RotInvMTL: Rotation Invariant MultiNet on Fisheye Images for Autonomous Driving Applications

Bruno Arsenali
Valeo Vision Systems
Ireland
bruno.arsenali@valeo.com

Prashanth Viswanath
Valeo Vision Systems
Ireland
prashanth.viswanath@valeo.com

Jelena Novosel
Valeo Vision Systems
Ireland
jelena.novosel@valeo.com

Abstract

Precise understanding of the scene around the car is of the utmost importance to achieve autonomous driving. Convolutional neural networks (CNNs) have been widely used for road scene understanding in the last few years with great success. Surround view (SV) systems with fisheye cameras have been in production in various cars and trucks for close to a decade. However, there are very few CNNs that are employed directly on SV systems due to the fisheye nature of its cameras. Typically, correction of fisheye distortion is applied to the data before it is processed by the CNNs, thereby increasing the system complexity and also reducing the field of view (FOV). In this paper, we propose RotInvMTL: a multi-task network (MTL) to perform joint semantic segmentation, boundary prediction, and object detection directly on raw fisheye images. We propose a rotation invariant object detection decoder that adapts to fisheye distortion and show that it outperforms YOLOv2 by 9% mAP. By combining the MTL outputs, an accurate foot-point information and a rough instance level segmentation may be obtained, both of which are critical for automotive applications. In conclusion, RotInvMTL is an efficient network that performs well for autonomous driving applications.

1. Introduction

Accurate and real-time understanding of the complex and dynamic environment around the ego-vehicle is of the utmost importance for autonomous cars and advanced driver-assistance systems (ADASs). At the present time, the state-of-the-art approaches for visual perception are based on convolutional neural networks (CNNs). For example, CNNs are used to solve object detection [24] and semantic segmentation [23] in real-time. They are also used to solve instance segmentation [14], however inference in real-time remains challenging due to the complexity of the solution and the limitations of the hardware.

Most of the state-of-the-art ADASs are vision based. Generally, an ego-vehicle contains a surround view (SV) system, which consists of four to six fisheye cameras that provide a 360 degree view (e.g., [35]). Each camera has a field of view (FOV) of at least 180 degrees that enables perception of objects in the close proximity to the vehicle. Since the state-of-the-art approaches for visual perception (e.g., [21, 31, 17, 2]) are often evaluated on databases with rectilinear images such as Cityscapes [7], KITTI [11], and CamVid [4], it is not known how they perform on fisheye images, which are affected by radial distortions.

To avoid problems that may arise due to the fisheye lens distortions, within the context of CNNs, fisheye images are first corrected and then passed on to the CNN-accelerated hardware. Once the CNN processing is complete, the result is back-projected to the original image space and combined with the results from the other visual perception algorithms that operate directly on fisheye images. An example of such process is depicted in Figure 1. This typical algorithmic pipeline for the SV system contradicts the original intent of having a wide angle lens, as the correction step reduces the FOV. More importantly, two additional steps are introduced into the pipeline of CNN processing. These steps increase the system frame buffer requirements (the system has to hold two sets of images), memory bandwidth, and processing latency, all of which are undesirable and can become a bottleneck in achieving real-time inference.

Additional issues, related to fisheye images, arise when dealing with object detection annotations (i.e., bounding rectangles) and the corresponding foot-points. Foot-point is defined as a point of contact between the object and the ground and is considered to be the bottom center point of the bounding rectangle. Accurate foot-point is essential for safety applications (e.g., braking) as it is used to determine how far an object is from the ego-vehicle. Annotations of bounding rectangles, on a front view fisheye image, for the standard object detectors such as YOLO [24] and SSD [21] are given in Figure 2. These rectangles tend to overestimate the object size, especially on fisheye images, which leads to

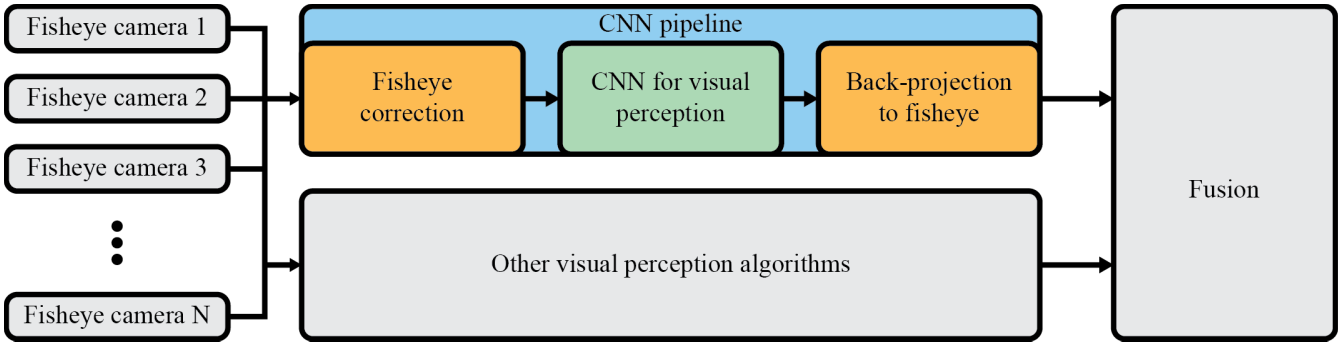


Figure 1: Typical processing block diagram for a surround view (SV) camera system. The convolutional neural network (CNN) algorithms for visual perception often have two additional steps (i.e., fisheye correction and back-projection to fisheye space). These steps are undesirable as they may become a bottleneck in achieving real-time performance.

an inaccurate foot-point location, and as such they are not suitable for object representation in the automotive industry. Instance segmentation provides high resolution information needed to improve the foot-point prediction, but it remains unfeasible for real-time applications.

