2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

OmniDet: Surround View Cameras based Multi-task Visual Perception Network for Autonomous Driving

Varun Ravi Kumar[1,5], Senthil Yogamani[2], Hazem Rashed[3], Ganesh Sitsu[2],
Christian Witt[1], Isabelle Leang[4], Stefan Milz[5] and Patrick Mäder[5]
[1]Valeo, Germany [2]Valeo, Ireland [3]Valeo, Egypt [4]Valeo, France [5]TU Ilmenau, Germany

***Abstract*—Surround View fisheye cameras are commonly deployed in automated driving for 360° near-field sensing around the vehicle. This work presents a multi-task visual perception network on unrectified fisheye images to enable the vehicle to sense its surrounding environment. It consists of six primary tasks necessary for an autonomous driving system: depth estimation, visual odometry, semantic segmentation, motion segmentation, object detection, and lens soiling detection. We demonstrate that the jointly trained model performs better than the respective single task versions. Our multi-task model has a shared encoder providing a significant computational advantage and has synergized decoders where tasks support each other. We propose a novel camera geometry based adaptation mechanism to encode the fisheye distortion model both at training and inference. This was crucial to enable training on the WoodScape dataset, comprised of data from different parts of the world collected by 12 different cameras mounted on three different cars with different intrinsics and viewpoints. Given that bounding boxes is not a good representation for distorted fisheye images, we also extend object detection to use a polygon with non-uniformly sampled vertices. We additionally evaluate our model on standard automotive datasets, namely KITTI and Cityscapes. We obtain the state-of-the-art results on KITTI for depth estimation and pose estimation tasks and competitive performance on the other tasks. We perform extensive ablation studies on various architecture choices and task weighting methodologies. A short video at https://youtu.be/xbSjZ5OfPes provides qualitative results.**

I. Introduction

Surround View fisheye cameras have been deployed in premium cars for over ten years, starting from visualization applications on dashboard display units to providing near-field perception for automated parking. Fisheye cameras have a strong radial distortion that cannot be corrected without disadvantages, including reduced field-of-view and resampling distortion artifacts at the periphery [1]. Appearance variations of objects are larger due to the spatially variant distortion, particularly for close-by objects. Thus fisheye perception is a challenging task, however it is relatively less explored despite its prevalence.

Autonomous Driving applications require various perception tasks to provide a robust system covering a wide variety of use cases. Alternate ways to detect objects in parallel are necessary to achieve a high level of accuracy. For example, objects can be detected based on appearance, motion, and depth cues. Despite increasing computation power in automotive embedded systems, there is always a need for efficient design due to the increasing number of cameras and perception tasks. Multi-task learning (MTL) is an efficient design pattern commonly used where most of the computation is shared across all the tasks [2], [3]. Furthermore, learning features for multiple tasks can act as a regularizer, improving generalization. Mao *et al.* [4] illustrated that multi-task learning improves adversarial robustness, which is critical for safety applications. In the automotive multi-task setting, MultiNet [5] was one of the first to demonstrate a three task network on KITTI, and further works have primarily worked on a three task setting.

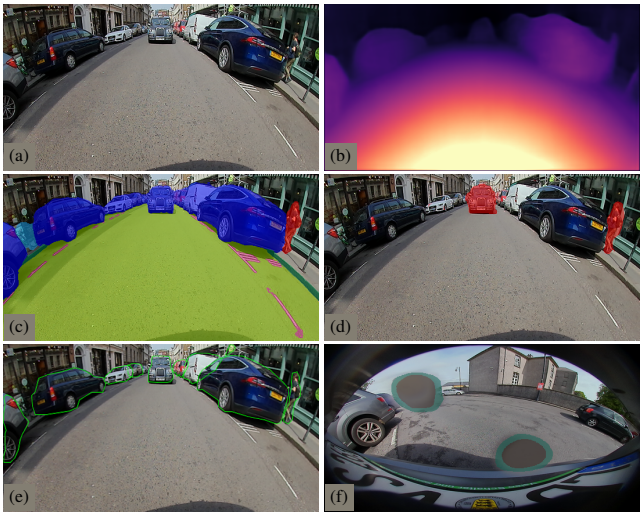

Fig. 1: **Results of our real-time OmniDet model on raw fisheye images.** (a) Rear-Camera Input Image, (b) Distance Estimation, (c) Semantic Segmentation, (d) Motion Estimation, (e) 24-sided Polygon based Object Detection and (f) Soiling Segmentation (asynchronous).

This work demonstrates a multi-task perception model for the six essential perception tasks on unrectified fisheye images (shown in Fig. 1). As we discuss a full perception system building upon a lot of our previous work, it is difficult to cover all the technical details, and we have focused on the main contributions. Our contributions are as follows:

- We demonstrate the first real-time six-task model for surround-view fisheye camera perception.
- We propose a novel camera tensor representation of radial distortion to enable the adaptation of CNN for the 12 different camera models in the WoodScape dataset.
- We propose novel design techniques, including VarNorm task weighting.
- We design synergized decoders where different tasks help each other in addition to a shared encoder.
- We showcase a six-task model on WoodScape and five-task model on KITTI and Cityscapes performing better

Manuscript received: Oct, 15th, 2020; Revised Jan, 9th, 2021; Accepted Feb, 3rd, 2021. This paper was recommended for publication by Editor Eric Marchand upon evaluation of the Associate Editor and Reviewers' comments.
Digital Object Identifier (DOI): see top of this page.

than the single task baselines.
- We obtain state-of-the-art results for depth and pose estimation tasks on KITTI among monocular methods.

II. Perception tasks and Losses in our MTL

Our goal is to build a multi-task model covering the necessary modules for near-field sensing use cases like Parking or Traffic Jam assist. This paper builds upon our previous papers focused on individual tasks, and we mainly discuss the new improvements. In the following sub-sections, we refer the reader to these papers for more details and a literature review. In general, there is minimal work in the area of fisheye perception. Specifically, there is only our previous work on multi-task learning: FisheyeMultiNet [6] which discusses a more straight forward three task network.

