Depth360: Monocular Depth Estimation using Learnable Axisymmetric Camera Model for Spherical Camera Image

Noriaki Hirose$^1$ and Kosuke Tahara$^1$

Abstract—Self-supervised monocular depth estimation has been widely investigated to estimate depth images and relative poses from RGB images. This framework is attractive for researchers because the depth and pose networks can be trained from just time sequence images without the need for the ground truth depth and poses.

In this work, we estimate the depth around a robot (360° view) using time sequence spherical camera images, from a camera whose parameters are unknown. We propose a learnable axisymmetric camera model which accepts distorted spherical camera images with two fisheye camera images. In addition, we trained our models with a photo-realistic simulator to generate ground truth depth images to provide supervision. Moreover, we introduced loss functions to provide floor constraints to reduce artifacts that can result from reflective floor surfaces. We demonstrate the efficacy of our method using the spherical camera images from the GO Stanford dataset and pinhole camera images from the KITTI dataset to compare our method’s performance with that of baseline method in learning the camera parameters.

I. INTRODUCTION

Accurately estimating three-dimensional (3D) information of objects and structures in the environment is essential for autonomous navigation and manipulation [1], [2]. Self-supervised monocular depth estimation without ground truth (GT) depth images is one of the most popular approaches for obtaining 3D information [3]–[7]. In self-supervised monocular depth estimation, there are several limitations including that this method requires the camera parameters, there is no scaling of the estimated depth image, and it functions poorly for highly reflective objects.

This paper proposes a novel self-supervised monocular depth estimation approach for a spherical camera image view. We propose a learnable axisymmetric camera model to handle images from a camera whose camera parameter is unknown. Because our learnable camera model can handle unknown distorted images, such as spherical camera images based on two fisheye images, we can obtain 360° 3D point clouds all around a robot from only one camera.

In addition to self-supervised learning with real images, we rendered many pairs of spherical RGB images and their corresponding depth images from a photo-realistic robot simulator. In training, we mixed these rendered images with real images to achieve sim2real transfer in an attempt to provide scaling for the estimated depth. Moreover, we introduce additional cost functions to improve the accuracy of depth estimation for reflective floor areas. We provide supervision for estimated depth images from the future and past robot footprints, which are obtained from the reference velocities in the data collection. Main contributions are summarized as:

- A novel learnable axisymmetric camera model capable of handling distorted images with unknown camera parameters,
- Sim2real transfer using ground truth depth from a photo-realistic simulator to sharpen the estimated depth image and provide scaling,
- Proposal of novel loss functions that use the robot footprints and trajectories to provide constraints against reflective floor surfaces.

In addition, we blended front-and back-side fisheye images to reduce the occluded area for image prediction in self-supervised learning. As a result of the above approaches, our method can estimate the depth image without the large artifacts that often result from a low-resolution spherical image, thus allowing our method fasts computation useful for online settings.

Our method was trained and evaluated on the GO Stanford (GS) dataset [9]–[11], as depicted in in Fig 1[a], with time sequence spherical camera images and associated reference velocities for dataset collection. Moreover, we evaluated our proposed axisymmetric camera model with the
KITTIdataset [12], as depicted in Fig. 1[b], to provide a comparison with other learnable camera models [13], [14]. We quantitatively and qualitatively evaluated our method with the GS and KITTIdatasets.

II. RELATED WORK

Monocular depth estimation using self-supervised learning has been widely investigated by several approaches that use deep learning [4], [6], [13], [15]–[34]. Zhou et al. [3], and Vijayanarasimhan et al. [35] applied spatial transformer modules [36] based on the knowledge of structure from motion to achieve self-supervised learning from time sequence images. Since the publication of [3] and [35], several subsequent studies have attempted to estimate accurate depth using different methods, including a modified loss function [37], an additional loss function to penalize geometric inconsistencies [5], [7], [13], [38], [39], a probabilistic approach to estimate the reliability of depth estimation [34], [40], masking dynamic objects [4], [13], and an entirely novel model structure [19], [41], [42]. We review the two categories most related to our method.