In this paper, we propose RotInvMTL, a network that performs semantic segmentation, boundary prediction, and object detection directly on fisheye images in real-time. Two rotation-invariant detectors (i.e., YOLO-RotRect and YOLO-Circ) are explored to improve object representation. Additionally, boundary prediction is proposed with object detection to enable accurate foot-point prediction and to provide rough instance segmentation. The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the proposed multi-task network. Section 4 provides information related to experimental evaluation. Finally, concluding remarks are given in Section 5.

2. Related Work

An overview of semantic segmentation and boundary prediction is given in subsections 2.1 and 2.2, respectively. Details of existing approaches for object detection follow in subsection 2.3. A summary of multi-task learning is given in subsection 2.4. Finally, work related to CNN-based perception on fisheye images is presented in subsection 2.5.

2.1. Semantic Segmentation

Long et al. are the first to propose a fully CNN (F-CNN) for semantic segmentation [22], in which fully connected layers are transformed into convolutional layers. Due to the large receptive field of the network, precise localization of boundary remains a challenge. To overcome this, in [5], conditional random fields are applied to the output of the CNN. Unfortunately, this solution, compared to the F-CNN, requires more memory and additional computation time. In SegNet [2], new shortcut connections are introduced. The

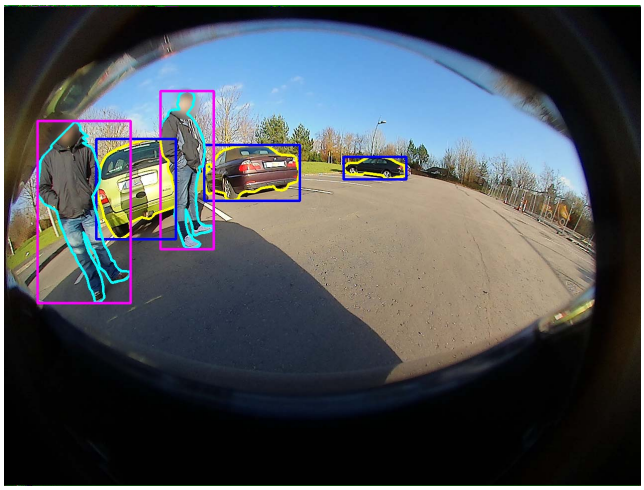


Figure 2: Ground truth annotations (bounding rectangles) in magenta (persons) and blue (vehicles) for standard object detectors (e.g., YOLO and SSD) on a fisheye image. The corresponding polygon annotations in cyan (persons) and yellow (vehicles) suggest that bounding rectangles tend to overestimate the object size and with that provide inaccurate foot-point information.

network requires less memory and has faster inference time than the F-CNN. Furthermore, Yu and Kotlun [34] propose dilated convolutions, which become the basis for DeepLab [5], whose newest version [6] is one of the best performing solutions for semantic segmentation. Unfortunately, many of the best performing solutions are far too complex to enable semantic segmentation in real-time for automotive industry. Due to this, less complex networks are proposed. Examples of these networks are ENet [23], MobileNet [30], ICNet [39], and ShuffleNet [38].

2.2. Semantic Boundary Prediction

In the recent years, boundary prediction has gained a lot of interest (e.g., [32], [36], and [1]). Xie et al. propose

