In Proceedings of the European Conference on Computer Vision (ECCV), 2018

Snap Angle Prediction for 360° Panoramas Supplementary Material

Bo Xiong[1] and Kristen Grauman[2]

[1] University of Texas at Austin
bxiong@cs.utexas.edu
[2] Facebook AI Research
grauman@fb.com*

In this document we provide the following:

1. Justification for predicting snap angle azimuths (Sec. 3.1 in main paper)
2. Details on network architecture (Sec. 3.2 in main paper)
3. Training details for the baseline Pano2Vid(P2V) [7]-adapted (Sec. 4.1 in main paper)
4. Interface for Amazon Mechanical Turk when collecting important objects and persons (Sec. 4.2 in main paper)
5. More results when ground-truth objectness maps are used as input (Sec. 4.2 in main paper)
6. Interface for user study on perceived quality (Sec. 4.3 in main paper)
7. The objectness map input for failure cases (Sec. 4.3 in main paper)
8. Additional cubemap examples (Sec. 4.3 in main paper)
9. Setup details for recognition experiment (Sec. 4.4 in main paper)

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°) 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 1 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.

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

Fig. 1. Example of cubemaps when rotating in elevation. Views from a horizontal camera position (elevation 0°) 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.

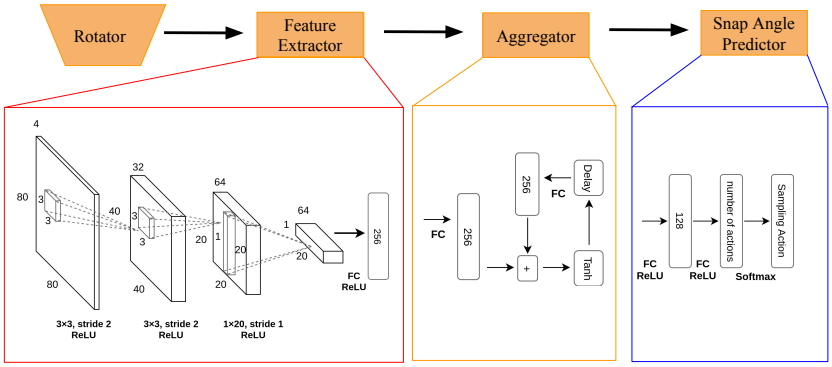

Fig. 2. A detailed diagram showing our network architecture. In the top, the small schematic shows the connection between each network module. Then we present the details of each module in the bottom. Our network proceeds from left to right. The *feature extractor* consists of a sequence of convolutions (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.

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 2.

3 Training details for Pano2Vid(P2V) [7]-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.

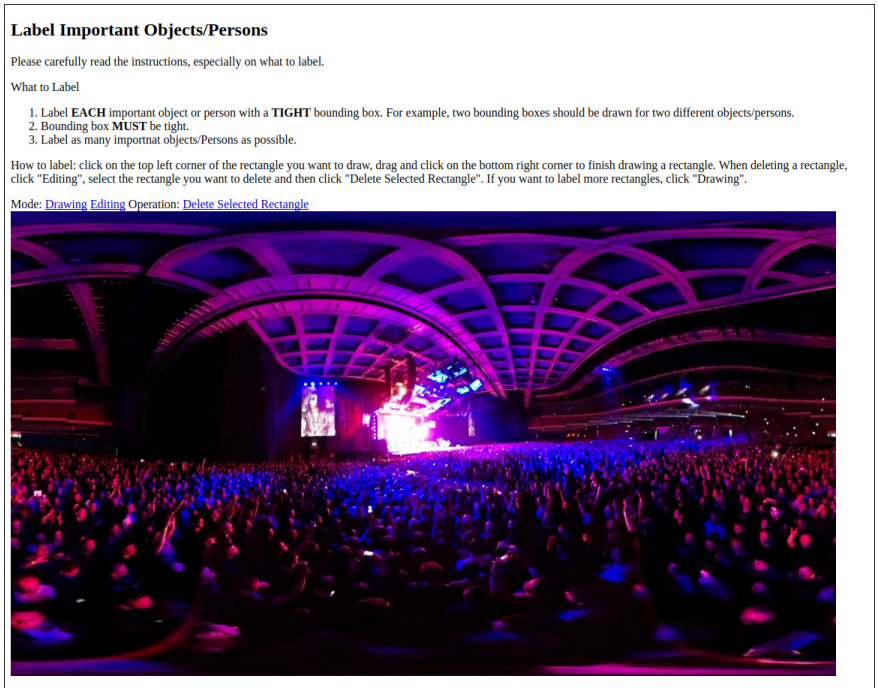

Fig. 3. Interface for Amazon Mechanical Turk when collecting important objects/persons. We present crowdworkers the panorama and instruct them to label any important objects with a bounding box—as many as they wish.

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 [1]. 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.

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

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

foreground methods we tried [5, 4] 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 serves as a good proxy for ground-truth foreground maps. Error decreases with each rotation. Table 1 pinpoints to what extent having even better foreground inputs would also improve snap angles.

