2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# 360° Depth Estimation from Multiple Fisheye Images with Origami Crown Representation of Icosahedron

Ren Komatsu[1], Hiromitsu Fujii[2], Yusuke Tamura[3], Atsushi Yamashita[1], and Hajime Asama[1]

***Abstract*— In this study, we present a method for all-around depth estimation from multiple omnidirectional images for indoor environments. In particular, we focus on plane-sweeping stereo as the method for depth estimation from the images. We propose a new icosahedron-based representation and ConvNets for omnidirectional images, which we name "CrownConv" because the representation resembles a crown made of origami. CrownConv can be applied to both fisheye images and equirectangular images to extract features. Furthermore, we propose icosahedron-based spherical sweeping for generating the cost volume on an icosahedron from the extracted features. The cost volume is regularized using the three-dimensional CrownConv, and the final depth is obtained by depth regression from the cost volume. Our proposed method is robust to camera alignments by using the extrinsic camera parameters; therefore, it can achieve precise depth estimation even when the camera alignment differs from that in the training dataset. We evaluate the proposed model on synthetic datasets and demonstrate its effectiveness. As our proposed method is computationally efficient, the depth is estimated from four fisheye images in less than a second using a laptop with a GPU. Therefore, it is suitable for real-world robotics applications. Our source code is available at `https://github.com/matsuren/crownconv360depth`.**

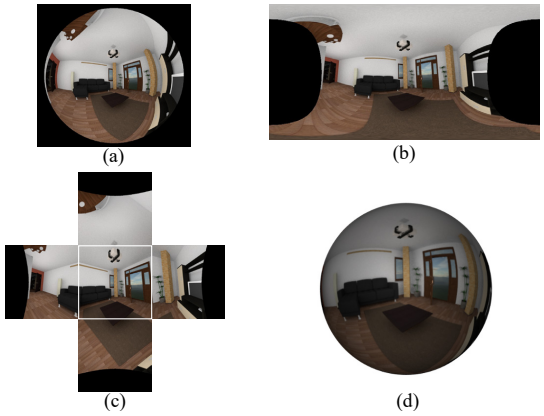


Fig. 1. Example of fisheye image representation from OmniHouse dataset [4]. (a) Original fisheye image with FoV of 220°. (b) Equirectangular image. (c) Cubemap representation. The back face is removed because all regions are beyond the camera FoV. (d) Fisheye image projected onto icosahedron at level 7. (a) and (d) appear similar; however, (d) indicates that the image is projected onto the icosahedron in 3D space.

## I. INTRODUCTION

The depth estimation of the surrounding environment of a vehicle is becoming increasingly important in the field of robotics and computer vision, as depth information is required for tasks such as autonomous navigation and object detection. The use of LiDAR is one approach to obtain depth information; however, RGB cameras are also commonly used for depth estimation owing to their low cost, light weight, and availability. In particular, fisheye cameras or omnidirectional cameras are used to estimate the depth of the surroundings owing to their wide field of view (FoV) [1]–[4].

Plane-sweeping stereo, which has been studied for numerous years [5]–[10], is one approach to depth estimation from multi-view images. In plane-sweeping stereo, multi-view images are projected onto virtual planes at several distances from the reference image plane to generate a cost volume. Thereafter, depth maps are estimated using this cost volume. Recently, convolutional neural networks (ConvNets) have been applied to plane-sweeping stereo using perspective images, with remarkable results achieved [11]–[16].

As opposed to perspective images, the direct application of ConvNets to fisheye images is not desirable as equivariance to translation is not satisfied for fisheye images owing to distortion. Fig. 1(a) presents an example of fisheye images with a FoV of 220°. As can be observed from Fig. 1(a), straight lines in the three-dimensional (3D) world are captured as curved lines in the image because of distortion. Several studies have applied ConvNets to fisheye images or equirectangular images directly, relying on the high flexibility of ConvNets to learn the distortion of the images [3], [4]. Won et al. proposed a method for estimating the all-around depth from four fisheye images [3]. They converted fisheye images into equirectangular images and applied ConvNets to extract the features directly. The cost volume was generated and refined using spherical sweeping [17], followed by semi-global matching (SGM) [18]. Later, they proposed replacing SGM with 3D ConvNets [4]. However, such approaches are not robust to changes in the camera alignment, as demonstrated in this study. This is undesirable because every time the camera alignment is changed, additional training is required on datasets with the corresponding camera alignment.

In very recent years, ConvNets that were designed specifically for images other than perspective images have been

*A part of this study is supported by the Nuclear Energy Science & Technology and Human Resource Development Project (through concentrating wisdom) from the Japan Atomic Energy Agency / Collaborative Laboratories for Advanced Decommissioning Science, and Initiative on Promotion of Supercomputing for Young or Women Researchers, Supercomputing Division, Information Technology Center, The University of Tokyo.

[1]R. Komatsu, A. Yamashita, and H. Asama are with the Department of Precision Engineering, Graduate School of Engineering, The University of Tokyo, Tokyo 113-8656, Japan (email: {komatsu, yamashita, asama}@robot.t.u-tokyo.ac.jp).

[2]H. Fujii is with the Department of Advanced Robotics, Faculty of Advanced Engineering, Chiba Institute of Technology, Narashino 275-0016, Japan (email: hiromitsu.fujii@p.chibakoudai.jp).

[3]Y. Tamura is with the Department of Robotics, Division of Mechanical Engineering, Tohoku University, Sendai 980-8579, Japan (email: y.tamura@srd.mech.tohoku.ac.jp).

studied intensively owing to the growing popularity of 360° cameras or omnidirectional cameras (for example, RICOH THETA and Insta360 ONE X). Certain studies [19], [20] have focused on ConvNets that are applied to equirectangular images, which is a common image representation technique for omnidirectional cameras. Fig. 1(b) presents an example of equirectangular images. As can be observed in Fig. 1(b), equirectangular images exhibit the characteristic that stronger distortion exists when the area is closer to the top and bottom of the image. Su et al. proposed SphConv [19], whereby different sizes of ConvNet kernels were applied to different rows to compensate for the distortion. However, the number of parameters was large because the weights were only shared in the row direction. To deal with this memory inefficiency, the authors proposed the Kernel Transformer Network [20] to share the same weights even in different rows. Although these approaches can handle distortion on equirectangular images, they are not computationally efficient because of oversampling in the top and bottom regions of the equirectangular images. Coors et al. proposed SphereNet [21], in which normal ConvNet kernels were distorted by sampling points on the tangent plane. Moreover, they proposed uniform sphere sampling to prevent oversampling for efficient computation.

