Poster: FFmpeg360 for 360-Degree Videos: Edge-Based Transcoding, View Rendering, and Visual Quality Comparison

Yao Liu
SUNY Binghamton
yaoliu@binghamton.edu

Chao Zhou
SUNY Binghamton
czhou5@binghamton.edu

Shuoqian Wang
SUNY Binghamton
swang130@binghamton.edu

Mengbai Xiao
The Ohio State University
xiao.736@osu.edu

ABSTRACT

360-degree video streaming is an emerging technology that provides immersive experiences to users. However, it both requires high streaming bandwidth and wastes a significant portion of the bandwidth. To improve bandwidth-efficiency, researchers have proposed oriented projections and FOV rendering. However, existing software and tools are unable to perform these tasks in an online manner under stringent processing latency and throughput constraints.

In this paper, we present `FFmpeg360`, an open source software that leverages the GPU to accelerate video processing, making it possible to deploy video transformation tasks on edge computing devices. In addition, "view videos" generated by `FFmpeg360` can be used to compare visual quality of views rendered from various spatial and bitrate adaptation schemes designed for 360-degree video streaming.

ACM Reference Format:
Yao Liu, Chao Zhou, Shuoqian Wang, and Mengbai Xiao. 2019. Poster: FFmpeg360 for 360-Degree Videos: Edge-Based Transcoding, View Rendering, and Visual Quality Comparison. In *SEC '19: ACM/IEEE Symposium on Edge Computing, November 7–9, 2019, Arlington, VA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3318216.3363372

1 INTRODUCTION

360-degree video streaming has increased in popularity in recent years. Compared to traditional video streaming, 360-degree videos encode video content fully surrounding a camera position, allowing users to freely explore views from any orientation.

360-degree content is best represented as pixels on a sphere. However, existing video codecs such as H.264 and HEVC cannot directly encode spherical pixels. This means a 360-degree video frame must be first projected to a rectangular frame before it can be encoded. For example, Figure 1 shows an 360-degree image in the equirectangular projection [4].

While a 360-degree video encodes omnidirectional views surrounding a camera, users can only observe a small field-of-view (FOV) from each omnidirectional frame. For example, to render views centered at the sphere's equator with 100° by 100° FOV, less than 15% of pixels on the equirectangular frame are needed (Figure 1). The remaining 85% of pixels are wasted, i.e., downloaded, decoded, but not rendered to users. As a result, much of the streaming bandwidth is wasted.

Of the many methods proposed to improve bandwidth consumption of 360-degree video streaming, two are of interest in this setting: *i)* spherical projections designed to encode a "focus area" in higher quality than other portions of spherical surface pixels and *ii)* edge-based-rendering to transcode and transmit frames including the FOV surrounding the current user-orientation [10]. Facebook's offset cubic projection is an example of method *i)* [2]. This projection encodes a front-facing 30° by 30° portion of the sphere in higher quality than a rear-facing 150° by 150° portion [13]. When combined with user's view prediction, both approaches allow the streaming client to efficiently use the available bandwidth to request 360-degree video content in high quality.

Existing approaches transform input 360-degree videos into oriented projections in an offline manner and incur a significant amount of storage overhead at the server-side [13]. If performed in an online manner, large amount of computation resources are required for processing video frames at the required throughput, e.g., 30 frames-per-second. Similarly, significant computation resources are required for edge-based FOV rendering to produce views in response to predicted view orientations in real-time and at required throughput.

In this paper, we present `FFmpeg360` [7], an open source software library that can perform spatial transcoding (i.e., re-projection) and render views correspond to user's view orientations. Compared to other 360-degree video processing tools, `FFmpeg360` leverages the GPU for processing, making it possible to perform online spatial transcoding and view rendering on edge computing resources while meeting processing latency and throughput requirements. We thus envision `FFmpeg360` to be used for edge-assisted 360-degree video streaming in the future. In addition, `FFmpeg360` enables visual quality assessment of different 360-degree streaming methods. It does so by generating "view videos" that mimic actual views observed by the user during streaming playback. These view videos can be generated under both a variety of different spherical projections and different spatial and bitrate adaptation approaches. Quality of videos generated under a baseline approach can then be compared with view videos generated under the target approach.

SEC '19, November 7–9, 2019, Arlington, VA, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6733-2/19/11.
https://doi.org/10.1145/3318216.3363372

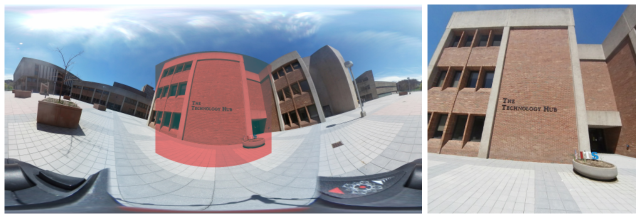

Figure 1: The left figure shows a 360-degree image in the equirectangular projection. The right figure shows a rendered view oriented at yaw=0, pitch=0, with a 100° by 100° FOV. The red-shaded area in the left figure shows only 15% pixels on the equirectangular image used for rendering the view in the right figure.

2 EXISTING SOFTWARE AND LIMITATIONS

Facebook `transform360` [5] can transform input equirectangular videos to output videos in various projections such as the cubic projection, the equi-angular cubic (EAC) projection [1], and the barrel projection [3]. Frame transformation is performed using the CPU. It is unable to produce output video at high resolution at throughputs required for online streaming. It also cannot render views observed by users.

