

# Snap Angle Prediction for 360° Panoramas

Bo Xiong<sup>1</sup> and Kristen Grauman<sup>2</sup>

<sup>1</sup> University of Texas at Austin

<sup>2</sup> Facebook AI Research

bxiong@cs.utexas.edu, grauman@fb.com\*

**Abstract.** 360° panoramas are a rich medium, yet notoriously difficult to visualize in the 2D image plane. We explore how intelligent rotations of a spherical image may enable content-aware projection with fewer perceptible distortions. Whereas existing approaches assume the viewpoint is fixed, intuitively some viewing angles within the sphere preserve high-level objects better than others. To discover the relationship between these optimal *snap angles* and the spherical panorama’s content, we develop a reinforcement learning approach for the cubemap projection model. Implemented as a deep recurrent neural network, our method selects a sequence of rotation actions and receives reward for avoiding cube boundaries that overlap with important foreground objects. We show our approach creates more visually pleasing panoramas while using 5x less computation than the baseline.

**Keywords:** 360° panoramas, content-aware projection, foreground objects

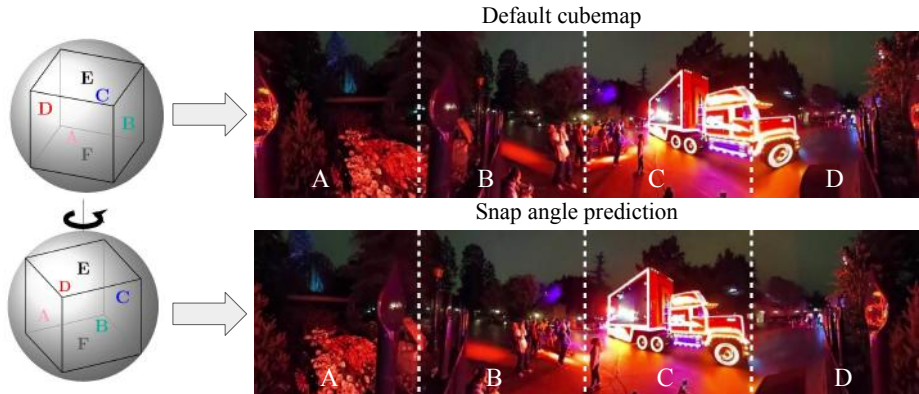
## 1 Introduction

The recent emergence of inexpensive and lightweight 360° cameras enables exciting new ways to capture our visual surroundings. Unlike traditional cameras that capture only a limited field of view, 360° cameras capture the entire visual world from their optical center. Advances in virtual reality technology and promotion from social media platforms like YouTube and Facebook are further boosting the relevance of 360° data.

However, viewing 360° content presents its own challenges. Currently three main directions are pursued: manual navigation, field-of-view (FOV) reduction, and content-based projection. In manual navigation scenarios, a human viewer chooses which normal field-of-view subwindow to observe, e.g., via continuous head movements in a VR headset, or mouse clicks on a screen viewing interface. In contrast, FOV reduction methods generate normal FOV videos by learning to render the most interesting or capture-worthy portions of the viewing sphere [1,2,3,4]. While these methods relieve the decision-making burden of manual navigation, they severely limit the information conveyed by discarding all unselected portions. Projection methods render a wide-angle view, or the entire sphere, onto a single plane (e.g., equirectangular or Mercator) [5] or multiple planes [6]. While they avoid discarding content, any projection inevitably introduces distortions that can be unnatural for viewers. Content-based projection methods can help reduce perceived distortions by prioritizing preservation of straight lines,

---

\* On leave from University of Texas at Austin (grauman@cs.utexas.edu).



**Fig. 1.** Comparison of a cubemap before and after snap angle prediction (dotted lines separate each face). Unlike prior work that assumes a fixed angle for projection, we propose to predict the cube rotation that will best preserve foreground objects in the output. For example, here our method better preserves the truck (third picture C in the second row). We show four (front, right, left, and back) out of the six faces for visualization purposes. Best viewed in color or pdf.

conformality, or other low-level cues [7,8,9], optionally using manual input to know what is worth preserving [10,11,12,13,14].

However, all prior automatic content-based projection methods implicitly assume that the *viewpoint* of the input  $360^\circ$  image is fixed. That is, the spherical image is processed in some default coordinate system, e.g., as the equirectangular projection provided by the camera manufacturer. This assumption limits the quality of the output image. Independent of the content-aware projection eventually used, a fixed viewpoint means some *arbitrary portions of the original sphere will be relegated to places where distortions are greatest*—or at least where they will require most attention by the content-aware algorithm to “undo”.

We propose to eliminate the fixed viewpoint assumption. Our key insight is that an intelligently chosen viewing angle can immediately lessen distortions, even when followed by a conventional projection approach. In particular, we consider the widely used cubemap projection [6,15,16]. A cubemap visualizes the entire sphere by first mapping the sphere to a cube with rectilinear projection (where each face captures a  $90^\circ$  FOV) and then unfolding the faces of the cube. Often, an important object can be projected across two cube faces, destroying object integrity. In addition, rectilinear projection distorts content near cube face boundaries more. See Figure 1, top. However, intuitively, some viewing angles—some cube orientations—are less damaging than others.

We introduce an approach to automatically predict *snap angles*: the rotation of the cube that will yield a set of cube faces that, among all possible rotations, most look like nicely composed human-taken photos originating from the given  $360^\circ$  panoramic image. While what comprises a “well-composed photo” is itself the subject of active research [17,18,19,20,21], we concentrate on a high-level measure of good composition, where the goal is to consolidate each (automatically detected) foreground object within the bounds of one cubemap face. See Figure 1, bottom.

Accordingly, we formalize our snap angle objective in terms of minimizing the spatial mass of foreground objects near cube edges. We develop a reinforcement learning (RL) approach to infer the optimal snap angle given a 360° panorama. We implement the approach with a deep recurrent neural network that is trained end-to-end. The sequence of rotation “actions” chosen by our RL network can be seen as a *learned* coarse-to-fine adjustment of the camera viewpoint, in the same spirit as how people refine their camera’s orientation just before snapping a photo.

We validate our approach on a variety of 360° panorama images. Compared to several informative baselines, we demonstrate that 1) snap angles better preserve important objects, 2) our RL solution efficiently pinpoints the best snap angle, 3) cubemaps unwrapped after snap angle rotation suffer less perceptual distortion than the status quo cubemap, and 4) snap angles even have potential to impact recognition applications, by orienting 360° data in ways that better match the statistics of normal FOV photos used for today’s pretrained recognition networks.

## 2 Related Work

*Spherical image projection* Spherical image projection models project either a limited FOV [7,22] or the entire panorama [5,23,6]. The former group includes rectilinear and Pannini [7] projection; the latter includes equirectangular, stereographic, and Mercator projections (see [5] for a review). Rectilinear and Pannini prioritize preservation of lines in various ways, but always independent of the specific input image. Since any projection of the full sphere must incur distortion, multi-view projections can be perceptually stronger than a single global projection [23]. Cubemap [6], the subject of our snap angle approach, is a multi-view projection method; as discussed above, current approaches simply consider a cubemap in its default orientation.

*Content-aware projection* Built on spherical projection methods, content-based projections make image-specific choices to reduce distortion. Recent work [8] optimizes the parameters in the Pannini projection [7] to preserve regions with greater low-level saliency and straight lines. Interactive methods [10,11,12,13] require a user to outline regions of interest that should be preserved or require input from a user to determine projection orientation [14]. Our approach is content-based and fully automatic. Whereas prior automatic methods assume a fixed viewpoint for projection, we propose to actively predict snap angles for rendering. Thus, our idea is orthogonal to 360° content-aware projection. Advances in the projection method could be applied in concert with our algorithm, e.g., as post-processing to enhance the rotated faces further. For example, when generating cubemaps, one could replace rectilinear projection with others [7,8,10] and keep the rest of our learning framework unchanged. Furthermore, the proposed snap angles respect high-level image content—detected foreground objects—as opposed to typical lower-level cues like line straightness [12,10] or low-level saliency metrics [8].

*Viewing wide-angle panoramas* Since viewing 360° and wide-angle data is non-trivial, there are vision-based efforts to facilitate. The system of [24] helps efficient exploration of gigapixel panoramas. More recently, several systems automatically extract normal FOV videos from 360° video, “piloting” a virtual camera by selecting the viewing angle and/or zoom level most likely to interest a human viewer [1,2,3,4].