Cheng et al. used cubemap representations for omnidirectional images [22]. As can be observed in Fig. 1(c), cubemap represents an omnidirectional image as multiple perspective images by projecting the image onto six cube faces; therefore, normal ConvNets can be applied to these perspective images. However, this approach suffers from discontinuity between the faces, and ambiguity in the kernel orientations for the top and bottom faces.

Another approach involves projecting omnidirectional images onto a sphere surface ($\mathcal{S}^2$) and applying ConvNets in non-Euclidean space. The application of ConvNets to manifold or graph structures has been studied in the field of geometric deep learning [23]. Cohen et al. proposed the use of ConvNets in the frequency domain using a generalized fast Fourier transform [24], whereas Esteves et al. proposed using ConvNets in the spherical harmonic domain [25]. These approaches extract features that are rotation invariant in the rotation group $\mathrm{SO}(3)$; however, they require significant memory and computational costs.

Various other studies have used icosahedrons, because a subdivided icosahedron can be used to generate nearly uniformly distributed points on $\mathcal{S}^2$ [26]. Fig. 1(d) displays a fisheye image projected onto an icosahedron at subdivision level 7. Jiang et al. proposed UGSCNN, whereby images were projected onto an icosahedron mesh and ConvNets were applied as linear combinations of differential operators on the mesh with learnable parameters [27]. However, operators on the mesh structures are computationally less efficient compared to normal two-dimensional (2D) ConvNets on images. Therefore, Liu et al., Cohen et al., and Zhang et al. proposed unfolding and distorting the icosahedron grid so that normal 2D ConvNets could be applied on the icosahedron [28]–[30]. Moreover, Zhang et al. proposed orientation-aware ConvNets for indoor scene semantic segmentation [30].

In this study, we focus on all-around depth estimation in indoor environments from multiple omnidirectional images. We use the icosahedron to apply ConvNets to omnidirectional images to deal with the distortion. Firstly, multiple omnidirectional images are projected onto the icosahedron and the image features are extracted by applying our proposed icosahedron-based ConvNets to the images on the icosahedron. Thereafter, a cost volume is generated from these features using our proposed icosahedron-based spherical sweeping and the cost volume is regularized using icosahedron-based 3D ConvNets. Finally, the depth is obtained by depth regression from the cost volume. Furthermore, we consider the orientation for the estimation as in [30], because this is beneficial to estimating the depth in indoor environments and making our proposed method robust to camera alignments. Our proposed method is computationally efficient, so that the depth can be estimated from four fisheye images in less than a second using a laptop with a GPU. Therefore, it is suitable for real-world robotics applications.

In summary, our contributions are as follows:

1) We propose a new method called IcoSweepNet for depth estimation from multiple fisheye images, which is robust to camera alignment by using the extrinsic camera parameters for the feature extraction.
2) We propose a new icosahedron-based representation and computationally efficient ConvNets for omnidirectional images, which we name CrownConv because the representation resembles a crown made of origami.
3) We propose icospherical sweeping that is icosahedron-based spherical sweeping to generate the cost volume from omnidirectional images.

## II. METHODS

### A. Overview

Our proposed method is similar to previous learning-based plane-sweeping methods. The greatest difference is that the operators are applied on 2D image planes in existing learning-based plane-sweeping methods, whereas the operators in our proposed method are all applied on the icosahedron. An overview is presented in Fig. 2. Multiple fisheye cameras facing various directions are mounted so that the surrounding environments are captured.

Firstly, as can be observed in Figs. 2(a) and (b), fisheye images are projected onto icosahedron level $l$ using the intrinsic and extrinsic camera parameters. The camera orientations are considered so that the image regions of the ceilings and floors are projected onto the north and south poles of the icosahedron, respectively, regardless of the camera alignments, as explained further in II-C.

Secondly, as can be observed in Figs. 2(b) and (c), features are extracted from images on the icosahedron by CrownConv, which is a ConvNet designed for features on an icosahedron. CrownConv and the feature extraction are explained in II-D and II-E, respectively.

Thirdly, as can be observed in Figs. 2(c) and (d), a cost volume is generated from the features by icospherical

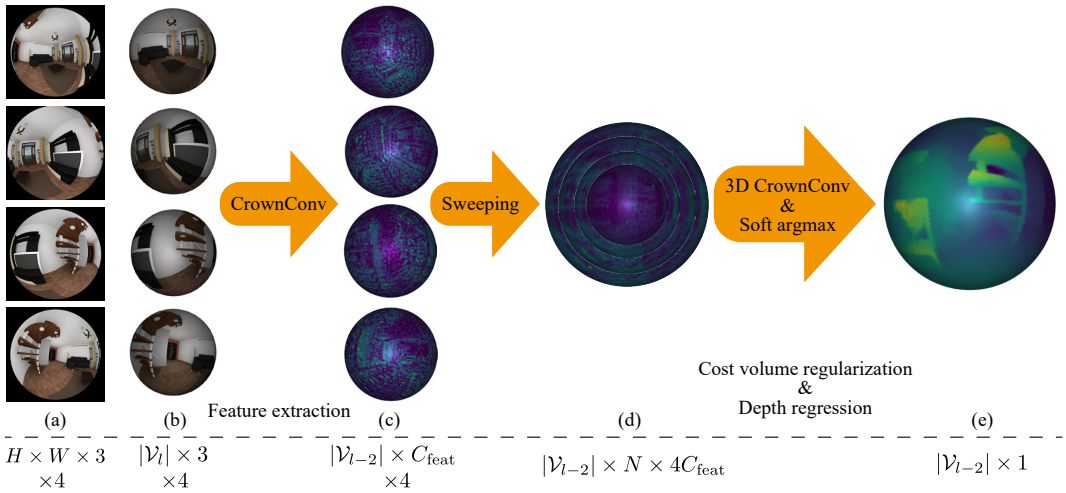


Fig. 2. Overview of proposed method. (a) Original input of fisheye images. (b) Images projected onto icosahedron level $l$. (c) Image features extracted on icosahedron level $l-2$ using CrownConv. (d) Cost volume generated using icospherical sweeping. (e) Cost volume regularized using 3D CrownConv, with depth obtained by depth regression. The row below indicates the feature shapes.

