# DepthFisheye: Efficient Fine-Tuning of Depth Estimation Models for Fisheye Cameras

Wenbin Wu
East China Normal University
No.3663, North Zhongshan Road, Shanghai, China
51255901049@stu.ecnu.edu.cn

Zhiwei Zhang
Shanghai Jiao Tong University
No.800, Dongchuan Road, Shanghai, China.
zhangzw12139@sjtu.edu.cn

Xin Tan, Zhizhong Zhang, Lizhuang Ma
East China Normal University
No.3663, North Zhongshan Road, Shanghai, China
{xtan,zzzhang,lzma}@cs.ecnu.edu.cn

## Abstract

**Vision foundation models have exhibited exceptional generalization capabilities across various tasks through large-scale pretraining. In the realm of monocular depth estimation, existing models have excelled at predicting relative depth from pinhole camera images. However, while pinhole cameras are widely used, they are not always ideal for scenarios requiring a broader field of view, such as autonomous driving and surveillance. Fisheye cameras, which provide nearly a 180-degree field of view, serve as a cost-effective alternative to LiDAR for close-range depth sensing. However, the scarcity of publicly available fisheye image datasets for depth estimation limits their application. In this paper, we present DepthFisheye, an efficient fine-tuning method that adapts existing depth estimation models from pinhole to fisheye cameras. Our approach addresses both input adaptation—transitioning from pinhole to fisheye images—and output adaptation, transforming relative depth into metric depth. We propose the Distortion-Aware Adapter (DAA) to manage fisheye distortions without network forgetting and introduce the ScaleFormer Head (SFH) to predict global depth scale. Experimental results demonstrate that DepthFisheye significantly enhances the performance of depth estimation models on fisheye images, enabling accurate depth predictions with minimal computational cost. Our code is available at https://github.com/world-executed/DepthFisheye.git**

***Keywords:*** ***Metric depth, monocular depth estimation, fisheye camera, parameter-efficient fine-tuning, adapter***

## 1. Introduction

Recent advancements in computer vision have seen an increasing emphasis on vision foundation models, which significantly enhance the generalization capabilities of base models through large-scale training on diverse datasets. The paradigm of pretraining and fine-tuning has proven widely successful in both natural language processing (NLP) and computer vision (CV), establishing a robust foundation for a variety of downstream tasks. Vision foundation models excel by offering a starting point with strong prior knowledge, which can then be adapted to specific tasks or domains with limited data.

For the task of monocular depth estimation, several foundation models, such as MiDaS [2] and Depth Anything [36], have demonstrated impressive generalization capabilities across a variety of environments, including both indoor and outdoor datasets. These models primarily function as relative depth estimation models, trained predominantly on images captured using pinhole cameras, the most widely employed type of camera. However, despite their versatility, these models are constrained by their training data, which is heavily biased towards pinhole camera images. Fisheye and panoramic cameras, which capture a significantly wider field of view, are largely underrepresented in available datasets, particularly for depth estimation tasks. This imbalance is especially pronounced in the academic domain, where public fisheye datasets for depth estimation are relatively scarce compared to their pinhole camera counterparts.

In real-world scenarios such as autonomous driving, automated parking systems, and surveillance, fisheye cameras offer significant advantages over pinhole cameras. Their ability to capture a field of view nearing 180 degrees allows them to excel in environments that require wide-angle cov-

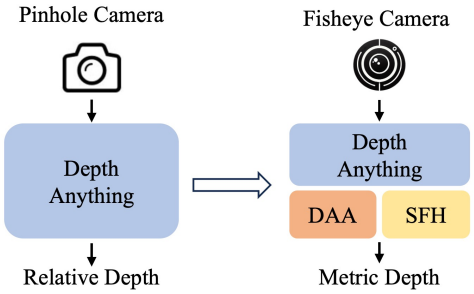


Figure 1. Central conception of our purposed DepthFisheye.

erage, making them ideal for situations such as close-range monitoring or confined spaces, including garages and curbsides. Moreover, fisheye cameras serve as a cost-effective alternative to LiDAR, which may not be as efficient or economical for covering certain areas. While LiDAR scans may not effectively cover certain objects at close range, fisheye cameras can offer a clear perception of nearby scenes. These advantages raise an important question: Can existing vision foundation models, developed primarily for pinhole camera images, be adapted to work effectively with fisheye cameras for depth estimation?

The central motivation of this paper is to leverage existing relative depth estimation models, pretrained on pinhole images, and efficiently fine-tune them to perform absolute depth estimation on images captured by fisheye cameras. This approach has the potential to unlock the capabilities of fisheye cameras in various applications, particularly where field-of-view and cost considerations are paramount.

In this work, we propose DepthFisheye, a novel and efficient fine-tuning method aimed at adapting vision foundation models from pinhole to fisheye cameras. To the best of our knowledge, we are the first to work on fine-tuning fisheye data using depth anything model. The core idea behind DepthFisheye, illustrated in Fig. 1, involves the adaptation of both the input transitioning from pinhole to fisheye images and the output, transforming relative depth into absolute depth. To achieve this, we introduce the Distortion-Aware Adapter (DAA), a mechanism designed to manage the unique distortions introduced by fisheye lenses, preventing catastrophic forgetting while maintaining training efficiency. Additionally, we propose the ScaleFormer Head (SFH), a lightweight global scale prediction module to aid in recovering the absolute depth scale from relative depth predictions.

