guided search for faster processing.

Vignette's search technique uses motion vector information from encoded video streams to estimate the size of video tiles. It enumerates tile configurations that group regions of high motion together, and selects a configuration minimizing the difference in motion vector values across tiles. Vignette computes this difference by evaluating the average standard deviation of motion vector values within tiles and comparing to the best result seen. This heuristic approximates the observation that high-motion areas should not be divided across multiple tiles. We define the algorithm in more detail in Algorithm 1.

Algorithm 1 Heuristic-based search for selecting a near-optimal saliency tile configuration

procedure GenTileConfig(v) ▷Generate tiling for video v
  bestStdDev$\leftarrow \infty$ ▷Initialize values
  $lastStdDev \leftarrow \infty$
  $bestConfig \leftarrow$ ***null***
  $\{maxRows, maxCols\} \leftarrow \{10, 10\}$
  $motionVectors \leftarrow$ MPEGFlow(v) ▷Extract motion vectors
  for all $\{rows, cols\} \leftarrow \{1, 1\}, \{maxRows, maxCols\}$ **do**
    $avgStdDev \leftarrow$ avgStdDev($motionVectors, rows, cols$)
    if $avgStdDev \leq bestStdDev$ **then**
      $bestStdDev \leftarrow avgStdDev$
      $bestConfig \leftarrow \{rows, cols\}$
    end if
    $lastStdDev \leftarrow avgStdDev$
    $\Delta avgStdDev(motionVectors) \leftarrow lastStdDev - avgStdDev$
    if $\Delta avgStdDev(motionVectors)) \geq .1$ **then**
      break
        ▷Break if $\Delta avgStdDev$ is below difference threshold
    end if
  end for
  return $bestConfig$ ▷Return best tile configuration
end procedure

The algorithm extracts motion vector information from encoded videos using MPEGflow [34] and requires one transcoding pass. Similar to the tile configuration search from §4.2, this heuristic search exhaustively evaluates tile configurations of the motion vectors. The search evaluates the motion encapsulated by tiles under a configuration and chooses the configuration with the minimum deviation of motion vectors in each tile. This heuristic approximates the result of exhaustive encoding by leveraging the observation good tile configurations are able to encapsulate redundant motion or frequency information with a single tile, rather than replicate it across tiles. Compared with an exhaustive search, which transcodes a video hundreds of times to empirically produce the optimal tile configuration, Vignette Storage's algorithm produces a result ~30× faster than the exhaustive method and within 1 decibel (dB) of the best-PSNR result when executed over the videos used in our evaluation.

6 METHODOLOGY

We next describe the datasets, quality metrics, and technical setup for our evaluation. We benchmarked across a range of video workloads (§6.2) and considered video quality metrics (§6.3) to holistically evaluate compression, quality, and performance.

(a) Input video frame from the Netflix dataset [39].

(b) Saliency map produced by MLNet [13], overlaid on input.

(c) Perceptually-compressed Vignette video, 85% smaller at iso-quality.

Figure 5: Example video still, neural network-generated saliency map, and output Vignette video.

6.1 Implementation

We implemented Vignette by extending LightDB [25], a database management system for VR videos. LightDB lets developers declaratively express queries over large-scale video and uses a rule-based optimizer to maximize performance. Developers can easily express HEVC-based saliency encoding in LightDB's query language by combining its `Encode`, `Partition`, and `Subquery` operators:

```
Decode("rtp://...")
  >> Partition(Time,1,Theta,π/rows,Phi,2π/cols)
  >> Subquery([](auto& partition) {
       return Encode(partition, saliency_mapping(
     partition) })
  >> Store("output");
```

In this example, `Partition` divides the input video into tiles, `Encode` transcodes each tile with the corresponding `saliency_mapping` value as an argument, and `Subquery` executes the given operation over all the partitioned tiles. We also wrote our object recognition queries for §7.2 in LightDB to simulate video analytics workloads.

In our experiments, we compared Vignette against the HEVC encoding implementations included with `FFmpeg`. We configured `FFmpeg` with support for software-based coding and NVENCODE [52] GPU-based encoding of HEVC video, as both are supported by large-scale video services and devices [14].

Some datasets provided overencoded videos, or, reference videos encoded at an very high bitrate that could be re-encoded to identical quality at a lower bitrate. To ensure against overencoding, we transcoded each video using CPU-based HEVC encoder and vary the rate factor, selecting the highest PSNR-quality result as our baseline for evaluation. We ran Vignette Compression on top of `FFmpeg` version `n4.1-dev`, and use the GPU-based NVENC HEVC encoder for tiled encoding. Unless otherwise specified, we targeted a constrained bitrate using maximum bitrate mode (VBV); while VBV does not provide the best-quality archival results, it is commonly used for entertainment or livestreaming due to its combination of speed and quality.

We performed all experiments on a single-node server running Ubuntu 16.04 and containing an Intel i7-6800K processor (3.4 Ghz, 6 cores, 15 MB cache), 32 GB DDR4 RAM at 2133 MHz, a 256 GB SSD drive (ext4 file system), and a Nvidia P5000 GPU with two discrete NVENCODE chipsets.

6.2 Video Datasets

We used a collection of video datasets, listed in Table 2, to evaluate the impact of our techniques across different classes of video.

Table 2: Video datasets used to characterize Vignette.

