Learning Compressible 360° Video Isomers

Yu-Chuan Su Kristen Grauman
{ycsu,grauman}@cs.utexas.edu
The University of Texas at Austin

Abstract

Standard video encoders developed for conventional narrow field-of-view video are widely applied to 360° *video as well, with reasonable results. However, while this approach commits arbitrarily to a projection of the spherical frames, we observe that some orientations of a* 360° *video, once projected, are more compressible than others. We introduce an approach to predict the sphere rotation that will yield the maximal compression rate. Given video clips in their original encoding, a convolutional neural network learns the association between a clip's visual content and its compressibility at different rotations of a cubemap projection. We validate our idea on thousands of video clips and multiple popular video codecs. The results show that this untapped dimension of* 360° *video compression has substantial potential—"good" rotations are typically* $8-10\%$ *more compressible than bad ones, and our learning approach can predict them reliably* 82% *of the time. The full report is published in the CVPR main conference [4].*

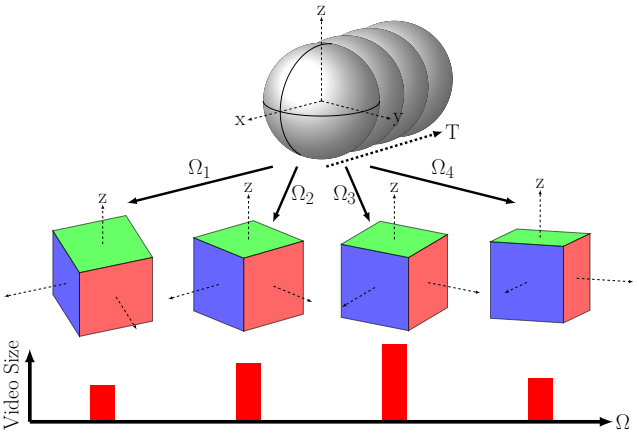

Figure 1: Our approach learns to automatically rotate the 360° video axis before storing the video in cubemap format. While the 360° videos are equivalent under rotation ("isomers"), the bit-streams are not because of the video compression procedures. Our approach analyzes the video's visual content to predict its most compressible isomer.

1. Introduction

The focus for 360° video compression is to find a proper projection that transforms a 360° frame into a rectangular planar image that will have a high compression rate. A current favorite is to project the sphere to a *cubemap* and unwrap the cube into a planar image [1–3] (see Fig. 2). Cubemaps can improve the compression rate by up to 25% compared to the previously popular equirectangular projection [3].

One unique property of 360° video is that each spherical video has an *infinite number of equivalents related by a rotation.* Therefore, each 360° video could be transformed into multiple possible cubemaps by changing the orientation of the cube, yet all of them represent the very same video content. We refer to these content-equivalent rotations as 360° *isomers*. We observe, however, that isomers are *not* equivalents in terms of compression. Different isomers interact differently with a given compression algorithm and so yield different compression rates (See Fig. 1). This is because the unwrapped cubemap is not a homogenous perspective image. Therefore, some of the properties that current compression algorithms exploit in perspective images do not hold. For example, while the content is smooth and continuous in perspective images, this need not be true along an inter-face boundary in an unwrapped cubemap. The discontinuity can introduce artificial high frequency signals and large abrupt motions, both of which harm the compression rate. Because the effect is content dependent, different orientations result in different compression rates.

We propose a learning-based approach to predict—from the video's visual content itself—the cubemap orientation that will minimize the video size [4]. First, we demonstrate empirically that the orientation of a cubemap does influence the compression rate, and the difference is not an artifact of a specific encoder but a general property over a variety of popular video formats. Based on that observation, we propose to automatically re-orient the cubemap for every group of pictures (GOP). Given encoded videos in a fixed orientation, we train a Convolutional Neural Network (CNN) that takes both the segmentation contours and motion vectors

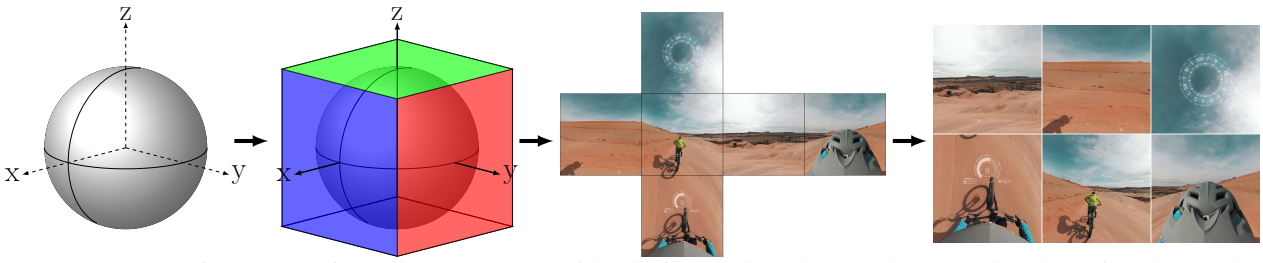

Figure 2: Cubemap format transformation. The 360° video is first projected to a cube enclosing the unit sphere and then unwrapped into 6 faces. The 6 faces are re-arranged to form a rectangular picture to fit video compression standards (2×3 frame on the right).

in the encoded bit-stream and predicts the orientation that will yield the minimum video size. By avoiding rendering and encoding the video clip in all possible orientations, our approach is computationally efficient and strikes a balance between speed and compression rate.

