Article:
Zhang, Chao, Liwicki, Stephan, He, Sen et al. (2 more authors) (2023) HexNet: An Orientation-aware Deep Learning Framework for Omni-directional Input. IEEE Transactions on Pattern Analysis and Machine Intelligence. pp. 14665-14681. ISSN: 0162-8828

https://doi.org/10.1109/TPAMI.2023.3307152

HexNet: An Orientation-aware Deep Learning Framework for Omni-directional Input

Chao Zhang*, Stephan Liwicki*, Sen He, William Smith and Roberto Cipolla, *Senior Member, IEEE*

Abstract—While omni-directional sensors provide holistic representations typical deep learning frameworks reduce the benefits by introducing distortions and discontinuities as spherical data is supplied as planar input. On the other hand, recent spherical convolutional neural networks (CNNs) often require significant memory and parameters, thus enabling execution only at very low resolutions and shallow architectures. We propose HexNet, an orientation-aware deep learning framework for spherical signals, that allows for fast computation as we exploit standard planar network operations on an efficiently arranged projection of the sphere. Furthermore, we introduce a graph-based version for partial spheres, allowing us to compete at high-resolution with planar CNNs using residual network architectures. Our kernels operate on the tangent of the sphere and thus standard feature weights, pretrained on perspective data, can be transferred, enabling spherical pretraining on ImageNet. As our design is free of distortions and discontinuity, our orientation-aware CNN becomes a new state of the art for semantic segmentation on the recent 2D3DS dataset, and the omni-directional version of SYNTHIA introduced in this work. Moreover, we experimentally show the benefit of our spherical representation over standard images on the Cityscapes dataset by reducing distortion effects of planar CNNs. We implement object detection for the spherical domain. Rotation invariant classification and segmentation tasks are additionally presented for comparison to prior art.

Index Terms—Spherical Deep Learning, Omni-directional Cameras, Semantic Segmentation, Object Detection

1 Introduction

Deep convolutional neural networks (CNNs) have pushed the performance on a wide array of high-level computer vision tasks, including image classification, object detection and semantic segmentation. However, most frameworks focus on perspective images with small field of view (*e.g.* [1], [2], [3], [4]). In our work, we focus on omni-directional input that provides a holistic understanding of the surrounding scene, which is especially important for autonomous driving systems and robotics. Further, with the rising availability of omni-directional cameras and the increasing number of datasets with omni-directional signals, CNN-based processing with spherical input is very relevant for modern applications.

While spherical input could be represented as planar equirectangular images where standard CNNs are directly applied, such choice is limiting due to latitude dependent distortions and discontinuities at boundaries. Instead, in [5] a perspective network is distilled to work on equirectangular input. The main drawback is that weight sharing is only enabled along longitudes. Therefore, the model requires more parameters than a perspective one. SphereNet [6] projects equirectangular input onto a latitude-longitude grid. A constant grid kernel is convolved with each vertex on the sphere by sampling on the tangent plane. However, it is not straightforward to implement pooling and up-sampling for dense prediction tasks.

Alternatively, the input could be seen as graph on 3D shape manifolds. One of the challenges in applying CNNs on such non-euclidean surfaces is how to define a natural convolution operator. Works like [7], [8], [9] have focused on networks for manifolds or graphs. However, unlike general 3D shapes, omni-directional images can be oriented if north and south poles are well defined. Therefore, the lack for shift-invariance on surfaces or graphs could be overcome with an orientation-aware representation.

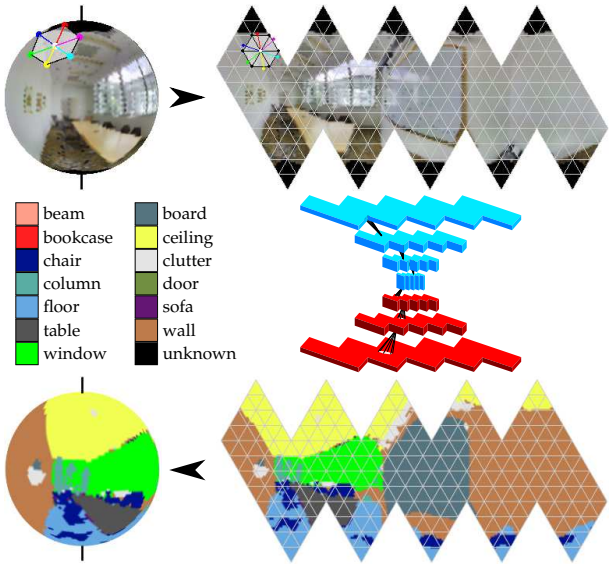


Fig. 1. Given spherical input, we convert it to an unfolded icosahedron mesh. Hexagonal filters are then applied under consideration of north alignment, as we efficiently interpolate vertices. Our approach is suited to most classical CNN architectures, *e.g.* U-Net [2]. Since we work with spherical data, final segmentation results provide a holistic labeling of the environment.

Most recently, several works proposed to use an icosahedron mesh as the underlying spherical data representa-

Manuscript received February ??, 2022.
**main contributors.*

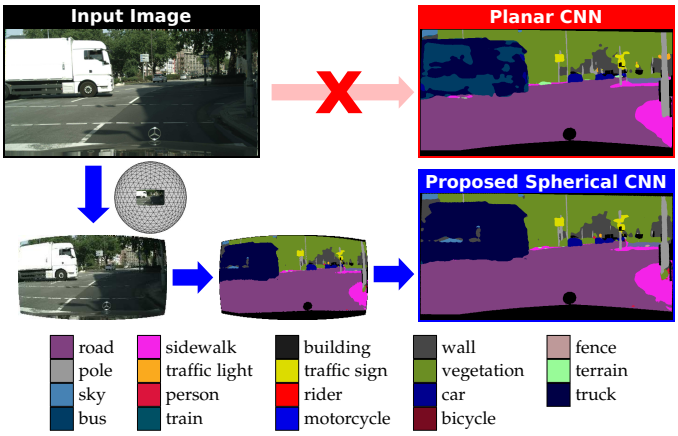


Fig. 2. Image distortion is challenging when a planar image representation is used. After formulating our spherical CNN as graph-based CNN for partial input, we propose an in-place substitution to work on the less distorted sphere manifold. Note, since geometric distortion is reduced, the 'truck' is correctly classified.

tion [10], [11], [12], [13]. The base icosahedron is the most regular polyhedron, consisting of 12 vertices and 20 faces. It also provides a simple way of resolution increase *via* subdivision. In [10], UGSCNN used the linear combinations of differential operators weighted by learnable parameters. Since the operators are precomputed, the number of parameters is reduced to 4 per kernel. The main issue of this approach, also confirmed in our experiments, is that a large amount of memory is required for the mesh convolution, especially at high-resolution input/output quality. Similar to our method in the use of icosahedron, [12] proposed a gauge equivariant CNN. Here, filter weights are shared across multiple orientations. Note however, while rotation equivariance and invariance is essential in applications such as 3D shape classification and climate pattern prediction, it might be undesired for semantic segmentation and object detection, which we consider here. On the contrary, we argue that the orientation information of cameras attached to vehicles or drones is an important cue and should be exploited. Hence, we propose and investigate a novel framework for the application of CNNs to omni-directional input, targeting semantic segmentation and object detection. We take advantage of both, the icosahedron representation for efficiency and orientation information to improve accuracy in orientation-aware tasks (Figure 1). Our hypothesis is that aligning all learnable filters to the north pole is essential for omnidirectional semantic segmentation. We also argue that high-resolution meshes are needed for detailed segmentation. Due to memory restrictions, CNN operations need to be implemented efficiently to reach such high resolution.

Further, apart from omni-directional input we also address CNN computation quality on standard images (Figure 2). Motivated by our observation of geometrical distortions on the image plane, we hypothesize results could be improved by projecting standard high-resolution images onto the manifold of a sphere using known camera intrinsics and then applying spherical CNN. Note however, while most planar CNNs work on high-resolution input and very deep architectures, existing spherical CNNs are not suitable for such settings. In particular, partial input is often not supported on spheres [10], [11], [12], while in practical scenarios (*e.g.* the driving environments of Cityscapes [14]) the active view covers less than 3% of the sphere's manifold. Processing the invalid region is computational inefficient and costly in memory. Thus, memory issues need to be addressed to enable a fair comparison between results using planar CNNs and spherical CNNs by exploiting partial spheres.

1.1 Contribution

In order to enable deep learning on spherical data without introducing distortions, we first map the spherical data to an icosahedron mesh, which we unfold along the equator, similarly to cube maps [15], [16] and [12], [17]. In the icosahedron, vertices have at most 6 neighbors. Therefore, we propose to use a hexagonal filter that is applied to each vertex's neighborhood. After simple manipulation of the unfolded mesh, standard planar CNN operations compute our hexagonal convolutions, pooling and up-sampling layers. We validate our approach on semantic segmentation and object detection tasks, as we use the omni-directional 2D3DS dataset [18] and additionally prepare our Omni-SYNTHIA dataset, which is produced from SYNTHIA data [19]. Qualitative as well as quantitative results demonstrate that our method outperforms previous state-of-the-art approaches in both scenarios. Moreover, we emphasize, since our filters are similar to standard 3×3 kernels applied to the tangent of the sphere, weight transfer from pretrained perspective CNNs is possible. Performance on spherical MNIST classification [20] and climate pattern segmentation [21] is also shown in comparison with previous methods in literature.

Building on our spherical CNN, we then reformulate our framework as a graph-based network, which facilitates selective computation on masked spherical data. Our implementation improves memory cost and running times, enabling deep spherical learning at much higher resolution than typically possible on spheres when partial input is used. In our evaluation we apply our spherical representation and consistently achieve performance gains on popular off-the-shelf architectures [2], [22] and common datasets [14], [19]. Finally, we introduce spherical pre-training with ImageNet [23], and further improve accuracy to a competitive level. In summary, our contributions are:

1) We propose a memory efficient icosahedron-based CNN framework for spherical data.
2) We introduce fast interpolation for orientation-aware filter convolutions on the sphere.
3) We reformulate our spherical CNN as graph-based CNN for arbitrarily masked input data, for efficient computation on partial spheres.
4) We present weight transfer from kernels learned through planar CNNs on perspective data.
5) We implement spherical pretraining from planar datasets such as ImageNet [23] for ResNet-50 [24].
6) We evaluate our method on both non-orientation-aware and orientation-aware, public datasets.

Our earlier results are presented in [25] and [26]. We extend the evaluation and introduce further ablation studies throughout. Furthermore, we enable the use of HexNet

for object detection on spheres, leveraging our introduced spherical bounding cycles in this work. Thus we show general applicability of HexNet to different tasks.

In the following, we discuss related work in Section 2. Our motiviation for our spherical CNN framework is presented in Section 3. Our HexNet framework and its partial input setup is introduced in Section 4 and Section 5 respectively. Pretraining and weight transfer from planar datasets is discussed in Section 6 and Section 7 defines our bounding cycles for the object detection task. Section 8 presents our evaluation on 3 tasks across 5 datasets, and Section 9 concludes this work. Additional details on experiments are given in our appendices.

2 Spherical Deep Learning

Spherical CNNs are gaining popularity with the increasing availability of omnidirectional sensor inputs in computer vision such as LiDAR, 360°-cameras and other domains. We summarize the most prominent methods here.

CNNs on Equirectangular Images: Although classical CNNs are not designed for omnidirectional data, they could still be used for spherical input if the data are converted to equirectangular form. Conversion from spherical coordinates to equirectangular images is a linear one-to-one mapping, but spherical inputs are distorted drastically especially in polar regions. Another artifact is that north and south poles are stretched to lines. Lai *et al.* [27] applied this method in the application of converting panoramic video to normal perspective. Another method along this line is to project spherical data onto multiple faces of convex polygons, such as a cube. In [15], omnidirectional images are mapped to 6 faces of a cube, and then trained with normal CNNs. However, distortions still exist and discontinuities between faces have to be carefully handled.

Spherical CNNs: In order to generalize convolution from planar images to spherical signals, the most natural idea is to replace shifts of the plane by rotations of the sphere. Cohen *et al.* [20] proposed a spherical CNN which is invariant in the $SO(3)$ group. Esteves *et al.* [28] used spherical harmonic basis to achieve similar results. Zhou *et al.* [29] proposed to extend normal CNNs to extract rotation-dependent features by including an additional orientation channel.