The perception system comprises semantic tasks, geometric tasks, and lens soiling detection. The standard semantic tasks are object detection (pedestrians, vehicles, and cyclists) and semantic segmentation (road, lanes, and curbs). Fisheye cameras are mounted low on a vehicle and are susceptible to lens soiling due to splashing of mud or water from the road. Thus, it is vital to detect soiling on the camera lens and trigger a cleaning system [7]. The semantic tasks typically require a large annotated dataset covering various objects. It is infeasible to practically cover every possible object. Thus, generic object detection using geometric cues like motion or depth for rare objects is typically used. They will also complement the detection of standard objects and provide more robustness. Thus we propose to include motion segmentation and depth estimation tasks. Motion is a dominant cue in automotive scenes, and it requires at least two frames or the use of dense optical flow [8]. Self-supervised methods have recently dominated depth estimation, which has also been demonstrated on fisheye images [9]. Finally, the visual odometry task is required to place the detected objects in a temporally consistent map.

A. Self-Supervised Distance and Pose Estimation Networks

We set up a self-supervised monocular structure-from-motion (SfM) framework using our previous work FisheyeDistanceNet [9] for distance and pose estimation and perform view synthesis by incorporating the polynomial projection model function. The total loss comprised of a reconstruction matching term $\mathcal{L}_r$, regularization term $\mathcal{L}_s$, which enforces edge-aware smoothness inside the distance map $\hat{D}_t$ as introduced in [10]. Additionally, the cross-sequence distance consistency loss $\mathcal{L}_{dc}$ and the scale recovery technique were used. We discuss the new improvements in the following paragraphs, which led to significant improvement in accuracy.

We incorporate feature-metric losses from [11] where $\mathcal{L}_{dis}$ and $\mathcal{L}_{cvt}$ are computed on I_t's feature representation, where we learn the features using a self-attention autoencoder. The main goal of these losses is to prevent the training objective from getting stuck at multiple local minima for homogeneous areas as fisheye images have considerably larger homogeneous areas than their rectilinear counterpart. It is essentially a loss function that penalizes small slopes and emphasizes the low-texture regions using the image gradients. The self-supervised loss landscapes are constrained to form proper convergence basins using the first-order derivatives to regularize the target features. However, merely imposing a discriminative loss cannot guarantee we move to the optimal solution during the gradient descent, since inconsistency exists among first-order gradients, *i.e.* spatially adjacent gradients point in opposite directions. Shu *et al.* [11] proposed a convergent loss to have a relatively large convergence radius to enable gradient descent from a remote distance. This is achieved by formulating the loss to have consistent gradients during the optimization step by encouraging the smoothness of feature gradients and large convergence radii accordingly. The total objective loss for distance estimation $\mathcal{L}_{dist}$ is

$$\begin{aligned}\mathcal{L}_{dist} = \mathcal{L}_r(I_t, \hat{I}_{t'\to t}) + \beta\ \mathcal{L}_s(\hat{D}_t) + \gamma\ \mathcal{L}_{dc}(\hat{D}_t, \hat{D}_{t'}) \\ + \mathcal{L}_r(\hat{F}_t, \hat{F}_{t'\to t}) + \omega\ \mathcal{L}_{dis}(I_t, \hat{F}_t) + \mu\ \mathcal{L}_{cvt}(I_t, \hat{F}_t)\end{aligned} \quad (1)$$

where β, γ, ω and μ weigh the distance regularization $\mathcal{L}_s$, cross-sequence distance consistency $\mathcal{L}_{dc}$, discriminative $\mathcal{L}_{dis}$ and convergent $\mathcal{L}_{cvt}$ losses respectively.

We calculate the image and feature reconstruction loss using the target I_t, estimated feature $\hat{F}_t$ frames, reconstructed target $\hat{I}_{t'\to t}$ and feature $\hat{F}_{t'\to t}$ frames. It is a linear combination of the general robust pixel-wise loss term [12] and the Structural Similarity (SSIM) [13] as described in [14].

B. Generalized Object Detection

The standard bounding box representation fails in fisheye cameras due to heavy radial distortion, particularly in the periphery. In concurrent work [15], we explored different output representations for fisheye images, including oriented bounding boxes, curved boxes, ellipses, and polygons. We have integrated this model in our MTL framework where we use a 24-sided polygon representation for object detection. We briefly summarize the details here and refer to our expanded paper [15] for more details on generalized object detection.

A polygon is a generic representation for any arbitrary shape; however, it is more expensive to annotate than a bounding box. The object contour can be uniformly sampled in the range of $360°$ split into N equal polygon vertices, each represented by the radial distance r from the object's centroid as used in PolyYOLO [16]. We observe that uniform sampling cannot efficiently represent high curvature variations in the fisheye image object contours. Thus, we make use of an adaptive sampling based on the curvature of the local contour. We distribute the vertices non-uniformly in order to optimally represent the object contour. We adopt the algorithm in [17] to detect the dominant points in a given curved shape, which best represents the object. Then we reduce the set of points using the algorithm in [18] to get the most representative simplified curves. We adapted the YOLOv3 [19] decoder to output polygons and other representations listed above for a uniform comparison.

C. Segmentation Tasks

Three of our tasks are modeled as segmentation problems. Semantic and soiling segmentation having seven and four output classes, respectively, on the WoodScape dataset. Motion

segmentation uses two frames and outputs a binary moving or static mask. During training, the network predicts the posterior probability Y_t which is optimized in a supervised fashion by *Lovasz-Softmax* [20] loss, and *Focal* [21] loss for handling class imbalance instead of the cross-entropy loss used in our previous work. We obtain the final segmentation mask M_t by applying a pixel-wise argmax operation on the posterior probabilities.

The soiling dataset is independently built, and thus it cannot be trained jointly in a traditional manner. Thus we freeze the shared encoder trained using five other tasks and train only the decoder for soiling. This demonstrates the potential of re-using the encoder features for additional tasks. We also trained soiling jointly using asynchronous backpropagation [22], but it achieved the same accuracy as using the frozen encoder. Compared to our previous work SoilingNet [7], we moved from the tiled output to pixel-level segmentation.

D. Joint Optimization

Balancing the task losses is an important problem for training multi-task models. Our contribution is two-fold. We evaluate various task weighting strategies for five tasks compared to the three tasks experiments in previous literature. We evaluate the uncertainty loss from Kendall [23], the gradient magnitude normalization GradNorm [24], the dynamic task prioritization DTP [25], the dynamic weight average DWA [26] and the geometric loss [27].