Type	Benchmark	Description	Bitrate (Mbps)	Size (MB)
Standard	vbench [43]	YouTube dataset	0.53–470	757
	Netflix [39]	Netflix dataset	52–267	1123
VR	VR-360 [42]	4K-360 dataset	10–21	1400
	Blender [21]	UHD / 3D movies	10–147	6817

Standard video formats and emerging VR formats comprise our evaluation datasets. The former include representative workloads from Netflix [39] and YouTube [43]. The VR and emerging video datasets highlight demands of ultra high-definition (UHD) formats such as 360° video [42] and the Blender stereoscopic and UHD open source films [21]. To construct a representative sampling of Blender video segments, we partitioned the movies in the Blender dataset ("Elephants Dream", "Big Buck Bunny", "Sintel", and "Tears of Steel") into 12-second segments, and selected five segments that covered the range of entropy rates present in each film.

In this collection of datasets, we found that the `vbench` "desktop" video, a 5-second computer screencast recording, performed poorly during all compression evaluations because of its low entropy and content style, so we excluded this outlier video from our evaluation results. We discuss this style of video in relation to Vignette further in §8. We also replaced Netflix's single "Big Buck Bunny" video segment with the same video content from Blender's stereoscopic, 4K, 60 frames-per-second version of the video.

6.3 Quantitative Quality Metrics

We measured video encoding quality using two quality metrics, peak signal-to-noise ratio (PSNR) and eye-weighted PSNR (EWPSNR). PSNR reports the ratio of maximum to actual error per-pixel, in decibels (dB), by computing the per-pixel mean squared error and comparing it to the maximum per-pixel error. PSNR is popular for video encoding research, but researchers acknowledge that it fails to capture some obvious perceptual artifacts [39]. Acceptable PSNR values fall between 30 and 50 dB, with values above 50 dB considered to be lossless [43]. For saliency prediction evaluations, researchers developed eye-weighted PSNR to more accurately represent human perception [40]. EWPSNR prioritizes errors perceived by the human visual system rather than evaluating PSNR uniformly across a video frame. We computed EWPSNR using the per-video saliency maps described in §4 as ground truth. While calculating

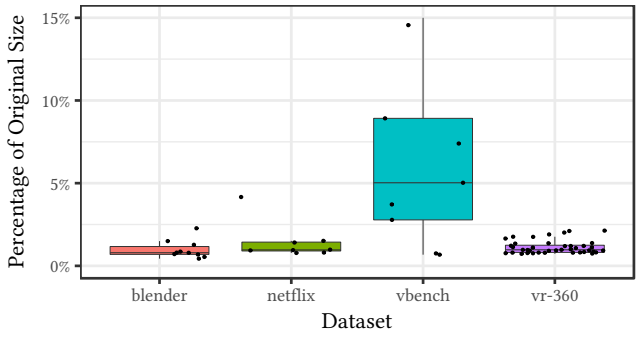

Figure 6: Aggregate storage savings by dataset. Vignette Compression reduces videos to 1–15% of their original size.

EWPSNR in this way does not faithfully measure "true" eye fixations, it still can assess "correctness" of our compression pipeline and thus the efficacy of Vignette as a system for leveraging saliency.

7 EVALUATION

We designed our evaluation to answer the following questions:

(1) **Storage:** What storage and bandwidth savings does Vignette provide? How do tile configurations affect compression gains and quality? How does Vignette compare against traditional saliency-based encoders?
(2) **Quality of Service:** How does Vignette's compression technique affect quality of service (QoS) of video services like video streaming (perceptual quality user study) or machine learning (speed, accuracy)?
(3) **Compute Overhead:** What is the computational overhead of Vignette's compression algorithm and storage manager?
(4) **Data Center & Mobile Cost:** How do Vignette's storage and network bandwidth savings impact video storage system and mobile viewing costs?

7.1 Storage and Bandwidth Savings

To evaluate the storage and bandwidth benefits of Vignette, we applied Vignette Compression to the corpus of videos described in §6. We transcoded our video library at iso-bitrate in salient regions and decreased bitrate linearly with saliency to a minimum 10% target bitrate in the lowest saliency tiles, as illustrated in Figure 2. In these experiments, we examine how our transcoding performs across a range of resolutions and workloads, as is expected in a video storage system.

7.1.1 Impact of Tiling on Compression and Quality. We first examined the impact of tiling on compression benefits using a fixed saliency map. We used an exhaustive tile configuration search and evaluated all tile sizes to identify an optimal number of tiles for each video. Our goal was to determine whether the number or shape of video tiling affects resulting size. The smallest tile size we evaluated were 64 pixels in breadth, but most videos performed best with tiles having a breadth of 300–400 pixels. We observed that, given a fixed saliency map, optimal tile configurations to maximize storage savings and quality varied based on entropy and video content. We found the optimal tile configuration varies from four tiles to forty and per-video tile configuration is an important component of tile-based compression. Some videos benefited from many small tiles, while others performed best with fewer large tiles.

7.1.2 Overall Compression, Bandwidth, Quality. We next explored peak compression, bandwidth, and quality savings by applying Vignette to our video corpus and measuring compression and quality. We used the results of our exhaustive tile search to identify the best compression-quality configurations for each video. Figure 6 shows aggregate storage savings, partitioned by dataset. Overall, we find that Vignette Compression produces videos that are 1–15% of the original size when maintaining the original bitrate in salient regions. These compression savings include the fixed overhead of perceptual metadata, which is <100 B for all videos. Datasets with higher video resolutions (Blender, VR-360) demonstrated the highest compression savings. The vbench dataset, which is algorithmically chosen to have a wide variance in resolution and entropy, exhibits a commensurately large variance in storage reduction. Of the videos with the lowest storage reduction, we find that each tends to have low entropy, large text, or other 2D graphics that are already efficiently encoded.