*Recurrent networks for attention* Though treating very different problems than ours, multiple recent methods incorporate deep recurrent neural networks (RNN) to make sequential decisions about where to focus attention. The influential work of [25] learns a policy for visual attention in image classification. Active perception systems use RNNs and/or reinforcement learning to select places to look in a novel image [26,27], environment [28,29,30], or video [31,32,33,34] to detect certain objects or activities efficiently. Broadly construed, we share the general goal of efficiently converging on a desired target “view”, but our problem domain is entirely different.

### 3 Approach

We first formalize snap angle prediction as an optimization problem (Sec. 3.1). Then present our learning framework and network architecture for snap angle prediction (Sec. 3.2).

We concentrate on the cubemap projection [6]. Recall that a cubemap maps the sphere to a cube with rectilinear projection (where each face captures a  $90^\circ$  FOV) and then unfolds the six faces of the cube. The unwrapped cube can be visualized as an unfolded box, with the lateral strip of four faces being spatially contiguous in the scene (see Fig. 1, bottom). We explore our idea with cubemaps for a couple reasons. First, a cubemap covers the entire  $360^\circ$  content and does not discard any information. Secondly, each cube face is very similar to a conventional FOV, and therefore relatively easy for a human to view and/or edit.

#### 3.1 Problem Formulation

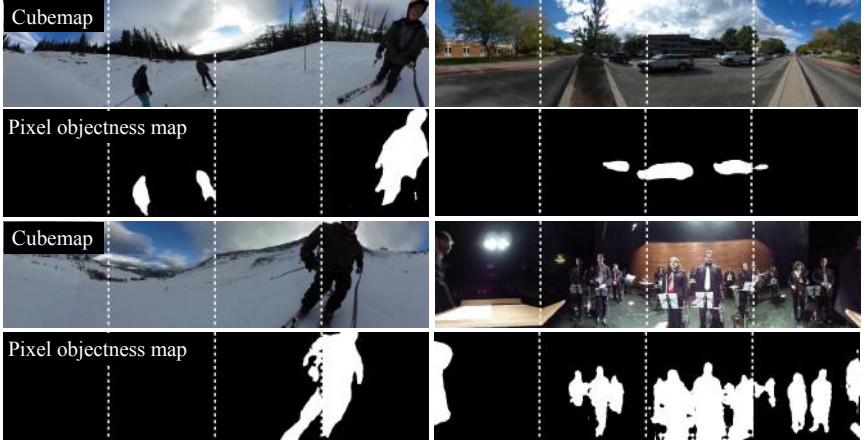
We first formalize snap angle prediction as an optimization problem. Let  $P(I, \theta)$  denote a projection function that takes a panorama image  $I$  and a projection angle  $\theta$  as input and outputs a cubemap after rotating the sphere (or equivalently the cube) by  $\theta$ . Let function  $F$  be an objective function that takes a cubemap as input and outputs a score to measure the quality of the cubemap. Given a novel panorama image  $I$ , our goal is to minimize  $F$  by predicting the snap angle  $\theta^*$ :

$$\theta^* = \underset{\theta}{\operatorname{argmin}} F(P(I, \theta)). \quad (1)$$

The projection function  $P$  first transforms the coordinates of each point in the panorama based on the snap angle  $\theta$  and then produces a cubemap in the standard manner.

Views from a horizontal camera position (elevation  $0^\circ$ ) are more informative than others due to human recording bias. The bottom and top cube faces often align with the sky (above) and ground (below); “stuff” regions like sky, ceiling, and floor are thus common in these faces and foreground objects are minimal. Therefore, rotations in azimuth tend to have greater influence on the disruption caused by cubemap edges. Hence, without loss of generality, we focus on snap angles in azimuth only, and jointly optimize the front/left/right/back faces of the cube.

The coordinates for each point in a panorama can be represented by a pair of latitude and longitude  $(\lambda, \varphi)$ . Let  $L$  denote a coordinate transformation function that takes the



**Fig. 2.** Pixel objectness [35] foreground map examples. White pixels in the pixel objectness map indicate foreground. Our approach learns to find cubemap orientations where the foreground objects are not disrupted by cube edges, i.e., each object falls largely within one face.

snap angle  $\theta$  and a pair of coordinates as input. We define the coordinate transformation function  $L$  as:

$$L((\lambda, \varphi), \theta) = (\lambda, \varphi - \theta). \quad (2)$$

Note when the snap angle is  $90^\circ$ , the orientation of the cube is the same as the default cube except the order of front, back, right, and left is changed. We therefore restrict  $\theta \in [0, \pi/2]$ . We discretize the space of candidate angles for  $\theta$  into a uniform  $N = 20$  azimuths grid, which we found offers fine enough camera control.

We next discuss our choice of the objective function  $F$ . A cubemap in its default orientation has two disadvantages: 1) It does not guarantee to project each important object onto the same cube face; 2) Due to the nature of the perspective projection, objects projected onto cube boundaries will be distorted more than objects in the center. Motivated by these shortcomings, our goal is to produce cubemaps that *place each important object in a single face* and avoid placing objects at the cube boundaries/edges.

In particular, we propose to minimize the area of foreground objects near or on cube boundaries. Supposing each pixel in a cube face is automatically labeled as either object or background, our objective  $F$  measures *the fraction of pixels that are labeled as foreground near cube boundaries*. A pixel is near cube boundaries if it is less than  $A\%$  of the cube length away from the left, right, or top boundary. We do not penalize objects near the bottom boundary since it is common to place objects near the bottom boundary in photography (e.g., portraits).

To infer which pixels belong to the foreground, we use “pixel objectness” [35]. Pixel objectness is a CNN-based foreground estimation approach that returns pixel-wise estimates for all foreground object(s) in the scene, no matter their category. While other foreground methods are feasible (e.g., [36,37,38,39,40]), we choose pixel objectness due to its accuracy in detecting foreground objects of any category, as well as its ability to produce a single pixel-wise foreground map which can contain multiple objects. Figure 2 shows example pixel objectness foreground maps on cube faces. We apply

pixel objectness to a given projected cubemap to obtain its pixel objectness score. In conjunction, other measurements for photo quality, such as interestingness [20], memorability [18], or aesthetics [41], could be employed within  $F$ .

### 3.2 Learning to Predict Snap Angles

On the one hand, a direct regression solution would attempt to infer  $\theta^*$  directly from  $I$ . However, this is problematic because good snap angles can be multi-modal, i.e., available at multiple directions in the sphere, and thus poorly suited for regression. On the other hand, a brute force solution would require projecting the panorama to a cubemap and then evaluating  $F$  for every possible projection angle  $\theta$ , which is costly.

We instead address snap angle prediction with reinforcement learning. The task is a time-budgeted sequential decision process—an iterative adjustment of the (virtual) camera rotation that homes in on the least distorting viewpoint for cubemap projection. Actions are cube rotations and rewards are improvements to the pixel objectness score  $F$ . Loosely speaking, this is reminiscent of how people take photos with a coarse-to-fine refinement towards the desired composition. However, unlike a naive coarse-to-fine search, our approach learns to trigger different search strategies depending on what is observed, as we will demonstrate in results.

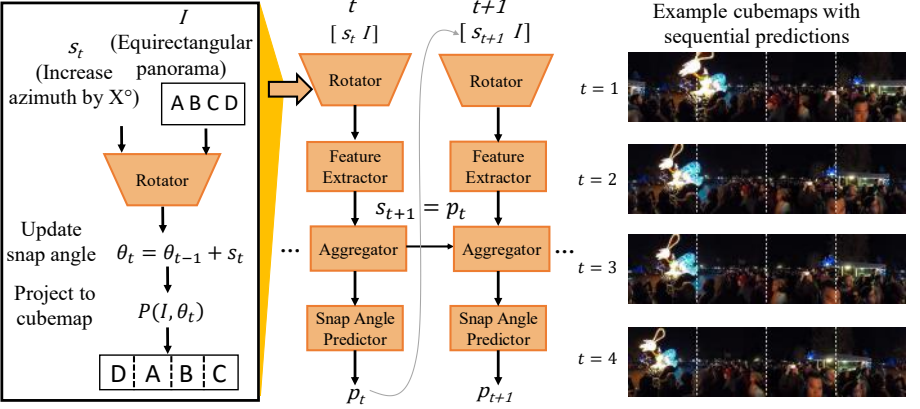
Specifically, let  $T$  represent the budget given to our system, indicating the number of rotations it may attempt. We maintain a history of the model’s previous predictions. At each time step  $t$ , our framework takes a relative snap prediction  $s_t$  (for example,  $s_t$  could signal to update the azimuth by  $45^\circ$ ) and updates its previous snap angle  $\theta_t = \theta_{t-1} + s_t$ . Then, based on its current observation, our system makes a prediction  $p_t$ , which is used to update the snap angle in the next time step. That is, we have  $s_{t+1} = p_t$ . Finally, we choose the snap angle with the lowest pixel objectness objective score from the history as our final prediction  $\hat{\theta}$ :

$$\hat{\theta} = \underset{\theta_t = \theta_1, \dots, \theta_T}{\operatorname{argmin}} F(P(I, \theta_t)). \quad (3)$$

To further improve efficiency, one could compute pixel objectness *once* on a cylindrical panorama rather than recompute it for every cubemap rotation, and then proceed with the iterative rotation predictions above unchanged. However, learned foreground detectors [35,38,37,39,40] are trained on Web images in rectilinear projection, and so their accuracy can degrade with different distortions. Thus we simply recompute the foreground for each cubemap reprojection. See Sec. 4.1 for run-times.