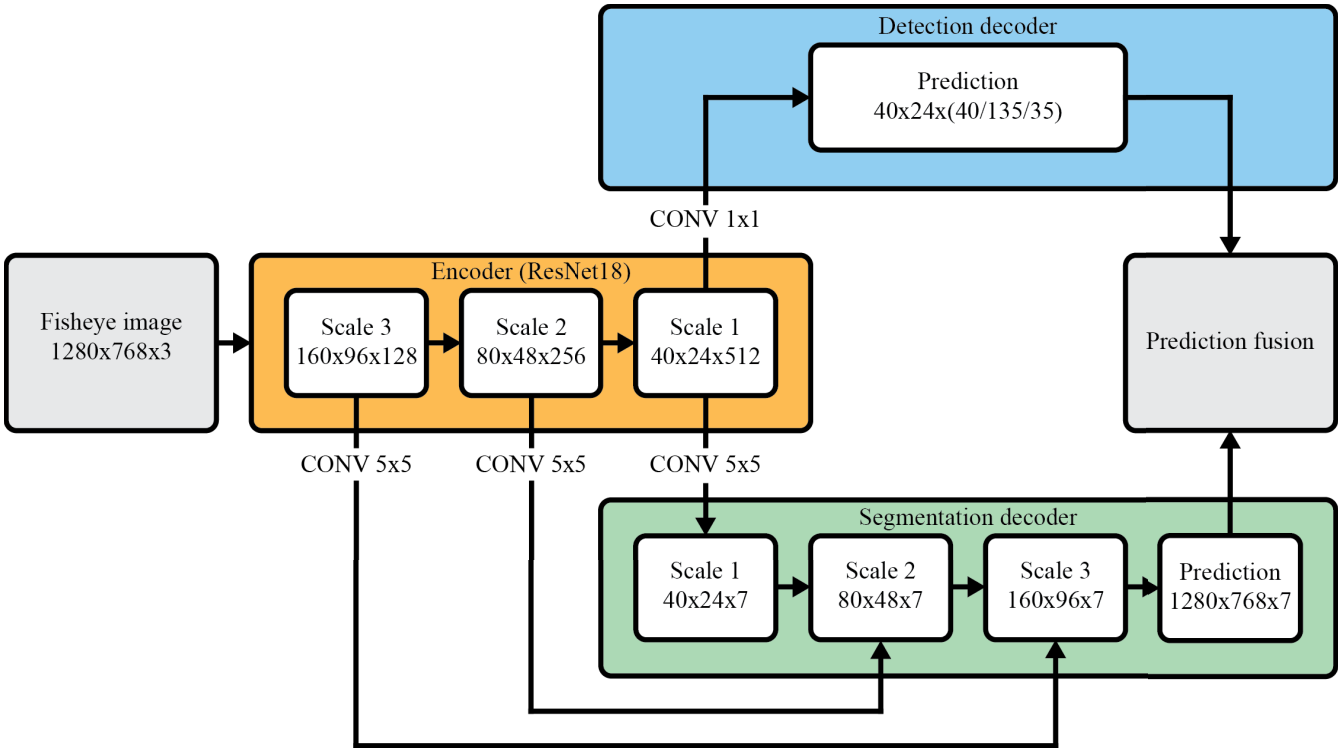


Figure 3: RotInvMTL architecture. The network operates on fisheye images and consists of a shared encored and two task specific decoders (one for object detection and the other for boundary-aware semantic segmentation). Each task outputs a prediction, and the predictions are fused to provide the final result.

CNN based class-agnostic edge detector for prediction of semantic boundary [32]. CASENet improves initial results by combining low and high-level features with a multi-label loss function [36]. Finally, in STEAL, a new thinning layer and a new loss are proposed [1], which further improves the results of boundary prediction.

2.3. Object Detection

Object detectors such as R-CNN [13] follow a two step approach: region proposal and scoring. Due to this, they are not suitable for real-time applications. Faster versions of the detectors are developed first by feeding an image directly to the region proposal CNN [12] and later by eliminating the need for the selective search [27]. Despite the proposed improvements, larger speed-related gains were not observed until the appearance of single stage detectors such as YOLO [24] (improved versions are described in [25] and [26]) and SSD [21]. These detectors are specially designed to process rectilinear images and as such may not be suitable to operate on fisheye images. Rotation-invariant detectors such as [20] seem more appropriate. To the best of our knowledge, they are not yet applied on fisheye images.

2.4. Multi-task Learning (MTL)

Multi-task learning (MTL) is a sub-field of machine learning in which shared domain information is used to improve generalization of complementary tasks that are learned in parallel. In other words, knowledge gained while training one task can be leveraged when training other tasks. As such, MTL can be viewed as a type of transfer learning.

In the context of CNNs, many existing methods use MTL to solve visual perception tasks. For example, Eigen et al. [10] propose a framework to predict semantic labels, depth, and surface normals. Teichmann et al., in MultiNet [31], present an approach for real-time joint semantic reasoning. UberNet [19] is a unified network that tackles a broad set of visual perception tasks. Kendall et al. [17] propose an architecture that combines semantic segmentation, instance segmentation, and depth prediction. The aforementioned methods process only rectilinear images. Hence, it is not yet known if they are suitable for use with other image types.

2.5. CNN-Based Perception on Fisheye Images

Visual perception is rarely explored on fisheye images due to the lack of a large-scale annotated dataset. The main focus of prior work is on semantic segmentation. Deng et al. propose an architecture for processing of fisheye images and train it on Cityscapes dataset warped by rectilinear-to-fisheye transformation [9, 8]. Blott et al. improve this transformation to achieve better performance [3]. Sàez et al. propose a network for real-time semantic segmentation

[29, 28]. Other examples of visual perception on fisheye images include parking slot detection [37] and autonomous driving model that integrates SV camera information with a route planner [16]. To the best of our knowledge, there are no approaches that perform joint object detection, semantic segmentation, and boundary prediction on fisheye images.

3. RotInvMTL

We propose a RotInvMTL architecture, which is depicted in Figure 3, to perform semantic segmentation, boundary prediction, and object detection in parallel on fisheye images. The first two tasks are merged to produce boundary-aware semantic segmentation. The network is trained in an end-to-end fashion and inference is done on fisheye images, without the need for distortion correction and back-projection, which are depicted in Figure 1. This results in reduced computation time and memory usage.

The network architecture consists of a shared encoder and two task specific decoders (one for semantic segmentation and the other for object detection). We formulate boundary prediction as semantic segmentation. For object detection, we explore two rotation-invariant detectors (i.e., YOLO-RotRect and YOLO-Circ). Subsection 3.1 provides details on the shared encoder. This is followed by subsections 3.2 and 3.3, which provide details on the semantic segmentation and object detection decoders, respectively.

3.1. Shared Encoder

The task of a shared encoder is to extract features from fisheye images that contain information required to perform segmentation, boundary prediction, and object detection. In RotInvMTL, we use ResNet18 [15]. This encoder provides a good balance between capacity and complexity which makes it suitable for real-time applications. The output feature maps of the final convolutional layer and the intermediate hidden layers are passed on to the segmentation and detection decoders as depicted in Figure 3.