Fisheye camera image. Various approaches have attempted to estimate depth images from fisheye images. [43] proposed supervised learning with sparse GT from LiDAR. [44]–[46] leveraged multiple fisheye cameras to estimate a 360° depth image. Similarly, Kumar et al. proposed self-supervised learning approaches with a calibrated fisheye camera model [47] and semantic segmentation [48].

Learning camera model. Gordon et al. [13] proposed a self-supervised monocular depth estimation method that can learn the camera’s intrinsic parameters. Vasiljevic et al. [14] proposed a learnable neural ray surface (NRS) camera model for use in highly distorted images.

In contrast to most related work [14], our learnable asymmetric camera model can be fully differentiable without using the softmax approximation. Hence, end-to-end learning can be achieved without adjusting the hyperparameters during training. In addition, we provide supervision for the estimated depth images by using a photo-realistic simulator and robot trajectories from the dataset. As a result, our method can accurately estimate depth from low-resolution spherical images.

III. PROPOSED METHOD

From the process in Fig. 2 we designed the following cost functions to train the depth and pose networks:

\[ J = J_{\text{bimg}} + \lambda_d J_{\text{depth}} + \lambda_f J_{\text{floor}} + \lambda_p J_{\text{pose}} + \lambda_s J_{\text{sm}}. \]

(1)

Since our spherical camera can capture the 360° view around the robot, the occluded area between two consecutive frames is less than that when using a pinhole camera. To leverage this advantage, we propose \( J_{\text{bimg}} \), which is an occlusion-aware image loss, inspired by monodepth2 [37]. \( J_{\text{depth}} \) penalizes the depth difference using the GT depth from photo-realistic simulator. By penalizing \( J_{\text{depth}} \) with \( J_{\text{bimg}} \), our model can learn sim2real transfer and can thereby estimate accurate depths from real images. \( J_{\text{floor}} \) is the proposed loss function that provides supervision against floor areas by using robot’s footprint and trajectory. In addition, \( J_{\text{pose}} \) penalizes the difference between the estimated pose and the GT pose calculated from the reference velocities in the dataset. \( J_{\text{sm}} \) penalizes the discontinuity of the estimated depth using the exact same objective, following [16], [37].

In the following sections, we first present \( J_{\text{bimg}} \) and \( J_{\text{pose}} \) in the overview of our training process. Then, we explain our camera model, with \( J_{\text{depth}} \) and \( J_{\text{floor}} \) as our main novel contributions. We define the robot and camera coordinates based on the robot pose \( X \), as shown in Fig. 3. \( \Sigma X_f \) is the base coordinate of the robot. In addition, \( \Sigma X_f^1 \) and \( \Sigma X_f^v \) are the coordinates of the front-and back-side fisheye cameras in the spherical camera, respectively. Our method assumes that the relative poses between each coordinate are known.

A. Overview

1) Process of depth estimation: Fig. 2[a] presents the calculation of depth estimation for real images. Because there are no GT depth images, we employed self-supervised learning approach using time sequence images. Unlike previous approaches [3], [37], our spherical camera can capture both the front-and back-side of the robot. Hence, we
propose a cost function to blend the front-and back-side images and thereby reduce the negative effects of occlusion. We feed the front-side images $I_{Af}$ and back-side image $I_{Bf}$ at camera pose $A$ into the depth network $f_{\text{depth}}$ to estimate the corresponding depth images as $D_{Af}, D_{Bf} = f_{\text{depth}}(I_{Af}, I_{Bf})$. Here, $D_{Af}$ and $D_{Bf}$ are estimated for each camera coordinate $\Sigma_{Af}$ and $\Sigma_{Bf}$, respectively. By back-projection $f_{\text{backproj}}()$ with our proposed camera model, we can obtain the corresponding point clouds $Q_{Af}$ and $Q_{Bf}$.

$$Q_{Af} = f_{\text{backproj}}(D_{Af}), \quad Q_{Bf} = f_{\text{backproj}}(D_{Bf}) \quad (2)$$

“Trans.” in Fig. 3[a] transforms the coordinates of the estimated point clouds as follows:

$$Q_{Af}^{Rf} = T_{Af}Q_{Af}, \quad Q_{Af}^{Bf} = T_{Af} \cdot T_{fb}^{-1} Q_{Bf}, \quad (3)$$

$$Q_{Af}^{Rf} = T_{Bf} \cdot T_{Af}Q_{Af}, \quad Q_{Bf}^{Rf} = T_{Bf} \cdot T_{Af} \cdot T_{fb}^{-1} Q_{Bf}.$$.

