2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

FisheyeDistanceNet: Self-Supervised Scale-Aware Distance Estimation using Monocular Fisheye Camera for Autonomous Driving

Varun Ravi Kumar[1], Sandesh Athni Hiremath[1], Markus Bach[1], Stefan Milz[1],
Christian Witt[1], Clément Pinard[2], Senthil Yogamani[3] and Patrick Mäder[4]
[1]Valeo DAR Kronach, Germany [2]ENSTA ParisTech Palaiseau, France
[3]Valeo Vision Systems, Ireland [4]Technische Universität Ilmenau, Germany

***Abstract*— Fisheye cameras are commonly used in applications like autonomous driving and surveillance to provide a large field of view ($> 180^{\circ}$). However, they come at the cost of strong non-linear distortions which require more complex algorithms. In this paper, we explore Euclidean distance estimation on fisheye cameras for automotive scenes. Obtaining accurate and dense depth supervision is difficult in practice, but self-supervised learning approaches show promising results and could potentially overcome the problem. We present a novel self-supervised scale-aware framework for learning Euclidean distance and ego-motion from raw monocular fisheye videos without applying rectification. While it is possible to perform piece-wise linear approximation of fisheye projection surface and apply standard rectilinear models, it has its own set of issues like re-sampling distortion and discontinuities in transition regions. To encourage further research in this area, we will release our dataset as part of the WoodScape project [1]. We further evaluated the proposed algorithm on the KITTI dataset and obtained state-of-the-art results comparable to other self-supervised monocular methods. Qualitative results on an unseen fisheye video demonstrate impressive performance[1].**

I. Introduction

There has been a significant rise in the usage of fisheye cameras in various automotive applications [2], [3], [4], surveillance [5] and robotics [6] due to their large Field of View (FOV). Recently, several computer vision tasks on fisheye cameras have been explored including object detection [7], soiling detection [8], motion estimation [9], image restoration [10] and SLAM [11]. Depth estimation is an important task in autonomous driving as it is used to avoid obstacles and plan trajectories. While depth estimation has been substantially studied for narrow FOV cameras, it has barely been explored for fisheye cameras [12], [13].

Previous learning-based approaches [14], [15], [16], [17] have solely focused on traditional 2D content captured with cameras following a typical pinhole projection model based on rectified image sequences. With the surge of efficient and cheap wide angle fisheye cameras and their larger FOV in contrast to pinhole cameras, there has been significant interest in the computer vision community to perform depth estimation from omnidirectional content similar to traditional 2D content via omnidirectional stereo [18], [19], [20], [21] and structure-from-motion (SfM) [22] approaches.

[1]see Fig. 1 and `https://youtu.be/Sgq1WzoOmXg`

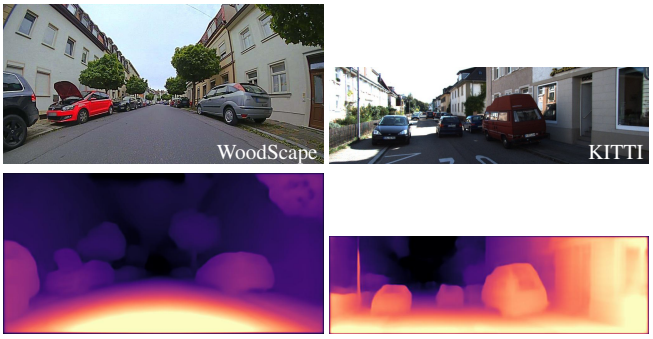

Fig. 1 **Distance and depth derived from a single fisheye image (left) and single pinhole image (right) respectively.** Our self-supervised model, **FisheyeDistanceNet**, produces sharp, high quality distance and depth maps.

Depth estimation models may be learned in a supervised fashion on LiDAR distance measurements, such as KITTI [23]. In previous work, we followed this approach and demonstrated the possibility to estimate high-quality distance maps using LiDAR ground truth on fisheye images [12]. However, setting up the entire rig for such recordings is expensive and time consuming, and therefore limits the amount of data on which a model can be trained. To overcome this problem, we propose *FisheyeDistanceNet*, the first end-to-end self-supervised monocular scale-aware training framework. FisheyeDistanceNet uses convolutional neural networks (CNN) on raw fisheye image sequences to regress a Euclidean distance map and provides a baseline for single frame Euclidean distance estimation. We summarize our contributions as follows:

- A self-supervised training strategy that aims at inferring a distance map from a sequence of distorted and unrectified raw fisheye images.
- A solution to the scale factor uncertainty with the bolster from ego-motion velocity allows outputting metric distance maps. This facilitates the map's practical use for self-driving cars.
- A novel combination of super resolution networks and deformable convolution layers [24] to output high resolution distance maps with sharp boundaries from a low resolution input. Inspired by the super resolution of images approach [25] this approach allows us to accurately resolve distances replacing the deconvolution [26] and naive nearest neighbor or bilinear upsampling.

