[Appendix] Panoramic Vision Transformer for Saliency Detection in 360° Videos

Heeseung Yun, Sehun Lee, and Gunhee Kim

Seoul National University, Seoul, Korea

In the appendix, we provide details on our Panoramic Vision Transformer (PAVER) framework, implementation details of our model and the baselines, and additional experimental results deferred from the main paper.

A Details on Panoramic Vision Transformer

A.1 Vision Transformers

A detailed explanation of the transformer encoder in Eq. 4 is as follows. The transformer encoder consists of L layers of transformer blocks, where each block is composed of a multi-head self-attention module (MHSA) and a position-wise feed-forward network (FFN) with layer normalization [1] and residual connection for $i = 0, ..., L-1$:

$$\mathbf{x'_0} = \mathbf{p_0} + \mathbf{PE}, \tag{13}$$

$$\mathbf{x'_i} = \text{LayerNorm}(\text{MHSA}(\mathbf{x_i}) + \mathbf{x_i}), \tag{14}$$

$$\mathbf{x_{i+1}} = \text{LayerNorm}(\text{FFN}(\mathbf{x'_i}) + \mathbf{x'_i}), \tag{15}$$

where $\mathbf{p_i}, \mathbf{x_i}, \mathbf{x'_i} \in \mathbb{R}^{(N+1)\times C}$ for layer i and $\mathbf{PE}$ denotes position encoding. Since our input resolution is not the same as the one used in the original vision transformer, we follow the approach in [5] and resize $\mathbf{PE}$ with bicubic interpolation.

Multi-Head Self-Attention. MHSA first generates a set of query Q, key K, and value V with the corresponding projection from X and processes scaled dot-product attention with multiple heads [12]. The equation is given by

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{16}$$

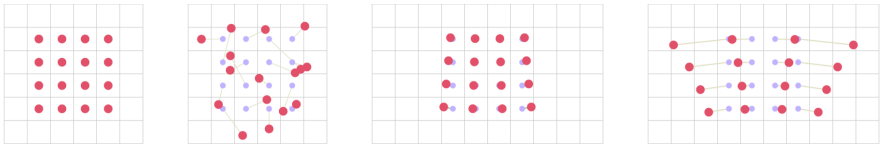
$$\text{head}_i(X) = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V), \tag{17}$$

$$\text{MHSA}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{18}$$

where $W_i^Q, W_i^K \in \mathbb{R}^{C\times d_k}, W_i^V \in \mathbb{R}^{C\times d_k}, W^O \in \mathbb{R}^{hd_v\times C}$ and $d_k = d_v = C/h$.

Position-wise Feed-Forward Network. After the multi-head self-attention module, each transformer block is connected with a fully connected feed-forward network made up of two linear layers with GELU [8] activation:

$$\text{FFN}(X) = \text{GELU}(XW_1 + b_1)W_2 + b_2. \tag{19}$$



(a) No offset (b) Learnable offset (c) Fixed offset ($\phi = -30°$) (d) Fixed offset ($\phi = 60°$)

Fig. 5: Intuition of the deformable convolution in PAVER.

A.2 Deformable Convolution

Motivation of Deformable Convolution in PAVER. The deformable convolution additionally introduces learnable offsets to the normal convolution to allow freeform kernels with marginal overhead. Fig. 5 visualizes this intuition. We fix the offset with respect to the underlying spherical geometry and load kernel weights from any pretrained models.

