Abstract

The 360° imaging has recently gained much attention; however, its angular resolution is relatively lower than that of a narrow field-of-view (FOV) perspective image as it is captured using a fisheye lens with the same sensor size. Therefore, it is beneficial to super-resolve a 360° image. Several attempts have been made, but mostly considered equirectangular projection (ERP) as one of the ways for 360° image representation despite the latitude-dependent distortions. In that case, as the output high-resolution (HR) image is always in the same ERP format as the low-resolution (LR) input, additional information loss may occur when transforming the HR image to other projection types. In this paper, we propose SphereSR, a novel framework to generate a continuous spherical image representation from an LR 360° image, with the goal of predicting the RGB values at given spherical coordinates for super-resolution with an arbitrary 360° image projection. Specifically, first we propose a feature extraction module that represents the spherical data based on an icosahedron and that efficiently extracts features on the spherical surface. We then propose a spherical local implicit image function (SLIIF) to predict RGB values at the spherical coordinates. As such, SphereSR flexibly reconstructs an HR image given an arbitrary projection type. Experiments on various benchmark datasets show that the proposed method significantly surpasses existing methods in terms of performance.

1. Introduction

The 360° imaging has recently gained much attention in many fields, including the AR/VR field. In general, raw 360° images are transformed into 2D planar representations while preserving the omnidirectional information, e.g., equirectangular projection (ERP) and cube map projection (CP) to ensure compatibility with imaging pipelines. Omnidirectional images (ODIs) are sometimes projected back onto a sphere or transformed with different types of projection and rendered for display in certain applications.

However, the angular resolution of a 360° image tends to be lower than that of a narrow field-of-view (FOV) perspective image, as it is captured using a fisheye lens with an identical sensor size. Moreover, the 360° image quality can be degraded during a transformation between different image projection types. Therefore, it is imperative to super-resolve the low-resolution (LR) 360° image by considering various projections to provide high-level visual quality under diverse conditions. Early studies attempted to reconstruct high-resolution (HR) ODIs by interpolating the missing data between the LR image pixels [3, 5, 25].

Recently, deep learning (DL) has brought a significant performance boost to 2D single image super-resolution (SISR) [17, 37, 44]. These methods mostly explore super-resolving 2D LR image using high-capacity convolutional neural networks (CNNs) via, e.g., residual connections [21], and learning algorithms including generative adversarial networks (GANs) [18, 40, 41]. However, directly using these methods for 360° images represented in 2D planar representations is less applicable as the pixel density and texture complexity vary across different positions in 2D planar representations of 360° images, as pointed in [10].
Consequently, several attempts were made to address SR problems in relation to 360° imaging [10, 28, 36, 46]. In particular, 360-SS [28] proposes a GAN-based framework using the Pix2Pix pipeline [14], however, it focuses only on the ERP format and does not fully consider the properties of 360° images. LAU-Net [10] introduces a method to identify ODI distortions on the latitude and upsample ODI pixels on segmented patches. However, this process leads to considerable disconnections along the patches. In a nutshell, existing methods for ODI SR ignore the projection process of 360° images in real applications and only take the ERP image as the LR input, producing the HR ERP output. Indeed, a 360° image can be flexibly converted into various projection types, as in real applications, the user specifies the projection type, direction, and FOV. Thus, it is vital to address the ERP distortion problems and strive to super-resolve an ODI image to an HR image with an arbitrary projection type rather than a fixed type.

In this paper, as shown in Fig. 1, we propose a novel framework, called SphereSR, with the goal of super-resolving an LR 360° image to an HR image with an arbitrary projection type via continuous spherical image representation. First, we propose a feature extraction module that represents spherical data based on icosahedron and efficiently extracts features on a spherical surface composed of uniform faces (Sec. 3.1). As such, we solve the ERP image distortion problem and resolve the pixel density difference according to the latitude. Second, we propose a spherical local implicit image function (SLIIF) that can predict RGB values at arbitrary coordinates on a sphere feature map, inspired by LIIF [7] (Sec. 3.2). SLIIF works on triangular faces, buttressed by position embedding based on normal plane polar coordinates to obtain relative coordinates on a sphere. Therefore, our method tackles pixel-misalignment issue when the image is projected onto another ODI projection. As a result, SphereSR can predict RGB values for any SR scale parameters. Additionally, to train SphereSR, we introduce a feature loss that measures the similarity between two projection types, leading to a considerable performance enhancement (Sec. 3.3). Extensive experiments on various benchmark datasets show that our method significantly surpasses existing methods.