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

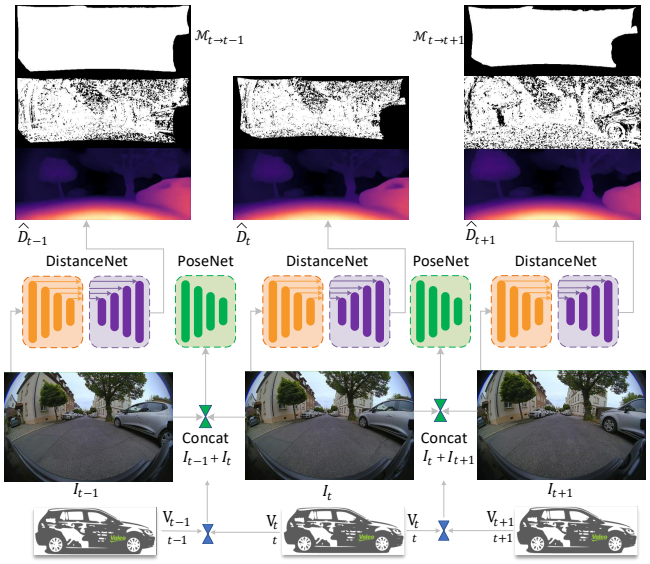

Fig. 2 **Overview of our method.** The first row represents our ego masks as described in Section II-D, $\mathcal{M}_{t\to t-1}$, $\mathcal{M}_{t\to t+1}$ indicate which pixel coordinates are valid when constructing $\hat{I}_{t-1\to t}$ from I_{t-1} and $\hat{I}_{t+1\to t}$ from I_{t+1} respectively. The second row indicates the masking of static pixels computed after 2 epochs, where black pixels are filtered from the photometric loss (i.e. $\omega = 0$). It prevents dynamic objects at similar speed as the ego car and low texture regions from contaminating the loss. The masks are computed for forward and backward sequences from the input sequence S and reconstructed images using Eq. 10 as described in Section II-D. The third row represents the distance estimates corresponding to their input frames. Finally, the vehicle's odometry data is used to resolve the scale factor issue.

- We depict the importance of using backward sequences for training and construct a loss for these sequences. Moreover, a combination of filtering static pixels and an ego mask is employed. The incorporated bundle-adjustment framework [27] jointly optimizes distances and camera poses within a sequence by increasing the baseline and providing additional consistency constraints.

II. Self-Supervised Scale-Aware FisheyeDistanceNet

Zhou et al.'s [15] self-supervised monocular structure-from-motion (SfM) framework aims at learning:

1) a monocular depth model $g_d : I_t \to D$ predicting a scale-ambiguous depth $\hat{D} = g_d(I_t(p))$ per pixel p in the target image I_t; and
2) an ego-motion predictor $g_x : (I_t, I_{t'}) \to I_{t\to t'}$ predicting a set of six degrees of freedom rigid transformations $T_{t\to t'} \in \mathrm{SE}(3)$, between the target image I_t and the set of reference images $I_{t'}$. Typically, $t' \in \{t+1, t-1\}$, i.e. the frames I_{t-1} and I_{t+1} are used as reference images, although using a larger window is possible.

A limitation of this approach is that both depth and pose are estimated up to an unknown scale factor in the monocular SfM pipeline.

The depth which acts as an intermediary variable is obtained from the network by constraining the model to perform image synthesis. Depth estimation is an ill-posed problem as there could exist a large number of possible incorrect depths per pixel, which can also recreate the novel view, given the relative pose between I_t and $I_{t'}$.

Using view-synthesis as the supervising technique we can train the network using the viewpoint of I_{t-1} and I_{t+1} to estimate the appearance of a target image I_t on raw fisheye images. A naive approach would be correcting raw fisheye images to piecewise or cylindrical projections and would essentially render the problem equivalent to Zhou et al.'s work [15]. In contrast, at the core of our approach there is a simple yet efficient technique for obtaining scale-aware distance maps.

This section starts with discussing the geometry of the problem and how it is used to obtain differentiable losses. We describe the scale-aware FisheyeDistanceNet and its effects on the output distance estimates. Additionally, we provide an in-depth discussion of the various losses.

A. Modeling of Fisheye Geometry

1) Projection from camera coordinates to image coordinates: The projection function $X_c \mapsto \Pi(X_c) = p$ of a 3D point $X_c = (x_c, y_c, z_c)^T$ in camera coordinates to a pixel $p = (u, v)^T$ in the image coordinates is obtained via a 4^{th} order polynomial in the following way:

$$\varphi = \arctan2(y_c, x_c) \tag{1}$$

$$\theta = \frac{\pi}{2} - \arctan2(z_c, r_c) \tag{2}$$

$$\varrho(\theta) = k_1 \cdot \theta + k_2 \cdot \theta^2 + k_3 \cdot \theta^3 + k_4 \cdot \theta^4 \tag{3}$$

$$p = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \varrho(\theta) \cdot \cos\varphi \cdot a_x + c_x \\ \varrho(\theta) \cdot \sin\varphi \cdot a_y + c_y \end{pmatrix} \tag{4}$$

where $r_c = \sqrt{x_c^2 + y_c^2}$, θ is the angle of incidence, $\varrho(\theta)$ is the mapping of incident angle to image radius, (a_x, a_y) is the aspect ratio and (c_x, c_y) is the principal point.