Through extensive experimentation, we demonstrate the effectiveness and efficiency of our approach across a range of datasets and use cases. The proposed methodology not only significantly enhances the performance of depth estimation models on fisheye images but also offers a practical solution for adapting vision foundation models to diverse camera modalities.

Our contributions can be summarized as the following three points:

- We propose DepthFisheye, the first work to use a relative depth foundation model to fine-tune for fisheye cameras and metric depth estimation.
- We introduce the Distortion-Aware Adapter (DAA) to manage fisheye-specific distortions, preventing catastrophic forgetting while enhancing training efficiency.
- We present the ScaleFormer Head (SFH), a lightweight module for global scale prediction, enabling the recovery of absolute depth from relative depth.

## 2. Related Work

### 2.1. Relative and Metric Depth Estimation

Relative depth estimation and metric depth estimation are two critical approaches in depth prediction tasks. Relative depth estimation focuses on predicting the depth relationship between points or regions in an image without requiring the absolute scale of the scene.

**Relative depth estimation** has ability to balance the differences in depth range between indoor and outdoor scenes, allowing the model to capture more detailed depth variations. Some approaches[36, 10, 37, 29] train on large mixed datasets, significantly enhancing the model's generalization performance. Other methods[3] employ scale-invariant loss to mitigate the effects of scale ambiguity. In the AIGC (AI-generated content) domain, relative depth can represent the scene structure of objects in an image[39, 40], helping to preserve the spatial structure of the original image when generating new styles. However, in downstream tasks like autonomous driving[19, 26], robot control[30], and 3D reconstruction[27], relative depth relationships are insufficient to provide adequate information.

**Metric depth estimation** requires precise knowledge of how far each pixel's corresponding real-world point is from the camera plane, measured in meters. Metric depth estimation is highly influenced by camera parameters[37, 10]. Some methods attempt to incorporate camera parameters into the network's decoder, while others normalize the camera model directly, eliminating scale variations caused by different focal lengths. In terms of network predictions, recent approaches have transformed the depth regression into a depth bins classification[1, 20], making it easier for networks to learn. From CNNs to Transformers[3], there is also growing interest in using diffusion models to generate depth maps[33]. Given that both relative and metric depth estimation have their own strengths, there are now works[41] leveraging relative depth estimation to enhance absolute depth estimation or jointly estimating both relative and absolute depth.

### 2.2. Pinhole and Fisheye Camera Model

The pinhole camera model and fisheye camera model represent two widely used paradigms in computer vision for projecting 3D points onto a 2D image plane.

**The pinhole camera model**, which assumes a simple projection through a small aperture, has been foundational in many applications like structure-from-motion and 3D reconstruction due to its straightforward linear mapping between 3D world coordinates and image coordinates[5]. Despite its popularity, the model struggles with large field-of-view (FoV) applications as it assumes an ideal, distortion-free scenario[12].

**The fisheye camera model**, in contrast, is designed for wide-angle lenses that capture a much broader scene, typically up to 180 degrees[12]. Fisheye cameras introduce significant radial distortion[4], where straight lines appear curved, but they provide a more comprehensive spatial context, making them valuable in applications like autonomous driving and panoramic imaging. Calibration methods[32, 15, 13] for fisheye lenses address the non-linear distortions through specialized projection models, such as the equidistant or stereographic projection models, which more accurately represent the actual image formation process of these lenses[32, 17]. Some works focused on developing unified camera models that handle both pinhole and fisheye lens distortions in a single framework, facilitating more flexible and robust calibration across different imaging setups[16]. This blending of models is particularly useful for multi-camera systems and for applications that require both narrow and wide-angle views[14, 21], improving accuracy and usability in practical systems. Recently, [6]proposed a training-free[28] method that directly adds corrected offsets to the convolutional kernels, enabling adaptation from pinhole cameras to fisheye cameras. However, this method has significant limitations: it can only adapt to a single fisheye camera model and requires the convolutional kernels to be the ones receiving the corresponding offsets.

### 2.3. Parameter Efficient Fine-tuning

Parameter-efficient fine-tuning (PEFT)[8] has emerged as a crucial technique for adapting large pre-trained models to specific tasks with minimal computational and memory overhead. Traditional fine-tuning methods require updating all parameters of a model. PEFT addresses this by updating only a small subset of parameters, leaving the majority of the pre-trained weights frozen[25]. Techniques like adapters[34], insert small trainable modules between layers of the pre-trained network, allowing task-specific adaptation while maintaining the bulk of the model unchanged. Another notable method is LoRA[9, 7, 23] (Low-Rank Adaptation), which reduces the dimensionality of the trainable parameter space, significantly reducing the number of parameters that need updating during fine-tuning. In visual tasks, PEFT (Parameter-Efficient Fine-Tuning) is also widely applied[35]. For example, methods like VPT[11] and AdaptFormer[3] have introduced model fine-tuning techniques specifically for visual tasks. Furthermore, bitfit[38], which fine-tunes only the bias terms of a network, has also demonstrated strong performance on image classification and object detection tasks while keeping the rest of the model fixed.