CNNs with Deformable Kernels: Some works [30], [31] consider adapting the sampling locations of convolutional kernels. Dai *et al.* [30] proposed to learn the deformable convolution which samples the input features through learned offsets. An Active Convolutional Unit is introduced in [31] to provide more freedom to a conventional convolution by using position parameters. These methods requires additional model parameters and training steps to learn the sampling locations.

CNNs with Grid Kernels: Another line of works aim to adapt the regular grid kernel to work on omnidirectional images. Su and Grauman [5] proposed to process equirectangular images as perspective ones by adapting the weights according to the elevation angles. Weight sharing is only enabled along longitudes. To reduce the computational cost and degradation in accuracy, a Kernel Transformer Network [32] is applied to transfer convolution kernels from perspective images to equirectangular inputs. Coors *et al.* [6] presented SphereNet to minimize the distortions introduced by applying grid kernels on equirectangular images. Here, a kernel of fixed shape is used to sample on the tangent plane according to the location on the sphere. Wrapping the kernel around the sphere avoids cuts and discontinuities.

CNNs with Reparameterized Kernels: For the efficiency of CNNs, several works are proposed to use parameterized convolution kernels. Boscani *et al.* [9] introduced oriented anisotropic diffusion kernels to estimate dense shape correspondence. Cohen and Welling [33] employed a linear combination of filters to achieve equivariant convolution filters. In [34], 3D steerable CNNs using linear combination of filter banks are developed. Recently, Jiang *et al.* [10] utilized parameterized differential operators as spherical convolution for unstructured grid data. Here, a convolution operation is a linear combination of four differential operators with learnable weights. However, these methods are limited to the chosen kernel types and are not maximally flexible.

CNNs on Icosahedron: Related to our approach in using discrete representation, several works utilize an icosahedron for spherical image representation. As the most uniform and accurate discretization of the sphere, the icosahedron is the regular convex polyhedron with the most faces. A spherical mesh can be generated by progressively subdividing each face into four equal triangles and reprojecting each node to unit length. Lee *et al.* [11] is one of the first to suggest the use of icosahedrons for CNNs on omnidirectional images. Here, convolution filters are defined in terms of triangle faces. In [10], UGSCNN is proposed to efficiently train a convolutional network with spherical data mapped to an icosahedron mesh. Liu *et al.* [17] used the icosahedron based spherical grid as the discrete representation of the spherical images and proposes an azimuth-zenith anisotropic CNN for 3D shape analysis. Cohen *et al.* [12] employed an icosahedron mesh to present a gauge equivariant CNN. Equivariance is ensured by enforcing filter weight sharing across multiple orientations.

3 Reducing Distortion through Spheres

Our motivation for spherical CNNs goes beyond spherical input, as we observe reduced geometric distortion by using a spherical input domain for standard planar images.

In particular, we note that an image consists of rays representing a reduced description of the 3D world. Let us denote 3D points $\mathbf{P}_i$ projected onto the image plane and the sphere's manifold as $\mathbf{p}_i = \mathtt{hom}(\mathbf{P}_i)$ and $\mathbf{s}_i = \frac{\mathbf{P}_i}{\|\mathbf{P}_i\|}$ respectively, where $\mathtt{hom}(\cdot)$ computes the homogeneous coordinate. For general out-of-plane camera rotation $\mathbf{R}$, we observe distortions for planar, but not spherical projections, since it typically holds that (Figure 3 and Appendix A)

$$\|\mathtt{hom}(\mathbf{R}^\mathrm{T}\mathbf{P}_i) - \mathtt{hom}(\mathbf{R}^\mathrm{T}\mathbf{P}_j)\| \neq \|\mathbf{p}_i - \mathbf{p}_j\|, \tag{1}$$

but for spheres the distance has no distortion, as

$$\left\| \frac{\mathbf{R}^\mathrm{T}\mathbf{P}}{\|\mathbf{R}^\mathrm{T}\mathbf{P}\|} - \frac{\mathbf{R}^\mathrm{T}\mathbf{Q}}{\|\mathbf{R}^\mathrm{T}\mathbf{Q}\|} \right\| = \|\mathbf{s}_i - \mathbf{s}_j\|. \tag{2}$$

Therefore we hypothesize that CNN learning on spherical images should be able to generalize to more pixel

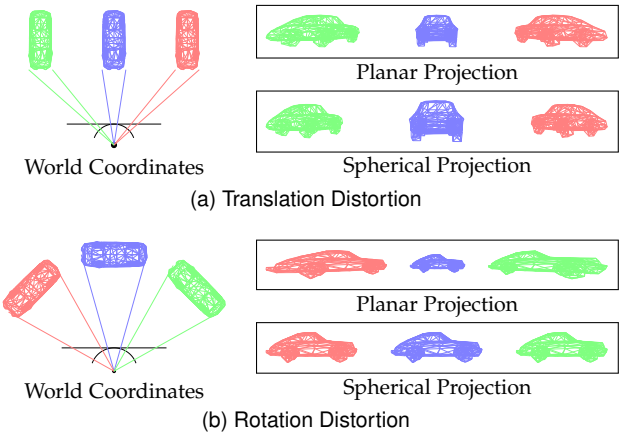


Fig. 3. Planar projection suffers from translation and rotation distortion, while spherical projection reduces distortions at translation and removes them completely for rotation.

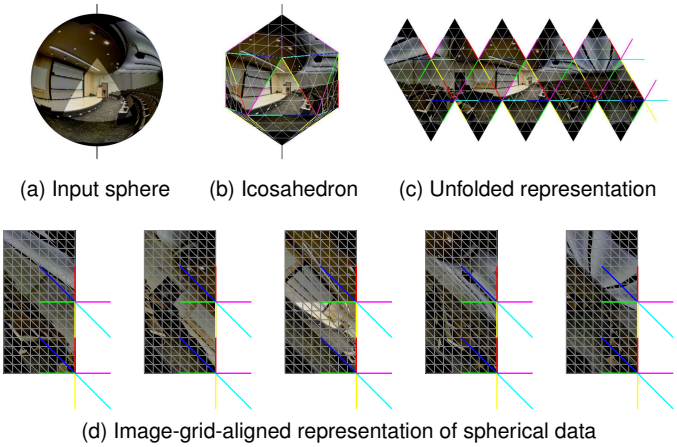


Fig. 4. Spherical input data (a) is represented by an icosahedron-based geodesic grid (b). Similar to cubes [15], [16], we unfold our mesh (c) and align its 5 components to the standard image grid (d) for efficient computation of convolution, pooling and up-sampling.

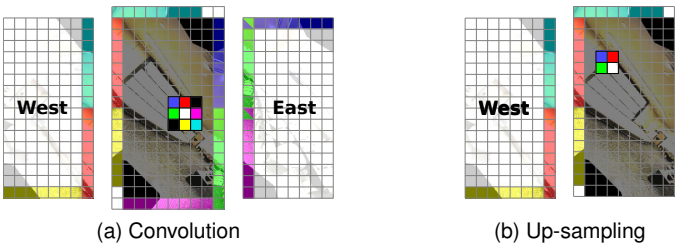


Fig. 5. Convolution with our hexagonal filters (a) and up-sampling (b) reduce to standard CNN operations after padding the sphere component with features from neighboring sphere parts. Pooling is computed with a standard 2x2 kernel with stride 2.

locations. As we strive for a confirmation of this, we note, since most datasets provide planar images, we require the camera calibration matrix to project to the sphere. Finally, we emphasize, distortion on alternative projections such as equirectangular or panorama images are reduced only along longitudes.

4 Proposed Full-HexNet Framework

We represent the spherical input through vertices on an icosahedron mesh (Figure 4). The mapping is based on the vertices' azimuth and zenith angles – *e.g.* the input color is obtained from an equirectangular input through interpolation. Similar to cube maps [15], [16], the icosahedron simplifies the sphere into a set of planar regions. While the cube represents the sphere only with 6 planar regions, the icosahedral representation is the convex geodesic grid with the largest number of regular faces. In total, our grid consists of 20 faces and 12 vertices at the lowest resolution, and $f_r = 20 * 4^r$ faces and $n_r = 2 + 10 * 4^r$ vertices at resolution level $r \geq 0$. Note, a resolution increase is achieved by subdivision of each triangular face into 4 equal triangular parts. In the following, we present an efficient orientation-aware implementation of convolutions and our down- and up-sampling techniques.

4.1 Orientation-aware Convolutions

If a camera is attached to a vehicle, the orientation and location of objects such as sky, buildings, sidewalks or roads are likely similar across the dataset. Therefore, we believe an orientation-aware system can be beneficial, while tasks with arbitrary rotations may benefit from rotation invariance [20] or weight sharing across rotated filters [12], [35].

4.1.1 Efficient Convolutions through Padding

We first define the north and south pole as any two nodes that have maximum distance on the icosahedron mesh. Similar to [12], [17], the mesh is then converted to a planar representation by unfolding it along the equator (Figure 4). Finally, we split the surface into five components, denoted $\{C_i\}_{i=1}^5$, and align the nodes with a regular image grid through a simple affine transformation.

Notice, each node has a neighborhood of either 5 or 6 points, denoted $p_i \in \mathcal{V}^{(r)} = \{p_i\}_{i=1}^{n_r}$ and $\mathcal{N}_i^{(r)} = \{q_j^i\}_{j=1}^5$ or $\{q_j^i\}_{j=1}^6$ respectively, where $\mathcal{N}_i^{(r)} \subset \mathcal{V}^{(r)}$ and j indexes the neighborhood of p_i in a clock-wise fashion. We write $\mathcal{V}^{(r)} \subset \mathcal{V}^{(r+1)}$, since only new nodes are introduced when resolution is increased. Note also, the connectivity at different resolutions changes (*i.e.* $\mathcal{N}_i^{(r)} \neq \mathcal{N}_i^{(r+1)}$). In the following we omit r for simplified notation where possible.

We employ hexagonal filters in our work, instead of regular 3×3 kernels. Let us ignore the vertices at the poles (*e.g.* through reasoning of dropout), and adjust the neighborhood cardinality to 6 for all vertices with 5 neighbors through simple repetition. Now, our planar representation of the icosahedron simplifies the convolution with hexagonal filters to standard 2D convolution with a masked kernel, after padding as shown in Figure 5.

4.1.2 North-alignment through Interpolation

In its natural implementation, our filters are aligned to the icosahedron mesh. Consequently, the filter orientation is inconsistent, since the surfaces near the north and south poles are stitched. We reduce the effect of such distortions by aligning filters vertically through interpolation (Figure 6).

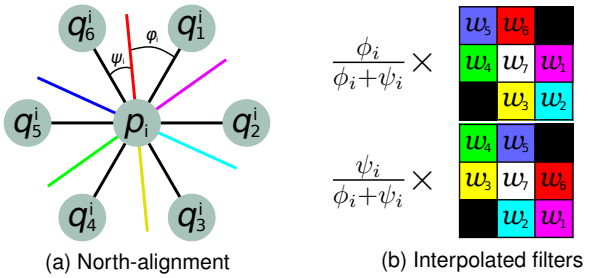


Fig. 6. Given arc-based interpolation of the neighborhood for north-alignment (a), our convolution is computed with 2 weighted filters (b). The weights are precomputed for all vertices.

The non-interpolated convolution with weights $\{w_j\}_{j=1}^{7}$ at node p_i and its neighbors $\{q_j^i\}_{j=1}^{6}$, is computed as $\sum_{j=1}^{6} w_j q_j^i + w_7 p_i$. Instead, we north-align the neighborhood with interpolations using arc-based weights $\{\theta_j^i\}_{j=1}^{6}$ as follows:

$$\sum_{j=1}^{6} w_j(\theta_j^i q_j^i + (1-\theta_j^i) q^i_{(j \bmod 6)+1}) + w_7 p_i. \tag{3}$$

Since the hexagonal neighborhood is approximately symmetric, we further simplify (3) by introducing a unified weight α_i, such that $\{\alpha_i \approx \theta_j^i\}_{j=1}^{6}$ holds. Hence we write

$$\alpha_i \left(\sum_{j=1}^{6} w_j q_j^i\right) + (1-\alpha_i)\left(\sum_{j=1}^{6} w_j q^i_{(j \bmod 6)+1}\right) + w_7 p_i. \tag{4}$$

Thus, north-aligned filters can be achieved through 2 standard convolutions, which are then weighted based on the vertices' interpolations α_i.

