Learning Compressible 360° Video Isomers

Yu-Chuan Su Kristen Grauman
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. Given a novel video, our learning-based approach efficiently infers the most compressible direction in one shot, without repeated rendering and compression of the source video. We validate our idea on thousands of video clips and multiple popular video codecs. The results show that this untapped dimension of* 360° *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.*

1. Introduction

Both the technology and popularity of 360° video has grown rapidly in recent years, for emerging Virtual Reality (VR) applications and others. Sales of 360° cameras are expected to grow by 1500% from 2016 to 2022 [45]. Foreseeing the tremendous opportunities in 360° video, many companies are investing in it. For example, Facebook and YouTube have offered 360° content support since 2015. Facebook users have since uploaded more than one million 360° videos [8], and YouTube plans to bring 360° videos to even broader platforms (TV, gaming consoles). 360° editing tools are now available in popular video editors such as PowerDirector and Premiere Pro. Meanwhile, on the research side, there is strong interest in improving 360° video display [22, 26, 21, 41, 40, 19, 29], and performing visual processing efficiently on the new format [23, 14, 39]. All together, these efforts make 360° video production and distribution easier and more prevalent than ever.

At the core of all video technologies is the data format.

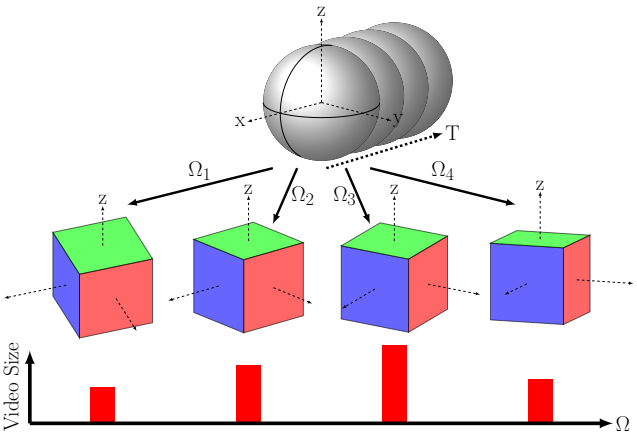


Figure 1: Our approach learns to automatically rotate the 360° video axis before storing the video in cubemap format. While the 360° videos are equivalent under rotation ("isomers"), the bit-streams are not because of the video compression procedures. Our approach analyzes the video's visual content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

In particular, a compressed video bit-stream format is the basis for all video related applications, ranging from video capture, storage, processing to distribution. Without adequate compression, all of the above suffer. 360° video is no exception. Thus far, 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 [27, 10, 32] (see Fig. 2). Cubemaps can improve the compression rate by up to 25% compared to the previously popular equirectangular projection [28].

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*.[1] The isomers, however, are *not*

[1]Strictly speaking isomers are equivalent only theoretically, because pixels are discretely sampled and rotating a cubemap requires interpolating the pixels. Nevertheless, as long as the pixel density, i.e. video resolution, is high enough, the information delta is negligible.

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 (cf. Sec. 3.2 and Fig. 5). In short, our key insight is that the compression rate of a 360° video will depend on the orientation of the cubemap it is projected on.

We propose a learning-based approach to predict—from the video's visual content itself—the cubemap orientation that will minimize the video size. 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).[2] A naive solution would enumerate each possible orientation, compress the GOP, and pick the one with the lowest encoded bit-stream size. However, doing so would incur substantial overhead during compression, prohibitively costly for many settings. Instead, our approach renders the GOP for a *single* orientation after predicting the optimal orientation from the video clip rendered in its canonical orientation. Given encoded videos in a fixed orientation, we train a Convolutional Neural Network (CNN) that takes both the segmentation contours and motion vectors 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 greatly reduces the computational cost 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. This is especially important in the realm of video compression, because a new video format often takes years to standardize and deploy, and so changing the bit-stream format would incur very high overhead. 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 part of the standard in the future [12]).

We evaluate our approach on 7,436 clips containing varying content. We demonstrate our idea has consistent impact across three popular encoders, with video size reductions up to 77% and typical reductions of about 8%. Across all videos, our learning approach achieves on average 82% of the best potential compression rate available for all feasible isomers.

[2] a collection of successive pictures within a coded video stream.

2. Related Work

360° video analysis Recent research explores ways to improve the user experience of watching 360° videos, including stabilizing the videos [22, 26, 21] or directing the field-of-view (FOV) automatically [41, 40, 19, 29]. Other works study visual features in 360° images such as detecting SIFT [18] or learning a CNN either from scratch [23, 14] or from an existing model trained on ordinary perspective images [39]. All of these methods offer new applications of 360° videos, and they assume the inputs are in some given form, e.g., equirectangular projection. In contrast, we address learning to optimize the data format of 360° video, which can benefit many applications.