*Network* We implement our reinforcement learning task with deep recurrent and convolutional neural networks. Our framework consists of four modules: a *rotator*, a *feature extractor*, an *aggregator*, and a *snap angle predictor*. At each time step, it processes the data and produces a cubemap (*rotator*), extracts learned features (*feature extractor*), integrates information over time (*aggregator*), and predicts the next snap angle (*snap angle predictor*).

At each time step  $t$ , the *rotator* takes as input a panorama  $I$  in equirectangular projection and a relative snap angle prediction  $s_t = p_{t-1}$ , which is the prediction from the previous time step. The *rotator* updates its current snap angle prediction with  $\theta_t =$



**Fig. 3.** We show the rotator (left), our model (middle), and a series of cubemaps produced by our sequential predictions (right). Our method iteratively refines the best snap angle, targeting a given budget of allowed computation.

$\theta_{t-1} + s_t$ . We set  $\theta_1 = 0$  initially. Then the *rotator* applies the projection function  $P$  to  $I$  based on  $\theta_t$  with Eq 2 to produce a cubemap. Since our objective is to minimize the total amount of foreground straddling cube face boundaries, it is more efficient for our model to learn directly from the pixel objectness map than from raw pixels. Therefore, we apply pixel objectness [35] to each of the four lateral cube faces to obtain a binary objectness map per face. The rotator has the form:  $\mathbb{I}^{W \times H \times 3} \times \Theta \rightarrow \mathbb{B}^{W_c \times W_c \times 4}$ , where  $W$  and  $H$  are the width and height of the input panorama in equirectangular projection and  $W_c$  denotes the side length of a cube face. The *rotator* does not have any learnable parameters since it is used to preprocess the input data.

At each time step  $t$ , the *feature extractor* then applies a sequence of convolutions to the output of the *rotator* to produce a feature vector  $f_t$ , which is then fed into the *aggregator* to produce an aggregate feature vector  $a_t = A(f_1, \dots, f_t)$  over time. Our *aggregator* is a recurrent neural network (RNN), which also maintains its own hidden state.

Finally, the *snap angle predictor* takes the aggregate feature vector as input, and produces a relative snap angle prediction  $p_t$ . In the next time step  $t + 1$ , the relative snap angle prediction is fed into the *rotator* to produce a new cubemap. The *snap angle predictor* contains two fully connected layers, each followed by a ReLU, and then the output is fed into a softmax function for the  $N$  azimuth candidates. The  $N$  candidates here are relative, and range from decreasing azimuth by  $\frac{N}{2}$  to increasing azimuth by  $\frac{N}{2}$ . The *snap angle predictor* first produces a multinomial probability density function  $\pi(p_t)$  over all candidate relative snap angles, then it samples one snap angle prediction proportional to the probability density function. See Figure 3 for an overview of the network, and Supp. for all architecture details.

**Training** The parameters of our model consist of parameters of the *feature extractor*, *aggregator*, and *snap angle predictor*:  $w = \{w_f, w_a, w_p\}$ . We learn them to maximize the total reward (defined below) our model can expect when predicting snap angles. The *snap angle predictor* contains stochastic units and therefore cannot be trained with the

standard backpropagation method. We therefore use REINFORCE [42]. Let  $\pi(p_t|I, w)$  denote the parameterized policy, which is a pdf over all possible snap angle predictions. REINFORCE iteratively increases weights in the pdf  $\pi(p_t|I, w)$  on those snap angles that have received higher rewards. Formally, given a batch of training data  $\{I_i : i = 1, \dots, M\}$ , we can approximate the gradient as follows:

$$\sum_{i=1}^M \sum_{t=1}^T \nabla_w \log \pi(p_t^i|I_i, w) R_t^i \quad (4)$$

where  $R_t^i$  denotes the reward at time  $t$  for instance  $i$ .

*Reward* At each time step  $t$ , we compute the objective. Let  $\hat{\theta}_t = \operatorname{argmin}_{\theta=\theta_1, \dots, \theta_t} F(P(I, \theta))$  denote the snap angle with the lowest pixel objectness until time step  $t$ . Let  $O_t = F(P(I, \hat{\theta}_t))$  denote its corresponding objective value. The reward for time step  $t$  is

$$\hat{R}_t = \min(O_t - F(P(I, \theta_t + p_t)), 0). \quad (5)$$

Thus, the model receives a reward proportional to the decrease in edge-straddling foreground pixels whenever the model updates the snap angle. To speed up training, we use a variance-reduced version of the reward  $R_t = \hat{R}_t - b_t$  where  $b_t$  is the average amount of decrease in pixel objectness coverage with a random policy at time  $t$ .

## 4 Results

Our results address **four main questions**: 1) How efficiently can our approach identify the best snap angle? (Sec. 4.1); 2) To what extent does the foreground “pixel objectness” objective properly capture objects important to human viewers? (Sec. 4.2); 3) To what extent do human viewers favor snap-angle cubemaps over the default orientation? (Sec. 4.3); and 4) Might snap angles aid image recognition? (Sec. 4.4).

*Dataset* We collect a dataset of 360° images to evaluate our approach; existing 360° datasets are typically narrow [43,1,3], restricting their use for our goal. We use YouTube with the 360° filter to gather videos from four activity categories—Disney, Ski, Parade, and Concert. After manually filtering out frames with only text or blackness, we have 150 videos and 14,076 total frames sampled at 1 FPS.

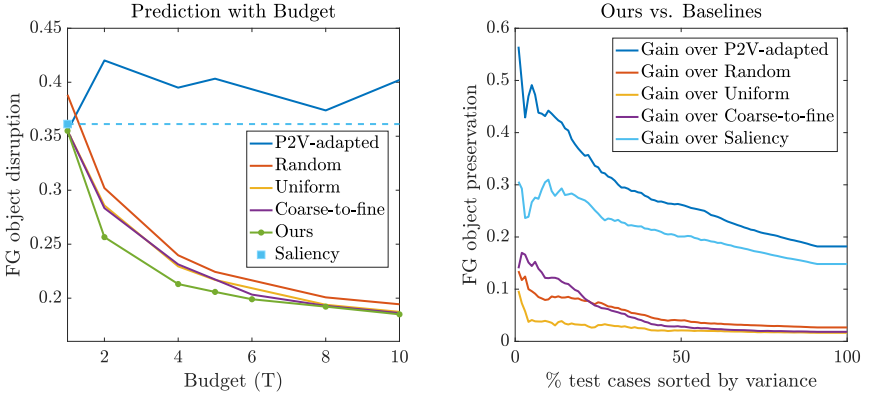
*Implementation details* We implement our model with Torch, and optimize with stochastic gradient and REINFORCE. We set the base learning rate to 0.01 and use momentum. We fix  $A = 6.25\%$  for all results after visual inspection of a few human-taken cubemaps (not in the test set). See Supp. for all network architecture details.

### 4.1 Efficient Snap Angle Prediction

We first evaluate our snap angle prediction framework. We use all 14,076 frames, 75% for training and 25% for testing. We ensure testing and training data do *not* come from the same video. We define the following baselines:

- RANDOM ROTATE: Given a budget  $T$ , predict  $T$  snap angles randomly (with no repetition).





