SphereSR: 360° Image Super-Resolution with Arbitrary Projection via Continuous Spherical Image Representation –Supplementary Material–

Youngho Yoon, Inchul Chung, Lin Wang*, and Kuk-Jin Yoon
Visual Intelligence Lab., KAIST, Korea
{dudgh1732,inchul1221,wanglin,kjyoon}@kaist.ac.kr

Abstract

Due to the lack of space in the main paper, we provide more details of the proposed methods and experimental results in the supplementary material. Specifically, in Sec.1, we provide more details of the feature extraction module for spherical images. Sec.2 and Sec.3 explain explicitly the proposed sphere-oriented cell decoding in SLIIF module and the feature loss. Sec.4 and Sec.5 provide implementation details and more limitations. Lastly, Sec.6 presents additional results about experiments and ablation study.

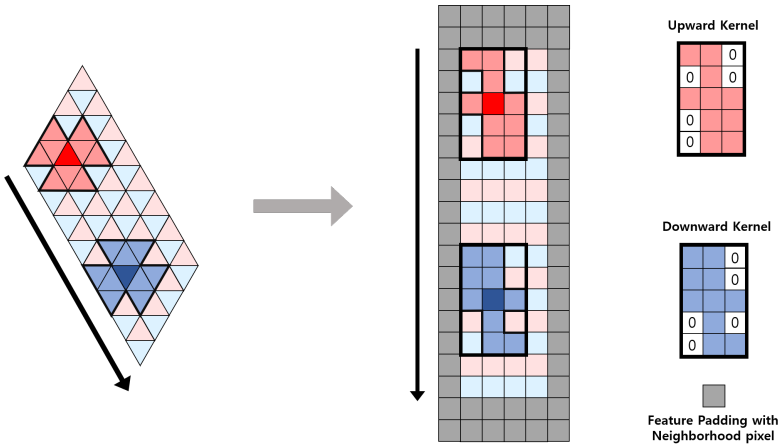

Figure 1. Proposed new data structure.

1. More Details of Feature Extraction for Spherical Images

1.1. Data structure

As shown in Fig. 1, we make the implementation of up/downward kernels possible with 5×3 convolution by using the proposed new data structure. At this time, among the pixels of the 5×3 convolution kernel, five pixels not included in the actual kernel are always multiplied by weight 0, and weight updating is not performed.

We also always conduct feature padding with neighborhood pixels directly before convolution to avoid losing the continuity of the spherical data structure. The padding size is 2×1 considering the kernel size.

Finally, as the convolutional kernels for odd-numbered and even-numbered rows are different, the convolution stride for the row axis is set to 2 such that convolution for the kernel can be performed for each row. After convolution for each kernel, the two feature maps are concatenated according to the order of the rows.

*Lin Wang is currently with HKUST.