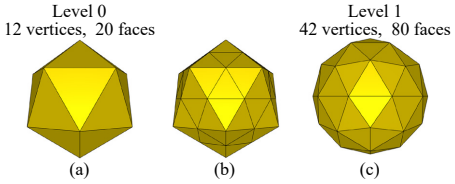


Fig. 3. Subdivision process of icosahedron. (a) Icosahedron at level 0 with 12 vertices and 20 faces. (b) Dividing triangle face into four triangle faces. (c) Icosahedron at level 1 with 42 vertices and 80 faces by normalizing distance from center to each vertex following (b).

sweeping, which is icosahedron-based spherical sweeping. Icospherical sweeping is explained in II-F.

Finally, as can be observed in Figs. 2(d) and (e), the cost volume is regularized using 3D CrownConv, and the depth is obtained by depth regression, as discussed in II-G.

Prior to elaborating on our proposed method, we briefly explain the icosahedron in the following subsection.

### B. Icosahedron

The regular icosahedron is one of the regular polyhedrons, which has 20 equilateral triangle faces and 12 vertices. The distance from the center to each vertex is the same, which is a suitable property for approximating the shape of a sphere. In this study, the distance is set to 1 and the center is set to the origin of the coordinate system; therefore, the icosahedron is suitable for approximation of the unit sphere. The resolution of the icosahedron can easily be increased by subdivision, whereby a triangle face is divided into four triangle faces. Figs. 3(a), (b), and (c) present the subdivision of an icosahedron from level 0 (regular icosahedron) to level 1. The number of vertices and faces of the icosahedron at level $l$ are $2 + 10 \cdot 4^l$ and $20 \cdot 4^l$, respectively. In this case, $\mathcal{V}_l$, the set of vertices at level $l$, is formulated as follows:

$$\mathcal{V}_l = \{\mathbf{v}_1^l, \mathbf{v}_2^l, \cdots, \mathbf{v}_{2+10\cdot 4^l}^l\}, \tag{1}$$

where $\mathbf{v}_i^l \in \mathbb{R}^3$ is a vertex of the icosahedron at level $l$ and $|\mathbf{v}_i^l| = 1$ because it is normalized.

### C. Projecting images onto icosahedron

The fisheye images are projected onto the icosahedron using the intrinsic and extrinsic camera parameters, which are estimated in advance. We first define the intrinsic and extrinsic camera parameters. The extrinsic parameter of camera $c_k$ is formulated as follows:

$$\mathbf{T}_{c_k w} = [\mathbf{R}_{c_k w}, \mathbf{t}_{c_k w}], \tag{2}$$

where $\mathbf{R}_{c_k w} \in \mathrm{SO}(3)$ and $\mathbf{t}_{c_k w} \in \mathbb{R}^3$ are the rotation matrix and translation vector, respectively. Specifically, $\mathbf{T}_{c_k w}$ transforms a point $\mathbf{p}^w$ in the world coordinates to a point $\mathbf{p}^{c_k}$ in the camera $c_k$ coordinates. Moreover, $\mathbf{T}_{w c_k}$ is defined in the same manner, which transforms $\mathbf{p}^{c_k}$ into $\mathbf{p}^w$. The intrinsic parameter of $c_k$ is represented as a projection function $\pi_{c_k} : \mathbb{R}^3 \to \Omega$, which projects $\mathbf{p}^{c_k}$ onto a point $\mathbf{u}$ in the image domain $\Omega$. We use the fisheye models proposed in [31], [32] for the projection. Our method can also be used on equirectangular images by replacing the fisheye projection models with equirectangular projection models.

We consider the direction of gravity, which is beneficial for estimating the depth in indoor environments simultaneously, which makes our proposed method robust to changes in the camera alignment. The motivation is that the image regions of the ceilings and floors will be projected onto the north and south poles of the icosahedron, respectively, regardless of the camera alignment. Therefore, a projected image on the icosahedron is formulated as follows:

$${}^{\text{ico}}\mathcal{I}_{c_k}^l = \{\mathcal{I}_{c_k}\left(\pi_{c_k}\left(\mathbf{R}_{c_k w}\mathbf{v}_i^l\right)\right) \mid \mathbf{v}_i^l \in \mathcal{V}_l\}, \tag{3}$$

where ${}^{\text{ico}}\mathcal{I}_{c_k}^l$ is the projected image of the camera $c_k$ on the icosahedron at level $l$ and $\mathcal{I}_{c_k}(\mathbf{u})$ is the pixel value of the image of the camera $c_k$ at location $\mathbf{u}$.