**Fig. 4.** Predicting snap angles in a timely manner. Left: Given a budget, our method predicts snap angles with the least foreground disruption on cube edges. Gains are larger for smaller budgets, demonstrating our method’s efficiency. Right: Our gain over the baselines (for a budget  $T = 4$ ) as a function of the test cases’ decreasing “difficulty”, i.e., the variance in ground truth quality for candidate angles. See text.

- **UNIFORM ROTATE:** Given a budget  $T$ , predict  $T$  snap angles uniformly sampled from all candidates. When  $T = 1$ , UNIFORM receives the CANONICAL view. This is a strong baseline since it exploits the human recording bias in the starting view. Despite the 360° range of the camera, photographers still tend to direct the “front” of the camera towards interesting content, in which case CANONICAL has some manual intelligence built-in.
- **COARSE-TO-FINE SEARCH:** Divide the search space into two uniform intervals and search the center snap angle in each interval. Then recursively search the better interval, until the budget is exhausted.
- **PANO2VID(P2V) [1]-ADAPTED:** We implement a snap angle variant inspired by the pipeline of Pano2Vid [1]. We replace C3D [44] features (which require video) used in [1] with F7 features from VGG [45] and train a logistic classifier to learn “capture-worthiness” [1] with Web images and randomly sampled panorama sub-views (see Supp.). For a budget  $T$ , we evaluate  $T$  “glimpses” and choose the snap angle with the highest encountered capture-worthiness score. We stress that Pano2Vid addresses a different task: it creates a normal field-of-view video (discarding the rest) whereas we create a well-oriented omnidirectional image. Nonetheless, we include this baseline to test their general approach of learning a framing prior from human-captured data.
- **SALIENCY:** Select the angle that centers a cube face around the maximal saliency region. Specifically, we compute the panorama’s saliency map [40] in equirectangular form and blur it with a Gaussian kernel. We then identify the  $P \times P$  pixel square with the highest total saliency value, and predict the snap angle as the center of the square. Unlike the other methods, this baseline is not iterative, since the maximal saliency region does not change with rotations. We use a window size  $P = 30$ . Performance is not sensitive to  $P$  for  $20 \leq P \leq 200$ .

We train our approach for a spectrum of budgets  $T$ , and report results in terms of the amount of foreground disruption as a function of the budget. Each unit of the budget corresponds to one round of rotating, re-rendering, and predicting foregrounds. We score foreground disruption as the average  $F(P(I, \theta_t^*))$  across all four faces.

Figure 4 (left) shows the results. Our method achieves the least disruptions to foreground regions among all the competing methods. UNIFORM ROTATE and COARSE-TO-FINE SEARCH perform better than RANDOM because they benefit from hand-designed search heuristics. Unlike UNIFORM ROTATE and COARSE-TO-FINE SEARCH, our approach is content-based and learns to trigger different search strategies depending on what it observes. When  $T = 1$ , SALIENCY is better than RANDOM but it underperforms our method and UNIFORM. SALIENCY likely has difficulty capturing important objects in panoramas, since the saliency model is trained with standard field-of-view images. Directly adapting PANO2VID [1] for our problem results in unsatisfactory results. A capture-worthiness classifier [1] is relatively insensitive to the placement of important objects/people and therefore less suitable for the snap angle prediction task, which requires detailed modeling of object placement on *all* faces of the cube.

Figure 4 (right) plots our gains sorted by the test images’ decreasing “difficulty” for a budget  $T = 4$ . In some test images, there is a high variance, meaning certain snap angles are better than others. However, for others, all candidate rotations look similarly good, in which case all methods will perform similarly. The righthand plot sorts the test images by their variance (in descending order) in quality across all possible angles, and reports our method’s gain as a function of that difficulty. Our method outperforms P2V-ADAPTED, SALIENCY, COARSE-TO-FINE SEARCH, RANDOM and UNIFORM by up to 56%, 31%, 17%, 14% and 10% (absolute), respectively. Overall Figure 4 demonstrates that our method predicts the snap angle more efficiently than the baselines.

We have thus far reported efficiency in terms of abstract budget usage. One unit of budget entails the following: projecting a typical panorama of size  $960 \times 1920$  pixels in equirectangular form to a cubemap (8.67 seconds with our Matlab implementation) and then computing pixel objectness (0.57 seconds). Our prediction method is very efficient and takes 0.003 seconds to execute for a budget  $T = 4$  with a GeForce GTX 1080 GPU. Thus, for a budget  $T = 4$ , the savings achieved by our method is approximately 2.4 minutes (5x speedup) per image compared to exhaustive search. Note that due to our method’s efficiency, even if the Matlab projections were 1000x faster for all methods, our 5x speedup over the baseline would remain the same. Our method achieves a good tradeoff between speed and accuracy.

## 4.2 Justification for Foreground Object Objective

Next we justify empirically the pixel objectness cube-edge objective. To this end, we have human viewers identify important objects in the source panoramas, then evaluate to what extent our objective preserves them.

Specifically, we randomly select 340 frames among those where: 1) Each frame is at least 10-seconds apart from the rest in order to ensure diversity in the dataset; 2) The difference in terms of overall pixel objectness between our method and the canonical view method is non-negligible. We collect annotations via Amazon Mechanical Turk. Following the interface of [3], we present crowdworkers the panorama and instruct

	CANONICAL	RANDOM	SALIENCY	P2V-ADAPTED	OURS	UPPERBOUND
Concert	77.6%	73.9%	76.2%	71.6%	<b>81.5%</b>	86.3%
Ski	64.1%	72.5%	68.1%	70.1%	<b>78.6%</b>	83.5%
Parade	84.0%	81.2%	86.3%	85.7%	<b>87.6%</b>	96.8%
Disney	58.3%	57.7%	60.8%	60.8%	<b>65.5%</b>	77.4%
All	74.4%	74.2%	76.0%	75.0%	<b>81.1%</b>	88.3%

**Table 1.** Performance on preserving the integrity of objects explicitly identified as important by human observers. Higher overlap scores are better. Our method outperforms all baselines.

them to label any “important objects” with a bounding box—as many as they wish. See Supp. for interface and annotation statistics.

Here we consider PANO2VID(P2V) [1]-ADAPTED and SALIENCY as defined in Sec. 4.1 and two additional baselines: 1) CANONICAL VIEW: produces a cubemap using the camera-provided orientation; 2) RANDOM VIEW: rotates the input panorama by an arbitrary angle and then generates the cubemap. Note that the other baselines in Sec. 4.1 are not applicable here, since they are search mechanisms.

Consider the cube face  $X$  that contains the largest number of foreground pixels from a given bounding box after projection. We evaluate the cubemaps of our method and the baselines based on the overlap score (IoU) between the foreground region from the cube face  $X$  and the corresponding human-labeled important object, for each bounding box. This metric is maximized when all pixels for the same object project to the same cube face; higher overlap indicates better preservation of important objects.

Table 1 shows the results. Our method outperforms all baselines by a large margin. This supports our hypothesis that avoiding foreground objects along the cube edges helps preserve objects of interest to a viewer. Snap angles achieve this goal much better than the baseline cubemaps. The UPPERBOUND corresponds to the maximum possible overlap achieved if exhaustively evaluating *all* candidate angles, and helps gauge the difficulty of each category. Parade and Disney have the highest and lowest upper bounds, respectively. In Disney images, the camera is often carried by the recorders, so important objects/persons appear relatively large in the panorama and cannot fit in a single cube face, hence a lower upper bound score. On the contrary, in Parade images the camera is often placed in the crowd and far away from important objects, so each can be confined to a single face. The latter also explains why the baselines do best (though still weaker than ours) on Parade images. An ablation study decoupling the pixel objectness performance from snap angle performance pinpoints the effects of foreground quality on our approach (see Supp.).

### 4.3 User Study: Perceived Quality

Having justified the perceptual relevance of the cube-edge foreground objective (Sec. 4.2), next we perform a user study to gauge perceptual quality of our results. Do snap angles produce cube faces that look like human-taken photos? We evaluate on the same image set used in Sec. 4.2.