Secondly, we propose a novel method called **VarNorm** for variance normalization. It consists of normalizing each loss by its variance over the last n epochs. The loss weight of task i at epoch t is formulated as below:

$$w_i(t) = \frac{1}{\sigma_i(t-1)}, \sigma_i(t) = \frac{1}{n-1}\sum_{k=0}^{n-1}(L_i(t-k) - \overline{L_i})^2 \quad (2)$$

where $\overline{L_i}$ is the average of task loss i over the last n epochs. We chose $n = 5$. This method is motivated by the simple idea that the task loss values can be seen as a distribution whose dispersion is its variance. Variance normalization re-scales the dispersion between the different task loss distributions based on the previous n epochs. A large dispersion leads to a lower task weight, whereas a small dispersion to a higher one. Its final effect tends to homogenize the learning speed of tasks. As shown in Table III, equal weighting is the worst, and our multi-task network performs better than the single task networks by using any dynamic task weighting method presented above. We employ the proposed VarNorm method for all the further experiments as it achieved the best results.

III. Network Details of MTL OmniDet

Encoder-decoder architectures are commonly used for dense prediction tasks. We use this type of architecture as it easily extends to a shared encoder for multiple tasks. We design our encoder by incorporating vector attention based pairwise and patchwise self-attention encoders from [28]. These networks efficiently adapt the weights across both spatial dimensions and channels. We adapt the Siamese (twin network) approach for the motion prediction network, where we concatenate the source and target frame features and pass them to the super-resolution motion decoder. As the weights are shared in the Siamese encoder, the previous frame's encoder can be saved and re-used instead of re-computing. Inspired by [11], we develop an auxiliary self-attention auto-encoder for single-view reconstruction. We detail our main novel contributions in the next two sub-sections. Firstly, we employ our novel camera geometry tensor to handle multiple viewpoints and change in the camera's intrinsic distance estimation. Secondly, we employ synergized decoders via cross task connections to improve each other's performance.

A. Camera Geometry Tensor C_t

1) Motivation: In an industrial deployment setting, we target the design of a model that can be deployed in millions of vehicles each having its own set of cameras. Although the underlying camera intrinsics model is the same for a particular family of vehicles, there are variations due to manufacturing processes, which require the calibration of each camera instance. Even after deployment, calibration can vary due to high environmental temperature or due to aging. Thus a calibration adaptation mechanism in the model is essential. This contrasts with public datasets, which have a single camera instance for both the training and test dataset. In the Woodscape fisheye dataset [29], there are 12 different cameras with slight intrinsic variations to evaluate this effect. A single model for these four cameras instead of four individual models would also have several practical advantages such as (1) an improved efficiency on the embedded system requiring less memory and data rate to transmit, (2) an improved training by access to a larger dataset and regularization through different views, and (3) maintenance and certification of one model instead of four.

We propose converting all camera geometry properties into a tensor called camera geometry tensor C_t that is then passed to the CNN model to tackle this problem. The closest work is CAM-Convs [30], which uses camera-aware convolutions for pinhole cameras. We build upon this work and generalize to arbitrary camera geometries, including fisheye cameras.

2) Approach: We introduce the camera geometry tensor C_t in the mapping from RGB features to 3D information for the Self-Attention Network (SAN) encoder module, as shown in Fig. 2. It is included in each self-attention stage and also applied to every skip-connection. The camera geometry tensor C_t is formulated in a three-step process: For efficient training, the pixel-wise coordinates and angle of incidence maps are pre-computed. The normalized coordinates per pixel are used for these channels by incorporating information from the camera calibration. We concatenate these tensors and represent them by C_t and pass it along with the input features to our SAN *pairwise* and *patchwise* operation modules. It comprises six channels in addition to the existing decoder channel inputs. The proposed approach can, in principle, be applied to any fisheye projection model of choice explained in [1]. The different maps included in our shared self-attention encoder are computed using the camera intrinsic parameters, where the distortion coefficients a_1, a_2, a_3, a_4 are used to create the angle of incidence maps (a_x, a_y), c_x, c_y are used to compute