## 3. Method

In this section, we first introduce the overall structure of the network and the prior knowledge in Sec. 3.1. We begin by encoding the fisheye camera, using distortion-adjusted camera rays as the initial encoding. Sec. 3.2 explains how spherical harmonic transformations (SHT) [42] are applied to upscale the feature dimensions, deriving a dense fisheye camera encoding. The Distortion-Aware Adapter (DAA) is used to fine-tune the intermediate features in the ViT, allowing the model to learn fisheye distortion information, as discussed in Sec. 3.3. Finally, in Sec. 3.4, the Scale-Former Head (SFH) uses learnable queries to obtain global scale information, which is then employed to recover absolute depth.

### 3.1. Preliminary

The relative depth estimation model, exemplified by Depth Anything, consists of two essential components: the image backbone and the depth head. The image backbone, based on a Vision Transformer (ViT) architecture, extracts image features. While the depth head, utilizing a Dense Prediction Transformer (DPT) architecture, predicts relative depth. Our work focuses on performing parameter-efficient fine-tuning of the pretrained Depth Anything model to accommodate image distortions and achieve metric depth estimation.

### 3.2. Dense Fisheye Camera Embedding

To enable the model to effectively capture the characteristics of fisheye cameras, we utilize dense camera embeddings rather than simply relying on an MLP to model the camera's distortion parameters. Using the SynWoodScape dataset as an example, its fisheye camera model follows a polynomial distortion model. For each pixel $(u, v)$ in the image, the distortion-adjusted ray direction is computed from the camera's optical center to that pixel.

First, the pixels are normalized according to the Eq. 1

$$x_n = \frac{u - c_x}{f_x}, y_n = \frac{v - c_y}{f_y}, \tag{1}$$

while $(c_x, c_y)$ denotes the optical center and $(f_x, f_y)$ represents the horizontal and vertical focal lengths, the relationship between the radial distance $r$ and the fisheye angle $\theta$

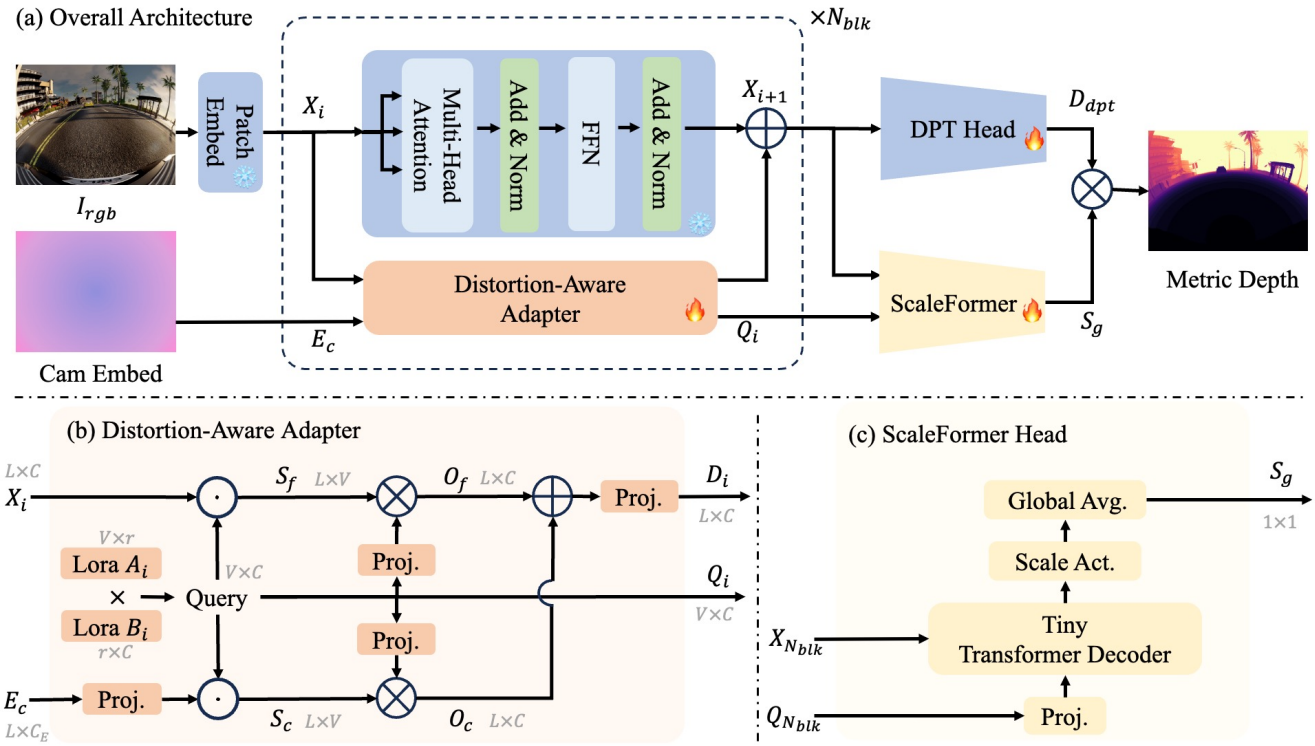


Figure 2. The workflow of our proposed DepthFisheye. **(a)Overall Architecture.**Our method mainly consists of three components: the Depth Anything (DA) model with a blue background, the parallel Distortion-Aware Adapter with an orange background, and the ScaleFormer Head with a yellow background. **(b)Distortion-Aware Adapter.** The DAA module employs low-rank factorized learnable matrices to perform attention modeling between intermediate features and dense fisheye camera encoding. **(c)ScaleFormer Head.** A tiny transformer decoder is used to predict rectified global scale from learnable scale query.

is defined by a distortion polynomial (Eq. **??**). The fisheye angle $\theta$ can be solved numerically. Different fisheye models can be accommodated by applying the corresponding distortion polynomials.