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 4. 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.

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. 5 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.

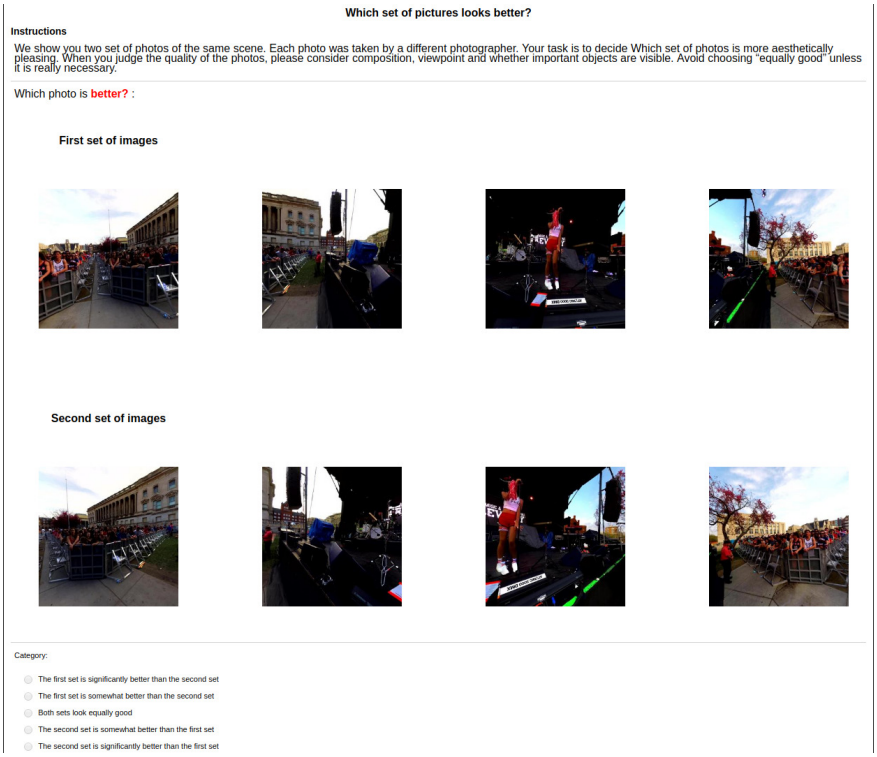

Fig. 4. Interface for user study on perceived quality. Workers were required to rate each image into one of five categories. We issue each sample to 5 distinct workers.

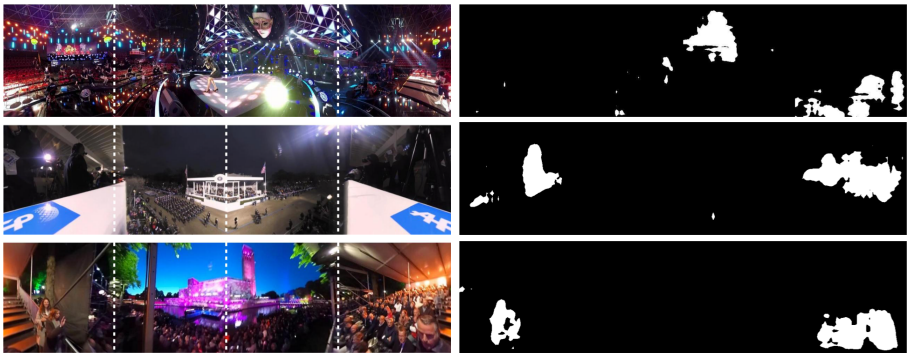

Fig. 5. Pixel objectness map (right) for failure cases of snap angle prediction. In the top row, the round stage is not found in the foreground, and so ends up distorted by our snap angle.

8 Additional cubemap output examples

This section accompanies Sec. 4.3 in the main paper.

Figure 6 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 6, our method places the person in a single cube face whereas the default cubemap splits the person onto two different cube faces.

Figure 7 shows failure cases. A common reason for failure is that pixel objectness [3] does not recognize some important objects as foreground. For example, in the top right of Figure 7, 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.

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. 3 above.

We use Resnet-101 architecture [2] as our classifier. The network is first pretrained on Imagenet [6] 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.

References

1. https://github.com/jianxiongxiao/ProfXkit
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
3. Jain, S.D., Xiong, B., Grauman, K.: Pixel objectness. arXiv preprint arXiv:1701.05349 (2017)
4. Jiang, B., Zhang, L., Lu, H., Yang, C., Yang, M.H.: Saliency detection via absorbing markov chain. In: ICCV (2013)
5. 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)
6. 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)
7. Su, Y.C., Jayaraman, D., Grauman, K.: Pano2vid: Automatic cinematography for watching 360 videos. In: ACCV (2016)

Fig. 6. 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. 7. Qualitative examples of default CANONICAL cubemaps and our snap angle cubemaps. We show failure cases here. In the top left, pixel objectness [3] 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.