360° video compression 360° video has sparked initial interest in new video compression techniques. A Call for Evidence this year for a meeting on video standards [46] calls attention to the need for compression techniques specific to 360° video, and responses indicate that substantial improvement can be achieved in test cases [13, 17, 7, 15]. Whereas these efforts aim for the next generation in video compression standards, our method is compatible with existing video formats and can be applied directly without any modification of existing video codecs. For video streaming, some work studies the value in devoting more bits to the region of 360° content currently viewed by the user [42, 38]. However, they require the *current* viewing direction of the user and reduce the video quality beyond the user's field of view. In contrast, our method does not know where the user will look and encodes the entire video with the same quality.

Projection of spherical images 360° image projection has long been studied in the field of map projection. As famously proven by Gauss, no single projection can project a sphere to a plane without introducing some kind of distortion. Therefore, many different projections are proposed, each designed to preserve certain properties such as distance, area, direction, etc. [37]. For example, the popular equirectangular projection preserves the distance along longitude circles. Various projection models have been developed to improve perceived quality for 360° images. Prior work [47] studies how to select or combine the projections for a better display, and others develop new projection methods to minimize visual artifacts [24, 11]. Our work is not about the human-perceived quality of a projected 360° image; rather, the mode of projection is relevant to our problem only in regards to how well the resulting stack of 2D frames can be compressed.

Cubemap is adopted as one of the two presentations for 360° video in the MPEG Omnidirectional MediA Format

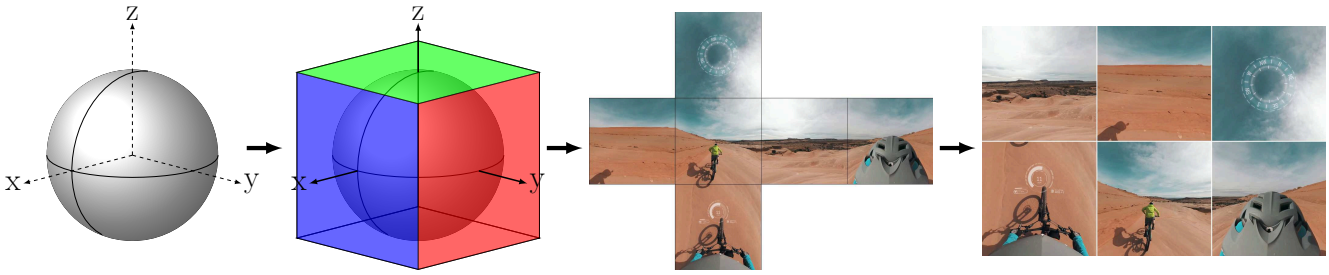


Figure 2: Cubemap format transformation. The 360° video is first projected to a cube enclosing the unit sphere and then unwrapped into 6 faces. The 6 faces are re-arranged to form a rectangular picture to fit video compression standards (2×3 frame on the right).

(OMAF) [32], i.e., the future 360° video standard, and major 360° video sharing sites such as YouTube and Facebook have turned to the new format [27, 10]. Cubemaps can improve the compression rate by 25% compared to equirectangular projection, which suffers from redundant pixels and distorted motions [28]. The Rotated Sphere Projection is an alternative to cubemap projection with fewer discontinuous boundaries [6]. Motivated by the compression findings [28], our approach is built upon the standard cubemap format. Our method is compatible with existing data formats and can further reduce video size at almost zero cost.

Deep learning for image compression Recent work investigates ways to improve image compression using deep neural networks. One common approach is to improve predictive coding using either a feed-forward CNN [35, 34] or recurrent neural network (RNN) [43, 44, 20]. The concept can also be extended to video compression [35]. Another approach is to allocate the bit rate dynamically using a CNN [30]. While we also study video compression using a CNN, we are the first to study 360° video compression, and—CNN or otherwise—the first to exploit spherical video orientation to improve compression rates. Our idea is orthogonal to existing video compression algorithms, which could be combined with our approach without any modification to further improve performance.

3. Cubemap Orientation Analysis

Our goal is to develop a computationally efficient method that exploits a cubemap's orientation for better compression rates. In this section, we perform a detailed analysis on the correlation between encoded video size and cubemap orientation. The intent is to verify that orientation is indeed important for 360° video compression. We then introduce our method to utilize this correlation in Sec. 4.