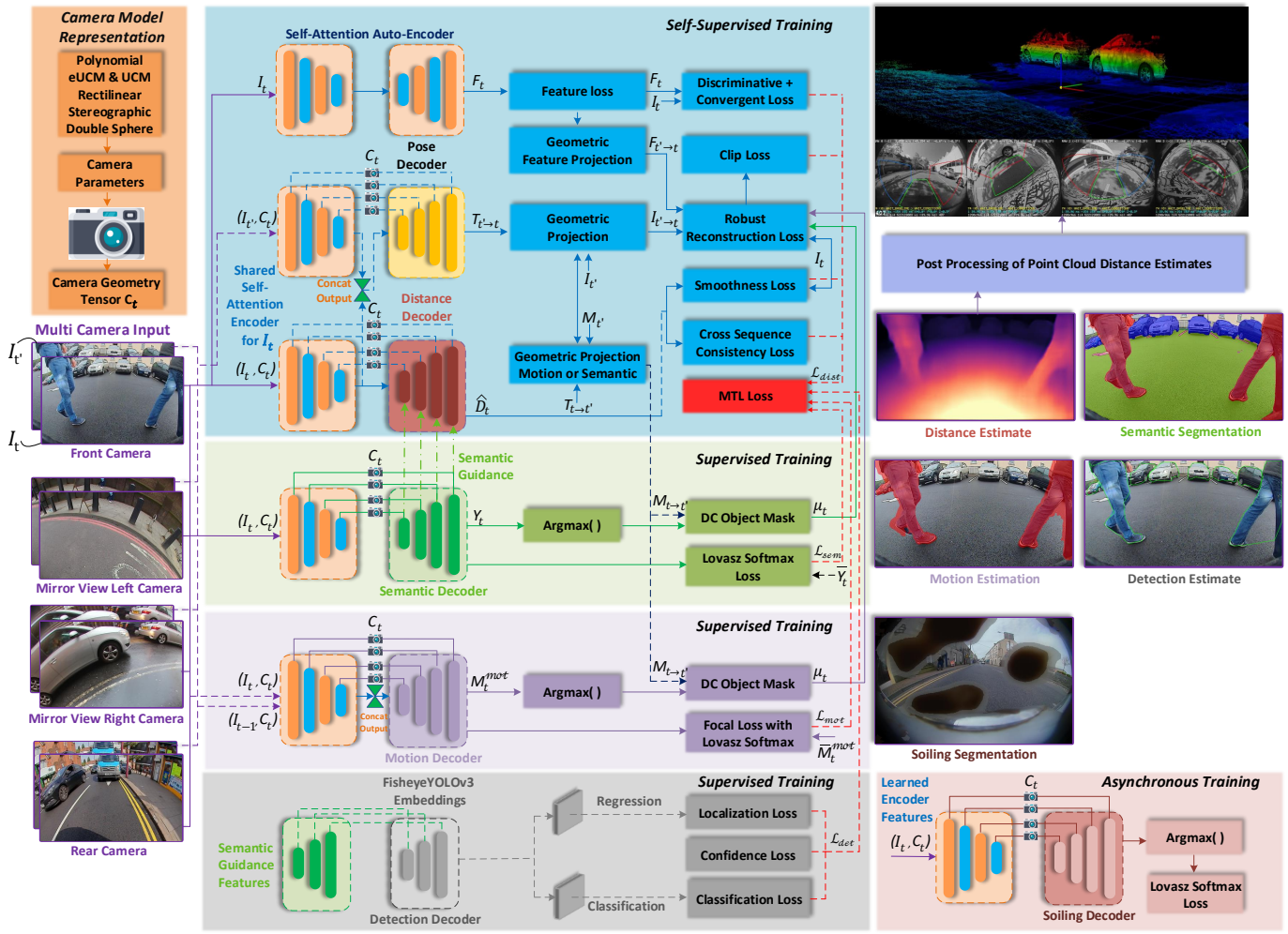

Fig. 2: **Overview of our Surround View cameras based multi-task visual perception framework.** The distance estimation task (blue block) makes use of semantic guidance and dynamic object masking from semantic/motion estimation (green and blue haze block) and camera-geometry adaptive convolutions (orange block). Additionally, we guide the detection decoder features (gray block) with the semantic features. The encoder block (shown in the same color) is common for all the tasks. Our framework consists of processing blocks to train the self-supervised distance estimation (blue blocks) and semantic segmentation (green blocks), motion segmentation (blue haze blocks), polygon-based fisheye object detection (gray blocks), and the asynchronous task of soiling segmentation (rose fog block). We obtain Surround View geometric information by post-processing the predicted distance maps in 3D space (perano block). The camera tensor C_t (orange block) helps our OmniDet yield distance maps on multiple camera-viewpoints and make the network camera independent.

the principal point coordinate maps (cc_x, cc_y) and the camera's sensor dimensions (width w and height h) are utilized to formulate the normalized coordinate maps.

3) Centered Coordinates (cc): Principal point position information is fed into the SAN's *pairwise* and *patchwise* operation modules by including the cc_x and cc_y coordinate channels centered at $(0, 0)$. We concatenate cc_x and cc_y by resizing them using bilinear interpolation to match the input feature size. We formulate cc_x and cc_y channels as:

$$cc_x = \begin{pmatrix} 0 - c_x \\ 1 - c_x \\ \vdots \\ w - c_x \end{pmatrix} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}^{\intercal}}_{(h+1)\times 1}, cc_y = \underbrace{\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}}_{(w+1)\times 1} \cdot \begin{pmatrix} 0 - c_y \\ 1 - c_y \\ \vdots \\ h - c_y \end{pmatrix}^{\intercal} \quad (3)$$

4) Angle of Incidence Maps (a_x, a_y): For the pinhole (rectilinear) camera model, the horizontal and vertical angle of incidence maps are calculated from the cc maps using the camera's focal length f: $a_{ch}[i, j] = \arctan(cc_{ch}[i, j]/f)$, where ch is either x or y (refer to Eq. 3). For the different fisheye camera models, the angle of incidence maps can analogously be deduced by taking the inverse of the radial distortion functions $r(\theta)$ explained in [1]. Specifically, for the polynomial model used in this paper, the angle of incidence θ is formulated by calculating the fourth order polynomial roots of $r(\theta) = \sqrt{x_I^2 + y_I^2} = a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4$ through a numerical method. We store the pre-calculated roots in a lookup table for all pixel coordinates to achieve training efficiency and create the a_x and a_y maps by setting $x_I = cc_x[i, j], y_I = 0$ and $x_I = 0, y_I = cc_y[i, j]$ respectively.

5) Normalized Coordinates (nc): Additionally, we add two channels of normalized coordinates [31], [30] whose values vary linearly between -1 and 1 with respect to the image coordinates. The channels are independent of the camera sensor's properties and characterize the spatial location in x and y directions. (e.g. a value of the $\hat{x}$ channel nearer to 1 indicates that the feature is nearer to the right border of the image).

B. Synergized Tasks

1) Dealing With Dynamic Objects and Solving Infinite Depth Issue: As dynamic objects violate the static world

assumption, information about their depth/distance is essential in autonomous driving; else, we would encounter the infinite depth issue during the inference stage and incur contamination to the reconstruction losses. We use the motion segmentation information to exclude potentially *moving* dynamic objects, while the distance is learned from *non-moving* dynamic objects. For this purpose, we define the pixel-wise mask μ_t, which contains a 1 if a pixel does not belong to a dynamic object from the current frame I_t and also not to a wrongfully projected dynamic object from the reconstructed frames $\hat{I}_{t'\to t}$ and a 0 otherwise. Accordingly, we predict a motion segmentation mask M_t^{mot} belonging to the target frame I_t, as well as motion masks $M_{t'}$ for the source frames $I_{t'}$. Dynamic objects inside the source frame are canonically detected inside M_t. However, to obtain the wrongfully projected dynamic objects, we need to warp the motion masks by nearest-neighbor sampling to the target frame, yielding projected motion masks $M_{t'\to t}$. We propose an alternative technique to enable the filtering of the dynamic objects if the motion segmentation task is unavailable. We leverage the semantic segmentation output and follow a similar method as described above. By defining the set dynamic object classes $\mathcal{S}_{\mathrm{DC}} \subset \mathcal{S}$, we can then reduce the semantic segmentation mask to a binary mask, fulfilling the above conditions. Its elements at location ij:

$$\mu_{t,ij} = \begin{cases} 1, \; M_{t,ij} \notin \mathcal{S}_{\mathrm{DC}} \; \wedge \; M_{t'\to t,ij} \notin \mathcal{S}_{\mathrm{DC}} \\ 0, \; \text{else} \end{cases} \tag{4}$$

Dynamic objects can be masked through a pixel-wise multiplication of the mask with the reconstruction loss for images and features.

