This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Sensor Equivariance: A Framework for Semantic Segmentation with Diverse Camera Models

Hannes Reichert, Manuel Hetzel, Andreas Hubert, Konrad Doll
University of Applied Sciences Aschaffenburg
Germany
{firstname.lastname}@th-ab.de

Bernhard Sick
University of Kassel
Germany
bsick@uni-kassel.de

Abstract

Objects are represented differently in projection-based sensors such as cameras depending on sensor resolution, field of view, and distortion, leading to distorted physical and geometric properties. As a result, sensor data processing depend on these properties. With the large variations of sensors on the market, an equivariant representation and suitable processing are necessary to become independent of the sensor used. In this work, we propose an extension of conventional image data by an additional channel in which the associated projection properties are encoded. Furthermore, we introduce a SensorConv layer as an extension to the conventional convolution layer. SensorConv enable using projection properties in convolutional neural networks. To that end, we propose an architecture for using the SensorConv layer in the Detectron2 [21] framework. We collected a dataset of equirectangular images for our experiments with the CARLA [3] simulator. To analyze multiple sensor models (i.e., sensor intrinsic), we created an augmentation method to emulate a high variability of sensors from the collected equirectangular panoramas. In our experiment, we show that our method can generalize better across different camera sensors.

1. Introduction

This work is motivated by the fact that a wide variety of sensors, all with different characteristics such as field of view, resolution, and lens distortion, are integrated into a large amount of products. The functionalities and capabilities of these products heavily rely on the sensors used. This means that new sensors will be continuously integrated into new products as they become available. Transferring a machine learning model for, e.g. semantic segmentation created on the data of a single sensor to another sensor is not trivial. In this work, we focus primarily on the transfer to sensors with other geometric properties (e.g., field of view, resolution, mounting angle) and do not consider other influences like the mounting position of a sensor. An important application for machine learning is scene understanding with autonomous vehicles.

In autonomous driving, perception modules usually consist of data-driven models based on sensor data. However, these models might be biased toward the sensor used for data acquisition. This bias can seriously impair the perception model's transferability to new sensor setups, which continuously occur due to the market's competitive nature.

For example, the industry uses fish-eye or wide-angle cameras in modern vehicles. However, in the research community, modern perception algorithms are developed and tested on datasets like Cityscapes [1], KITTI [5], etc., which use perspective cameras. This leads to a discrepancy between academic research and industrial development.

1.1. Definition: Sensor Equivariance

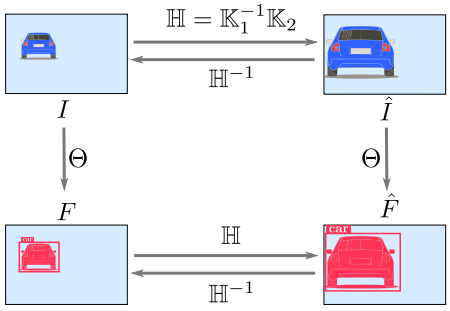

Figure 1. Concept of sensor equivariance.

As shown in Fig. 1, we want feature extractor Θ to work robustly, regardless of the camera used. Transformations $\mathbb{H}$ are suitable methods to simulate versatile data concerning the sensor used. $\mathbb{H}$ can be used to transform an image I with intriniscs $\mathbb{K}_1$, or its features F, into the representation of a novel sensor $\hat{I}$, with intrinsic $\mathbb{K}_2$, or $\hat{F}$, respectively. Note that we use the linear matrix notation $\mathbb{H}$ to formulate

a transformation for simplicity. However, $\mathbb{H}$ can also contain the transformation of nonlinear intrinsic. If a method Θ achieves equivariance, the extracted features F should also be transformable to novel sensors, making Θ symmetric and equivariant to transformations regarding combinations of shifting, rotations, and scaling. Generally speaking, a scale equivariance for a function Θ is always desirable, since it can extract useful features in many computer vision applications. Equivariances in shift and rotation are also important. However, they are usually solved in standard methods for Θ (e.g., convolutions or convolutional layers in neural networks). Therefore, we prioritize scale equivariance in our work as important for many computer vision tasks, from object detection and segmentation to monocular 3D perception.