First we briefly review fundamental video compression concepts, which will help in understanding where our idea has leverage. Modern video compression standards divide a video into a series of groups of pictures (GOPs), which can be decoded independently to allow fast seeking and error recovery. Each GOP starts with an *I-frame*, or intra-coded picture, which is encoded independently of other frames like a static image. Other frames are encoded as inter-coded pictures, and are divided into rectangular blocks. The encoder finds a reference block in previous frames that minimizes their difference. Instead of encoding the pixels directly, the encoder encodes the relative location of the reference block, i.e., the *motion vector*, and the residual. This inter-frame prediction allows encoders to exploit temporal redundancy. Note that the encoder has the freedom to fall back to intra-coding mode for blocks in an inter-coded frame if no reference block is found.

Just like static image compression, the encoder transforms the pixels in I-frames and residuals in inter-coded frames into the frequency domain and encodes the coefficients. The transformation improves the compression rate because high frequency signals are usually few in natural images, and many coefficients will be zero. To further reduce the video size, video codecs also exploit spatial redundancy through intra-prediction, i.e. predicting values to be encoded using previously encoded values. The encoder will encode only the residual between the prediction and real value. This applies to both the motion vector and transformed coefficients encoding. Most of the residuals will be small and can be encoded efficiently using entropy coding. For a more complete survey, see [33].

3.1. Data Preparation

To study the correlation between cubemap orientation and compression rate, we collect a 360° video dataset from YouTube. Existing datasets [41, 19] contain videos with arbitrary quality, many with compression artifacts that could bias the result. Instead, we collect only high quality videos using the 4K filter in YouTube search. We use the keyword "360 video" together with the 360° filter to search for videos and manually filter out those consisting of static images or CG videos. The dataset covers a variety of video content and recording situations, including but not limited to 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 high profile.

We next transcode the video into cubemap format and extract the video size in different orientations. Because it

		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.

is impossible to enumerate all possible cubemap orientations over time, we discretize the problem by dividing the video into 2 second clips and encode each clip independently. This is compliant with the closed GOP structure, except that video codecs usually have the flexibility to adjust the GOP length within a given range. For example, the default x264 encoder limits the GOP length between 25-250 frames, i.e. roughly 1-10 seconds, and a common constraint for Blu-ray videos is 1-2 seconds [9]. 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 the transform360 filter [1] in FFMPEG released by Facebook with 960 pixels resolution for each face. Fig. 2 illustrates the transformation. The video is then encoded using off-the-shelf encoders. We encode the video into three popular formats—H264 using x264 [2], HEVC using x265 [3], and VP9 using libvpx [4]. Among them, H264 is currently the most common video format. HEVC, also known as H265, is the successor of H264 and is the latest video compression standard. VP9 is a competitor of HEVC developed by Google and is most popular in web applications. We use lossless compression for all three formats to ensure rotational symmetry and extract the size of the final encoded bit-stream. See supp. for the exact encoding parameters. Note that we use popular open source tools for both cubemap rendering and video compression to ensure that they are well optimized and tested. This way any size changes we observe can be taken as common in $360°$ video production instead of an artifact of our implementation. The dataset is available on our project webpage[3].

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

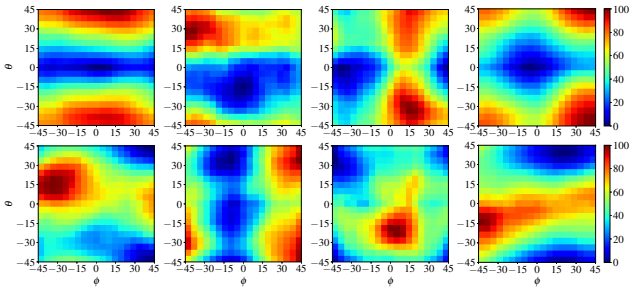


Figure 3: Relative clip size distribution w.r.t. Ω. We cluster the distribution into 16 clusters and show 8 of them.

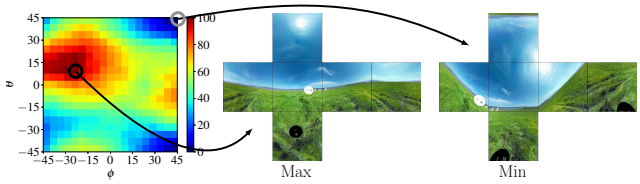


Figure 4: Clip size distribution of a single clip. We also show the cubemaps corresponding to $\Omega_{max}/\Omega_{min}$.

3.2. Data Analysis

Next we investigate how much and why the orientation of an isomer matters for compressibility. If not mentioned specifically, all the results are obtained from H264 fromat.