2) Unprojection from image coordinates to camera coordinates: The unprojection function $(p, \hat{D}) \mapsto \Pi^{-1}(p, \hat{D}) = X_c$ of an image pixel $p = (u, v)^T$ and it's distance estimate $\hat{D}$ to the 3D point $X_c = (x_c, y_c, z_c)^T$ is obtained via the following steps. Letting $(x_i, y_i)^T = \big((u - c_x)/a_x, (v - c_y)/a_y\big)^T$, we obtain the angle of incidence θ by numerically calculating the 4^{th} order polynomial roots of $\varrho = \sqrt{x_i^2 + y_i^2}$ using the distortion coefficients k_1, k_2, k_3, k_4 (see Eq. 3). For training efficiency, we pre-calculate the roots and store them in a lookup table for all the pixel coordinates. Now, θ is used to get

$$r_c = \hat{D} \cdot \sin\theta \quad \text{and} \quad z_c = \hat{D} \cdot \cos\theta \tag{5}$$

where the distance estimate $\hat{D}$ from the network represents the Euclidean distance $\|X_c\| = \sqrt{x_c^2 + y_c^2 + z_c^2}$ of a 3D point X_c. The polar angle φ and the x_c, y_c components can be obtained as follows:

$$\varphi = \arctan2(y_i, x_i), \quad x_c = r_c \cdot \cos\varphi, \quad y_c = r_c \cdot \sin\varphi.$$

B. Photometric Loss

Let us consider the image reconstruction error from a pair of images $I_{t'}$ and I_t, distance estimate $\hat{D}_t$ at time t, and the relative pose for I_t, with respect to the source image $I_{t'}$'s pose, as $T_{t\to t'}$. Using the distance estimate $\hat{D}_t$ of the network a point cloud P_t is obtained via:

$$P_t = \Pi^{-1}(p_t, \hat{D}_t) \tag{6}$$

where Π^{-1} represents the unprojection from image to camera coordinates as explained in Section II-A.2, p_t the pixel set of image I_t. The pose estimate $T_{t\to t'}$ from the pose network is used to get an estimate $\hat{P}_{t'} = T_{t\to t'} P_t$ for the point cloud of the image $I_{t'}$. $\hat{P}_{t'}$ is then projected onto the fisheye camera at time t' using the projection model Π described in Section II-A.1. Combining transformation and projection with Eq. 6 establishes a mapping from image coordinates $p_t = (u, v)^T$ at time t to image coordinates $\hat{p}_{t'} = (\hat{u}, \hat{v})^T$ at time t'. This mapping allows for the reconstruction $\hat{I}_{t'\to t}$ of the target frame I_t by backward warping the source frame $I_{t'}$.

$$\hat{p}_{t'} = \Pi\big(T_{t\to t'}\Pi^{-1}(p_t, \hat{D}_t)\big), \quad \hat{I}^{uv}_{t'\to t} = \Big\langle I^{\hat{u}\hat{v}}_{t'} \Big\rangle \tag{7}$$

Since the warped coordinates $\hat{u}, \hat{v}$ are continuous, we apply the differentiable spatial transformer network introduced by [28] to compute $\hat{I}_{t'\to t}$ by performing bilinear interpolation of the four pixels from $I_{t'}$ which lie close to $\hat{p}_{t'}$. The symbol $\langle \dots \rangle$ denotes the corresponding sampling operator.

Following [16], [29] the image reconstruction error between the target image I_t and the reconstructed target image $\hat{I}_{t'\to t}$ is calculated using the L1 pixel-wise loss term combined with Structural Similarity (SSIM) [30], as our photometric loss $\mathcal{L}_p$ given by Eq. 8 below.

$$\begin{aligned}\tilde{\mathcal{L}}_p(I_t, \hat{I}_{t'\to t}) &= \alpha\ \frac{1 - \text{SSIM}(I_t, \hat{I}_{t'\to t}, \mathcal{M}_{t\to t'})}{2} \\ &\quad + (1-\alpha)\ \Big\| (I_t - \hat{I}_{t'\to t}) \odot \mathcal{M}_{t\to t'} \Big\|_{l^1} \\ \mathcal{L}_p &= \min_{t'\in\{t+1,t-1\}} \tilde{\mathcal{L}}_p(I_t, \hat{I}_{t'\to t})\end{aligned} \tag{8}$$

where $\alpha = 0.85$, $\mathcal{M}_{t\to t'}$ is the binary mask as discussed in Section II-D and the symbol $\odot$ denotes element-wise multiplication. Following [14] instead of averaging the photometric error over all source images, we adopt per-pixel minimum. This significantly sharpens the occlusion boundaries and reduces the artifacts resulting in higher accuracy.