Samsung `360tools` [9] can convert videos in uncompressed YUV formats [8] from various supported input projections to output projections using the CPU. It can be used for generating visual quality metrics of 360-degree videos, including the traditional peak signal-to-noise ratio (PSNR), spherical PSNR (S-PSNR) based on the PSNR of over 600K points uniformly distributed on the sphere, and weighted spherical PSNR (WS-PSNR) that assigns different weights to points at different latitudes on the sphere [12]. However, it cannot render user-observed views, and its supported visual quality metrics do not represent the actual visual quality of views presented to users.

`omnieval` [12] supports conversion of 360-degree videos among various projections and can render user-observed views. However, similar to `transform360` and `360tools`, it uses the CPU for frame transformation. It also can only work with the YUV format and does not work with the more common video codec/formats such as H.264 in a MP4 container. `omnieval` can generate visual quality metrics include S-PSNR and WS-PSNR, but it cannot output the the visual quality of the viewport, e.g., viewport PSNR (V-PSNR).

3 FFMPEG360

`FFmpeg360` is implemented as a fork of the FFmpeg [6] utility for video processing. As a result, it can take input 360-degree videos in various projections, produce output re-projected 360-degree videos, and render "view videos" in codec/formats supported by FFmpeg.

The core component of `FFmpeg360` is the implementation of an FFmpeg video filter, `360-project`. This filter takes as input a decoded video frame in Y, U, and V channels. It performs required processing (i.e., re-projection or view rendering) on the three channels individually. It then returns the output video frame to the FFmpeg pipeline.

To speed up frame processing, `FFmpeg360` uses the cross-platform OpenGL framework, leveraging the GPU for pixel sampling. It implements different fragment shaders to support re-projection among various spherical projections and rendering of views from various input projections.

Spatial transcoding in `FFmpeg360` is performed via a perl script, `remap.pl`. Internally, `remap.pl` calls the `360-project` filter to perform the re-projection. Re-encoding of the output video is done by the FFmpeg encoding pipeline. For example, the following commands convert an input 360-degree video in EAC projection into standard cubic projection in 3000x2000 resolution.

```
$ ./remap.pl iv=eac.mp4 ov=cube.mp4 res=3000x2000 il=cube.lt \
    ofs=uneqdeg-ecoef.glsl ovs=vertex.glsl ol=cube.lt crf=18
```

View rendering is performed by calling the `FFmpeg360` executable directly. To render "view videos" from a recorded trace of user view orientations, the `360-project` filter can take an "orientation file" as input. Each line in the "orientation file" should contain a timestamp and a view orientation represented as an Euler angle <pitch, yaw, roll> in degrees. For example,

```
orientation.txt
timestamp pitch      yaw       roll
0.027396  -11.230908 -8.936484 4.323502
```

Given an input 360-degree video and an orientation file, we can create the "view video" as follows:

```
$ ./ffmpeg360 -loglevel "info" -y -i equirectangular.mp4 \
  -filter:v "project=1000:1000:90:90:0:0:0:simpleVertex.glsl:
       ↪ equirectangular.glsl:orientation.txt:equirectangular.lt"
       ↪  \
  view_video.mp4
```

Here, the resolution of the generated view video is configured to 1000x1000, and the view's FOV is configured to 90° by 90°.

Visual quality comparison can be performed using the `psnr` and `ssim` [11] filters provided by FFmpeg once "view videos" are generated. These filters take two input videos and outputs per-frame visual quality, i.e., viewport-PSNR, to a specified file. For example,

```
$ ./ffmpeg360 -i view_video.mp4 -i ground_view_video.mp4 \
  -lavfi psnr -f psnr.txt
```

Here, the "ground view video" is the reference video used in visual quality comparison. It is generated from the original 360-degree video in highest quality, e.g., an equirectangular video in 4K or 8K quality.

4 CONCLUSION

`FFmpeg360` is an open source software that can perform spatial transcoding, view rendering, and visual quality comparison for 360-degree videos. It uses the GPU to accelerate pixel sampling required for video processing, making it possible to perform online spatial transcoding and view rendering on edge computing devices.

Acknowledgment. This work is partially supported by NSF under grant CNS-1618931.

REFERENCES

[1] Bringing pixels front and center in VR video. https://blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/.

[2] End-to-end optimizations for dynamic streaming. https://code.fb.com/virtual-reality/end-to-end-optimizations-for-dynamic-streaming/.
[3] Enhancing high-resolution 360 streaming with view prediction. https://code.fb.com/virtual-reality/enhancing-high-resolution-360-streaming-with-view-prediction/.
[4] Equirectangular Projection. http://mathworld.wolfram.com/EquirectangularProjection.html.
[5] Facebook Transform360. https://github.com/facebook/transform360.
[6] FFmpeg. http://www.ffmpeg.org/.
[7] FFmpeg360. https://github.com/bingsyslab/ffmpeg360.
[8] Recommended 8-Bit YUV Formats for Video Rendering. https://docs.microsoft.com/en-us/windows/win32/medfound/recommended-8-bit-yuv-formats-for-video-rendering.
[9] Samsung 360tools. https://github.com/Samsung/360tools.
[10] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran, and Marco Dias Silva. Vr is on the edge: How to deliver 360 videos in mobile networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 30–35. ACM, 2017.
[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
[12] Matt Yu, Haricharan Lakshman, and Bernd Girod. A framework to evaluate omnidirectional video coding schemes. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, pages 31–36. IEEE, 2015.
[13] Chao Zhou, Zhenhua Li, and Yao Liu. A measurement study of oculus 360 degree video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 27–37. ACM, 2017.