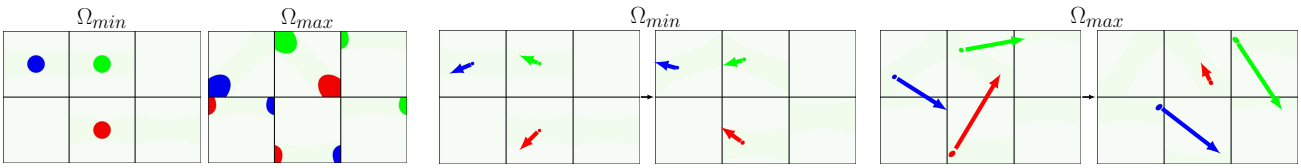
Achievable video size reduction We first examine the size reduction we can achieve by changing the cubemap orientation. In particular, we compute the *reduction*

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

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. For example, 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. If we restrict the rotation to ϕ and fix $\theta = 0°$, $\overline{r}$ will drop to 2.35%. This result suggests that it is important to allow rotation along both axes. Finally we see that the average and range of reductions is quite similar across encoders, indicating that compressibility of isomers is not unique to a particular codec.

Video size distribution w.r.t. Ω We next show the video size distribution with respect to Ω. We compute the *normal-*



(a) Content discontinuity. (b) Motion discontinuity.

Figure 5: Explanations for why different Ω have different compression rates, shown for good (Ω_{min}) and bad (Ω_{max}) rotations. (a) From a static picture perspective, some Ω introduce content discontinuity and reduce spatial redundancy. (b) From a dynamic picture perspective, some Ω make the motion more disordered and break the temporal redundancy.

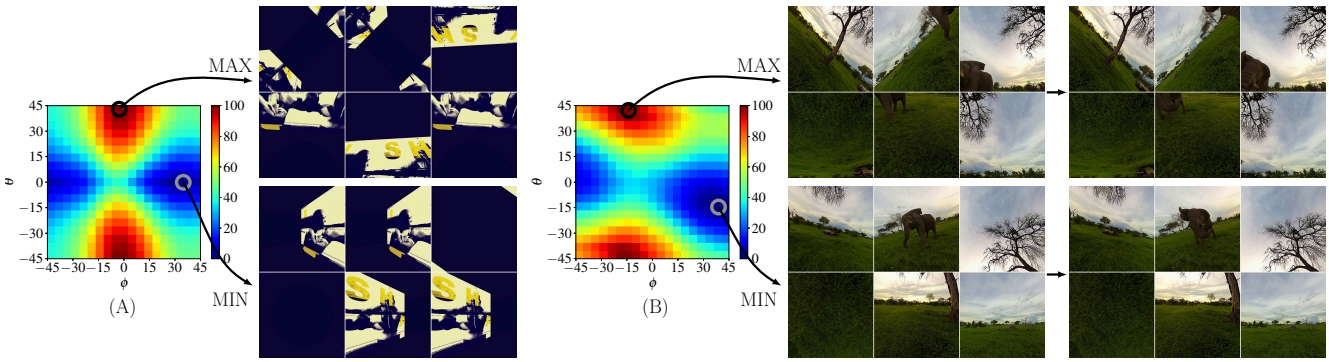


Figure 6: Real examples for the explanations in Fig. 5. (A) shows content discontinuity introduced by rotation. (B) shows motion discontinuity, where the encoder fails to find reference blocks and the number of intra-coded blocks increases.

ized clip size

$$\tilde{S}_\Omega = 100 \times \frac{S_\Omega - S_{\Omega^{min}}}{S_{\Omega^{max}} - S_{\Omega^{min}}} \tag{3}$$

for every Ω and cluster the size distribution of each clip using K-Means. Each cluster is represented by the nearest neighbor to the center.

Fig. 3 shows the results. We can see Ω^{min} lies on or near θ=0° in half the clusters. In general, this corresponds to orienting the cubemap perpendicular to the ground such that the top face captures the sky and the bottom face captures the camera and ground. See Fig. 2 for example. The top and bottom faces tend to have smaller motion within the faces in these orientations, and the compression rate is higher because the problem reduces from compressing six dynamic pictures to four dynamic pictures plus two near static pictures. However, θ=0° is not best for every clip, and there are multiple modes visible in Fig. 3. For example, the minimum size occurs at θ=ϕ=45° in Fig. 4. Therefore, again we see it is important to allow two-dimensional rotations.