3.2. Segmentation Decoder

A single decoder based on the architecture in [22] is used for boundary-aware semantic segmentation. Given that the features produced by the encoder are 32 times smaller than the image resolution, we use five transpose convolution layers in the decoder. Skip connections are used to extract high resolution features from the lower layers of the encoder. 5×5 convolutions are used instead of 1×1 convolutions, since they result in better contextual information. The following classes are used for semantic segmentation: road, lane markings, and curb. Two additional classes are added in the segmentation task (i.e., boundaries of persons and vehicles) to provide rough instance-level segmentation. An example of a segmentation mask is shown in Figure 4.

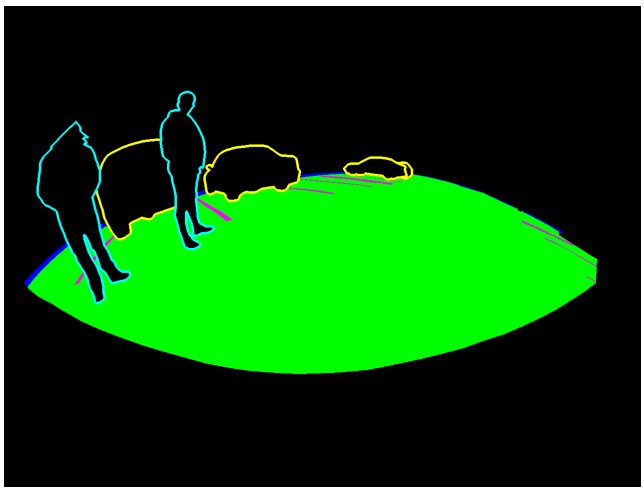


Figure 4: Ground truth segmentation for the image in Figure 2. Person boundaries, vehicle boundaries, curb stones, lane markings, and road surface are depicted in cyan, yellow, blue, magenta, and green, respectively.

3.3. Object Detection Decoders

To improve object representation on fisheye images, two new object detectors are proposed: YOLO-RotRect and YOLO-Circ. They are derived from YOLO-Rect, which is derived from the second version of YOLO (i.e., YOLOv2) [25]. To reduce the computational time of YOLO-Rect, a decoder with only a single 1×1 convolutional layer is used. It transforms the output of the final encoder layer into a tile-based representation for object detection. Each tile has dimensions of 32×32 pixels, which results in 40×24 tiles for an input image size of 1280×768. Details on the decoders are provided in the following subsections: 3.3.1 (YOLO-Rect), 3.3.2 (YOLO-RotRect), and 3.3.3 (YOLO-Circ).

3.3.1 YOLO-Rect

YOLO-Rect is a standard object detector [25]. It predicts class probabilities and rectangle parameters for each anchor. The softmax function is used to predict the former for persons, riders and vehicles. For the latter, we predict:

$$c_b = \sigma(c_u) \tag{1}$$

$$x_b = \sigma(x_u) + x_t \tag{2}$$

$$y_b = \sigma(y_u) + y_t \tag{3}$$

$$w_b = w_a e^{w_u} \tag{4}$$

$$h_b = h_a e^{h_u} \tag{5}$$

where c denotes the confidence, $\sigma(\cdot)$ denotes the sigmoid function, x and y denote the coordinates of the rectangle center, x_t and y_t denote the coordinates of the top left tile corner, w and h denote the rectangle width and height, w_a and h_a denote the anchor width and height. Finally,

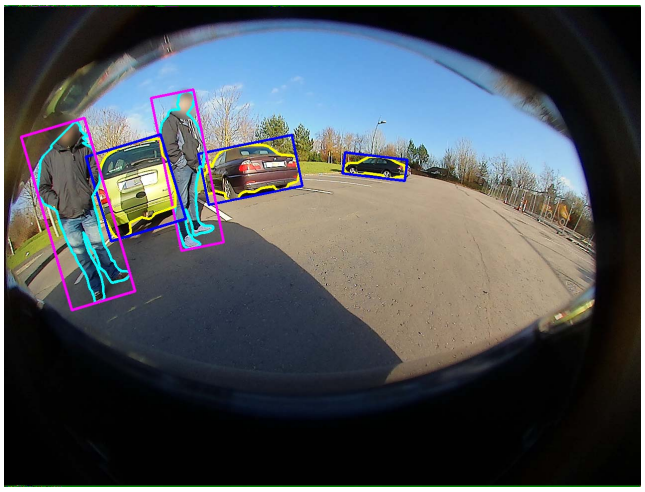


Figure 5: Ground truth annotations (rotated rectangles) in magenta (persons) and blue (vehicles) for YOLO-RotRect on a fisheye image. The corresponding polygon annotations in cyan (persons) and yellow (vehicles) suggest that rotated rectangles provide more accurate estimation of the object size and foot-point than standard rectangles in Figure 2.