2) Semantically-Guided Distance and Detection Decoder: Following our previous work [14], to better incorporate the semantic knowledge extracted from the segmentation branch of the multi-task network into the distance estimation, we incorporate it using pixel-adaptive convolutions [32] (PAC) to distill the knowledge from the semantic features into the distance decoder. This, in particular, breaks up the spatial invariance of the convolutions and allows the incorporation of location-specific semantic knowledge into the multi-level distance features. As shown in Fig. 2 (green block), the features are extracted at different levels of the segmentation decoder. Here, an input signal x to the pixel-adaptive convolution is processed as

$$x'_{ij} = \sum_{ab \in \mathcal{N}_k(ij)} K(F_{ij}, F_{ab}) W[r_{a-i,b-j}] x_{ab} + B \tag{5}$$

with pixel location ij, distance $r_{a-i,b-j}$ between pixel locations and a $k \times k$ neighbourhood window $\mathcal{N}_k(ij)$ around location ij. The elements of x inside the neighbourhood window $\mathcal{N}_k(ij)$ are respectively used as input to the convolution with weights W, bias $B \in \mathbb{R}^1$ and a kernel function K, used to calculate the correlation between the semantic guidance features $F \in \mathbb{R}^D$, extracted from the segmentation branch.

3) Linking Self-Attention and Semantic features to 2D detection: To leverage the multi-task learning setup, at first, we extract the Self-Attention Network (SAN) [28] encoder features and feed it as an input signal to the Eq. 5. We bypass the spatial information from the SAN encoder to the semantic decoder and fuse these features (skip-connections). Finally, we fuse these features and the detection decoder embeddings by applying PAC and obtaining content-agnostic features. This novel fusion technique in our OmniDet framework significantly improves the detection decoder's accuracy, which can be seen in Table I.

C. Implementation Details

We use Pytorch and employ a single stage learning process for our OmniDet framework to facilitate network optimization. We incorporate the recently proposed SAN in our encoder. The authors proposed two convolution variants, namely *pairwise* and *patchwise*. We mainly use patchwise but perform an ablation study on pairwise. We employ the Ranger (RAdam [33] + LookAhead [34]) optimizer to minimize the training objective function. We train the model for 20 epochs, with a batch size of 24 on a 24GB Titan RTX with an initial learning rate of 4×10^{-4} for the first 15 epochs, which is reduced to 10^{-5} for the last 5 epochs. The sigmoid output σ from the distance decoder is converted to distance with $D = m \cdot \sigma + n$, where m and n are chosen to constrain D between 0.1 and 100 units. Finally, we set β, γ, ω and μ to 10^{-3}. All images from the surround-view cameras with multiple viewpoints are shuffled thoroughly and fed to the distance and pose networks along with their respective intrinsics to create the camera geometry tensor C_t, as shown in Fig. 2, and described in Section III-A.

IV. Experimental Results

A. Datasets

We systematically train and test all our single and multi-task models on Woodscape [29], a Surround View fisheye dataset with annotations for multiple tasks and on the pinhole camera datasets KITTI [35] and Cityscapes [36].

1) WoodScape: The WoodScape dataset consists of 10,000 images split into training, validation, and test in a 6:1:3 ratio. Additional proprietary data was used for pre-training and initialization of our model. We train our 2D box detection on 5 most essential categories of objects — *pedestrians, vehicles, riders, traffic sign, and traffic lights*. Polygon prediction task on raw fisheye is limited to only 2 classes — *pedestrians, and vehicles*. Unlike traffic lights and traffic signs, these categories are non-rigid in nature and quite diverse in appearance, making them suitable for polygon regression. We sample 24 points with high curvature values from each object instance contour for the polygon regression task. Learning these points helps to regress better polygon shapes, as these points at high curvature define the shape of the object contours. We perform semantic segmentation on *road, lanes, and curbs categories*. Images are resized to 544×288 px from the native 1MP resolution.

2) Cityscapes: In the case of Cityscapes, we extracted the 2D boxes from the instance polygons. We train our OmniDet MTL model on 2,975 images in both single-task and multi-task settings. We report all our results on the validation split consisting of 500 images. Images are resized to 640×384 px for training and validation.

TABLE I: **Ablation study on the effect of our contributions** up to our final OmniDet model on the Fisheye Woodscape dataset [29].

Network	*Robust loss*	*Feature loss*	*Semantic Guide Dist.*	*Semantic Mask*	*Motion Mask*	*Semantic Guide Det.*	*CGT*	*Cyl Rect.*	***RMSE*** ↓	$\delta < 1.25$↑	***mIoU Seg***	***mIoU Mot.***	***mAP Det***
OmniDet (SAN10-patch)	✓	✗	✗	✗	✗	✗	✗	✗	2.153	0.875	73.2	71.8	63.3
	✓	✓	✗	✗	✗	✗	✗	✗	1.764	0.897	73.6	72.3	63.5
	✓	✓	✗	✗	✗	✗	✓	✗	1.681	0.902	74.2	73.5	63.8
	✓	✓	✓	✓	✗	✗	✗	✗	1.512	0.905	74.5	73.4	63.6
	✓	✓	✓	✓	✗	✗	✓	✗	1.442	0.908	74.8	74.0	64.1
	✓	✓	✓	✗	✓	✗	✗	✗	1.397	0.915	75.2	74.3	64.0
	✓	✓	✓	✗	✓	✗	✓	✗	1.352	0.916	75.5	74.8	64.3
	✓	✓	✓	✗	✓	✓	✗	✗	1.348	0.915	75.9	74.9	67.8
	✓	✓	✓	✗	✓	✓	✓	✗	**1.332**	**0.918**	**76.6**	**75.3**	**68.4**
	✓	✓	✓	✗	✓	✓	✓	✓	1.210	0.929	78.9	79.2	74.1
OmniDet (SAN10-pair)	✓	✓	✓	✓	✗	✗	✓	✗	1.492	0.904	74.1	73.1	63.3
	✓	✓	✓	✗	✓	✓	✓	✗	1.321	0.911	75.4	74.6	67.6
	✓	✓	✓	✗	✓	✓	✓	✓	1.272	0.919	77.1	77.4	72.6
OmniDet (SAN19-patch)	✓	✗	✗	✗	✗	✗	✗	✗	2.138	0.880	73.9	72.4	64.7
	✓	✓	✗	✗	✗	✗	✗	✗	1.749	0.903	74.3	73.0	64.8
	✓	✓	✗	✗	✗	✗	✓	✗	1.662	0.906	74.6	74.1	65.2
	✓	✓	✓	✓	✗	✗	✗	✗	1.495	0.910	74.9	73.8	64.9
	✓	✓	✓	✓	✗	✗	✓	✗	1.427	0.916	75.4	74.7	65.5
	✓	✓	✓	✗	✓	✗	✗	✗	1.378	0.918	75.7	75.1	65.3
	✓	✓	✓	✗	✓	✗	✓	✗	1.331	0.922	76.2	75.6	65.9
	✓	✓	✓	✗	✓	✓	✗	✗	1.320	0.927	76.8	76.2	69.6
	✓	✓	✓	✗	✓	✓	✓	✗	**1.304**	**0.931**	**77.4**	**77.0**	**71.5**
	✓	✓	✓	✗	✓	✓	✓	✓	1.177	0.938	80.2	80.5	76.3