Reasons for the compression rate difference Why does the video size depend on Ω? The fundamental reason is that all the video compression formats are designed for perspective images and heavily exploit the image properties. The unwrapped cubemap format is a perspective image only locally within each of the six faces. The cubemap projection introduces perspective distortion near the face boundaries and artificial discontinuities across face boundaries, both of which make the cubemap significantly different from perspective images and can degrade the compression rate. Because the degradation is content dependent, different orientations result in different compression rates.

More specifically, the reasons for the compression rate difference can be divided into two parts. From the static image perspective, artificial edges may be introduced if continuous patterns fall on the face boundary. See Fig. 5 (a) and Fig. 6 for examples. The edges introduce additional high frequency signals and reduce the efficiency of transform coding. Furthermore, the single continuous patch is divided into multiple patches that are dispersed to multiple locations in the image. This reduces the spatial redundancy and breaks the intra-prediction.

From the dynamic video perspective, the face boundaries can introduce abrupt jumps in the motion. If an object moves across the boundary, it may be teleported to a distant location on the image. See Fig. 5 (b) and Fig. 6 for examples. The abrupt motion makes it difficult to find the reference block during encoding, and the encoder may fall back to intra-coding mode which is much less efficient. Even if the encoder successfully finds the reference block, the motion vectors would have very different magnitude and direction compared to those within the faces, which breaks intra-prediction. Finally, because the perspective distortion is location dependent, the same pattern will be distorted differently when it falls on different faces, and the residual of inter-frame prediction may increase. The analysis applies similarly across the three formats, which makes sense, since their compression strategies are broadly similar.

Encoders	H264 / H265	H264 / VP9	H265 / VP9
Avg. ρ	0.8757	0.9533	0.8423

Table 2: The correlation of relative video sizes across video formats. The high correlation indicates that the dependency between video size and Ω is common across formats.

Video size correlation across formats Next we verify the correlation between video size and orientation is not an artifact of the specific video codec. We compare the size reduction that can be achieved through rotation using different encoders (Table 1). The result clearly shows that the dependency between the compression rate and Ω is a common property across current video compression formats. This is further verified by the high correlation between the relative video size, i.e.

$$S'_\Omega = S_\Omega - S_{0,0}, \tag{4}$$

of different encoders in Table 2.

4. Approach

In this section, we introduce our approach for improving 360° video compression rates by predicting the most compressible 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 takes more than 15 seconds to encode one single clip using the default x264 encoder on a 48 core machine with Intel Xeon E5-2697 processor, which corresponds to $15s \times 360 \approx 1.5$ hours for one clip if we try to enumerate Ω. 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. The final prediction of the model is

$$\Omega^{min} = \arg\min_{\Omega} S_\Omega. \tag{5}$$

See Fig. 7. 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.

Because our goal is to find Ω^{min} for a given video clip, exact prediction of S_Ω is not necessary. Instead, the model predicts the relative video size S'_Ω from Eq. 4. The value S'_Ω is scaled to $[0, 100]$ over the entire dataset to facilitate training. We treat it as a regression problem and learn a model that predicts 361 real values using L2 loss as the objective function. Note that we do not predict S_Ω in Eq. 3 because it would amplify the loss for clips with smaller size, which may be harmful for the absolute size reduction.

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 using SLIC [5] and take the segmentation contour map as feature. The segmentation contour represents edges in the frame, 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. Specifically, we sample one motion vector every 8 pixels and take both the forward and backward motion vectors as the feature. Because each motion vector consists of both spatial and temporal displacement, this results in a 6-dimensional feature. For regions without a motion vector, we simply pad 0 for the input regardless of the encoding mode. We concatenate the appearance and motion feature to construct a feature map with depth 7. Because the motion feature map has lower resolution than the video frame, we downscale the appearance feature map by 8 to match the spatial resolution. The input resolution of each face of the cube map is therefore $960/8 = 160$ pixels.

The feature maps for each segment are then fed into a CNN and concatenated together as the video feature. We use the VGG architecture [36] 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, following models for image segmentation [31]. 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.

Aside from predicting S_Ω, in preliminary research we tried other objective functions such as regression for Ω^{min} directly or predicting Ω^{min} from the 361 possible Ω with 361-way classification, but none of them perform as well as the proposed approach. Regressing Ω^{min} often falls back to predicting $(\theta, \phi) = (0, 0)$ because the distribution is symmetric. Treating the problem as 361-way classification has very poor accuracy, i.e., slightly better than random ($\approx$5%),