By projecting the points on the image onto a 3D unit sphere, the 3D coordinates $(x, y, z)$ corresponding to each pixel can be obtained through Eq. 2.

$$\begin{aligned} x &= \tfrac{x_n}{r} \cdot \sin(\theta), \\ y &= \tfrac{y_n}{r} \cdot \sin(\theta), \\ z &= \cos(\theta). \end{aligned} \tag{2}$$

We utilize spherical harmonic transformations (SHT) [42] to perform feature upscaling on the camera rays, thereby enhancing the model's ability to capture richer signal representations. Each order of the spherical harmonics provides a decomposition of the spherical signal at different frequency levels. As the order increases, the spherical harmonic basis functions capture finer details of the spherical surface. The transformation is formulated as follows:

$$E_c = Resize(SHT(R_c)), \tag{3}$$

where $SHT_L$ denotes the spherical harmonic transformation, $R_c$ denotes the camera ray with dimensions $(H, W, 3)$, and $E_c$ represents the dense fisheye camera embedding. To maintain feature scale consistency, the features must also be resized to match the scale of the network's features. Thus, the dimension of $E_c$ is $(L, C_E)$, where $L$ corresponds to the patch length and $C_E$ corresponds to the order of the SHT.

### 3.3. Distortion-Aware Adapter

We argue that models pretrained on large-scale pinhole datasets degrade the camera-specific visual biases, making it challenging for the models to generalize to specific camera models, such as fisheye cameras. To address this, we propose a Distortion-Aware Adapter (DAA) to mitigate the generalization issues of the model on fisheye camera models.

The feature extraction process of the ViT backbone typically consists of the following components. First, the patch embedding layer encodes the 2D image $I_{rgb}$ into a sequence $X_0 = PatchEmbed(I_{rgb})$. Subsequently, $N_{blk}$ stacked Transformer blocks are employed for feature extraction, formulated as $X_{i+1} = Block_i(X_i)$, where $i \in \{0, 1, \ldots, N_{blk} - 1\}$. Our DAA module is inserted in parallel into each Transformer blocks, specifically as follows:

$$X_{i+1} = Block_i(X_i) + DAA_i(X_i, E_c). \tag{4}$$

| Methods | AbsRel ↓ | RMSE ↓ | SiLog ↓ | $\delta 1$ ↑ | $\delta 2$ ↑ | $\delta 3$ ↑ |
|---|---|---|---|---|---|---|
| *Domain-specific* | | | | | | |
| OmniDet[18] | 1.0800 | 5.6230 | - | 0.0310 | 0.0640 | 0.7530 |
| BTS[22] | 0.1790 | 2.9980 | - | 0.6530 | 0.8450 | 0.9320 |
| Adabins[1] | 0.0402 | 1.5547 | 0.0971 | **0.9793** | **0.9941** | 0.9972 |
| *Fine-tuned* | | | | | | |
| DA-ViT-S (Full) | 0.0465 | 1.8779 | 0.0998 | 0.9600 | 0.9867 | 0.9945 |
| DA-ViT-S (Lora) | 0.0392 | 1.7047 | 0.0901 | 0.9704 | 0.9914 | 0.9966 |
| DA-ViT-L (Full) | 0.0432 | 1.8523 | 0.0944 | 0.9703 | 0.9901 | 0.9954 |
| Ours-S | 0.0355 | 1.5875 | 0.0850 | 0.9746 | 0.9925 | 0.9968 |
| **Ours-L** | **0.0333** | **1.5409** | **0.0804** | 0.9763 | 0.9931 | **0.9972** |

Table 1. Comparison with other methods in SynWoodScape dataset.

| Methods | AbsRel ↓ | RMSE ↓ | SiLog ↓ | $\delta 1$ ↑ | $\delta 2$ ↑ | $\delta 3$ ↑ |
|---|---|---|---|---|---|---|
| *Domain-specific* | | | | | | |
| OmniDet[18] | 0.1326 | 2.5471 | - | 0.8304 | 0.9287 | 0.9663 |
| BTS[22] | 0.1123 | 2.3981 | - | 0.8432 | 0.9323 | 0.9729 |
| Adabins[1] | 0.1095 | 2.2343 | 0.1573 | 0.8869 | 0.9673 | 0.9873 |
| *Fine-tuned* | | | | | | |
| DA-ViT-S (Full) | 0.0999 | 2.2251 | 0.1561 | 0.8965 | 0.9712 | 0.9890 |
| Ours-S | **0.0978** | **2.1631** | **0.1540** | **0.9030** | **0.9729** | **0.9897** |

Table 2. Comparison with other methods in KITTI360 dataset.

The input to the DAA consists of the features from each layer and the dense camera encoding $E_c$, and its output is the distortion-aware transformed features. Once the DAA is integrated, the weights of each block are frozen, and during the training phase, only the weights of the DAA are updated.