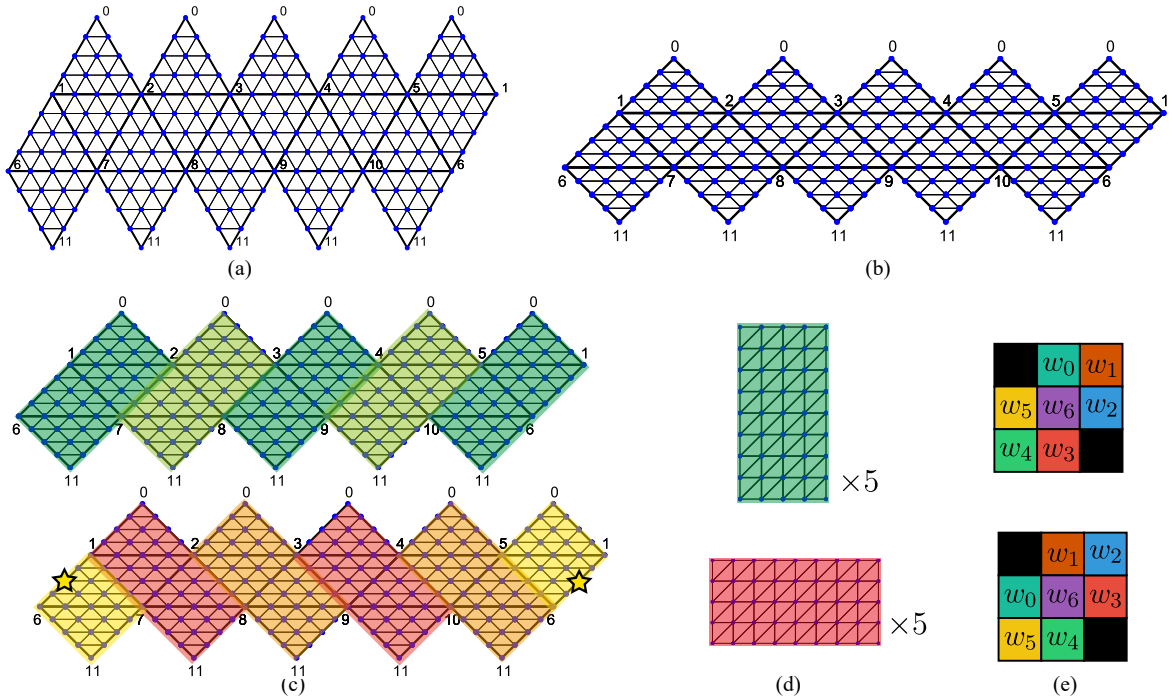


Fig. 4. Origami crown representation for omnidirectional images. (a) Grid based on unfolded icosahedron at level 2. (b) Grid distorted so that normal ConvNet can be applied. (c) Origami crown representation. Rectangles are extracted from the distorted grid in the two different directions. The stars at the ends of the grid indicate connectivity. (d) Final representation of five rectangles for two directions. (e) Weights of ConvNets shared but directions differ. The weights $w_0$ and $w_3$ are facing the longer direction. The numbers in (a), (b), and (c) represent the vertex indices of the icosahedron at level 0.

Fig. 5. Crown made of origami.

### D. CrownConv

We first describe CrownConv, which is a ConvNet designed for features on an icosahedron. As in [28]–[30], we unfold and distort the icosahedron grid so that normal 2D ConvNets can be applied to the icosahedron. Figs. 4(a) and (b) present the unfolded icosahedron grid and distorted unfolded icosahedron grid, respectively. Thereafter, we propose representing the features on the icosahedron as five rectangles in two different directions, namely upper right and lower right, as illustrated in Figs. 4(c) and (d). As the representation in Fig. 4(c) resembles a crown made of origami, we refer to it as the origami crown representation. An origami crown is presented in Fig. 5 for reference. Subsequently, the features ($|\mathcal{V}_l| \times C$) on the icosahedron at level $l$ are converted into five vertical rectangle features ($\left[2^{l+1}+1\right] \times \left[2^{l}+1\right] \times C$) and five horizontal rectangle features ($\left[2^{l}+1\right] \times \left[2^{l+1}+1\right] \times C$), as indicated in Fig. 4(d), where $C$ is the number of channels of the features. These are formulated as follows:

$$\begin{aligned} {}^{\mathrm{col}}\mathcal{G}^l, {}^{\mathrm{row}}\mathcal{G}^l &= \Pi(\mathcal{F}^l), \\ {}^{\mathrm{col}}\mathcal{G}^l &= \{{}^{\mathrm{col}}g_i^l \mid i \in 1, 2, \cdots, 5\}, \\ {}^{\mathrm{row}}\mathcal{G}^l &= \{{}^{\mathrm{row}}g_i^l \mid i \in 1, 2, \cdots, 5\}, \end{aligned} \tag{4}$$

where $\Pi(\cdot)$ is a function for the conversion and $\mathcal{F}^l$ is the features on the icosahedron at level $l$. ${}^{\mathrm{col}}\mathcal{G}^l$ and ${}^{\mathrm{row}}\mathcal{G}^l$ are the five vertical and horizontal rectangle features, respectively.

CrownConv is applied to ${}^{\mathrm{col}}\mathcal{G}^l$ and ${}^{\mathrm{row}}\mathcal{G}^l$ using normal 2D ConvNets. We use shared weights of the ConvNets but different directions for ${}^{\mathrm{col}}\mathcal{G}^l$ and ${}^{\mathrm{row}}\mathcal{G}^l$, similar to the process in [30], as illustrated in Fig. 4(e). The upper and lower sides of Fig. 4(e) represent ${}^{\mathrm{col}}W$, which are the weights for ${}^{\mathrm{col}}\mathcal{G}^l$, and ${}^{\mathrm{row}}W$, which are the weights for ${}^{\mathrm{row}}\mathcal{G}^l$, respectively. As can be observed from Fig. 4(e), the weights $w_0$ and $w_3$ face the longer directions. CrownConv is formulated as follows:

$$\begin{aligned} {}^{\mathrm{col}}\check{\mathcal{G}}^l &= \{\mathsf{conv}(\mathsf{pad}({}^{\mathrm{col}}g_i^l), {}^{\mathrm{col}}W) \mid {}^{\mathrm{col}}g_i^l \in {}^{\mathrm{col}}\mathcal{G}^l\}, \\ {}^{\mathrm{row}}\check{\mathcal{G}}^l &= \{\mathsf{conv}(\mathsf{pad}({}^{\mathrm{row}}g_i^l), {}^{\mathrm{row}}W) \mid {}^{\mathrm{row}}g_i^l \in {}^{\mathrm{row}}\mathcal{G}^l\}, \end{aligned} \tag{5}$$

where $\check{x}$ means that CrownConv is applied to a feature $x$. Moreover, $\mathsf{conv}(x, W)$ and $\mathsf{pad}(\cdot)$ represent a convolution operator of $x$ with weight $W$ and a padding operator, respectively. Instead of zero padding, we apply padding with the replicated border to alleviate artifacts on the edges. CrownConv is computationally efficient because all of the operators used in (5) are the same for the 2D images, which are highly optimized for GPU computations.