Figure 7a shows the average reduction in bitrate and resulting quality, measured in PSNR and EWPSNR. Our results show that EWPSNR results are near-lossless for each benchmark dataset, while the PSNR values—which do not take the human visual processing system into account—nonetheless remain acceptable for viewing. Figure 5 highlights a Vignette video frame from the Netflix dataset, with an output PSNR of 36 dB and EWPSNR of 48 dB. Overall, the results indicate that Vignette Compression provides acceptable quality for its compression benefit.

7.1.3 Comparison with Saliency-based Encoder. Vignette differs from traditional encoder-based solutions for incorporating saliency information into the video compression pipeline. To evaluate the performance of Vignette Compression relative to a custom saliency-based encoder, we use the `x264_saliency_mod` fork of H.264 from Lyudvichenko et al. [45]. Although the rest of our evaluation uses HEVC as the baseline codec, no existing saliency-based encoders use HEVC, so we instead compare with Vignette Compression using H.264, the same encoder Lyudvichenko et al.modified. Because Vignette can extensibly be used with any codec, it is straightforward to switch Vignette Compression to use H.264 instead of HEVC, as we did for other experiments. For this experiment, we transcoded benchmark videos to a target bitrate of 20% the original bitrate using (1) H.264, (2) a saliency-based encoder, and (3) Vignette Compression using H.264 as a base codec. As in §7.1.2, we measured achieved bitrate reduction over the baseline videos and the resulting PSNR and EWPSNR.

Figure 7b details the results. As expected, when tasked with reducing video bitrate to 20%, the standard H.264 encoder generally meets that bitrate reduction. Overall, the saliency-based encoder and Vignette Compression perform favorably in bitrate reduction compared to H.264: for `vbench` and `Netflix`, Vignette Compression videos are smallest, but the custom encoder outperforms Vignette Compression for the higher-resolution `VR-360` and `Blender` benchmarks. Examining quality, however, the custom encoder maintains a higher PSNR and EWPSNR than Vignette Compression across

(a) Vignette Compression with HEVC.

Benchmark	Bitrate Reduction	PSNR (dB)	Eye-weighted PNSR (dB)
vbench	85.6 %	39	51
Netflix	98.6	34	45
VR-360	98.8	36	45
Blender	98.2	39	49

(b) Saliency-based compression with H.264 targeting 20% bitrate reduction.

	Bitrate Reduction			PSNR (dB)			EWPSNR (dB)		
Benchmark	x264	[44]	Vignette	x264	[44]	Vignette	x264	[44]	Vignette
vbench	23.3%	27.0%	7.6%	30	50	32	40	53	40
Netflix	32.0	12.4	5.1	28	34	32	40	47	43
VR-360	28.9	2.4	10.7	28	35	37	38	49	48
Blender	24.2	3.7	8.18	29	36	39	39	46	49

Figure 7: Bitrate reduction and quantitative quality metrics comparing (a) HEVC and Vignette Compression using HEVC, and (b) H.264, a custom saliency-based encoder extending H.264, and Vignette Compression with H.264. For PSNR and EWPSNR, > 30 dB is acceptable for viewing, 50 dB+ is lossless.

all benchmarks. This quality gain can be attributed to the custom encoder's ability to finely tune quality on a per-macroblock scale.

We also observe saliency-based encoders also bear the additional burden of full-size saliency information; even the test saliency image provided by the saliency-based encoder, a single saliency map replicated for the length of the video, is 56KB. Vignette Storage's metadata, on the other hand, can be reduced to 100B bytestreams.

7.2 Quality of Service

To understand the impact of perceptual compression on common video system workloads, we evaluated the quality of service (QoS) delivered by Vignette for two applications: entertainment streaming with a user study and evaluation of a video analytics application that performs object recognition. These applications optimize for different QoS metrics: perceptual quality for entertainment video; throughput and accuracy for object recognition.

7.2.1 Perceptual Quality User Study. We we received IRB approval to run a user study to quantify viewer perception of our saliency-based compression. The study presented users with two versions of the same video: one encoded with HEVC at 20 Mbps (as our baseline), the other with Vignette Compression. The Vignette Compression videos were randomly chosen to be either 1 Mbps, 5 Mbps, 10 Mbps, or 20 Mbps. For each video target bandwidth, we predicted saliency and encoded the most-likely salient tiles of the video to the target bitrate (1,5,10,20Mbps) and other tiles at lower bitrates, as in earlier experiments. The study asked users their preference between the matched pairs for 12 videos. The goal was to discover if viewers prefer Vignette Compression to HEVC, and, if so, if those preferences are more or less pronounced at different bitrate levels for Vignette.

The 12 videos included three videos from each dataset, selected to cover a range of entropy levels, and all videos' original bitrate exceeded 20Mbps, except two from vbench. Each video was encoded at a target bitrate (1Mbps, 5Mbps, 10Mbps, or 20Mbps), and the questionnaire randomly selected which bitrate to serve. We distributed the questionnaire as a web survey and ensured videos played in all browsers by losslessly re-encoding to H.264. Users viewed the study videos in the web browser on their personal devices; devices used ranged from phones on WiFi to laptops and wired desktops. To eliminate concerns of buffering or network quality, the study website pre-loaded all videos before allowing playback.

We recruited 35 naive participants aged 20–62 (51% women, 49% men) from a college campus to participate in the study. Figure 8 shows the results averaged across subjects and videos. When Vignette videos are encoded at 1 Mbps in the most salient regions, 72% users preferred the HEVC baseline. However, for Vignette videos encoded at 5, 10, and 20 Mbps, users either could not tell the difference between HEVC and Vignette, or preferred Vignette videos 60%, 79%, and 81% of the time, respectively. This suggests that video systems can deliver Vignette-encoded videos at 50-75% lower bitrate with little perceived impact.