1.2. Related Work

For image data, there is some preliminary work, addressing the sensor equivariance formulation stated in subsection 1.1: Liu et al. introduce *CoordConv* [13], as a solution for the coordinate transform problem. The convolution itself is spatial invariant. However, for some tasks, spatial variance might be needed. In object detection, for example, the coordinate transform problem arises from processing features in pixel space and output bounding boxes in Cartesian space. A *CoordConv* layer is a simple extension of the standard convolutional layer. It has the same functional signature as a convolutional layer, but utilizes extra channels for the incoming representation. These channels contain hard-coded coordinates, the most basic version of which is one channel for the u coordinate and one for the v coordinate. The authors claim that the *CoordConv* layer keeps the properties of few parameters and efficient computation from convolutions but allows the network to learn if spatial variance or invariance is needed for learning the task. This is useful for coordinating transform-based tasks where regular convolutions can fail. However, when changing an image's FOV or resolution, the u and v coordinates should change accordingly and be reversible. This is not the case for *CoordConv* and, therefore, disagrees with the equivariance condition formulated in subsection 1.1.

Wang et al. [20][19] utilize the *CoordConvs* concept as a component in an instance segmentation framework. The authors argue that spatially variant convolutions are necessary for instance segmentation. Furthermore, they concluded that few *CoordConvs* layers within the backbone are enough to achieve this.

Facil et al. teach camera-aware multi-scale convolutions (*CamConv*) for depth estimation from image data [4] supplied to a neural network. The method extends u- and v-maps of the *CoordConv* by horizontal and vertical field-of-view maps fov_h and fov_v, and center point shift maps cc_x and cc_y. These maps are supplied to the neural network with different resolutions and on different layers to allow the network to learn and predict depth patterns that depend on the camera calibration. The authors conclude that the neural network supplied with the respective maps can generalize over camera intrinsics and allow depth prediction networks to be camera-independent. This work can be seen as the work that is most closely related to our approach. Even if this method can be adapted for image data in general, it utilizes four additional channels exclusively designed for camera sensors. *CamConv* are usually used in depth estimation tasks. Their impact on tasks like object detection or segmentation has not been analyzed yet. *CamConv* are used in [9].

However, *CamConv* are designed for pinhole camera models. They are incapable of modeling distortion, which appears in fish-eye and wide-angle cameras to address the requirements stated in subsection 1.1. *CamConv* must be extended to a unified representation suitable for various sensor models.

In [14], *CamConv* are extended to fish-eye cameras by using the unified camera model (UCM) [18]. The extension to the UCM is done by using fov_h and fov_v from *CamConv* and distorting them based on the distortion parameters of the camera model used. In addition, two channels of normalized coordinates n_x and n_y are introduced, whose values span linearly with respect to the image coordinates between -1 and 1. In an experiment based on the WoodScapes dataset [23], the authors show that their approach is capable of generalizing over the fish-eye cameras used in the WoodScape setup for self-supervised distance estimation. However, the four fish-eye cameras used in WoodScapes have very similar intrinsic parameters. Therefore, no statement can be made about how well the approach generalizes across different sensors with different intrinsic parameters.

In [22], the authors propose the encoding of camera parameters as images of Fourier features for geometrical embeddings. The authors claim that Fourier features, as a higher-dimensional encoding, are better suited for further processing by neural networks. The benefits of this approach were demonstrated in [6] for the task of zero-shot monocular depth estimation. The features use camera centers and viewing ray directions. A pinhole camera model parameterizes both. Thus, the method is designed for pinhole cameras and, without modifications, is not capable of handling fish-eye or wide-angle cameras. Fourier features massively increase the channels used to encode the geometric sensor properties. With $3(F+1)$ channels required, F stands for the number of Fourier bands used.

1.3. Main Contributions

This work proposes a deflection metric for encoding projection properties to an image representation based on a pro-

jection model of a sensor. The deflection metric is a one-channel image that can resolve ambiguities in the projection and homogenize the data from various sensors. We propose a *SensorConv*-Layer for combined processing of the deflection metric and projected sensor data. We suggest using equirectangular images and propose an augmentation method to emulate various sensor models. Furthermore, we propose an experiment with statistical evaluation to analyze the transferability to novel sensors, which we use to confirm the usability of our method on sensors with different resolutions, fields of view, and lens distortions. For our experimental setup, we use the CARLA [3] simulator for data generation. We publish the code for our data generation pipeline used to acquire equirectangular images (see code).

2. Method

An image reflects reality ambiguously. For example, by purely looking at an image, the scale of an object cannot be determined. This ambiguity in scale and distortion of objects poses a major problem with the training of CNNs. We aim to resolve these ambiguities with the proposed method by adding knowledge, what we call deflection metric, about the sensor used. In the following, we first define a deflection metric, which encodes the projection properties of a sensor such that it can be interpreted or processed by a CNN (see subsection 2.1). Second, we show how the deflection metric is provided to a state-of-the-art semantic segmentation architecture (see subsection 2.3).