Each layer of the Distortion-Aware Adapter has two learnable low-rank matrices, referred to as $A_i \in \mathbb{R}^{V\times r}$ and $B_i \in \mathbb{R}^{r\times C}$, where $V$ denotes the query length and $C$ denotes the feature dimension. The other dimension of matrices $A_i$ and $B_i$ is set to a small value $r$ to enforce low-rank constraints. The query matrix is obtained by multiplying the two low-rank matrices (Eq. 5).

$$Q_i = A_i \times B_i. \tag{5}$$

We use this query to perform attention modeling with the image features and camera encoding, enabling the network to learn distortion-aware features. For the output $X_i$ from the previous Transformer block, we first compute the similarity between $X_i$ and the query, and then apply Eq. 6 to obtain the attention score $S_f$.

$$S_f = Softmax(\frac{Q_i X_i^T}{\sqrt{C}}), \tag{6}$$

$$S_c = Softmax(\frac{Q_i Proj(E_c)^T}{\sqrt{C}}). \tag{7}$$

Next, a projector is used to project the query to value matrix. This value is multiplied by the corresponding attention score to produce the final output $O_f$, formulated as Eq. 8.

$$O_f = S_f \times Proj(Q_i), \tag{8}$$

$$O_c = S_c \times Proj(Q_i). \tag{9}$$

The dense camera embedding $E_C$ needs to first pass through a projector to adjust its feature dimension, ensuring it matches the query's dimension. Using the aforementioned similarity-based modeling approach, we obtain the output of the camera feature output $O_c$.

$$D_i = Proj(O_f + O_c). \tag{10}$$

In Eq. 10, the final output of the DAA is obtained by adding $O_f$ and $O_c$, followed by passing through a projector initialized to zero. The zero initialization ensures stability dur-

| Components | | Performance | | |
|---|---|---|---|---|
| DAA | SFH | AbsRel ↓ | RMSE ↓ | SiLog ↓ |
| | | 0.0465 | 1.8779 | 0.0998 |
| ✓ | | 0.0370 | 1.6526 | 0.0874 |
| ✓ | ✓ | **0.0355** | **1.5875** | **0.0850** |

Table 3. Ablation study on different components.

ing adapter training. All projector layers in the Distortion-Aware Adapter share parameters, while only the LoRA matrices $A_i$ and $B_i$ are layer-specific. This benefits the fine-tuning of features from low-level to high-level, while significantly reducing the number of parameters, thus achieving efficient fine-tuning.

### 3.4. ScaleFormer Head

Our fine-tuning objectives are twofold: adapting to fisheye images and recovering metric depth. We propose the ScaleFormer Head, which employs a lightweight transformer decoder to predict global scale information. ScaleFormer head takes $X_{N_{blk}}$ and $Q_{N_{blk}}$ as input, denotes the last layer of transformer block and Distortion-Aware adapter respectively. The $Q_{N_{blk}}$ pass through a projector, transformed to scale query. Each query is related to a certain area of feature map, and the tiny transformer decodes each query into its corresponding scale information.

The output is then activated using an appropriate scale activation function, followed by global averaging to obtain the global scale $S_g$.

The choice of activation function is crucial, as it needs to align with the output of the DPT head. Since the output of our pre-trained model's output is relative disparity, which is inversely proportional to the metric depth, the final output requires an inversion operation. Our global scale is applied before this inversion, so sigmoid, which value range from 0 to 1, is ultimately selected as the activation function for the scale.

## 4. Experiment

In the experimental section, we first introduce the datasets used and some basic information about them in Sec. 4.1. Then, in Sec. 4.2, we describe the experimental setup, including the runtime environment, evaluation metrics, and loss functions. In Sec. 4.3, we provide quantitative and qualitative comparisons between our method and other approaches, demonstrating the superiority of our method. Finally, in Sec. 4.4, we conduct ablation experiments to analyze the contribution of each component of the proposed method and compare the parameter counts.

| Component | Params(M) ↓ Ours-S | Ours-L |
|---|---|---|
| Backbone | 22.05 | 304.37 |
| DAA | 1.01 | 5.17 |
| DPT head | 2.72 | 30.94 |
| SFH | 0.09 | 0.13 |
| Total | 25.88 | 340.62 |

Table 4. Parameter of each component in our framework.

| Methods | Params(M) ↓ |
|---|---|
| Omnidet[18] | 34.92 |
| BTS[22] | **21.16** |
| Adabins[1] | 78.21 |
| Ours-S | <u>25.88</u> |

Table 5. Parameter comparison of different method.

### 4.1. Dataset

The two datasets used in this work are SynWoodScape [31] and KITTI360 [24]. SynWoodScape is a publicly available synthetic dataset, with the public version containing 2,000 images. The dataset includes four fisheye cameras labeled *FV*, *RV*, *MVL*, and *MVR*, positioned at the front, rear, left, and right of the vehicle, centered on the ego-vehicle. The distortion equation for each fisheye camera in this dataset follows a polynomial model. Since SynWoodScape is collected in a simulated environment, the images exhibit stylistic differences from real-world images. However, the dataset provides dense depth ground truth, meaning every pixel in the image has an associated depth value. Due to the significant distortion of fisheye cameras compared to pinhole cameras, we set the maximum depth range to 40 meters. We randomly split the dataset into training and validation sets with a 4:1 ratio, and all subsequent evaluation metrics are based on the results from the validation set.