The self-supervised framework assumes a static scene, no occlusion and change of appearance (e.g. brightness constancy). A large photometric cost is incurred, potentially worsening the performance, if there exist dynamic objects and occluded regions. These areas are treated as outliers similar to [27] and clip the photometric loss values to a 95^{th} percentile. Zero gradient is obtained for errors larger than 95%. This improves the optimization process and provides a way to strengthen the photometric error.

C. Solving Scale Factor Ambiguity at Training Time

For a pinhole projection model, $depth \propto 1/disparity$. Henceforth, the network's sigmoided output σ can be converted to depth with $D = 1/(a\sigma + b)$, where a and b are chosen to constrain D between 0.1 and 100 units [14]. For a spherical image, we can only obtain angular disparities [31] by rectification. To perform distance estimation on raw fisheye images, we would require metric distance values to warp the source image $I_{t'}$ onto the target frame I_t. Due to the limitations of the monocular SfM objective, both the monocular depth g_d and ego-motion predictor $g_{\mathbf{x}}$ predict *scale-ambiguous* values which would make it impossible to estimate distance maps on fisheye images. To achieve scale-aware distance values, we normalize the pose network's estimate $T_{t\to t'}$ and scale it with Δx, the displacement magnitude relative to target frame I_t which is calculated using vehicle's instantaneous velocity estimates $v_{t'}$ at time t' and v_t at time t. We also apply this technique on KITTI [23] to obtain metric depth maps.

$$\overline{T}_{t\to t'} = \frac{T_{t\to t'}}{\|T_{t\to t'}\|} \cdot \Delta x \tag{9}$$

D. Masking Static Pixels and Ego Mask

Following [14], we incorporate a masking approach to filter out static pixels which do not change their appearance from one frame to the other in the training sequence. The approach would filter out objects which move at the same speed as the ego-car, and also ignore the static frame when the ego-car stops moving. Similar to other approaches [14], [15], [32], [33] the per-pixel mask ω is applied to the loss by weighting the pixels selectively. Instead of being learned from the object motion [34], the mask is computed in the forward pass of the network, yielding a binary mask output where $\omega \in \{0, 1\}$. Wherever the photometric error of the warped image $\hat{I}_{t'\to t}$ is not lower than that of the original unwarped source frame $I_{t'}$ in each case compared to the target frame I_t, ω is set to ignore the loss of such pixels, i.e.

$$\omega = \Big[\ \min_{t'} pe(I_t, \hat{I}_{t'\to t}) < \min_{t'} pe(I_t, I_{t'})\ \Big] \tag{10}$$

where $[\,]$ is the Iverson bracket. Additionally, we add a binary ego mask $\mathcal{M}_{t\to t'}$ proposed in [35] that ignores computing the photometric loss on the pixels that do not have a valid mapping i.e. some pixel coordinates of the target image I_t may not be projected onto the source image $I_{t'}$ given the estimated distance $\hat{D}_t$.

E. Backward Sequence

In the forward sequence, we synthesize the target frame I_t with the source frames I_{t-1} and I_{t+1} (i.e. as per above discussion $t' \in \{t+1, t-1\}$). Analogously, backward sequence is carried out by using I_{t-1} and I_{t+1} as target frames and I_t as source frame. We include warps $\hat{I}_{t\to t-1}$ and $\hat{I}_{t\to t+1}$, thereby inducing more constraints to avoid overfitting and resolve unknown distances in the border areas at the test time, as also observed in previous works [14], [15], [36]. We construct the loss for the additional backward sequence in a similar manner to the forward. This comes at the cost of high computational effort and longer training time as we perform two forward and backward warps which yields superior results on the Fisheye and KITTI dataset compared

to the previous approaches [14], [15] which train only with one forward sequence and one backward sequence.

F. Edge-Aware Smoothness Loss

In order to regularize distance and avoid divergent values in occluded or texture-less low-image gradient areas, we add a geometric smoothing loss. We adopt the edge-aware term similar to [16], [35], [37]. The regularization term is imposed on the inverse distance map. Unlike previous works, the loss is not decayed for each pyramid level by a factor of 2 due to down-sampling, as we use a super resolution network (see Section III-A)

$$\mathcal{L}_s(\hat{D}_t) = |\partial_u \hat{D}_t^*| e^{-|\partial_u I_t|} + |\partial_v \hat{D}_t^*| e^{-|\partial_v I_t|} \tag{11}$$

To discourage shrinking of estimated distance [17], mean-normalized inverse distance of I_t is considered, i.e. $\hat{D}_t^* = \hat{D}_t^{-1}/\overline{D}_t$, where $\overline{D}_t$ denotes the mean of $\hat{D}_t^{-1} := 1/\hat{D}_t$.