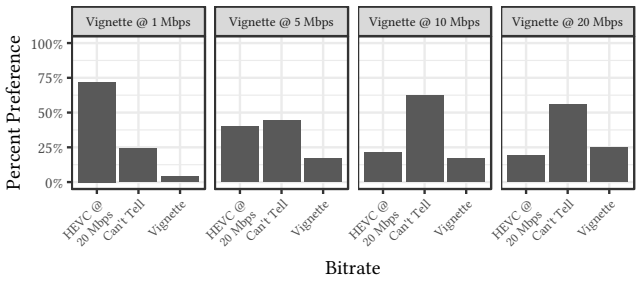

Figure 8: Results of perceived quality preference user study, averaged across participants and videos by bitrate.

Table 3: Vignette Speedup and Accuracy Compared to HEVC Baseline on YOLO Object Recognition.

Decode Speedup	Total Speedup (Decode + YOLO)	Average Accuracy
34.6% ± 14.3%	2.6% ± 2.2%	84% ± 14%

7.2.2 Object classification. Video storage and processing systems often perform analytics and machine learning tasks on their video libraries at scale [53, 56, 62]. To evaluate any performance degradation in latency or quality from using Vignette Compression, we profile Vignette while running YOLO [54], a popular fast object recognition algorithm used in recent video analytics systems [27, 31–33]. We compare against baseline HEVC-encoded videos to evaluate if Vignette incurs any additional cost in a video processing setting, and run YOLO inference on the GPU. Table 3 shows that using Vignette-compressed videos provides some speedup when decoding videos for object recognition, but this benefit is overshadowed by the fixed cost of running YOLO.

Examining accuracy, we find that Vignette videos maintain 84% accuracy on average, compared to the baseline HEVC videos. We find that accuracy on the YOLO task is lowest for the videos in the VR360 suite, and tends to correspond to the areas where the video is distorted from the equirectangular projection. Overall, we find saliency-compressed videos can provide slight benefits for video analytics latency, especially if video decoding is the system bottleneck. Future work, however should investigate how to optimize saliency-based compression for video analytics. For instance, Vignette Compression could provide functionality to support multiple perceptual prediction algorithms, including models tuned for analytics.

7.3 Compute Overhead

Vignette Compression bears the additional processing overhead of executing a neural network to generate or update saliency maps. Vignette Storage can switch between the exhaustive or heuristic-based tile configuration search to uncover optimal tile configurations for a video. We benchmarked the latency of the combined saliency and transcoding pipeline in two modes: exhaustive, which generates saliency maps per frame and exhaustively evaluates tiling, and heuristic, which uses the heuristic search algorithm to select a tile configuration within 0.25 dB of the best-PSNR choice (§5.4).

Table 4 shows generating saliency maps with MLNet dominates computation time. This step, however, needs only to be performed once per video, and is off the critical path for video streaming workloads. Moreover, improving the performance of the saliency prediction (MLNet) would significantly reduce these overheads. The neural network used runs as unoptimized Theano code that could be improved by using an optimizing machine learning framework.

7.4 Analytical Model of Data Center and Mobile Costs

We use our evaluation results to model Vignette's system costs at scale for data center storage and end-user mobile power consumption. While these results are a first-order analysis, they suggest the potential benefit of deploying Vignette in the cloud.

7.4.1 Data center compute, storage, and network costs. Given the high compute cost of Vignette, we evaluate the break-even point for systems that store and deliver video content. We used Amazon Web Services (AWS) prices from July 2018 in the Northern California region to characterize costs.

We use a `c5.xlarge` instance's costs for compute, S3 for storage, and vary the number of videos transferred to the Internet as a proxy for video views. We assume a video library of 1 million videos that are 10 MB each, encoded at 100 different resolution-bitrate settings (as in [28, 35]) to produce ~500 TB of video data. We measured baseline compute cost to be a two-pass encoding for each video at $0.212 / sec and Vignette's transcode computation to be 5× a baseline transcode, averaged from transcode costs for videos across the datasets. Larger companies likely use Reserved or Spot Instance offerings, which provide better value for years-long reservation slots or non-immediate jobs; they are 36% and 73% cheaper, respectively. For storage, we measured costs to be $0.023 / GB on S3 and estimate Vignette-compressed videos would be 10% of the original videos (§7.1). Transferring data out from S3 costs $0.05 / GB; this cost is where Vignette achieves the majority of its savings.

Table 4: Mean processing time per video, evaluated over all videos in our datasets.

	Exhaustive		Heuristic	
Task	Time (s)	%	Time (s)	%
Generate saliency map	1633	49%	1633	95%
Compute tile configuration	1696	50	59	4
Saliency-based transcode	21	1	21	1
Total	3350		1713	

Figure 9 shows how different compute pricing models produce lower savings at small numbers of video library views, but that Vignette becomes cost-effective at large video viewing rates. For all pricing tiers, a system would need to service ~2 billion views across a million-video library before Vignette's compute overhead would be amortized across transmission and storage savings. This number is plausible for large video services; Facebook reported 8 billion daily views in 2016 [48]. Even with Vignette's substantial overhead, streaming services need just 2,000 views per video to break even: this could be 2 billion views across a million video library or 2,000 views of a single video.

7.4.2 Mobile Power Consumption. We explicitly designed Vignette to work with the HEVC standard so off-the-shelf software and hardware codecs could decompress Vignette videos. Vignette Compression's tiling strategy, however, makes video bitstream density highly non-uniform across the visual plane. This results in inefficiency for hardware architectures that decode variably-sized tiles in parallel. On the other hand, even such designs could achieve a higher overall power efficiency because of the reduced file sizes to decode and display. To investigate whether Vignette videos reduce or increase mobile power consumption, we profiled power consumption on a Google Pixel 2 phone during video playback of Vignette videos and standard HEVC-encoded videos.