The other dataset used is KITTI360, which includes both pinhole and fisheye camera images along with LiDAR scans. Since KITTI360 does not provide direct depth ground truth, we project the LiDAR data onto the 2D plane using the camera's intrinsic and extrinsic parameters to obtain distance values as depth ground truth. Only fisheye cameras *image_02* and *image_03* are selected as inputs, with a total of 50,000 training samples and 1,000 validation samples. Similarly, the depth ground truth range for KITTI360 is limited to 40 meters. Compared to the SynWoodScape dataset, the depth ground truth in KITTI360 is much sparser, making the model more challenging to train.

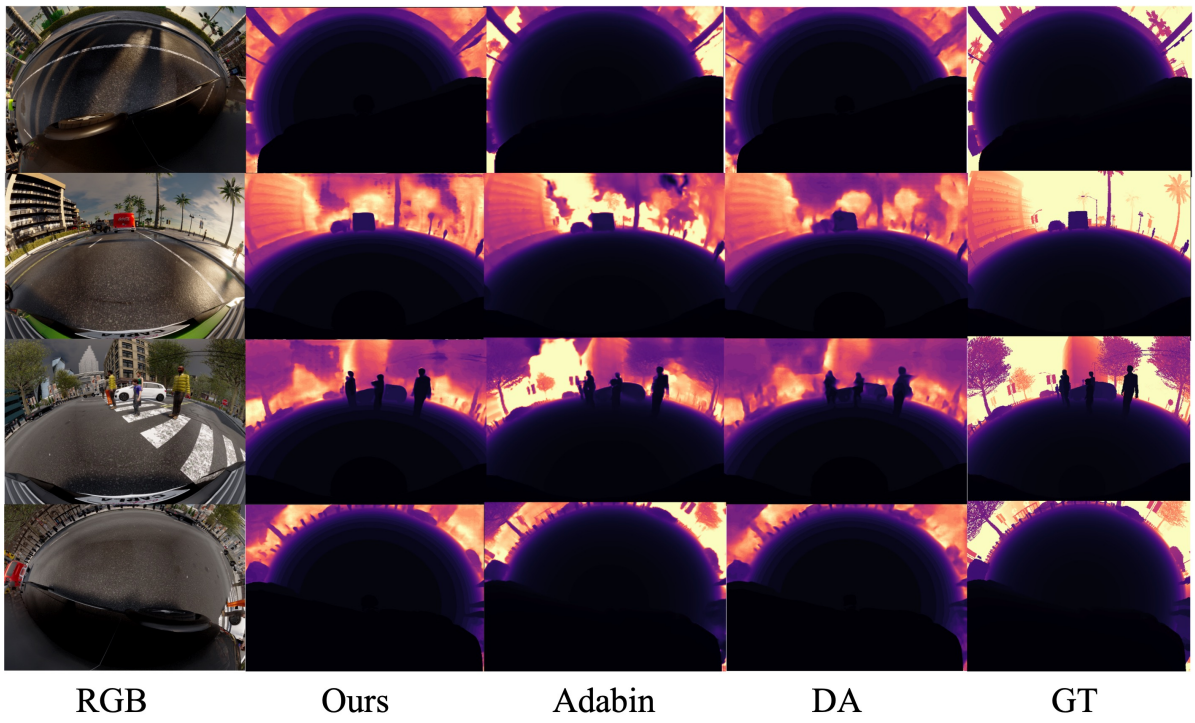


Figure 3. Visualization comparison with other methods. The first row is input RGB images, and the last row is the ground truth metric depth.The middle three rows represent a comparison between our method and other approaches. "DA" refers to Depth Anything, which involves full fine-tuning.

### 4.2. Experiments Setup

Our proposed method is implemented in PyTorch, utilizing the Adam optimizer with a learning rate of $1 \times 10^{-5}$ and a multistep learning rate schedule. The model is trained for 50,000 iterations. All experiments were conducted on an RTX A6000 GPU. Following the setup of Depth Anything, all input images are resized to $518 \times 518$. The network produces output depth maps of the same size, which are then rescaled to their original resolution using bilinear interpolation. For SynWoodScape, the original resolution is $966 \times 1280$, and for KITTI360, it is $1400 \times 1400$. Finally, disparity values are inverted to obtain the final depth output.

**Evaluation metrics.** In our experiments, we evaluate the performance of the depth estimation model using several common metrics: Abs Rel, RMSE, SiLog, and $\delta_1$, $\delta_2$, $\delta_3$. The Absolute Relative Difference (Abs Rel) is defined as:

$$\text{Abs Rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|d_i - d_i^*|}{d_i^*}, \tag{11}$$

where $d_i$ is the predicted depth, $d_i^*$ is the ground truth, and $N$ is the number of pixels. RMSE (Root Mean Squared Error) is given by:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - d_i^*)^2}. \tag{12}$$

SiLog (Scale-Invariant Logarithmic Error) measures the scale-invariant error between predicted and ground truth depths, defined as:

$$\text{SiLog} = \frac{1}{N}\sum_{i=1}^{N}(\log d_i - \log d_i^*)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N}\log d_i - \log d_i^*\right)^2. \tag{13}$$

Lastly, $\delta_1$, $\delta_2$, $\delta_3$ represent the percentage of predicted pixels where the ratio between the predicted and ground truth depth satisfies $\max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) < \delta$, where $\delta_1 = 1.25$, $\delta_2 = 1.25^2$ and $\delta_3 = 1.25^3$.