TABLE II: **Comparative study of SAN10-Patch MTL model and the equivalent single task models on three datasets.**

Dist. & Pose Est.	*Sem. Seg.*	*Mot. Seg.*	*Obj. Det.*	*Soil. Seg.*	***RMSE***	***mIoU Seg.***	***mIoU Mot.***	***mAP Det.***	***mIoU Soil.***	*Infer. (fps)*
WoodScape										
✓	✗	✗	✗	✗	1.681	✗	✗	✗	✗	210
✗	✓	✗	✗	✗	✗	72.5	✗	✗	✗	190
✗	✗	✓	✗	✗	✗	✗	68.1	✗	✗	105
✗	✗	✗	✓	✗	✗	✗	✗	63.5	✗	190
✗	✗	✗	✗	✓	✗	✗	✗	✗	80.8	190
✓	✓	✗	✗	✗	1.442	74.8	✗	✗	✗	143
✗	✓	✗	✓	✗	✗	77.1	✗	67.9	✗	143
✓	✓	✓	✗	✗	1.352	75.5	74.8	✗	✗	69
✓	✓	✓	✓	✗	1.332	76.6	75.3	68.4	✗	60
KITTI										
✓	✗	✗	✗	NA	4.126	✗	✗	✗	NA	160
✗	✓	✗	✗	NA	✗	67.7	✗	✗	NA	148
✗	✗	✓	✗	NA	✗	✗	68.3	✗	NA	78
✗	✗	✗	✓	NA	✗	✗	✗	80.1	NA	182
✓	✓	✗	✗	NA	3.984	72.1	✗	✗	NA	103
✓	✓	✓	✗	NA	3.892	71.9	71.7	✗	NA	47
✓	✓	✓	✓	NA	3.859	72.4	72.2	82.3	NA	43
CityScapes										
✓	✗	✗	✗	NA	4.906	✗	✗	✗	NA	156
✗	✓	✗	✗	NA	✗	78.7	✗	✗	NA	132
✗	✗	✓	✗	NA	✗	✗	70.4	✗	NA	64
✗	✗	✗	✓	NA	✗	✗	✗	51.7	NA	167
✓	✓	✗	✗	NA	4.741	79.4	✗	✗	NA	91
✓	✓	✓	✗	NA	4.725	79.1	72.0	✗	NA	36
✓	✓	✓	✓	NA	4.691	81.2	72.7	53.0	NA	31

3) KITTI: The KITTI dataset consists of 42,382 stereo sequences with corresponding raw LiDAR scans, 7,481 images with bounding box annotations, and 200 training images with semantic annotations. We use the data split according to Eigen *et al.* [37] for self-supervised depth estimation with an input size of 640×192 px for all the tasks. For further details on the split, refer to [9]. For the motion segmentation, we use the annotations provided by DeepMotion [38] for Cityscapes and KITTI MoSeg [8]. Here the labels are available only for the cars category.

B. Single Task vs Multi-Task Learning

In Table II, we perform an extensive ablation of our proposed framework on all the datasets, as mentioned above. Quantitative results from our experiments indicate that a multi-task network with 6 tasks, 5 diverse tasks perform better than the single task models along with our proposed synergies explained in Section III-B. For KITTI and CityScapes, we employ our novel VarNorm task weighting technique. With this synergy of perception tasks, we obtain state-of-the-art depth and pose estimation results on the KITTI dataset, as shown in Table V and Table VI respectively. We infer our models using the TensorRT (FP16bit) on NVIDIA's Jetson AGX platform and report the FPS for all the tasks.

C. Ablation Study of our Contributions

For our ablation analysis of the main features shown in Table I, we consider two variants of the self-attention encoder, namely pairwise and patchwise. First, we replace the L_1 loss with a generic parameterized loss function and test it using the self-attention encoder's patchwise variant. We cap the distance estimates to 40m. We achieve significant gains in this setting by attributing better-supervised signal provided by using discriminative features loss $\mathcal{L}_{dis}$. In this case, incorrect distance values are appropriately penalized with more considerable losses along with the combination of $\mathcal{L}_{cvt}$ wherein a correct optimization direction is provided. These losses help the gradient descent approaches to transit smoothly towards optimal solutions. When adding the camera geometry tensor to this setting, we observe a significant increase in accuracy since we train multiple cameras with different camera intrinsics and viewing angles. For the OmniDet framework to be operational in the first place, this an important feature. The aforementioned training strategy makes the network camera-independent and better generalizes to images taken from a different camera.

To achieve synergy between geometry and semantic features, we add semantic guidance to the distance decoder. It helps to reason about geometry and content within the same shared features and disambiguate photometric ambiguities. To establish a robust reconstruction loss free from the dynamic

TABLE III: **Comparison of task-weighting methods on the WoodScape dataset. PA denotes pixel accuracy.**

Task Weighting	*Distance Estimation*		*Semantic Segmentation*		*Motion Segmentation*		*Object Detection*
	Sq. Rel ↓	**Abs Rel** ↓	**mIoU** ↑	**PA** ↑	**mIoU** ↑	**PA** ↑	**mAP** ↑
Single Task	0.060	0.304	72.5	94.8	68.1	94.1	63.5
Equal	0.058	0.302	70.3	92.7	67.3	93.3	64.6
DTP [25]	0.047	0.281	75.8	95.6	75.3	95.3	67.9
DWA [26]	0.054	0.293	75.4	95.2	74.7	95.1	67.5
Geometric [27]	0.061	0.297	74.2	94.1	73.2	94.3	66.7
GradNorm [24]	0.050	0.283	75.9	95.7	74.9	96.0	67.7
Uncertainity [23]	0.044	0.279	76.1	96.2	75.1	95.8	68.0
VarNorm	**0.046**	**0.276**	**76.6**	**96.4**	**75.3**	**96.1**	**68.4**