The key benefit of our approach is a higher compression rate for 360° video that requires only to re-render the cubemap. In particular, our idea does not require changing the video format nor the compression algorithm, which makes it fully compatible with any existing video codec. The only additional information that our method needs to encode is the selected orientation of each GOP, which can easily be encoded as meta data (and may become the standard [2]).

2. Cubemap Orientation Analysis

First, we verify that the orientation of a cubemap projection is indeed important for 360° video compression.

Data Preparation To study the correlation between cubemap orientation and compression rate, we collect a 360° video dataset from YouTube with 4K quality. The dataset covers a variety of video content and recording situations, including aerial, underwater, sports, animal, news, and event videos, and the camera can be either static or moving. We download the videos in equirectangular projection with 3,840 pixels width encoded in H264.

We next transcode the video into cubemap format and extract the video size in different orientations. We divide the video into 2 second clips and encode each clip independently following the GOP structure in video codecs. This results in a dataset consisting of 7,436 video clips from 80 videos with 4.2 hours total length. For each clip, we sample the cubemap orientation

$$\Omega = (\phi, \theta) \in \Phi \times \Theta \tag{1}$$

with different yaw (ϕ) and pitch (θ) in $\Theta = \Phi = \{-45°, -40°, \cdots, 45°\}$, i.e., every $5°$ between $[-45°, 45°]$. This yields $|\Phi \times \Theta| = 361$ different orientations. We restrict the orientation within $90°$ because of the rotational symmetry along each axis.

For each orientation, we transform the video into cubemap format using FFMPEG with 960 pixels resolution for each face. Fig. 2 illustrates the transformation. The video is then encoded using off-the-shelf encoders into three popular formats—H264 using x264, HEVC using x265, and VP9 using libvpx. We use lossless compression for all three formats and extract the size of the final encoded bit-stream. The dataset is available on our project webpage[1].

		H264	HEVC	VP9
Video r (%)	Avg.	8.43 ± 2.43	8.11 ± 2.03	7.83 ± 2.34
	Range	[4.34, 15.18]	[4.58, 13.67]	[3.80, 14.72]
Clip r (%)	Avg.	10.37 ± 8.79	8.88 ± 8.23	9.78 ± 8.62
	Range	[1.08, 76.93]	[1.40, 74.95]	[1.70, 75.84]

Table 1: Achievable video size reduction through rotation for different formats. We can reduce the video size by up to 77% by optimally changing the cubemap orientation.

Data Analysis Next we examine the size reduction we can achieve by changing the cubemap orientation. In particular, we compute the *reduction*

$$r = 100 \times \frac{S_{\Omega^{max}} - S_{\Omega^{min}}}{S_{\Omega^{max}}}, \tag{2}$$

where S_Ω is the encoded bit-stream size with orientation Ω and Ω^{max}/Ω^{min} corresponds to the orientation with maximum/minimum bit-stream size.

Table 1 shows the results. The average video size reduction $\overline{r}$ is 8.43% for H264, which means that we can reduce the overall 360° video size by more than 8% through rotating the video axis. This corresponds to a 2GB reduction in our 80 video database and would scale to 25.3TB for a million video database. The range of r for each clip is $[1.08, 76.93]$, which indicates that the compression rate is strongly content dependent, and the size reduction can be up to 77% for a single video if we allow the encoder to re-orient the 360° video. Finally, we see that the average and range of reductions are quite similar across encoders, indicating that compressibility of isomers is not unique to a particular codec.