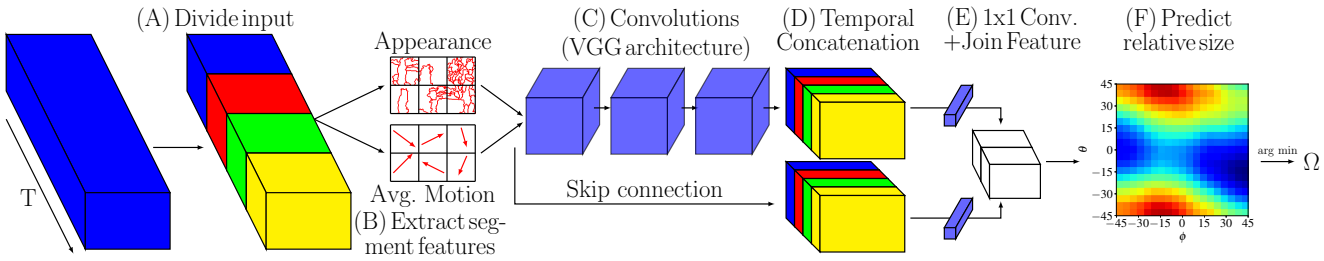


Figure 7: Our model takes a video clip as input and predicts Ω^{min} as output. (A) It first divides the video into 4 segments temporally and (B) extracts appearance and motion features from each segment. (C) It then concatenates the appearance and motion feature maps and feeds them into a CNN. (D) The 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 relative video size S'_Ω for all Ω and takes the minimum one as the predicted optimally compressible isomer.

because the number of training data is small and imbalanced. We also examined different input features. For motion features, we tried 3D convolution instead of explicitly feeding the motion information as input, but 3D convolution performs $4-30\%$ worse than 2D convolution despite having a higher computational cost. For appearance features, we tried raw pixels with various network architectures but find that segmentation contours consistently perform better.

5. Experiments

To evaluate our method, we compute the size reduction it achieves on the 360° video dataset introduced in Sec. 3.

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 $\tilde{r} = 1 - \tilde{S}$ for each video. Specifically, we compute the largest full-video size by choosing Ω^{max} for every clip and sum the clip sizes. Similarly, we compute the minimum video size. Given the predicted orientation for each clip, we compute the video size by rotating the cubemap by the predicted orientation. 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.

Implementation details We initialize the weights using an ImageNet pre-trained VGG model [36]. For the first layer, we replicate the weights of the original network to increase the number of input channels. Weights that are not in the original model are randomly initialized using Xavier initialization [16]. We train the model using ADAM [25] for 4,000 iterations with batch size 64 parallelized to 16 GPUs.

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

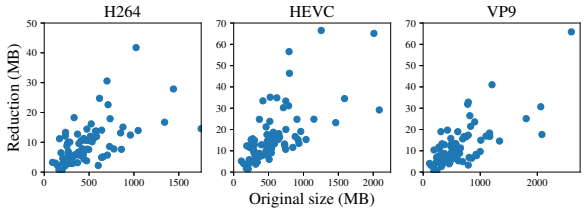


Figure 8: Absolute size reduction (MB) of each video. Each point represents the input video size vs. size reduction relative to CENTER achieved by our model.

The base learning rate is initialized to 1.0×10^{-3} and is decreased by a factor of 10 after 2,000 iterations. We also apply L_2 regularization with the weight set to 5.0×10^{-4} and use dropout for the fully-connected layers with ratio 0.5. For SLIC, we segment each face of the cubemap independently into 256 superpixels with compactness m=1. The low compactness value leads to more emphasis on the color proximity in the superpixels.

5.1. Results

We first examine the size reduction our method achieves. Table 3 shows the results. 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

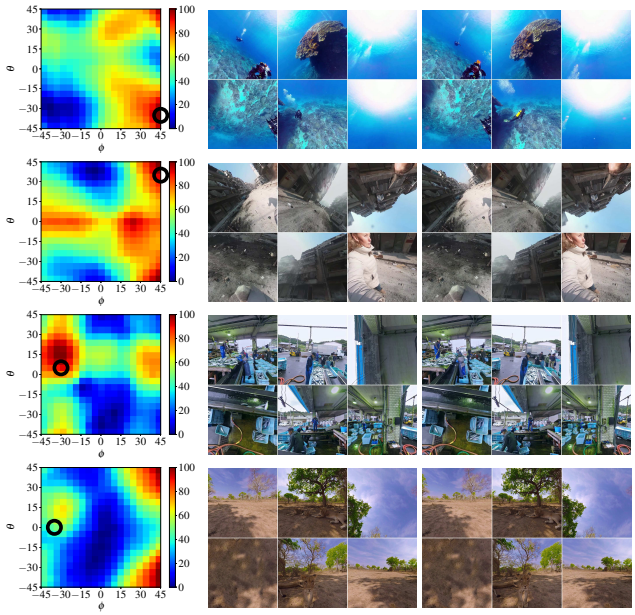


Figure 9: Qualitative examples. The heatmap shows the normalized reduction, and the overlaid circle shows our predicted result. The two images are the first and last frame of the clip rendered in the predicted orientation. Last row shows a failure example. Best viewed in color.