In summary, the contributions of our paper are four-fold. (I) We propose a novel framework, called SphereSR, with the goal of super-resolving an LR 360° image to an HR image with an arbitrary projection type. (II) We propose a feature extraction module that represents spherical data based on an icosahedron and extracts features on a spherical surface. (III) We propose SLIIF, which predicts RGB values from the spherical coordinates. (IV) Our method achieves the significantly better performance in the extensive experiments.

2. Related Works

Omnidirectional Image SR and Enhancement. Early ODI SR methods [2, 4, 6, 15, 26] focused on assembling and optimizing multiple LR ODIs on spherical or hyperbolic surfaces. On the other hand, as the distortion in ODI arises due to the projection of the original spherical image onto a 2D planar image plane, recent research has focused on tackling and solving distortion in ODI using 2D convolution to achieve a qualitative result in the observation space, i.e., a spherical surface. Su et al. [36] and Zhou et al. [46] proposed evaluation methods for ODI weighted with the projected area on a spherical surface. Two other works [27, 30] adapted existing SISR models to ERP SR by fine-tuning or by adding a distortion map as an input to tackle different distortions. Ozcinar et al. [28] leveraged GAN to super-resolve an ODI by applying WS-SSIM [46]. Zhang et al. [45] also proposed the GAN-based framework employing multi-frequency structures to enhance the panoramic image quality up to the high-end camera quality. Liu et al. [22] focused on the 360° image SR utilizing single-frame and multi-frame joint learning and a loss function weighted differently along the latitude. Deng et al. [10] considered varying pixel density and texture complexity along latitude by proposing network allowing distinct up-scaling factors along the latitude bands. Unlike the aforementioned methods, we propose to predict RGB values at the given spherical coordinates of an HR image with respect to an arbitrary project type from an LR 360° image.

2D SISR with an Arbitrary Scale. Research on SISR with an arbitrary scale has been actively conducted. Lim et al. [21] first proposed a method that enables multiple scale factors over one network. MetaSR [13] achieves SR with the non-integer scale factors. However, both methods are limited to SR with the symmetric scales. Later on, Wang et al. [39] proposed a framework that enables asymmetric scale factors along horizontal and vertical axes. Moreover, SRWarp [34] generalizes SR toward an arbitrary image transformation. Although these methods are effective for 2D SISR with an arbitrary scale factor, they fail in that they are not directly applicable to 360° image SR due to the difference between the xy-coordinate (2D) and the spherical coordinates in ODI domains. We overcome this challenge by proposing SphereSR, which leverages SLIIF to predict RGB values for arbitrary spherical coordinates.

Continuous Image Representation. Research on implicit neural representation (INR) has been conducted to express 3D spaces, e.g., 3D reconstruction and novel view synthesis via continuous ways [23, 24, 32]. Since then, continuous image representation has been explored on the (x,y) coordinate. Some studies used networks to predict the RGB value of each pixel from the latent vector on (x,y) coordinate without a spatial convolution for 2D image generation [1, 33]. LIIF [7] proposes to bridge between discrete and continu-
3. Method

Overview. As shown in Fig. 2, we propose a novel framework, SphereSR, the goal of which is to obtain continuous spherical image representation from a given icosahedron input. First, we introduce a feature extraction method for spherical images that efficiently extracts features from an image on an icosahedron (Sec. 3.1). Second, we propose the Spherical Local Implicit Image Function (SLIIF), which predicts RGB values through the extracted features in order to reconstruct an HR image flexibly with an arbitrary projection type (Sec. 3.2). Lastly, we propose a feature loss to obtain support from features of other projection types by utilizing the advantage of SLIIF, i.e., conversion to an arbitrary projection type (Sec. 3.3).

3.1. Feature Extraction for Spherical Images

Feature extraction is crucial yet challenging for spherical image SR, as we focus on very large scale factors, e.g., \( \times 16 \). In this situation, it is imperative to tackle the memory overload issue while ensuring high SR performance. Hence, the proposed SphereSR represents the spherical data based on an icosahedron and efficiently extracts features on a spherical surface composed of uniform faces. This is achieved by a new data structure on the icosahedron combined with weight sharing between kernels of different directions.