TABLE IV: **Evaluation of various object detection representations.**

Representation	*mIoU GT*	*mIOU*	*No. of params*
Standard Box	51.3	31.6	4
Curved Box	52.5	32.3	6
Oriented Box	53.9	33.6	5
Ellipse	55.5	35.4	5
24-sided Polygon	**86.6**	**44.6**	**48**

objects' contamination, we introduce semantic and motion masks as described in Section III-B1, to filter all the dynamic objects. Motion mask based filtering yields superior gains along with CGT compared to using semantic masks as semantics might not contain all the dynamic objects in its set of classes as indicated in Eq. 4. Additionally, this contribution possesses the potential to solve the infinite distance issue. Finally, to complete our synergy, we use semantically guided features to the detection decoder as described in Section III-B2, which yields significant gains in mAP, and overall results for all the tasks are inherently improved with better-shared features. All our contributed features and the synergy between tasks help the OmniDet framework to achieve a good scene understanding with high accuracy in each task's predictions. We also experiment using cylindrical rectification (Cyl Rect.), which provides a good trade-off between loss of field-of-view and reducing distortion [29].

For object detection on native fisheye images, in addition to the standard 2D box representation, we benchmark oriented boxes, ellipse, curved boxes, and 24-sided polar polygon representations in Table IV. Here mIoU GT represents the maximum performance we can achieve in terms of instance segmentation by using each representation. It is computed between the ground truth instance segmentation and the ground truth of the corresponding representation. Whereas mIoU represents the performance achieved with our network. We also list the number of parameters involved for each representation for the complexity comparison.

D. State-of-the-Art Comparison on KITTI

To facilitate comparison to previous methods, we also train our distance estimation method in the classical depth estimation setting on the KITTI Eigen split [39] whose results are shown in Table V. With the synergy between depth, semantic, motion, and detection tasks along with the features ablated in Table I and their importance explained in the Section IV-C, *we outperform all the previous monocular methods.* Following

TABLE V: **Evaluation of depth estimation on the KITTI Eigen split**.

	Method	Abs_{rel}	Sq_{rel}	$RMSE$	$RMSE_{log}$	δ_1	δ_2	δ_3
		lower is better				higher is better		
Original [40]	Monodepth2 [10]	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	PackNet-SfM [44]	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	FisheyeDistanceNet [9]	0.117	0.867	4.739	0.190	0.869	0.960	0.982
	UnRectDepthNet [1]	0.107	0.721	4.564	0.178	0.894	0.971	**0.986**
	SynDistNet [14]	0.109	0.718	4.516	0.180	0.896	0.973	**0.986**
	Shu *et al.* [11]	0.104	0.729	4.481	0.179	0.893	0.965	0.984
	OmniDet	**0.092**	**0.657**	**3.984**	**0.168**	**0.914**	**0.975**	**0.986**
	Struct2Depth* [42]	0.109	0.825	4.750	0.187	0.874	0.958	0.983
	GLNet* [45]	0.099	0.796	4.743	0.186	0.884	0.955	0.979
	Shu* *et al.* [11]	0.088	0.712	4.137	0.169	0.915	0.965	0.982
	OmniDet*	**0.077**	**0.641**	**3.859**	**0.152**	**0.931**	**0.979**	**0.989**
Improved [41]	Monodepth2 [10]	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM [44]	0.078	0.420	3.485	0.121	0.931	0.986	0.996
	UnRectDepthNet [1]	0.081	0.414	3.412	0.117	0.926	0.987	0.996
	SynDistNet [14]	0.076	0.412	3.406	0.115	0.931	0.988	0.996
	OmniDet	**0.067**	**0.306**	**3.098**	**0.101**	**0.944**	**0.991**	**0.997**
	OmniDet*	**0.048**	**0.287**	**2.913**	**0.081**	**0.948**	**0.991**	**0.998**

TABLE VI: **Evaluation of the pose estimation** on the KITTI Odometry Benchmark [39].

Method	*No. of Frames*	*GT*	*Sequence 09*	*Sequence 10*
GeoNet [46]	5	✓	0.012 ± 0.007	0.012 ± 0.009
Struct2Depth [42]	5	✓	0.011 ± 0.006	0.011 ± 0.010
Ranjan [47]	5	✓	0.011 ± 0.006	0.011 ± 0.010
PackNet-SfM [44]	5	✓	0.010 ± 0.005	0.009 ± 0.008
PackNet-SfM [44]	5	✗	0.014 ± 0.007	0.012 ± 0.008
OmniDet	5	✓	**0.009 ± 0.004**	0.008 ± **0.005**
OmniDet	5	✗	**0.010 ± 0.005**	**0.010 ± 0.008**

best practices, we cap depths at 80 m. We also evaluate using the *Original* [40] as well as *Improved* [41] ground truth depth maps. Method* indicates the online refinement technique [42], where the model is trained during the inference. Using the online refinement method from [42], we obtain a significant improvement.

In Table VI, we report the average trajectory error in meters of the pose estimation network by following the same protocols by Zhou [43] on the official KITTI odometry split (containing 11 sequences with ground-truth (GT) odometry acquired with the IMU/GPS measurements, which is used for evaluation purpose only), and use sequences 00-08 for training and 09-10 for testing. We outperform the previous methods listed in Table VI, mainly by applying our bundle adjustment framework using our cross-sequence distance consistency loss [9] that induces more constraints and simultaneously optimizes distances and camera pose for an implicitly extended training input sequence. This provides additional consistency constraints that are not induced by previous methods.

V. Conclusion

We successfully demonstrated a six-task network with a shared encoder and synergized decoders on fisheye surround-view images in this work. The majority of the automated driving community's research continues to focus on individual tasks, and there is much progress to be made in designing and training optimal multi-task models. We have several novel contributions, including camera geometry tensor usage for encoding radial distortion and variance-based normalization task weighting, and generalized object detection representations. To enable comparison, we evaluate our network on five tasks on KITTI and Cityscapes, achieving competitive results. There are still many practical challenges in scaling

to a higher number of tasks: building a diverse and balanced dataset, corner case mining, stable training mechanisms, and designing an optimal map representation that combines all tasks reducing the post-processing. We hope that this work encourages further research in building a unified perception model for autonomous driving.