1.2. Kernel Weight Sharing

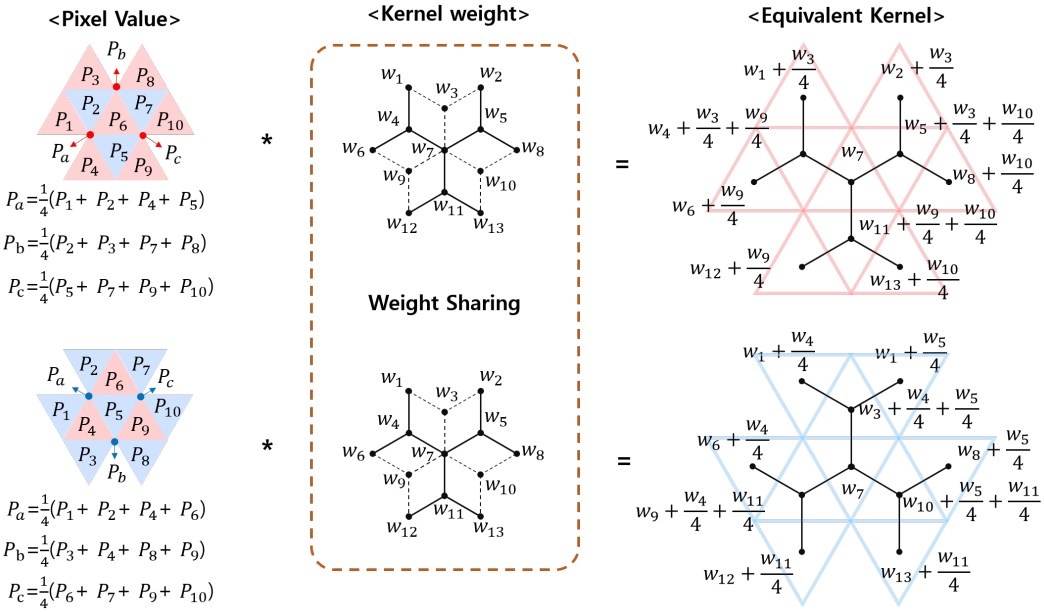

Figure 2. Geometry-Aligned Convolution(GA-Conv) and equivalent kernel.

In this subsection, we describe in detail the derivation of the kernel weight sharing scheme for GA-Conv (Geometry-Aligned Convolution). As shown in Fig. 2, we set the three vertices of the center pixel to the imaginary pixels P_a, P_b, and P_c to equalize the shapes of the pixel combinations. At this time, each pixel can be regarded as the midpoint of the surrounding four pixels. Therefore, the values of pixels P_a, P_b, and P_c are set as the mean value of the four pixels through the formulas in Fig. 2. We present a method by which to utilize weight sharing without actually creating a pixel when implementing this setting. As shown in Fig. 2, convolution with pixel values and kernel weights is the same as the value convolutional using the equivalent kernel. Therefore, an appropriate weight sharing scheme is used when using GA-Conv, and through this, more sophisticated kernel weight sharing is possible without kernel rotation as previously performed in SpherePHD.

2. Sphere-oriented Cell Decoding

In this section, we explain more details about sphere-oriented cell-decoding method. A rectangle $\overrightarrow{\Delta X}, \overrightarrow{\Delta Y}$ on the projected plane makes parallelogram $\overrightarrow{\Delta x}, \overrightarrow{\Delta y}$ on the unit sphere. To define a cell decoding according to point p on unit sphere, we initially define a vector $(\overrightarrow{n_1}, \overrightarrow{n_2})$. Let $o = center\ point(r = 0)$, $p = (\theta, \phi)$ s.t. point p we want to get RGB and $v_x = (\theta_v, \phi_v)$s.t. one of vertices 1, 2, 3. Then,

$$\overrightarrow{n_1} = \overrightarrow{v_x p} = (\theta - \theta_v)\hat{\theta} + (\phi - \phi_v)\sin\theta\hat{\phi} \tag{1}$$

$$\overrightarrow{n_2} = \overrightarrow{op} \times \overrightarrow{n_1} = -(\phi - \phi_v)\sin\theta\hat{\theta} + (\theta - \theta_v)\hat{\phi} \tag{2}$$

To make $(\overrightarrow{n_1}, \overrightarrow{n_2})$ as unit vectors, the vectors are scaled as $\overrightarrow{n_1} \leftarrow \overrightarrow{n_1}/\parallel \overrightarrow{n_1} \parallel$ and $\overrightarrow{n_2} \leftarrow \overrightarrow{n_2}/\parallel \overrightarrow{n_2} \parallel$. By these equations, we can find the variables $\alpha_1, \beta_1, \alpha_2, \beta_2$ that satisfies the following equation,

$$\begin{pmatrix} \hat{\theta} \\ \hat{\phi} \end{pmatrix} = \begin{pmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \end{pmatrix} \begin{pmatrix} \overrightarrow{n_1} \\ \overrightarrow{n_2} \end{pmatrix} \tag{3}$$

