Applying VertexShuffle Toward 360-Degree Video Super-Resolution

Na Li
Rutgers University
Piscataway, NJ, USA
na.li@rutgers.edu

Yao Liu
Rutgers University
Piscataway, NJ, USA
yao.liu@rutgers.edu

ABSTRACT

With the recent successes of deep learning models, the performance of 2D image super-resolution has improved significantly. Inspired by recent state-of-the-art 2D super-resolution models and spherical CNNs, in this paper, we design a novel spherical super-resolution (SSR) approach for 360-degree videos. To address the bandwidth waste problem associated with 360-degree video transmission/streaming and save computation, we propose the Focused Icosahedral Mesh to represent a small area on the sphere and construct matrices to rotate spherical content to the focused mesh area. We also propose a novel VertexShuffle operation on the mesh, motivated by the 2D PixelShuffle operation. We compare our SSR approach with state-of-the-art 2D super-resolution models. We show that SSR has the potential to achieve significant benefits when applied to spherical signals.

CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Computing methodologies → Image processing**.

KEYWORDS

VertexShuffle, 360-degree videos, Focused Icosahedral Mesh

ACM Reference Format:
Na Li and Yao Liu. 2022. Applying VertexShuffle Toward 360-Degree Video Super-Resolution. In *The 32nd edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'22), June 17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3534088.3534353

1 INTRODUCTION

360-degree image/video, also known as spherical image/video, is an emerging format of media that captures views from all directions surrounding the camera. Unlike traditional 2D image/video that limits the user's view to wherever the camera is facing during capturing, a 360-degree image/video allows the viewer to freely navigate a full omnidirectional scene around the camera position.

Despite its substantial promise of immersiveness, the utility of streaming/transmission of 360-degree video is limited by the huge bandwidths required by most implementations. For example, when watching a 360-degree video, users can only watch a small portion of the full omnidirectional view. That is, while the 360-degree video encodes frames that cover the full $360^\circ \times 180^\circ$ field-of-view (FoV), the user may only observe a "view" (e.g., $100^\circ \times 100^\circ$ FoV) of the omnidirectional frame at a time. If the omnidirectional frame is projected to the 2D frame using the equirectangular projection [1], then only roughly 15% of the pixels on the frame is viewed. The rest 85% pixels are not viewed, resulting in significant bandwidth waste when streaming 360-degree videos. To improve the bandwidth efficiency, a number of spatial adaptation approaches have been proposed [8, 16, 26, 27, 31, 33]. A core difficulty with such approaches involves predicting which portions of the 360-degree frame will be viewed by the user. If accurate predictions can be made of the user's behavior (a difficult task), then only the portions of the 360-degree frame to be viewed need to be transmitted. A "soft" version of this family of approaches involves transmitting spatial portions of the 360-degree video at a level of quality in proportion to the probability that they will be viewed [37]. These approaches, however, can still suffer from difficulties, as there can be significant delays between the time that view prediction is made and the time that spatial portions of the sphere are actually viewed, resulting in inaccurate predictions [3].

A recent approach toward mitigating these types of mis-predictions involves applying super-resolution (SR) [19] to low-resolution video segments already present in the playback buffer. The super-resolution task requires reconstructing a high-resolution image from low-resolution input. To date, many approaches for super-resolution over standard 2D images via deep convolutional network have been proposed [10–12, 32, 36]. However, a problem with the current practice of 360-degree videos is the distortions caused by spherical-to-2D projections. The omnidirectional views captured by 360-degree cameras are most naturally represented as uniformly dense pixels over the surface of a sphere. When spherical pixels are projected to planar surfaces, distortions are introduced. For example, the equirectangular projection [1] is a widely used spherical projection for representing 360-degree data. However, significant distortions occur around the north and south pole areas in the projection. Such distortions can reduce the efficiency of CNN operations by adding "over-represented" pixels, Further, training a CNN directly on the distorted representation could cause CNN models to learn characteristics of the planar distortion rather than relevant details of the high resolution representation.

Recent works on spherical CNNs [6, 20] perform convolutional operations directly on spherical signals to avoid the distortion issue. Their works show that it is possible to analyze spherical signals directly without 2D projections. Furthermore, extensive experiments were conducted in these works to show the efficiency of their proposed spherical CNNs.