Derivation of Geometric Offset for DeformConv. We elaborate the derivation process of geometric offset for DeformConv in equirectangular format. As discussed in Eq. 1–3, we first set up the reference patch $\mathcal{P} \in \mathbb{R}^{S\times S\times 3}$, *i.e.*, $\mathcal{P} = \{(\frac{\pi}{28}, \frac{\pi}{28}, 1), (\frac{S-2}{S-1}\frac{\pi}{14} - \frac{\pi}{28}, \frac{\pi}{28}, 1), ..., (-\frac{\pi}{28}, -\frac{\pi}{28}, 1)\}$. This is equivalent to $S\times S$ points sampled from bounded plane $z = 1$ such that $(x, y) \in \mathbb{R}^{(-\frac{\pi}{28}, \frac{\pi}{28})\times(-\frac{\pi}{28}, \frac{\pi}{28})}$. $\mathcal{P}\times R(\theta_i, \phi_i)$ in Eq. 2 rotates the points in $\mathcal{P}$ by (θ_i, ϕ_i) and normalizes each point to have a length of 1, which moves all the points onto the surface of the sphere. Then we convert the points from cartesian to spherical coordinates ($f_{\text{3D}\to\text{Sph}}$) and from spherical coordinates to 2D equirectangular coordinates ($f_{\text{Sph}\to\text{ER}}$):

$$f_{\text{3D}\to\text{Sph}}(x, y, z) = (\arctan\frac{y}{x}, \arctan\frac{\sqrt{x^2 + y^2}}{z}), \tag{20}$$

$$f_{\text{Sph}\to\text{ER}}(\theta, \phi) = (\frac{W}{2\pi}\phi, \frac{H}{\pi}\theta). \tag{21}$$

Throughout this process, we obtain $\Theta_{\text{DC-offset}} \in \mathbb{R}^{2\times S\times S\times h\times w}$, *i.e.*, $S \times S$ 2D points for all $h \times w$ patches.

A.3 Geometric Offsets for Various Video Formats

Using the geometric offsets for equirectangular format obtained in Sec. A.2, we compute the geometric offsets for other 360° video formats like cubemap projection (RCMP) and truncated square pyramid projection (TSP). To achieve this, we leverage an open source 360° format conversion tool[1] to transform the offsets. We first generate $S \times S \times h \times w$ images with one pixel marked, which corresponds to each offset point in $\Theta_{\text{DC-offset}}$. Then we convert these images from the equirectangular format to the target format using the conversion tool. Finally, we collect the coordinates of marked pixels, which constitute the geometric offsets for the target 360° video format.

[1] https://github.com/Samsung/360tools

B Implementation Details

B.1 Details on Experiment Setting

We further report the details of the experiments in our paper. Some variants of PAVER require larger learning rates or longer training epochs for convergence. To be specific, we use two times longer training time for PAVER (NoGlobal) and PAVER (NoDecoupled) in Table 1, five times larger learning rate for DINO backbone in Table 2-(b), and both for time-only saliency score in Table 2-(a).

For baseline models using the vision transformer as a backbone, we run experiments with different pretrained weights and report the best performing configuration. We use DINO with ViT-B/8 [3] as weights for experiments with DINO [3], LOST [10], and TokenCut [13] in Table 1, as the original ViT-B/16 [5] was detrimental to the performance. For TS-CAM [7], we use maximum class activation maps for generating saliency maps as in CubePad [4] due to lack of supervision.

B.2 Details on Video Quality Assessment

As explained in Sec. 4.1, we compute PSNR, S-PSNR [14] and WS-PSNR [11] with saliency map $w^{\mathrm{SAL}}(\mathbf{p})$ as weights:

$$\mathrm{PSNR}_{\mathrm{SAL}} = 10 \log \frac{Y_{\mathrm{max}}^2 \cdot \sum_{\mathbf{p} \in \mathbb{P}} w^{\mathrm{SAL}}(\mathbf{p})}{\sum_{\mathbf{p} \in \mathbb{P}} (Y(\mathbf{p}) - Y'(\mathbf{p}))^2 \cdot w^{\mathrm{SAL}}(\mathbf{p})}, \tag{22}$$