Data structure. Inspired by the convolution of icosahedron data in SpherePHD [19], we propose a new spherical data structure. To implement the convolution operation, SpherePHD [19] uses the subdivision process of an icosahedron, described in Fig. 3, and creates a call-table containing the indices of \( N \) neighboring pixels for each pixel, subsequently using it to stack every neighboring pixel. Convolution is then performed with a kernel of size \([N + 1, 1]\). However, this implementation is not memory-efficient, as it requires additional \( N \) channels for stacking the neighboring pixels for every convolution operation. To solve this problem, we propose a new data structure by which the convolution operation can be directly applied without stacking the neighbors in a call-table. As shown on the left side of Fig. 4, we rearrange the original data in the direction of the arrow while transforming the triangular pixels to rectangular pixels such that conventional 2D convolution can be applied.
Here, an upward kernel (red kernel) for each upward (△) aligned pixel is arranged in the odd-numbered rows, and a downward kernel (blue kernel) for each downward (▽) aligned pixel is arranged in the even-numbered rows. (More details can be found in the supplementary.)

Kernel Weight Sharing. While the memory overload issue can be resolved by the proposed data structure, it is still necessary to ensure high SR performance. SpherePHD [19] rotates each upward or downward kernel by 180° to obtain the same kernel shape. Therefore, it is possible to share weights of the up/downward kernels whose directions and shapes are symmetric to each other. However, as the direction of the kernel weight changes for adjacent pixels, high performance cannot be ensured if the characteristics of the texture according to the direction need to be identified.

To solve this problem, we introduce a kernel weight sharing scheme called GA-Conv which geometrically aligns up/downward directional kernels without rotation. As shown on the right side of Fig. 4, the pixel (face) combinations of two kernels, where up/downward kernels are applied, are shaped differently depending on the direction of the center pixel. However, if three vertices of the center pixel (denoted by the red and blue dots on the right side of Fig. 4) are included in the pixel combination as imaginary pixels, the shapes of two different up/downward pixel combinations can be made to be geometrically identical. To this end, rather than averaging and creating imaginary pixels, we distribute the kernel weight to nearby pixels. For the upward kernel, image pixel weights \( w_3 \), \( w_9 \), and \( w_{10} \) are distributed to the nearest four pixels, except the center pixel. The downward kernel weights \( w_4 \), \( w_5 \), and \( w_{11} \) are distributed in the same way. Details are presented on the right side of Fig. 4. In this way, the feature extraction module can be applied to any pixel without rotation.

3.2. Spherical Local Implicit Image Function (SLIIF)

Overall Process of SLIIF. With the data efficiently represented, we now describe the method used to super-resolve ODIs efficiently on an arbitrary scale. Our main idea is to predict an RGB value for an arbitrary coordinate on the unit sphere \( S^2 \) using a feature map extracted by means of GA-Conv, as described in Sec. 3.1. Inspired by LIIF [7], we propose SLIIF, which learns an implicit image function on \( S^2 \) using icosahedral faces. SLIIF takes a spherical coordinate of the point on the unit sphere and its neighboring feature vectors as inputs and predicts the RGB value. It can be formulated as:

\[
I(s) = f_{dec}(z, s), s \in S^2
\]

where \( f_{dec} \) is a decoding function shared with all icosahedral faces, \( s \) is the point on the unit sphere \( S^2 \), \( z \) represents a feature vector formed by concatenating the neighboring feature vectors of \( s \), and \( I(s) \) is the predicted RGB value of \( s \).