2.1. Deflection Metric

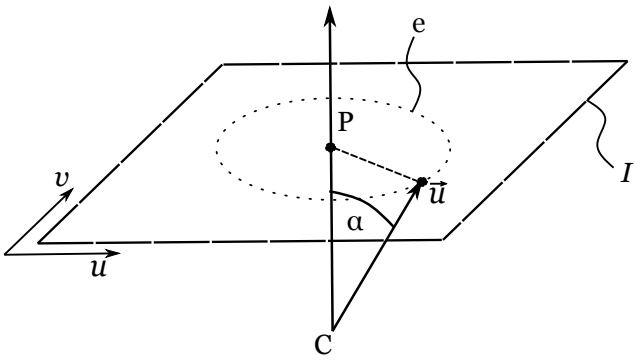

Figure 2. **Deflection Metric** α: Calculated as an angle between the optical axis, defined by a sensor's origin C and a principal point P, and a pixel $\vec{u} = [u, v, 1]^T$ [15].

With the deflection metric, as shown in Figure 2, we propose a method to encode the geometric characteristics of a sensor alongside its data. We use a deflection image I_α, which encodes an inclination angle α in every pixel as a one-channel image. α provides a consistent relation between a projected 3D point and the position of this 3D point in relation to the sensor and is, therefore, compatible among sensors with different characteristics. The deflection metric α is an angle between the optical axis, defined by a sensor's origin C and a principal point P, and a pixel $\vec{u} = [u, v, 1]^T$. The deflection metric α for a pixel position $\vec{u}$ can be determined based on a parameterized sensor model. With $\mathbb{K}$ as a sensor intrinsic matrix. The calculation is shown here exemplary for the pinhole camera model. Based on the image coordinates, α of a pixel $\vec{u}$ with respect to the projection center P can be determined:

$$\alpha(\vec{u}) = \arctan\left(\sqrt{\underbrace{\left[\mathbb{K}^{-1}\vec{u}\right]^T\left[\mathbb{K}^{-1}\vec{u}\right] - 1}_{d(\vec{u})}}\right) \quad (1)$$

All the same values of the deflection metric α are arranged on a contour e (see dashed circle in Figure 2). The deflection image I_α is a one-channel image in which every pixel $\vec{u}$ is aligned with the data image I. This increases the information content of each pixel by the geometric sensor properties. The deflection image I_α can be processed with the sensor data by convolutional layers of a CNN. The deflection image is not invariant to translation, rotation, and scale. However, since the convolutional layers of CNNs are learned, a CNN can decide whether to utilize this additional information in the learning process.

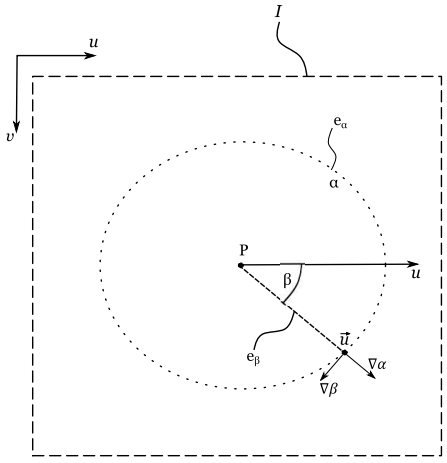

Figure 3. Definition of a rotation angle β form I_α.

Using simple convolution filters, a angle β can be determined from α. From I_α and the curvature of α in its pixel neighborhood I_β and β can be derived by building the gradient direction $\nabla\alpha$, as shown in Figure 3. This can be done utilizing simple convolution operations, i. e. Scharr [17]. The Scharr filter might be beneficial due to numerical precision and rotational symmetry. Since these are convolutional filters, a CNN can learn the calculation of I_β and β implic-

itly, if needed. α and β together uniquely define a viewing ray. With a range measure r (e.g., from a LiDAR sensor or implicitly learned from geometric priors), a unique metric 3D point can be defined. Thus, form the deflection image I_α and a range image I_r an 3D measure can be defined.

2.2. SensorConv Layer

Since many computer vision methods are based on convolution operations, we propose using the *SensorConv* Layer (see Fig. 4). *SensorConv* can enforce a sensor equivariant processing and resolve ambiguities by extending convolution layers with projection properties of the deflection metric α. Similar to *CamConv* and *CoordConvs*, *SensorConv* are additional inputs. A learnable convolutional layer can decide if the additional information is applicable or can be discarded. Hence, it does not break the conventional properties of convolution operations.