G. Cross-Sequence Distance Consistency Loss

The SfM setting uses an N-frame training snippet $S = \{I_1, I_2, \cdots, I_N\}$ from a video as input. The FisheyeDistanceNet can estimate the distance of each image in the training sequence. Another constraint can be enforced among the frames in S, since the distances of a 3D point estimated from different frames should be consistent.

Let us assume $\hat{D}_{t'}$ and $\hat{D}_t$ are the estimates of the images $I_{t'}$ and I_t respectively. For each pixel $p_t \in I_t$, we can use Eq. 7 to obtain $\hat{p}_{t'}$. Since it's coordinates are real valued, we apply the differentiable spatial transformer network introduced by [28] and estimate the distance value of $\hat{p}_{t'}$ by performing bilinear interpolation of the four pixel's values in $\hat{D}_{t'}$ which lie close to $\hat{p}_{t'}$. Let us denote the distance map obtained through this as $\hat{D}_{t\to t'}(p_t)$. Next, we can transform the point cloud in frame I_t to frame $I_{t'}$ by first obtaining P_t using Eq. 6. We transform the point cloud P_t using the pose network's estimate via $\hat{P}_{t'} = T_{t\to t'} P_t$. Now, $D_{t\to t'}(p_t) := \|\hat{P}_{t'}\|$ denotes the distance generated from point cloud $\hat{P}_{t'}$. Ideally, $D_{t\to t'}(p_t)$ and $\hat{D}_{t\to t'}(p_t)$ should be equal. Therefore, we can define the following cross-sequence distance consistency loss (CSDCL) for the training sequence S:

$$\mathcal{L}_{dc} = \sum_{t=1}^{N-1} \sum_{t'=t+1}^{N} \left(\sum_{p_t} \mathcal{M}_{t\to t'} \left| D_{t\to t'}(p_t) - \hat{D}_{t\to t'}(p_t) \right| + \sum_{p_{t'}} \mathcal{M}_{t'\to t} \left| D_{t'\to t}(p_{t'}) - \hat{D}_{t'\to t}(p_{t'}) \right| \right) \tag{12}$$

Eq. 12 contains one term for which pixels and point clouds are warped forwards in time (from t to t') and one term for which they are warped backwards in time (from t' to t).

In prior works [32], [37], the consistency error is limited to only two frames, whereas we apply it to the entire training sequence S. This induces more constraints and enlarges the baseline, inherently improving the distance estimation [27].

H. Final Training Loss

The overall self-supervised structure-from-motion (SfM) *from motion* objective consists of a photometric loss $\mathcal{L}_p$ imposed between the reconstructed target image $\hat{I}_{t'\to t}$ and the target image I_t, included once for the forward and once for the backward sequence, and a distance regularization term $\mathcal{L}_s$ ensuring edge-aware smoothing in the distance estimates. Finally, $\mathcal{L}_{dc}$ a cross-sequence distance consistency loss derived from the chain of frames in the training sequence S is also included. To prevent the training objective getting stuck in the local minima due to the gradient locality of the bilinear sampler [28], we adopt 4 scales to train the network as followed in [15], [16]. The final objective function is averaged over per-pixel, scale and image batch.

$$\mathcal{L} = \sum_{n=1}^{4} \frac{\mathcal{L}_n}{2^{n-1}}, \tag{13}$$
$$\mathcal{L}_n = {}^n\mathcal{L}_p^f + {}^n\mathcal{L}_p^b + \gamma\ {}^n\mathcal{L}_{dc} + \beta\ {}^n\mathcal{L}_s$$

III. Network Details

A. Deformable Super-Resolution Distance and PoseNet

The distance estimation network is mainly based on the U-net architecture [42], an *encoder-decoder* network with skip connections. After testing different variants of ResNet family, such as ResNet50 with 25M parameters, we chose a ResNet18 [43] as the encoder. The key aspect here is replacing normal convolutions with deformable convolutions since regular CNNs are inherently limited in modeling large, unknown geometric distortions due to their fixed structures, such as fixed filter kernels, fixed receptive field sizes, and fixed pooling kernels [44], [24].

In previous works [14], [15], [16], [17], [36], the decoded features were upsampled via a nearest neighbor interpolation or with learnable transposed convolutions. The main drawback of this process is that it may lead to large errors at object boundaries in the upsampled distance map as the interpolation simply combines distance values of background and foreground. For effective and detailed preservation of the decoded features, we leverage the concept of sub-pixel convolutions [25] to our super resolution network. We use pixel shuffle convolutions and replace the convolutional feature upsampling, performed via a nearest neighbor interpolation or with learnable transposed convolutions. The resulting distance maps are super-resolved, have sharp boundaries and expose more details of the scene.

The backbone of our pose estimation network is based on [14] and predicts rotation using Euler angle parameterization. The output is a set of six DOF transformations between I_{t-1} and I_t as well as I_t and I_{t+1}. We have replaced normal convolutions with deformable convolutions for the encoder-decoder setting.