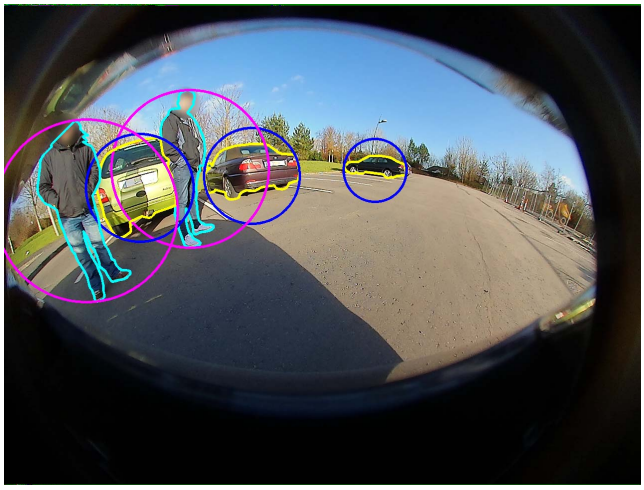


Figure 6: Ground truth annotations (bounding circles) in magenta (persons) and blue (vehicles) for YOLO-Circ on a fisheye image. The corresponding polygon annotations are depicted in cyan (persons) and yellow (vehicles). YOLO-Circ is explored in this study as a simple rotation-invariant alternative to YOLO.

bounded and unbounded predictions are denoted by b and u, respectively. In this study, we predict five anchors per tile, which results in 40 parameters per tile. Anchor parameters are predicted as in [25]. For a detailed description of the equations 1-5, we refer the reader to [25].

3.3.2 YOLO-RotRect

We propose a new object detector (i.e., YOLO-RotRect) that predicts a rotation angle for each bounding rectangle in addition to the standard parameters. Hence, YOLO-RotRect is rotation-invariant. Figure 5 shows an example of annotations for this detector.

YOLO-RotRect predicts class probabilities in the same way as YOLO-Rect. Furthermore, the same equations as in YOLO-Rect are used to predict the width, height, and both coordinates (i.e., x and y) of the rotated rectangle. Additionally, the rotation angle is predicted as:

$$a_b = (\sigma(a_u) - 0.5)\pi \tag{6}$$

where a denotes the angle. We define the angle with respect to the width of the rotated rectangle and the width is defined as its larger side. All angles are in the range from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

The YOLO procedure for anchor estimation described in [25] was modified to account for the rotation. First, we bin rectangles according to the angle into the following bins: $[-\frac{\pi}{2}, -\frac{\pi}{6}]$, $[-\frac{\pi}{6}, \frac{\pi}{6}]$, and $[\frac{\pi}{6}, \frac{\pi}{2}]$. Then, to estimate the width and the height of a total of five anchors per bin, we apply the clustering procedure described in [25] while ignoring the angles. Finally, to each anchor we assign an angle equal to the mean of the corresponding bin limits (i.e., $-\frac{\pi}{3}$, 0, and $\frac{\pi}{3}$). As we use a total of 15 anchors (three bins and five anchors per bin) in our experiments, the output of YOLO-RotRect has 135 parameters per tile.

YOLO-RotRect provides a good object representation on fisheye images (e.g., Figure 5) and with that accurate foot-point information. However, the complexity is larger due to the rotation and additional anchors. Hence, we also propose a simpler rotation-invariant detector (i.e., YOLO-Circ).

3.3.3 YOLO-Circ

As a simple alternative to YOLO-RotRect, we propose an object detector that predicts bounding circles (i.e., YOLO-Circ). This detector is also rotation-invariant, but estimates a smaller number of parameters. The annotations for this type of detector are presented in Figure 6.

YOLO-Circ predicts class probabilities in the same way as YOLO-Rect. Furthermore, the confidence and the circle center are also predicted in the same way (i.e., equations 1-3). YOLO-Circ does not predict the width and the height. Instead, it predicts the radius:

$$r_b = r_a e^{r_u} \tag{7}$$

where r and r_a denote the radius of circle and anchor, respectively. Hence, YOLO-Circ has less parameters per prediction than YOLO-Rect and YOLO-RotRect. To estimate the anchor radius, we used the procedure described in [25], where instead of estimating the width and the height we estimated the radius. In our experiments, we use five anchors

per tile, which results in 35 parameters per tile.

YOLO-Circ is specially designed to operate on fisheye images. Despite this, accurate estimation of foot-point is a challenge since YOLO-Circ overestimates the object size, similar to YOLO-Rect.

4. Experimental Evaluation

This section is structured as follows. First, we provide details regarding the dataset in subsection 4.1. Then, training details and metrics details are provided in subsections 4.2 and 4.3, respectively. Finally, results and discussion are given in subsection 4.4.

4.1. Dataset

To the best of our knowledge, there is no suitable public dataset to evaluate the proposed method. Consequently, RotInvMTL is evaluated on a proprietary dataset with fisheye images. A subset of this dataset is described in [33]. In this study, we use a larger subset of 47543 images with a resolution of 1280×768 from front, rear, left, and right cameras. Each image comes with instance level semantic annotations of 40 classes. In the present study, we use a subset of six classes: road, lane marking, curb, person, two/four wheeled vehicle, and rider. An 80-10-10 split is used to divide the images into training, validation and test datasets.

4.2. Training Details

The RotInvMTL network is trained from scratch (i.e., we do not use any pre-training). As the SV systems use the YUV format more often than the RGB format, we use the former format to train the network. To reduce system complexity and improve the generalization of the network, we train it with the images from different cameras (i.e., front, rear, left, and right). The network is trained jointly for object detection and boundary-aware semantic segmentation in an end-to-end fashion. As the dataset is annotated at instance level, we use the same images to train the two tasks. Detection and segmentation tasks have weights of 1 and 250, respectively. Brightness augmentation and horizontal flipping are used randomly to increase the complexity and size of the training data.