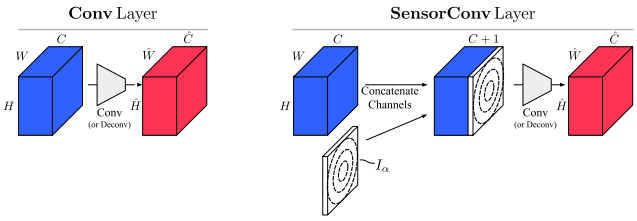

Figure 4. **SensorConv** Layer.

2.3. Backbone Architecture

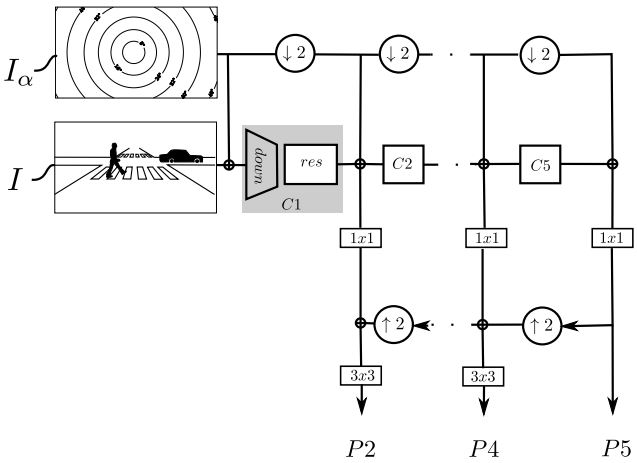

Figure 5. **Backbone Architecture**: The deflection metric is injected before every pyramid stage. $\downarrow 2$ denotes a down-sample operation, $\uparrow 2$ denotes an up-sample operation, $\bigoplus$ denotes a channel wise concatenation.

An existing backbone meta-architecture is modified to inject the deflection metric into the model at the input and selected locations. Based on the findings in [20][19], as described in subsection 1.2, we decided to use a ResNet50-FPN [7, 12]. The modification to the ResNet-FPN is shown in Figure 5. At the input stage, the deflection image I_α is concatenated with a three-channel image I, resulting in an input shape of $h \times w \times 4$. The image input is processed top-down in five down-sampling stages. Each stage halves the height h and width w. This is done for the image data using strided convolutions, followed by a residual block ($C1$ to $C5$). The deflection image I_α is down-sampled in parallel and concatenated to the features of the stages $C1$ to $C5$. This ensures that the feature map can be used in every stage. After the injection, a 1×1 convolution fuses the stage features with the deflection metric. This allows the network to keep or discard the deflection metric for a particular stage. The feature maps are up-sampled from the bottom up prior to a fusion with the pristine feature map from the respective stages. The fusion is performed by channel-wise concatenating the feature maps and a subsequent 3×3-convolution for anti-aliasing, as with common FPN architectures. This results in pyramid stages P_i with the respective shapes $(h/2^i) \times (w/2^i) \times 256$ (i denotes the stage index). The pyramid stages are then fed into a semantic segmentation head, as described in [10].

2.4. Data Augmentation

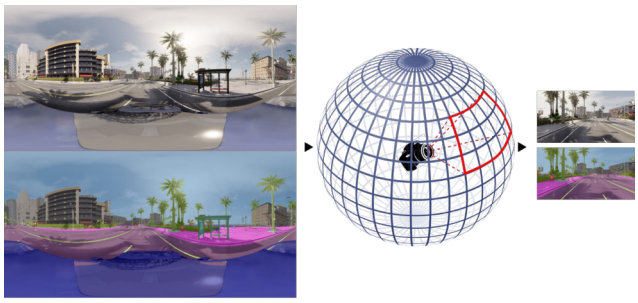

Figure 6. **Data Augmentation**: We create diverse camera models and sensor rotations from equirectangular images.

Sensor equivariance is a generalization problem; one needs a lot of variable data to consider such a problem. With our method, we provide an inductive bias on the sensor model used. Thus, we need data from different sensor types with differences in resolution, field of view, and non-linear distortion to allow our method to generalize over different sensor models. To the best of our knowledge, no data set satisfies these conditions. Therefore, we have developed a concept to emulate such data. For this, we start with equirectangular panoramic images. We assume that for these images, a pixel-accurate ground truth exists for the task under consideration. We emulate cameras with different intrinsic parameters and perspectives from the equirectangular images (as shown in Fig 6). For our implementation

we used the OmniCV-lib [16] and used the unified camera model (UCM) [18] which has five parameters (f_x, f_y, c_x, c_y, and γ), for the intrinsics and is able to model various cameras from pinhole, over wide angle, to fish eye. The UCM projection model for 3D points $\vec{x} = [x, y, z]^T$ is defined as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{x}{\gamma d + (1-\gamma) z} \\ f_y \frac{y}{\gamma d + (1-\gamma) z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2}$$