B. Implementation Details

We use Pytorch [45] and employ Adam [46] optimizer to minimize the training objective function (13) with $\beta_1 = 0.9$,

Approach	Abs Rel	Sq Rel	RMSE	RMSE (log)	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
	lower is better				higher is better		
KITTI							
Zhou [15]†	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [38]	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Vid2depth [35]	0.163	1.240	6.220	0.250	0.762	0.916	0.968
GeoNet [36]†	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DDVO [17]	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [37]	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Ranjan [39]	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [33]	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth '(M)' [34]	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Zhou [27]	0.139	1.057	5.213	0.214	0.831	0.940	0.975
PackNet-SfM [40]	0.120	0.892	4.898	0.196	0.864	0.954	0.980
Monodepth2 [14]	**0.115**	0.903	4.863	0.193	**0.877**	0.959	0.981
FisheyeDistanceNet	0.117	**0.867**	**4.739**	**0.190**	0.869	**0.960**	**0.982**
FisheyeDistanceNet (1024 × 320)	0.109	0.788	4.669	0.185	0.889	0.964	0.982
WoodScape							
FisheyeDistanceNet cap 80 m	0.167	1.108	3.814	0.216	0.794	0.953	0.972
FisheyeDistanceNet cap 40 m	0.152	0.768	2.723	0.210	0.812	0.954	0.974
FisheyeDistanceNet cap 30 m	0.149	0.613	2.402	0.204	0.810	0.957	0.976

TABLE I Quantitative results of leaderboard algorithms on KITTI dataset [23] and FisheyeDistanceNet on Fisheye dataset part of WoodScape [1]. Single-view depth estimation results using the Eigen Split [41] for depths reported less than $80\,m$, as indicated in [41] for pinhole model. All the approaches are self-supervised on monocular video sequences. At test-time, all monocular methods excluding our FisheyeDistanceNet, scale the estimated depths using median ground-truth LiDAR depth. For the fisheye dataset, we estimate distance rather than depth. † marks newer results reported on GitHub.

$\beta_2 = 0.999$. We train the model for 25 epochs, with a batch size of 20 on 24GB Titan RTX with initial learning rate of 10^{-4} for the first 20 epochs, then drop to 10^{-5} for the last 5 epochs. The sigmoided output σ from the distance decoder is converted to distance with $D = a \cdot \sigma + b$. For the pinhole model, depth $D = 1/(a \cdot \sigma + b)$, where a and b are chosen to constrain D between 0.1 and 100 units. The original input resolution of the fisheye image is 1280×800 pixels, we crop it to 1024×512 to remove the vehicle's bumper, shadow and other artifacts of the vehicle. Finally the cropped image is downscaled to 512×256 before feeding to the network. For pinhole model on KITTI, we use 640×192 pixels as the network input.

We experimented with batch normalization [47] and group normalization [48] layers in the encoder-decoder setting. We have found that group normalization with $G = 32$ significantly improves the results [49]. The smoothness weight term β and cross-sequence distance consistency weight term γ have been set to 0.001. We applied deformable convolutions to the 3 x 3 conv layers in stages conv3, conv4, and conv5 in ResNet18 and ResNet50, with 12 layers of deformable convolution in the encoder part compared to 3 layers in [44], all in the conv5 stage for ResNet50. We replaced the subsequent layers of the decoder with deformable convolutions for the distance and pose network. For the pinhole model, on KITTI Eigen split in Section IV-A.2 we used normal convolutions instead of deformable convolutions.

Finally, to alleviate checkerboard artifacts from the output distance maps using sub-pixel convolution [25], we initialized the last convolutional layer in a specific way before the pixel shuffle operation as described in [50].

IV. Experiments

A. Datasets

*1) **WoodScape – Fisheye Dataset**:* The dataset contains roughly 40,000 raw images obtained with a fisheye camera and point clouds from a sparse Velodyne HDL-64E rotating 3D laser scanner as ground truth for the test set. The training set contains 39,038 images collected by driving around various parts of Bavaria, Germany. The validation and the test split contain 1,214 and 697 images respectively. The dataset distribution is similar to the KITTI Eigen split used in [14], [15] for the pinhole model. The training set comprises three scene categories: *city*, *residential* and *suburban*. While training, these categories are randomly shuffled and fed to the network. We filter static scenes based on the speed of the vehicle with a threshold of $2\,\mathrm{km/h}$ to remove image frames that only observe minimal camera ego-motion, since distance cannot be learned under these circumstances. Comparable to previous experiments on pinhole SfM [15], [14], we set the length of the training sequence to 3.