Here, $Q_{Af}^{Rf}$ denotes the point clouds on the coordinate $\Sigma_{Af}$ estimated from image $I_{Af}$. $T_{Af}$ is the estimated transformation matrix between $\Sigma_{Af}$ and $\Sigma_{Bf}$. $T_{Bf}$ is the known transformation matrix between $\Sigma_{Bf}$ and $\Sigma_{Af}$. By projecting these point clouds onto the image coordinates of $I_{Af}$ or $I_{Bf}$, we can estimate four matrices $M_{Af}^{Rf}$, $M_{Af}^{Bf}$, $M_{Bf}^{Rf}$, $M_{Bf}^{Bf}$.

$$M_{Af}^{Rf} = f_{\text{proj}}(Q_{Af}^{Rf}) \quad (4)$$

where, $\alpha = \{f, b\}$ and $\beta = \{f, b\}$. According to [36], we estimate $I_{Af}$ by sampling the pixel value of $I_{Bf}$ as $I_{Af} = f_{\text{sample}}(M_{Af}^{Rf}, I_{Bf})$. Here $I_{Af}^{Rf}$ denotes the estimated image of $I_{Af}$ by sampling $I_{Bf}$. Note that we estimated four images from the combination of $\alpha = \{f, b\}$ and $\beta = \{f, b\}$, as shown in Fig. 2[a]. We calculate the blended image loss $J_{\text{img}}$ to penalize the model during training.

$$J_{\text{img}} = \lambda_1 J_{L1} + \lambda_2 J_{\text{SSIM}} \quad (5)$$

$$J_{L1} = \sum_{\alpha \in \{f, b\}} \min_{\beta \in \{f, b\}} (M[I_{Af}^{Rf} - I_{Af}^{Bf}], M[I_{Af}^{Bf} - I_{Af}^{Rf}]),$$

$$J_{\text{SSIM}} = \sum_{\alpha \in \{f, b\}} f_{\text{min}}(M_{\text{dsim}}(I_{Af}^{Rf}, I_{Af}^{Bf}), M_{\text{dsim}}(I_{Af}^{Bf}, I_{Af}^{Rf})).$$

Here, $f_{\text{min}}(\cdot, \cdot)$ selects a smaller value at each pixel and calculates the mean of all the pixels. $M_{\text{dsim}}(\cdot, \cdot)$ is the pixel-wise structural similarity (SSIM) [49], following [37]. By selecting a smaller value in $f_{\text{min}}(\cdot, \cdot)$, we can equivalently select the non-occluded pixel value of $I_{Af}^{Rf}$ or $I_{Af}^{Bf}$ to calculate $L1$ and $\text{SSIM}$ [37]. In addition, $M$ is a mask to remove the pixels without RGB values and those of the robot itself, which are depicted in gray color in Fig. 2[a].

2) Process of pose estimation: Fig. 2[a] denotes the process to estimate $T_{AB}$. Unlike previous monocular depth estimation approaches, we use the GT transformation matrix $T_{AB}$ from the integral of the reference velocities $\{v_i, \omega_i\}_{i=0\ldots N_t}$ to move between poses $A$ and $B$. $J_{\text{pose}}$ is designed as $J_{\text{pose}} = \sum(T_{AB} - T_{AB})^2$.

B. Axisymmetric camera model

To handle the dataset without camera parameters and videos in the wild for depth estimation, we propose a novel camera model. Our proposed camera model, depicted in Fig. 2 has some constraints, in that the projection plane(yellow lines in Fig. 4) is axisymmetric and convex upwards. Following our design, it is a fully differentiable model without approximations by a softmax function.