We present cube faces produced by our method and one of the baselines at a time in arbitrary order and inform subjects the two sets are photos from the same scene but taken by different photographers. We instruct them to consider composition and view-point in order to decide which set of photos is more pleasing (see Supp.). To account for

	Prefer OURS	Tie	Prefer CANONICAL	Prefer OURS	Tie	Prefer RANDOM
Parade	54.8%	16.5%	28.7%	70.4%	9.6%	20.0%
Concert	48.7%	16.2%	35.1%	52.7%	16.2%	31.1%
Disney	44.8%	17.9%	37.3%	72.9%	8.5%	18.6%
Ski	64.3%	8.3%	27.4%	62.9%	16.1%	21.0%
All	53.8%	14.7%	31.5%	65.3%	12.3%	22.4%

**Table 2.** User study result comparing cubemaps outputs for perceived quality. Left: Comparison between our method and CANONICAL. Right: Comparison between our method and RANDOM.

	Concert	Ski	Parade	Disney	All (normalized)
Image Memorability [21]					
CANONICAL	<b>71.58</b>	69.49	67.08	70.53	46.8%
RANDOM	71.30	69.54	67.27	70.65	48.1%
SALIENCY	71.40	69.60	67.35	70.58	49.9%
P2V-ADAPTED	71.34	69.85	67.44	70.54	52.1%
OURS	71.45	<b>70.03</b>	<b>67.68</b>	<b>70.87</b>	<b>59.8%</b>
UPPER	72.70	71.19	68.68	72.15	–
Image Aesthetics [17]					
CANONICAL	33.74	41.95	30.24	32.85	44.3%
RANDOM	32.46	41.90	30.65	32.79	42.4%
SALIENCY	34.52	41.87	30.81	32.54	47.9%
P2V-ADAPTED	34.48	41.97	30.86	<b>33.09</b>	48.8%
OURS	<b>35.05</b>	<b>42.08</b>	<b>31.19</b>	32.97	<b>52.9%</b>
UPPER	38.45	45.76	34.74	36.81	–

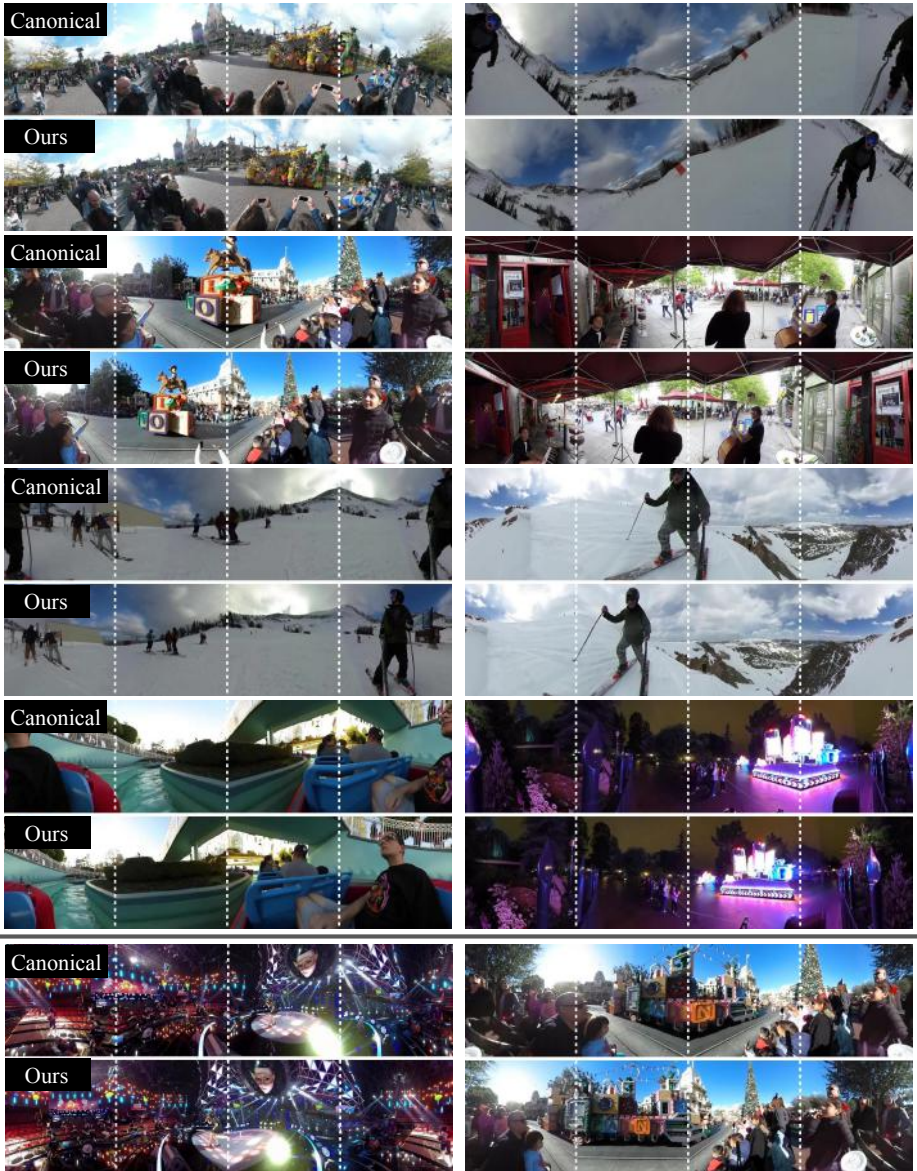
**Table 3.** Memorability and aesthetics scores.

the subjectivity of the task, we issue each sample to 5 distinct workers and aggregate responses with majority vote. 98 unique MTurk crowdworkers participated in the study.

Table 2 shows the results. Our method outperforms the CANONICAL baseline by more than 22% and the RANDOM baseline by 42.9%. This result supports our claim that by preserving object integrity, our method produces cubemaps that align better with human perception of quality photo composition. Figure 5 shows qualitative examples. As shown in the first two examples (top two rows), our method is able to place an important person in the same cube face whereas the baseline splits each person and projects a person onto two cube faces. We also present two failure cases in the last two rows. In the bottom left, pixel objectness does not recognize the stage as foreground, and therefore our method places the stage on two different cube faces, creating a distorted heart-shaped stage. Please see Supp. for pixel objectness map input for failure cases.

So far, Table 1 confirms empirically that our foreground-based objective does preserve those objects human viewers deem important, and Table 2 shows that human viewers have an absolute preference for snap angle cubemaps over other projections. As a final test of snap angle cubemaps’ perceptual quality, we score them using state-of-the-art metrics for *aesthetics* [17] and *memorability* [21]. Since both models are trained on images annotated by people (for their aesthetics and memorability, respectively), higher scores indicate higher correlation with these perceived properties (though of course no one learned metric can perfectly represent human opinion).

Table 3 shows the results. We report the raw scores  $s$  per class as well as the score over all classes, normalized as  $\frac{s-s_{min}}{s_{max}-s_{min}}$ , where  $s_{min}$  and  $s_{max}$  denote the lower and upper bound, respectively. Because the metrics are fairly tolerant to local rotations, there is a limit to how well they can capture subtle differences in cubemaps. Nonethe-



**Fig. 5.** Qualitative examples of default CANONICAL cubemaps and our snap angle cubemaps. Our method produces cubemaps that place important objects/persons in the same cube face to preserve the foreground integrity. Bottom two rows show failure cases. In the bottom left, pixel objectness [35] does not recognize the round stage as foreground, and therefore our method splits the stage onto two different cube faces, creating a distorted heart-shaped stage. In the bottom right, the train is too large to fit in a single cube.

	CANONICAL	RANDOM	OURS
Single	68.5	69.4	<b>70.1</b>
Pano	66.5	67.0	<b>68.1</b>

**Table 4.** Image recognition accuracy (%). Snap angles help align the 360° data’s statistics with that of normal FOV Web photos, enabling easier transfer from conventional pretrained networks.

less, our method outperforms the baselines overall. Given these metrics’ limitations, the user study in Table 2 offers the most direct and conclusive evidence for snap angles’ perceptual advantage.

#### 4.4 Cubemap Recognition from Pretrained Nets

Since snap angles provide projections that better mimic human-taken photo composition, we hypothesize that they also align better with conventional FOV images, compared to cubemaps in their canonical orientation. This suggests that snap angles may better align with Web photos (typically used to train today’s recognition systems), which in turn could help standard recognition models perform well on 360° panoramas. We present a preliminary proof-of-concept experiment to test this hypothesis.

We train a multi-class CNN classifier to distinguish the four activity categories in our 360° dataset (Disney, Parade, etc.). The classifier uses ResNet-101 [46] pretrained on ImageNet [47] and fine-tuned on 300 training images per class downloaded from Google Image Search (see Supp.). Note that in all experiments until now, the category labels on the 360° dataset were invisible to our algorithm. We randomly select 250 panoramas per activity as a test set. Each panorama is projected to a cubemap with the different projection methods, and we compare the resulting recognition rates.