*2) **KITTI – Eigen Split**:* We use the KITTI dataset and data split according to Eigen et al. [51] for the experiments with pinhole image data. We filter static frames as proposed by Zhou et al. [15]. The resulting training set contains 39,810 images and the validation split comprises 4,424 images. We

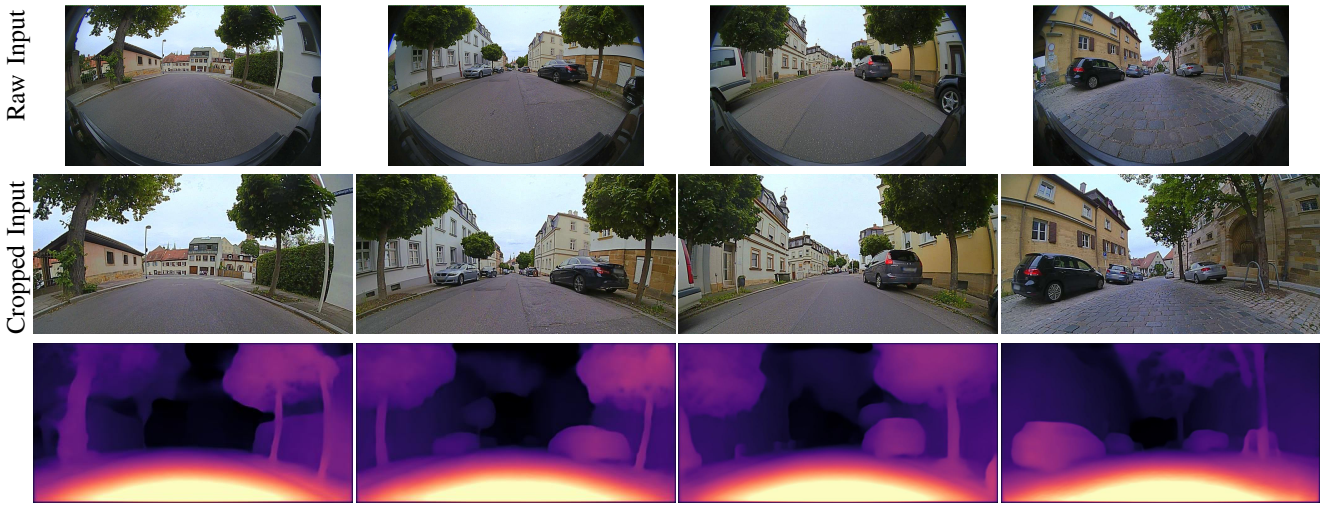

Fig. 3 **Qualitative results on the Fisheye WoodScape dataset.** Our FisheyeDistanceNet produces sharp distance maps on raw fisheye images.

Method	FS	BS	SR	CSDCL	DCN	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours	✓	✓	✓	✓	✓	0.152	0.768	2.723	0.210	0.812	0.954	0.974
Ours	✓		✓	✓	✓	0.172	0.829	2.925	0.243	0.802	0.952	0.970
Ours	✓			✓	✓	0.181	0.913	3.180	0.250	0.823	0.938	0.963
Ours	✓				✓	0.190	0.997	3.266	0.258	0.796	0.930	0.963
Ours	✓					0.201	1.282	3.589	0.276	0.590	0.898	0.949

TABLE II Ablation study on different variants of our FisheyeDistanceNet using the Fisheye WoodScape dataset [1]. Distances are capped at $40\,m$. BS, SR, CSDCL and DCN represent backward sequence, super-resolution network with PixelShuffle or sub-pixel convolution initialized to convolution NN resize (ICNR) [50], cross-sequence distance consistency loss and deformable convolutions respectively. The input resolution is 512×256 pixels.

use the standard test set of 697 images. The length of the training sequence is set to 3.

B. Evaluation

We evaluate FisheyeDistanceNet's depth and distance estimation results using the metrics proposed by Eigen et al. [41] to facilitate comparison. The quantitative results shown in the Table I illustrate that our scale-aware self-supervised approach outperforms all the state-of-the-art monocular approaches. We could not leverage the Cityscapes dataset into our training regime to benchmark our scale-aware framework, due to absence of odometry data.

Since the projection operators are different, previous SfM approaches will not be feasible on Fisheye Woodscape dataset without adaption of the network and projection model. It is important to note that due to the geometry of the fisheye, it would not be a fair comparison to evaluate the distance estimates up to $80\,m$. Our fisheye automotive cameras also undergo high data compression and our dataset contains images of inferior quality when compared with KITTI. Our fisheye cameras can perform well up to a range of $40\,m$. Therefore, we also report results on a $30\,m$ and a $40\,m$. range (see Table I).

C. Fisheye Ablation Study

We conduct an ablation study to evaluate the importance of different components. We ablate the following components and report their impact on the distance evaluation metrics in Table II: (i) *Remove Backward Sequence*: The network is only trained for the forward sequence which consists of two warps as explained in Section II-E; (ii) *Additionally remove Super Resolution using sub-pixel convolution*: Removal of sub-pixel convolution has a huge impact on Woodscape compared to KITTI. This is mainly attributed to the fisheye model, as far-away objects are tiny and cannot be resolved accurately with naive nearest neighbor interpolation or transposed convolution [26]; (iii) *Additionally remove cross-sequence distance consistency loss*: Removing the CSDCL mainly diminishes the baseline; (iv) *Additionally remove deformable convolutions*: If we remove all the major components, especially deformable convolution layers [24], our model will fail miserably as the distortion introduced by fisheye model will not be learned correctly by normal convolutional layers.

V. Conclusion

We propose a novel self-supervised training strategy to obtain metric distance maps on unrectified fisheye images. Through extensive experiments, we show that our FisheyeDistanceNet establishes a new state of the art in self-supervised monocular distance and depth estimation on Fisheye WoodScape and KITTI dataset respectively. We obtain promising results demonstrating the potential of using a CNN based approach for deployment in commercial automotive systems, in particular for replacing current classical depth estimation approaches. To encourage further research on fisheye distance estimation, we will release the dataset as a part of WoodScape [1] project.

REFERENCES

[1] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9308–9318.

[2] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Computer vision in automated parking systems: Design, implementation and challenges," *Image and Vision Computing*, vol. 68, pp. 88–101, 2017.

[3] A. Dahal, J. Hossen, C. Sumanth, G. Sistu, K. Malhan, M. Amasha, and S. Yogamani, "Deeptrailerassist: Deep learning based trailer detection, tracking and articulation angle estimation on automotive rear-view camera," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[4] J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Vision-based driver assistance systems: Survey, taxonomy and advances," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2032–2039.

[5] M. Drulea, I. Szakats, A. Vatavu, and S. Nedevschi, "Omnidirectional stereo vision using fisheye lenses," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2014, pp. 251–258.

[6] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 141–148.

[7] 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.

[8] M. Uřičář, P. Křížek, G. Sistu, and S. Yogamani, "Soilingnet: Soiling detection on automotive surround-view cameras," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 67–72.

[9] M. Yahiaoui, H. Rashed, L. Mariotti, G. Sistu, I. Clancy, L. Yahiaoui, V. R. Kumar, and S. Yogamani, "Fisheyemodnet: Moving object detection on surround-view cameras for autonomous driving," *arXiv preprint arXiv:1908.11789*, 2019.

[10] 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 International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[11] N. Tripathi, G. Sistu, and S. Yogamani, "Trained trajectory based automated parking system using visual slam," *arXiv preprint arXiv:2001.02161*, 2020.

[12] V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Monocular fisheye camera depth estimation using sparse lidar supervision," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2853–2858.

[13] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, "Omnidepth: Dense depth estimation for indoors spherical panoramas," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 448–465.

[14] C. Godard, O. Mac Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," *arXiv preprint arXiv:1806.01260*, 2018.

[15] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[16] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[17] C. Wang, J. Miguel Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] S. Li, "Binocular spherical stereo," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 4, pp. 589–600, 2008.

[19] C. Ma, L. Shi, H. Huang, and M. Yan, "3d reconstruction from full-view fisheye camera," *arXiv preprint arXiv:1506.06273*, 2015.

[20] S. Pathak, A. Moro, A. Yamashita, and H. Asama, "Dense 3d reconstruction from two spherical images via optical flow-based equirectangular epipolar rectification," in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2016, pp. 140–145.

[21] S. Li and K. Fukumori, "Spherical stereo for the construction of immersive vr environment," in *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE, 2005, pp. 217–222.

[22] J. Huang, Z. Chen, D. Ceylan, and H. Jin, "6-dof vr videos with a single 360-camera," in *2017 IEEE Virtual Reality (VR)*. IEEE, 2017, pp. 37–44.

[23] 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.

[24] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.

[25] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[26] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, p. e3, 2016.

[27] L. Zhou, J. Ye, M. Abello, S. Wang, and M. Kaess, "Unsupervised learning of monocular depth estimation with bundle adjustment, super-resolution and clip loss," *arXiv preprint arXiv:1812.03368*, 2018.

[28] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[29] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[31] Z. Arıcan and P. Frossard, "Dense depth estimation from omnidirectional images," 2009.

[32] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *arXiv preprint arXiv:1704.07804*, 2017.

[33] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding," *arXiv preprint arXiv:1810.06125*, 2018.

[34] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8001–8008.

[35] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5667–5675.

[36] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1983–1992.

[37] Y. Zou, Z. Luo, and J.-B. Huang, "Df-net: Unsupervised joint learning of depth and flow using cross-task consistency," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 36–53.

[38] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, "Unsupervised learning of geometry from videos with edge-aware depth-normal consistency," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 240–12 249.

[40] V. Guizilini, R. Ambrus, S. Pillai, and A. Gaidon, "Packnet-sfm: 3d packing for self-supervised monocular depth estimation," *arXiv preprint arXiv:1905.02693*, 2019.

[41] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *CoRR*, vol. abs/1406.2283, 2014. [Online]. Available: http://arxiv.org/abs/1406.2283

[42] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[43] 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.
[44] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.
[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.
[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167
[48] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.
[49] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," *arXiv preprint arXiv:1811.08883*, 2018.
[50] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," *arXiv preprint arXiv:1707.02937*, 2017.
[51] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2650–2658.