After applying CrownConv, ${}^{\mathrm{col}}\check{\mathcal{G}}^l$ and ${}^{\mathrm{row}}\check{\mathcal{G}}^l$ should be integrated into the features on the icosahedron. Therefore, the inverse function of $\Pi(\cdot)$ is formulated as follows:

$$\begin{aligned} \check{\mathcal{F}}^l &= \Pi^{-1}({}^{\mathrm{col}}\check{\mathcal{G}}^l, {}^{\mathrm{row}}\check{\mathcal{G}}^l), \\ &= \{\frac{1}{|\check{\mathcal{G}}^{\mathbf{v}_i^l}|} \sum_{g \in \check{\mathcal{G}}^{\mathbf{v}_i^l}} g \mid \mathbf{v}_i^l \in \mathcal{V}_l\}, \end{aligned} \tag{6}$$

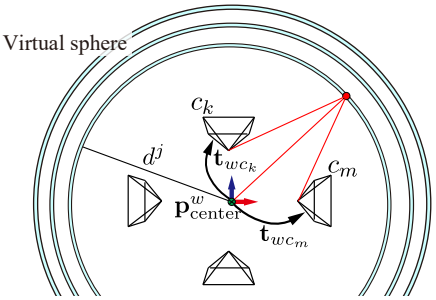


Fig. 6. Illustration of icospherical sweeping. Virtual spheres are generated at several distances $d^j$ from the center of the camera rigs and the extracted features are projected onto the spheres to calculate the cost.

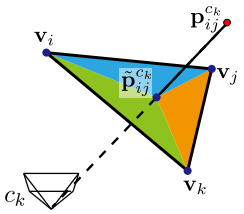


Fig. 7. Triangle interpolation, where $\mathbf{p}^{c_k}_{ij}$ is projected onto the icosahedron of camera $c_k$ and the final value of the projected point $\tilde{\mathbf{p}}^{c_k}_{ij}$ is calculated by triangle interpolation of the neighboring vertex values.

where $\check{\mathcal{F}}^l$ represents the integrated features on the icosahedron and $\Pi^{-1}(\cdot)$ is the inverse function of $\Pi(\cdot)$. Furthermore, $\check{\mathcal{G}}\mathbf{v}^l_i$ is a set of values corresponding to $\mathbf{v}^l_i$ in ${}^{\text{col}}\check{\mathcal{G}}^l$ and ${}^{\text{row}}\check{\mathcal{G}}^l$. We explain (6) by means of an example: all 10 of the rectangle features have a value for $\mathbf{v}^l_0$, which is an index numbering zero in Fig. 4(c). Then, the final value on the icosahedron is the mean of the values in the 10 rectangle features. This integration also enables the exchange of information between ${}^{\text{col}}\check{\mathcal{G}}^l$ and ${}^{\text{row}}\check{\mathcal{G}}^l$.

The concept of CrownConv is similar to HexConv [30]; therefore, the differences between CrownConv and HexConv are described here. For HexConv, five rectangles were extracted from the distorted unfolded icosahedron grid in only one direction, not two directions. Thereafter, HexConv copied features on the edges between five rectangles in every layer before applying normal 2D ConvNets, which helped HexConv extract global contextual information even if an object appeared across the line between vertices 2 and 7 in Fig. 4(c). However, copying features in every layer prevented efficient computation. Meanwhile, CrownConv can extract global contextual information without copying features as the full shape of the object appears in either the five horizontal rectangles or the five vertical rectangles, which makes CrownConv more computationally efficient.

### *E. Feature extraction*

Our feature extraction module is based on ResNet [33]. Instead of normal 2D ConvNets, we use CrownConv to extract features from the images on the icosahedron. We place $\Pi^{-1}(\cdot)$ and $\Pi(\cdot)$ several times in the middle of the feature extraction module to exchange information between the rectangle features ${}^{\text{col}}\check{\mathcal{G}}^l$ and ${}^{\text{row}}\check{\mathcal{G}}^l$, and we place $\Pi^{-1}(\cdot)$ at the end of the module to obtain the extracted features on the icosahedron. During the extraction, the features are downsampled twice by CrownConv with stride 2. Consequently, $\mathcal{F}^{l-2}_{c_k}$ is obtained from ${}^{\text{ico}}\mathcal{I}^l_{c_k}$. The shape of $\mathcal{F}^{l-2}_{c_k}$ is $|\mathcal{V}_{l-2}| \times C_{\text{feat}}$, where $C_{\text{feat}}$ is the number of channels of the extracted features. For the detailed network architecture, see our GitHub page.

### *F. Icospherical sweeping*

The extracted features $\mathcal{F}^{l-2}_{c_k}$ are warped by icospherical sweeping and a cost volume is generated on the icosahedron by concatenating the warped features. In icospherical sweeping, similar to spherical sweeping [17], the features are projected onto the virtual spheres that are generated at several distances $d^j$ from the center of the camera rigs, as illustrated in Fig. 6. The cost volume is formulated as follows:

$$\begin{aligned} V(i,j) &= \mathsf{concat}\left(\left\{\mathsf{sample}\left(\mathcal{F}^{l-2}_{c_k}, \mathbf{p}^{c_k}_{ij}\right) \mid c_k \in \mathcal{C}\right\}\right), \\ \mathbf{p}^{c_k}_{ij} &= \mathbf{p}^w_{\text{center}} + d^j \cdot \mathbf{v}^{l-2}_i - \mathbf{t}_{wc_k}, \end{aligned} \tag{7}$$

where $\mathsf{concat}(\cdot)$, $\mathcal{C}$, and $\mathbf{p}^w_{\text{center}} \in \mathbb{R}^3$ are the concatenation operator, the set of all cameras, and the center of the camera rigs in the world coordinates, respectively. $\mathbf{p}^{c_k}_{ij} \in \mathbb{R}^3$ is a point projected onto the virtual sphere in the camera $c_k$ coordinates. It should be noted that only the translation $\mathbf{t}_{wc_k}$ is required for the coordinate transformation, as the rotation has already been applied in (3).