Table 4 shows the results. We report recognition accuracy in two forms: *Single*, which treats each individual cube face as a test instance, and *Pano*, which classifies the entire panorama by multiplying the predicted posteriors from all cube faces. For both cases, snap angles produce cubemaps that achieve the best recognition rate. This result hints at the potential for snap angles to be a bridge between pretrained normal FOV networks on the one hand and 360° images on the other hand. That said, the margin is slim, and the full impact of snap angles for recognition warrants further exploration.

## 5 Conclusions

We introduced the snap angle prediction problem for rendering 360° images. In contrast to previous work that assumes either a fixed or manually supplied projection angle, we propose to automatically predict the angle that will best preserve detected foreground objects. We present a framework to efficiently and accurately predict the snap angle in novel panoramas. We demonstrate the advantages of the proposed method, both in terms of human perception and several statistical metrics. Future work will explore ways to generalize snap angles to video data and expand snap angle prediction to other projection models.

*Acknowledgements* This research is supported in part by NSF IIS-1514118 and a Google Faculty Research Award. We also gratefully acknowledge a GPU donation from Facebook.

## References

1. Su, Y.C., Jayaraman, D., Grauman, K.: Pano2vid: Automatic cinematography for watching 360 videos. In: ACCV. (2016)
2. Su, Y.C., Grauman, K.: Making 360 video watchable in 2d: Learning videography for click free viewing. In: CVPR. (2017)
3. Hu, H.N., Lin, Y.C., Liu, M.Y., Cheng, H.T., Chang, Y.J., Sun, M.: Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos. In: CVPR. (2017)
4. Lai, W.S., Huang, Y., Joshi, N., Buehler, C., Yang, M.H., Kang, S.B.: Semantic-driven generation of hyperlapse from 360 video. *IEEE Transactions on Visualization and Computer Graphics* (2017)
5. Snyder, J.P.: Flattening the earth: two thousand years of map projections. University of Chicago Press (1997)
6. Greene, N.: Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications* (1986)
7. Sharpless, T.K., Postle, B., German, D.M.: Pannini: a new projection for rendering wide angle perspective images. In: International Conference on Computational Aesthetics in Graphics, Visualization and Imaging. (2010)
8. Kim, Y.W., Jo, D.Y., Lee, C.R., Choi, H.J., Kwon, Y.H., Yoon, K.J.: Automatic content-aware projection for 360 videos. In: ICCV. (2017)
9. Li, D., He, K., Sun, J., Zhou, K.: A geodesic-preserving method for image warping. In: CVPR. (2015)
10. Carroll, R., Agrawala, M., Agarwala, A.: Optimizing content-preserving projections for wide-angle images. In: *ACM Transactions on Graphics*. (2009)
11. Tehrani, M.A., Majumder, A., Gopi, M.: Correcting perceived perspective distortions using object specific planar transformations. In: ICCP. (2016)
12. Carroll, R., Agarwala, A., Agrawala, M.: Image warps for artistic perspective manipulation. In: *ACM Transactions on Graphics*. (2010)
13. Kopf, J., Lischinski, D., Deussen, O., Cohen-Or, D., Cohen, M.: Locally adapted projections to reduce panorama distortions. In: *Computer Graphics Forum, Wiley Online Library* (2009)
14. Wang, Z., Jin, X., Xue, F., He, X., Li, R., Zha, H.: Panorama to cube: a content-aware representation method. In: *SIGGRAPH Asia Technical Briefs*. (2015)
15. <https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/>
16. <https://www.blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/>
17. Kong, S., Shen, X., Lin, Z., Mech, R., Fowlkes, C.: Photo aesthetics ranking network with attributes and content adaptation. In: ECCV. (2016)
18. Isola, P., Xiao, J., Torralba, A., Oliva, A.: What makes an image memorable? In: CVPR. (2011)
19. Xiong, B., Grauman, K.: Detecting snap points in egocentric video with a web photo prior. In: ECCV. (2014)
20. Gygli, M., Grabner, H., Riemenschneider, H., Nater, F., Van Gool, L.: The interestingness of images. In: ICCV. (2013)
21. Khosla, A., Raju, A.S., Torralba, A., Oliva, A.: Understanding and predicting image memorability at a large scale. In: ICCV. (2015)
22. Chang, C.H., Hu, M.C., Cheng, W.H., Chuang, Y.Y.: Rectangling stereographic projection for wide-angle image visualization. In: ICCV. (2013)
23. Zelnik-Manor, L., Peters, G., Perona, P.: Squaring the circle in panoramas. In: ICCV. (2005)

24. Kopf, J., Uyttendaele, M., Deussen, O., Cohen, M.F.: Capturing and viewing gigapixel images. In: *ACM Transactions on Graphics*. (2007)
25. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: *NIPS*. (2014)
26. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: *ICCV*. (2015)
27. Mathe, S., Pirinen, A., Sminchisescu, C.: Reinforcement learning for visual object detection. In: *CVPR*. (2016)
28. Jayaraman, D., Grauman, K.: Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In: *ECCV*. (2016)
29. Jayaraman, D., Grauman, K.: Learning to look around: Intelligently exploring unseen environments for unknown tasks. In: *CVPR*. (2018)
30. Jayaraman, D., Grauman, K.: End-to-end policy learning for active visual categorization. *PAMI* (2018)
31. Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: *CVPR*. (2016)
32. Alwassel, H., Heilbron, F.C., Ghanem, B.: Action search: Learning to search for human activities in untrimmed videos. *arXiv preprint arXiv:1706.04269* (2017)
33. Singh, B., Marks, T.K., Jones, M., Tuzel, O., Shao, M.: A multi-stream bi-directional recurrent neural network for fine-grained action detection. In: *CVPR*. (2016)
34. Su, Y.C., Grauman, K.: Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video. In: *ECCV*. (2016)
35. Xiong, B., Jain, S.D., Grauman, K.: Pixel objectness: Learning to segment generic objects automatically in images and videos. *PAMI* (2018)
36. Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: *ECCV*. (2014)
37. Carreira, J., Sminchisescu, C.: CPMC: Automatic object segmentation using constrained parametric min-cuts. *PAMI* (2011)
38. Jiang, P., Ling, H., Yu, J., Peng, J.: Salient region detection by ufo: Uniqueness, focusness, and objectness. In: *ICCV*. (2013)
39. Pinheiro, P.O., Collobert, R., Dollár, P.: Learning to segment object candidates. In: *NIPS*. (2015)
40. Liu, T., Yuan, Z., Sun, J., Wang, J., Zheng, N., Tang, X., Shum, H.Y.: Learning to detect a salient object. *PAMI* (2011)
41. Dhar, S., Ordonez, V., Berg, T.L.: High level describable attributes for predicting aesthetics and interestingness. In: *CVPR*. (2011)
42. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* (1992)
43. Xiao, J., Ehinger, K.A., Oliva, A., Torralba, A.: Recognizing scene viewpoint using panoramic place representation. In: *CVPR*. (2012)
44. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: *ICCV*. (2015)
45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014)
46. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *CVPR*. (2016)
47. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. *IJCV* (2015)
48. <https://github.com/jianxiongxiao/ProfXkit>



49. Liu, T., Yuan, Z., Sun, J., Wang, J., Zheng, N., Tang, X., Shum, H.Y.: Learning to detect a salient object. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2011)
50. Jiang, B., Zhang, L., Lu, H., Yang, C., Yang, M.H.: Saliency detection via absorbing markov chain. In: *ICCV*. (2013)

## 6 Supplementary Material

### 6.1 Justification for predicting snap angle in azimuth only

This section accompanies Sec. 3.1 in the main paper.

As discussed in the paper, views from a horizontal camera position (elevation  $0^\circ$ ) are typically more plausible than other elevations due to the human recording bias. The human photographer typically holds the camera with the “top” to the sky/ceiling.

Figure 6 shows examples of rotations in elevation for several panoramas. We see that the recording bias makes the 0 (canonical) elevation fairly good as it is. In contrast, rotation in elevation often makes the cubemaps appear tilted (e.g., the building in the second row). Without loss of generality, we focus on snap angles in azimuth only, and jointly optimize the front/left/right/back faces of the cube.