For a pixel in an image that can be formed by an arbitrary projection from the unit sphere, there is a corresponding point \( s \) on unit sphere \( S^2 \). The face containing \( s \) is denoted as \( f_s \), and the three vertices surrounding \( f_s \) are denoted as \( v_1^s \), \( v_2^s \), and \( v_3^s \) (see Fig. 5). The RGB values of \( s \) w.r.t. the coordinate system of the three vertices are first calculated and then assembled based on the triangular areas \( A_1, A_2, \) and \( A_3 \) to obtain the final RGB value of point \( s \). The RGB value of \( s \) w.r.t. each vertex \( v_j^s \) is calculated with the features of six faces containing the vertex and relative polar coordinate. The features of the six faces are concatenated clock-wise starting from \( f_s \) to preserve geometrical consistence. Here, we denote the concatenated features as \( z_j \) and the polar coordinate of \( s \) with respect of \( v_j^s \) as \( (r_j, \theta_j) \). To better utilize the positional information, the \( (r_j, \theta_j) \) values are encoded with \( \gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), ..., \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \)

Note: The coordinate of \( s \) is computed by using the center point of a pixel.
to extend the dimension of the relative coordinate, as introduced in [24, 38]. As such, we can predict the RGB value of point \((\theta, \phi) \in S^2\), which can be formulated as follows:

\[
I(\theta, \phi) = \sum_{j=1}^{3} \frac{A_j}{A} \cdot f_{\text{dec}}(z_j, [\gamma(r_j), \gamma(\theta_j)])
\]  

(2)

When the pixel in the image corresponds to the vertex on \(S^2\), we can still utilize the aforementioned procedure because any choice of neighboring vertices results in the same RGB value due to the triangle area-based weighting.

**Sphere-oriented Cell Decoding.** Through SLIIF, we can predict the RGB value for any point on \(S^2\). That is, we can generate the desired HR image for any projection type by predicting the RGB value of each pixel.

However, SLIIF provides the RGB value only for the center of the pixel and discards the information within the pixel area, except for the center value. To handle this, LIIF [7] defines the cell decoding value as the width and height of the query pixel. Nonetheless, this definition cannot be directly applied to a sphere, as the corresponding region on the sphere does not have a rectangular shape and the direction of the reference vertex, where the RGB value is initially calculated, continually changes. Thus, we propose sphere-oriented cell decoding, a method that takes into account the relation between the pixel region on the projected output and the corresponding region on \(S^2\). By adding the cell decoding value as input to SLIIF, we can fully utilize the information within the pixel area. As shown in Fig. 6, we aim to obtain the RGB value of a rectangular pixel on the projected plane. We call this rectangular pixel a plane cell, which can be expressed using the two vectors \(\Delta X, \Delta Y\). The sphere cell, the corresponding area of the plane cell on the sphere, is located on the face on which the corresponding point of pixel center is located on the sphere. The sphere cell can also be expressed using the two vectors \(\Delta x, \Delta y\). The relationship between \(\Delta X, \Delta Y\) and \(\Delta x, \Delta y\) depends on the projection type and location of the pixel center (refer to the supplementary for details).

For geometrical consistency of the orders of the concatenated features, the relative coordinate of \(s\), and the cell decoding value among the pixels, we need to define new axis vectors, \(\overrightarrow{n_1}\) and \(\overrightarrow{n_2}\), invariant to the face orientation. The unit vector \(\overrightarrow{n_1}\) is defined as a vector between the reference vertex and the face center, and the unit vector \(\overrightarrow{n_2}\) is defined by the 90 degree counter-clockwise rotation of \(\overrightarrow{n_1}\). To determine the height and width based on this coordinate system, we approximate the parallelogram sphere cell to an axis-aligned rectangle. The approximated sphere cell is defined as a rectangle which can be expressed using the two vectors \(\overrightarrow{\Delta x_{eq}}, \overrightarrow{\Delta y_{eq}}\) with an area identical to that of the parallelogram sphere cell and with the largest intersection area within the parallelogram sphere cell. Based on the approximated rectangular sphere cell, we finally formulate the sphere-oriented cell decoding value as shown below:

\[
\begin{align*}
\left(\begin{array}{c}
\Delta x \\
\Delta y
\end{array}\right) & \approx \left(\begin{array}{c}
\Delta x_{eq} \\
\Delta y_{eq}
\end{array}\right) = \left(\begin{array}{c}
c_x \overrightarrow{n_1} \\
c_y \overrightarrow{n_2}
\end{array}\right) \\
\Rightarrow c & = [c_x, c_y] = \left(\begin{array}{c}
|\overrightarrow{\Delta x_{eq}}| \\
|\overrightarrow{\Delta y_{eq}}|
\end{array}\right)
\end{align*}
\]  

(3)

(4)

As a result, we can predict the RGB value \(I(X,Y)\) for any point on the projected plane based on the following equation:

\[
I(X,Y) = I(\theta, \phi, c)
\]  

\[
= \sum_{j=1}^{3} \frac{A_j}{A} \cdot f_{\text{dec}}(z_j, [\gamma(r_j), \gamma(\theta_j)], [c_x, c_y])
\]  

(5)

### 3.3. Loss Function

We train the proposed framework using two loss terms. First, we use the multi-scale L1 loss. With the L1 loss defined on multiple scales, our framework can learn more about the relative coordinates and cell decoding values. Second, we design a feature loss module to measure the similarity between the features extracted from the sphere and other projection types.