3. Approach

Next, we introduce our approach for improving 360° video compression rates by predicting the most compress-

[1] http://vision.cs.utexas.edu/projects/360isomers

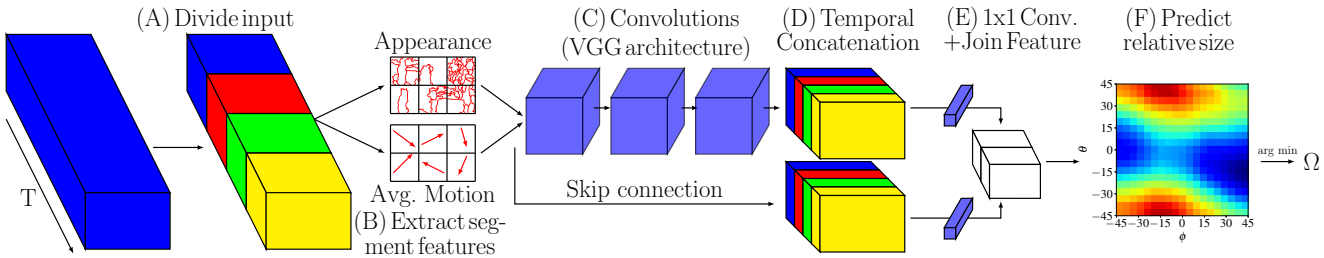

Figure 3: Our model takes a video clip as input and predicts Ω^{min} as output. (A) It first divides the video into 4 segments temporally and (B) extracts appearance and motion features from each segment. (C) It then concatenates the appearance and motion feature maps and feeds them into a CNN. (D) The model concatenates the outputs of each segment together and joins the output with the input feature map using skip connections to form the video feature. (F) It then learns a regression model that predicts the video size S_Ω for all Ω and takes the minimum one as the predicted optimally compressible isomer.

ible isomer. Given a 360° video clip, our goal is to identify Ω^{min} to minimize the video size. A naive solution is to render and compress the video for all possible Ω and compare their sizes. While this guarantees the optimal solution, it introduces a significant computational overhead, i.e., 360 times more computation than encoding the video with a fixed Ω. For example, it would take more than 1.5 hours to encode one clip using the default x264 encoder on a 48 core machine. Moreover, the computational cost will grow quadratically if we allow more fine-grained control. Therefore, enumerating Ω is not practical.

Instead, we propose to predict Ω^{min} from the raw input without rerendering the video. Given the input video in cubemap format, we extract both motion and appearance features (details below) and feed them into a CNN that predicts the video size S_Ω for each Ω. We treat it as a regression problem and learn a model that predicts 361 real values using L2 loss. The final prediction of the model is

$$\Omega^{min} = \arg\min_{\Omega} S_\Omega. \tag{3}$$

See Fig. 3. The computational cost remains roughly the same as transcoding the video because the prediction takes less than a second, which is orders of magnitude shorter than encoding the video and thus negligible. Since no predictor will generalize perfectly, there is a chance of decreasing the compression rate in some cases. However, experimental results show that it yields very good results and strikes a balance between computation time and video size.

More specifically, we first divide the input video into 4 equal length segments. For each segment, we extract the appearance and motion features for each frame and average them over the segment. For appearance features, we segment the frame into regions and take the segmentation contour map as the feature. The segmentation contour captures edges, which imply object boundaries and high frequency signals that take more bits in video compression. For motion features, we take the motion vectors directly from the input video stream encoding, as opposed to computing optical flow. The motion vectors are readily available in the input and thus this saves computation. Furthermore, motion vectors provide more direct information about the encoder. For regions without a motion vector, we simply pad 0 for the input regardless of the encoding mode. The final feature map is the concatenation of appearance and motion feature.

The feature maps for each segment are then fed into a CNN and concatenated together as the video feature. We use the VGG architecture except that we increase the number of input channels in the first convolution layer. Because fine details are important in video compression, we use skip connections to combine low level information with high level features. In particular, we combine the input feature map and final convolution output as the segment feature after performing 1x1 convolution to reduce the dimension to 4 and 64 respectively. The video feature is then fed into a fully-connected layer with 361 outputs as the regression model. Note that we remove the fully-connected layers in the VGG architecture to keep the spatial resolution for the regression model and reduce model size.

4. Experiments

To evaluate our method, we compute the size reduction it achieves on the 360° video dataset introduced in Sec. 2.

Baselines Because we are the first to study how to predict the cubemap orientation for better compression, we compare our method with the following two heuristics:

- RANDOM — Randomly rotate the cubemap to one of the 361 orientations. This represents the compression rate when we have no knowledge about the video orientation.
- CENTER — Use the orientation provided by the videographer. This is a strong prior, usually corresponding to the direction of the videographer's gaze or movement and lying on the horizon of the world coordinate.

Evaluation metrics We compare each method using the normalized size reduction

$$\tilde{r} = 100 \times \frac{S_{\Omega^{max}} - S_\Omega}{S_{\Omega^{max}} - S_{\Omega^{min}}} \tag{4}$$

for each video. Specifically, we compute the largest full-video size by choosing Ω^{max} for every clip and sum the

	H264	HEVC	VP9
RANDOM	50.75	51.62	51.20
CENTER	74.35	63.34	72.92
OURS	82.10	79.10	81.55

Table 2: Size reduction of each method. The range is $[0, 100]$, the higher the better.