The cross-entropy loss is used to train the segmentation task. For the object detection task, we use the YOLOv2 loss [25]. Non-maximum suppression (NMS) is used for each class of the detection output to remove redundant rectangles and circles. Adam optimizer [18] with a learning rate of $5e^{-4}$ is used in combination with a linear learning rate decay strategy. The network is trained for 60 epochs.

4.3. Evaluation Metrics

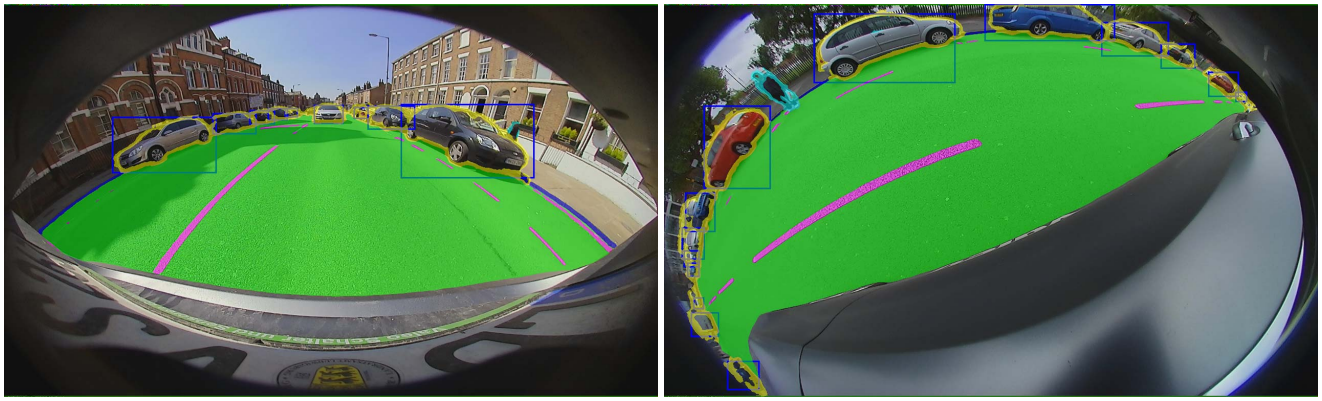
To evaluate the performance of object detection, we use per-class and mean (m) average precision (AP), per-class precision, and per-class recall. The standard approach to compute all of them is to use intersection over union (IoU) between the ground truth and the prediction. If this IoU is above a predefined threshold, the prediction is considered to be true, otherwise it is false. In the standard approach, the ground truth has the same shape as the prediction (e.g., rectangle and circle). The problem with this is that such ground truth overestimates the size of an object, especially on raw fisheye images. To account for this, we use the instance of an object (i.e., polygon annotation) as ground truth for the IoU calculation. We report the results for both standard and non-standard metrics. Finally, to evaluate the performance of boundary-aware semantic segmentation we use only the standard metrics: average and per-class accuracy, precision, recall, F1-score, and mIoU.

4.4. Results and Discussion

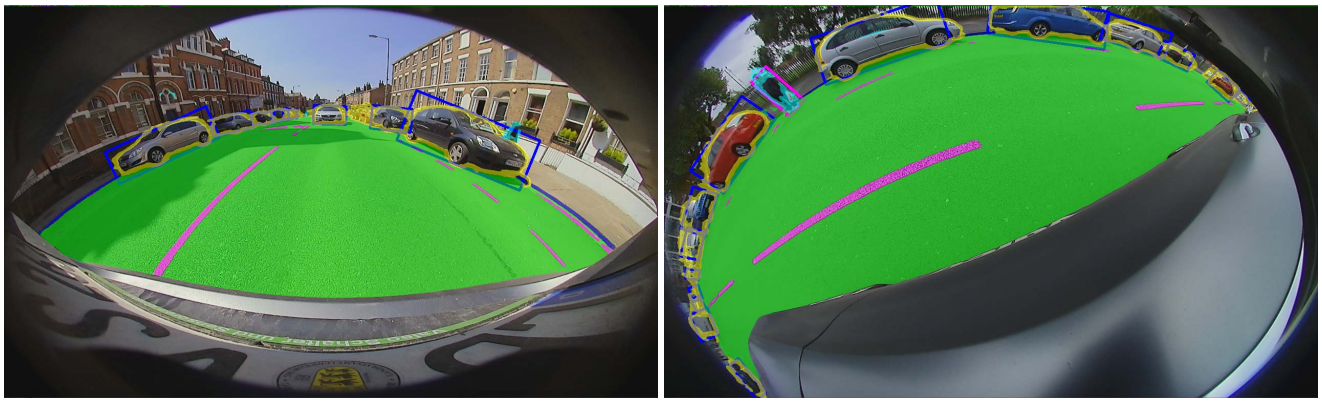
Various experiments are done to evaluate the performance of the proposed method and the results for both boundary-aware semantic segmentation and object detection are reported and discussed. The evaluation is performed on the test dataset. Although the network is trained for 60 epochs, the evaluation results are provided for the epoch with the minimum loss on the validation dataset. Examples of RotInvMTL results are shown in Figure 7. The results are reported for all three object detectors (i.e., YOLO-Rect, YOLO-RotRect, and YOLO-Circ).