REFERENCES

[1] V. Ravi Kumar, S. Yogamani, M. Bach, C. Witt, S. Milz, and P. Mader, "UnRectDepthNet: Self-Supervised Monocular Depth Estimation using a Generic Framework for Handling Common Camera Distortion Models," in *Proc. of IROS*, 2020.

[2] 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.

[3] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE Transactions on PAMI*, 2021.

[4] C. Mao, A. Gupta, V. Nitin, B. Ray, S. Song, J. Yang, and C. Vondrick, "Multitask Learning Strengthens Adversarial Robustness," *arXiv preprint arXiv:2007.07236*, 2020.

[5] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and Raquel, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1013–1020.

[6] P. Maddu, W. Doherty, G. Sistu, I. Leang, M. Uricar, S. Chennupati, H. Rashed, J. Horgan, C. Hughes, and S. Yogamani, "FisheyeMultiNet: Real-time Multi-task Learning Architecture for Surround-view Automated Parking System," in *Irish Machine Vision and Image Processing Conference*, 2019.

[7] M. Uřičář, P. Křížek, G. Sistu, and S. Yogamani, "SoilingNet: Soiling Detection on Automotive Surround-View Cameras," in *2019 IEEE ITSC*. IEEE, 2019, pp. 67–72.

[8] M. Siam, H. Mahgoub, and S. Zahran, Yogamani, "ModNet: Motion and appearance based moving object detection network for autonomous driving," in *2018 (ITSC)*. IEEE, 2018, pp. 2859–2864.

[9] V. Ravi Kumar, S. A. Hiremath, M. Bach, S. Milz, C. Witt, C. Pinard, S. Yogamani, and P. Mäder, "FisheyeDistanceNet: Self-supervised scale-aware distance estimation using monocular fisheye camera for autonomous driving," in *Proc. of ICRA*. IEEE, 2020, pp. 574–581.

[10] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," in *Proc. of ICCV*, Seoul, Korea, Oct. 2019, pp. 3828–3838.

[11] C. Shu, K. Yu, Z. Duan, and K. Yang, "Feature-metric Loss for Self-supervised Learning of Depth and Egomotion," in *Proc. of ECCV*, Glasgow, UK, Aug. 2020.

[12] J. T. Barron, "A General and Adaptive Robust Loss Function," in *Proc. of CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 4331–4339.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[14] V. Ravi Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mäder, "SynDistNet: Self-Supervised Monocular Fisheye Camera Distance Estimation Synergized with Semantic Segmentation for Autonomous Driving," in *Proc. of WACV*, Waikoloa, HI, USA, Jan. 2021.

[15] H. Rashed, E. Mohamed, G. Sistu, V. Ravi Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "Generalized Object Detection on Fisheye Cameras for Autonomous Driving: Dataset, Representations and Baseline," in *Proc. of WACV*, 2021, pp. 2272–2280.

[16] 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," *arXiv preprint arXiv:2005.13243*, 2020.

[17] C.-H. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *Proc. of PAMI*, vol. 11, no. 8, pp. 859–872, 1989.

[18] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[20] M. Berman, A. Rannen Triki, and Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proc. of CVPR*, 2018, pp. 4413–4421.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. of ICCP*, 2017, pp. 2980–2988.

[22] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proc. of CVPR*, 2017, pp. 6129–6138.

[23] A. Kendall, Y. Gal, and R. Cipolla, "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *Proc. of CVPR*, Salt Lake City, UT, USA, Jun. 2018, pp. 7482–7491.

[24] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *ICML*. PMLR, 2018, pp. 794–803.

[25] M. Guo, A. Haque, Huang, and S. Yeung, "Dynamic task prioritization for multitask learning," in *Proc. of ECCV*, 2018, pp. 270–287.

[26] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proc. of CVPR*, 2019, pp. 1871–1880.

[27] S. Chennupati, G. Sistu, S. Yogamani, and S. A Rawashdeh, "MultiNet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning," in *Proc. of CVPR Workshops*, 2019, pp. 1200–1210.

[28] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. of CVPR*, 2020, pp. 10 076–10 085.

[29] 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," in *Proc. of ICCV*, Seoul, Korea, Oct. 2019, pp. 9308–9318.

[30] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera, "CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth," in *Proc. of CVPR*, USA, 2019, pp. 11 826–11 835.

[31] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An Intriguing Failing of Convolutional Neural Networks and the Coordconv solution," in *Proc. of NIPS*, Montréal, QC, Canada, Dec. 2018, pp. 9605–9616.

[32] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz, "Pixel-Adaptive Convolutional Neural Networks," in *Proc. of CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 11 166–11 175.

[33] 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*, no. 1908.03265, 2019.

[34] M. Zhang, J. Lucas, and G. E. Hinton, "Lookahead Optimizer: k steps forward, 1 step back," in *Proc. of NIPS*, Canada, 2019, pp. 593–604.

[35] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. of CVPR*. IEEE, 2012, pp. 3354–3361.

[36] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, and S. Roth, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of CVPR*, 2016, pp. 3213–3223.

[37] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture," in *Proc. of ICCV*, Santiago, Chile, Dec. 2015, pp. 2650–2658.

[38] J. Vertens, A. Valada, and W. Burgard, "Smsnet: Semantic motion segmentation using deep convolutional neural networks," in *Proc. of IROS*. IEEE, 2017, pp. 582–589.

[39] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[40] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network," in *Proc. of NIPS*, Montréal, QC, Canada, Dec. 2014, pp. 2366–2374.

[41] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity Invariant CNNs," in *Proc. of 3DV*, Italy, Oct. 2017, pp. 11–20.

[42] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *Proc. of AAAI*, USA, Jan. 2019, pp. 8001–8008.

[43] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. of CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 1851–1860.

[44] V. Guizilini, R. Ambrus, S. Pillai, and A. Gaidon, "3D Packing for Self-Supervised Monocular Depth Estimation," in *Proc. of CVPR*, Seattle, WA, USA, Jun. 2020, pp. 2485–2494.

[45] Y. Chen, C. Schmid, and C. Sminchises, "Self-Supervised Learning With Geometric Constraints in Monocular Video Connecting Flow, Depth, and Camera," in *Proc. of ICCV*, Korea, Oct. 2019, pp. 7063–7072.

[46] Z. Yin and J. Shi, "GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose," in *Proc. of CVPR*, Salt Lake City, UT, USA, Jun. 2018, pp. 1983–1992.

[47] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation," in *Proc. of CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 12 240–12 249.