A Spherical Approach to Planar Semantic Segmentation

Chao Zhang*,1
chao.zhang@crl.toshiba.co.uk
Sen He*,2
sh752@exeter.ac.uk
Stephan Liwicki1
stephan.liwicki@crl.toshiba.co.uk

1 Cambridge Research Laboratory
Toshiba Europe Limited
Cambridge, United Kingdom

2 Department of Computer Science
University of Exeter
Exeter, United Kingdom

Abstract

We investigate a geometrically motivated modification to semantic segmentation. In particular, we reformulate typical planar CNN as a projected spherical CNN where image distortions are reduced, and thus generalisation increased. Since prior formulations of spherical CNNs require computation on full spheres, fair comparison between planar and spherical methods have not been previously presented. In this work, we first extend spherical deep learning to support high-resolution images by exploiting the reduced field of view of classical images. Then, we employ our spherical representation to reduce distortion effects of standard deep learning systems. On typical benchmarks, we apply our spherical representation and consistently outperform the classical representation of multiple existing architectures. Additionally, we introduce direct spherical pretraining from planar datasets to further improve results. Finally, we compare our method on non-planar datasets, where we improve accuracy, and outperform running time of spherical state of the art for non-complete input spheres.

1 Introduction

We present a spherical approach to planar semantic segmentation. Specifically, motivated by our observation of geometrical distortions on the image plane, we propose to project images onto the manifold of a sphere using known camera intrinsic, and then apply a spherical convolutional neural network (CNN). We aim for a fair comparison between planar and spherical methods, as we present an in-place substitution without need for changing network architectures.

Research on CNN computations in the spherical domain include [1], which replaces translation invariance in planar space with rotation invariance in $SO(3)$ for spherical input. In [3], convolution filters are projected onto the tangent plane of the sphere. Recently, spherical CNNs adopt an icosahedron mesh (polyhedron with 20 faces) to compute convolutions efficiently [2, 6, 9, 12, 18]. Four low-order differential operators are utilized to approximate spherical convolutions in [9]. A rotation equivariant CNN for spheres is presented in [2], and [12] introduce SpherePHD for spherical object detection and semantic segmentation.

*Equal Contribution

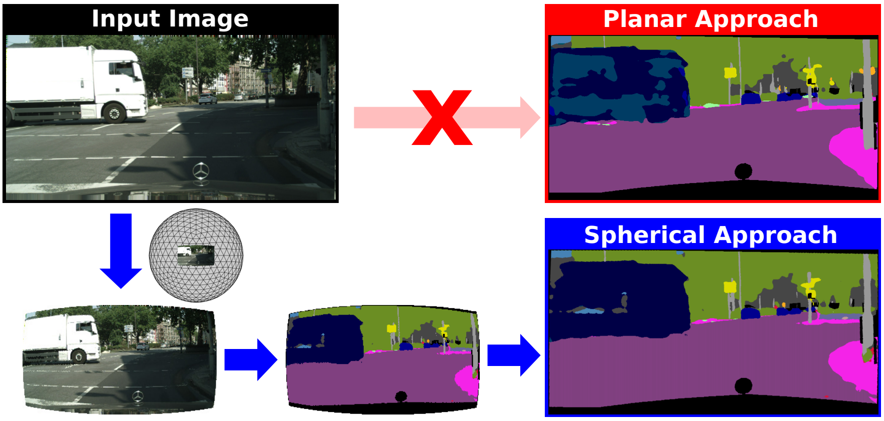


Figure 1: Image distortion is challenging when a planar image representation is used. After formulating spherical CNN as graph-based CNN for partial input, we propose an in-place substitution to work on the less distorted sphere manifold. Note, since geometric distortion is reduced, the 'truck' is correctly classified.

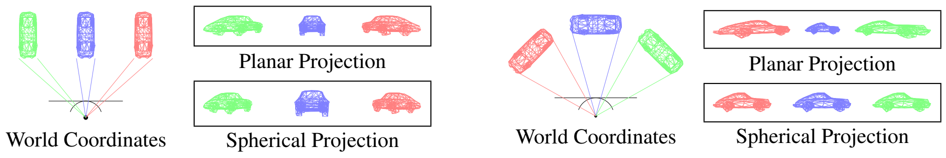
Orientation-aware convolutions on the icosahedron for efficient computation with regular filter alignment is introduced in [18]. Alternatively, [11] employ kernels on HEALPix spheres.

While most planar CNNs work on high-resolution input, existing spherical CNNs are not suitable for such data. Note, partial input is often not supported on spheres [2, 9, 12, 18]. In practical scenarios however, *e.g*. the driving environments of Cityscapes [4], the active view covers less than 3% of the sphere's manifold. Thus, 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. Meanwhile, processing the invalid region is computational inefficient and costly in memory.

1.1 Contributions

In this paper, we investigate the problem of distortion as we apply a spherical projection to high-resolution images for improved semantic segmentation results (Fig. 1). In particular, we first reformulate spherical CNN [18] 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 [7, 15] and common datasets [4, 16]. Finally, we introduce spherical pretraining with ImageNet [5], and further improve accuracy to a competitive level. Since our model support partial spheres not only for planar images, we conclude our evaluation with panoramic data. In summary, our contributions are:

1. We address the resolution problem of partial sphere images, and reformulate spherical CNN as graph-based CNN for arbitrarily masked input data.
2. We further improve run-time and memory requirements by introducing grouped convolutions and exploiting highly connected masks (*e.g*. images).



(a) Translation Distortion (b) Rotation Distortion

Figure 2: Planar projection suffers from translation and rotation distortion, while spherical projection reduces distortions at translation and removes them completely for rotation.

3. We implement the first direct spherical pretraining from planar datasets such as ImageNet [5] for ResNet-50 [8].
4. We perform extensive comparisons between planar and spherical projected images, using multiple datasets and network architectures, and we evaluate running-time in comparison to other state-of-the-art spherical CNNs on non-planar partial input.

2 Reduced Distortion on the Sphere's Manifold

An image consists of rays that represent 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 (Fig. 2 and supplementary material)

$$\|\mathtt{hom}(\mathbf{R}^\mathrm{T}\mathbf{P}_i) - \mathtt{hom}(\mathbf{R}^\mathrm{T}\mathbf{P}_j)\| \neq \|\mathbf{p}_i - \mathbf{p}_j\|, \text{ but } \left\| \frac{\mathbf{R}^\mathrm{T}\mathbf{P}}{\|\mathbf{R}^\mathrm{T}\mathbf{P}\|} - \frac{\mathbf{R}^\mathrm{T}\mathbf{Q}}{\|\mathbf{R}^\mathrm{T}\mathbf{Q}\|} \right\| = \|\mathbf{s}_i - \mathbf{s}_j\|. \tag{1}$$

Therefore we hypothesize that CNN learning on spherical images should be able to generalize to more pixel locations. As we strive for a conformation 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.

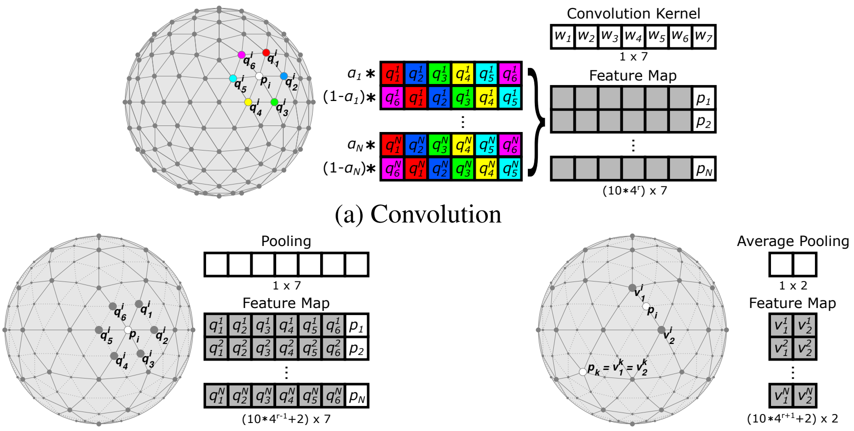
3 Our Spherical CNN for Partial Input

To date, icosahedron-based spherical CNNs cannot compete on high-resolution datasets [2, 6, 9, 12, 18]. We introduce a spherical CNN for partial input to overcome this shortfall.

3.1 Introducing a Graph-based Interpretation of Spherical CNN

Let us reformulate the icosahedron-based CNN in [18]. The base of the icosahedron mesh consists of 12 vertices, forming 20 triangular faces of equal size. A resolution increase is achieved by direct subdivision of the triangles. Thus, at resolution $r \geq 0$, there are $N^{(r)} = 10 \times 4^r + 2$ vertices on the icosahedron mesh. Note, resolution size requires $r \in \mathbb{N}$.

We denote the set of vertices at resolution r by $\mathcal{V}^{(r)} = \{p_i\}_{i=1}^{N^{(r)}}$, and the clock-wise sorted neighborhood of vertex p_i as $\mathcal{N}_i^{(r)} = [q_j^i]_{j=1}^{6}$, where $q_j^i \in \mathcal{V}$ (we omit r for simplified notation). We write $\mathcal{V}^{(r)} \subset \mathcal{V}^{(r+1)}$, since only new vertices are introduced when resolution is



(b) Pooling (c) Up-sampling

Figure 3: 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.

increased. The connectivity at different resolutions changes (*i.e.* $\mathcal{N}_i^{(r)} \neq \mathcal{N}_i^{(r+1)}$). We define convolutions, pooling and up-sampling below, and visualize the computations in Fig. 3.

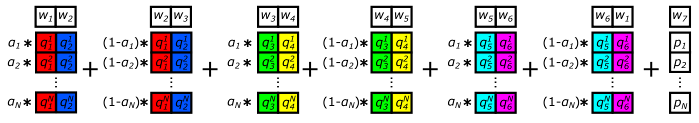
3.1.1 Oriented Convolution for Spherical CNN

All vertices that are not on the base icosahedron, *i.e.* $p_i \notin \mathcal{V}^{(0)}$, have a neighborhood cardinality of six. We increase the neighborhood of base vertices $p_i \in \mathcal{V}^{(0)}$ to six through duplication to simplify the representation. Now, we can apply 1×7 convolutions on the gathered neighborhood structure, an $N^{(r)}\times 7$ feature map where each row contains p_i and its neighborhood $\mathcal{N}_i^{(r)}$. [1] We emphasize, the application of convolutions on the icosahedron mesh directly will lead to orientation jumps of the kernel due to the triangular structure of the icosahedron mesh. Similar to [18], we apply arc-based interpolation to efficiently enforce consistent north-alignment of the convolution kernel. In particular, given the angle $\phi_i^{(r)}$ between the vectors $\overrightarrow{p_iq_1^i}$ and $\overrightarrow{p_ip_1}$, and $\psi_i^{(r)}$ between $\overrightarrow{p_iq_6^i}$ and $\overrightarrow{p_ip_1}$, where p_1 is the north pole, we find the precomputed interpolation weight $\alpha_i^{(r)}$ for each vertex that corrects the kernel to north-alignment. Thus the convolution with weights $[w_i]_{i=1}^7$ is given by (Fig. 3(a)):

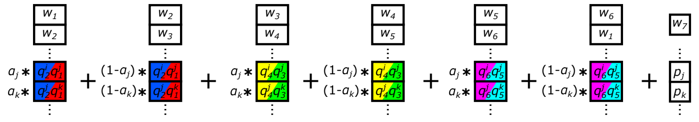
$$\texttt{conv}(p_i) = w_1\left(\alpha_i^{(r)}q_1^i + (1-\alpha_i^{(r)})q_6^i\right) + \sum_{j=2}^{6} w_j\left(\alpha_i^{(r)}q_j^i + (1-\alpha_i^{(r)})q_{j-1}^i\right) + w_7 p_i. \quad (2)$$

Note, the naïve implementation of (2) requires significant memory and is slow in running time. We address implementation details of masking and the execution in §3.2.

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



(a) Sequential Convolution



(b) Stacked Convolution

Figure 4: 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.

3.1.2 Pooling and Up-sampling on the Sphere

Since the pooling and up-sampling operations of [18] are not applicable to our graph structure, we redefine these 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 the 1×7 pooling (Fig. 3(b)). Note, any pooling operator can be used. Fig. 3(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.

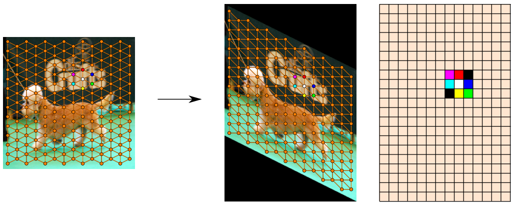
3.2 Masking the Active Areas of the Sphere

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 [2, 9, 18]. 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 $\mathcal{M}$.

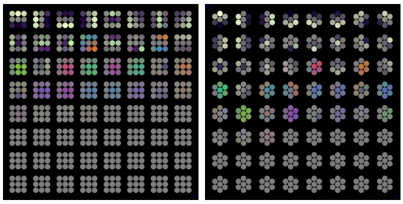
In a naïve implementation, two $M^{(r)} \times 6$ feature maps are gathered for each convolution (Fig. 3(a)). Since this is costly in memory and computation, we present an alternative approach. By rearranging the convolution weights, and the summations we only require three $M^{(r)} \times 2$ feature maps (Fig. 4(a) and supplementary). While the mask can be arbitrary, we can further reduce memory and run-time requirements, if vertices in the mask are highly connected (as is the case in image data). Specifically, we utilize 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 $g_j^{(r)} = \left[\begin{array}{cccccc} \dots & q_j^i & q_{j+1}^i = q_j^k & q_{j+1}^k & \dots \end{array}\right]$ of size $L_j^{(r)} \times 1$, where $M^{(r)} < L_j^{(r)} \ll 2M^{(r)}$ (Fig. 4(b) and supplementary). Now, a 1×2 kernel is applied on the $1 \times L_j^{(r)}$ feature map. Note, the ordering of $g_j^{(r)}$ is precomputed. Alg. 1 details the computations.

Algorithm 1: 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 $w_j \in \mathbb{R}^{1\times 1\times O}$, 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 $q_j^i, q_{j+1}^i$
    $G_j^a \leftarrow g_j \circledast [\; w_j \quad w_{j+1} \;]$                          // ⊛ computes convolution
    $G_j^b \leftarrow g_j \circledast [\; w_{j+1} \quad w_{((j+1) \mod 6)+1} \;]$            // interpolation ⇒ 2 convolutions
    $F_j^a \leftarrow$ 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
$F_7 \leftarrow [p_i]_{i=1}^{M^{(r)}} \circledast [w_7]$
$F \leftarrow [\alpha_i^{(r)}]_{i=1}^{M^{(r)}} \odot (F_1^a + F_3^a + F_5^a) + [1 - \alpha_i^{(r)}]_{i=1}^{M^{(r)}} \odot (F_1^b + F_3^b + F_5^b) + F_7$ // ⊙ is element-product
```



(a) Projected Sphere Kernel



(b) Planar and Spherical Filters

Figure 5: 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.

4 Spherical Pretraining from Planar Datasets

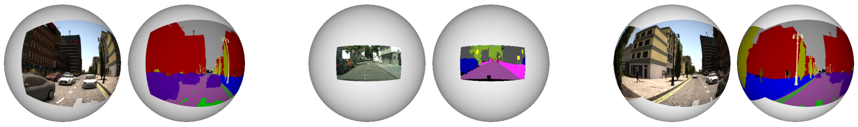
Pretraining leverages large datasets (*e.g.* ImageNet [5]) to provide improved initialization of common network parameters (*e.g.* ResNet-50 [8]). Unfortunately, however, spherical datasets are rare. In [18] weight transfer for spherical parameter initialization from standard 3x3 convolution kernels optimized on planar data is proposed. However, since the transfer is through interpolation, thus it is not accurate. We now present a direct training algorithm for spherical parameter initialization from planar data.

Since our kernels operate on the tangent plane of the sphere, planar equivalents can be found. We visualize our input data in (Fig. 5(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 intrinsic. Note, pretraining only needs to provide parameters a good initialization.

In our work, we utilize ResNet-50 [8]. Following [19], 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 Fig. 5(b) we visualize the filter weights of the initial layer. Note, our weights have similar properties to standard planar convolution layers [17].

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

Table 1: Datasets' training and test samples, the native resolution and coverage of sphere.



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

Figure 6: We use single view Synthia (a) and Cityscapes (b) to compare with planar alternatives, and omni-directional view (c) to compare with state-of-the-art spherical CNNs.

5 Evaluation and Ablation Studies

Motivated by recent works on spherical semantic segmentation, our main focus of this study is to investigate whether a spherical representation is beneficial even for planar images.

5.1 Datasets and Experimental Setup

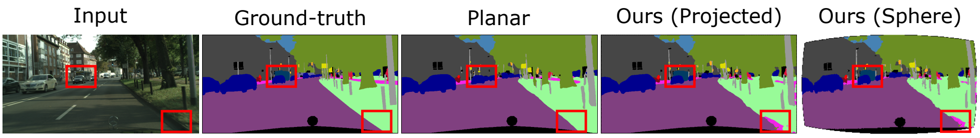
It is important to realize that we require camera calibration matrices to project onto the sphere. Therefore, two datasets with known camera intrinsic are used to compare to planar and spherical methods. We project planar images onto the sphere using bi-linear interpolation. All results are evaluated after back projecting into the planar domain at full resolution for fair comparison.[2]

SYNTHIA: The SYNTHIA dataset [16] contains photo-realistic synthetic sequences. It provides pixel-level semantics for 13 classes. In our experiments, we follow [18] and use 5 sequences in two setups: as **S**ingle-view or **O**mni-directional, denoted Synthia-S and Synthia-O respectively (Fig. 6(a) and (c)). In Synthia-S, images of different viewpoints are projected to the same place on the sphere, and treated as individual samples. In Synthia-O, we use the omni-directional images merged from 4 single views [18]. Both setups are interesting because they contain inactive areas. Further details are given in Table 1.

Cityscapes: We also evaluate on Cityscapes [4]. 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. Here, the coverage on the sphere is less than 3% (Table 1, Fig. 6(b)).

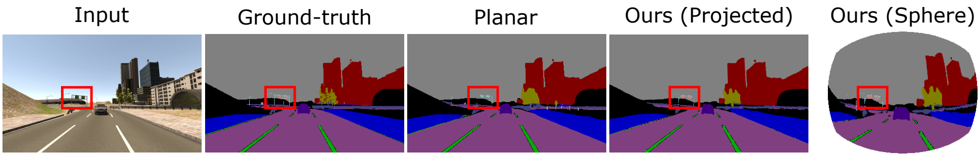
Our graph-based interpretation of spherical CNN provides essential building blocks for existing CNN architectures. We choose U-Net [15][3] and DANet [7][4] for evaluation. Specifically, we employ a residual U-Net which comprises a residual encoder and decoder branch [9, 18]. As for DANet, ResNet-50 [8] is adopted as feature extraction backbone, then dual attention blocks (spatial and channel) are employed to facilitate accurate segmentation [7]. In our implementation we follow [19], and replace the initial 7×7 convolution with two 3×3

[2]In fact, the evaluation is biased towards improved planar accuracy due to evaluation on planar ground truth.
[3]We reimplement original U-Net with residual blocks used in [9]
[4]Code available at https://github.com/junfu1115/DANet



🟪 road 🟪 sidewalk ⬛ building ⬛ wall 🟫 fence ⬜ pole 🟧 traffic light 🟨 traffic sign 🟩 vegetation 🟩 terrain
🟦 sky 🟥 person ⬛ building 🟥 rider 🟦 car 🟦 truck 🟦 bus 🟦 train 🟦 motorcycle 🟥 bicycle

Figure 7: 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.



🟥 building 🟪 car 🟦 cyclist 🟦 fence 🟩 marking ⬛ misc 🟫 pedestrian
🟨 pole 🟪 road 🟦 sidewalk 🟫 sign ⬜ sky 🟩 vegetation

Figure 8: 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.

convolutions, or more specifically our hexagonal kernel with 1-ring neighborhood. Note, one limitation of our spherical convolutions is that only an 1-ring neighborhood is supported. In all experiments, we use Adam optimizer [10] 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.

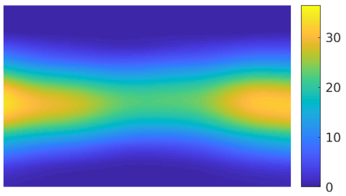
5.2 Spherical Projection versus Planar Images

We study different input size as we match spherical resolution to similar planar equivalents (Table 2). 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.

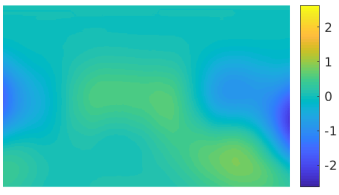
In Table 2 we report mean intersection over union (mIoU) for the residual U-Net and the DANet architecture. First, we discuss results without pretraining. Our spherical representation consistently achieves improved results over the standard planar version on both datasets. We also note, both methods improve at higher resolution. Therefore we conclude, it is necessary to develop spherical CNN methods that support high-resolution data.

DANet employs ResNet-50 for its feature extraction. We now compare spherical and planar pretraining. In Table 2, 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.[5] We believe, since our pretraining utilizes ImageNet data with planar images,

[5]Cityscapes accuracy @ 1/2 and @ 1/4 is 71.8% [14] and 59.1% [13], hence 67.8% @ lv-10 is competitive.



(a) Training Distribution



(b) $TP_s(i) - TP_p(i)$

Figure 9: Training data location bias (a) and results difference (b) between spherical and planar method for 'pedestrian' in Synthia-S.

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

Table 2: 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. Pretraining with ImageNet [5] is additionally reported for DANet.

improvements are slightly reduced. A spherical version of ImageNet with camera calibration matrices may be beneficial in future work. In Fig. 7 and Fig. 8, qualitative results are given. We observe, 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.* $TP_s(i) - TP_p(i)$. Fig. 9 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 §2, where we suggest that fewer distortion aids generalization.

5.3 Comparing Running Time Efficiency

In Table 3, we compare our implementation of Alg. 1 (Sph-v3) with naïve convolutions in Fig. 3(a) (Sph-v1) and sequential version in Fig. 4(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. 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.

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 3: Comparison of memory usage and computation time for training and testing 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	mIoU (%)	Memory (MB)	Time (s)
UGSCNN[9]	37.6	6,831	1.52
HexRUNet [18]	48.3	3,596	0.16
Our (Full)	49.5	1,097	0.19
Our (Partial)	**50.1**	**595**	**0.07**

Table 4: Synthia-O results for our method, as full and partial input, compared to HexRUNet and UGSCNN. Inference memory and run-time is given for batch size 1, per sample.

5.4 Our Graph-based CNN with Partial Input versus Spherical CNN

In this experiment, we compare our implementation of spherical CNN with HexRUNet [18][6] and UGSCNN [9][7] using omni-directional Synthia-O. Similar to HexRUNet and UGSCNN, we employ the residual U-Net architecture in this section. Table 4 shows accuracy, memory and run-time at mesh level-7. Overall, our partial implementation performs best, since only active areas are used for computations.

6 Conclusion

In this work, we set out to investigate the benefit of the spherical representation of planar images for the semantic segmentation task with high-resolution datasets. Specifically, we propose to employ spherical CNNs to reduce distortions otherwise introduced by the image plane. In our approach, we overcome memory and run-time limitations of state-of-the-art icosahedron-based CNNs, by introducing an equivalent graph-based networks that allow for arbitrary input masks. Furthermore, exploiting highly connected masks of typical image projections, we present optimisation for improved memory needs and speed. Finally, we present spherical pretraining from planar ImageNet for our icosahedron-based convolutions.

In our evaluation, we show that a simple introduction of the spherical representation consistently improves upon planar results. After applying pretraining from planar ImageNet, we achieve competitive results to planar state of the art (even often improved results). Finally, we show improved run-time, memory and performance to existing spherical state of the art, when partial input is used.

Acknowledgements. Sen He was an intern at the Cambridge Research Laboratory of Toshiba Europe Limited during this work.

[6]Code kindly provided by the authors

[7]Code available at https://github.com/maxjiang93/ugscnn

References

[1] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Convolutional networks for spherical signals. In *ICLR*, 2018.

[2] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1321–1330, 2019.

[3] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 518–533, 2018.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[6] Marc Eder and Jan-Michael Frahm. Convolutions on spherical images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–5, 2019.

[7] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

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

[9] Chiyu Jiang, Jingwei Huang, Karthik Kashinath, Philip Marcus, Matthias Niessner, et al. Spherical cnns on unstructured grids. In *ICLR*, 2019.

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

[11] Nicoletta Krachmalnicoff and Maurizio Tomasi. Convolutional neural networks on the healpix sphere: a pixel-based algorithm and its application to cmb data analysis. *Astronomy & Astrophysics*, 628:A129, 2019.

[12] Yeonkun Lee, Jaeseok Jeong, Jongseob Yun, Wonjune Cho, and Kuk-Jin Yoon. Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9181–9189, 2019.

[13] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE international conference on computer vision*, pages 1377–1385, 2015.

[14] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4151–4160, 2017.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[16] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

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

[19] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.