With c_x and c_y as the center point, f_y, and f_y as the focal length, and $\gamma \in [0, 1]$ as a distortion parameter. The value of $d = \sqrt{x^2 + y^2 + z^2}$ is the norm of the 3D points. From an equirectangular image, the 3D points $\vec{x}$ are not given directly. However, since we know that the pixel in an equirectangular image spans a mapping of pixels (u_e,v_e) to an angular coordinate system $[u_e, v_e, 1]^T \rightarrow [\phi, \theta, 1]^T$ with $\phi \in [-\pi, \pi]$ and $\theta \in [-\pi/2, \pi/2]$ we can apply a projection matrix $\mathbb{K}_e^{-1}$ to obtain the angles ϕ and θ from pixels u_e and v_e in an equirectangular image:

$$\begin{bmatrix} \phi \\ \theta \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \triangle\phi & 0 & -c_\phi \\ 0 & \triangle\theta & -c_\theta \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbb{K}_e^{-1}} \begin{bmatrix} u_e \\ v_e \\ 1 \end{bmatrix} \tag{3}$$

The parameters of $\mathbb{K}_e^{-1}$ are defined by the height h and the width w of the equirectangular image. With $\triangle\phi = 2\pi/w$ and $\triangle\theta = \pi/h$, as well as the center shifts $c_\phi = \pi$ and $c_\theta = \pi/2$. Form ϕ and θ the normalized 3D coordinates can be computed:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} sin(\theta)cos(\phi) \\ sin(\theta)sin(\phi) \\ cos(\theta) \end{bmatrix} \tag{4}$$

We also enable a rotation $\mathbb{R}$ of $\vec{x}$ to get a variation of viewing angles. This gives us eight degrees of freedom for our augmentation approach. From the projection model in Equ. 2 we can build the deflection metric α and the deflection image I_α, as described in Sec. 2.1, for the augmented images.

3. Experiment

For our experimental setup, we use data generated by the CARLA simulator as we will describe in subsection 3.1. In this work, we focus on semantic segmentation as an important task in computer vision and autonomous driving. We compare our work against five baselines described in subsection 3.2. To test the significance of our approach, we perform a statistical test in subsection 3.4

3.1. Dataset

For this work, we want to consider the sensor equivariance in isolation. Therefore, we use the CARLA simulator to generate a data set to augment different sensor models. We created a simulation pipeline in CARLA to obtain 360° panoramas. This allows us to consider the properties of a sensor equivariant model independent of other influences, such as domain shifts. The code for our data generation pipeline is publicly available (see code). Our pipeline generates equirectangular RGB images and equirectangular ground truth images for semantic segmentation, instance segmentation, and depth (or range, to be exact). Furthermore, we include 3D annotations like 3D cuboids of vehicles and VRUs and 3D human key points. Thus, the dataset can be used for object detection, semantic segmentation, instance segmentation, monocular depth estimation, monocular 6 DoF pose estimation, and human pose estimation (2D/3D). However, in this work, we focus on semantic segmentation.

We created the dataset in CARLA 9.14 [3] and used six cameras to stitch the equirectangular panoramas at a resolution of 2048×4096, as shown in Fig. 8. Using the augmentation pipeline as described in Sec 2.4, we can create a vast amount of images with various camera models and distortions from the equirectangular panoramas. We simulated a total of 25k equirectangular panoramas with various conditions (traffic, weather, and maps) and used 20k of the frames for training and 5k of the frames for testing.

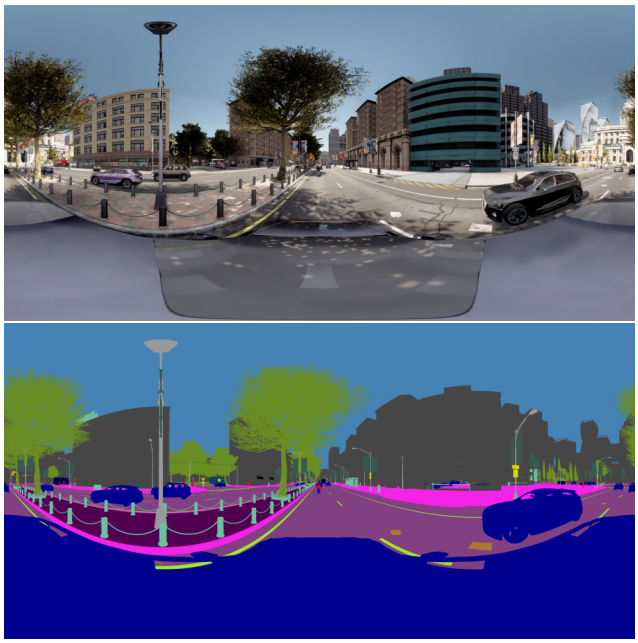

Figure 8. CARLA Panorama Dataset. From top: color image, bottom: semantic segmentation.