In Fig. 4, $[X_i, Y_i, Z_i]$ is the $i$-th estimated 3D point, and $[x_i, y_i]$ is the corresponding position between $-1.0$ and $1.0$ on the XY plane. We define $[x_i, y_i] = [\tilde{x}_i/r_x + o_x, \tilde{y}_i/r_y + o_y]$, where $[\tilde{x}_i, \tilde{y}_i]$ is the position on the image plane, $r_x$ and $r_y$ are learnable variables for the angle of view, and $o_x$ and $o_y$ are learnable variables for the offset between the image plane and the $XY$ plane. The $W$ axis is defined as the direction from the origin to $[X_i, Y_i, 0]$.

According to the above constraints, the projection plane on the $XY$ plane can be depicted as a unit circle whose center is the origin, as shown in Fig. 4[a]. In addition, the projection plane on the $WZ$ plane monotonically decreases. In our camera model, we designed the projection plane on the $WZ$ plane with consecutive line segments by the points $[b_i, h_i]_{i=0\ldots N_t}$, as shown in Fig. 4[b].

1) Convex upward projection plane: To derive $[b_i, h_i]_{i=0\ldots N_t}$ which ensures the convex upward projection plane, our camera parameter network $f_{\text{cam}}()$ estimates the camera parameters as follows:

$$\{\Delta b_i, \Delta h_i / \Delta b_i\}_{i=0\ldots N_t}, r_x, r_y, o_x, o_y = f_{\text{cam}}(f_{\text{img}}) \quad (6)$$

where $\Delta h_i = \tilde{h}_{i-1} - \tilde{h}_i$, $\Delta b_i = \tilde{b}_{i-1} - \tilde{b}_i$, and $f_{\text{img}}$ are the image features from the depth encoder. $[b_i, h_i]$ are the $i$-th points before normalization.

In $f_{\text{cam}}(\cdot)$, we provide a sigmoid function to achieve $\Delta h_i / \Delta b_i > 0$ and $\Delta h_i > 0$, to ensure a convex upward constraint. By simple algebra with $\tilde{h}_{N_t} = 0.0$ and $\tilde{b}_{N_t} = 1.0$, we can obtain $[b_i, h_i]_{i=0\ldots N_t}$. By performing
normalization to stabilize the training process, we can obtain $h_i = h_i / \sum_{k=0}^{N_b} h_k$ and $b_i = b_i / \sum_{k=0}^{N_b} b_k$.

2) Back-projection: Back-projection is the process of calculating the point clouds $[X_i, Y_i, Z_i]$ from the estimated depth $Z_i$ at $[x_i, y_i]$. First, we calculate $[w_i, z_i]$, which is the intersection point on the projection plane. The line of the $j$-th line segment can be expressed as:

$$Z = \frac{\Delta h_j}{\Delta b_j} \cdot W + \frac{h_{j-1} b_j - h_j b_{j-1}}{\Delta b_j} = \alpha_j W + \beta_j.$$  \hspace{1cm} (7)

To be a fully differentiable process, we calculate the intersection points between all lines of the projection plane and the vertical line $W = w_i = (\sqrt{x_i^2 + y_i^2})$. Then, we select the minimum height at all intersections as $z_i$ because the projection plane is upwardly convex.

$$z_i = \min\{\{\alpha_j w_i + \beta_j\}_{j=1,...,N_b}\}. \hspace{1cm} (8)$$

Note that the min function is a differentiable function. As a result, we can obtain $X_i = \frac{z_i}{\gamma_i} \cdot x_i$ and $Y_i = \frac{z_i}{\gamma_i} \cdot y_i$.