As shown in Fig. 7, we design a feature mask from the ERP or cube map feature. The spatial part of the mask is generated from the difference between the predicted SR
ODI and the HR ground truth. The channel part of the mask is generated from the features via channel-wise global average pooling. In this way, we obtain a feature mask emphasizing the relevant parts with high accuracy. Also, the spherical features are converted to the shapes of other projection types via the SLIIF feature conversion module. Finally, the converted features are subtracted and masked to formulate the feature loss $L_{feat}$. The total loss is as follows:

$$\text{Loss} = \frac{1}{N} \sum_{j=1}^{N} \| I_{j}^{est} - I_{j}^{gt} \|_1 + \lambda L_{feat}$$  \hspace{1cm} (6)

4. Experiments

4.1. Dataset and Implementation

We train and test SphereSR using the ODI-SR dataset [10] and the SUN360 panorama dataset [42]. For training, 750 out of 800 ODI-SR training images are used and the remaining 50 images are used for validation. For testing, we use 100 images from the ODI-SR test dataset and another 100 images from the SUN360 panorama dataset. The resolution of an HR ODI is $1024 \times 2048$, and the ERP GT image is also interpolated using the bicubic method to the desired projection type for a performance evaluation. We use PSNR and SSIM as evaluation metrics.

4.2. Evaluation on ERP

We use the ODI-SR and SUN360 Panorama datasets for an evaluation. We compare SphereSR with 9 models for 2D SISR, including SRCNN [11], VDSR [16], LapSRN [17], MemNet [37], MSRN [20], EDSR [21], D-DBPN [12], RCAN [44], EBRN [29] and 2 models for ODI-SR, i.e., 360-SS [28] and LAU-Net [10]. We use WS-PSNR [46] and WS-SSIM [46] as evaluation metrics.

Quantitative results. Table 1 shows the results of quantitative comparisons of $\times 8$ and $\times 16$ SR on the ODI-SR and the SUN 360 panorama datasets. As shown here, SphereSR outperforms all of the other methods on both datasets, except in the $\times 8$ SR case on the ODI-SR dataset, where SphereSR shows performance comparable to that of LAU-Net. However, for $\times 16$ SR, SphereSR shows better performance compared to LAU-Net in WS-PSNR and WS-SSIM on both the ODI-SR and the SUN360 panorama datasets. Qualitative comparison. Figure 8 shows the results of a visual comparison of $\times 8$ SR images on the ODI-SR dataset. As shown here, SphereSR reconstructs clear textures and more accurate structures, while the other methods compared in this case are affected by the problems of blurred edges or distorted structures. From this visual comparison, we can conclude that SphereSR produces textures of repeated patterns more accurately than ERP networks.

4.3. SR for Other Projection Types

In this section, we verify whether the proposed SphereSR, trained using the ERP images on the ODI-SR dataset, can perform well for any projection type. First, we conduct an experiment involving conversion to a FOV $90^\circ$ perspective image with a size $512 \times 512$. We then conduct another experiment on the conversion to a FOV $180^\circ$ fisheye image with a size $1024 \times 1024$. In addition, we use circular fisheye projection, one of several types of fisheye projections. For a comparison with other SR models, we use bicubic interpolation to convert to the desired projection type. The ERP GT image is also interpolated using the bicubic method to the desired projection type for a performance evaluation. We use PSNR and SSIM as evaluation metrics. Note that we select five random directions, generate a projection output suitable for the corresponding direction, and calculate the mean value for PSNR and SSIM.

Perspective Image. Table 2 shows the quantitative results for perspective image SR. SphereSR again achieves the best performance on both the ODI-SR and the SUN360 datasets. LAU-Net [10] achieves PSNR values of 26.39dB
Figure 8. Visual comparisons of x8 SR results of different methods on ODI-SR dataset.