**Loss function.** During the training phase, the model is trained using a combination of scale-invariant log loss and L1 loss. $\mathcal{L}_{total} = \lambda_{si}\mathcal{L}_{silog} + \mathcal{L}_1$, where $\lambda_{si} = 10$.

### 4.3. Comparison

**Quantitative comparison.** We need to compare the proposed method through experimental results to demonstrate

its effectiveness. The results are presented in Tab. 2. First, we selected some open-source supervised[1, 22] and self-supervised methods to train directly on the fisheye dataset, which we refer to as domain-specific methods. For a fair comparison, we added a supervised loss function to OmniDet[18]. We then used a fine-tuning-based approach for further comparison. Specifically, we compare full fine-tuning, LoRA fine-tuning[9], and our proposed fine-tuning method.

Last five rows in the table show results from fine-tuning the Depth Anything pre-trained model. "ViT-S" and "ViT-L" represent the size of the backbone selected for the model. The "Full" in parentheses refers to full parameter fine-tuning. It can be observed that full parameter fine-tuning leads to significant forgetting, making it difficult to transfer from pinhole data to fisheye data. The final metrics are worse than those of AdaBins trained from scratch. The third row shows the results of fine-tuning the model's backbone using LoRA. Here, LoRA is set with $r = 16$, fine-tuning the attention and linear layers in the backbone while freezing all biases.

The last two rows of the table show the results of our proposed method. With the DAA fine-tuning approach and SFH global scale recovery, our method outperforms both AdaBins and LoRA fine-tuning. It effectively avoids the forgetting problem while leveraging the prior knowledge from pinhole data, achieving efficient fine-tuning.

**Qualitative comparison.** Fig. 3 present the visual comparison between different methods. Compared to the fully fine-tuned Depth Anything model, our approach demonstrates greater advantages in metric depth estimation tasks. Full fine-tuning can cause the Depth Anything model's performance to degrade on fisheye data. In contrast, our method effectively avoids this degradation while accurately capturing fisheye distortion patterns, allowing for better depth perception of objects with significant distortion near the edges. For example, in the second row of the figure, the building on the right side of the image has obvious distortion, and Adabins' prediction shows jagged edges. In the third row, the edges of the subject in the center of the image are also clearer.

### 4.4. Ablation Study

**Ablation study on different component.** To validate the effectiveness of the proposed method, we conducted ablation experiments on different components of the network, analyzing the performance gains each part contributes. From Tab. 3, we can conclude that using the Distortion-Aware Adapter (DAA) results in an AbsRel improvement of 0.0187 compared to the baseline. The RMSE also improves by 0.31 meters. Additionally, incorporating the ScaleFormer Head (SFH) provides further gains, especially in the RMSE metric, as the inclusion of global scale information is particularly beneficial for improving this metric.

**Ablation study on parameter.** To demonstrate the efficiency of our method, Tab.4 provides a comparison of the parameter count across different layers of the network. We offer both small and large versions of the model. The parameter count for the DAA module is 1.01M and 5.17M for the small and large versions, respectively, accounting for only 4.5% and 1.6% of the total ViT backbone parameters. The lightweight SFH has a parameter count of approximately 0.1M, which is negligible compared to the other components. In the end, the parts we fine-tune include the DAA, DPT head, and SFH. For the small and large versions, the fine-tuned parameters account for 14.6% and 10.5% of the total parameters, respectively. We also compare our proposed method with previous methods in terms of the number of intermediate references. From the Tab5, we observe that our method's small version has the second highest number of parameters, but its performance surpasses that of other methods. Notably, the number of parameters we need to train is significantly smaller than those required by the other methods.

## Limitation

This work primarily focuses on adapting models from pinhole cameras to fisheye cameras and from relative depth to absolute depth, which may lead to a loss of the original generalization capability across indoor and outdoor scenes in the base model. Additionally, the proposed method introduces a small amount of extra parameters, which could impact the model's inference latency.

## 5. Conclusion

In this work, we propose an efficient method for fine-tuning a relative depth estimation model to an absolute depth estimation model for fisheye cameras. Our method introduces the Distortion-Aware Adapter (DAA) to model the dense fisheye distortion encoding, preventing catastrophic forgetting in the network. Additionally, we propose the ScaleFormer Head (SFH) to predict global scale information, enabling the model to recover absolute depth from relative depth. Comprehensive experimental results validate the effectiveness and efficiency of our approach. We hope this work inspires further research into fisheye absolute depth estimation and parameter efficient fine-tuning.

## Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (62302167, U23A20343, 62222602, 62176092, and 62476090); in part by Shanghai Sailing Program (23YF1410500); Natural Science Foundation of Shanghai (23ZR1420400); in part by Chenguang Program of Shanghai Education Development Foun-

dation and Shanghai Municipal Education Commission (23CGA34), in part by CCF-Tencent RAGR20240122.