3) Projection: The projection process calculates $[x_i, y_i]$ from $[X_i, Y_i, Z_i]$. Similar to back-projection, we first calculate the intersection point $[w_i, z_i]$ and derive $[x_i, y_i]$. The line between the origin and $[X_i, Y_i, Z_i]$ can be expressed as $Z = \frac{Z_i}{W_i} \cdot W = \gamma_i W$. Much like the back-projection process, the minimum value at all intersections on the $Z$ axis can be $z_i$. Hence, $z_i$ and $w_i$ can be derived as follows:

$$z_i = \min\left\{\left\{\frac{\gamma_i \beta_j}{\gamma_i - \alpha_j}\right\}_{j=1,...,N_b}\right\}, \hspace{0.5cm} w_i = z_i / \gamma_i. \hspace{1cm} (9)$$

Thus, $[x_i, y_i] = \left[w_i \cdot X_i, w_i \cdot Y_i\right]$. Here, $W_i = \sqrt{x_i^2 + y_i^2}$.

C. Floor loss

The highly reflective floor surface in indoor environments can often cause significant artifacts in depth estimation. Our floor loss function provides geometric supervision for floor areas. Assuming that the camera is horizontally mounted on the robot and its height is known, the floor loss function is constructed by two components: $J_{floor} = J_{fcl} + J_{lbld}$.

1) Footprint consistency loss $J_{fcl}$: The robot footprint is almost horizontally flat, and its height is obtained from the camera mounting position. According to the method shown below, we obtain the GT depth $\{\tilde{D}_{A^\alpha}\}_{\alpha=(f,b)}$ only around the robot footprint between $\pm M_r$ steps and provide the supervision as follows:

$$J_{fcl} = \sum_{\alpha \in \{f,b\}} \frac{1}{N_m} \sum_{m=1}^{N_m} \sum_{i=1}^{N_m} M_f [\tilde{D}_{A^\alpha} - D_{A^\alpha}],$$  \hspace{1cm} (10)

where $M_f$ is the mask, which masks out the pixels without the GT values in $D_{A^\alpha}$, and $N_m$ is the number of pixels with the GT depth. To make $D_{A^\alpha}$, we take the point clouds of the robot footprint between $\pm M_r$ steps as follows:

$$Q_{foot} = \{Q_{foot}^0, \cdots, Q_{foot}^{M_r}\}, \hspace{1cm} (11)$$

where $Q_{foot}^i = T_i^T Q_{foot}^0$ is the point cloud of the robot footprint at the pose of the $i$-th step, where $Q_{foot}^0$ is that at the origin (= pose $A$) and $T_i$ is the transformation matrix between the origin and $i$-th robot pose from the teleoperator’s velocities, similar to $\bar{T}_{AB}$. $D_{A^\alpha}$ can be derived as:

$$\tilde{D}_{A^\alpha}[x_i^\alpha, y_i^\alpha] = Z_i^\alpha,$$  \hspace{1cm} (12)

where $[x_i^\alpha, y_i^\alpha] = f_{proj}(T_{ro} Q_{foot}^j)$ and $Z_i^\alpha$ are the values of the $Z$ axis of $T_{ro} Q_{foot}^j$. $T_{ro}$ is the known transformation matrix from $\Sigma_x^r$ to $\Sigma_x^s$. $Q_{foot}^j$ is the $j$-th 3D point in $Q_{foot}$. By assigning all point clouds into $D_{A^\alpha}$ using (12), we can take $\tilde{D}_{A^\alpha}$ to calculate $J_{fcl}$. Note that $D_{A^\alpha}$ is a sparse matrix. No GT pixel in $D_{A^\alpha}$ is excluded by $M_f$ in $J_{fcl}$.

2) lower boundary loss $J_{lbld}$: In indoor scenes, floor areas often reflect ceiling lights. This can cause it to appear that there are holes on the floor in the estimated depth image. To provide a lower boundary for the height of estimated point clouds, we observe two key points: 1) the camera is horizontally mounted on the robot, and 2) most objects around the robot are higher than the floor. One exception would be anything that is downstairs, which would be lower than the floor. However, it is rare in to find such occurrences in the dataset, because teleoperation around the stairs is dangerous and ill-advised.