Moreover, $\mathsf{sample}\left(\mathcal{F}^{l-2}_{c_k}, \mathbf{p}^{c_k}_{ij}\right)$ is the sampling operation by triangle interpolation, as illustrated in Fig. 7. As can be observed in Fig. 7, $\mathbf{p}^{c_k}_{ij}$ is projected onto the icosahedron of camera $c_k$ and the final value is calculated as follows:

$$\begin{aligned} \mathsf{sample}\left(\mathcal{F}^{l-2}_{c_k}, \mathbf{p}^{c_k}_{ij}\right) &= \mathcal{F}^{l-2}_{c_k}(\mathbf{v}_i)\frac{\mathcal{A}(\tilde{\mathbf{p}}^{c_k}_{ij}, \mathbf{v}_j, \mathbf{v}_k)}{\mathcal{A}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)} \\ &+ \mathcal{F}^{l-2}_{c_k}(\mathbf{v}_j)\frac{\mathcal{A}(\mathbf{v}_i, \tilde{\mathbf{p}}^{c_k}_{ij}, \mathbf{v}_k)}{\mathcal{A}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)} \\ &+ \mathcal{F}^{l-2}_{c_k}(\mathbf{v}_k)\frac{\mathcal{A}(\mathbf{v}_i, \mathbf{v}_j, \tilde{\mathbf{p}}^{c_k}_{ij})}{\mathcal{A}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)}, \\ \tilde{\mathbf{p}}^{c_k}_{ij} &= \frac{\mathbf{p}^{c_k}_{ij}}{|\mathbf{p}^{c_k}_{ij}|}, \end{aligned} \tag{8}$$

where $\mathcal{A}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)$ represents the area of the triangle of vertices $\mathbf{v}_i$, $\mathbf{v}_j$, and $\mathbf{v}_k$, and $\mathcal{F}^{l-2}_{c_k}(\mathbf{v}_i)$ represents the value of $\mathcal{F}^{l-2}_{c_k}$ at vertex $\mathbf{v}_i$. These operations depend on only the camera poses, depth from the center, and level of the icosahedron; therefore, the indices and factors of the interpolation are cached for efficient computation.

The distance $d^j$ from the center of the camera rigs is formulated as follows:

$$\frac{1}{d^j} = \frac{j-1}{N-1} \cdot \frac{1}{d_{\min}} + \epsilon, \quad j \in \{1, 2, \ldots, N\}, \tag{9}$$

where $N$, $d_{\min}$, and $\epsilon$ are the number of virtual spheres, the minimum distance of the virtual sphere, and a small positive number to avoid division by zero, respectively.

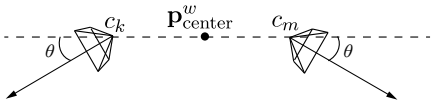


Fig. 8. Camera alignment of rotated OmniHouse dataset. The cameras are facing downwards. Only two cameras are presented for simplicity. The arrows represent the camera-facing direction.

### G. Depth regression

The cost volume $V$ is regularized using the 3D CrownConv to obtain $V^*$. The 3D CrownConv is defined in the same manner as CrownConv by adding another dimension.

The final depth is obtained as the inverse depth index on the icosahedron at level $l-2$ by means of depth regression from $V^*$. We use soft argmax [11] as the depth regression method to enable the inverse depth index to have a floating-point number. The depth is calculated from the regularized cost volume $V^*$, as follows:

$$\hat{D}\left(i\right) = \sum_{j=1}^{N} j\sigma\left(V^*\left(i,\cdot\right)\right)_j, \quad i \in \{1,2,\ldots,|\mathcal{V}_{l-2}|\}, \quad (10)$$

where $\sigma(\cdot)$ is a softmax operation.

### H. Loss function

The loss function formulated below is used to minimize the errors between the prediction $\hat{D}$ and ground truth inverse depth index $D_{\text{gt}}$:

$$L(\hat{D}, D_{\text{gt}}) = \frac{1}{|\mathcal{V}_{l-2}|} \sum_{i \in |\mathcal{V}_{l-2}|} f_\delta\left(\hat{D}\left(i\right), D_{\text{gt}}\left(i\right)\right), \quad (11)$$

where $f_\delta$ is the Huber loss [34] with $\delta = 1$ and $D_{\text{gt}}$ is generated as follows:

$$D_{\text{gt}}\left(i\right) = 1 + \frac{d_{\min}}{d_{\text{gt}}\left(i\right)} \cdot \left(N-1\right), \quad (12)$$

where $d_{\text{gt}}$ is the ground truth depth map on the icosahedron, which is calculated by (3).

## III. EXPERIMENTS

### A. Datasets

We used synthetic datasets, namely OmniThings and OmniHouse [4], for the training and evaluation. These datasets contain images from four fisheye cameras, depths from the center of the camera rig, and the extrinsic and intrinsic camera parameters. It should be noted that OmniThings and OmniHouse contain unique camera alignments; therefore, the extrinsic and intrinsic camera parameters are the same for both OmniThings and OmniHouse.

As the test sets of OmniThings and OmniHouse are not publicity available, we used the training sets in this experiment. We used 7,216 sets from OmniThings for training, 2,000 sets from OmniThings for validation, and 2,048 sets from OmniHouse for evaluation. The model with the lowest validation error during training was selected as the best model and evaluated.

### B. Evaluation for robustness to camera alignment

To evaluate the robustness to changes in the camera alignment, we used OmniHouse to create another dataset named as rotated OmniHouse, in which the camera-facing directions were changed. In the original OmniHouse, the four cameras were facing the front, right, back, and left. Meanwhile, all cameras were facing downwards in rotated OmniHouse, as indicated in Fig. 8. We evaluated the proposed method on rotated OmniHouse at angles $\theta$ of 0° (the same as OmniHouse), 15°, 30°, and 45°. In the evaluation using rotated OmniHouse, certain regions of the estimated depth exhibited large errors owing to insufficient overlapping for stereo estimation. Therefore, those regions were excluded from the evaluation.

### C. Training details

We used the PyTorch framework to implement the proposed network. The training was conducted in an end-to-end manner on four NVIDIA Tesla P100 GPUs with 16 GB of memory. We trained the network for 54,000 iterations with a batch size of 4. During training, Adam [35] was used as the optimizer. The learning rate was set to 1e-3 for the first 36,000 iterations and 1e-4 for the remainder. The fisheye images were projected onto the icosahedron at level 7; thus, the depth was estimated on the icosahedron at level 5. Moreover, $d_{\min}$ was set to 0.55 m, whereas both $C_{\text{feat}}$ and $N$ were set to 32.

### D. Metrics

The depth estimation results were evaluated using the same metrics as those in [4]. The error was measured as follows:

$$E(i) = 100 \times \frac{|\hat{D}(i) - D_{\text{gt}}(i)|}{N} \quad (13)$$

and the mean absolute error (MAE), root-mean-square error (RMS), and ratio (%) of errors larger than $n$ ($>n$) were used as the evaluation metrics.

## IV. RESULTS

### A. Evaluation on rotated OmniHouse

We compared our proposed method, referred to as IcoSweepNet, to OmniMVS [4]. We implemented OmniMVS by ourselves[1] because the official implementation of OmniMVS is not publicity available at this time. We set the disparity number for OmniMVS as 48.