We measured battery capacity on a Google Pixel 2 while playing our video library in a loop. The phone ran Android version 8.1.0 and kernel version 4.4.88-g3acf2d53921d, and MX Player 1.9.24 with ARMv7 NEON instructions enabled. Whenever possible, MX Player used hardware acceleration to render videos.[2] We disabled nonessential display and button backlights, as well as any configurable sensors or location monitors, to minimize extraneous power consumption. We logged battery statistics each minute using 3C Battery Monitor Widget v3.21.8. We conducted three trials, playing the 93-file video library in a loop until battery charge dissipated from 100% to 30%, for our HEVC baseline and Vignette videos.

Figure 10 shows our results. We found that Vignette video enabled 1.6× longer video playback time with the same power consumption. While hardware decoder implementations are typically proprietary, these results indicate that perceptual compression benefits mobile viewers, as well as cloud infrastructure. For video decoding ASICs where the hardware design is known [46, 64], Vignette's

[2] MX Player only supported decoding stereoscopic videos with the software decoder.

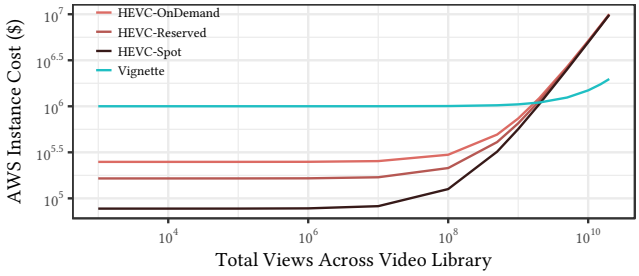

Figure 9: Estimated AWS costs for deploying Vignette versus traditional video transcoding. Vignette's additional compute cost is amortized after ~2 billion video views over a 1-million video library.

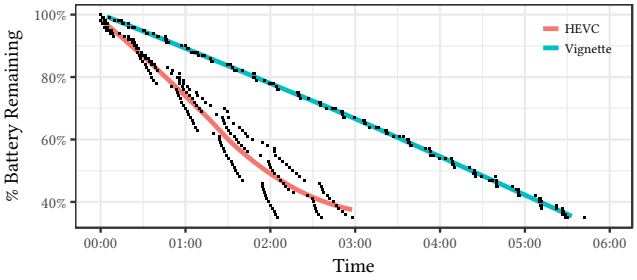

Figure 10: Time to dissipate a Google Pixel 2 phone battery from 100% to 30% when viewing HEVC and Vignette videos continuously. Vignette videos provide 1.67× longer video playback on mobile phones.

heuristic search algorithm could include power consumption of tiles as an optimization target to produce perceptually compressed videos for a "power-save" mode. We leave further optimization of Vignette on mobile devices, including network download and decoder optimization, to future work.

8 LIMITATIONS AND FUTURE WORK

This section considers limitations in our approach and evaluation.

8.1 Vignette Design Limitations

8.1.1 High compute overhead. Vignette's high one-time compression cost is its biggest drawback, but we consider our algorithm to be a reference implementation which will be improved upon in future work. The compression performance is hindered by the use of a highly accurate but slow neural network for saliency prediction, which does not yet use a GPU or any modern deep learning optimizations. Further, this expensive compression is run only once and is easily amortized across many views (§7.4). Future work could characterize the compute overhead of other saliency prediction techniques [36] or tailor existing deep prediction networks to existing cloud infrastructure [46] for improved performance.

8.1.2 Dependency on tiles. We argue Vignette's use of tiling features in video codecs is more flexible and forward-compatible than rewriting a codec for each new type of perceptual information. If, however, conventional codecs choose to integrate saliency or other perceptual information, the impact of a cloud storage system designed to support tile-based perceptual encoding will be smaller.

8.1.3 Integration with networking and other video system optimizations. We could further improve Vignette by building on other optimizations that work with off-the-shelf video standards. Notably, Vignette does not yet support video streaming using adaptive bitrate (ABR) algorithms. Future work could pair Vignette's saliency-based tiling with Fouladi et al.'s codesigned video transcode and network transport protocol could achieve better streaming quality [19, 20], or with VideoCoreCluster [41]'s energy-efficient cloud transcoding using low-cost transcoding ASICs. Vignette's heuristic search algorithm could include power and performance information from open-source video transcoding ASICs [46, 64] to target more power-efficient tiling configurations. At the physical storage layer, Jevdjic et al.'s approximate video storage framework, which maps video streams to different layers of error correction, could be coupled with Vignette's saliency mapping for more aggressive approximation of non-salient video regions [30]. Integrating Vignette with these systems optimizations could further improve power efficiency during playback, transcoding latency, or archival video storage durability.

8.1.4 Using Vignette in other storage systems. While we only evaluated Vignette using LightDB, but we crafted the policies, metadata extensions, and compression scheme to enable compatibility with pre-existing video storage or transcoding systems. For instance, an Amazon MediaConvert instance with a storage layer in Amazon EBS and Glacier can easily use the policies and metadata in §5 to implement Vignette, and Vignette can easily be used with other codecs, like the upcoming AV1.

8.1.5 Integration with other perceptual techniques. This paper described using predicted saliency to perform perceptual video compression. As mentioned in §2, other kinds of perceptual indicators could be leveraged to improve video compression. As new technologies to capture other perceptual cues become available, Vignette's techniques can be extended to encode multiple cues.

8.2 Evaluation Limitations

8.2.1 Comparison with ground truth. To compute EWPSNR, we use saliency maps generated by MLNet as ground truth. This evaluates Vignette quality compared to the saliency map, but not quality for real users. Evaluating on more precise ground truth, such as eye tracker positions from users, however, requires resolving some open questions. For instance, how should a perceptual compression-based video storage system manage saliency for multiple users? How does Vignette's overhead and compression benefit change when tuning compression for multiple users or kinds of devices (mobile, desktop, television)?