In this paper, inspired by recent advances in spherical CNNs [6, 20] and state-of-the-art 2D super-resolution methods [10–12, 32, 36], we propose a spherical super-resolution approach that operates on a direct, mesh representation of spherical pixels of 360-degree videos. First, we propose an efficient mesh representation, the Focused Icosahedral Mesh, This representation both makes our SR approach compatible with 360-degree spatial adaptation and improves memory efficiency of the training and prediction steps of model operation compared to Full Icosahedral Mesh. Second, motivated by the 2D PixelShuffle operation [30], we propose a novel VertexShuffle operation. The VertexShuffle operation significantly increases both the visual quality metric (peak signal-to-noise ratio (PSNR)) and improves inference time over comparable transposed convolution operations. Evaluation results show that our spherical super-resolution approach can achieve 31.97 dB PSNR on average when super-resoluting 16x vertices on the mesh created from 360-degree video inputs.

2 RELATED WORK

2.1 360-degree video

Despite its potential for delivering more-immersive viewing experiences than standard video streams, current 360-degree video implementations require bandwidths that are too high to deliver satisfying experiences for many users. Numerous approaches have been proposed for improving 360-degree bandwidth efficiency. These approaches have both attempted to improve the efficiency of how the 360-degree view is represented during transmission [8, 15, 16, 26–29, 31, 33, 34] as well as improving a system's ability to avoid delivering unviewed pixels [37].

2.2 Spherical convolutional neural networks

Spherical CNN has been studied by the computer vision community recently as a number of real-world applications require processing signals in the spherical domain, including self-driving cars, panoramic videos, omnidirectional RGBD images, and climate science. Recent works such as Cohen et al. [6] gave theoretical support of spherical CNNs for rotation-invariant learning problems.

UGSCNN [20] presents a novel CNN approach on unstructured grids using parameterized differential operators for spherical signals. It introduces a basic convolution operation, called MeshConv, that can be applied on meshes directly. It achieves significantly better performance and parameter efficiency compared to state-of-the-art network architectures for 3D classification tasks since it does not require large amounts of geodesic computations and interpolations. Zhang et al. [35] proposed to perform semantic segmentation on omnidirectional images by designing an orientation-aware CNN framework for the icosahedron mesh. They introduced fast interpolation of kernel convolutions and presented weight transfer from learned through classical CNNs to perspective data. Recently, Eder et al. [13] proposed a spherical image representation that mitigates spherical distortion by rendering a set of oriented, low-distortion images tangent to icosahedron faces. They also presented the utilities of their approaches by applying standard CNN to the spherical data.

While these existing works demonstrate their effectiveness in classification and segmentation tasks, the super-resolution task was not considered. In this work, we find it possible to apply their work on the super-resolution task. Our work is based on the MeshConv operation proposed by Jiang et al. [20] since it achieves better performance and parameter efficiency than other spherical convolutional networks. We also conduct experiments to show significant improvements over the baseline spherical super-resolution model that uses the simple MeshConv Transpose operation [20].

2.3 Super-resolution

Research in super-resolution has advanced rapidly from its origins in the deep learning age. The SRCNN [10, 11] model was the first to apply CNNs to SR. FSRCNN [12] was an evolution of SRCNN. It operated directly on a low-resolution input image and applied a deconvolution layer to generate the high-resolution output. VDSR [21] was the first to apply residual layers [17] to the SR task, allowing for deeper SR networks. DRCN [22] introduced recursive learning in a very deep network for parameter sharing. Shi et al. [30] proposed "PixelShuffle", a method for mapping values at low-resolution positions directly to positions in a higher-resolution image more efficiently than the deconvolution operation. SRResNet [24] introduced a modified residual layer tailored for the SR application. EDSR [25] further modified the SR-specific residual layer from SSResNet and introduced a multi-task objective in MDSR. SRGAN [24] applied a Generative Adversarial Network (GAN) [14] to SR, allowing better resolution of high-frequency details. These works focus on 2D planar data, which may not be ideal for 360-degree image super-resolution due to the distortions introduced in the projected representation. Our proposed model, however, operates directly on spherical signals so that we can avoid the distortion problem. Focusing on optimizing 360-degree video streaming, Chen et al. [5] and Dasari et al. [9] applied existing 2D super-resolution to 360-degree video tiles. Unlike their works, we focus on designing a novel spherical super-resolution approach for 360-degree images/videos.

3 METHODOLOGY

In this section, we first introduce Focused icosahedral mesh for representing a small area of the sphere. We then illustrate the novel VertexShuffle operation after a brief discussion of the MeshConv operation proposed by Jiang et al. [20]. Finally, we describe our model architecture and loss function.