The evaluation results using rotated OmniHouse are presented in Table I. As can be observed from Table I, OmniMVS performed effectively when the camera alignment did not change substantially from the training. However, the performance of OmniMVS was degraded when the camera alignments were changed drastically from the training. Meanwhile, IcoSweepNet demonstrated stable performances with the different camera alignments.

Examples of the estimated depth maps for rotated OmniHouse are presented in Fig. 9. It can be observed from

[1] https://github.com/matsuren/omnimvs_pytorch

TABLE I

EVALUATION RESULTS USING ROTATED OMNIHOUSE AT DIFFERENT ANGLES $\theta$. THE BEST SCORES ARE INDICATED IN BOLD.

| Angle $\theta$ | Model | Error (smaller is better) >1 | >3 | >5 | MAE | RMS |
|---|---|---|---|---|---|---|
| 0° | IcoSweepNet | 28.69 | 9.13 | 5.55 | 1.48 | 3.36 |
| | OmniMVS | **26.14** | **5.33** | **2.69** | **1.05** | **2.12** |
| 15° | IcoSweepNet | **27.46** | 8.39 | 4.93 | 1.34 | 3.05 |
| | OmniMVS | 30.71 | **6.78** | **3.45** | **1.24** | **2.67** |
| 30° | IcoSweepNet | **26.86** | **7.76** | **4.48** | **1.29** | **2.99** |
| | OmniMVS | 40.72 | 12.07 | 6.56 | 1.80 | 3.77 |
| 45° | IcoSweepNet | **26.30** | **6.51** | **3.60** | **1.15** | **2.53** |
| | OmniMVS | 50.19 | 19.73 | 11.95 | 2.62 | 5.17 |

Fig. 9 that the depth maps estimated by OmniMVS contained more artifacts with more rotated datasets, whereas our model succeeded in estimating the depth maps despite the camera alignments being changed drastically.

### *B. Real-world experiment*

Another experiment was conducted to demonstrate the performance in real world. We trained IcoSweepNet with $N$=16, 32 on a combination of OmniThings and OmniHouse datasets, and data augmentation with color jitting and random shift was applied to the fisheye images to mitigate the difference between synthetic data and real-world data. Four fisheye cameras with FoV of 185° facing downward were mounted on corners of a UGV as illustrated in Fig. 10(a), and the depth was estimated from the four fisheye images using a laptop with NVIDIA GeForce RTX 2080 Max-Q.

IcoSweepNet estimated the depth from four fisheye images in 0.73 s ($N = 32$) and 0.42 s ($N = 16$) including capturing fisheye images, projecting the images on icosahedron, and estimating the depth. Therefore, our proposed method is applicable to robotics applications if the agility of the UGV is not very high. An example of the estimated depth map is presented in Fig. 10(b). As can be observed in Fig. 10(b), the depths of the objects were estimated correctly, whereas the floor regions seemed to have wrong estimations. We believe it was because of reflections on the floor, which does not exist in synthetic datasets. Therefore, the training on real-world data may be suggested to improve the performance in real world. The upper regions of Fig. 10(b) exhibited large errors owing to insufficient overlapping for stereo estimation.

## V. CONCLUSIONS

We presented a method for estimating the all-round depth estimation from multiple omnidirectional images. We proposed a new origami crown representation of the icosahedron for omnidirectional images named as CrownConv, which is a ConvNet designed for the icosahedron. Furthermore, icospherical sweeping was proposed for plane-sweeping stereo using omnidirectional images. IcoSweepNet was demonstrated as robust to the camera alignments by using the extrinsic camera parameters; therefore, the proposed method performed effectively even when the camera alignment was changed drastically from the training. Finally, CrownConv is computationally efficient, and thus, our proposed method could estimate the depth from four fisheye images in less than a second using a laptop with a GPU. Therefore, it is suitable for real-world robotics applications.

Although only fisheye images were used in this study, the proposed method should be effective for equirectangular images without additional training, because all of the operators were applied on an icosahedron instead of 2D image planes.

In future work, improvements of the performance in real-world data and the inference in real-time are expected to make our model more applicable to robotics applications.

## ACKNOWLEDGMENT

The authors would like to thank the members of the Intelligent Construction Systems Laboratory, The University of Tokyo for their useful suggestions, especially Mr. Shingo Yamamoto and Mr. Takumi Chiba from Fujita Corporation and Dr. Kazuhiro Chayama from KOKANKYO Engineering Corporation.

## REFERENCES

[1] L. Heng, B. Choi, Z. Cui, M. Geppert, S. Hu, B. Kuan, P. Liu, R. Nguyen, Y. C. Yeo, A. Geiger, G. H. Lee, M. Pollefeys, and T. Sattler, "Project AutoVision: Localization and 3D Scene Perception for an Autonomous Vehicle with a Multi-Camera System", *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 4695–4702, 2019.

[2] Z. Cui, L. Heng, Y. C. Yeo, A. Geiger, M. Pollefeys, and T. Sattler, "Real-Time Dense Mapping for Self-Driving Vehicles using Fisheye Cameras", *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 6087–6093, 2019.

[3] C. Won, J. Ryu, and J. Lim, "SweepNet: Wide-baseline Omnidirectional Depth Estimation", *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 6073–6079, 2019.

[4] C. Won, J. Ryu, and J. Lim, "OmniMVS: End-to-End Learning for Omnidirectional Stereo Matching", *Proceedings of the 2019 IEEE International Conference on Computer Vision*, pp. 8987–8996, 2019.

[5] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity", *Science*, vol. 194, no. 4262, pp. 283–287, 1976.

[6] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections", *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[7] M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, 1993.

[8] R. T. Collins, "A Space-Sweep Approach to True Multi-Image Matching", *Proceedings of the 1996 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 358–363, 1996.

[9] D. Gallup, J. M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions", *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[10] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys, "Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo", *Proceedings of the 2014 2nd International Conference on 3D Vision*, pp. 57–64, 2014.

[11] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-End Learning of Geometry and Context for Deep Stereo Regression", *Proceedings of the 2017 IEEE International Conference on Computer Vision*, 2017, pp. 66–75.