Figure 9. Visual comparison for x8 SR of perspective images on ODI-SR dataset.

Figure 10. Visual comparison for x8 SR of fisheye images on ODI-SR dataset.

Table 2 shows the quantitative results for fisheye image SR. It can be seen that SphereSR has the highest performance in terms of the PSNR and SSIM values on the ODI-SR and SUN360 panorama datasets. Among the methods for 2D SISR, RCAN achieves the second-highest PSNR value of 24.40dB on the ODI-SR dataset. On the SUN360 panorama dataset, LAU-Net achieves the second-highest PSNR of 24.97dB. Our method results in the highest PSNR and SSIM values, showing the best SR performance. In Fig. 10, we show a visual comparison between SphereSR with SLIIF, SphereSR without SLIIF, RCAN [44] and LAU-Net [10]. Specifically, we crop the area to view the SR results at the south pole. As shown in the figure, RCAN(b) and LAU-Net(c) generate inappropriate textures with several lines rushing to the south pole. On the other hand, SphereSR(w/o SLIIF)(d) and SphereSR(w/ SLIIF)(e) do not encounter such a problem. Moreover, in the case of (e), it eliminates the triangle-shaped artifact generated in (d).

4.4. Ablation Study and Analysis

In this section, we study the effectiveness of each of our proposed modules, e.g., GA-Conv, SLIIF, and feature loss. In addition, we validate the memory load during CNN operation using the proposed data structure and using SpherePHD [19].

GA-Conv. We compare the results of Models 1 and 3 in Table 3 when adding or removing GA-Conv. GA-Conv is used in the feature extraction module to obtain the feature vector to be used in SLIIF. If GA-Conv is not used, the kernel weight sharing proposed by SpherePHD [19], which gives 180 degrees rotation per kernel, is used. Table 3 shows that using GA-Conv improves the PSNR score by 0.14dB and 0.12dB on the ODI-SR and SUN360 Panorama datasets, respectively, for x8 SR.

SLIIF. SphereSR uses SLIIF to present SR results for the ERP projection type through the feature vectors presented on the sphere. Identical to an earlier method [31], we implement a pixel-shuffle algorithm capable of performing SR on an icosahedron without SLIIF for a performance comparison. When using the pixel-shuffle step, the last feature map was subdivided by the scale factor multiple, after which the
Table 2. Perspective and fisheye SR results on the ODI-SR and SUN 360 Panorama Dataset. **Bold** indicates the best results.

<table>
<thead>
<tr>
<th>Projection Type</th>
<th>Perspective</th>
<th>Fisheye</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>PSNR</td>
<td>SSIM</td>
</tr>
<tr>
<td>Bicubic</td>
<td>23.40</td>
<td>0.6858</td>
</tr>
<tr>
<td>SRCNN [11]+Bicubic</td>
<td>26.04</td>
<td>0.7005</td>
</tr>
<tr>
<td>EDSR [21]+Bicubic</td>
<td>26.53</td>
<td>0.7192</td>
</tr>
<tr>
<td>D-DBPN [12]+Bicubic</td>
<td>26.59</td>
<td>0.7139</td>
</tr>
<tr>
<td>RCAN [44]+Bicubic</td>
<td>26.70</td>
<td>0.7191</td>
</tr>
<tr>
<td>360-SS [28]+Bicubic</td>
<td>23.28</td>
<td>0.6528</td>
</tr>
<tr>
<td>LAU-Net [10]+Bicubic</td>
<td>26.39</td>
<td>0.7197</td>
</tr>
<tr>
<td>SphereSR(w/o SLIIF)(+Bicubic)</td>
<td>26.66</td>
<td>0.7176</td>
</tr>
<tr>
<td>SphereSR(Ours)</td>
<td><strong>26.76</strong></td>
<td><strong>0.7208</strong></td>
</tr>
</tbody>
</table>