Table 1 shows results for boundary-aware semantic segmentation. The performance is similar for all three approaches (e.g., a difference of 0.14% is observed between the model with the best and the worst mIoU). We hypothesize the results are comparable since the segmentation decoder architecture remains the same. The results also show a higher per-class accuracy for the vehicle boundary when compared to the person boundary in all three cases (e.g., 82.87% vs. 73.24% for YOLO-Rect). This is probably caused by the vehicle size, which is often larger than the pedestrian size.

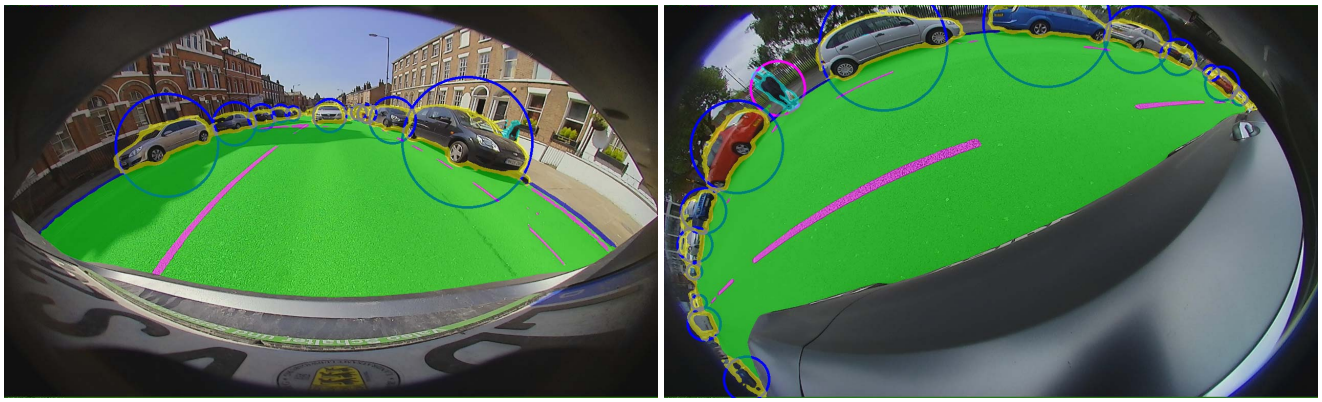
Object detection results for the standard and non-standard metrics are reported in Table 2 and Table 3, respectively. For all object detection metrics calculations, an IoU threshold of 0.5 is used. To calculate per-class precision and recall, a confidence threshold of 0.5 is used. Standard metrics report YOLO-Circ as the best performing decoder for most of the reported metrics. In contrast, according to the non-standard metrics, YOLO-Circ is the worst performing decoder. This indicates that the standard metrics may not be optimal to measure performance on fisheye images. Interestingly, results for YOLO-RotRect remain similar for both standard and non-standard metrics (absolute difference of up to 2% between the corresponding metrics) which suggests rotated rectangles provide a good representation for objects on fisheye images and do not overestimate their size.



(a) YOLO-Rect predictions for the rear (left) and mirror (right) camera image.



(b) YOLO-RotRect predictions for the rear (left) and mirror (right) camera image.



(c) YOLO-Circ predictions for the rear (left) and mirror (right) camera image.

Figure 7: Examples of RotInvMTL results for three different object detection decoders. Object detection outputs are denoted in blue (vehicles) and magenta (persons), whereas boundary-aware semantic segmentation outputs are denoted in cyan (person boundaries), yellow (vehicle boundaries), blue (curb stones), magenta (lane markings), and green (road surface).

Foot-point information of an object can be obtained by combining object detection, semantic segmentation and boundary prediction results. Semantic segmentation of the road provides ground information, object detection provides information about the individual objects, whereas boundary prediction gives information about the position of objects within the bounding shape (e.g., rectangle or circle). Generally, predicted boundary intersects the bounding shape. In the case of YOLO-Circ, the boundary intersect the circle at points located on its opposite sides (e.g., for person, the boundary intersects the circle at the head and the foot or for vehicle, the boundary intersects the circle at the opposite sides of a vehicle). This can be observed in Figure 7c. The intersection can further be combined with the segmentation of road to obtain the foot-point information of an object. Similary to YOLO-Circ, in the case of YOLO-

Table 1: Semantic segmentation results on the test dataset. Average accuracy, precision, recall, F1-score, mean intersection over union (mIOU), and per-class accuracy are reported.