### 6.2 Details on network architecture

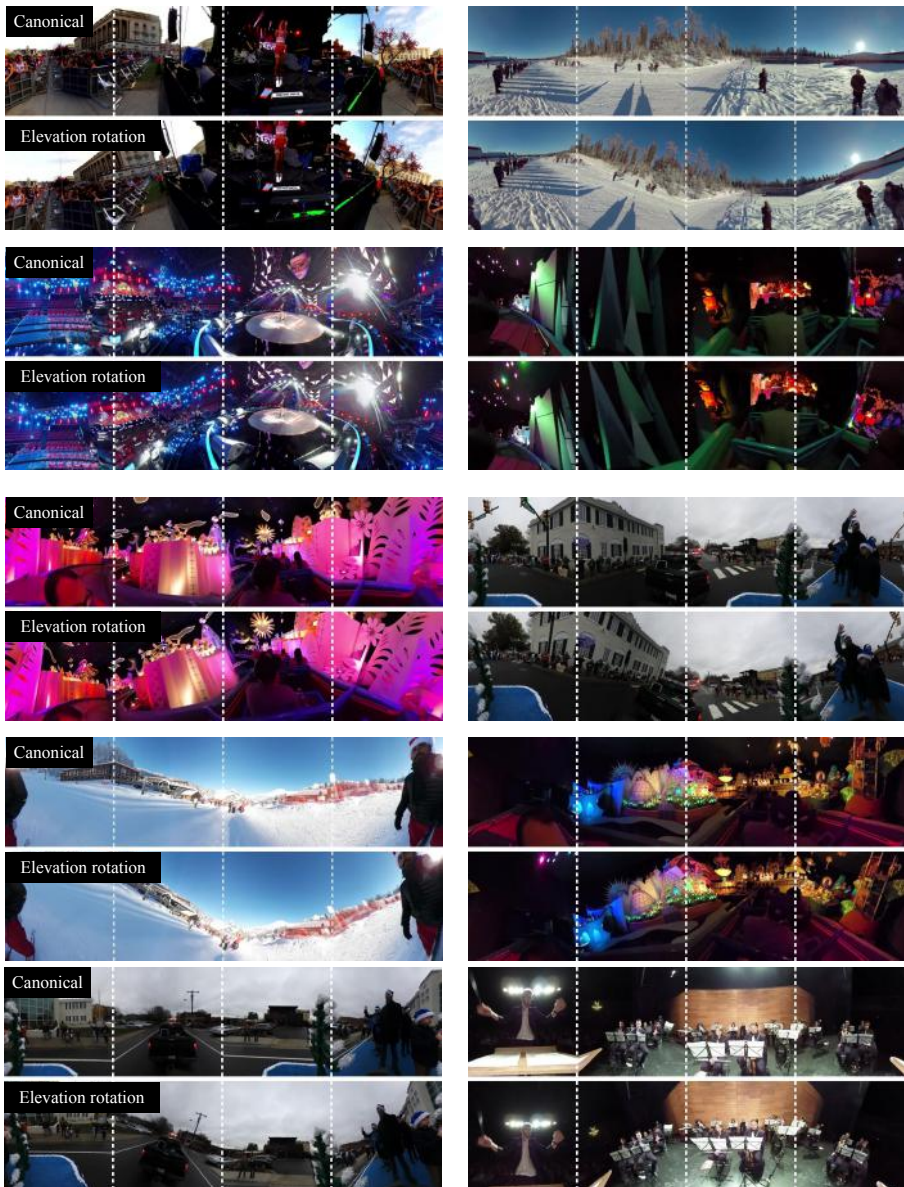
This section accompanies Sec. 3.2 in the main paper.

Recall that our framework consists of four modules: a *rotator*, a *feature extractor*, an *aggregator*, and a *snap angle predictor*. At each time step, it processes the data and produces a cubemap (*rotator*), extracts learned features (*feature extractor*), integrates information over time (*aggregator*), and predicts the next snap angle (*snap angle predictor*). We show the details of each module in Figure 7.

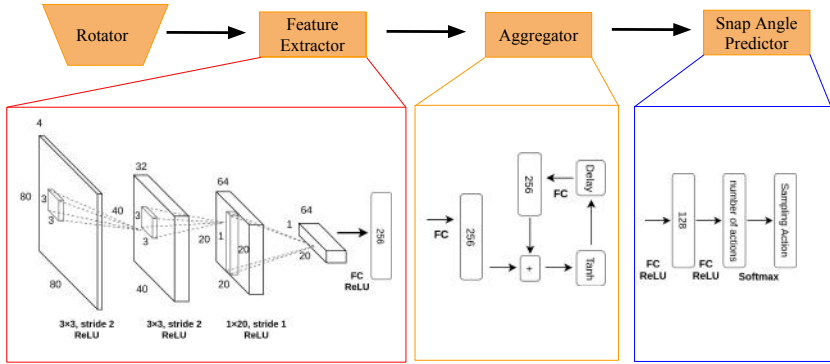
### 6.3 Training details for PANO2VID(P2V) [1]-ADAPTED

This section accompanies Sec 4.1 in the main paper.

For each of the activity categories in our 360° dataset (Disney, Parade, etc.), we query Google Image Search engine and then manually filter out irrelevant images. We obtain about 300 images for each category. We use the Web images as positive training samples and randomly sampled panorama subviews as negative training sampling.



**Fig. 6.** Example of cubemaps when rotating in elevation. Views from a horizontal camera position (elevation  $0^\circ$ ) are more informative than others due to the natural human recording bias. In addition, rotation in elevation often makes cubemap faces appear tilted (e.g., building in the second row). Therefore, we optimize for the azimuth snap angle only.



**Fig. 7.** A detailed diagram showing our network architecture. In the top, the small schematic shows the connection between each network module. Then we present the details of each module in the bottom. Our network proceeds from left to right. The *feature extractor* consists of a sequence of convolutions (with kernel size and convolution stride written under the diagram) followed by a fully connected layer. In the bottom, “FC” denotes a fully connected layer and “ReLU” denotes a rectified linear unit. The *aggregator* is a recurrent neural network. The “Delay” layer stores its current internal hidden state and outputs them at the next time step. In the end, the *predictor* samples an action stochastically based on the multinomial pdf from the Softmax layer.

### Label Important Objects/Persons

Please carefully read the instructions, especially on what to label.

#### What to Label

1. Label **EACH** important object or person with a **TIGHT** bounding box. For example, two bounding boxes should be drawn for two different objects/persons.
2. Bounding box **MUST** be tight.
3. Label as many important objects/Persons as possible.

How to label: click on the top left corner of the rectangle you want to draw, drag and click on the bottom right corner to finish drawing a rectangle. When deleting a rectangle, click "Editing", select the rectangle you want to delete and then click "Delete Selected Rectangle". If you want to label more rectangles, click "Drawing".

Mode: Drawing Editing Operation: Delete Selected Rectangle



**Fig. 8.** Interface for Amazon Mechanical Turk when collecting important objects/persons. We present crowdworkers the panorama and instruct them to label any important objects with a bounding box—as many as they wish.

## 6.4 Interface for collecting important objects/persons

This section accompanies Sec. 4.2 in the main paper.

We present the interface for Amazon Mechanical Turk that is used to collect important objects and persons. The interface is developed based on [48]. We present crowdworkers the panorama and instruct them to label any important objects with a bounding box—as many as they wish. A total of 368 important objects/persons are labeled. The maximum number of labeled important objects/persons for a single image is 7.

## 6.5 More results when ground-truth objectness maps are used as input

This section accompanies Sec. 4.2 in the main paper.

We first got manual labels for the pixel-wise foreground for 50 randomly selected panoramas (200 faces). The IoU score between the pixel objectness prediction and ground truth is 0.66 with a recall of 0.82, whereas alternate foreground methods we tried [49,50] obtain only 0.35-0.40 IoU and 0.67-0.74 recall on the same data.

We then compare results when either ground-truth foreground or pixel objectness is used as input (without re-training our model) and report the foreground disruption with respect to the ground-truth. “Best possible” is the oracle result. Pixel objectness

Budget (T)	2	4	6	8	10	Best Possible	CANONICAL
Ground truth input	0.274	0.259	0.248	0.235	0.234	0.231	0.352
Pixel objectness input	0.305	0.283	0.281	0.280	0.277	0.231	0.352

**Table 5.** Decoupling pixel objectness and snap angle performance: error (lower is better) when ground truth or pixel objectness is used as input.

serves as a good proxy for ground-truth foreground maps. Error decreases with each rotation. Table 5 pinpoints to what extent having even better foreground inputs would also improve snap angles.