Table 3. Ablation studies on ERP SR on ODI-SR and SUN360 Panorama Dataset for both ×8 and ×16 SR.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Component</th>
<th>Feature loss</th>
<th>ODI-SR</th>
<th>SUN 360 Panorama</th>
<th>ODI-SR</th>
<th>SUN 360 Panorama</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GA-Conv</td>
<td>SLIIF</td>
<td>WS-PSNR</td>
<td>WS-SSIM</td>
<td>WS-PSNR</td>
<td>WS-SSIM</td>
</tr>
<tr>
<td>Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>✓</td>
<td>24.20</td>
<td>0.6688</td>
<td>23.98</td>
<td>0.6719</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>x</td>
<td>24.31</td>
<td>0.6731</td>
<td>24.07</td>
<td>0.6749</td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td>✓</td>
<td>24.34</td>
<td>0.6765</td>
<td>24.10</td>
<td>0.6816</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>24.37</td>
<td>0.6777</td>
<td>24.17</td>
<td>0.6820</td>
</tr>
</tbody>
</table>

Table 4. Comparisons of activation memory between SpherePHD and the proposed data structure. The network architectures of SpherePHD and ours have the same number of convolution layers (16) and hidden feature dimension (32).

<table>
<thead>
<tr>
<th>Level</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpherePHD(MB)</td>
<td>660</td>
<td>1896</td>
<td>6714</td>
<td>26032</td>
</tr>
<tr>
<td>New Data Structure(MB)</td>
<td><strong>374</strong></td>
<td><strong>724</strong></td>
<td><strong>2138</strong></td>
<td><strong>7450</strong></td>
</tr>
</tbody>
</table>

final ERP output was derived by means of bicubic interpolation. As shown in Models 2 and 3 in Table 3, using SLIIF for continuous image presentation achieves higher performance (24.34dB vs. 24.31dB) than the pixel-shuffle method of subdivision of the icosahedron for ×8 SR.

**Feature loss.** We propose a feature loss that measures the feature similarity of the crucial areas through feature masking using features generated from other projection types. To confirm the effectiveness of feature loss, we compare the SR performance when adding and removing this loss. Models 3 and 4 in Table 3 show the ablation results. The results of the performance comparison indicate a performance improvement for all metrics in the ×8 SR scale.

**Data Representation Efficiency.** In Sec. 3.1, we point out that the CNN implementation of SpherePHD is not efficient for SR. We thus propose a new data structure to tackle this problem. To ascertain the efficiency of the new data structure, we implement a simple CNN model and then conduct an experiment to compare the activation memory. The CNN model is a simple structure in which the convolution layers are stacked; the number of convolution layer is set to 16 and the hidden feature dimension is set to 32. Table 4 shows the experiments from input level 4 to input level 7. As indicated in the table, the new data structure in GA-Conv has a much lower activation memory level. In addition, it is found that as the input level is increased, the ratio of using new data structure memory to SpherePHD decreases. Based on this, the proposed data structure is shown to be more efficient in terms of memory compared to SpherePHD. Moreover, the efficiency increases as the input resolution is increased.

5. Conclusion

In this paper, we proposed a novel framework, SphereSR, which generates a continuous spherical image representation from an LR 360° image. SphereSR predicts the RGB values at the given spherical coordinates of an HR image corresponding to an arbitrary project type. First, we proposed geometry-aligned convolution to represent spherical data efficiently, after which we proposed SLIIF to extract RGB values from the spherical coordinates. As such, SphereSR flexibly reconstructed an HR image with an arbitrary projection type and SR scale factors. Experiments on various benchmark datasets demonstrated that our method significantly surpasses existing methods.

**Limitation and Future Work.** We focused on finding an efficient data structure and kernel weight sharing method to extract meaningful features with the ODI input based on GA-Conv (Sec. 3.1). Future studies will therefore need to improve the network architecture using the properties of ODIs compared to perspective images, and then we can achieve better SR results via SLIIF.

**Acknowledgement.** This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2022R1A2B5B03002636).
References


[38] Hajime Nagahara, Yasushi Yagi, and Masahiko Yachida. Super-resolution from an omnidirectional image sequence. volume 4, pages 2559–2564 vol.4, 2000. 2


[34] Sanghyun Son and Kyoung Mu Lee. SRWarp: Generalized image super-resolution under arbitrary transformation. In CVPR, 2021. 2


[44] Yulun Zhang, Kunpeng Li, Kai Li, WangLichen, Binzhong, and Yun Raymond Fu. Image super-resolution using very deep residual channel attention networks. In ECCV, 2018. 1, 6, 7, 8