To denote vectors $\overrightarrow{\Delta x}, \overrightarrow{\Delta y}$ representing parallelogram sphere cells using $\overrightarrow{n_1}, \overrightarrow{n_2}$, we first denote using $\hat{\theta}, \hat{\phi}$ through the projection process of each projection type, as shown in Fig. 3. Therefore, we have to find as the equation,

$$\left.\begin{aligned} \overrightarrow{\Delta x} &= \gamma_1\hat{\theta} + \delta_1\hat{\phi} \\ \overrightarrow{\Delta y} &= \gamma_2\hat{\theta} + \delta_2\hat{\phi} \end{aligned}\right) \Rightarrow \begin{array}{c}\text{decided by output of} \\ \text{projection type}\end{array} \tag{4}$$

Table 1. Derivation of $\overrightarrow{\Delta x}, \overrightarrow{\Delta y}$ for various projection types.

ERP	Perspective
$\begin{pmatrix}\overrightarrow{\Delta x}\\ \overrightarrow{\Delta y}\end{pmatrix} = \begin{pmatrix}\Delta Y \sin\theta & 0\\ 0 & \Delta X\end{pmatrix}\begin{pmatrix}\hat{\theta}\\ \hat{\phi}\end{pmatrix}$	$\begin{pmatrix}\overrightarrow{\Delta x}\\ \overrightarrow{\Delta y}\end{pmatrix} = \begin{pmatrix}\alpha\Delta X \cos\theta \sin\phi & \alpha\Delta X\cos\phi\\ -\alpha\Delta Y\sin\theta & 0\end{pmatrix}\begin{pmatrix}\hat{\theta}\\ \hat{\phi}\end{pmatrix}$
Fisheye	
$\begin{pmatrix}\overrightarrow{\Delta x}\\ \overrightarrow{\Delta y}\end{pmatrix} = \begin{pmatrix}\alpha\Delta X\cos\theta\sin\phi + \alpha\Delta X F_X(X,Y)\cos\theta\cos\phi & \alpha\Delta X\cos\phi - \alpha\Delta X F_X(X,Y)\sin\phi\\ -\alpha\Delta Y\sin\theta + \alpha\Delta Y F_Y(X,Y)\cos\theta\cos\phi & -\alpha\Delta Y F_Y(X,Y)\sin\phi\end{pmatrix}\begin{pmatrix}\hat{\theta}\\ \hat{\phi}\end{pmatrix}$	

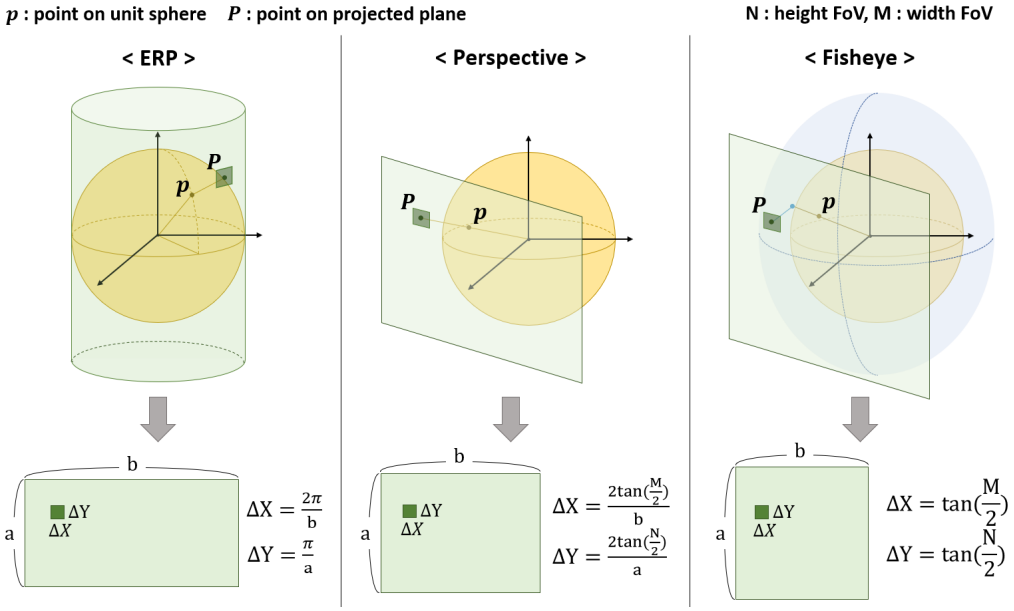

Figure 3. Visualization of projection process of each projection type.