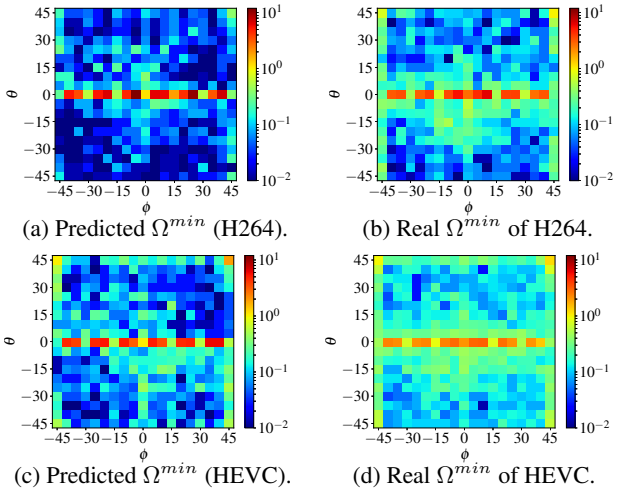
cubemap orientation during compression as more advanced video codecs are used. While there remains a 20% room for improvement compared to the optimal result (as ascertained by enumerating Ω), our approach is significantly faster and takes less than 0.3% the computation. We also show the absolute file size reduction for each video in Fig. 8. Because the video size depends very much on the video content and length and is hard to compare across examples, we show the reduction versus the original video size. 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. 9 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 in Fig. 6); 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.

We next examine whether the model can be transferred across video formats, e.g. can the model trained on H264 videos improve the compression rate of HEVC videos? Table 4 shows the results. Overall, the results show our approach is capable of generalizing across video formats given common features. We find that the model trained on H264 is less transferable, while the models trained on HEVC and VP9 perform fairly well on H264. In particular, the model trained on HEVC performs the best across all formats. The

H264		HEVC		VP9	
HEVC	VP9	H264	VP9	H264	HEVC
70.82	78.17	85.79	84.61	83.19	75.16

Table 4: Size reduction of our approach. Top row indicates training source, second row is test sources.



(a) Predicted Ω^{min} (H264). (b) Real Ω^{min} of H264.

(c) Predicted Ω^{min} (HEVC). (d) Real Ω^{min} of HEVC.

Figure 10: Distribution of Ω^{min} (%). Predictions are on H264 videos with different training data.

reasons are twofold. First, the models trained on HEVC and VP9 focus on the appearance feature which is common across all formats. Second, the models trained on H264 suffer more from overfitting because the distribution of Ω^{min} is more concentrated.

The distribution of Ω^{min} provides further insight into the advantage of the model trained on HEVC. See Fig. 10. The predicted Ω^{min} tend to be more concentrated around θ=0 than the real Ω^{min}. Because the distribution of Ω^{min} is more dispersed in HEVC, so is the prediction of Ω^{min} by the model trained on HEVC.

6. 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. Thanks to Intel for access to their vLab Machine Learning clusters.

References

[1] https://github.com/facebook/transform360. 4

[2] https://www.videolan.org/developers/x264.html. 4

[3] http://x265.org. 4

[4] https://chromium.googlesource.com/webm/libvpx. 4

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012. 6

[6] D. N. Adeel Abbas. A novel projection for omni-directional video. In *Proc.SPIE 10396*, 2017. 3

[7] E. Alshina, K. Choi, V. Zakharchenko, S. N. Akula, A. Dsouza, C. Pujara, K. K. Ramkumaar, and A. Singh. Samsung's response to joint cfe on video compression with capability beyond hevc (360 category). JVET-G0025, 2017. 2

[8] B. Ayrey and C. Wong. Introducing facebook 360 for gear vr. https://newsroom.fb.com/news/2017/03/introducing-facebook-360-for-gear-vr/, March 2017. 1

[9] Blu-ray Disc Association. White paper blu-ray disc read-only format coding constraints on hevc video streams for bd-rom version 3.0, June 2015. 4

[10] 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, 3

[11] C.-H. Chang, M.-C. Hu, W.-H. Cheng, and Y.-Y. Chuang. Rectangling stereographic projection for wide-angle image visualization. In *ICCV*, 2013. 2