3.1 Focused icosahedral mesh

The icosahedral spherical mesh [4] is a discretization of the spherical surface. It starts with a unit icosahedron (i.e., an icosahedron with all 12 vertices re-projected to the unit sphere). The icosahedral spherical mesh can then be obtained by first selecting midpoints of all edges, progressively sub-dividing each face of the unit icosahedron into four equal triangles by connecting 3 midpoints of the face, and re-projecting the midpoints to the unit sphere.

Operations on a full spherical mesh, refined to a granularity that can include approximately all pixels from a planar representation of a 360-degree video frame, however, require a significant amount of computation. In addition, operations on the full mesh cannot

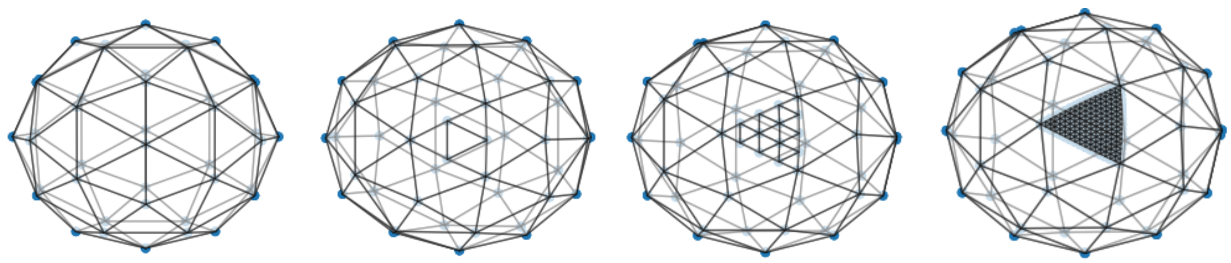

(a) **Full** Level-1 Mesh
(b) **Focused** Level-2 Mesh
(c) **Focused** Level-3 Mesh
(d) **Focused** Level-5 Mesh

Figure 1: Example of meshes in Level-1, Level-2, Level-3 and Level-5. To create "Focused icosahedral meshes", we select one face in the full Level-1 mesh and repeatedly refine triangles in this Level-1 face to obtain Focused Level-X meshes.

easily support operations on sub-areas of the spherical surface. Performing super-resolution on "sub-areas" of the spherical surface can be beneficial for real-world 360-degree applications. This is because human eyes as well as their viewing devices (e.g., the head-mounted display) have limited field-of-view (FoV), usually represented as the angular extent of the field that can be observed. To render the view shown in this figure, only part of the sphere is required. Such "sub-areas" would be useful in "tiling" schemes that can be used to support spatial-adaptive super-resolution over the 360-degree view. That is, if only a small area on the sphere will be viewed by the user, we may only need to apply super-resolution to a sub-portion of the sphere instead of the full sphere. As a result, performing super-resolution on the full icosahedral mesh may no longer be necessary as it requires more computation resources.

To support both faster operation and super-resolution on a sub-portion of the sphere, we propose a partial refinement scheme to generate the "Focused Icosahedral Mesh". We first create a Level-1 icosahedral mesh by refining each face on a unit icosahedron into 4 faces. In this way, the 20-face icosahedron is refined into a Level-1 icosahedral mesh with 80 faces. An example full Level-1 mesh with 80 faces is shown in Figure 1(a). We then select one face out of the 80 faces of the Level-1 icosahedral mesh and only refine triangles located inside the selected Level-1 face.

Specifically, in our focused mesh representation, we select the face of the Level-1 mesh that covers the position of <latitude=0, longitude=0> on the sphere since very little distortion is introduced when pixels near this area are projected to the 2D plane. Figure 1(b) shows the Focused Level-2 mesh where the selected Level-1 face is refined into 4 smaller faces. Figures 1(c) and 1(d) show the Focused Level-3 and Focused Level-5 meshes, respectively.

3.1.1 Rotating content to the Focused Level-1 origin. While a full Level-1 mesh has 80 faces, we only generate one single Focused icosahedral mesh by refining one selected face, and our spherical super-resolution model only operates on this single Focused icosahedral mesh. To allow our model to perform super-resolution for any area on the sphere, we thus need to map spherical pixel content that belongs to any arbitrary full Level-1 mesh face to the face that is selected to be refined. To do so, we pre-compute a rotation matrix $R \in \mathbb{R}^{N_F \times N_V \times C}$, where N_F represents the total number of faces in a full Level-1 mesh, which is 80, and N_V is the number of vertices in a Level-1 face, as shown in Figure 1. A Level-1 mesh consists of 80 triangles, each containing 3 vertices. C represents the number of dimensions of Euclidean coordinates in the sphere, namely xyz.

