Supplementary Material for SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images

Benjamin Coors[1,3], Alexandru Paul Condurache[2,3], and Andreas Geiger[1]

[1] Autonomous Vision Group, MPI for Intelligent Systems and University of Tübingen
[2] Institute for Signal Processing, University of Lübeck
[3] Robert Bosch GmbH

Abstract. In this supplementary document, we present a visualization of the sampling error made by regular convolutional filters on omnidirectional images. Next, we provide details on the Spherical Transformer Network, which is able to transform inputs to a canonical orientation in support of a recognition model. Furthermore, we present additional experiments on the impact of varying object scale, object elevation and choice of interpolation on the Omni-MNIST classification task. Finally, we give a qualitative comparison of the CNN and SphereNet transfer learning models on the omnidirectional parked cars (OmPaCa) datasets.

1 Sampling Error

When applying a regular convolutional neural network to omnidirectional images, the normalized average absolute geodesic error is relatively small close to the equator region but grows large towards the poles when compared to the optimal sampling locations used by SphereNet (see Fig. 1).

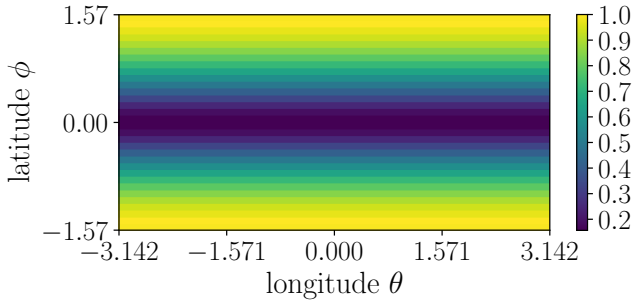


Fig. 1: **Average Geodesic Error** of the sampling locations of a regular convolution kernel applied to the equirectangular image representation. Note how the error increases towards the poles $\phi \in \{-\frac{\pi}{2}, +\frac{\pi}{2}\}$.

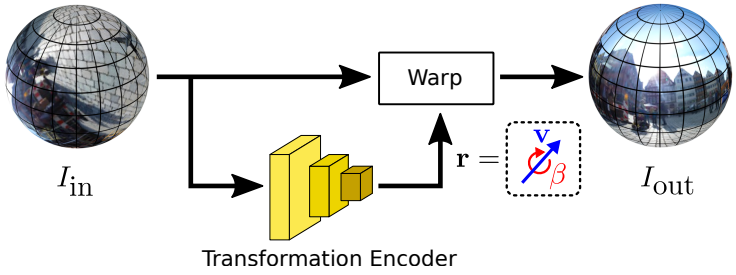


Fig. 2: **Spherical Transformer Network.** A transformation encoder network outputs the rotation parameters $\mathbf{r}$ specifying the axis $\mathbf{v}$ and angle β of rotation. The input I_{in} is warped to the output I_{out} via bilinear interpolation according to $\mathbf{r}$.

2 Spherical Transformer Network

As a baseline for the classification and detection tasks, we investigate the use of a *Spherical Transformer Network (SphereTN)*, which aims to undistort objects by performing a global rotation of the spherical image conditioned on the input image. Compared to SphereNet, the Spherical Transformer needs to learn how to undistort an object. Additionally, it only performs a single global transformation of the input and may therefore be unable to undistort multiple objects at once.

Similar to a Spatial Transformer Network (STN) [2], the Spherical Transformer Network uses a localization network to predict a set of transformation parameters which are used to resample the input image. However, unlike a Spatial Transformer which typically outputs an affine transformation, the Spherical Transformer Network predicts and applies a 3D rotation of the spherical image representation.

In order to avoid the gimbal lock problem, we do not represent the axial rotations using Euler angles but instead leverage the axis-angle representation. More formally, we rotate 3D points $\mathbf{y} \in \mathbb{R}^3$ located on the surface of the unit sphere S by applying a rotation $\mathbf{y}' = \mathbf{R}\mathbf{y}$ where the rotation matrix $\mathbf{R} \in SO(3)$ is given as follows:

$$\mathbf{R} = \mathbf{I} + (\sin\beta)[\mathbf{v}]_\times + (1 - \cos\beta)[\mathbf{v}]_\times^2 \tag{1}$$

Here, $\mathbf{v} \in \mathbb{R}^3$ is a unit vector defining the rotation axis and β denotes the rotation angle. $[\mathbf{v}]_\times$ denotes the cross-product matrix and is an element of the Lie algebra $so(3)$.

As $\mathbf{v}$ is a unit vector with two degrees of freedom, we are able to encode the rotational component β as its length. More specifically, we parameterize the rotation axis $\mathbf{v}$ and rotation angle β with a 3-dimensional vector $\mathbf{r} = \beta \cdot \mathbf{v}$ which is the output of the localization network conditioned on the input image. Note that after prediction, $\mathbf{r}$ can be easily decomposed into $\beta = \|\mathbf{r}\|_2$ and $\mathbf{v} = \mathbf{r}/\beta$. The predicted transformation is applied to the omnidirectional image via differentiable image sampling with bilinear interpolation [2]. See Fig. 2 for an illustration.

Table 1: **Digit Scale Evaluation on Omni-MNIST.** Performance comparison on the omnidirectional MNIST dataset for varying digit sizes.

Method	Large	Medium	Small
GCNN [3]	17.21	20.35	28.54
S2CNN [1]	11.86	19.90	38.80
CubeMapCNN	10.03	11.37	24.46
EquirectCNN	9.61	9.10	8.60
EquirectCNN+SphereTN	8.22	8.69	8.46
SphereNet (Uniform)	7.16	6.91	8.77
SphereNet (NN)	7.03	6.32	7.51
SphereNet (BI)	**5.59**	**5.03**	**5.89**

Table 2: **Digit Elevation Evaluation on Omni-MNIST.** Performance comparison on the omnidirectional MNIST dataset for varying digit elevation.

Method	$\lvert\phi_d\rvert \in [0, \frac{\pi}{8}]$	$\lvert\phi_d\rvert \in [\frac{\pi}{8}, \frac{\pi}{4}]$	$\lvert\phi_d\rvert \in [\frac{\pi}{4}, \frac{3\pi}{8}]$	$\lvert\phi_d\rvert \in [\frac{3\pi}{8}, \frac{\pi}{2}]$
GCNN [3]	17.48	19.48	18.39	20.63
S2CNN [1]	11.63	11.45	11.41	11.49
CubeMapCNN	8.70	8.70	9.10	13.80
EquirectCNN	7.02	8.64	9.17	11.34
EquirectCNN+SphereTN	6.46	7.76	7.74	11.08
SphereNet (Uniform)	6.25	6.87	7.18	9.14
SphereNet (NN)	6.15	6.36	6.23	8.67
SphereNet (BI)	**4.50**	**4.87**	**4.89**	**6.84**

Table 3: **Interpolation Evaluation for SphereNet on Omni-MNIST.** Performance comparison of SphereNet when varying the location of the bilinear interpolation (BI) layers.

Bi-layers	$none$	$conv_1$	$conv_2$	$pool_1$	$pool_2$	$conv_{1,2}$	$pool_{1,2}$	all
Test error	7.03	6.31	6.25	6.47	6.25	6.18	6.17	**5.59**

3 Omni-MNIST Analysis

In order to analyze the image classification results on the Omni-MNIST dataset more closely, we conduct further studies. First, we test the effect of varying the digit scale on the performance of the different models. To do so, we generate the Omni-MNIST

dataset at three different scales (small, medium, large). We then train and evaluate each model on each of the three variants. The results are shown in Table 1 and demonstrate that the performance of the nearest neighbor (NN) and bilinear interpolation (BI) variants of SphereNet are not significantly impacted by changes in digit scale. On the other hand, the uniformly sampled variant (Uniform) of SphereNet drops in performance for smaller digits, as in this case important information is lost at a smaller object scale. In contract, the EquirectCNN baseline performs slightly better for small digit scales, as smaller digits minimize the amount of object distortion in the equirectangular image. The CubeMapCNN, S2CNN and GCNN baselines all show significantly degraded performance for smaller digit sizes, with the S2CNN model particularly struggling with the classification of digits of smaller size.

Second, we evaluate the performance of all models for different ranges of digit elevation ϕ_d (see Table 2). While CubeMapCNN and EquirectCNN models perform gradually worse with increasing elevation and object distortion, the SphereNet variants offer near constant performance for elevations $|\phi_d| \leq \frac{3\pi}{8}$. When further investigating the decrease in SphereNet's performance for elevations $|\phi_d| \in [\frac{3\pi}{8}, \frac{\pi}{2}]$, we find it to be caused by a sudden drop in performance to nearly 20% test error at the poles ($|\phi_d| = \frac{\pi}{2}$). The reason for SphereNet's loss in performance at this elevation is that its assumption of an upright object orientation no longer holds at the poles. Unlike SphereNet, S2CNN encodes full rotation equivariance and therefore is the only baseline with near-constant performance over all digits elevations.

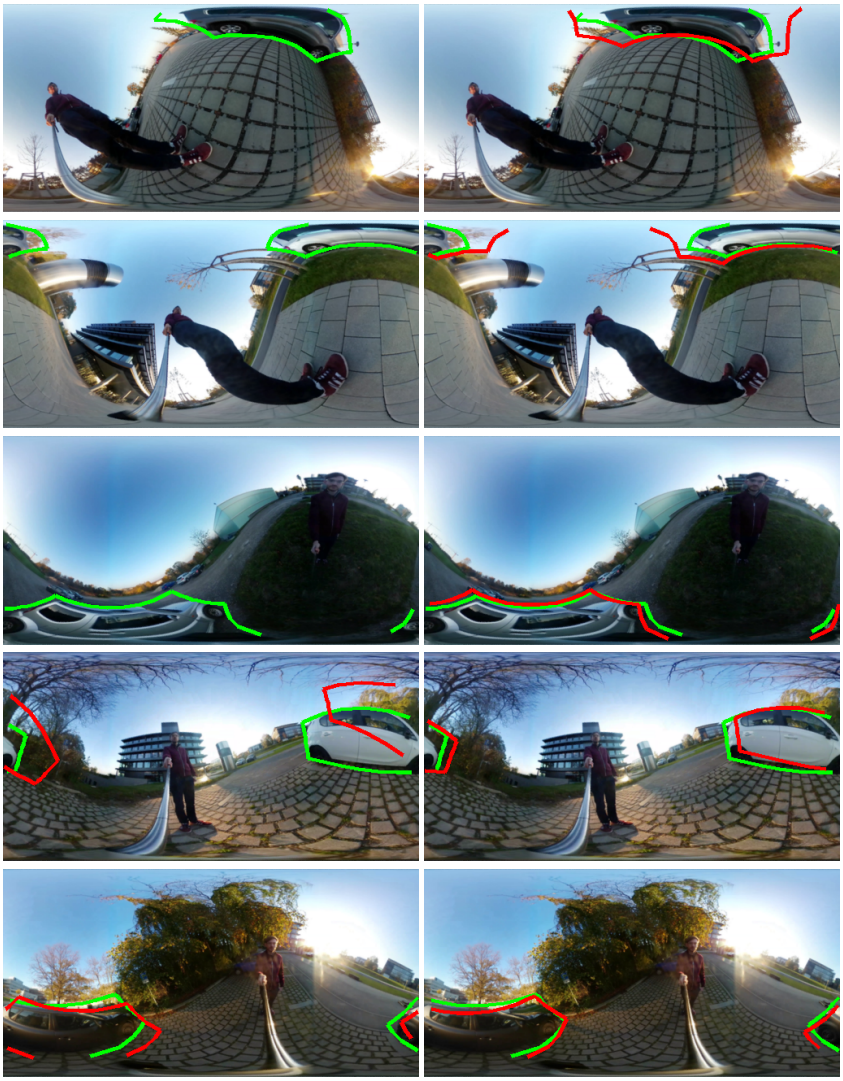
Finally, we evaluate which network layers in the SphereNet model benefit most from the use of bilinear interpolation (BI) over a nearest neighbor interpolation (NN). Table 3 indicates that all layers benefit from bilinear interpolation with the benefit being slightly larger in the second layers and increasing further when both convolutional or pooling layers utilize bilinear interpolation.

4 OmPaCa Detection Comparison

In order to visualize the performance benefit of a SphereNet model compared to the EquirectCNN baseline on the omnidirectional parked cars (OmPaCa) detection task, we provide a qualitative comparison of the detection results in Fig. 3.

References

1. Cohen, T.S., Geiger, M., Köhler, J., Welling, M.: Spherical CNNs. In: International Conference on Learning Representations (2018) 3
2. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (NIPS) (2015) 2
3. Khasanova, R., Frossard, P.: Graph-based classification of omnidirectional images. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops (2017) 3



(a) EquirectCNN (b) SphereNet (NN)

Fig. 3: **Qualitative Performance Comparison** between EquirectCNN and SphereNet (NN) model on the OmPaCa dataset. The ground truth is shown in green, detections are shown in red. Unlike SphereNet, the baseline EquirectCNN model struggles to detect objects in the polar regions of omnidirectional images (row $1-3$) and, in general, outputs less tight bounding boxes (row $4-5$).