[12] B. Choi, Y.-K. Wang, and M. M. Hannuksela. Wd on iso/iec 23000-20 omnidirectional media application format. ISO/IEC JTC1/SC29/WG11, 2017. 2

[13] M. Coban, G. V. der Auwera, and M. Karczewicz. Qualcomms response to joint cfe in 360-degree video category. JVET-G0023, 2017. 2

[14] T. Cohen, M. Geiger, and M. Welling. Convolutional networks for spherical signals. *arXiv preprint arXiv:1709.04893*, 2017. 1, 2

[15] A. Gabriel and E. Thomas. Polyphase subsampling applied to 360-degree video sequences in the context of the joint call for evidence on video compression. JVET-G0026, 2017. 2

[16] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 7

[17] P. Hanhart, X. Xiu, F. Duanmu, Y. He, and Y. Ye. Interdigitals response to the 360 video category in joint call for evidence on video compression with capability beyond hevc. JVET-G0024, 2017. 2

[18] P. Hansen, P. Corke, W. Boles, and K. Daniilidis. Scale-invariant features on the sphere. In *ICCV*, 2007. 2

[19] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360° sports video. In *CVPR*, 2017. 1, 2, 3

[20] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. *arXiv preprint arXiv:1703.10114*, 2017. 3

[21] M. Kamali, A. Banno, J.-C. Bazin, I. S. Kweon, and K. Ikeuchi. Stabilizing omnidirectional videos using 3d structure and spherical image warping. In *IAPR MVA*, 2011. 1, 2

[22] S. Kasahara, S. Nagai, and J. Rekimoto. First person omnidirectional video: System design and implications for immersive experience. In *ACM TVX*, 2015. 1, 2

[23] R. Khasanova and P. Frossard. Graph-based classification of omnidirectional images. *arXiv preprint arXiv:1707.08301*, 2017. 1, 2

[24] Y. Kim, C.-R. Lee, D.-Y. Cho, Y. Kwon, H.-J. Choi, and K.-J. Yoon. Automatic content-aware projection for 360° videos. In *ICCV*, 2017. 2

[25] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7

[26] J. Kopf. 360° video stabilization. *ACM Transactions on Graphics (TOG)*, 35(6):195, 2016. 1, 2

[27] E. Kuzyakov and D. Pio. Under the hood: Building 360 video. https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/, October 2015. 1, 3

[28] 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, 3

[29] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang. Semantic-driven generation of hyperlapse from 360° video. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2017. 1, 2

[30] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. *arXiv preprint arXiv:1703.10553*, 2017. 3

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 6

[32] Moving Picture Experts Group. Point cloud compression mpeg evaluates responses to call for proposal and kicks off its technical work [press release]. https://mpeg.chiariglione.org/meetings/120, October 2017. 1, 3

[33] K. R. Rao, D. N. Kim, and J. J. Hwang. *Video Coding Standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*. Springer Netherlands, 2014. 3

[34] O. Rippel and L. Bourdev. Real-time adaptive image compression. In *ICML*, 2017. 3

[35] S. Santurkar, D. Budden, and N. Shavit. Generative compression. *arXiv preprint arXiv:1703.01467*, 2017. 3

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6, 7

[37] J. P. Snyder. *Map projections–A working manual*, volume 1395. US Government Printing Office, 1987. 2

[38] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *IEEE ISM*, 2016. 2

[39] Y.-C. Su and K. Grauman. Learning spherical convolution for fast features from 360° imagery. In *NIPS*, 2017. 1, 2

[40] Y.-C. Su and K. Grauman. Making 360° video watchable in 2d: Learning videography for click free viewing. In *CVPR*, 2017. 1, 2

[41] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2vid: Automatic cinematography for watching 360° videos. In *ACCV*, 2016. 1, 2, 3

[42] Y. Snchez, R. Skupin, and T. Schierl. Compressed domain video processing for tile based panoramic streaming using hevc. In *ICIP*, 2015. 2

[43] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. In *ICLR*, 2016. 3

[44] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *CVPR*, 2017. 3

[45] V. Ukonaho. Global 360 camera sales forecast by segment: 2016 to 2022. https://www.strategyanalytics.com/access-services/devices/mobile-phones/emerging-devices/market-data/report-detail/global-360-camera-sales-forecast-by-segment-2016-to-2022, March 2017. 1

[46] M. Wien, V. Baroncini, J. Boyce, A. Segall, and T. Suzuki. Joint call for evidence on video compression with capability beyond hevc. JVET-F1002, 2017. 2

[47] L. Zelnik-Manor, G. Peters, and P. Perona. Squaring the circle in panoramas. In *ICCV*, 2005. 2