We denote the Level-1 face selected to be refined as face F_0. To rotate an arbitrary face $F_i, i \in (0, 80)$ on the Level-1 mesh to the refined face F_0, we need to find a rotation matrix R_i for face F_i such that $F_i = R_i \cdot F_0$, where F_i and F_0 are 3×3 matrices that represent the xyz coordinates of three vertices of a triangle face.

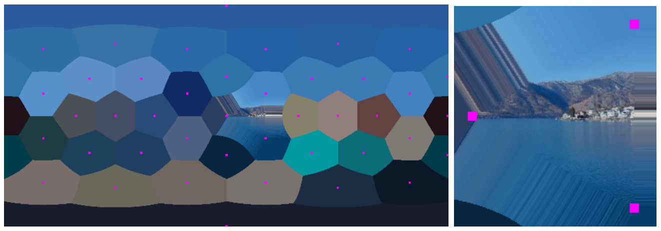
Therefore, we can obtain R_i as: $R_i = F_i \cdot F_0^{(-1)}$. We first rotate the vertices in the Focused Level-X Mesh with the rotation matrix R, and then compute a mapping from each pixel in the input planar representation (e.g., an equirectangular image) to the rotated Focused Level-X vertex. Instead of refining all 80 faces in a mesh, in this way, we can represent all 80 different faces on the full Level-1 mesh through a single Focused Mesh file with the most important face refined and other faces discarded, which has the potential to save a significant amount of computation and storage resources and achieves better parameter efficiency.

Figure 2 visualizes how one focused icosahedral mesh can be used to represent all 80 different Level-1 faces. Figure 2(a) shows an original equirectangular-projected 360-degree image. In this image, we highlight two areas marked by magenta circles. In Figure 2(b), the left-hand-side image shows the Focused Level-9 mesh visualized on an equirectangular image. Magenta points in this figure represent vertices in the full Level-1 mesh. There are 42 vertices in the full Level-1 mesh. The right-hand-side image in Figure 2(b) magnifies the refined face in the Focused icosahedral mesh to show details. We can see that content in this face are in the same position as in the original equirectangular-projected image. Figure 2(c) shows the resulting visualization when we rotate a different Level-1 face to the refined face. The image on the right magnifies the refined face to show details.

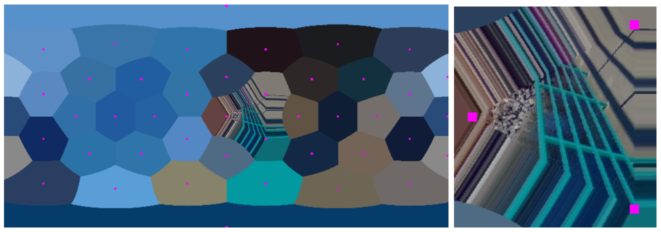
3.1.2 Mesh sizes. Table 1 shows the number of vertices in both Full and Focused icosahedral meshes in different levels of refinement. A Full Level-9 mesh has more than 2.6 million vertices and requires more than 1.9 GB of space for storage. On the other hand, a Focused Level-9 mesh has only about 33K vertices, requiring only about 24 MB storage space.

We know that the area of a unit spherical surface is 4π. A frame generated through the equirectangular projection covers a corresponding area of $2\pi \times \pi = 2\pi^2$. Suppose there are N_x vertices in the Full Level-X mesh, given that vertices on the icosahedral mesh are roughly uniformly distributed on the sphere, we can estimate the equivalent 2D equirectangular-projected frame resolution as follows: $W = \sqrt{N_x \times \pi}$, $H = W/2$, where W and H are the width and height of the equirectangular projection, respectively. The results

(a) This figure shows an equirectangular-projected 360-degree image. Magenta circles, b and c, in this figure mark areas corresponding to two different refined faces. (Original photo by Timothy Oldfield on Unsplash: https://unsplash.com/photos/luufnHoChRU)



(b) The left-hand-side image displays the Focused Level-9 mesh visualized on an equirectangular image. The right-hand-side image displays a magnified view of the refined face. In both images, magenta points represent vertices in the full Level-1 mesh.



(c) This figure displays a different Level-1 icosahedral face rotated to the face refined in the Focused Mesh. Pixel values from the original image are attached to rotated vertices by inverting the rotation for positions of the mesh vertices then finding the nearest neighbor pixel of this rotated position.