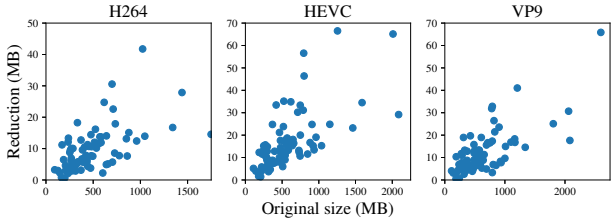

Figure 4: Absolute size reduction (MB) of each video. Each point represents the input video size vs. size reduction relative to CENTER achieved by our model.

clip sizes. Similarly, we compute the minimum video size. Given the predicted orientation, we compute the video size by rotating the cubemap using the predicted Ω. The result indicates the fraction of reduction the method achieves compared to the optimal result. We report results with 4-fold validation, where each fold contains 20 videos.

Results Table 2 shows the size reduction achieved by our method. Our method performs better than the baselines in all video compression formats by $7\% - 16\%$. The improvement over the baseline is largest in HEVC, which indicates that the advantage of our approach will become more significant as HEVC gradually replaces H264. Interestingly, the CENTER baseline performs particularly worse in HEVC. The reason is that HEVC allows the encoder to achieve good compression rates in more diverse situations, so the distribution of Ω^{min} becomes more dispersed. The result further shows the value in considering cubemap orientation during compression as more advanced video codecs are used. While there remains a 20% room for improvement compared to the optimal result (as ascertained by enumerating Ω), our approach is significantly faster and takes less than 0.3% the computation. We also show the absolute file size reduction versus the original video size for each video in Fig. 4. The size reduction by our method, though depending on the video content, is roughly linear to the original video size. Note that the original videos are encoded with orientation $\Omega_{0,0}$.

Fig. 5 shows example prediction results. Our approach performs well despite the diversity in the video content and recording situation. The complexity in the content would make it hard to design a simple rule-based method to predict Ω^{min} (such as analyzing the continuity); a learning based method is necessary. The last row shows a failure case of our method, where the distribution of video size is multimodal, and the model selects the suboptimal mode.

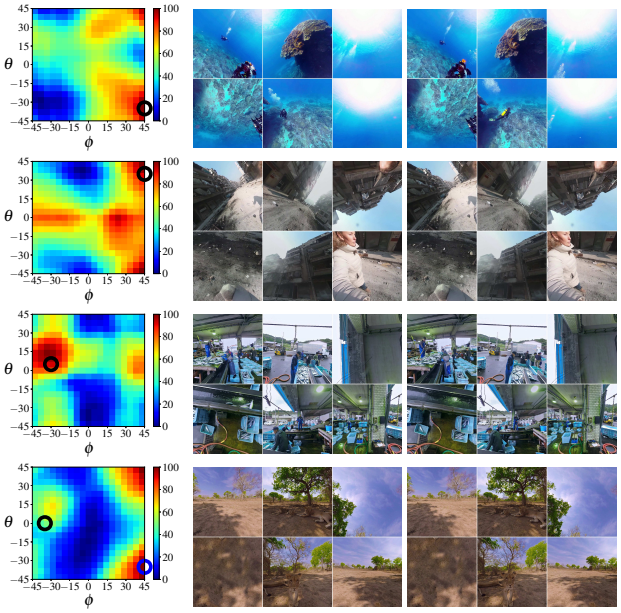

Figure 5: Qualitative examples. The heatmap shows the normalized reduction, and the overlaid circle shows our predicted result. The two images are the first and last frame of the clip rendered in the predicted orientation. Last row shows a failure example. Best viewed in color.

5. Conclusion

This work studies how to improve $360°$ video compression by selecting a proper orientation for cubemap projection. Our analysis across 3 popular codecs shows scope for reducing video sizes by up to 77% through rotation, with an average of more than 8% over all videos. We propose an approach that predicts the optimal orientation given the video in a single orientation. It achieves 82% the compression rate of the optimal orientation while requiring less than 0.3% of the computation of a non-learned solution (fraction of a second vs. 1.5 hours per GOP).

Acknowledgement. This research is supported in part by NSF IIS-1514118, an AWS gift, a Google PhD Fellowship, and a Google Faculty Research Award.

References

[1] C. Brown. Bringing pixels front and center in VR video. https://www.blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/, March 2017. 1

[2] B. Choi et al. Wd on iso/iec 23000-20 omnidirectional media application format. ISO/IEC JTC1/SC29/WG11, 2017. 1, 2

[3] E. Kuzyakov and D. Pio. Next generation video encoding techniques for 360 video and vr. https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/, January 2016. 1

[4] Y.-C. Su and K. Grauman. Learning compressible $360°$ video isomers. In *CVPR*, 2018. 1