$$\text{WS-PSNR}_{\mathrm{SAL}} = 10 \log \frac{Y_{\mathrm{max}}^2 \cdot \sum_{\mathbf{p} \in \mathbb{P}} w^{\mathrm{SAL}}(\mathbf{p}) \cos \theta_{\mathbf{p}}}{\sum_{\mathbf{p} \in \mathbb{P}} (Y(\mathbf{p}) - Y'(\mathbf{p}))^2 \cdot w^{\mathrm{SAL}}(\mathbf{p}) \cos \theta_{\mathbf{p}}}, \tag{23}$$

$$\text{S-PSNR}_{\mathrm{SAL}} = 10 \log \frac{Y_{\mathrm{max}}^2 \cdot \sum_{\mathbf{p} \in \mathrm{sph}(\mathbb{P})} w^{\mathrm{SAL}}(\mathbf{p})}{\sum_{\mathbf{p} \in \mathrm{sph}(\mathbb{P})} (Y(\mathbf{p}) - Y'(\mathbf{p}))^2 \cdot w^{\mathrm{SAL}}(\mathbf{p})}, \tag{24}$$

where $\mathrm{Y}_{\mathrm{max}}$ denotes maximum intensity value of the video, $Y(\mathbf{p})$ and $Y'(\mathbf{p})$ indicate intensities of pixel $\mathbf{p}$ in the reference and impaired omnidirectional video frames, and $\theta_{\mathbf{p}}$ is the latitude of pixel $\mathbf{p}$. $\mathbb{P}$ and $\mathrm{sph}(\mathbb{P})$ indicate points sampled from equirectangular plane and sphere, respectively.

B.3 Computation Environment

- GPU: NVIDIA TITAN X/Xp
- CPU: Intel(R) Xeon(R) Gold 6130 CPU
- OS: Ubuntu 16.04 LTS
- RAM: SAMSUNG DDR4 8G
- Relevant software libraries: Anaconda distribution of python ($\approx$ 3.7) and PyTorch ($\approx$ 1.9)

Please refer to our source code for more details.

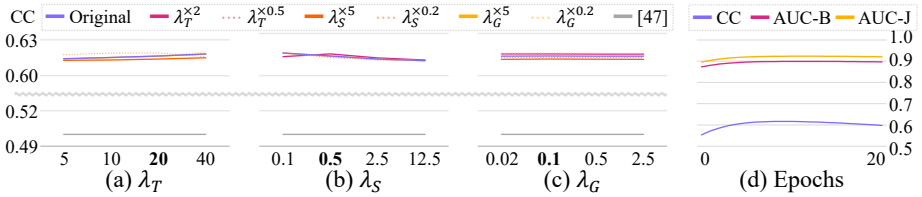


Fig. 6: Performance of PAVER under different hyperparameter values and training schedule.