## References


[1] S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021. 2, 5, 6, 8

[2] R. Birkl, D. Wofk, and M. Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation, 2023. 1

[3] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022. 2, 3

[4] 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)*, pages 6087–6093. IEEE, 2019. 3

[5] K. M. Dawson-Howe and D. Vernon. Simple pinhole camera calibration. *International Journal of Imaging Systems and Technology*, 5(1):1–6, 1994. 3

[6] R. Griffiths and D. G. Dansereau. Adapting cnns for fisheye cameras without retraining. *arXiv preprint arXiv:2404.08187*, 2024. 3

[7] J. Gu, S. Chen, Z. Wang, Y. Zhang, and P. Gong. Sara: Singular-value based adaptive low-rank adaption. *arXiv preprint arXiv:2408.03290*, 2024. 3

[8] Z. Han, C. Gao, J. Liu, S. Q. Zhang, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024. 3

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3, 8

[10] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024. 2

[11] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 3

[12] J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1335–1340, 2006. 3

[13] 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)*, pages 574–581. IEEE, 2020. 3

[14] V. R. Kumar, M. Klingner, S. Yogamani, M. Bach, S. Milz, T. Fingscheidt, and P. Mäder. Svdistnet: Self-supervised near-field distance estimation on surround view fisheye cameras. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10252–10261, 2021. 3

[15] 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*, pages 61–71, 2021. 3

[16] V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech. Near-field depth estimation using monocular fisheye camera: A semi-supervised learning approach using sparse lidar data. In *CVPR Workshop*, volume 7, page 2, 2018. 3

[17] V. R. Kumar, S. Yogamani, M. Bach, C. Witt, S. Milz, and P. Mäder. Unrectdepthnet: Self-supervised monocular depth estimation using a generic framework for handling common camera distortion models. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8177–8183. IEEE, 2020. 3

[18] V. R. Kumar, S. Yogamani, H. Rashed, G. Sitsu, C. Witt, I. Leang, S. Milz, and P. Mäder. Omnidet: Surround view cameras based multi-task visual perception network for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):2830–2837, 2021. 5, 6, 8

[19] J. N. Kundu, P. K. Uppala, A. Pahuja, and R. V. Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2656–2665, 2018. 2

[20] J. Lee, G. Cho, J. Park, K. Kim, S. Lee, J.-H. Kim, S.-G. Jeong, and K. Joo. Slabins: Fisheye depth estimation using slanted bins on road environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8765–8774, 2023. 2

[21] J. Lee, D. Park, D. Lee, and D. Ji. Semi-supervised 360 depth estimation from multiple fisheye cameras with pixel-level selective loss. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2290–2294. IEEE, 2022. 3

[22] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 5, 6, 8

[23] J. Li, Q. Nie, W. Fu, Y. Lin, G. Tao, Y. Liu, and C. Wang. Lors: Low-rank residual structure for parameter-efficient network stacking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15866–15876, 2024. 3

[24] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 6

[25] T. Liu, X. Liu, L. Shi, Z. Xu, S. Huang, Y. Xin, and Q. Yin. Sparse-tuning: Adapting vision transformers with efficient fine-tuning and inference. *arXiv preprint arXiv:2405.14700*, 2024. 3

[26] M. Maximov, K. Galim, and L. Leal-Taixé. Focus on defocus: bridging the synthetic to real domain gap for depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1071–1080, 2020. 2

[27] V. Patil, C. Sakaridis, A. Liniger, and L. Van Gool. P3depth: Monocular depth estimation with a piecewise planarity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1610–1621, 2022. 2

[28] E. Plaut, E. Ben Yaacov, and B. El Shlomo. 3d object detection from a single fisheye image without a single fisheye training image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2021. 3

[29] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 2

[30] M. Rey-Area, M. Yuan, and C. Richardt. 360monodepth: High-resolution 360deg monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3762–3772, 2022. 2

[31] A. R. Sekkat, Y. Dupuis, V. R. Kumar, H. Rashed, S. Yogamani, P. Vasseur, and P. Honeine. Synwoodscape: Synthetic surround-view fisheye camera dataset for autonomous driving. *IEEE Robotics and Automation Letters*, 7(3):8502–8509, 2022. 6

[32] E. Son, J. Choi, J. Song, Y. Jin, and S. J. Lee. Monocular depth estimation from a fisheye camera based on knowledge distillation. *Sensors*, 23(24):9866, 2023. 3

[33] F. Tosi, P. Z. Ramirez, and M. Poggi. Diffusion models for monocular depth estimation: Overcoming challenging conditions. *arXiv preprint arXiv:2407.16698*, 2024. 2

[34] Z. Wei, L. Chen, Y. Jin, X. Ma, T. Liu, P. Ling, B. Wang, H. Chen, and J. Zheng. Stronger fewer & superior: Harnessing vision foundation models for domain generalized semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28619–28630, 2024. 3

[35] Y. Xin, S. Luo, H. Zhou, J. Du, X. Liu, Y. Fan, Q. Li, and Y. Du. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv preprint arXiv:2402.02242*, 2024. 3

[36] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024. 1, 2

[37] W. Yin, C. Zhang, H. Chen, Z. Cai, G. Yu, K. Wang, X. Chen, and C. Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023. 2

[38] E. B. Zaken, S. Ravfogel, and Y. Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021. 3

[39] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023. 2

[40] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2

[41] R. Zhu, C. Wang, Z. Song, L. Liu, T. Zhang, and Y. Zhang. Scaledepth: Decomposing metric depth estimation into scale prediction and relative depth estimation. *arXiv preprint arXiv:2407.08187*, 2024. 2

[42] A. Zonca, L. Singer, D. Lenz, M. Reinecke, C. Rosset, E. Hivon, and K. Gorski. healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in python. *Journal of Open Source Software*, 4(35):1298, 2019. 3, 4