The arc-interpolation α_i is based on the angle distance between the direction towards the position of the first and sixth neighbors, denoted $\mathbf{q}_1^i$ and $\mathbf{q}_6^i$ respectively, and the north-south axis when projected onto the surface of the sphere. In particular, we first find the projective plane of the north-south axis $\mathbf{a} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\mathsf{T}$ towards the position $\mathbf{p}_i$ of p_i as the plane with normal $\mathbf{n}_i = \frac{\mathbf{p}_i \times \mathbf{a}}{|\mathbf{p}_i \times \mathbf{a}|}$. Since the spherical surface is approximated by the plane of vectors $\mathbf{p}_i - \mathbf{q}_1^i$ and $\mathbf{p}_i - \mathbf{q}_6^i$, we only require the angles between these vectors and the plane given by $\mathbf{n}_i$, to find interpolation $\alpha_i = \frac{\phi_i}{\phi_i + \psi_i}$ with

$$\begin{aligned} \phi_i &= \arccos \frac{(\mathbf{p}_i - \mathbf{q}_1^i)^\mathsf{T}(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^\mathsf{T})(\mathbf{p}_i - \mathbf{q}_1^i)}{\left|(\mathbf{p}_i - \mathbf{q}_1^i)\right| \left|(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^\mathsf{T})(\mathbf{p}_i - \mathbf{q}_1^i)\right|} \\ \psi_i &= \arccos \frac{(\mathbf{p}_i - \mathbf{q}_6^i)^\mathsf{T}(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^\mathsf{T})(\mathbf{p}_i - \mathbf{q}_6^i)}{\left|(\mathbf{p}_i - \mathbf{q}_6^i)\right| \left|(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^\mathsf{T})(\mathbf{p}_i - \mathbf{q}_6^i)\right|}. \end{aligned} \tag{5}$$

The resulting interpolation is visualized in Figure 7.

4.2 Pooling and Up-sampling

Down-sampling through pooling and bi-linear up-sampling are important building blocks of CNNs, and are frequently employed in the encoder-decoder framework of semantic segmentation (*e.g.* [2]). Pooling is aimed at summarising the neighborhood of features to introduce robustness towards image translations and omissions. Typically, a very small and non-overlapping neighborhood of 2×2 pixels is considered in standard images, to balance detail and redundancy.

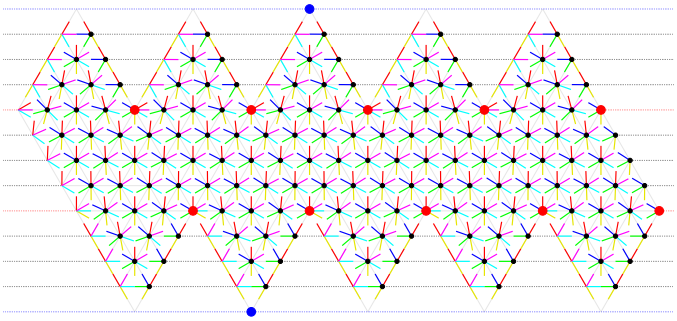


Fig. 7. Interpolated convolutions on the unfolded mesh ($r = 2$). Orientations are north-south aligned, while 5-degree connections (red) are padded through duplication, and poles (blue) are ignored.

Algorithm 1: Pad & WestPad (top & left only)

Result: Given sphere components $\{C_i\}_{i=0}^{4}$ of height $2W$ and width W compute padded $\{P_i\}_{i=0}^{4}$

for $i \leftarrow \{0, \dots, 4\}$ **do** // pad each component

$C_w \leftarrow C_{(i-1) \bmod 5}$; // west neighbor

$T \leftarrow \begin{bmatrix} C_w(W,W) \text{ to } C_w(1,W) & 0 \end{bmatrix}$;

$L \leftarrow \begin{bmatrix} \begin{bmatrix} C_w(W+1,W) \text{ to } C_w(2W,W) \end{bmatrix}^\mathsf{T} \\ \begin{bmatrix} C_w(2W,W-1) \text{ to } C_w(2W,1) \end{bmatrix}^\mathsf{T} \\ 0 \end{bmatrix}$;

$P_i \leftarrow \begin{bmatrix} T \\ \begin{bmatrix} L & C_i \end{bmatrix} \end{bmatrix}$; // top & left

if *pad all sides* **then**

$C_e \leftarrow C_{(i+1) \bmod 5}$; // east neighbor

$B \leftarrow \begin{bmatrix} 0 & C_e(2W,1) \text{ to } C_e(W+1,1) \end{bmatrix}$;

$R \leftarrow \begin{bmatrix} 0 \\ \begin{bmatrix} C_e(1,W) \text{ to } C_e(1,1) \end{bmatrix}^\mathsf{T} \\ \begin{bmatrix} C_e(1,1) \text{ to } C_e(W+1,1) \end{bmatrix}^\mathsf{T} \end{bmatrix}$;

$P_i \leftarrow \begin{bmatrix} \begin{bmatrix} P_i \\ B \end{bmatrix} & R \end{bmatrix}$; // bottom & right

end

end

Bi-linear up-sampling is used in the decoder to increase sub-sampled feature-maps to larger resolutions.

We note, in our icosahedron mesh the number of nodes increases by a factor of 4 for each resolution (excluding poles). Therefore during down-sampling from resolution r to $r-1$, we summarize a neighborhood of 4 at r with 1 node at $r-1$. A natural choice is to pool over $\{p_i, q_6^i, q_5^i, q_4^i\}$ for nodes $p_i \in \mathcal{V}^{(r-1)}$. Thus, we apply a simple standard 2×2 strided pooling with kernel 2×2 on each icosahedron part.

Analogously, bi-linear up-sampling or transposed convolutions are applied by padding the icosahedron parts at left and top followed by up-sampling by a factor of 2 in height and width (Figure 5). Due to padding, this results in a 1-pixel border at each size which we simply remove to provide the expected up-sampling result. Finally we emphasize, methods like pyramid pooling [3] can be computed by combining our pooling and up-sampling techniques.

4.3 Implementation Detail

We include the pseudo code of our main CNN operators, applied to the icosahedron mesh components, denoted $\{C_i\}_{i=0}^{4}$. Note, many operations will be a direct result of a combination of these operators (*i.e.* Pyramid Pooling Layers

Algorithm 2: Hexagonal Convolution (HexConv)

Result: Given components $\{C_i\}_{i=0}^4$ and precomputed interpolation weights $\{A_i\}_{i=0}^4$ get filter results $\{F_i\}_{i=0}^4$ of same size.

$\{C_i\}_{i=0}^4 \leftarrow \text{Pad}(\{C_i\}_{i=0}^4)$; `// Alg. 1`

$\mathbf{W}_1 \leftarrow \begin{bmatrix} w_5 & w_6 & 0 \\ w_4 & w_7 & w_1 \\ 0 & w_3 & w_2 \end{bmatrix}$; `// Hexagon filter`

$\mathbf{W}_2 \leftarrow \begin{bmatrix} w_4 & w_5 & 0 \\ w_3 & w_7 & w_6 \\ 0 & w_2 & w_1 \end{bmatrix}$; `// Shift weights`

for $i \leftarrow \{0, \dots, 4\}$ **do**

  $F_i^1 \leftarrow \text{conv2d}(C_i, \mathbf{W}_1)$; `// standard 2D conv`

  $F_i^2 \leftarrow \text{conv2d}(C_i, \mathbf{W}_2)$;

  `// Element-wise Interpolation`

  $F_i \leftarrow A_i \otimes F_i^1 + (1 - A_i) \otimes F_i^2$

end

Algorithm 3: Bi-linear Up-sampling on Sphere

Result: Given components $\{C_i\}_{i=0}^4$ get bi-linear up-sampling $\{F_i\}_{i=0}^4$.

$\{C_i\}_{i=0}^4 \leftarrow \text{WestPadding}(\{C_i\}_{i=0}^4)$; `// Alg. 1`

for $i \leftarrow \{0, \dots, 4\}$ **do**

  $F_i \leftarrow \text{upsample}(C_i)$; `// 2× up-sampling`

  Cut 1 pixel width from all sides of F_i;

end

[3]). First we detail padding in Algorithm 1. Our orientation-aware hexagonal convolutions with arc-based interpolations for north-alignment are given in Algorithm 2. Note, interpolation weights are precomputed. Algorithm 3 presents up-sampling. We emphasize, convolutions with kernel size 1, pooling, batch normalization, non-linearities and biases are directly computed on the spherical components without padding, through standard CNN operators (*e.g.* pooling in Algorithm 4).

4.4 s-Ring Convolutions

It is possible to include multiple rings to the convolution. Specifically, for an s-ring convolution, we include all nodes that have less than s graph distance to the convolution center. Again, after simple padding following the discontinuities on the icosahedron, an s-ring convolution can be expressed by a $(2s+1) \times (2s+1)$-kernel similar to the 3×3-kernel in Figure 6. We note however, since the arc-based interpolation is defined between neighbors $\mathbf{q}_1$ and $\mathbf{q}_6$ at 1 graph distance, it will be located somewhere between 3 nodes at graph distance 2 (Figure 8). Our uniform convolution across the sphere can thus be computed by an interpolation of $s+1$ standard convolutions. It is worth mentioning here that an interpolation weight for each graph distance – as later presented in Section 5.3 – is also possible to similar effect.

5 Masked-HexNet for Partial Input

Most deep learning literature focuses on novel network architectures (*e.g.* [1], [2], [3], [4], [22]) and new datasets (*e.g.* [14], [19], [36], [37]) for incremental performance improvements. In contrast, we aim to improve accuracy on

Algorithm 4: Pooling on Sphere

Result: Given components $\{C_i\}_{i=0}^4$ get pooling $\{F_i\}_{i=0}^4$.

for $i \leftarrow \{0, \dots, 4\}$ **do**

  $F_i \leftarrow \text{pooling}(C_i)$; `// stride 2 pooling`

end

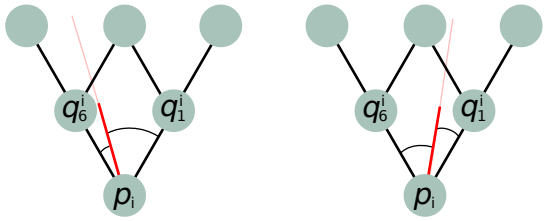


Fig. 8. North alignment, as computed by nodes at graph distance 1, will be located somewhere between $s+1$ nodes at graph distance s.

planar images through our geometric motivation in Section 3. Hence, we project onto the manifold of a sphere using known camera intrinsics, and reduce the distortion problem with spherical CNN. However to date, most existing spherical CNN methods cannot compete on high-resolution datasets [10], [11], [12], [13]. In particular, existing spherical CNNs assume that the visual information covers the whole sphere. As a result, the convolution operations have to be applied on all vertices of the icosahedron mesh. In practical scenarios however, such as the driving environments of Cityscapes [14], the active view covers less than 3% of the sphere's manifold. Consequently, a complete icosahedron of more than 9.6×10^6 vertices is required to be equivalent to the resolution of a $380px \times 760px$ image. We reformulate our spherical CNN for partial input to overcome this shortfall.

5.1 Graph-based Formulation of HexNet

Let us reformulate the full-HexNet from Section 4 using a graph-based interpretation, as we redefine convolutions, pooling and up-sampling (Figure 9).

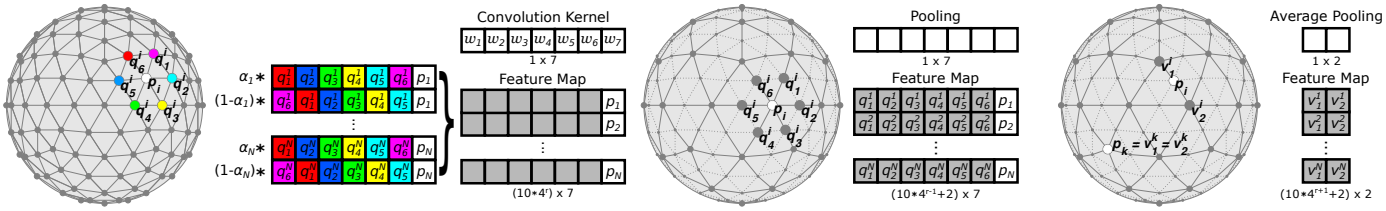
5.1.1 *Orientation-aligned Convolutions*

All vertices that are not on the base icosahedron, *i.e.* $p_i \notin \mathcal{V}^{(0)}$, have a neighborhood cardinality of six. As above, we increase the neighborhood of base vertices $p_i \in \mathcal{V}^{(0)}$ to six through duplication. Now, we can apply 1×7 convolutions on the gathered neighborhoods, an $n_r \times 7$ feature map where each row contains p_i and its neighborhood $\mathcal{N}_i^{(r)}$ (Figure 9(a)).[1] We apply our arc-based interpolation to efficiently enforce north-alignment of the kernel, as in (4). Note, the naïve implementation of (4) requires significant memory and is slow in running time. We address implementation details of masking and the execution in Section 5.2.