To provide the constraint for the $Y$ axis (= hight) value of the estimated point clouds, $J_{lbld}$ can be given as follows:

$$J_{lbld} = \sum_{\alpha \in \{f,b\}} \frac{1}{N} \sum_{i=1}^{N} (\max(0,0, Y_{A^\alpha} - h_{cam})), \hspace{1cm} (13)$$

where $Q_{\alpha} = \{X_{A^\alpha}, Y_{A^\alpha}, Z_{A^\alpha}\}$. $J_{lbld}$ penalizes $Y_{A^\alpha}$, which is larger (= lower) than the floor height (= $h_{cam}$).

D. Sim2real transfer with $J_{depth}$

In conjunction with self-supervised learning using real-time sequence real images, we used the GT depth image $\{D_{sl}, D_{sb}\}$ from a photo-realistic robot simulator [50], as shown in Fig. 2[b]. Although there is an appearance gap between real and simulated images, the GT depth from the simulator can help the model understand the 3D geometry of the environment from the image. Here, $J_{depth} = \sum_{\alpha \in \{f,b\}} d_{depth} (\tilde{D}_{A^\alpha}, D_{A^\alpha})$, where $D_{sl}, D_{sb} = f_{depth}(I_{sl}, I_{sb}), I_{sl}$ and $I_{sb}$ are front-and back-side fishey images, respectively, from the simulator. We employed the same metric $d_{depth}$ as the baseline method [51] to measure the depth differences. To achieve a sim2real transfer, we simultaneously penalized $J_{depth}$ and $J_{binimg}$.

IV. EXPERIMENT

A. Dataset

Our method was mainly evaluated using the GO Stanford (GS) dataset with time-sequence spherical camera images. In addition, we used the KITTI dataset with pinhole camera images for comparison with the baseline methods, which attempts to learn the camera parameters.

1) GO Stanford dataset: We used the GS dataset [9][11] with time sequence spherical camera images ($256 \times 128$) and the reference velocities, which were collected by teleoperating turtlebot2 with Ricoh THETA S. The GS dataset contains 10.3 hours of data from twelve buildings at the Stanford
To evaluate our proposed camera model against the baseline methods, we employed the KITTI raw dataset [12]. Similar to the baseline methods, we separated the KITTI raw dataset via Eigen split [53] with 40,000 images for training, 4,000 images for validation, and 697 images for testing. To compare with baseline methods, we employed the widely used 640×192 image size as input.

2) KITTI dataset: To evaluate our proposed camera model against the baseline methods, we employed the KITTI raw dataset [12]. Similar to the baseline methods, we separated the KITTI raw dataset via Eigen split [53] with 40,000 images for training, 4,000 images for validation, and 697 images for testing. To compare with baseline methods, we employed the widely used 640×192 image size as input.

B. Training

In training with the GS dataset, we randomly selected 12 real images from the training dataset as \( \{ I_{A} \}_{\alpha \in \{ f, b \}} \). Then, we randomly selected \( \{ I_{B} \}_{\alpha \in \{ f, b \}} \) between ±5(= \( N_g \)) steps. In addition, we selected 12 simulator images \( \{ I_{\text{sim}} \}_{\alpha \in \{ f, b \}} \) and the GT depth \( \{ D_{\text{GT}} \}_{\alpha \in \{ f, b \}} \) for \( J_{\text{depth}} \).

To calculate \( J \), we set the camera height and camera model size to \( h_{\text{cam}} = 0.57 \) and \( N_{\text{e}} = 32 \), respectively. The weighting factors for the loss function were designed as \( \lambda_1 = 0.85 \), \( \lambda_2 = 0.15 \), \( \lambda_5 = 0.001 \), \( \lambda_7 = 0.1 \), and \( \lambda_9 = 1.0 \). \( \lambda_1 \) and \( \lambda_5 \) were exactly the same as in previous studies. We only determine \( \lambda_f \) by trial and error.

The robot footprint shape was defined as a circle with a diameter of 0.5 m. The point cloud of the footprint \( \mathcal{C}_{\text{foot}} \) was set as 1,400 points inside the circle. The number of steps for the robot footprints was \( M_r = 5 \) for \( J_{\text{cl}} \).