3.2. Baselines and Implementation Details

For our experiment, we use five baselines in total. The **Cityscapes** baseline is the first and the most rigorous baseline. We use the **Cityscapes** baseline to demonstrate the limitations of a method that's trained exclusively on the data of a single camera sensor. For the **Cityscapes** baseline, we

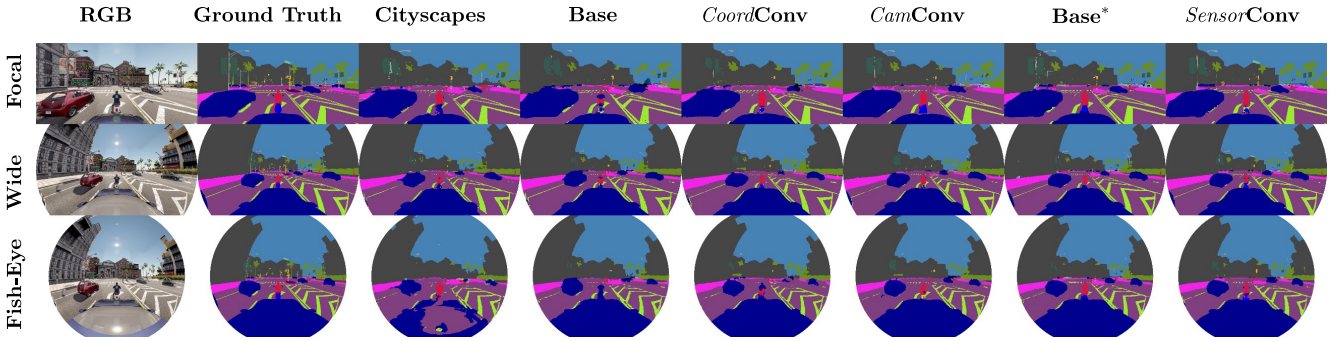

Figure 7. Qualitative results for the considered semantic segmentation methods at different camera configurations.

used a Resnet50-FPN network, trained on the Cityscapes dataset [1], and retrained it on data of our dataset, for a fair comparison. However, we only emulated a sensor with the intrinsic values of the camera used in the Cityscapes dataset while allowing the sensor to rotate. We expect this baseline to overfit on the sensor used, resulting in a strong bias when tested on data from different sensors. Our second baseline, denoted as **Base**, is a vanilla Resnet50-FPN trained from scratch on our dataset. For the augmentation, we allow arbitrary rotations of the sensor, horizontal and vertical resolutions from the intervals $w \in [320, 1080]$ and $h \in [320, 1080]$, a focal length from the interval $f \in [100mm, 3000mm]$, and a distortion $\gamma \in [0, 1]$. With **Base**, we want to examine how well the network architecture can generalize with data from many sensors. With **Base***, we use the same setup as in **Base** but pre-trained on the COCO dataset [11]. **Base*** serves as a baseline for transfer learning. With **CoordConv** [13] and **CamConv** [4], we use two convolutional representations that serve as strong baselines due to the similarity to our proposed methods. For both, we incorporate the modified convolutional layers at the same stages as we incorporate our proposed **SensorConv** layers to a Resnet50-FPN (see subsection 2.3. **SensorConv**, **CoordConv**, and **CamConv** are trained from scratch.

3.3. Setup

We use the mIoU and the IoU of the pedestrian class, to evaluate the semantic segmentation. We use the IoU of the pedestrian class exemplary for smaller objects. We generated 81 test sets with different sensor intrinsics from our equirectangular test data for our evaluation. During the test, we turned off the rotation in our augmentation method to get only forward-facing sensors. We use forward-facing sensors exclusively for testing, to primarily address the differences in sensor intrinsics and not differences in extrinsics such as rotations. We conducted a statistical analysis based on the significance ranking described in [2] and used Autorank [8] to generate critical distance diagrams. We further use box and swarm plots to visualize the median and spread of the IoU across the various sensors. The statistical analysis was conducted for the six considered models with 81 paired samples. We use 5% as the family-wise significance level of the tests. An equivariant solution must perform well on data from all 81 considered emulated cameras. With the significance test, we can make statements about this. Since each frame in a unique test set is captured by the same emulated camera, a test set can be seen as a paired population by statistical means.