8.2.2 Saliency for screencasts and 2D graphics. We eliminated one outlier video, a screencast of a slideshow, because the saliency model performed poorly and provided little compression benefit. While our work targets reductions of storage size and network bandwidth for the large corpora of videos stored for social media,

entertainment, and video processing services, optimizing transmission of screencasts and other 2D graphics videos requires including information from different saliency models. Incorporating recent saliency models specifically designed for 2D visualizations [8] would likely resolve the issue.

9 RELATED WORK

9.1 Video Streaming and Storage Systems

The rise of video applications has driven renewed interest in processing and storage systems for video content, with recent work specializing for subdomains like social media, entertainment, and video streaming. Social media services distribute user-uploaded content from many types of video capture devices to many types of viewing devices, typically serving a small number of popular or livestreamed videos at high quality and low latency, as well as a long tail of less popular videos [16, 59]. These workloads motivated custom media storage infrastructure and fault-tolerant frameworks for video uploads at scale [3, 4, 28, 43, 50]. Entertainment video platforms like Netflix have smaller amounts of video data than social media services, but incur significantly more network traffic to distribute videos broadly. These services maintain user experience by transcoding videos for a range of heterogeneous devices and bitrate requirements, tailoring encode settings by title, streaming device, and video scene [1, 35, 47, 51].

Aside from entertainment, recent work in systems proposes tailoring machine learning pipelines for video analytics applications [27, 31–33, 63]. Specifically, they combine multiple classification algorithms to intelligently distribute computing effort to video frames that are determined to have interesting content, similar to semantic-based processing. BlazeIt [32], for instance, couples of simple, efficient classifiers with complex, precise ones using "scrubbing queries". Focus [27] uses the technique to build up an "approximate index" of relevant videos. For these domains, Vignette is a complementary design integrating perceptual information with video storage and can likely compound performance improvements.

9.2 Saliency-based Compression

Vignette builds on a large body of work in saliency-based compression. Early work improved the accuracy of saliency prediction [36, 40], the speed of computing saliency [22, 23, 65], or coding efficiency [22, 24, 45, 57, 65]. These existing solutions require custom versions of outdated codecs or solving costly optimization problems during each transcoding run. Vignette fundamentally differs from other contributions in perceptual compression by introducing a system design that can flexibly use *any* saliency prediction algorithm or video codec, rather than focusing only on accuracy, speed, or efficiency of saliency prediction. The limitations of prior work specifically influenced Vignette's design as a storage manager compatible with existing codecs, uses low-overhead metadata, and exposes a simple API for integration.

More recently, multimedia and networking research optimized streaming bandwidth requirements for 360° and VR video by decreasing quality outside the VR field-of-view [15, 26, 42, 55]; while similar in spirit to perceptual compression, this only compresses non-visible regions of a video. Semantic-based compression, another variant of perceptual compression, optimizes compression for regions with objects of interest. Recent work [37, 38] targets streaming applications, demonstrating the potential for energy efficiency and reduced network cost for end-devices, but not storage. Examining concerns for future VR pipelines, Sitzmann et al. [57] observe the impact of leveraging saliency for VR video storage and identified key perceptual requirements, but do not address the production or distribution of saliency-compressed videos.

10 CONCLUSION

Video data continues to grow with increased video capture and consumption trends, but leveraging perceptual cues can help manage this data. This paper proposes integrating perceptual compression techniques with cloud video storage infrastructure to improve storage capacity and video bitrates while maintaining perceptual quality. Vignette combines automatic generation of perceptual information with a video transcoding pipeline to enable large-scale perceptual compression with minimal data overhead. Our offline compression techniques deliver storage savings of up to 95%, and user trials confirm little perceptual quality loss for Vignette videos 50-75% smaller in size.

Vignette's design complements the contributions of existing large-scale cloud video storage and processing systems. Video systems can use Vignette to further improve storage capacity or in anticipation of video workloads using perceptual cues. As VR video consumption and new perceptual markers — such as eye trackers in VR headsets — grow in popularity, Vignette's techniques will be critical in integrating perceptual compression at large scale for higher quality, lower bitrate video storage.

ACKNOWLEDGMENTS

We thank members of the UW Architecture and Systems groups, and the anonymous reviewers for their feedback on earlier versions of this work; our shepherd Mahadev Satyanarayan for his guidance during the revision process; and the participants in our user study. This work is supported by the NSF through grants CCF-1703051, IIS-1546083, CCF-1518703, and CNS-1563788; DARPA award FA8750-16-2-0032; DOE award DE-SC0016260; a Google Faculty Research Award; an award from the University of Washington Reality Lab; gifts from the Intel Science and Technology Center for Big Data, Intel Corporation, Adobe, Amazon, Facebook, Huawei, and Google; and by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] Anne Aaron, Zhi Li, Megha Manohara, Jan De Cock, and David Ronca. 2015. *Per-Title Encode Optimization.* https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2 Accessed: 2019-10-01.
[2] Anne Aaron and David Ronca. 2015. *High Quality Video Encoding at Scale.* http://techblog.netflix.com/2015/12/high-quality-video-encoding-at-scale.html Accessed: 2019-10-01.
[3] Lixiang Ao, Liz Izhikevich, Geoffrey M Voelker, and George Porter. 2018. Sprocket: A serverless video processing framework. In *Proceedings of the Ninth ACM Symposium on Cloud Computing (SoCC '18).* ACM.
[4] Doug Beaver, Sanjeev Kumar, Harry Li, Jason Sobel, and Peter Vajgel. 2010. Finding a needle in Haystack: Facebook's photo storage. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI).* USENIX Association. https://www.usenix.org/legacy/event/osdi10/
[5] Fabrice Bellard. [n.d.]. FFmpeg. https://ffmpeg.org
[6] Andreas Bulling, Daniel Roggen, and Gerhard Tröster. 2009. Wearable EOG Goggles: Seamless Sensing and Context-awareness in Everyday Environments.