5.1.2 *Pooling and Up-sampling on the Sphere*

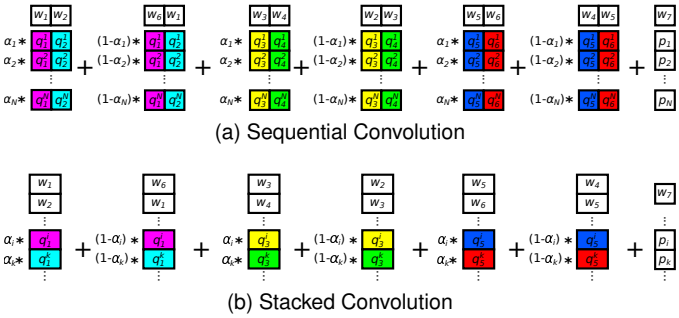
We redefine the pooling mechanisms as follows: During pooling we sub-sample from resolution r to $r-1$. Specifically, we gather each vertex $p_i \in \mathcal{V}^{(r-1)}$ and its neighborhood $\mathcal{N}_i^{(r)}$ into an $n_{r-1} \times 7$ feature map and apply a 1×7 pooling (Figure 9(b)). Note, any pooling operator can be

1. We omit notation of input and output channels for simplicity.



(a) Convolution (b) Pooling (c) Up-sampling

Fig. 9. Proposed graph-based CNN operations on the (full) icosahedron at resolution r. (a) For convolutions, sorted neighborhoods are gathered and north-aligned to generate a $(10 * 4^r + 2) \times 7$ feature map. (b) During pooling, a feature map of the $10 * 4^{r-1} + 2$ vertices at resolution $r - 1$ is built. (c) Up-sampling finds the parents at resolution $r + 1$ and interpolates.



(a) Sequential Convolution

(b) Stacked Convolution

Fig. 10. We present alternative graph-convolutions to save memory and running time. (a) Only three $m_r \times 2$ feature maps are needed to compute equivalent convolutions sequentially. (b) Stacking and padding the neighbourhood further reduces gathering needs.

used. Figure 9(c) shows the up-sampling process, where we increase resolution from r to $r + 1$. We find the new vertices $p_i \in \mathcal{V}^{(r+1)} \setminus \mathcal{V}^{(r)}$ through averaging their parents, given by $v_1^i, v_2^i \in \mathcal{V}^{(r)} \cap \mathcal{N}_i^{(r+1)}$. The existing vertices $p_i \in \mathcal{V}^{(r)}$ remain unchanged.

It is worth noting, in Section 4 only 3 nodes are used for efficiency, while here we use all 6 nodes for pooling. Either mechanism is possible, but since we want to reflect the up-sampling where two parents create a new node, we chose to have each node affecting the pooling of its two parents.

5.2 Masking the Active Areas of the Sphere

Typical camera setups utilize only one or few camera views. The active area which these views project onto are usually small. We emphasize, the computation of icosahedron-based CNNs on high-resolution usually requires the full sphere, and are thus unfeasible for such data [10], [12]. In contrast, with our graph-based implementation, it is possible to compute convolutions efficiently on a subset of pixels. We denote subset $\mathcal{M} \subset \mathcal{V}$, with cardinality $m_r = |\mathcal{V}^{(r)} \cap \mathcal{M}|$, and typically $m_r \ll n_r$. Convolution, pooling and up-sampling only requires the points in $\mathcal{V} \cap \mathcal{M}$, and we apply zero padding for neighbourhoods outside $\mathcal{M}$.

5.2.1 Efficiency Consideration with Connected Masks

In a naïve implementation, a 1×7 convolution needs to be applied to two sets of $m_r \times 7$ feature maps simultaneously to compute an interpolated convolution (Figure 9(a)). Since this is costly in memory, we present an alternative approach. By rearranging the convolution weights and the summations, we only require three sequential 1×2 convolutions on two $m_r \times 2$ feature maps which reduces memory requirements (Figure 10(a) and Appendix B). While the mask can be arbitrary, we can also reduce run time, if vertices in the mask are highly connected (as is the case in image data). Specifically, we exploit the fact that the neighborhoods of vertices frequently coincide, *i.e.* often there exists two vertices p_i and p_k such that $q_{j+1}^i = q_j^k$. Thus we optimize neighborhood connectivity, denoted $g_j = \begin{bmatrix} \dots & q_j^i & q_{j+1}^i = q_j^k & q_{j+1}^k & \dots \end{bmatrix}$ of size $l_r^{(j)} \times 1$, where $m_r < l_r^{(j)} \ll 2m_r$ (Figure 10(b) and Appendix B). Now, three sequential 1×2 kernels are applied on the $1 \times l_r^{(j)}$ feature map. Note, the ordering of g_j is precomputed. Algorithm 5 details the computations.

Algorithm 5: Graph-based Convolutions

Result: Given $1 \times M^{(r)} \times C$ input, gather-indices for g_j, and reverse gather-indices for F_j, compute convolution with filter $w_j \in \mathbb{R}^{1\times1\times O}$, where C and O are number of input and output channels, and $j = 1, \dots, 7$.
Output is $F \in \mathbb{R}^{1\times M^{(r)}\times O}$.

for $j = \{1,3,5\}$ **do**
- $g_j \leftarrow$ gathered $1 \times L_j^{(r)} \times C$ feature map for $q_{j+1}^i = q_j^k$
- // interpolation $\Rightarrow$ 2 convolutions as $\circledast$
- $G_j^a \leftarrow g_j \circledast \begin{bmatrix} w_j & w_{j+1} \end{bmatrix}$
- $G_j^b \leftarrow g_j \circledast \begin{bmatrix} w_{((j+4) \bmod 6)+1} & w_j \end{bmatrix}$
- $F_j^a \leftarrow$ gather $1 \times M^{(r)} \times O$ results from G_j^a
- $F_j^b \leftarrow$ gather $1 \times M^{(r)} \times O$ results from G_j^b

end

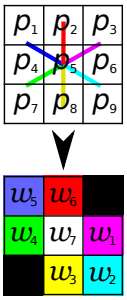
$F_7 \leftarrow [p_i]_{i=1}^{M^{(r)}} \circledast [w_7]$

// Using $\odot$ as element-product

$F \leftarrow [\alpha_i^{(r)}]_{i=1}^{M^{(r)}} \odot (F_1^a + F_3^a + F_5^a) + [1 - \alpha_i^{(r)}]_{i=1}^{M^{(r)}} \odot (F_1^b + F_3^b + F_5^b) + F_7$

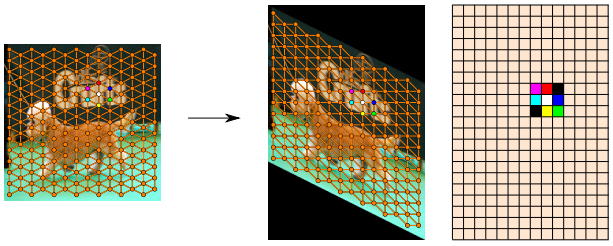
5.3 s-Ring Convolutions

We separate the neighbors within the s-ring based on the graph distances to implement graph-based s-ring convolutions: $\mathcal{N}_i = \bigcup_{j=1}^{s} \mathcal{D}_i^j$, where nodes in D_i^j have graph distance j. Here north-alignment is defined for each D_i^j separately by arc-interpolation for the 2 most northern nodes in D_i^j, resulting in s interpolation weights. While all nodes in the s-ring are gathered at each convolution center, only 2 convolutions are required in contrast to $s + 1$ in Section 4.4.

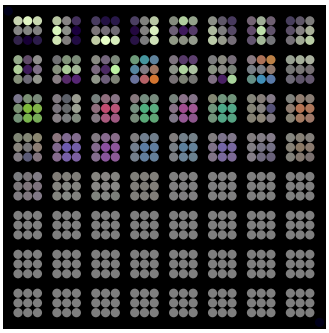


$$
\begin{aligned}
w_6 &= p_2 \\
w_5 &= \sin\tfrac{\pi}{3}\tfrac{p_1+p_4}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_2+p_5}{2} \\
w_4 &= \sin\tfrac{\pi}{3}\tfrac{p_4+p_7}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_5+p_8}{2} \\
w_3 &= p_8 \\
w_2 &= \sin\tfrac{\pi}{3}\tfrac{p_6+p_9}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_5+p_8}{2} \\
w_1 &= \sin\tfrac{\pi}{3}\tfrac{p_3+p_6}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_2+p_5}{2} \\
w_7 &= p_5
\end{aligned}
$$

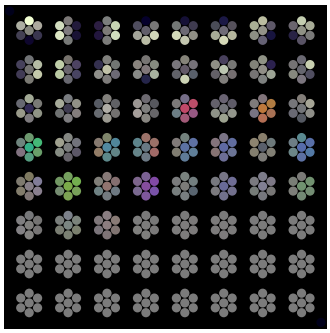
Fig. 11. The weights of conventional 3×3 kernels trained on perspective data can be transferred to our model *via* simple interpolation as our filters operate on the sphere's tangent planes.



(a) Projected Sphere Kernel



(b) Planar Filters



(c) Spherical Filters

Fig. 12. Spherical ImageNet pretraining: (a) Data is sampled on a hexagonal grid and projected with a shear transform to provide input with standard image grid on which masked 3×3 convolutions are employed. (b) First layer filter response is similar to planar filters. (Visualization is normalized and sorted by intensities.)

6 Weight Initialization from Planar Data

We argue that spherical methods are advantageous as they allow for a holistic interpretation of the environment. However, standard CNNs have so far focused mainly on planar projected images. Since we want to benefit from the vast research and datasets of planar domains, we discuss planar to spherical weight transfer in Section 6.1 and pretraining from planar data in Section 6.2.

6.1 Weight Transfer from Trained Planar Networks

Similar to SphereNet [6], our network applies an oriented filter at the local tangent plane of each vertex on the sphere. Consequently, the transfer of pretrained perspective network weights is naturally possible in our setup. Since we apply hexagonal filters with 7 weights, we interpolate from the standard 3×3 kernels as shown in Figure 11. Specifically, we align north and south of the hexagon with the second and eighth weight of the standard convolution kernel respectively. Bi-linear interpolation provides the remaining values for our filter. After transfer, weight refinement is necessary, but can be computed on a much smaller dataset (as done in [6]), or reduced learning iterations.

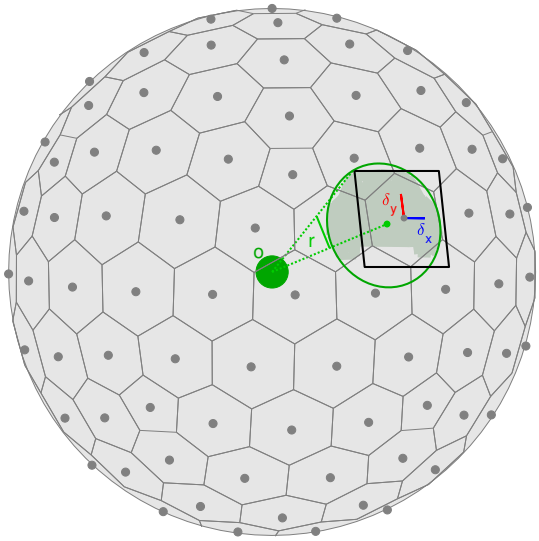


Fig. 13. Bounding circles are defined by the nearest vertex on the sphere using the offset δ_x and δ_y on the sphere's tangent plane, and object size is given by radius r, which is the angle between center and boundary of the object on the sphere ($o = (0, 0, 0)$ denotes the sphere's center).

6.2 Spherical Pretraining from Planar Datasets

Pretraining leverages large datasets (*e.g.* ImageNet [23]) to provide improved initialization of common network parameters (*e.g.* ResNet-50 [24]). Unfortunately, however, spherical datasets are rare. Weight transfer from planar networks is an option. However, since we have access to training data, a direct training algorithm for spherical parameter initialization from planar data is likely more suited.

Since our kernels operate on the tangent plane of the sphere, planar equivalents can be found. We visualize our input data in (Figure 12(a)), where we also link spherical convolutions to masked 3×3 kernels on the planar image domain. Since the camera matrix is unknown for ImageNet, we apply scale and crop data augmentation to simulate different camera intrinsics. Note, pretraining only needs to provide parameters with a good initialization.

In our work, we utilize ResNet-50 [24]. Following [3], we replace the initial 7×7 filter by two consecutive 3×3 kernels, or more specifically, the hexagonal kernel. The ImageNet classification task completes with 25.03% error rate (7.55% error for top 5). In Figure 12(b) we visualize the filter weights of the initial layer. Note, our weights have similar properties to standard planar convolution layers [38].

7 Bounding Circles on Spheres

HexNet is a general convolutional framework and so not limited to spherical semantic segmentation. The core layers described above enable most common deep learning tasks. The use of HexNet for pixel-based segmentation and holistic classification is easily derived from planar approaches. The task of object detection with boundaries is however non-trivial and thus discussed in this section.

Classical object detection employs axis aligned bounding boxes on planar images to highlight detected objects. Specifically, Redmon *et al.* [39] defined YOLO-v3 with bounding boxes that use quantized pixel location x and y with related offset δ_x and δ_y, and height and width h and w respectively, such that the boundary is given by coordinates

$$\text{top\_left} = (x + \delta_x - \frac{w}{2}, y + \delta_y - \frac{h}{2}) \quad (6)$$

$$\text{bottom\_right} = (x + \delta_x + \frac{w}{2}, y + \delta_y + \frac{h}{2}). \quad (7)$$

We now introduce an alternative representation of ground truth for the spherical domain (Figure 13). Similarly to YOLO-v3, we first find the nearest vertex on the icosahedron mesh. Using the orientation aligned tangent plane of the vertex, we define offsets δ_x and δ_y. The size of the object is given by radial radius r.

Typically, object detection loss and non-max suppression needs object overlaps. On spheres, we calculate the overlap A of two bounding circles through the intersection between two caps [40]:

$$\begin{aligned} A = \; & 2\pi - 2\pi\cos(r_1) - 2\pi\cos(r_2) \\ & -2\arccos\left(\frac{\cos(\delta) - \cos(r_1)\cos(r_2)}{\sin(r_1)\sin(r_2)}\right) \\ & +2\cos(r_1)\arccos\left(\frac{\cos(r_1)\cos(\delta) - \cos(r_2)}{\sin(r_1)\sin\delta}\right) \\ & +2\cos(r_2)\arccos\left(\frac{\cos(r_2)\cos(\delta) - \cos(r_1)}{\sin(r_2)\sin\delta}\right) \quad (8) \end{aligned}$$

where r_1 and r_2 is the estimate radius of each prediction, and δ is the arc-distance between the centres of the sphere caps on the sphere's manifold. Finally we note, while this representation is suitable for simple objects, elongated objects may be represented by slanted ellipses in future work.

8 Evaluation

Our evaluation presents HexNet on 3 tasks across 5 datasets, comparing to planar and spherical methods. First, in Section 8.1 and Section 8.2, rotation invariant classification and segmentation is shown. Section 8.3 presents results on an orientation aware indoor segmentation task. Spherical semantic segmentation in urban road scenes are compared to spherical and planar state of the art in Section 8.4, which also shows a detailed ablation study of HexNet. Finally, object detection results are shown in Section 8.5.

8.1 Spherical MNIST: Rotation-invariant Classification

We follow [20] in the preparation of the spherical MNIST dataset, as we prepare non-rotated training and testing (N/N), non-rotated training with rotated testing (N/R) and rotated training and testing (R/R) tasks. Specifically, planar images of digits are projected onto the unit sphere surface. For the non-rotated version, digits are moved to the equator to prevent ambiguity at north or south pools, while for the rotated version, a random rotation is applied without ruling out pole-coverage. Both non-rotated and rotated versions are generated using public source code provided by UGSCNN [10].[2] Training set and test set include 60,000 and 10,000 digits, respectively. Input signals for this experiment are on a level-4 mesh (*i.e.* $r = 4$). The residual U-Net architecture of [10], including the necessary modifications to adapt to the classification task, is used in our experiments (Appendix C.1). We call this network Hex-RUNet-C.

2. https://github.com/maxjiang93/ugscnn

TABLE 1
Spherical MNIST with non-rotated (N) and rotated (R) training and test data. Orientation-aware Hex-RUNet-C is competitive only when training and test data match (*i.e.* N/N and R/R).

Method	Orientation	N/N	N/R	R/R
Spherical CNN [20]	invariant	96.__	**94.__**	95.__
Gauge Net [12]	part-invariant	99.43	69.99	**99.31**
UGSCNN [10]	aware	99.23	35.60	94.92
Hex-RUNet-C	aware	**99.45**	29.84	97.05

Hex-RUNet-C is compared to other spherical frameworks: Spherical CNN [20], Gauge Net [12] and UGSCNN [10] in Table 1. Our method outperforms previous methods for N/N, achieving 99.45% accuracy. In R/R, our method performs better than competing Spherical CNN and UGSCNN. Gauge Net benefits from weight sharing across differently oriented filters, and achieves best accuracy for this task amongst all approaches. Similar to [10], our method is orientation-aware by design and thus not rotation-invariant. Therefore, it is expected to not generalize well to randomly rotated test data in the N/R setting, while Spherical CNN performs best in this case.

8.2 Climate Pattern: Orientation-free Segmentation

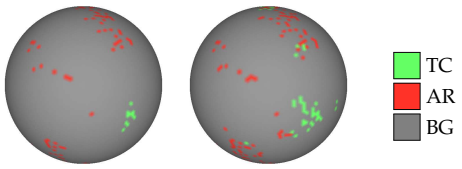


Fig. 14. Semantic segmentation results of Hex-RUNet-32 on climate pattern (right) in comparison to ground truth (left).

We further evaluate our method on the task of climate pattern segmentation. The task is first proposed by Mudigonda *et al.* [21], and the goal is to predict extreme weather events, *i.e.* Tropical Cyclones (TC) and Atomospheric Rivers (AT), from simulated global climate data. The training set consists of 43,916 patterns, and 6,274 samples are used for validation. As above, we use the same residual U-Net architecture as UGSCNN [10] (Appendix C.2). We include two variants using different numbers of parameters: Hex-RUNet-8 and Hex-RUNet-32 use 8 and 32 as output channels for the first convolution layer, respectively. Eval-

TABLE 2
Climate pattern segmentation results. We include mean class accuracy and mean average precision (mAP) where available. (The background class is denoted BG.)

Method	BG	TC	AR	Mean	mAP
Gauge Net [12]	**97.4**	**97.9**	**97.8**	**97.7**	**0.759**
UGSCNN [10]	97._	94._	93._	94.7	-
Hex-RUNet-8	95.71	95.57	95.19	95.49	0.518
Hex-RUNet-32	97.31	96.31	97.45	97.02	0.555

TABLE 3
Mean intersection over union (IoU) comparison on 2D3DS dataset. Per-class IoU is shown when available.

Method	mIoU	beam	board	bookcase	ceiling	chair	clutter	column	door	floor	sofa	table	wall	window
UNet-sphere	35.9	8.5	27.2	30.7	78.6	35.3	28.8	4.9	33.8	89.1	8.2	38.5	58.8	23.9
Gauge Net	39.4	–	–	–	–	–	–	–	–	–	–	–	–	–
UGSCNN	38.3	8.7	32.7	33.4	82.2	42.0	25.6	10.1	41.6	87.0	7.6	41.7	61.7	23.5
Hex-RUNet	**43.3**	**10.9**	**39.7**	**37.2**	**84.8**	**50.5**	**29.2**	**11.5**	**45.3**	**92.9**	**19.1**	**49.1**	**63.8**	**29.4**

TABLE 4
Mean class accuracy (mAcc) comparison on 2D3DS dataset. Per-class accuracy is shown when available.

Method	mAcc	beam	board	bookcase	ceiling	chair	clutter	column	door	floor	sofa	table	wall	window
UNet-sphere	50.8	17.8	40.4	59.1	91.8	50.9	**46.0**	8.7	44.0	94.8	26.2	68.6	77.2	34.8
Gauge Net	55.9	–	–	–	–	–	–	–	–	–	–	–	–	–
UGSCNN	54.7	19.6	48.6	49.6	93.6	63.8	43.1	**28.0**	63.2	**96.4**	21.0	70.0	74.6	39.0
Hex-RUNet	**58.6**	**23.2**	**56.5**	**62.1**	**94.6**	**66.7**	41.5	18.3	**64.5**	96.2	**41.1**	**79.7**	**77.2**	**41.1**

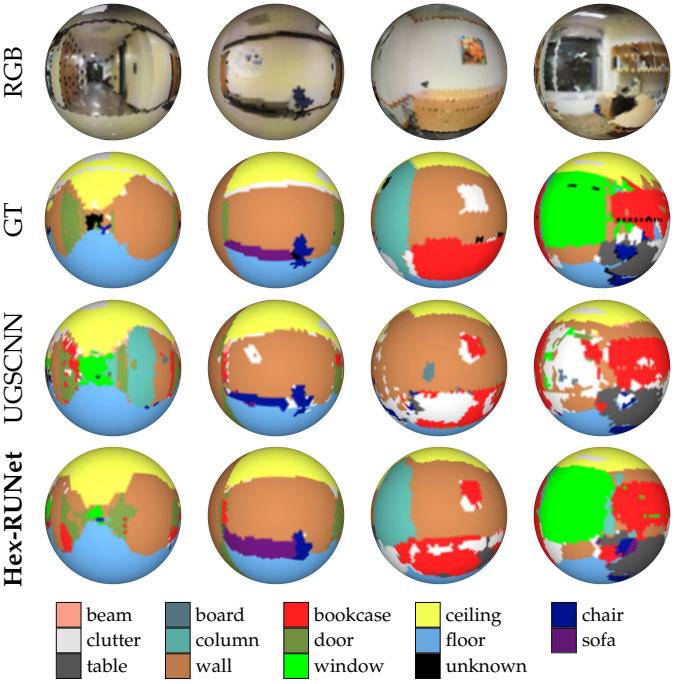
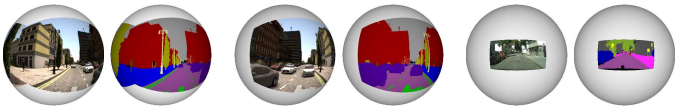


Fig. 15. Qualitative segmentation results on 2D3DS dataset.



(a) Synthia-O (b) Synthia-S (c) Cityscapes

Fig. 16. We use (a) omni-directional view Synthia to compare with state-of-the-art spherical CNNs, and (b) single view Synthia and (c) Cityscapes to compare with planar alternatives.

uation results on the validation set are shown in Table 2 and Figure 14. Hex-RUNet-8 and Hex-RUNet-32 outperform UGSCNN in terms of mean accuracy. With 32 features, Hex-RUNet-32's mean accuracy is similar to best performing Gauge Net. However, our method does not match Gauge Net in terms of mean average precision (mAP). We attribute this to the fact that there is no direct orientation information to exploit in this climate data. In contrast, Gauge Net shows its advantage of weight sharing across orientations.

8.3 Stanford 2D3DS: Indoor Segmentation

For our first orientation-aware omnidirectional semantic segmentation experiment, we evaluate our HexNet framework on the 2D3DS dataset [18], which consists of 1,413 equirectangular RGB-D images. The groundtruth attributes each pixel to one of 13 classes. Following [10], we convert the depth data to be in meter unit and clip to between 0 and 4 meters. RGB data is converted to be in the range of [0, 1] by dividing 255. Finally, all data is mean subtracted and standard deviation normalized. The preprocessed signals are sampled on a level-5 mesh ($r = 5$) using bi-linear interpolation for images and nearest-neighbors for labels. Class-wise weighted cross-entropy loss is used to balance the class examples. Using our proposed network operators, we employ the residual U-Net architecture of [10], which we call Hex-RUNet (Appendix C.3). We evaluate following the 3-fold splits of 2D3DS.

Qualitative results are shown in Figure 15 and we report the mean intersection over union (mIoU) and class accuracy (mAcc) in Table 3 and Table 4, respectively. Proposed Hex-RUNet outperforms orientation-aware UGSCNN [10], rotation-equivariant Gauge Net [12] and the U-Net baseline [2] on equirectangular images (denoted UNet-sphere) that have been sub-sampled to match level-5 mesh resolution. As for per-class evaluations, our method achieves best performance in most classes. This demonstrates that semantic segmentation benefits from an orientation-aware framework with more expressive filters than UGSCNN [10].

8.4 Segmentation on Urban Roads

We now test HexNet in more detail, first in comparison to other spherical methods, and then as substitute to other planar CNNs using partial spheres.

8.4.1 Datasets

It is important to realize that we require camera calibration matrices to project onto the sphere. Therefore, two datasets with known camera intrinsic are used.

SYNTHIA: The SYNTHIA dataset [19] contains photo-realistic frames of synthetic sequences rendered from a virtual scene. It provides pixel-level semantics for 13 classes.

TABLE 5
Mean IoU comparison at $r = 6$ on Synthia-O dataset.

Method	mIoU	building	car	cyclist	fence	marking	misc	pedestrian	pole	road	sidewalk	sign	sky	vegetation
UNet	38.8	80.8	59.4	0.0	0.3	54.3	12.1	4.8	16.4	74.3	58.2	0.2	90.4	49.6
UGSCNN	36.9	63.3	33.3	0.0	0.1	73.7	1.2	2.3	10.0	79.9	**69.3**	1.0	89.1	**56.3**
Hex-UNet	**43.6**	**81.0**	**66.9**	0.0	**2.9**	71.0	**13.7**	**5.6**	**30.4**	83.1	67.0	**1.5**	**93.3**	50.2

TABLE 6
Per-class accuracy comparison at $r = 6$ on Synthia-O dataset.

Method	mAcc	building	car	cyclist	fence	marking	misc	pedestrian	pole	road	sidewalk	sign	sky	vegetation
UNet	45.1	91.9	63.6	0.0	4.5	57.1	17.9	5.0	19.7	88.8	73.9	0.2	94.8	69.3
UGSCNN	50.7	**93.2**	**81.4**	0.0	**5.3**	83.2	33.7	2.5	14.9	90.8	82.7	1.3	96.1	74.0
Hex-UNet	**52.2**	88.7	72.7	0.0	3.3	**85.9**	36.6	**6.2**	**42.5**	89.6	**83.7**	**1.6**	95.6	71.6

TABLE 7
Datasets' training and test samples, the native resolution and coverage of sphere.

Dataset	#Train	#Test	Resolution	Coverage
Synthia-O	1818	451	2096×4192	54.69%
Synthia-S	7272	1804	760×1280	14.92%
Cityscapes	2975	500	1024×2048	2.74%

We use a subset of the SYNTHIA datset, and create an omni-directional version. In particular, we select the "Summer" sequences of all five places ($2\times$New York-like, $2\times$Highway and $1\times$European-like) to create omni-directional data. We split the dataset into a training set of 1818 images (from New York-like and Highway sequences) and use 451 images of the European-like sequence for validation. Only RGB channels are used in our experiments. The icosahedron mesh is populated with data from 4 viewing direction per camera pose using interpolation for RGB data and nearest neighbor for labels. We use two setups: **O**mnidirectional and **S**ingle-view, denoted Synthia-O and Synthia-S respectively (Figure 16). In Synthia-O, we use the omnidirectional images merged from 4 single-view images. In Synthia-S, images of different viewpoints are projected to the same place on the sphere, and treated as individual samples. Further details are given in Table 7.

Cityscapes: We also evaluate on Cityscapes [14]. This dataset contains a diverse set of high-resolution images captured from 50 different cities in Europe. It comes with high quality semantic labels and we use 19 classes for our evaluation. The training set consists of 2,975 images and 500 images are used for validation (Table 7).

8.4.2 Comparing Spherical CNNs

We report mIoU and mAcc. Here we use the standard U-Net architecture [2]. We call this network Hex-UNet. We compare our method to UGSCNN [10] using data sampled at mesh level-6 ($r = 6$). We also include planar U-Net [2] using original perspective images, which have been sub-sampled to match the icosahedron resolution. Specifically, we count the number of vertices on the icosahedron mesh that fall onto the image region. We then set the image resolution to be approximately equivalent to this number of vertices, resulting in image resolution 48×80 for level-6, 96×160 for level-7 and 192×320 for level-8 meshes. Table 5 and 6 report mIoU and mAcc respectively, while

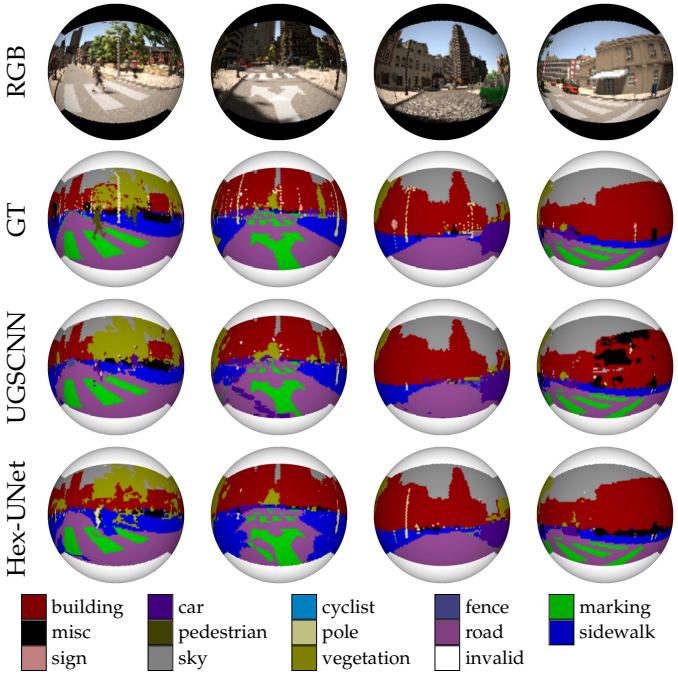


Fig. 17. Segmentation results on Synthia-O dataset.

Figure 17 shows qualitative results. Hex-UNet outperforms previous state of the art with significant margin across most classes. The performance on small objects, *e.g.* "pedestrian" and "sign", is poor, while all methods fail for "cyclist". We attribute this to an unbalanced dataset. It is worth noting here, class-wise weighted cross-entropy loss is not used.

Finally we note that most previous methods report results only up to mesh resolution level $r = 5$ which consists of merely 2,562 vertices to represent omnidirectional input. We evaluate our method at different resolutions ($r = \{6, 7, 8\}$), shown in Table 8. Our method achieves best performance at $r = 7$. Since we use a standard U-Net structure consisting of only 4 encoder (and decoder) layers, perception of context is reduced at $r = 8$. This is further illustrated by Figure 18 (last column), where a car's wheel is misclassified as road-markings at $r = 8$. Resolution $r = 6$ and $r = 7$ are able to adequately label this.

8.4.3 Comparing Spherical with Planar CNNs

Motivated by reduced distortion (Section 3) we investigate the impact of the spherical representation for planar datasets

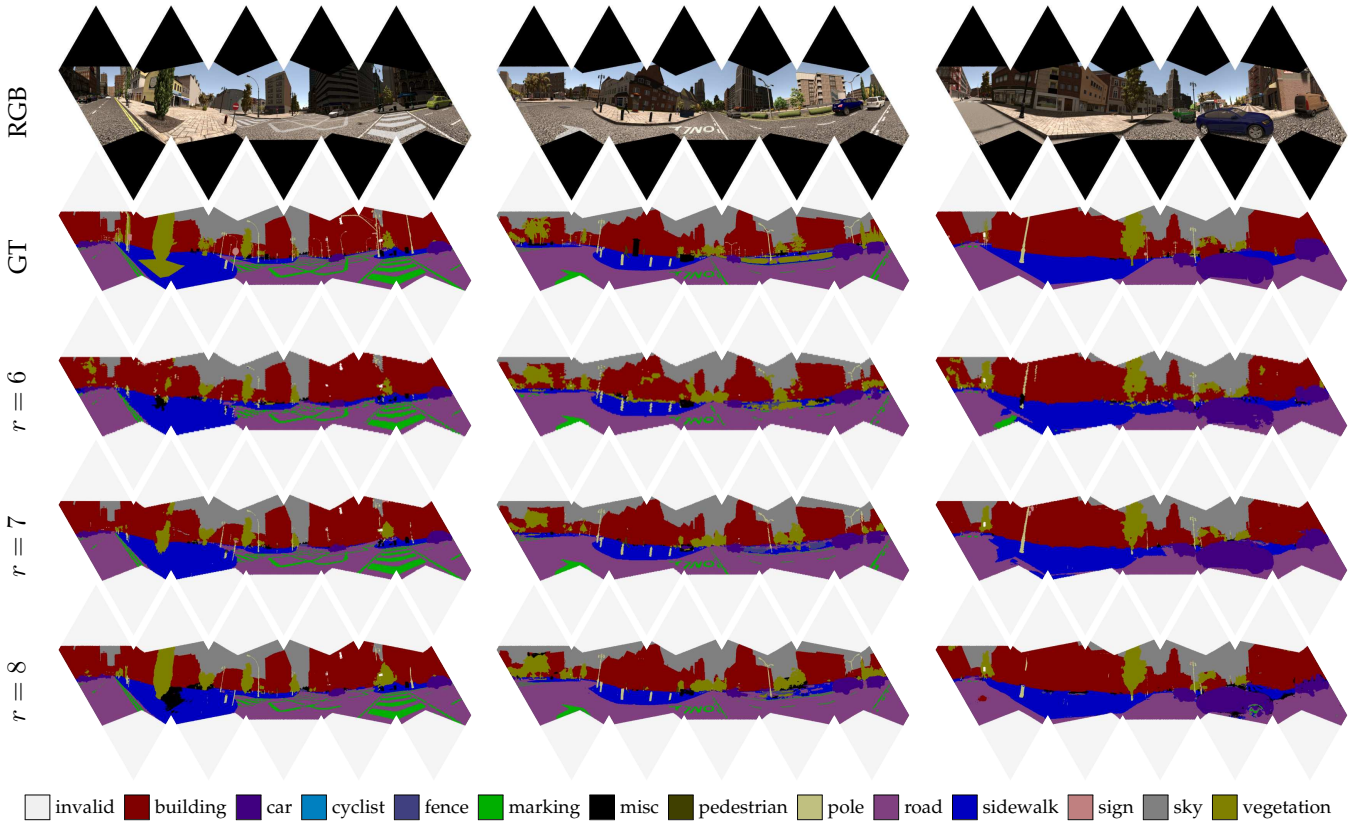


Fig. 18. Unfolded visualization of semantic segmentation results of Hex-UNet at different resolutions on Synthia-O dataset.

TABLE 8
Evaluation at different resolution on Synthia-O. (Current implementation of [10] could not fit data with resolution at $r = 8$. Note ground-truth at lower resolution is sub-sampled, thus evaluations of different resolutions are only indicative.)

Method	$r = 6$		$r = 7$		$r = 8$	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
UNet	38.8	45.1	44.6	52.6	43.8	52.4
UGSCNN	36.9	50.7	37.6	48.9	–	–
Hex-UNet	**43.6**	**52.2**	**49.5**	**57.1**	**47.1**	**55.1**
Hex-UNet-T	36.7	44.8	38.0	47.2	45.3	52.8
Hex-UNet-nI	42.4	50.6	45.1	53.4	45.4	53.2

as we compare planar and spherical projections on the single-view semantic segmentation task of Synthia-S and Cityscapes. Both datasets have known camera intrinsics, which is necessary for projecting planar images onto the unit sphere. We project planar images onto the sphere using bi-linear interpolation. All results are evaluated after back projecting into the planar domain at full resolution for fair comparison.[3]

Our graph-based interpretation of HexNet provides essential building blocks for existing CNN architectures. We choose U-Net [2][4] and DANet [22][5] for evaluation. Specifically, we employ a residual U-Net which comprises a residual encoder and decoder branch [10], [25]. As for DANet, ResNet-50 [24] is adopted as feature extraction backbone, then dual attention blocks (spatial and channel) are employed to facilitate accurate segmentation [22]. In our implementation we follow [3], and replace the initial 7×7 convolution with two 3×3 convolutions, or more specifically our hexagonal kernel with 1-ring neighborhood.[6] In all experiments, we use Adam optimizer [41] with learning rate 0.001, without learning rate decay, and train until convergence. Data augmentation is not used. The number of trainable parameters for spherical U-Net is 3.3M (5.0M for planar), and 41.8M for DANet (50.1M for planar). Since we reduce 3×3 kernels with 1-ring neighborhood kernels of 7 weights, our networks use less parameters.

We study different input size as we match spherical resolution to similar planar equivalents (Table 9). On Synthia-S, a mesh for level-7 ($r = 7$) and level-8 ($r = 8$) is matched to $1/6$ and $1/3$ of full resolution respectively. In Cityscapes, level-9 and level-10 is employed and matched to $1/6$ and $1/3$ resolution respectively. Since $r \in \mathbb{N}$, mesh resolution cannot be arbitrary (*e.g.* $1/2$ or $1/4$). We sub-sample to ensure minimal interpolation artifacts.

In Table 9 we report mean intersection over union (mIoU) for the residual U-Net and the DANet architecture. First, we discuss results without pretraining. Our spherical representation consistently achieves improved results over the standard planar version on both datasets. We also note, both methods improve at higher resolution. Therefore we conclude, it is necessary to develop spherical CNN methods that support high-resolution data.

3. In fact, the evaluation is biased towards improved planar accuracy due to evaluation on planar ground truth.
4. We reimplement original U-Net with residual blocks used in [10]
5. Code available at https://github.com/junfu1115/DANet
6. In the appendix we show that two 1-ring convolutions are more efficient with similar accuracy to one 3-ring convolution

TABLE 9
U-Net and DANet results on Synthia-S and Cityscapes for planar and spherical images at different resolutions. Results are computed on ground truth in planar domain at full resolution. Planar and spherical pretraining with ImageNet (Section 6.2), and pretrained weight transfer (Section 6.1) is additionally reported for DANet.

Dataset	Input Resolution	#Points	U-Net mIoU(%)	DANet mIoU (%) No Pretraining	Pretraining	Transfer
Synthia-S	planar@1/6	24,563	53.2	47.0	51.0	-
	sphere@lv-7	24,467	**54.3**	**50.0**	**51.4**	49.29
	planar@1/3	98,494	56.5	54.9	58.6	-
	sphere@lv-8	97,750	**57.6**	**56.0**	**58.7**	59.35
Cityscapes	planar@1/6	72,200	51.5	51.2	**57.9**	-
	sphere@lv-9	71,652	**54.3**	**52.5**	56.6	56.56
	planar@1/3	288,800	55.5	63.0	67.0	-
	sphere@lv-10	286,175	**56.3**	**63.1**	**67.8**	66.69

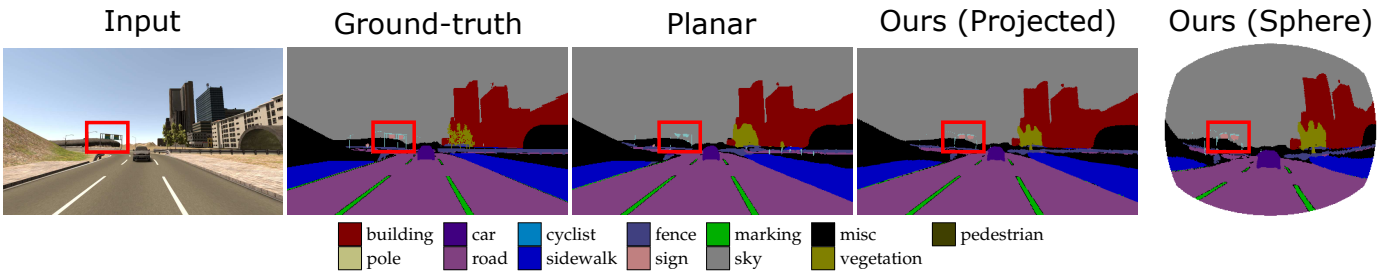


Fig. 19. Qualitative results using DANet on Synthia-S for mesh level-8 or resolution 1/3. The sign is missed by planar methods, while our spherical CNN labels this correctly.

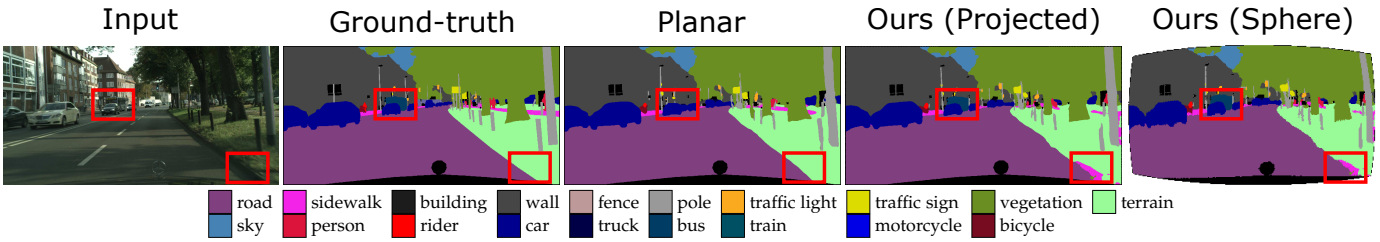
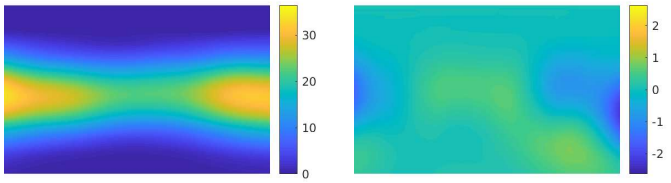


Fig. 20. Qualitative results using DANet on Cityscapes for mesh level-10 or resolution 1/3. The bus in the centre of the image is missed with planar distortions, while spherical projection correctly labels this. Planar methods detect terrain on image boarder more accurately.



(a) Training Distribution (b) $TP_s(i) - TP_p(i)$

Fig. 21. Training data location bias (a) and results difference (b) between spherical and planar method for 'pedestrian' in Synthia-S (after Gaussian filter).

DANet employs ResNet-50 for its feature extraction. We now compare planar and spherical pretraining (Section 6.2), and the spherical weight transfer of pretrained weights (Section 6.1). In Table 9, the planar representation benefits more from pretraining, *e.g.* achieving 6.74% gain at 1/6 resolution for Cityscapes. Nevertheless, pretraining improves the spherical performance by more than 4% in Cityscapes data. Overall, we can improve segmentation accuracy to a competitive level with spherical pretraining throughout all experiments.[7] We believe that since our pretraining utilizes ImageNet data with planar images, improvements are slightly reduced. A spherical version of ImageNet with camera calibration matrices may be beneficial in future work. Finally we note for most cases, that even simple interpolated weight transfer improves the overall performance of spherical methods, but at a reduced scale.

In Figure 19 and Figure 20, qualitative results are given. We observe that objects with fixed size are distortion dependent and therefore the spherical projection improves results (*e.g.* 'bus', 'traffic sign'), while continuous objects suffer less from distortion (*e.g.* 'terrain'). We further compare, and compute a per-class prediction heatmap, which shows the difference between the true positive numbers per method, $TP_s(i)$ and $TP_p(i)$ for spherical and planar respectively, at pixel location i, *i.e.* $TP_s(i) - TP_p(i)$. Figure 21 shows the heatmap for 'pedestrian' in Synthia-S. Note, while the training distribution biases the class to left and right part of the frame, our method is able to improve recognition results in centre and bottom of the image. This supports our hypothesis of Section 3, where we suggest that fewer

7. Cityscapes accuracy @ 1/2 and @ 1/4 is 71.8% [42] and 59.1% [43], hence 67.8% @ lv-10 is competitive.

TABLE 10
Omni-directional Synthia-O results for our method, as full and partial input, compared to UGSCNN. Inference memory and run-time is given for batch size 1, per sample.

Method	mIoU (%)	Memory (MB)	Time (s)
UGSCNN	37.6	6,831	1.52
HexNet (full)	48.3	1,063	0.02
Graph-HexNet (full)	49.5	3,596	0.16
Graph-HexNet (masked)	**50.1**	**595**	**0.07**

distortion aids generalization.

8.4.4 Ablation Study

Weights Transfer: While we have already shown perspective weight transfer for pretraining in Table 9, we now apply direct weight transfer from a perspective network. Initialized with the learned filters (3×3 kernels) from perspective U-Net, we perform weight refinement of only 10 epochs (in contrast to up-to 500 epochs otherwise), and report results as "Hex-UNet-T" in Table 8. The proposed filter transfer obtains competitive results, especially at resolution level $r = 8$.

Arc-based Interpolation: We evaluate our method without north-alignment (Section 4.1), denoted as "Hex-UNet-nI" in Table 8. Here, Hex-UNet performs better than Hex-UNet-nI, thus verifying the importance of orientation-aware filters in semantic segmentation.

Spherical CNN Runtime: It is not uncommon that the valid information only covers partial areas on the sphere. In this scenario, most spherical CNNs require costly memory and runtime by consuming full spheres. In this experiment, we compare execution of Full-HexNet and Masked-HexNet with UGSCNN [10] using Synthia-O data. All networks employ the residual U-Net architecture. Table 10 shows accuracy, memory and run-time at mesh level-7. Overall, our partial implementation performs best, since only active areas are used for computations. It is worth noting that Synthia-O input is not of full spheres, thus Full-HexNet remains less efficient than Masked-HexNet due to computation of unused data.

Gathering Efficiency: In Table 11, we compare our implementation of Algorithm 5 (Sph-v3) with naïve convolutions in Figure 9(a) (Sph-v1) and sequential version in Figure 10(a) (Sph-v2). We include planar DANet as baseline. We note, for spherical convolutions, the grouped version (Sph-v3) has overall best memory and run-time performance. However, compared to planar training, spherical CNN is still inferior. Here we note, our current implementation of feature gathering is costly in back-propagation as we do not exploit the one-to-one mapping nature of most indices. Nevertheless, our method is competitive for test time where planar is only twice as fast, due to the arc-based interpolation needed for spherical data, making spherical versions of semantic segmentation feasible for deployment.

Checking Kernel Bias: We check if the improved results in Table 9 are due to the hexagonal filter, rather than the spherical projection. In particular, using the method applied to pretraining in Section 6.2, we now apply a hexagonal kernel on a planar version of Cityscapes. Table 12 shows the results, where the hexagonal kernel consistently performs with slight reduced accuracy to standard 3×3 kernels. Thus we conclude, the hexagonal kernel is not the reason for improved results.

TABLE 11
Comparison of memory usage and computation time for training and inference on Cityscapes (level-10 and 1/3) with NVidia Titan X (Maxwell) using Pytorch v1.12. Batch size 1 for all cases. Run-time is reported per sample.

Method	Training		Inference	
	mem. (MB)	time (s)	mem. (MB)	time (s)
DANet	3,524	0.92	1,367	0.23
Sph-v1	9,110	11.47	2,889	0.40
Sph-v2	8,112	12.57	1,771	0.50
Sph-v3	5,358	10.36	1,847	0.43

TABLE 12
Ablation study for hexagonal kernel without spherical projection (hexagonal) on Cityscapes. Overall, hexagonal performs very comparable to planar, but consistently with slightly reduced accuracy.

Resolution	U-Net	DANet mIoU (%)	
	mIoU(%)	No Pretraining	Pretraining
planar@1/6	51.5	51.2	57.9
hexagonal@1/6	51.4	51.1	57.1
planar@l/3	55.5	63.0	67.0
hexagonal@1/3	55.2	62.9	66.9

8.5 Object Detection with Bounding Circles

TABLE 13
Synthia-O vehicle detection results, evaluated by average precision (AP). Resolution is given by W for ERP, and r for icosahedron methods.

Method	Resolution	Predictions	Average Precision
ERP	256	43,008	8.9
	512	172,032	46.2
	1024	688,128	52.8
HexNet	6	53,766	56.8
	7	215,046	67.6
spherePHD	6	107,520	39.8
	7	430,080	52.4

In this section, we apply HexNet to object detection following the YOLO-v3 framework [39]. Each node in the icosahedron mesh predicts an offset, a circle dimension (Section 7) and a detection confidence score. All methods predict at multiple scales, using feature pyramid networks [44]. Specifically, we append convolutional layers after each block of the feature extractor and predict the 3D tensor encoding of bounding circles (δ_x, δ_y, r), and detection confidence c. Similar to [39] we predict circles based on three anchor radii. The anchors' radius sizes is computed using k-means clustering on training data. We use ResNet-50 as our base feature extractor but using HexNet operations. Non-maximum suppression using (8) is employed to filter overlapping predictions. Again, we use Synthia-O with the same training and test split as in Section 8.4, and we extract car object ground truth from instance segmentation labels of SYNTHIA [19]. We keep every 5$^{\text{th}}$ frame to reduce temporal redundancy in the dataset. In total, our training and

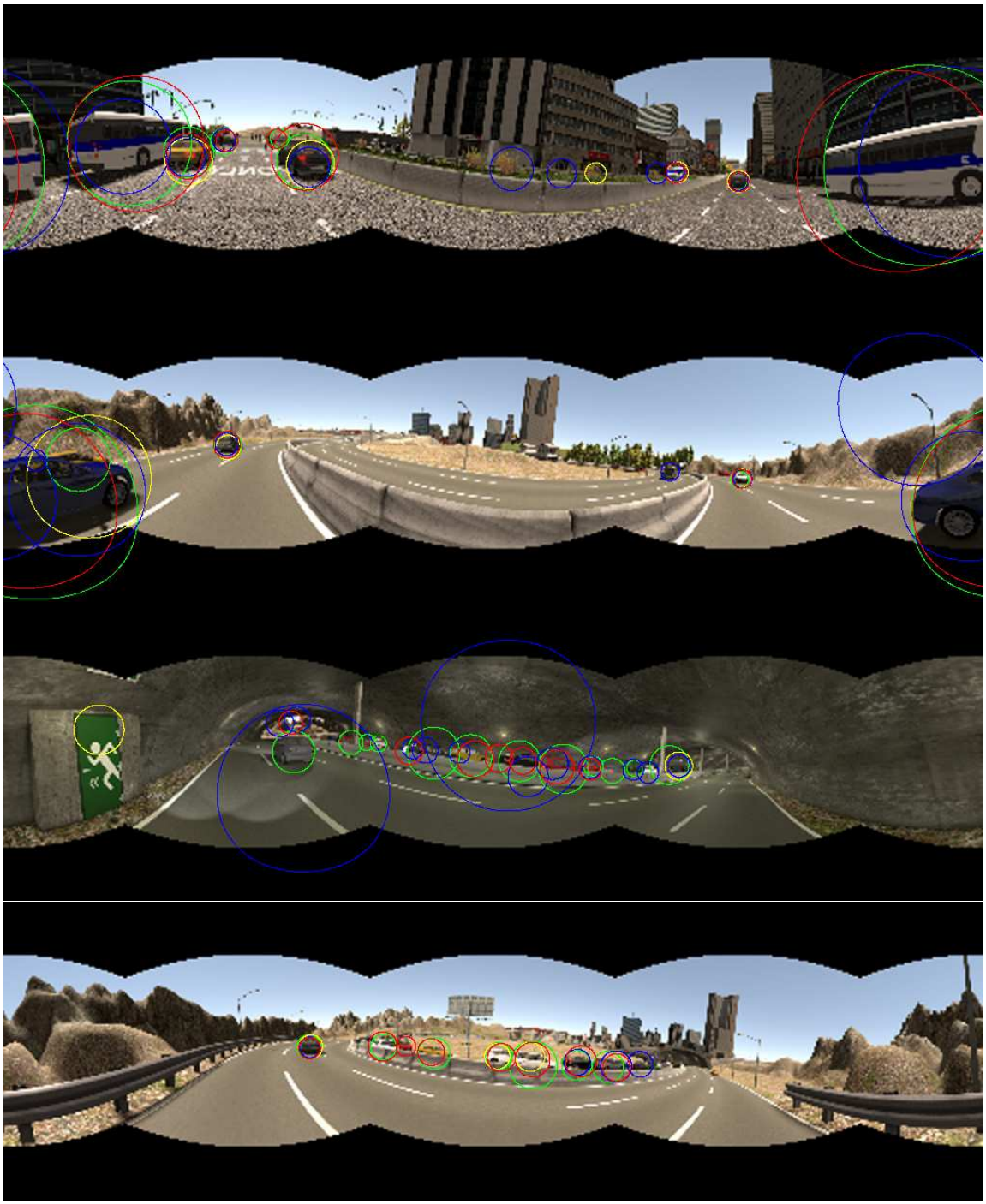


Fig. 22. Visualization of vehicle detection results on Synthia-O. Green represents ground-truth bounding circles, and red, yellow and blue denote HexNet (ours), ERP (equirectangular images), and spherePHD [11], respectively.

testing set has 354 and 102 images respectively. We compare HexNet results to planar YOLO-v3 [39] directly applied to equirectangular images (ERP), and spherePHD [11]. Note, since the number of YOLO predictions depends on the number of pixels/vertices, each method has different prediction quantitites: For equirectangular height H and width W, and icosahedron resolution r, we get $H * W$ predictions for ERP, $20 * 4^r$ for spherePhD, and $2 + 10 * 4^r$ for HexNet.[8] During training, all methods use per-pixel color jittering and left-right flipping for data augmentation. Performance is based on average precision (AP). The Network architecture is the same for all methods.

Table 13 shows the results. A higher resolution leads to better performance for all methods. Nevertheless, even though HexNet has a relatively small number of predictions available for its resolution, the best performances are reached. The distortions of ERP make object detection challenging, while spherePHD does not use axis aligned filters as kernels are mesh aligned and their shape differ

8. Numbers differ as HexNet uses icosahedron nodes as vertices, while SpherePhD uses faces.

TABLE 14
Evaluation of rotation invariance and equivariance for $\mathcal{SO}(2)$ rotations round the north-south axis, and random $\mathcal{SO}(3)$ rotations.

Task	Dataset	r	Original	$\mathcal{SO}(2)$	$\mathcal{SO}(3)$
Classification	MNIST	4	99.45	98.39	29.84
Segmentation	Synthia-O	7	49.5	47.9	22.4
Detection	Synthia-O	6	56.8	55.9	5.3

at each icosahedron face [11]. In particular, we see an AP of 67.6, while spherePHD reaches 55.6 and ERP only 50.8 at highest resolution level. Qualitative results are shown in Figure 22. While ERP struggles with discontinuities at the equirectangular image edge, and distortions, spherePHD fails to detect the smaller objects in the scene as the kernels are not as optimal at fine detail.

8.6 Rotation Equivariance

In this section, we conduct experiments on multiple tasks with horizontal rotations, and random rotations. Specifically, we consider three tasks: MNIST classification, urban semantic segmentation and vehicle detection as introduced above. For each task, the model trained on the original (non-rotated) training set is evaluated on two versions of the test set: horizontally rotated ($\mathcal{SO}(2)$ around the north-south axis) and randomly rotated in full $\mathcal{SO}(3)$. Table 14 shows that HexNet performs with similar accuracy for horizontal rotations in all experiments, while it fails for random rotations. This supports HexNet as horizontal rotation equivariant spherical convolution framework. We node, true rotation equivariance is achieved for horizontal rotations with equivalent quantization of the icosahedron (*i.e.* rotations with angles that are multiples of 72°).

9 Conclusion

We introduce HexNet, an orientation-aware deep learning framework for processing omnidirectional input data. Our method builds on common CNN operations, and can therefore operate efficiently on full spherical data. Furthermore, we introduce a partial equivalent method to overcome the resolution challenge for spherical images where only a partial field of view is used. Finally, weight-transfer and pretraining is introduced for HexNet which exploits commonly available planar datasets or pretrained network weights. In our experiments, we outperform alternative spherical networks in multiple orientation-aware experiments. Moreover, through reasoning of geometric distortion, we motivate improved accuracy for common planar estimation tasks, which we then confirm on popular planar datasets by exploiting our Masked-HexNet at high resolution. Our evaluation presents competitive performance of the proposed HexNet framework on 3 tasks and 5 public datasets.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR'15*, 2015, pp. 3431–3440.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI'15*, 2015, pp. 234–241.

[3] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR'17*, 2017, pp. 2881–2890.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[5] Y.-C. Su and K. Grauman, "Learning spherical convolution for fast features from 360 imagery," in *NIPS'17*, 2017, pp. 529–539.

[6] B. Coors, A. P. Condurache, and A. Geiger, "SphereNet: Learning spherical representations for detection and classification in omnidirectional images," in *ECCV'18*, 2018, pp. 518–533.

[7] 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.

[8] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR'17*, 2017, pp. 5115–5124.

[9] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *NIPS'16*, 2016, pp. 3189–3197.

[10] C. M. Jiang, J. Huang, K. Kashinath, Prabhat, P. Marcus, and M. Nießner, "Spherical CNNs on unstructured grids," in *ICLR'19*, 2019.

[11] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon, "Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9181–9189.

[12] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral CNN," *Int. Conf. Machine Learning, ICML'19*, 2019.

[13] M. Eder and J.-M. Frahm, "Convolutions on spherical images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 1–5.

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[15] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, "Salnet360: Saliency maps for omni-directional images with cnn," *Signal Processing: Image Communication*, 2018.

[16] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, and T.-L. Liu, "Cube padding for weakly-supervised salience prediction in 360° videos," in *CVPR'19*, 2019.

[17] M. Liu, F. Yao, C. Choi, S. Ayan, and K. Ramani, "Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution," in *ICLR'19*, 2019.

[18] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.

[19] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *CVPR'16*, 2016, pp. 3234–3243.

[20] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," in *ICLR'18*, 2018.

[21] M. Mudigonda, S. Kim, A. Mahesh, S. Kahou, K. Kashinath, D. Williams, V. Michalski, T. O'Brien, and M. Prabhat, "Segmenting and tracking extreme climate events using neural networks," in *Deep Learning for Physical Sciences Workshop, held with NIPS'17*, 2017.

[22] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[26] C. Zhang, S. He, and S. Liwicki, "A spherical approach to planar semantic segmentation," in *British Machine Vision Conference*, 2020.
[27] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, "Semantic-driven generation of hyperlapse from 360 degree video," *IEEE Trans. Visualization and Computer Graphics*, vol. 24, no. 9, pp. 2610–2621, 2018.
[28] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical cnns," in *ECCV'18*, 2018, pp. 54 – 70.
[29] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," in *CVPR'17*, 2017, pp. 4961–4970.
[30] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV'17*, 2017, pp. 764–773.
[31] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *CVPR'17*, 2017, pp. 4201–4209.
[32] Y.-C. Su and K. Grauman, "Kernel transformer networks for compact spherical convolution," *arXiv preprint arXiv:1812.03115*, 2018.
[33] T. S. Cohen and M. Welling, "Steerable CNNs," *arXiv preprint arXiv:1612.08498*, 2016.
[34] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen, "3d steerable cnns: Learning rotationally equivariant features in volumetric data," in *NIPS'18*, 2018, pp. 10 402–10 413.
[35] D. Worrall and G. Brostow, "Cubenet: Equivariance to 3d rotation and translation," *arXiv preprint arXiv:1804.04458*, 2018.
[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.
[37] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan 2015.
[38] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV'18*, 2014, pp. 818–833.
[39] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[40] J. Dupuy, E. Heitz, and L. Belcour, "A spherical cap preserving parameterization for spherical distributions," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[42] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4151–4160.
[43] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1377–1385.
[44] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

Chao Zhang received the PhD degree in computer science from the University of York, United Kingdom. During his PhD study, he worked on 3D statistical shape modelling and collaborated with researchers from University of Bonn, Germany and Max Planck Institute for Intelligent Systems, Tübingen. He is currently a research scientist at Toshiba Europe Ltd, Cambridge, United Kingdom. His research interests are in omni-directional signal processing and scene understanding.

Stephan Liwicki received the M.Phil. degree from the University of Cambridge, and the Ph.D. degree from Imperial College London. He then conducted his postdoctoral research at the University of Oxford in the topic of 3D reconstruction and online learning. In 2015 he joined Toshiba's Cambridge Research Laboratory as computer vision researcher, and he is now leading the Vision and Learning Group. Dr Liwicki has been recognized on multiple occasions for his abilities. Most notably, he received the prestigious Intel Fellowship, which honors the most promising computing and engineering Ph.D. students throughout Europe.

Sen He received the PhD degree in computer science from the University of Exeter, United Kingdom. He was a postdoctoral research fellow at University of Surrey from 2020 to 2022. He is currently a research scientist at Meta AI, London, United Kingdom. His research interests are in generative models for computer vision and deep learning.

William Smith received the BSc degree in computer science, and the PhD degree in computer vision from the University of York, United Kingdom. He is currently a Reader with the Department of Computer Science, University of York, United Kingdom. He holds a Royal Academy of Engineering/The Leverhulme Trust Senior Research Fellowship. His research interests are in shape and appearance modelling, model-based supervision and physics-based vision. He has published more than 100 papers in international conferences and journals.

Roberto Cipolla is Professor of Information Engineering at the University of Cambridge and a researcher in computer vision. He is also the Director of Toshiba's Cambridge Research Laboratory, Professor of Computer Vision at the Royal Academy of Arts Schools and a Fellow of the Royal Academy of Engineering. He has authored three books and co-authored over 400 articles in computer vision and related fields. He is known for his contributions to the reconstruction, registration and recognition of three-dimensional objects from images. In 2013, he was elected a Distinguished Fellow of the British Machine Vision Association; in 2020 he became a Fellow of the International Association for Pattern Recognition; and in 2022 he was elected a Fellow of the Royal Society.