Figure 2: Visualizing the Focused Icosahedral mesh.

are listed in Table 1. We find that Level-6 mesh is roughly equivalent to the 2D equirectangular projection in 360x180 resolution, and that Level-9 mesh is roughly equivalent to the 2D equirectangular projection in 2880x1440 resolution.

3.2 MeshConv

The MeshConv operation introduced by Jiang et al. [20] is performed by taking a linear combination of linear operator values computed on a set of input mesh vertex values. MeshConv can be formulated as follows:

$$\text{MeshConv}(F;\ \theta) = \theta_0 IF + \theta_1 \nabla_{lat} F + \theta_2 \nabla_{lng} F + \theta_3 \nabla^2 F, \quad (1)$$

where I represents the identity, which can be regarded as the $0th$ order differential, same as ∇_{00}. ∇_{lat} and ∇_{lng} are derivatives in two orthogonal spatial dimensions, which can be viewed as the $1st$ order differential. ∇_2 stands for the Laplacian operator, which can be considered as the $2nd$ order differential.

Table 1: Number of vertices in Full icosahedral mesh, Focused icosahedral mesh, and their roughly-equivalent 2D planar resolution in the equirectangular projection.

Level	Level-6	Level-7	Level-8	Level-9
Full	40,962	163,842	655,362	2,621,442
Focused	600	2,184	8,424	33,192
2D planar	360x180	720x360	1440x720	2880x1440

At a high level, these linear operators can be viewed as computing a set of local information near each vertex of the mesh. The standard 3×3 cross correlation operation can be viewed as a set of nine linear operators. Each of the linear operators returns a value of either the pixel itself or an adjacent pixel. Compared to the 3×3 convolution, it is clear that the set of four linear operators used by MeshConv is less expressive. They not only extract less information per pixel, but this information also can drop information about a vertex's surrounding. For example, the gradient operation on the mesh computes a 3-dimensional average of either six or seven values. Another degree-of-freedom is dropped from the gradient when taking only the east-west and north-south components of the gradient. We hypothesize that some of the information excluded from the linear operator computations could be useful for the super-resolution task. To attempt to mitigate this information loss, rather than including single MeshConv ops in our network architecture, we include pairs of composed MeshConv ops (as shown in Figure 3 ResBlock depiction). These paired operations aggregate more local information around a vertex before the non-linearity is applied, allowing the network to capture more-useful characteristics needed for the super-resolution task.

3.3 VertexShuffle

MeshConv Transpose. To upscale the downsampled low level mesh in semantic segmentation task, UGSCNN [20] proposed a MeshConv Transpose operation. MeshConv Transpose takes Level-i mesh for input and outputs a Level-$(i+1)$ mesh. It can be described as follows:

$$M_{i+1} = \text{MeshConv}(\text{Padding}(M_i)), \quad (2)$$

where Padding represents zero padding, M_{i+1} and M_i are Level-$(i+1)$ mesh and Level-i mesh, respectively. In general, MeshConv Transpose simply pads 0s on new vertices in Level-$(i+1)$ mesh, then applies a regular MeshConv on the new zero-padded Level-$(i+1)$ mesh. While this operation is easy to implement, it is inefficient.

VertexShuffle. Motivated by PixelShuffle [30] commonly used in 2D super-resolution models, we propose a novel VertexShuffle operation to use in our spherical super-resolution model. The VertexShuffle operation can be described as follows:

$$M_{i+1} = \text{VertexShuffle}(M_i) \quad (3)$$

$$M_i = M_{i0}, M_{i1}, M_{i2}, M_{i3} \quad (3a)$$

$$N'_{ij} = \text{MidPoint}(M_{i(j+1)}), j \in \{0, 1, 2\} \quad (3b)$$

$$N'_i = concat(N'_{i0}, N'_{i1}, N'_{i2}) \quad (3c)$$

$$N_i = unique(N'_i) \quad (3d)$$

$$\text{VertexShuffle}(M_i) = concat(M_{i0}, N_i) \quad (3e)$$

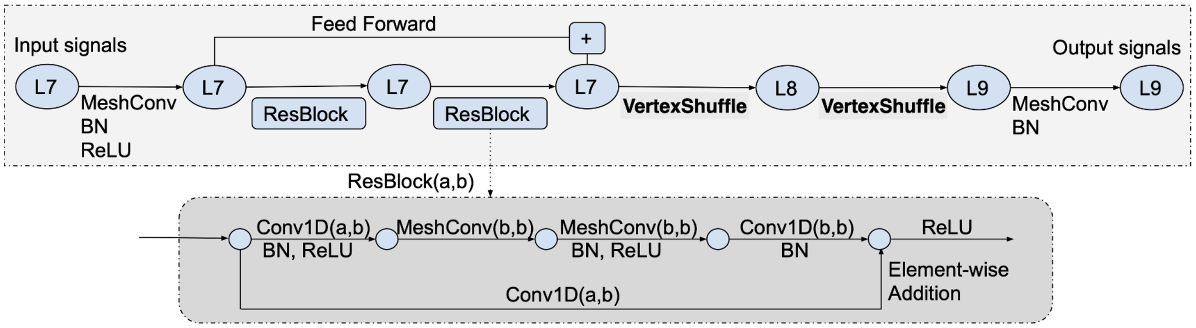


Figure 3: Architecture of our proposed Spherical Super-Resolution (SSR) model that uses MeshConv and VertexShuffle. L7 represent the input Level-7 mesh, and L9 represents the output Level-9 mesh.

The input of our basic VertexShuffle operation can be represented as $M_i \in \mathbb{R}^{F \times V_i}$, where F is the feature dimension in Level-i, and V_i represents the number of vertices of Level-i mesh. The output is $M_{i+1} \in \mathbb{R}^{F' \times V_{i+1}}$, where F' is the feature dimension in Level-$(i+1)$, which is $F/4$ in our work, and V_{i+1} represents the number of vertices of the Level-$(i+1)$ mesh.

We first split M_i into four parts $\{M_{i0}, M_{i1}, M_{i2}, M_{i3}\}$ along feature map dimension, where $M_{ij} \in \mathbb{R}^{F' \times V_i}, j = \{0, 1, 2, 3\}$ and $F' = F/4$. We keep M_{i0} as our Level-i mesh, which will be used later. M_{i1}, M_{i2}, M_{i3} are used to refine vertices in Level-$(i+1)$ mesh.

As we introduced before, a spherical mesh can be obtained by progressively sub-dividing each face of the unit icosahedron into four equal triangles. Here, we treat a single triangle face as a sequence of vertices, v_0, v_1, v_2 and a sequence of edges $(v_0, v_1), (v_1, v_2), (v_2, v_0)$. The refinement process can be regarded as progressively constructing midpoint vertex on associated edges, and new edges in Level-$(i+1)$ are created between each pair of midpoint vertices, thus a single face in Level-i is refined into four new faces in Level-$(i+1)$.

To fully make use of feature maps in Level-i, we use M_{i1}, M_{i2}, M_{i3} to refine vertices in Level-$(i+1)$ mesh. Specifically, we use M_{i1} to calculate midpoint between (v_0, v_1), M_{i2} to calculate midpoint between (v_1, v_2), and M_{i3} to calculate midpoint between (v_2, v_0). Midpoint vertex values are constructed by averaging the values associated with the original two vertices on a edge. Thus, equation 3b can be described as follows:

$$N'_{i0} = \text{MidPoint}(M_{i1}) = (M_{i1}(v_0) + M_{i1}(v_1))/2 \quad (4a)$$

$$N'_{i1} = \text{MidPoint}(M_{i2}) = (M_{i2}(v_1) + M_{i2}(v_2))/2 \quad (4b)$$

$$N'_{i2} = \text{MidPoint}(M_{i3}) = (M_{i3}(v_2) + M_{i3}(v_0))/2 \quad (4c)$$

Thus, we can get a set of midpoint vertices N'_i, which are new vertices generated in Level-$(i+1)$ mesh. However, there exist redundant midpoints due to the shared edges that may be calculated twice. We have to perform deduplication on the set of midpoint vertices. There can be a few ways to select midpoint between the two calculated midpoints, such as `max`, `min`, `average`, and `weighted average`. In this work, we simply select the first instance of a midpoint. Then, we have a set of unique midpoint vertices that used to refine the next level mesh $N_i \in \mathbb{R}^{F' \times A_i}$, where $A_i = V_{i+1} - V_i$. Finally, we concatenate partial feature map in Level-i, M_{i0}, with the new calculated midpoint vertices N_i to create our Level-$(i+1)$ mesh.

Compared to MeshConv Transpose, VertexShuffle does not have extra learnable parameters. Thus, our implementation of VertexShuffle is not only more parameters efficient, but also achieves significantly better performance.