3.4. Results

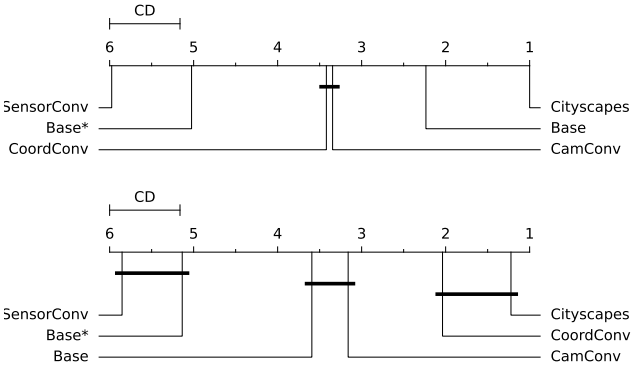

Figure 9. Critical difference (CD) diagrams for mIoU (Top) and IoU-pedestrian (Bottom). CD diagrams show the mean ranks of each model tested on different sensor resolutions. The higher the rank (further to the left), the better the model's performance. A line indicates no significant difference among the models crossed by that particular line.

We display relative results as critical distance diagrams in Fig. 9 and absolute values as box plots in Fig. 10. An ranking can be seen from the critical distance diagram (Fig. 9). The **Cityscapes** configuration is ranked the lowest. In Fig. 9, one can also see that the mIoU of the Cityscapes configuration is by far the lowest. This confirms our hypothesis that a model trained only on data from a fixed camera configuration does not generalize well. In other words, there is a bias towards the sensor used. The **Base** configu-

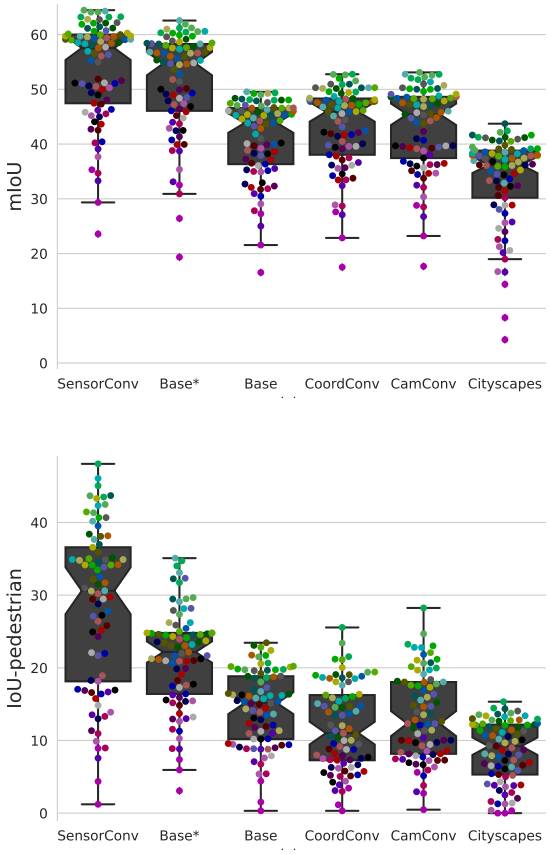

Figure 10. Box and Swarmplot for mIoU (Top) and IoU-pedestrian (Bottom) over the tested camera configurations. Each colored point marks the IoU of the test set with a certain camera configuration.

ration ranks the second lowest when considering the mIoU. This points to a lack of generalizability. However, **Base** configuration ranks third for the pedestrian IoU, showing better performance for smaller objects. The **CamConv** and **CoordConv** configurations are ranked third and fourth for the mIoU with no significant differences. Such similarity is surprising since *CamConv* is an extension of *CoordConv*. While *CoordConv* allows only a spacial equivariance, *CamConv* should allow a spacial and scale equivariance. *CamConv* include *CoordConv* in two of the four channels. We suspect that when using the *CamConv*, the channels containing the FOV are ignored by the model, and thus, effectively, only the *CoordConv* is used. This may be because *CamConv* are designed for pinhole cameras, and nonlinearities due to lens distortions are not considered by *CamConv*. The model thus learns that the FOV channels are untrustworthy for substantial distortions and ignores them. However, the influence of distortion is not that large for smaller objects like pedestrians. In the pedestrian IoU, a significant difference can be seen between **CamConv** and **CoordConv**. The **Base*** configuration ranks second for mIoU and pedestrian-IoU. **Base*** represents a model based on a large and variable data set. We see that such a model can provide a strong foundation. We want to acknowledge that transfer learning is a valid baseline. However, transfer learning requires a holistic data basis. Depending on the domain under consideration, it is difficult to obtain such a data basis. A significant difference remains compared to the highest ranked **SensorConv** in mIoU. This demonstrates the efficiency of our *SensorConv*-Layer and the deflection metric. In Fig. 10 we observe that for IoU-pedestrian **SensorConv** performs better on average, compared to **Base***. However, we can't find a significant difference in the IoU-pedestrian due to a large spread across different sensors. Furthermore, we display the impact of certain sensor parameters on the evaluation by the color coding in Fig. 10. We colored each camera configuration based on the focal length (blue color channel), the resolution (green color channel), and the distortion (red color channel) in the swarm plot. From Fig. 10, we see that pinhole configurations (green points) work best across all considered methods. All considered methods perform worse on wide-angle and fish-eye configurations (magenta points). In Fig. 7, qualitative results can be seen.

4. Broader Impact and Limitation

In the considered experiment, our approach outlines a solid procedure towards a sensor equivariant solution for the problem of semantic segmentation with diverse camera models. Our key finding is that for a sensor equivariant solution it is insufficient to record data with a single camera. Variant data from employing multiple sensors is mandatory to avoid a sensor bias. Likewise, our experiment shows that using only large and variable data is insufficient due to the infamous bias-variance tradeoff. Our model architecture composed of the deflection metric and *SensorConv*-Layer, together with the use of panoramic images and our proposed augmentation method, gives a sound solution. However, due to the use of simulated data, our experimental setup has to be seen as a proof of concept. An evaluation based on real data must be made. In this work we have focused on sensor intrinsics. We cannot state how well our approach generalizes across different sensor mounting positions. Furthermore, using equirectangular panoramic cameras is currently not a focus in self-driving research, which questions a solid data basis at scale. However, with mapping approaches like Google Street View, a large and holistic data pool of equirectangular images can be utilized to scale our approach.

5. Conclusion

This paper presents a novel framework towards the creation of sensor equivariant solutions. We evaluated our approach in an experimental setup on the task of semantic segmentation and compared it with selected baselines. Our approach achieves better segmentation results than the baselines and shows advanced generalization potential despite its simplicity. We are confident that in scenarios where computer vision methods need to generalize across different camera sensors, while the sensor intrinsic properties are given, our *SensorConv* layer will be helpful.

References

[1] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 6

[2] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006. 6

[3] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 3, 5

[4] José M. Fácil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Camconvs: Camera-aware multi-scale convolutions for single-view depth. *CoRR*, abs/1904.02028, 2019. 2, 6

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1

[6] Vitor Guizilini, Igor Vasiljevic, Dian Chen, Rareș Ambruș, and Adrien Gaidon. Towards zero-shot scale-aware monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9233–9243, October 2023. 2

[7] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 4

[8] Steffen Herbold. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173, 2020. 6

[9] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[10] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 4

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 6

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 4

[13] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *CoRR*, abs/1807.03247, 2018. 2, 6

[14] Varun Ravi Kumar, Marvin Klingner, Senthil Yogamani, Markus Bach, Stefan Milz, Tim Fingscheidt, and Patrick Mäder. Svdistnet: Self-supervised near-field distance estimation on surround view fisheye cameras. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10252–10261, 2022. 2

[15] Hannes Reichert and Konrad Doll. An image encoding method for recording projection information of two-dimensional projections, WO2023118163A1, Dec. 2022, (pending). 3

[16] Kaustubh Sadekar. Omnicv-lib. https://github.com/kaustubh-sadekar/OmniCV-Lib, 2020. 5

[17] Hanno Scharr. *Optimale Operatoren in der Digitalen Bildverarbeitung*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2000. 3

[18] Vladyslav C. Usenko, Nikolaus Demmel, and Daniel Cremers. The double sphere camera model. *2018 International Conference on 3D Vision (3DV)*, pages 552–560, 2018. 2, 5

[19] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation, 2020. 2, 4

[20] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Solo: A simple framework for instance segmentation, 2021. 2, 4

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1

[22] Wang Yifan, Carl Doersch, Relja Arandjelović, João Carreira, and Andrew Zisserman. Input-level inductive biases for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6176–6186, June 2022. 2

[23] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. *arXiv preprint arXiv:1905.01489*, 2019. 2