## 6.6 Interface for user study on perceived quality

This section accompanies Sec. 4.3 in the main paper.

We present the interface for the user study on perceived quality in Figure 9. Workers were required to rate each image into one of five categories: (a) The first set is significantly better, (b) The first set is somewhat better, (c) Both sets look similar, (d) The second set is somewhat better and (e) The second set is significantly better. We also instruct them to avoid to choose option (c) unless it is really necessary. Since the task can be ambiguous and subjective, we issued each task to 5 distinct workers. Every time a comparison between the two sets receives a rating of category (a), (b), (c), (d) or (e) from any of the 5 workers, it receives 2, 1, 0, -1, -2 points, respectively. We add up scores from all five workers to collectively decide which set is better.

## 6.7 The objectness map input for failure cases

This section accompanies Sec. 4.3 in the main paper.

Our method fails to preserve object integrity if pixel objectness fails to recognize foreground objects. Please see Fig. 10 for examples. The round stage is not found in the foreground, and so ends up distorted by our snap angle prediction method. In addition, the current solution cannot effectively handle the case when a foreground object is too large to fit in a single cube face.





Instructions

Which set of pictures looks better?





We show you two set of photos of the same scene. Each photo was taken by a different photographer. Your task is to decide Which set of photos is more aesthetically pleasing. When you judge the quality of the photos, please consider composition, viewpoint and whether important objects are visible. Avoid choosing "equally good" unless it is really necessary.

Which photo is better? :

First set of images



Second set of images



Category:

☐ The first set is significantly better than the second set

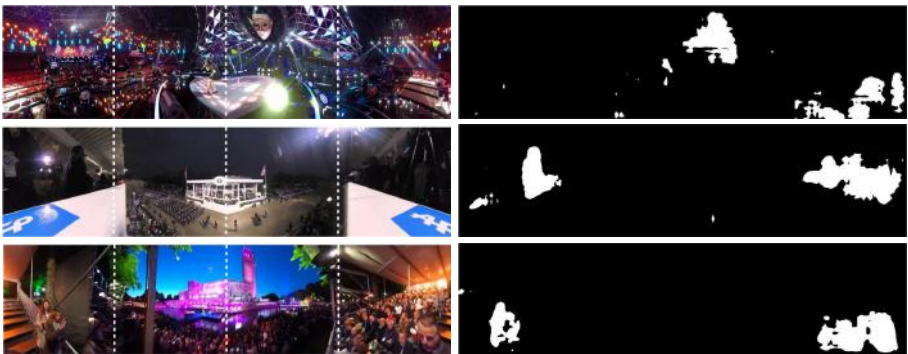
☐ The first set is somewhat better than the second set

☐ Both sets look equally good

☐ The second set is somewhat better than the first set

☐ The second set is significantly better than the first set

**Fig. 9.** Interface for user study on perceived quality. Workers were required to rate each image into one of five categories. We issue each sample to 5 distinct workers.



**Fig. 10.** Pixel objectness map (right) for failure cases of snap angle prediction. In the top row, the round stage is not found in the foreground, and so ends up distorted by our snap angle.

## 6.8 Additional cubemap output examples

This section accompanies Sec. 4.3 in the main paper.

Figure 11 presents additional cubemap examples. Our method produces cubemaps that place important objects/persons in the same cube face to preserve the foreground integrity. For example, in the top right of Figure 11, our method places the person in a single cube face whereas the default cubemap splits the person onto two different cube faces.

Figure 12 shows failure cases. A common reason for failure is that pixel objectness [35] does not recognize some important objects as foreground. For example, in the top right of Figure 12, our method creates a distorted train by splitting the train onto three different cube faces because pixel objectness does not recognize the train as foreground.

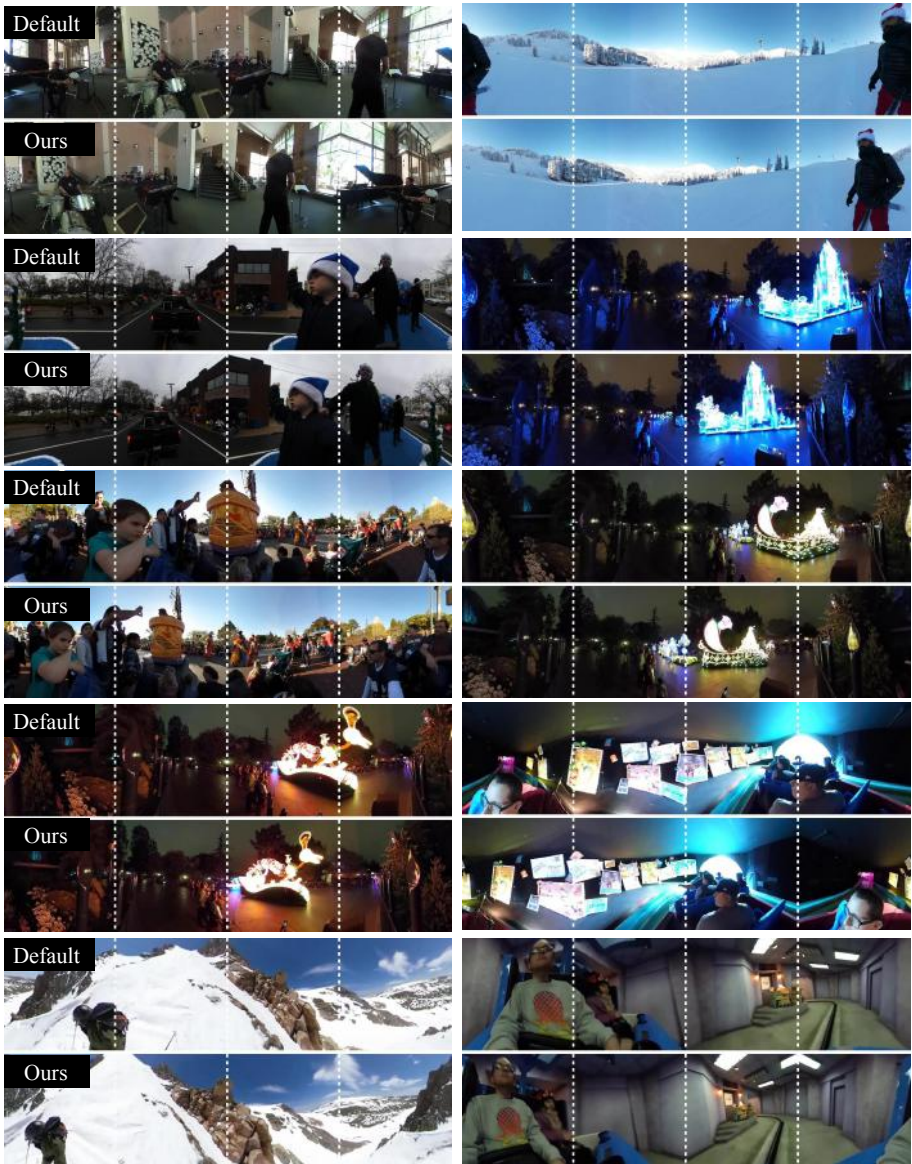
## 6.9 Setup details for recognition experiment

This section accompanies Sec. 4.4 in the main paper.

Recall our goal for the recognition experiment is to distinguish the four activity categories in our 360 dataset (Disney, Parade, etc.). We build a training dataset by querying Google Image Search engine for each activity category and then manually filtering out irrelevant images. These are the same as the positive images in Sec. 6.3 above.

We use Resnet-101 architecture [46] as our classifier. The network is first pre-trained on Imagenet [47] and then we finetune it on our dataset with SGD and a mini-batch size of 32. The learning rate starts from 0.01 with a weight decay of 0.0001 and a momentum of 0.9. We train the network until convergence and select the best model based on a validation set.





**Fig. 11.** Qualitative examples of default CANONICAL cubemaps and our snap angle cubemaps. Our method produces cubemaps that place important objects/persons in the same cube face to preserve the foreground integrity.





**Fig. 12.** Qualitative examples of default CANONICAL cubemaps and our snap angle cubemaps. We show failure cases here. In the top left, pixel objectness [35] does not recognize the stage as foreground, and therefore our method splits the stage onto two different cube faces, creating a distorted stage. In the top right, our method creates a distorted train by splitting the train onto three different cube faces because pixel objectness does not recognize the train as foreground.