3.4 Model architecture

We apply our Focused icosahedral mesh and VertexShuffle operation in the super-resolution task. The architecture of our spherical super-resolution (SSR) model is shown in Figure 3. In this figure, we show the input of our model as a Level-7 Focused icosahedral mesh, it first goes through a MeshConv layer with Batch Normalization [18] followed by a ReLU activation function. Then, we use two adapted Residual Blocks [17] to further extract features. The adapted residual blocks include two MeshConv layers (as discussed in Section 3.2). We concatenate the output of the first MeshConv and the output from the two ResBlocks by element-wise addition. After that, we use two VertexShuffle layers to upscale feature maps. Finally, our model ends up with a MeshConv layer, generating a Level-9 Focused icosahedral mesh with the correct number of channels.

3.5 Loss function

Similar to general super-resolution tasks, our goal is to minimize the loss between the reconstructed images Y_i and the corresponding ground truth high-resolution images H_i. Given a set of high-resolution images H_i and their corresponding low-resolution images X_i, we represent the loss as follows:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (H_i - Y_i)^2, \text{ where } Y_i = F(X_i) \quad (5)$$

$$\text{Loss} = 10 \times \log_{10}(\text{MSE}) \quad (6)$$

where N is the number of training samples. The loss function is the negative peak signal-to-noise ratio (PSNR), which is more straightforward for our task.

4 EVALUATION

We used a publicly-available 360-degree video dataset [7] for experiments. This dataset contains 5 videos of 4K quality at the frame rate of 30 frames-per-second (fps). For each video in the dataset, we use FFmpeg [2] to cut the full video temporally into segments of 1-second long each. For each segment, we train a super-resolution

Table 2: PSNR (dB) results for all 5 videos in our testing dataset.

Model	Diving	Timelapse	Venice	Paris	Rollercoaster	Average
2D with Upsample	30.34	31.75	35.90	28.00	25.31	30.26
2D with PixelShuffle	33.42	34.13	40.07	32.45	33.62	34.74
Spherical: MeshConv with transposed MeshConv	31.50	30.27	38.28	26.64	28.14	30.97
Spherical: MeshConv with VertexShuffle (**SSR**)	33.20	32.06	33.72	28.90	31.99	31.97

Table 3: Comparison of total number of model parameters, model storage size, and per-frame inference time.

Model	Total # of Parameters	Storage Size	Per-frame Inference Time
2D with Upsample	86163	343 KB	23.39 ms
2D with PixelShuffle	84948	338 KB	13.54 ms
Spherical: MeshConv with transposed MeshConv	64265	273 KB	997.91 ms
Spherical: MeshConv with VertexShuffle (**SSR**)	79925	333 KB	74.35 ms

model for the 30 frames it contains. We conducted our experiments on a desktop machine with Intel(R) Core(TM) i7-8700K CPU and GeForce RTX 2080 Ti GPU.

To use our Focused Mesh in training, for each frame in a video segment, we train for all 80 Level-1 faces. As discussed, the refined Level-1 face is selected as a face located on the equator to avoid distortion near pole areas. Our low-resolution input data is in Level-7, which is roughly equivalent to a 2D equirectangular-projected frame in 720x360 resolution. Our high-resolution target data is in Level-9, which is roughly equivalent to 2880x1440 equirectangular-projected frames. The upscaling factor in our experiment set up is $\times 4$. That is, the number of output vertices is 16x the number of input vertices.

For each video in the dataset, we train the first video segment for 40 epochs. For the rest video segments, we take advantage of temporal locality of videos and use a previous video segment's model as a pre-trained model for the current segment. This allows us to pursue training efficiency by training each video segment for only 15 epochs. We set the learning rate to 0.01 and use Adam [23] as our optimizer.

Baseline models. We train models for all videos in the dataset using our proposed spherical super-resolution (SSR) model and compare its results with three baseline models: 2D super-resolution with Upsample, 2D super-resolution with PixelShuffle, and spherical super-resolution with transposed MeshConv (i.e., instead of VertexShuffle). Our baseline models are also trained from scratch.

We focus our comparison on three aspects: visual quality (i.e., the PSNR value), model size, and inference time. For 2D baseline models, we tile the frames spatially into small patches. Here, we use the patch size of 45×45 pixels for input data, and thus, 180×180 pixel for output and target data. The upscaling factor is thus $\times 4$, which is the same as our SSR model.

PSNR results. The mean PSNR values obtained by all models for each video are shown in Table 2. 2D baseline model with PixelShuffle achieves the best PSNR value because of the efficiency of 2D convolution. Compared to the spherical baseline model that uses transposed MeshConv operation, our SSR model achieves better results, which shows the improvement of our new VertexShuffle operation.