Journal of Ambient Intelligence and Smart Environments 1, 2 (April 2009). http://dl.acm.org/citation.cfm?id=2350315.2350320

[7] Zoya Bylinskii, Tilke Judd, Ali Borji, Laurent Itti, Frédo Durand, Aude Oliva, and Antonio Torralba. [n.d.]. MIT Saliency Benchmark.

[8] Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning Visual Importance for Graphic Designs and Data Visualizations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM. https://doi.org/10.1145/3126594.3126653

[9] Zoya Bylinskii, Adrià Recasens, Ali Borji, Aude Oliva, Antonio Torralba, and Frédo Durand. 2016. Where Should Saliency Models Look Next?. In *European Conference on Computer Vision (ECCV)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing.

[10] François Chollet et al. 2015. Keras. https://keras.io.

[11] Cisco. 2008. *Cisco Visual Networking Index: Forecast and Methodology, 2008–2013*. https://www.cisco.com/c/dam/global/pt_br/assets/docs/whitepaper_VNI_06_09.pdf Accessed: 2019-10-01.

[12] Cisco. 2016. *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf Accessed: 2019-10-01.

[13] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. 2016. A Deep Multi-Level Network for Saliency Prediction. In *International Conference on Pattern Recognition (ICPR)*.

[14] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne Aaron. 2016. A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications. In *Applications of Digital Image Processing XXXIX*, Vol. 9971. International Society for Optics and Photonics.

[15] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360&Deg; Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM. https://doi.org/10.1145/3083165.3083180

[16] fblive 2018. Facebook Live | Live Video Streaming. https://live.fb.com/

[17] Robert E. Fontana and Gary M. Decad. 2018. Moore's law realities for recording systems and memory storage components: HDD, tape, NAND, and optical. *AIP Advances* 8, 5 (2018). https://doi.org/10.1063/1.5007621

[18] Alliance for Open Media (AOM). 2018. AV1 Software Repository. https://aomedia.googlesource.com/aom

[19] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association. https://www.usenix.org/conference/nsdi18/presentation/fouladi

[20] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi

[21] Blender Foundation. 2002. *Blender Open Projects*. https://www.blender.org/about/projects/ Accessed: 2019-10-01.

[22] C. Guo and L. Zhang. 2010. A Novel Multiresolution Spatiotemporal Saliency Detection Model and Its Applications in Image and Video Compression. *IEEE Transactions on Image Processing* 19, 1 (Jan 2010). https://doi.org/10.1109/TIP.2009.2030969

[23] Rupesh Gupta, Meera Thapar Khanna, and Santanu Chaudhury. 2013. Visual saliency guided video compression algorithm. *Signal Processing: Image Communication* 28, 9 (2013). https://doi.org/10.1016/j.image.2013.07.003

[24] H. Hadizadeh and I. V. Bajić. 2014. Saliency-Aware Video Compression. *IEEE Transactions on Image Processing* 23, 1 (Jan 2014). https://doi.org/10.1109/TIP.2013.2282897

[25] Brandon Haynes, Amrita Mazumdar, Armin Alaghi, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2018. LightDB:A DBMS for Virtual Reality. *Proceedings of the VLDB Endowment* 11, 10 (2018).

[26] Brandon Haynes, Artem Minyaylov, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2017. VisualCloud Demonstration: A DBMS for Virtual Reality. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM. https://doi.org/10.1145/3035918.3058734

[27] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodík, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *OSDI*. https://www.usenix.org/conference/osdi18/presentation/hsieh

[28] Qi Huang, Petchean Ang, Peter Knowles, Tomasz Nykiel, Iaroslav Tverdokhlib, Amit Yajurvedi, Paul Dapolito, IV, Xifan Yan, Maxim Bykov, Chuen Liang, Mohit Talwar, Abhishek Mathur, Sachin Kulkarni, Matthew Burke, and Wyatt Lloyd. 2017. SVE: Distributed Video Processing at Facebook Scale. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. ACM. https://doi.org/10.1145/3132747.3132775

[29] X. Huang, C. Shen, X. Boix, and Q. Zhao. 2015. SALICON: Reducing the Semantic Gap in Saliency Prediction by Adapting Deep Neural Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV.2015.38

[30] Djordje Jevdjic, Karin Strauss, Luis Ceze, and Henrique S. Malvar. 2017. Approximate Storage of Compressed and Encrypted Videos. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17)*. ACM. https://doi.org/10.1145/3037697.3037718

[31] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *SIGCOMM*. https://doi.org/10.1145/3230543.3230574

[32] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: Fast Exploratory Video Queries using Neural Networks. (2018). arXiv:1805.01046 http://arxiv.org/abs/1805.01046

[33] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *PVLDB* 10, 11 (2017).

[34] V. Kantorov and I. Laptev. 2014. Efficient feature extraction, encoding and classification for action recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[35] Ioannis Katsavounidis. 2018. *Dynamic optimizer – a perceptual video encoding optimization framework*. https://medium.com/netflix-techblog/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f Accessed: 2019-10-01.