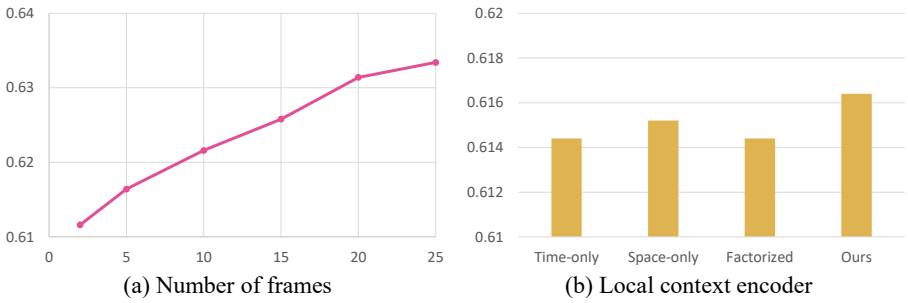


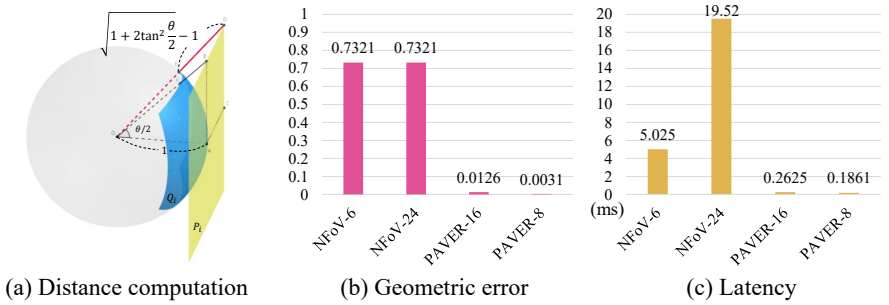
Fig. 7: Influence of video length and local context fusion module on Wild360 test split. Y axis indicates linear correlation coefficient (CC).

C Additional Experiments

C.1 Influence of Hyperparameters

The selection of hyperparameters is important in unsupervised scenarios. We investigate the influence of hyperparameters in terms of losses and training schedules. Fig. 6-(a–c) visualizes the performance of our PAVER under a varying set of hyperparameters. The performance gap between different hyperparameter values is no more than 1%p unless the value gets dramatically larger or smaller. Still, the performance of PAVER is consistently better than the state-of-the-art method (*i.e.*, [13]) in this range. That is, our approach is not sensitive to changes in hyperparameters.

Fig. 6-(d) reports the performance of all checkpoints for a prolonged training schedule. After maxing out, the performance gradually declines instead of converging to a certain point. This mostly has to do with the temporal and spatial consistency loss. The two losses cause the saliency map to gradually diffuse, which is slightly detrimental to performance. Therefore, we refer to the convergence of the saliency map update ratio per epoch during training and use fixed epochs to measure the performance, which is in line with other works [3–5].



(a) Distance computation (b) Geometric error (c) Latency

Fig. 8: Geometric error and latency analysis.

C.2 Analysis on Video Length and Temporal Modeling

Fig. 7-(a) summarizes the influence of video length in the Wild360 dataset. Although we use $T = 5$ frames for a fair comparison with the previous state-of-the-art model [4], using longer videos as input is beneficial for performance. For instance, if we use 25 frames at once as input, linear correlation coefficient performance boosts by 1.7%p, which implies that providing a longer temporal context is beneficial for saliency prediction on 360° videos.

Fig. 7-(b) reports the influence of the local context encoder, which compares our spatiotemporal fusion module against time-only attention, space-only attention, and factorized attention used in [2]. As discussed, both space attention and time attention contribute to the full model's performance, while stacking one after another as in factorized attention slightly reduces the performance.

C.3 Analysis on Latency and Geometric Error

Fig. 8 reports latency and geometric error of PAVER compared to general NFoV counterparts. For NFoV models, we use the NFoV models with six projections (*e.g.*, CubePad [4]) and 24 projections as general configuration. We also analyze the performance of PAVER with a patch size of $S = 8$ and $S = 16$. We only take into account the latency and geometric error incurred from the geometric projection module instead of the full model for a fair comparison.

Fig. 8-(a,b) summarizes the maximum geometric error induced by tangential projection assuming a unit sphere (*i.e.*, a sphere with radius 1). As investigated in [6], even subtle radial distortion in an input image incurs a significant drop in performance. In order to process omnidirectional inputs with a set of projected patches below a certain error bound, a geometric error between the original sphere and projected patches should be defined beforehand. We use the Hausdorff distance [9] to measure this value, which is one of the standard distance metrics between two arbitrary surfaces. Hausdorff distance computes the greatest of all the distances from a point on the source surface to the closest point on the target surface. For a set of projected patches $\mathbf{P} = \{P_i\}_{i=1}^{N}$ and its corresponding sphere segment $\mathbf{Q} = \{Q_i\}_{i=1}^{N}$, the Hausdorff distance $h(\mathbf{P}, \mathbf{Q})$ is derived as follows:

$$
\begin{aligned}
h(\mathbf{P}, \mathbf{Q}) &= \max_{i \in \{1,\dots,N\}} h(P_i, Q_i) \\
&= \max_{p \in P_i} \min_{q \in Q_i} ||p - q|| \\
&= \max_{p \in P_i} \min_{q \in Q_i} ||(p - O) - (q - O)|| \\
&= \max_{p \in P_i} ||p - O|| - 1 \\
&= \sqrt{1 + 2\tan^2 \frac{\theta}{2}} - 1, \qquad (25)
\end{aligned}
$$

where θ denotes the angular distance of patch P_i and O denotes the center of the sphere.

Since NFoV models project the unit sphere onto a circumscribed cube, the geometric error induced from projection is $\sqrt{3} - 1 \approx 7.321 \times 10^{-1}$ per NFoV projection. On the other hand, our model displays a maximum geometric error of $\sqrt{1 + 2\tan^2 \frac{\pi}{28}} - 1 \approx 1.262 \times 10^{-2}$ for patch length $S = 16$ and $\sqrt{1 + 2\tan^2 \frac{\pi}{56}} - 1 \approx 3.149 \times 10^{-3}$ for patch length $S = 8$, which is an order of magnitude smaller.

Fig. 8-(c) summarizes the time required to process 360° input, where we report the average latency per input for processing 1,000 random inputs, averaging five independent runs. To be specific, we compare the time required to project NFoV images against the relative overhead incurred by deformation in DeformConv with respect to general convolution operation with a batch size of 1. For NFoV projection, we use the implementation of [4]. Our approach takes considerably less time to process 360° inputs compared to conventional NFoV-based approaches, without the need for explicitly saving projected patches.

C.4 More Qualitative Examples

Fig. 9 visualizes more qualitative examples of saliency prediction results from PAVER. Our approach can discern the relative importance of varying foreground objects while maintaining temporally consistent saliency maps.

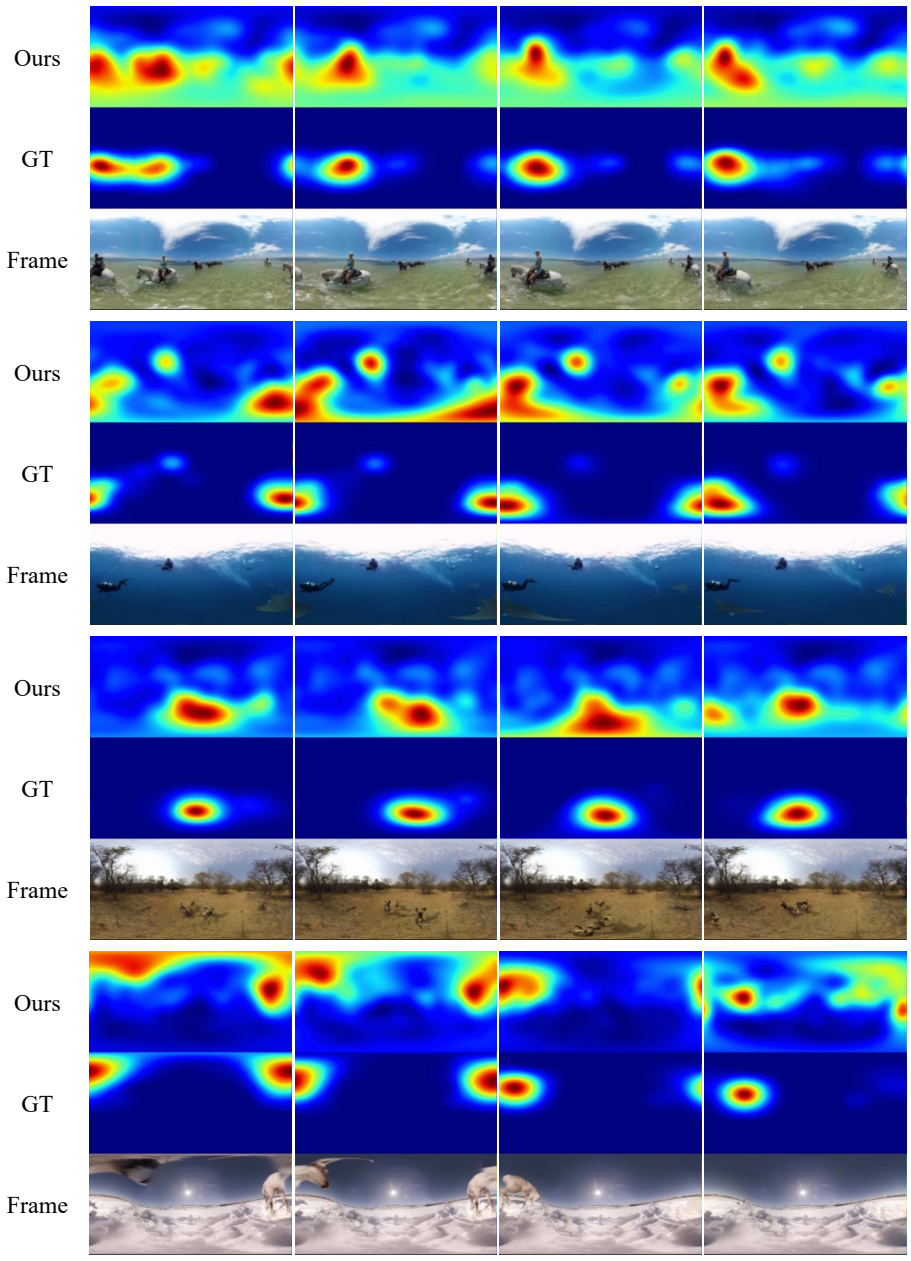


Fig. 9: Qualitative examples of 360° video saliency prediction by PAVER.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 (2016)
2. Bertasius, G., Wang, H., Torresani, L.: Is Space-Time Attention All You Need for Video Understanding? In: ICML (2021)
3. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging Properties in Self-Supervised Vision Transformers. In: ICCV (2021)
4. Cheng, H.T., Chao, C.H., Dong, J.D., Wen, H.K., Liu, T.L., Sun, M.: Cube padding for weakly-supervised saliency prediction in 360 videos. In: CVPR (2018)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020)
6. Eder, M., Shvets, M., Lim, J., Frahm, J.M.: Tangent images for mitigating spherical distortion. In: CVPR (2020)
7. Gao, W., Wan, F., Pan, X., Peng, Z., Tian, Q., Han, Z., Zhou, B., Ts-cam, Q.Y.: Token Semantic Coupled Attention Map for Weakly Supervised Object Localization. In: ICCV (2021)
8. Hendrycks, D., Gimpel, K.: Gaussian Error Linear Units (GELUs). arXiv:1606.08415 (2016)
9. Rockafellar, R.T., Wets, R.J.B.: Variational Analysis, vol. 317. Springer Science & Business Media (2009)
10. Siméoni, O., Puy, G., Vo, H.V., Roburin, S., Gidaris, S., Bursuc, A., Pérez, P., Marlet, R., Ponce, J.: Localizing Objects with Self-Supervised Transformers and no Labels. In: BMVC (2021)
11. Sun, Y., Lu, A., Yu, L.: Weighted-to-spherically-uniform quality evaluation for omnidirectional video. SPL (2017)
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NIPS (2017)
13. Wang, Y., Shen, X., Hu, S., Yuan, Y., Crowley, J., Vaufreydaz, D.: Self-Supervised Transformers for Unsupervised Object Discovery using Normalized Cut. In: CVPR (2022)
14. Yu, M., Lakshman, H., Girod, B.: A framework to evaluate omnidirectional video coding schemes. In: ISMAR (2015)