Model parameters and storage sizes. Table 3 shows the total number of parameters of all models and their corresponding storage sizes. The spherical baseline model with transposed MeshConv operations used as one of our baseline models has a smaller model size. However, its inference time is significantly longer than all other models.

Inference time. In our spherical models, we separate the original omnidirectional frame into 80 faces. In 2D baseline models, we tile the image into 128 tiles. Thus, it is not fair to simply compare the inference time for each face or tile. Instead, we use the per-frame inference time for comparison. The experiment results are shown in Table 3. 2D baseline models have the fastest inference time. Our SSR model is slower than 2D baseline models but significantly faster than the spherical baseline with transposed MeshConv operations. In addition, given that a user will only watch a sub-portion of the spherical frame in real scenarios, we do not need to perform super-resolution for all 80 faces of a frame. This indicates our SSR model can be performed in real-time to meet the 30 fps frame rate required by most videos.

5 CONCLUSION

In this paper, we proposed a spherical super-resolution model – SSR. SSR directly operates on spherical signals, which can avoid issues in applying 2D super-resolution to spherical data, such as distortion, oversampled pixels, etc. We created a memory- and bandwidth-efficient representation of the spherical mesh – the Focused Icosahedral mesh, which is more flexible than full meshes and saves a significant amount of computation resources. We also created a novel VertexShuffle operation to further improve our model. We compared our model with three other baseline models. Results show that our model achieves 31.97 dB PSNR on average for super-resolution with a scale factor of x4 while maintaining parameter efficiency.

ACKNOWLEDGMENTS

We appreciate constructive comments from anonymous referees. This work is partially supported by NSF under grants CNS-1618931 and CNS-2200042.

REFERENCES

[1] 2022. Equirectangular Projection. http://mathworld.wolfram.com/EquirectangularProjection.html.

[2] 2022. FFmpeg. http://www.ffmpeg.org/.

[3] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 1161–1170.

[4] John R Baumgardner and Paul O Frederickson. 1985. Icosahedral discretization of the two-sphere. *SIAM J. Numer. Anal.* 22, 6 (1985), 1107–1115.

[5] Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang, and Di Wu. 2020. SR360: boosting 360-degree video streaming with super-resolution. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 1–6.

[6] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2017. Convolutional networks for spherical signals. *arXiv preprint arXiv:1709.04893* (2017).

[7] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 2017. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 199–204.

[8] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 1–7.

[9] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R Das. 2020. Streaming 360-degree videos using super-resolution. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1977–1986.

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*. Springer, 184–199.

[11] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence* 38, 2 (2015), 295–307.

[12] Chao Dong, Chen Change Loy, and Xiaoou Tang. 2016. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*. Springer, 391–407.

[13] Marc Eder, Mykhailo Shvets, John Lim, and Jan-Michael Frahm. 2020. Tangent Images for Mitigating Spherical Distortion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12426–12434.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[15] Mario Graf, Christian Timmerer, and Christopher Mueller. 2017. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 261–271.

[16] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*. 394–407.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[19] Michal Irani and Shmuel Peleg. 1991. Improving resolution by image registration. *CVGIP: Graphical models and image processing* 53, 3 (1991), 231–239.

[20] Chiyu Jiang, Jingwei Huang, Karthik Kashinath, Philip Marcus, Matthias Niessner, et al. 2019. Spherical cnns on unstructured grids. *arXiv preprint arXiv:1901.02039* (2019).

[21] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1646–1654.

[22] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1637–1645.

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

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

[26] Anahita Mahzari, Afshin Taghavi Nasrabadi, Aliehsan Samiei, and Ravi Prakash. 2018. FoV-Aware Edge Caching for Adaptive 360Â° Video Streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 173–181.

[27] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 1689–1697.

[28] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 306–314.

[29] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 99–114.

[30] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1874–1883.

[31] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2018. Multi-path multi-tier 360-degree video streaming in 5G networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 162–173.

[32] Ying Tai, Jian Yang, and Xiaoming Liu. 2017. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3147–3155.

[33] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo. 2017. 360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 315–323.

[34] Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 601–605.

[35] Chao Zhang, Stephan Liwicki, William Smith, and Roberto Cipolla. 2019. Orientation-aware semantic segmentation on icosahedron spheres. In *Proceedings of the IEEE International Conference on Computer Vision*. 3533–3541.

[36] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2472–2481.

[37] Chao Zhou, Zhenhua Li, and Yao Liu. 2017. A measurement study of oculus 360 degree video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 27–37.