[36] J. S. Lee and T. Ebrahimi. 2012. Perceptual Video Compression: A Survey. *IEEE Journal of Selected Topics in Signal Processing* 6, 6 (Oct 2012). https://doi.org/10.1109/JSTSP.2012.2215006

[37] Yue Leng, Chi-Chun Chen, Qiuyue Sun, Jian Huang, and Yuhao Zhu. 2018. Semantic-Aware Virtual Reality Video Streaming. In *Proceedings of the 9th Asia-Pacific Workshop on Systems (APSys '18)*. ACM, Article 21. https://doi.org/10.1145/3265723.3265738

[38] Yue Leng, Chi-Chun Chen, Qiuyue Sun, Jian Huang, and Yuhao Zhu. 2019. Energy-Efficient Video Processing for Virtual Reality. (2019).

[39] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. *Toward A Practical Perceptual Video Quality Metric*. http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html Accessed: 2019-10-01.

[40] Zhicheng Li, Shiyin Qin, and Laurent Itti. 2011. Visual attention guided bit allocation in video compression. *Image and Vision Computing* 29, 1 (2011).

[41] Peng Liu, Jongwon Yoon, Lance Johnson, and Suman Banerjee. 2016. Greening the Video Transcoding Service with Low-Cost Hardware Transcoders. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association. https://www.usenix.org/conference/atc16/technical-sessions/presentation/liu

[42] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. 360&Deg; Video Viewing Dataset in Head-Mounted Virtual Reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM. https://doi.org/10.1145/3083187.3083219

[43] Andrea Lottarini, Alex Ramirez, Joel Coburn, Martha A. Kim, Parthasarathy Ranganathan, Daniel Stodolsky, and Mark Wachsler. 2018. Vbench: Benchmarking Video Transcoding in the Cloud. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18)*. ACM. https://doi.org/10.1145/3173162.3173207

[44] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. 2016. Optasia: A Relational Platform for Efficient Large-Scale Video Analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing (SoCC '16)*. ACM. https://doi.org/10.1145/2987550.2987564

[45] V. Lyudvichenko, M. Erofeev, Y. Gitman, and D. Vatolin. 2017. A semiautomatic saliency model and its application to video compression. In *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. https://doi.org/10.1109/ICCP.2017.8117038

[46] Ikuo Magaki, Moein Khazraee, Luis Vega Gutierrez, and Michael Bedford Taylor. 2016. ASIC Clouds: Specializing the Datacenter. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press. https://doi.org/10.1109/ISCA.2016.25

[47] Megha Manohara, Anush Moorthy, Jan De Cock, Ioannis Katsavounidis, and Anne Aaron. 2018. *Optimized shot-based encodes: Now Streaming!* https://medium.com/netflix-techblog/optimized-shot-based-encodes-now-streaming-4b9464204830 Accessed: 2019-10-01.

[48] Mediakix. 2016. The Facebook Video Statistics Everyone Needs to Know. http://mediakix.com/2016/08/facebook-video-statistics-everyone-needs-know

[49] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. 2013. An Overview of Tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing* 7, 6 (Dec 2013). https://doi.org/10.1109/JSTSP.2013.2271451

[50] Subramanian Muralidhar, Wyatt Lloyd, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Viswanath Sivakumar, Linpeng Tang,

and Sanjeev Kumar. 2014. f4: Facebook's Warm BLOB Storage System. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/muralidhar

[51] Andrey Norkin, Jan De Cock, Aditya Mavlankar, and Anne Aaron. 2016. *More Efficient Mobile Encodes for Netflix Downloads.* https://medium.com/netflix-techblog/more-efficient-mobile-encodes-for-netflix-downloads-625d7b082909 Accessed: 2019-10-01.

[52] nvenc [n.d.]. Nvidia Video codec. https://developer.nvidia.com/nvidia-video-codec-sdk

[53] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient Video Analysis at Scale. *ACM Trans. Graph.* 36, 4 (2018).

[54] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[55] Jihoon Ryoo, Kiwon Yun, Dimitris Samaras, Samir R. Das, and Gregory Zelinsky. 2016. Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, Article 6. https://doi.org/10.1145/2910017.2910592

[56] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. 2017. Fast Video Classification via Adaptive Cascading of Deep Models. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[57] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (April 2018). https://doi.org/10.1109/TVCG.2018.2793599

[58] Gary J. Sullivan, Jens-Rainer Ohm, Woojin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012).

[59] Linpeng Tang, Qi Huang, Amit Puntambekar, Ymir Vigfusson, Wyatt Lloyd, and Kai Li. 2017. Popularity Prediction of Facebook Videos for Higher Quality Streaming. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. USENIX Association. https://www.usenix.org/conference/atc17/technical-sessions/presentation/tang

[60] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. (2016). arXiv:1605.02688 http://arxiv.org/abs/1605.02688

[61] Eric Whitmire, Laura Trutoiu, Robert Cavin, David Perek, Brian Scally, James Phillips, and Shwetak Patel. 2016. EyeContact: Scleral Coil Eye Tracking for Virtual Reality. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM. https://doi.org/10.1145/2971763.2971771

[62] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodík, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*.

[63] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodík, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *NSDI*. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang

[64] Haibo Zhang, Prasanna Venkatesh Rengasamy, Shulin Zhao, Nachiappan Chidambaram Nachiappan, Anand Sivasubramaniam, Mahmut T. Kandemir, Ravi Iyer, and Chita R. Das. 2017. Race-to-sleep + Content Caching + Display Caching: A Recipe for Energy-efficient Video Streaming on Handhelds. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50 '17)*. ACM. https://doi.org/10.1145/3123939.3123948

[65] Fabio Zund, Yael Pritch, Alexander Sorkine-Hornung, Stefan Mangold, and Thomas Gross. 2013. Content-aware compression using saliency-driven image retargeting. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE.