FisheyeDetNet: 360° Surround view Fisheye Camera based Object Detection System for Autonomous Driving

Ganesh Sistu[1†] and Senthil Yogamani[2†]
[1]University of Limerick, Ireland
[2]Valeo Vision Systems, Ireland [†]co-first authors

***Abstract*—Object detection is a mature problem in autonomous driving with pedestrian detection being one of the first deployed algorithms. It has been comprehensively studied in the literature. However, object detection is relatively less explored for fisheye cameras used for surround-view near field sensing. The standard bounding box representation fails in fisheye cameras due to heavy radial distortion, particularly in the periphery. To mitigate this, we explore extending the standard object detection output representation of bounding box. We design rotated bounding boxes, ellipse, generic polygon as polar arc/angle representations and define an instance segmentation mIOU metric to analyze these representations. The proposed model FisheyeDetNet with polygon outperforms others and achieves a mAP score of 49.5% on Valeo fisheye surround-view dataset for automated driving applications. This dataset has 60K images captured from 4 surround-view cameras across Europe, North America and Asia. To the best of our knowledge, this is the first detailed study on object detection on fisheye cameras for autonomous driving scenarios.**

I. Introduction

When an autonomous vehicle moves from source to destination, a navigator like google maps or HD Maps generate a high-level route. This route is made up of a series of connected nodes at finite distances. The vehicle moves from one node to another in a repetitive way until it reaches the destination. Maneuver occurs in a five-stage process, Sensing, Perception and Localization, Scene Representation, Planning, and Controlling (illustrated in Figure 2).

In the sensing stage, the vehicle collects information about the surroundings via sensors like Camera, LiDAR, RADAR, and Ultrasonics. Perception involves the extraction of useful information from the raw data like lane positions, presence of pedestrians, and other vehicles [1], [2], semantic segmentation [3], [4], moving objects detection [5], [6], depth estimation [7], [8], feature correspondence [9], [10] and recognition of drivable regions [11], [12]. In recent times deep learning algorithms have shown tremendous success in almost all the perception related tasks. Localization is the vehicle's ability to precisely know its position in the real world at decimeter accuracy [13], [14]. In simple words, perception answers what is around the vehicle, and localization answers where is the vehicle precisely. Path planning algorithms [15] make use of this related information to define a path to navigate from one node to another. While defining the path, the algorithms use different driving policies like safety, rules of driving, road conditions, and pleasant ride experience for the passengers in the car [16]. These digital instructions from the algorithms are converted into the vehicle's physical movement via the controlling unit [17].

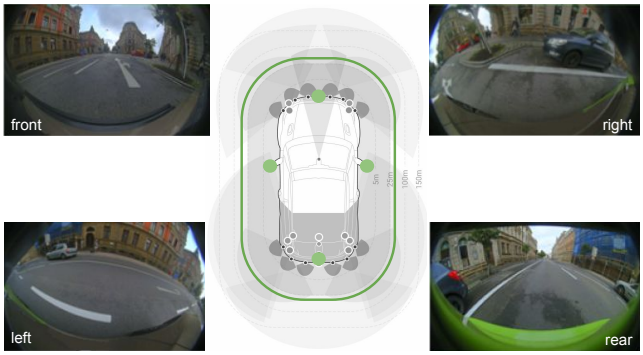

Fig. 1: Surround-view camera network images showing near field sensing and wide field of view

In the autonomous driving pipeline, perception is a computationally expensive and sophisticated block, and efforts to build unified models for all perception related tasks is an active area of research [18] [19]. Though there is a considerable debate on what sensors are needed for this task, equivocally, cameras are considered as an essential sensor for any perception pipeline. Visual perception is the task of perceiving the information around the vehicle via cameras. A modern automated vehicle consists of anywhere between 4 to 20 cameras of different field of view (FoV) performing different tasks. Visual perception can be described as a combination of Recognition, Reconstruction, and Re-localization. Recognition is knowing what is around the vehicle and involves tasks like Detection, Segmentation, and lens soiling [20], [21]. Reconstruction consists of depth estimation, motion estimation to know where the objects are in the 3D world. Re-localization knows where the ego vehicle is in the world. It involves pose estimation and SLAM [13]. Other than this perception also involves lesser-known tasks like trailer angle estimation, measuring sun glare on the lens.

A. Low-Power Hardware for Automated Driving

Recently demand for low-power SoC based automated vehicles has increased significantly as features like pedestrian detection, emergency braking and lane keep assist started to attract more consumers. Typical low-power SOCs include Renesas V3H, TI TDA4x, and Nvidia Xavier. The choice of SoC is based on the criteria of performance (Tera Operations Per Second (TOPS), utilization, bandwidth), cost, power con-

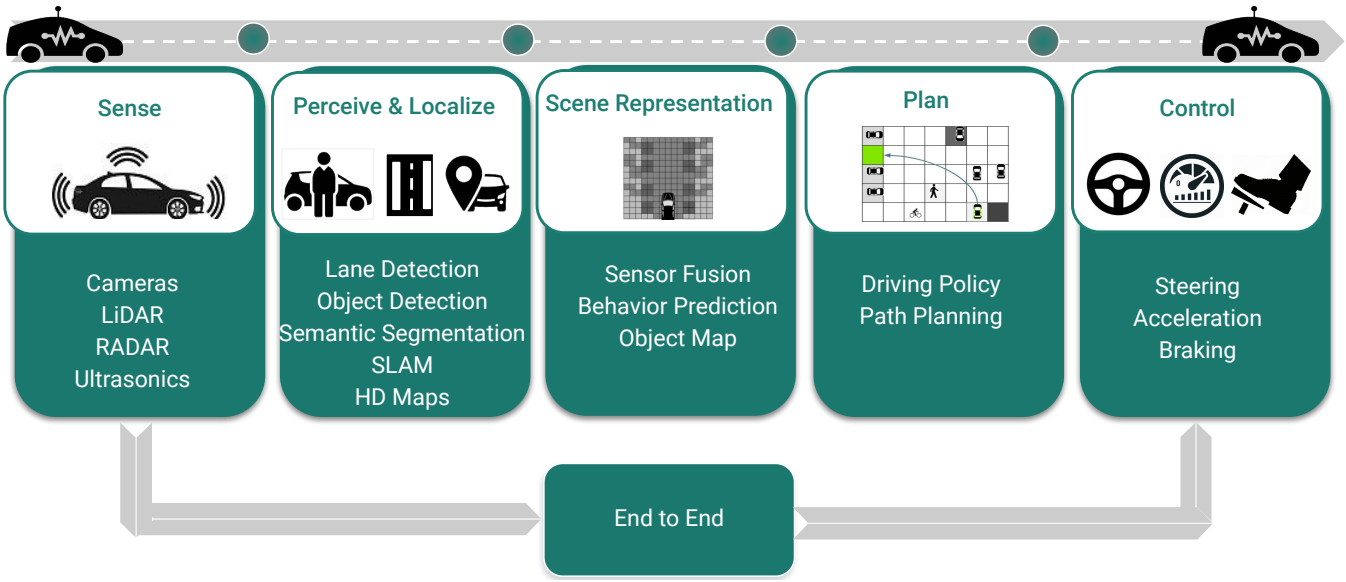

Fig. 2: Autonomous Driving Pipeline

sumption, heat dissipation, high to low-end scalability and programmability. The SOC choice provides the computational bounds in the design of algorithms. The progress in Convolutional Neural Networks (CNNs) has also led the hardware manufacturers to include their custom hardware computing units to provide a high throughput of over 10 TOPS targeting Level-3 automation. But Level-2 automation systems still rely on computing units less than 2 TOPS. In [22], authors developed a multitask learning algorithm for hardware with 1 TOPS of computing power, consuming less than 10 watts of power.

B. Object Detection

Object detection is the first and foremost problem in visual perception, which involves the recognition and localization of the objects in the image. It has use-cases like emergency braking and collision avoidance etc. Hence object detection models' performance directly influence the success and failure of autonomous driving systems. A simple and efficient representation of objects in images is bounding box representation. State-of-the-art methods for object detection based on deep learning can be broadly classified into two types,

- Two stage detectors
- Single-Stage detectors.

1) Two Stage Object Detection: In two-stage approaches, object detection is split into two tasks, (i) Extraction of Region of Interest (ROI)s and encoding them as features (ii) Regressing for bounding boxes in these ROIs using encoded features. A common practice is to have a high Recall in the first stage to ensure all possible objects like patterns go through the second stage. RCNN[23] was the first to use this approach. In RCNN first stage, a selective search algorithm is used to propose ROIs, followed by a CNN feature extraction. In the second stage, an SVM is trained to classify the objects based on the CNN features. Unlike RCNN, which extracts CNN features separately on each ROI, Fast-RCNN [24] process the whole image. So the CNN feature extraction is performed only once per image. It has introduced a 25x speed in the inference stage compared to RCNN. Also, Fast-RCNN has replaced SVM with a linear classifier and introduced a linear regressor for bounding box fine-tuning. This moved the Fast-RCNN a step closer to the end differentiable training strategy. Both Fast-RCNN [24] and SPP-net [25] improved RCNN [23] by extracting RoIs from the feature maps. SPP-net introduced a spatial pyramid pooling (SPP) layer to handle images of arbitrary sizes and aspect ratios. It applies SPP layer over the feature maps generated from convolution layers and outputs fixed-length vectors required for fully connected layers. It eliminates fixed-size input constraints and can be used in any CNN-based classification model. However, Fast-RCNN and SPPnet are not end-to-end trainable as they depend on the region proposal approach. Faster-RCNN [26] solved this limitation by introducing Region Proposal Network (RPN), which made end-to-end training possible. RPN generates RoIs by regressing a set of reference boxes, known as anchor boxes. This introduced two streams for object detection, i.e., a common encoder and two decoders. The efficiency of Faster-RCNN is further improved by RFCN [27], which replaces fully connected layers with fully convolutional layers.

2) Single Stage Object Detection: These approaches eliminate the RoI extraction stage and carry out classification and regression of bounding boxes directly on CNN feature maps and hence a single state encoder-decoder style network performs localization and classification tasks. Overfeat [28] proposed a unified framework to perform two tasks: classification and localization using a multi-scale, sliding window approach. YOLO [29] divides the input image into grids and predicts bounding boxes directly by regression and classification at each grid. This soon became a defacto style for single state real-time object detection on low power hardware like mobile phones and Level-3 autonomous driving engines. YOLO9000 (YOLOv2) [30] improved the performance by introducing

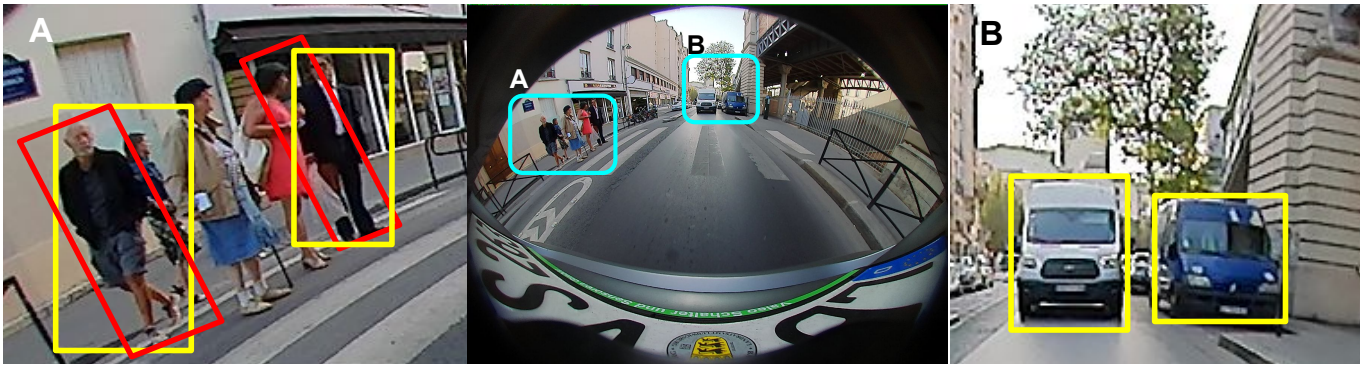

Fig. 3: Center: Front camera image. Right(B): Bounding boxes representing objects correctly. Left(A): Bounding boxes and oriented boxes fail to represent objects accurately, more details in Section I

batch normalization and replacing fully connected layers of YOLOv1 with anchor boxes for bounding box prediction. Anchor boxes are computed over the dataset, representing the average variation of height and width of the objects in the dataset. Instead of directly regressing for object width and height, YOLOv2 predicts off-sets from a predetermined set of anchors with particular height-width ratios. YOLOv3 [31], a faster and accurate object detector than previous versions, uses Darknet-53 as its feature extraction backbone. YOLOv3 can detect small objects with a multi-scale prediction approach, a significant drawback in earlier versions.

Single Shot Multibox Detection(SSD) [32] places dense anchor boxes over the input image and extract feature maps from multiple scales. It then classifies and regresses the bounding boxes relative to anchor boxes. DSSD [33] replaced the VGG network of SSD with Residual-101. It is then augmented with a deconvolution module to integrate feature maps from the early stage with the deconvolution layers. It outperforms the SSD in detecting small objects. MDSSD [34] further extends DSSD with fusion blocks to handle feature maps at different scales.

RetinaNet [35] introduced focal loss to address foreground and background class imbalance during training. It matches or surpasses the accuracy of state-of-the-art two-stage detectors while running at faster speeds. The architecture shares 'anchors' from RPN and builds a single Fully Convolutional Network (FCN) with Feature Pyramid Network (FPN) on top of the ResNet backbone.

C. Instance Segmentation

Instance segmentation involves predicting both object bounding box and pixel-level object mask.

1) Two stage Instance Segmentation: Intuitively instance segmentation can be modeled as a bounding box detection followed by a binary segmentation within the box. This paradigm is referred to as 'Detection then Segmentation'. Models following this approach often achieve a state of the art performance but are quite slow to adapt to real-time applications. MaskRCNN [36] adapted this approach by using FasterRCNN for bounding box detection and an additional decoder for object mask segmentation. Here segmentation is performed as a binary classification to differentiate object pixels from the background or other object pixels. Multi-task Network Cascade (MNC) [37] uses a similar approach to MaskRCNN. It uses RPN for box proposals, followed by class agnostic instances generation on the proposed regions and finally categorical classification of these mask instances. Figure 4 shows the similarity between MNC and MaskRCNN algorithms. During the inference time on a 12 GB, 7 TFLOPs NVIDIA M40 GPU, MaskRCNN reported a 6 FPS run time. Today even the Level-5 autonomous vehicles use only 1.3 to 2 TFLOPs computing engines for running the complete deep learning stack, making a state of the art two-stage approach far from reality for L3 automated vehicles. This led to a recent trend of simplistic single-stage object detection style instance segmentation techniques like PolarMask [38] YOLOACT [39] and PolyYOLO [40].

2) Single stage Instance Segmentation: YOLOACT [39] uses a single encoder dual parallel decoder style architecture for instance-level image segmentation. Encoder is same as RetinaNet backbone, i.e Feature Pyramid Network with Resnet101 [41]. The first decoder generates a set of k prototype masks at image resolution. These masks do not depend on any single object class. However, these masks represent instance masks of an object when multiplied with the correct set of coefficients. The second decoder is a standard bounding box decoder with extra computation to predict mask coefficients for each object instance. Instance masks for objects are generated as a linear combination of prototype masks and mask coefficients. Though YOLOACT performance is lower than MaskRCNN, it is 5x faster in run time.

3) Polygon Instance Segmentation: PolarMask [38] and PolyYOLO [40] regress for contour boundaries in polar space. It is hence removing computational overheads of an extra decoder and segmentation of pixels at images level.

Other approaches to instance segmentation range from clustering of instance embedding [42], [43] to prediction of instance centers using offset regression [44]. These methods appear intuitively designed but are lagging in terms of accuracy and computational efficiency. The major drawback of these methods is the usage of compute-intensive clustering methods like OPTICS [45], DBSCAN [46].

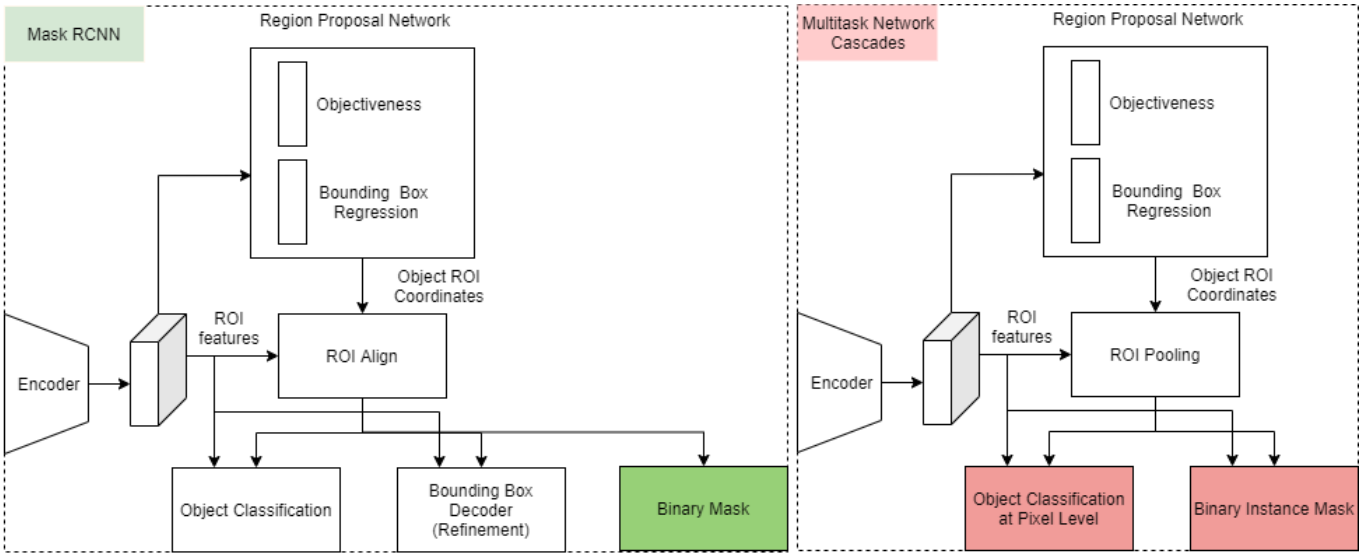

Fig. 4: Comparison between MaskRCNN and Multi-task Network Cascade. Both models are two stage approaches and use FasterRCNN components (blocks not colored)

Fig. 5: Undistorting the fisheye image: (a) Rectilinear correction; (b) Piecewise linear correction; (c) Cylindrical correction. Left: raw image; Right: undistorted image.

II. Object Detection on Fisheye Cameras

Fisheye cameras make use of non-linear mapping to generate a large field of view. With just four surround-view fisheye cameras, we can achieve a dense 360° near field perception, making them suitable for automated parking, low-speed maneuvering, and emergency braking. A commercial fisheye camera usually has a 190° horizontal field of view as shown in Figure 1. It is usually available from 2MP to 20MP resolution. However, this advantage comes at the cost of non-linear distortions in the image. Objects at different angles from the project center look quite different, making the object detection a challenge.

A common practice is to rectify distortions in the image by a 4^{th} order polynomial model or Unified camera model [47], [48]. The fact is, there is no ideal projection or correction. These corrections are application-driven, and every correction technique has its disadvantages (Figure 5). Rectilinear correction suffers from loss of Field of View (FoV) and sampling issues, Piece-wise linear with artifacts at transition areas and massive bleeding in the image, and Cylindrical as a quasi-linear correction, offers a practical trade-off. Another overhead is extra computational resources needed for correction as the visual perception pipeline usually have different algorithms demanding different view projections. Though Look Up Tables (LUTs) make this correction process accelerated, LUTs rely on online calibration that needs to be generated every time there is a change in the online calibration.

Despite these disadvantages, image correction is encouraged due to the limitations of the non or early deep learning object recognition and segmentation algorithms. With a push in deep neural networks, this trend is slowly changing. Modern CNN based object detection algorithms like YOLO and FasterRCNN can detect objects on raw fisheye images and main issue with object detection on raw fisheye images is representation of objects as bounding boxes.

A. Bounding boxes on Fisheye

Objects go though serious deformations due to radial distortion in fisheye images and box representation fails in many practical scenarios [49]. Here are two scenarios where the correct representation of objects is as important as the detection.

1) Pedestrian Localization Issue: In Figure 3.B, vehicles are near the center region of the image, and hence the lower part of the bounding boxes represent the object intersection with the road quite well. However, in Figure 3.A, standard

bounding boxes in yellow color are not good enough to represent the object road intersection.

The common idea is to orient the boxes as shown in red color in Figure 3.A. In the case of the person on the left side, this orientation concept works. As the box with optimal orientation is also a box with optimal IoU with the ground truth. However, in the case of the person in a black suit, the optimally oriented box is not the optimal IoU. So simple orientation works in some cases, but it does not solve the problem. 3D boxes work, but both annotating and inferring a 3D box is a noisy process for small objects.

2) Missing Parking Spot: A correctly detected but improperly represented objects can result in failure cases like missing a parking spot or in non-optimal path planning. Figure 6 shows an automated car maneuvering to a parking slot between the two cars. Two cars got detected by bounding box, oriented box, ellipse and polygon object detection algorithms. However, only in instance segmentation case objects are located correctly outside the free parking spot. In rest of the cases, objects seems to be present inside the parking spot and in those cases the free parking spot in maneuver mapping shows as occupied (bottom row images). This shows that the detection of objects is as important as correct representation in fisheye based visual navigation systems.

A full-fledged solution to this problem is instance segmentation, but most state-of-the-art algorithms like MaskRCNN demand higher computing powers and are unrealistic to work on low-power hardware that is generally used in Level2 and Level 3 autonomous vehicles. Hence there is a need for memory and computation efficient models. It encouraged us to develop FisheyeDetNet, a single network to perform object detection and instance segmentation to deploy on low power hardware accelerators. It is an efficient, small footprint network that uses ResNet18 as a backbone and YOLO style head for polygon based instance segmentation.

III. Proposed Method

Objects detection can be represented as bounding boxes, rotated bounding boxes, ellipses, and polygons. Irrespective of the algorithm used, each representation has a limitation on maximum performance it can achieved on the given dataset. We term this as empiricall upper bound' or simply upper bound. The same is shown in Table I, where mean IoU score between the annotations from each representation and ground truth instance annotations is presented for our fihseye dataset. In case of polygon representation, instance annotations are generated by sampling 12, 24, 36, 60 and 120 points per 360° in polar coordinates. We modified the YOLOv3 [31] network to accommodate all these four different representations.

- Bounding Box
- Oriented Bounding Box
- Ellipse
- Polygon

To make the network feasible to port onto a low power automotive hardware, we used ResNet18 [41] as an encoder. Compared to standard Darknet53 encoder [31], this has nearly 60% fewer parameters. Proposed network architecture is shown in Figure 9. Different representations are implemented in representation block.

A. Bounding Box

Our Bounding box model is the same as YOLOv3 except Darknet53 encoder is replaced with ResNet18 encoder. Similar to YOLOv3, object detection is performed at multiple scales. For each grid in each scale, object width($\hat{w}$), height($\hat{h}$), object center coordinates($\hat{x}$, $\hat{y}$) and object class is predicted. Finally, a non-maximum suppression is used to filter out the redundant detections. Instead of using L_2 loss for categorical and objectness classification, we used standard categorical cross-entropy and binary entropy losses, respectively.

Representing the modified YOLO loss as a combination of sub-losses,

$$\mathcal{L}_{xy} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \quad (1)$$

$$\mathcal{L}_{wh} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \quad (2)$$

$$\mathcal{L}_{obj} = -\sum_{i=0}^{S^2} \sum_{j=0}^{B} C_i log(\hat{C}_i) \quad (3)$$

$$\mathcal{L}_{class} = -\sum_{i=0}^{S^2} l_{ij}^{obj} \sum_{c=classes} c_{i,j} log(p(\hat{c_{i,j}})) \quad (4)$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{wh} + \mathcal{L}_{obj} + \mathcal{L}_{class} \quad (5)$$

where height and width are predicted as offsets from precomputed anchor boxes.

$$\hat{w} = a_w * e^{fw} \quad (6)$$

$$\hat{h} = a_h * e^{fh} \quad (7)$$

$$\hat{x} = g_x + f_x \quad (8)$$

$$\hat{h} = g_y * f_y \quad (9)$$

where a_w, a_h anchor box width and height. f_w, f_h, f_x, f_y are the outputs from last layer of the network at grid location g_x, g_y.

B. Oriented Bounding Box

In this model along with the regular box information ($\hat{w}$, $\hat{h}$, $\hat{x}$, $\hat{y}$), orientation of the box $\hat{\theta}$ is also regressed. Orientation ground truth range (-180 to +180°) is normalized between -1 to +1. The loss function is same as the regular box loss but with an additional term for orientation loss.

$$\mathcal{L}_{orn} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [\theta_i - \hat{\theta}_i]^2 \quad (10)$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{wh} + \mathcal{L}_{obj} + \mathcal{L}_{class} + \mathcal{L}_{\mathbf{orn}} \quad (11)$$

where $\mathcal{L}_{total}$, is the total loss minimized for oriented box regression.

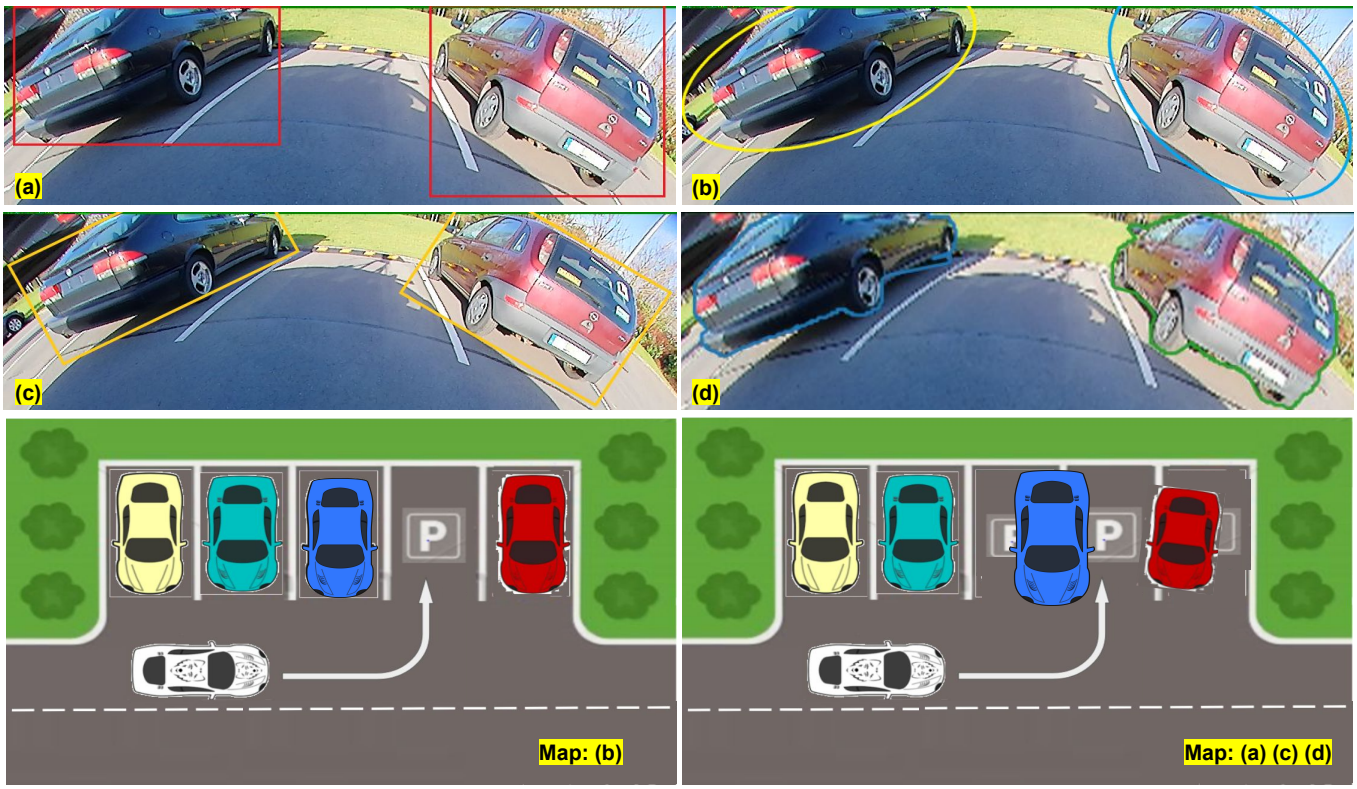

Fig. 6: Parking spot failure case. Bottom left: 2D map of navigation showing free parking spot. Bottom right: Same 2d map showing no free parking spot. Object detection as bounding box(a), ellipse(b), oriented box (c) and instance segmentation (d)

Representation	Bounding Box	Rotated Box	12 points	24 points	36 points	60 points	120 points
IoU	0.552	0.643	0.853	0.897	0.918	0.942	0.984

TABLE I: Upper bound on performance of various representations

C. Ellipse Detection

Ellipse regression is the same as oriented box regression. The only difference is in the output representation. Hence the loss function is also the same as oriented boxes loss.

D. Polygon Detection

Our proposed approach for polygon-based instance segmentation is quite similar to PolarMask [38] and PolyYOLO [40] approaches. Instead of using sparse polygon points and single scale predictions like PolyYOLO. We use dense polygon annotations and multi-scale predictions. Instead of heavy backbone architecture like PolarMask, we employed lightweight ResNet-18 as our encoder. These changes enabled us to develop a small footprint instance segmentation model with just 13M parameters. As there is no heavy encoder backbone or feature map upscaling to image level and segmentation at the pixel level, our model is quite suitable for real-time applications like object detection on Level-3 automotive ECUs. Keeping the network architecture similar in all the four experiments results in a fair comparison between different representations.

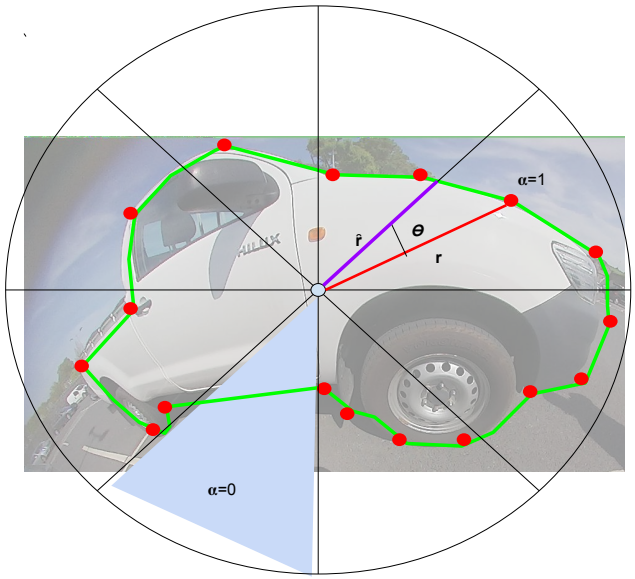

Fig. 7: Dense pixel level annotation sampling (purple) vs sparse polygon points annotation sampling (red) in polar space.

The polygon regression loss is given by,

$$\mathcal{L}_{poly} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \sum_{k=0}^{R} l_{ij}^{obj} [r_{i,k} - \hat{r}_{i,k}]^2 \tag{12}$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{wh} + \mathcal{L}_{obj} + \mathcal{L}_{class} + \mathcal{L}_{\mathbf{poly}} \tag{13}$$

Experiment	Representation vs Representation			Representation vs Instance Annotation		
	Vehicle	Pedestrian	mAP	Vehicle	Pedestrian	mAP
Bounding Box	0.6627	0.3157	0.4892	0.5132	0.3150	0.4141
Oriented Box	0.6548	0.3010	0.4779	0.5234	0.3185	0.4210
Ellipse	0.6601	0.2900	0.4751	0.5290	0.2889	0.4090
Polygon (24 points)	0.6624	0.3140	0.4882	0.6761	0.3155	0.4958

TABLE II: Comparison of various representations. In case of polygon experiment, Representation vs Representation metric is between a bounding box annotation and bounding box predicted along with polygons

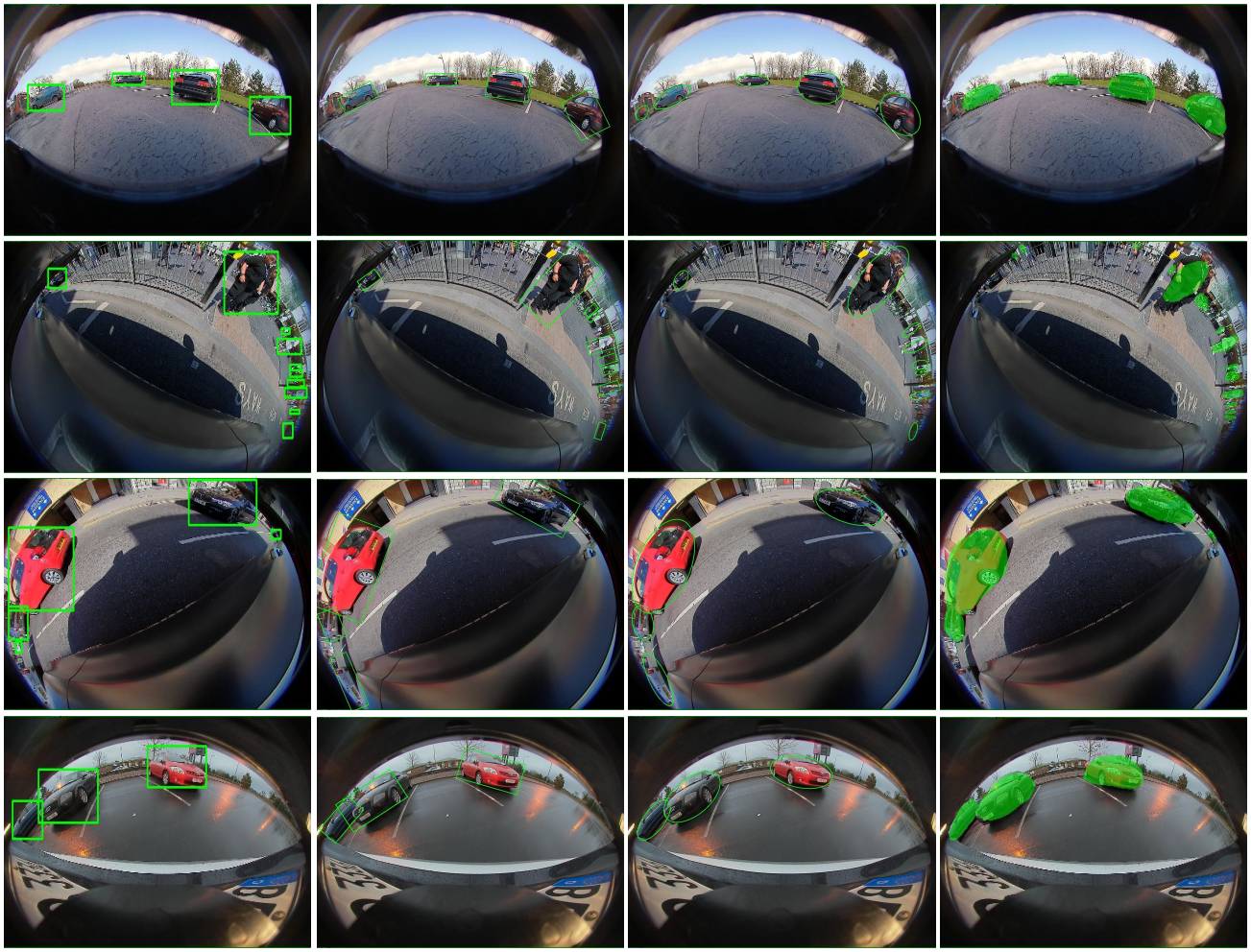

Fig. 8: Qualitative results: Each row shows output of all four models on the same image. Each column shows images captured by FV, MRV, MLV and RV cameras on the vehicle

The total loss is given by $\mathcal{L}_{total}$, where R corresponds to the number of sampling points, each point is sampled with a step size of $360/R$ angle in polar coordinates, as shown in Figure 7. We used dense pixel-level annotations, and hence there is only one parameter needed to represent each polygon point in the polar coordinate system. It is similar to PolarMask. PolyYOLO, on the other hand, uses sparse polygon points (in red), and thus requires 3 parameters r, θ and α. Hence the total required parameters for R sampling points are 3*R in case of sparse polygon points-based annotations. The effect of different sampling rates w.r.t actual pixel-level annotation masks is presented in Table I.

IV. Experiments

All the four models are systematically tested on large scale automotive fisheye surround-view dataset. Dataset deatils, metrics and evaluation criteria and training details are presented in the following subsections.

A. Dataset

We presented results on Valeo proprietary dataset for automated driving applications. This dataset comprises of 50,000 images captured from four surround-view cameras [50], [51]. Instance segmentation for vehicles and pedestrians. A subset of this dataset with more annotation classes and annotation for different tasks is presented in [19], [52]. Figure 10 shows the

diversity of geographical, climatic conditions of the dataset. The density maps show that majority of the vehicles and pedestrians are within 20 meters of the vehicle. It is an important metric as fisheye surround-view cameras are usually mounted for near filed visual perception applications. Dataset is divided into train, validation, and test splits at 70, 15, 15 proportions. A random sampling technique is used for this purpose.

B. Training Details

All four models are trained on nearly 35K images at an input resolution of 544X288 $(widthXheight)$. A pre-trained ResNet18 model without classification layers is used as Encoder and horizontal image flip as data augmentation technique. All models are developed on PyTorch v1.4 [53]. Training, evaluation and inference are performed on a NVIDIA GTX 1080Ti GPU. All models are trained for 80 epochs with early stopping criteria based on validation loss. Ranger optimizer [54] and one cycle learning rate scheduler [55] is used for optimization. Ranger uses gradient stabilization, combines RAdam [56] and LookAhead [57] in one optimizer. Hence helps in stabilized training.

C. Results

All models are compared using a mean average precision metric (mAP) with an IoU threshold of 50%. Results are presented in Table II. Each algorithm is evaluated based on two criteria - Performance on same representation and on instance segmentation. For example, a bounding box detection model predictions are compared with bounding box ground truth (Representation vs Representation) and instance mask ground truth (Representation vs Instance Annotation). While the comparison with the same representation shows the performance of the algorithm, comparison with instance masks shows its closeness to its upper bound. Results in Table II are in alignment with the empirical upper bounds shown in Table I. This shows that many practical failure cases like missing parking spots can be solved with a change in representation as opposite to increasing network capacity.

Qualitative results on test set for all four representations on all four cameras are shown in Figure 8, In Row-1: Though all four models detected the vehicles, polygon segmentation is the only representation to solve the missing parking spot problem. In Row-2: Oriented boxes and ellipse are able to locate the pedestrian precisely, while standard box and polygon failed. Row-4: Missing parking spot problem is handled well by both ellipse and polygon segmentation representation models.

V. Conclusion

In this work, we studied the problem of bounding box object detection on fisheye images. First, we demonstrated that due to strong radial distortions the bounding box is not a good representation of object detection on fisheye images due to strong radial distortion. Then, we explored several improved representations starting from a rotated bounding box, ellipse, and then finally a generic polygon. We proposed a novel algorithm by extending YOLO to regress a generic representation across the representations, as mentioned above. We call our algorithm FisheyeDetNet, and the implementation demonstrates significant improvements over the baseline representations. We also showed that many of the practical problems can be solved by learning the right representations instead of increasing the model complexity with same models.

References

[1] G. Sistu, I. Leang, and S. Yogamani, "Real-time joint object detection and semantic segmentation network for automated driving," *arXiv preprint arXiv:1901.03912*, 2019.

[2] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz, and P. Mader, "Bevdetnet: bird's eye view lidar point cloud based real-time 3d object detection for autonomous driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 2809–2815.

[3] S. Chennupati, V. Narayanan, G. Sistu, S. Yogamani, and S. A. Rawashdeh, "Learning panoptic segmentation from instance contours," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9586–9593.

[4] S. Chennupati, G. Sistu., S. Yogamani., and S. Rawashdeh., "Auxnet: Auxiliary tasks enhanced semantic segmentation for automated driving," in *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2019, pp. 645–652.

[5] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Moving object detection network with motion and appearance for autonomous driving," *arXiv preprint arXiv:1709.04821*, 2017.

[6] E. Mohamed, M. Ewaisha, M. Siam, H. Rashed, S. Yogamani, W. Hamdy, M. El-Dakdouky, and A. El-Sallab, "Monocular instance motion segmentation for autonomous driving: Kitti instancemotseg dataset and multi-task baseline," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 114–121.

[7] V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Near-field depth estimation using monocular fisheye camera: A semi-supervised learning approach using sparse LiDAR data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[8] V. R. Kumar, M. Klingner, S. Yogamani, M. Bach, S. Milz, T. Fingscheidt, and P. Mäder, "Svdistnet: Self-supervised near-field distance estimation on surround view fisheye cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 252–10 261, 2021.

[9] A. Konrad, C. Eising, G. Sistu *et al.*, "FisheyeSuperPoint: Keypoint Detection and Description Network for Fisheye Images," *Proceedings of the International Conference on Computer Vision Theory and Applications*, vol. abs/2103.00191, 2021.

[10] S. Shen, L. Kerofsky, and S. Yogamani, "Optical flow for autonomous driving: Applications, challenges and improvements," in *Electronic Imaging*. Society for Imaging Science and Technology, 2023.

[11] C. Hughes, S. Chandra, G. Sistu, J. Horgan, B. Deegan, S. Chennupati, and S. Yogamani, "Drivespace: towards context-aware drivable area detection," *Electronic Imaging*, vol. 31, pp. 1–9, 2019.

[12] F. Stapleton, E. Galván, G. Sistu, and S. Yogamani, "Neuroevolutionary multi-objective approaches to trajectory prediction in autonomous vehicles," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 675–678.

[13] N. Tripathi and S. Yogamani, "Trained trajectory based automated parking system using visual slam," 2020.

[14] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 247–257.

[15] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[16] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[17] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[18] G. Sistu, I. Leang, S. Chennupati, S. Yogamani, C. Hughes, S. Milz, and S. Rawashdeh, "Neurall: Towards a unified visual perception model for automated driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 796–803.

[19] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, S. Chennupati, M. Uricar, S. Milz, M. Simon, K. Amende, C. Witt, and et al., "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2019.00940

[20] M. Uricár, J. Ulicny, G. Sistu, H. Rashed, P. Krizek, D. Hurych, A. Vobecky, and S. Yogamani, "Desoiling dataset: Restoring soiled areas on automotive fisheye cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[21] M. Uricár, G. Sistu, H. Rashed, A. Vobeckỳ, P. Krízek, F. Burger, and S. K. Yogamani, "Let's get dirty: Gan based data augmentation for soiling and adverse weather classification in autonomous driving," *arXiv preprint arXiv:1912.02249*, 2019.

[22] T. Boulay, S. El-Hachimi, M. K. Surisetti, P. Maddu, and S. Kandan, "Yuvmultinet: Real-time yuv multi-task cnn for autonomous driving," *arXiv preprint arXiv:1904.05673*, 2019.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.81

[24] R. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.169

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[27] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," 2016.

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.91

[30] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.690

[31] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[33] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[34] L. Cui, R. Ma, P. Lv, X. Jiang, Z. Gao, B. Zhou, and M. Xu, "Mdssd: Multi-scale deconvolutional single shot detector for small objects," *arXiv preprint arXiv:1805.07009*, 2018.

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.324

[36] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.322

[37] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.343

[38] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 193–12 202.

[39] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 9157–9166.

[40] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba, "Poly-yolo: higher speed, more precise detection and instance segmentation for yolov3," 2020.

[41] 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.

[42] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan, "Proposal-free network for instance-level object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, p. 2978–2991, Dec 2018. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2017.2775623

[43] D. Neven, B. D. Brabandere, M. Proesmans, and L. Van Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00904

[44] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. [Online]. Available: http://dx.doi.org/10.1109/cvpr42600.2020.01249

[45] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 49–60. [Online]. Available: https://doi.org/10.1145/304182.304187

[46] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[47] V. R. Kumar, C. Eising, C. Witt, and S. K. Yogamani, "Surround-view fisheye camera perception for automated driving: Overview, survey & challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3638–3659, 2023.

[48] C. Eising, J. Horgan, and S. Yogamani, "Near-field perception for low-speed vehicle automation using surround-view fisheye cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 13 976–13 993, 2021.

[49] H. Rashed, E. Mohamed, G. Sistu, V. R. Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "Fisheyeyolo: Object detection on fisheye cameras for autonomous driving," in *Proceedings of the Machine Learning for Autonomous Driving NeurIPS 2020 Virtual Workshop, Virtual*, vol. 11, 2020.

[50] M. Uricár, D. Hurych, P. Krizek *et al.*, "Challenges in designing datasets and validation for autonomous driving," in *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2019.

[51] L. Yahiaoui, J. Horgan, B. Deegan, S. Yogamani, C. Hughes, and P. Denny, "Overview and empirical analysis of isp parameter tuning for visual perception in autonomous driving," *Journal of Imaging*, vol. 5, no. 10, p. 78, 2019.

[52] S. Ramachandran, G. Sistu, J. McDonald, and S. Yogamani, "Woodscape fisheye semantic segmentation for autonomous driving–cvpr 2021 omnicv workshop challenge," *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2021.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[54] H. Yong, J. Huang, X. Hua, and L. Zhang, "Gradient centralization: A new optimization technique for deep neural networks," 2020.

[55] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[56] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[57] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead optimizer: k steps forward, 1 step back," 2019.

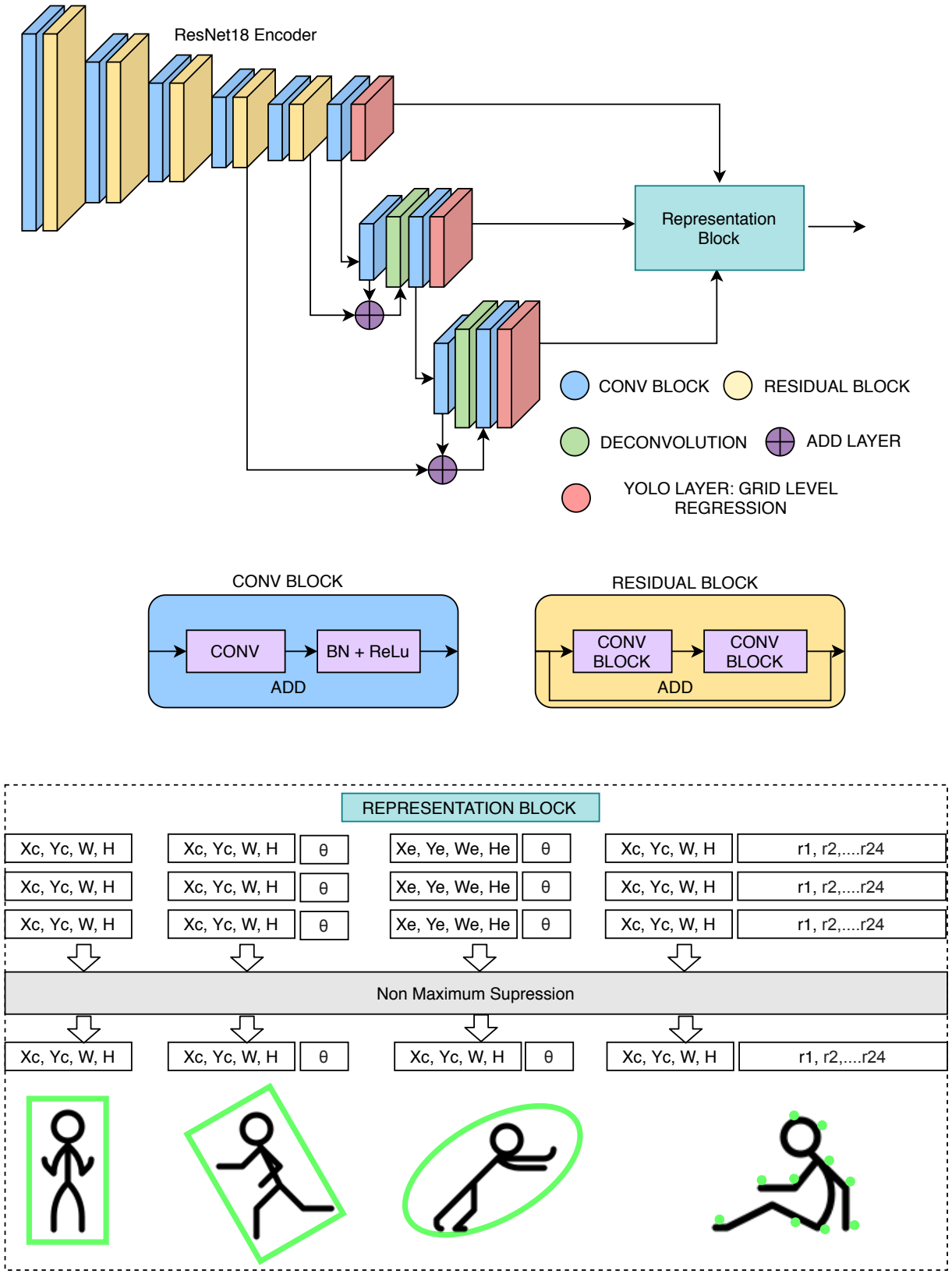

Fig. 9: Proposed object detection network architecture and comparison between different representations. Total four models tested and benchmarked on Valeo fisheye dataset

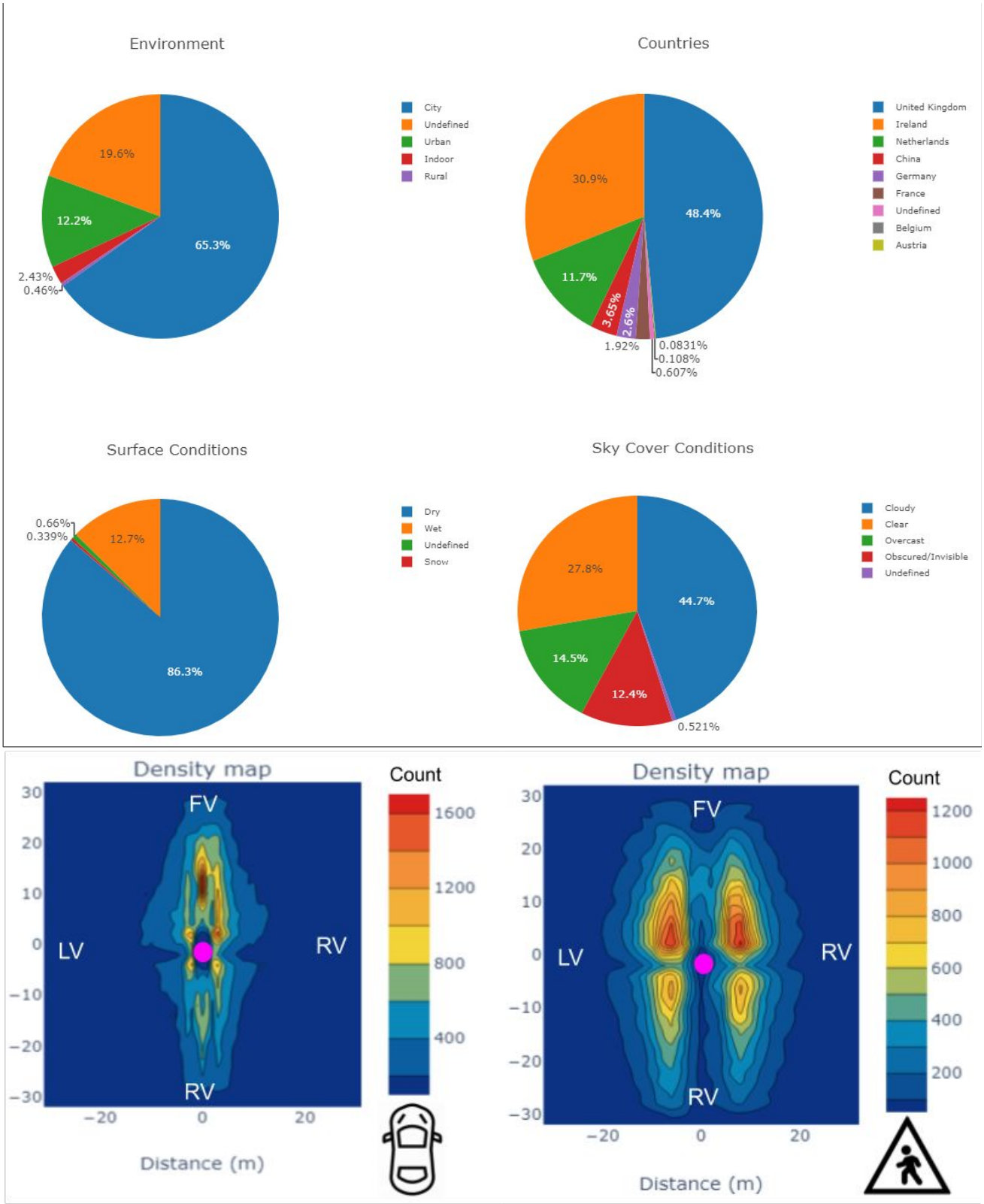

Fig. 10: Dataset Statistics: Pink Dot in density maps is an ego vehicle center. FV: Front View, RV: Rear View, MLV: Mirror Left View, and MRV: Mirror Right View w.r.t the ego vehicle. The pie charts show the diversity in geographical and climatic conditions.