Let P be the point (X, Y) where point p is projected onto the projected plane. As the derivation results, each projection can be organized as a table. 1. Fisheye projection type is the result of the projection process by setting the reflective surface $Z = F(X, Y)$ as mentioned in [2]. Also, in case of Perspective projection,

$$\alpha = \frac{1}{\sqrt{\tan^2\phi + \frac{1}{\tan^2\theta\cos^2\phi}}}, \Delta X = \frac{2\tan\frac{M}{2}}{b}, \Delta Y = \frac{2\tan\frac{N}{2}}{a} \tag{5}$$

in case of Fisheye projection,

$$\alpha = \frac{1}{\sqrt{X^2+Y^2+F_X(X,Y)^2}}, \Delta X = \frac{F(X',0)}{b} \text{ s.t. } \frac{X'}{F(X',0)} = \tan\frac{M}{2}, \Delta Y = \frac{F(0,Y')}{a} \text{ s.t. } \frac{Y'}{F(0,Y')} = \tan\frac{N}{2} \tag{6}$$

Finally, we are able to represent $\overrightarrow{\Delta x}, \overrightarrow{\Delta y}$ using two vectors $\overrightarrow{n_1}, \overrightarrow{n_2}$ by Eqs. (3) and (4).

$$\begin{pmatrix}\overrightarrow{\Delta x}\\ \overrightarrow{\Delta y}\end{pmatrix} = \begin{pmatrix}\gamma_1 & \delta_1\\ \gamma_2 & \delta_2\end{pmatrix}\begin{pmatrix}\alpha_1 & \beta_1\\ \alpha_2 & \beta_2\end{pmatrix}\begin{pmatrix}\overrightarrow{n_1}\\ \overrightarrow{n_2}\end{pmatrix} \tag{7}$$

We then find a $\overrightarrow{\Delta x_{eq}}, \overrightarrow{\Delta y_{eq}}$ as mentioned in the main paper.

3. Details of Feature Loss

In this section, we explain more details about feature loss. The feature loss takes advantage of the SR features generated from other projection types by utilizing the SLIIF module. It induces spherical features that can be easily converted into features appropriate for arbitrary projection types. However, applying it to the entire feature map causes performance degradation, so we designed a 'feature mask extraction module' to apply the feature loss only to areas with reasonable performance in the feature map generated from ERP and CubeMap.

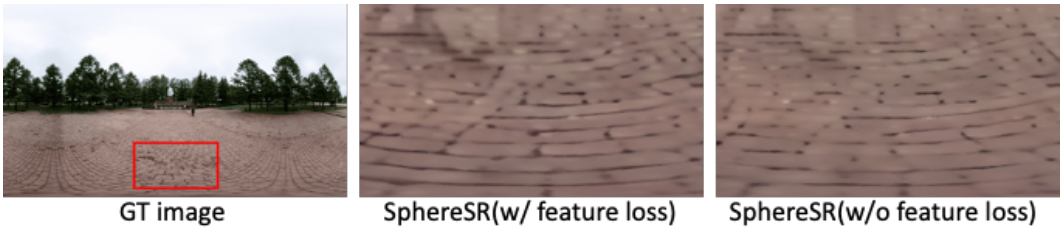

Figure 4. Visual effectiveness of feature loss on the ERP image.

In this module, to activate the spatial areas with small errors, the *spatial mask* is generated from the error map (difference between SR output and GT), while the *channel mask* is generated through GAP to extract the activated channels.To prevent masks from becoming 0, as Fig. 7 in the main paper, we perform a softmax function before extracting the spatial/channel masks. Therefore, the sum of values in each mask should be 1.

The feature loss subtly minimizes feature uncertainty among different projection types, helping to improve better the perceptual quality of the SR images, as verified in Fig. 4. Using the feature loss (2nd column) produces a more precise image pattern and details.

4. Implementation Details

The spherical feature extractor is implemented using GA-conv based on EDSR [1] and outputs a feature with 128 dimensions. The RGB prediction decoding function f_{dec} is a 5-layer MLP, and the hidden layer dimension is 256. The EDSR models, pre-trained with the ODI-SR dataset and frozen, are used to extract features from ERP and cube map images for feature loss. SphereSR is trained for 500 epochs using the Adam Optimizer with a batch size of 1. The initial learning rate is set to 0.0001 with a tenfold reduction after 400 epochs. The feature loss scale parameter λ is set to 0 for the first 100 epochs and set to 0.3 for the rest of the training.

5. More Limitations

The spherical feature map has uniform information in all directions. Therefore, on the pole area of the ERP image, SR for an extremely high scale factor is inevitably made due to distortion. If the pole area of the ERP image has a complex texture, ours has poor performance compared to other ERP-based models. Future studies will therefore need to improve this case.

6. Visualization

6.1. Additional Qualitative Comparison

We additionally visualize the qualitative comparison with the proposed method and other methods on the ODI-SR dataset and the SUN360 Panorama dataset in Fig. 5 and Fig. 6. We trained the proposed network using the $\times 8$ ERP images on the ODI-SR dataset. We then conduct an experiment involving conversion to a FOV 90° perspective image with a size 512×512. We also conduct another experiment on the conversion to a FOV 180° fisheye image with a size 1024×1024.

6.2. Visualization of SR with Arbitrary Projection Types

We additionally visualize the SR results for arbitrary projection type from LR ERP images in Fig. 7 and Fig. 8. We also visualize images for different directions to show that our model can predict SR results for arbitrary directions.

References

[1] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 4

[2] Davide Scaramuzza, Agostino Martinelli, and Roland Y. Siegwart. A toolbox for easily calibrating omnidirectional cameras. pages 5695–5701, 2006. 3

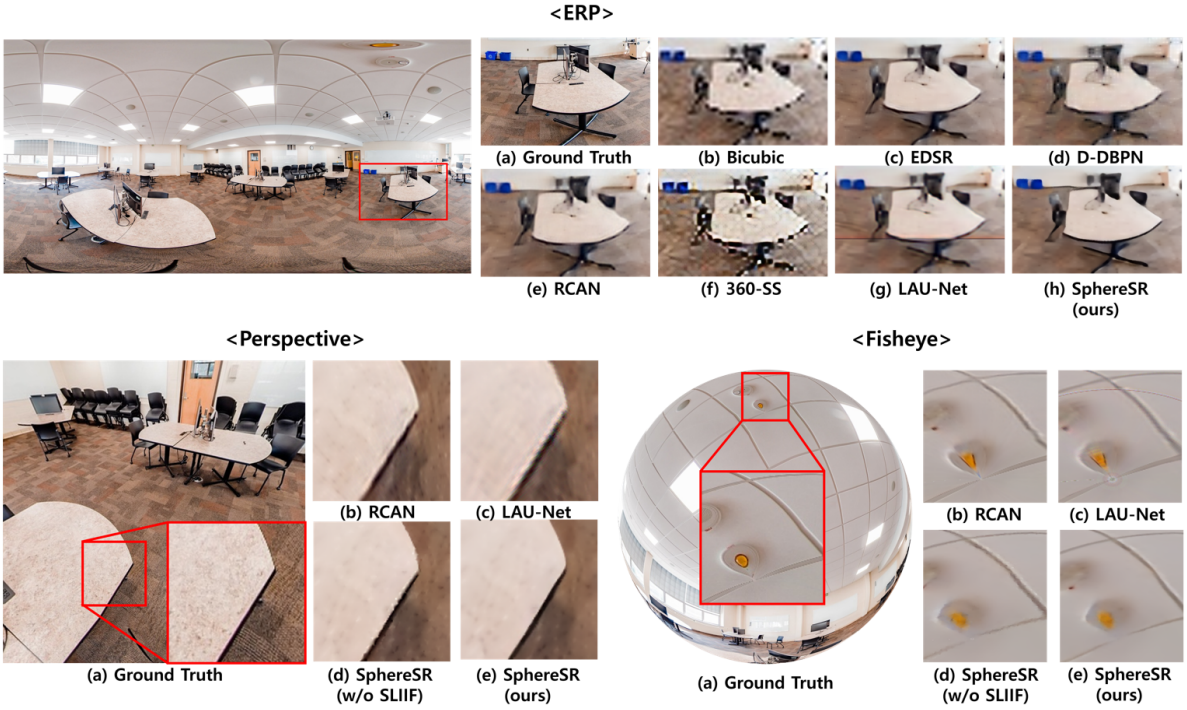

Figure 5. Visual comparisons of $\times 8$ SR results of ERP, perspective, fisheye projection on SUN 360 Panorama dataset.

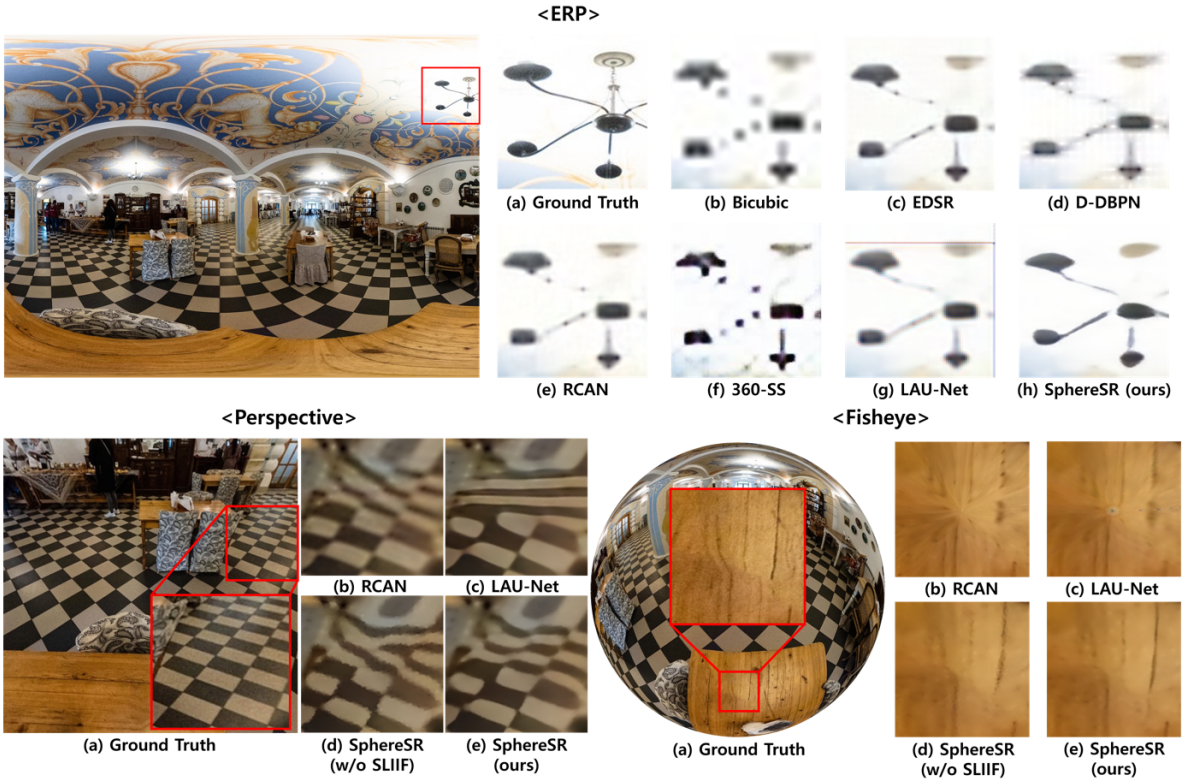

Figure 6. Visual comparisons of $\times 8$ SR results of ERP, perspective, fisheye projection on ODI-SR dataset.

Figure 7. Visualization of SR results with various projection types on the SUN360 Panorama dataset.

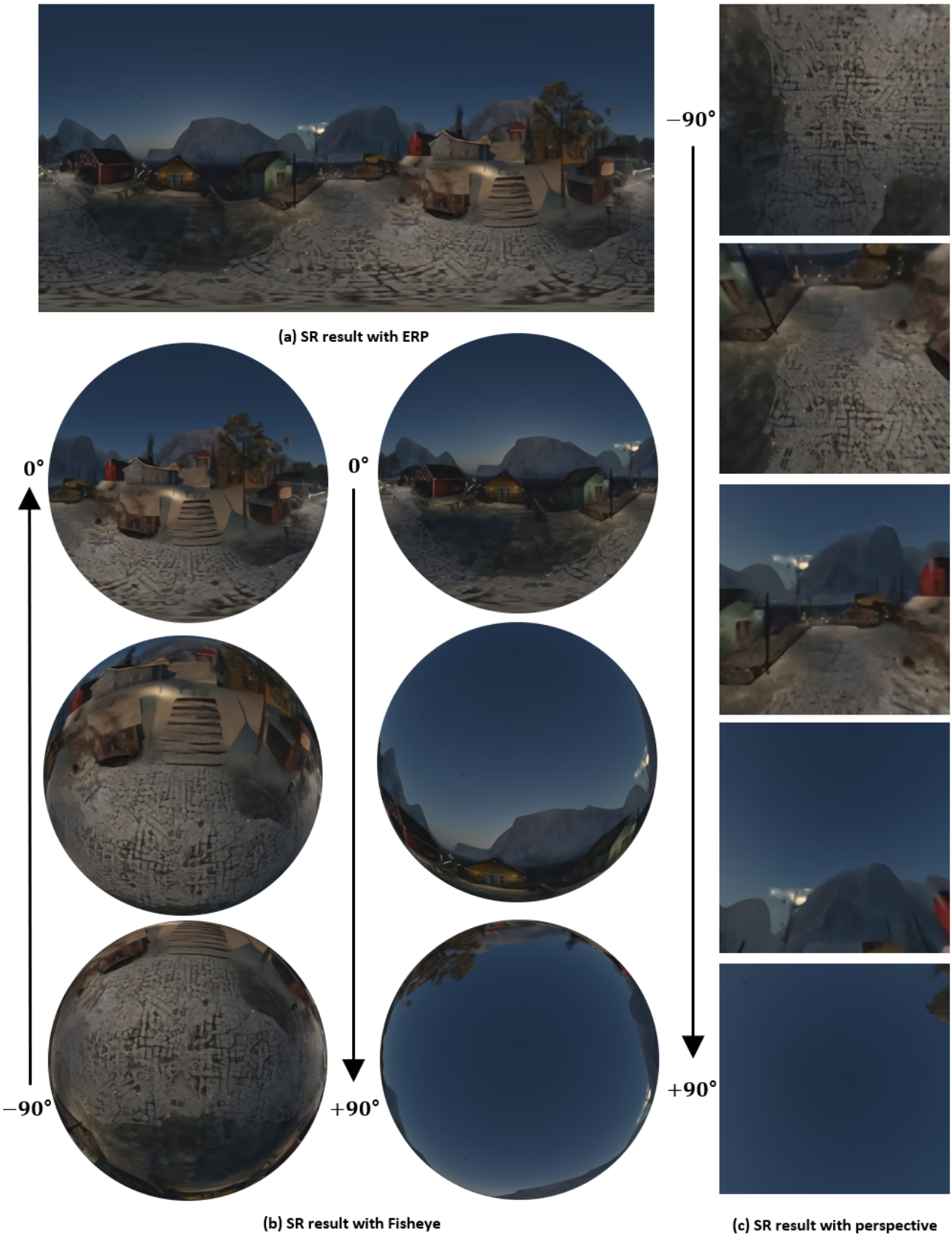

Figure 8. Visualization of SR results with various projection types on the ODI-SR dataset.