The network structures of \( f_{\text{depth}}(\cdot) \) and \( f_{\text{pose}}(\cdot) \) were exactly the same as those of monodepth2 [37]. In addition, \( f_{\text{cam}}(\cdot) \) was designed with three convolutional layers, with the ReLu function and two fully connected layers with a sigmoid function, to estimate the camera parameters. We used the Adam optimizer with a learning rate of 0.0001 and conducted 40 epochs.

During training, we iteratively calculated \( J \) and derived the gradient to update all the models. Hence, we could simultaneously penalize \( J_{\text{depth}} \) from the simulator and the others from the real images to achieve a sim2real transfer.

For the KITTI dataset, we employed the source code of monodepth2 [37] and replaced the camera model with our proposed camera model. The other settings were exactly the same as those of monodepth2, to focus the experimentation...
We trained the depth network by the cost function of the above baseline to train the models. In addition, we proposed to provide supervision for the estimated depth using the GT depth from the photo-realistic simulator. Moreover, we trained our models by mixing the KITTI, Cityscape [55], bike [5], and GS datasets [11] to evaluate its ability to handle various cameras and evaluated on KITTI images. During training, all images were aligned into KITTI’s image size by center cropping. In GS dataset, we use front-side fisheye images. Our method showed improved performance by adding datasets from various cameras with various distortions, as shown at the bottom of Table II.

D. Qualitative Analysis

Figure 6 shows the estimated depth images from simulated images and real images from the GS dataset. The depth images estimated by monodepth2 are blurred. This is caused by the small size of the input image and the camera model’s error. The GT depth from the simulator can sharpen the simulated depth of images in monodepth2 with sim. GT. However, there are many artifacts, particularly on the reflected floor. Alhashim et al. observed errors in the estimated depth from real images. Alhashim et al. failed sim2real transfer because the depth network was trained only from simulated images. However, our method (rightmost side) can accurately estimate depth images of both real and simulated images by reducing these artifacts. Additional examples are provided in the supplemental videos.

Finally, we present the depth images of KITTI in Fig. [b]. Our method can handle the pinhole camera image and estimate an accurate depth image without camera calibration.

V. CONCLUSIONS

We proposed a novel learnable axisymmetric camera model for self-supervised monocular depth estimation. In addition, we proposed to provide supervision for the estimated depth using the GT depth from the photo-realistic simulator. By mixing real and simulator images during training, we can achieve a sim2real transfer in depth estimation. Additionally, we proposed loss functions to provide the constraints for the floor to reduce artifacts that may result from reflective floors. The effectiveness of our method was quantitatively and qualitatively validated using the GS and KITTI datasets.

C. Quantitative Analysis

Table II shows the ablation study of our method and the results of three baseline methods for comparison. We trained the following baseline methods with the same dataset.

monodepth2 [37] We applied the following half-sphere model [54] into monodepth2 [37], instead of the pinhole camera model, and trained depth and pose networks.

- back-projection: \((x_i, y_i, Z_i) \rightarrow (X_i, Y_i)\),
  \[
  X_i = \frac{Z_i}{\sqrt{1 - x_i^2 - y_i^2}} \cdot x_i, \\
  Y_i = \frac{Z_i}{\sqrt{1 - x_i^2 - y_i^2}} \cdot y_i
  \] (14)

- Projection: \((X_i, Y_i, Z_i) \rightarrow (x_i, y_i)\),
  \[
  x_i = \frac{X_i}{\sqrt{X_i^2 + Y_i^2 + Z_i^2}}, \\
  y_i = \frac{Y_i}{\sqrt{X_i^2 + Y_i^2 + Z_i^2}}
  \] (15)

monodepth2 with sim. GT [37], [51] We added \(J_{depth}\) to the cost function of the above baseline to train the models.

Alhashim et al. [51] We trained the depth network by minimizing \(J_{depth}\), which is the same cost function as [51].