[12] J.-R. Chang and Y.-S. Chen, "Pyramid Stereo Matching Network", *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[13] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "MVSNet: Depth Inference for Unstructured Multi-view Stereo", *Proceedings of the 15th European Conference on Computer Vision*, 2018, pp. 767–783.

[14] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon, "DPSNet: End-to-end Deep Plane Sweep Stereo", *Proceedings of the 7th International Conference on Learning Representations*, 2019.

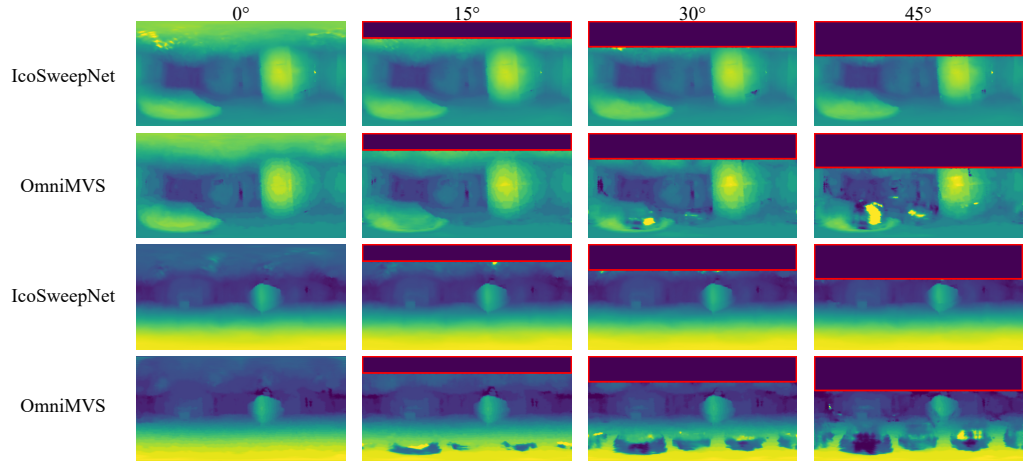


Fig. 9. Examples of estimated depth maps. The depth estimations on the icosahedron are converted into equirectangular depth maps by linear interpolation for visualization. The model names are indicated on the left. The four columns from left to right correspond to the results on rotated OmniHouse at angles of 0°, 15°, 30°, and 45°. The red rectangle in each figure indicates that the region was masked out and excluded from the evaluation owing to insufficient overlapping for stereo estimation.

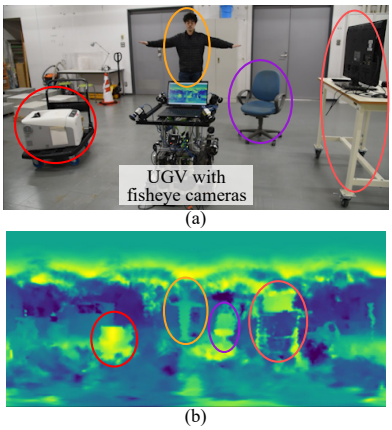


Fig. 10. Example of real-world experiment. (a) Experimental scene. Four fisheye cameras were mounted on corners of a UGV. (b) Depth estimation visualized by equirectangular projection. The corresponding objects were circled with the same color.


[15] P. H. Huang, K. Matzen, J. Kopf, N.Ahuja, and J. B. Huang, "DeepMVS: Learning Multi-view Stereopsis", *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2821–2830, 2018.

[16] R. Komatsu, H. Fujii, Y. Tamura, A. Yamashita, and H. Asama, "Octave Deep Plane-Sweeping Network: Reducing Spatial Redundancy for Learning-Based Plane-Sweeping Stereo", *IEEE Access*, vol. 7, pp. 150306–150317, 2019.

[17] S. Im, H. Ha, F. Rameau, H. G. Jeon, G. Choe, and I. S. Kweon, "All-around Depth from Small Motion with A Spherical Panoramic Camera", *Proceedings of the 14th European Conference on Computer Vision*, pp. 156–172, 2016.

[18] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[19] Y. C. Su, and K. Grauman, "Learning Spherical Convolution for Fast Features from 360° Imagery", *Advances in Neural Information Processing Systems*, pp. 529–539, 2017.

[20] Y. C. Su, and K. Grauman, "Kernel Transformer Networks for Compact Spherical Convolution", *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9442–9451, 2019.

[21] B. Coors, A. Paul Condurache, and A. Geiger, "SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images", *Proceedings of the 15th European Conference on Computer Vision*, pp. 518–533, 2018.

[22] H. T. Cheng, C. H. Chao, J. D. Dong, H. K. Wen, T. L. Liu, and M. Sun, "Cube Padding for Weakly-Supervised Saliency Prediction in 360° videos", *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1420–1429, 2018.

[23] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data", *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[24] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs", *Proceedings of the 6th International Conference on Learning Representations*, pp. 203–209, 2018.

[25] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) Equivariant Representations with Spherical CNNs", *Proceedings of the 15th European Conference on Computer Vision*, pp. 52–68, 2018.

[26] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon, "SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360 degree Images", *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9181–9189, 2019.

[27] C. M. Jiang, J. Huang, K. Kashinath, Prabhat, P. Marcus, and M. Niessner, "Spherical CNNs on Unstructured Grids", *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[28] M. Liu, F. Yao, C. Choi, A. Sinha, and K. Ramani, "Deep Learning 3D Shapes Using Alt-az Anisotropic 2-Sphere Convolution", *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[29] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge Equivariant Convolutional Networks and the Icosahedral CNN", *Proceedings of the 36th International Conference on Machine Learning*, pp. 1321–1330, 2019.

[30] C. Zhang, S. Liwicki, W. Smith, and R. Cipolla, "Orientation-aware semantic segmentation on icosahedron spheres", *Proceedings of the 2019 IEEE International Conference on Computer Vision*, pp. 3533–3541, 2019.

[31] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras", *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5695–5701, 2006.

[32] S. Urban, J. Leitloff, and S. Hinz, "Improved Wide-Angle, Fisheye and Omnidirectional Camera Calibration", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 108 ,pp. 72–79, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[34] P. J. Huber, "Robust Estimation of a Location Parameter," *Annals of Statistics*, vol. 53, no. 1, pp. 73–101, 1964.

[35] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", 2014, `arXiv:1412.6980 [cs.LG]`.