	Avg. Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	mIoU (%)	Per-Class Accuracy (%)				
						Road	Lanes	Curb	Person boundary	Vehicle boundary
YOLO-Rect (baseline)	97.17	76.09	67.60	68.17	65.66	97.33	70.38	58.36	73.24	82.87
YOLO-RotRect	97.15	76.20	67.36	68.07	65.54	97.25	70.57	59.11	74.41	82.19
YOLO-Circ	97.16	76.17	67.28	68.07	65.52	97.29	71.16	58.34	74.15	82.49

Table 2: Object detection results on the test dataset, computed using rectangle or circle ground truth. Mean average precision (mAP), per-class average precision (AP) computed with IoU threshold of 50%, and per-class precision and recall computed with IoU and confidence threshold of 50% are reported.

	mAP (%)	Per-Class AP (%)			Per-Class Precision (%)			Per-Class Recall (%)		
		Person	Rider	Vehicle	Person	Rider	Vehicle	Person	Rider	Vehicle
YOLO-Rect (baseline)	46.97	42.23	34.49	64.20	47.84	50.74	67.39	56.14	36.99	65.59
YOLO-RotRect	40.74	33.82	26.47	61.91	44.62	52.72	70.36	49.06	23.37	60.85
YOLO-Circ	51.00	50.57	37.32	65.10	54.63	54.44	69.61	60.12	39.16	64.47

Table 3: Object detection results on the test dataset, computed using polygon object ground truth. Mean average precision (mAP), per-class average precision (AP) computed with IoU threshold of 50%, and per-class precision and recall computed with IoU and confidence threshold of 50% are reported.

	mAP (%)	Per-Class AP (%)			Per-Class Precision (%)			Per-Class Recall (%)		
		Person	Rider	Vehicle	Person	Rider	Vehicle	Person	Rider	Vehicle
YOLO-Rect (baseline)	31.55	19.36	28.38	46.91	32.35	46.78	56.64	37.96	34.10	55.12
YOLO-RotRect	40.84	32.62	28.44	61.45	43.74	53.26	69.80	48.10	23.61	60.36
YOLO-Circ	10.81	2.11	16.75	13.58	11.14	36.18	30.41	12.26	26.02	28.17

RotRect, the intersection of the rotated rectangle and the predicted boundary occurs on the opposite sides of the rectangle, as shown in Figure 7b. Thus, similar deductions, as in the case of YOLO-Circ, can be used to obtain foot-point information. Unfortunately, these deductions are not applicable to YOLO-Rect as the intersection of the boundary does not occur on the opposite sides of the rectangle which can be seen in Figure 7a.

Our approach provides a rough instance segmentation for vehicles and persons. Similarly to foot-point information, each bounding shape marks an individual object, whereas the predicted boundary denotes the position of that object. In other words, within the bounding shape, the boundary of an object can be traced. This becomes invalid in case of an occlusion, where a single object becomes separated. Hence, additional work is required to further improve the proposed version of rough instance segmentation.

5. Conclusion

In this paper, we propose RotInvMTL, an efficient CNN architecture, which jointly performs semantic segmentation, object detection and boundary prediction on fisheye images. Two approaches, YOLO-RotRect and YOLO-Circ, are proposed to improve object detection. Further, by integrating the outputs of the network, a reliable foot-point information and a rough instance segmentation can be obtained. The proposed approach provides a simple solution that is trained in an end-to-end fashion. Overall, RotInvMTL is a very efficient network with reduced system complexity that performs well for autonomous driving and parking applications.

Acknowledgement

We would like to thank our colleagues from the Valeo Deep Learning team for their support.

References

[1] D. Acuna, A. Kar, and S. Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11075–11083, 2019.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] G. Blott, M. Takami, and C. Heipke. Semantic segmentation of fisheye images. In *The European Conference on Computer Vision (ECCV)*, 2018.

[4] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] L. Deng, M. Yang, H. Li, T. Li, B. Hu, and C. Wang. Restricted deformable convolution based road scene semantic segmentation using surround view cameras. *arXiv preprint arXiv:1801.00708*, 2018.

[9] L. Deng, M. Yang, Y. Qian, C. Wang, and B. Wang. CNN based semantic segmentation for urban traffic scenes using fisheye camera. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 231–236, 2017.

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The Kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[12] R. Girshick. Fast R-CNN. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1448, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] S. Hecker, D. Dai, and L. Van Gool. Learning driving models with a surround-view camera system and a route planner. *arXiv preprint arXiv:1803.10158*, 2018.

[17] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] I. Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6129–6138, 2017.

[20] L. Liu, Z. Pan, and B. Lei. Learning a rotation invariant detector with rotatable bounding box. *arXiv preprint arXiv:1711.09405*, 2017.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *The European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[25] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

[26] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[27] 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 (NIPS)*, pages 91–99, 2015.

[28] Á. Sáez, L. M. Bergasa, E. López-Guillén, E. Romera, M. Tradacete, C. Gómez-Huélamo, and J. del Egido. Real-time semantic segmentation for fisheye urban driving images based on ERFNet. *Sensors*, 19(3):503, 2019.

[29] Á. Sáez, L. M. Bergasa, E. Romeral, E. López, R. Barea, and R. Sanz. CNN-based fisheye image real-time semantic segmentation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1039–1044, 2018.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[31] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun. MultiNet: Real-time joint semantic reasoning for autonomous driving. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020. IEEE, 2018.

[32] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[33] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende, et al. WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving. *arXiv preprint arXiv:1905.01489*, 2019.

[34] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[35] M. Yu and G. Ma. 360 surround view system with parking guidance. *SAE International Journal of Commercial Vehicles*, 7(2014-01-0157):19–24, 2014.

[36] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam. CASENet: Deep category-aware semantic edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5964–5973, 2017.

[37] L. Zhang, J. Huang, X. Li, and L. Xiong. Vision-based parking-slot detection: A DCNN-based approach and a large-scale benchmark dataset. *IEEE Transactions on Image Processing*, 27(11):5350–5364, 2018.

[38] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[39] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. ICNet for real-time semantic segmentation on high-resolution images. In *The European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.