From Table II we can observe that our method significantly outperforms all baseline methods. In addition, we confirmed the advantages of our proposed components via the ablation study. The use of the GT depth from the simulator and the learning axisymmetric camera model were fairly effective. In addition, the GT depth can provide the correct scaling to the model.

In Table II we present the quantitative results for the KITTI dataset. Similar to our method, the baseline methods (shown in Table I) learned the camera model. All methods presented in Table II used ResNet-18 for their depth network to allow for fair comparisons.

In our method with known camera parameters, we set \(\{b_q\}_{q=0..N_q}, r_x, r_y, o_x, \) and \(o_y\) as the constant values for the GT camera’s intrinsic parameters. \(\{b_q\}_{q=0..N_q}\) was designed with equal intervals between 0.0 and 1.0. Our method improved the accuracy of depth estimation by learning the camera model. In addition, our method outperformed all baseline methods, including the original monodepth2.

Moreover, we trained our models by mixing the KITTI, Cityscape [55], bike [5], and GS datasets [11] to evaluate its ability to handle various cameras and evaluated on KITTI images. During training, all images were aligned into KITTI’s image size by center cropping. In GS dataset, we use front-side fisheye images. Our method showed improved performance by adding datasets from various cameras with various distortions, as shown at the bottom of Table II.

D. Qualitative Analysis

Figure 6 shows the estimated depth images from simulated images and real images from the GS dataset. The depth images estimated by monodepth2 are blurred. This is caused by the small size of the input image and the camera model’s error. The GT depth from the simulator can sharpen the simulated depth of images in monodepth2 with sim. GT. However, there are many artifacts, particularly on the reflected floor. Alhashim et al. observed errors in the estimated depth from real images. Alhashim et al. failed sim2real transfer because the depth network was trained only from simulated images. However, our method (rightmost side) can accurately estimate depth images of both real and simulated images by reducing these artifacts. Additional examples are provided in the supplemental videos.

Finally, we present the depth images of KITTI in Fig. [b]. Our method can handle the pinhole camera image and estimate an accurate depth image without camera calibration.

V. CONCLUSIONS

We proposed a novel learnable axisymmetric camera model for self-supervised monocular depth estimation. In addition, we proposed to provide supervision for the estimated depth using the GT depth from the photo-realistic simulator. By mixing real and simulator images during training, we can achieve a sim2real transfer in depth estimation. Additionally, we proposed loss functions to provide the constraints for the floor to reduce artifacts that may result from reflective floors. The effectiveness of our method was quantitatively and qualitatively validated using the GS and KITTI datasets.
[44] C. Won, J. Ryu, and J. Lim, “Sweepnet: Wide-baseline omnidirectional depth estimation,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 6073–6079.

[45] R. Komatsu, H. Fujii, Y. Tamura, A. Yamashita, and H. Asama, “360 depth estimation from multiple fisheye images with origami crown representation of icosahedron,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 10092–10099.

[46] Z. Cui, L. Heng, Y. C. Yeo, A. Geiger, M. Pollefeys, and T. Sattler, “Real-time dense mapping for self-driving vehicles using fisheye cameras,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 6087–6093.

[47] V. R. 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 2020 IEEE international conference on robotics and automation (ICRA). IEEE, 2020, pp. 574–581.

[48] V. R. Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader, “Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 61–71.

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

[50] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, “Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 713–720, 2020.

[51] I. Alhashim and P. Wonka, “High quality monocular depth estimation via transfer learning,” arXiv preprint arXiv:1812.11941, 2018.

[52] I. Vasiljevic, V. Guizilini, R. Ambrus, S. Pillai, W. Burgard, G. Shakhnarovich, and A. Gaidon, “Neural ray surfaces for self-supervised learning of depth and ego-motion,” in 2020 International Conference on 3D Vision (3DV), 2020, pp. 1–11.

[53] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in Advances in neural information processing systems, 2014, pp. 2366–2374.

[54] J. Courbon, Y. Mezouar, L. Eckt, and P. Martinet, “A generic fisheye camera model for robotic applications,” in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007, pp. 1683–1688.

[55] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.