EasyChair Preprint

№ 14171

# MonoViM: Enhancing Self-Supervised Monocular Depth Estimation via Mamba

Qiang Gao, Gang Peng, Zeyuan Chen and Bingchuan Yang



July 25, 2024

# MonoViM: Enhancing Self-supervised Monocular Depth Estimation via Mamba

Qiang Gao[1,2], Gang Peng[1,2(✉)], Zeyuan Chen[3], and Bingchuan Yang[1,2]

[1] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China
{gao_qiang,penggang,yangbingchu}@hust.edu.cn
[2] Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, Wuhan 430074, China
[3] Graduate School of Comprehensive Human Sciences, University of Tsukuba, Tsukuba 305-8550, Japan
chen.zeyuan.tkb_gu@u.tsukuba.ac.jp

**Abstract.** In recent years, self-supervised monocular depth estimation has been widely applied in fields such as autonomous driving and robotics. While Convolutional Neural Networks (CNNs) and Transformers are predominant in this area, they face challenges with efficiently handling long-term dependencies and reducing computational complexity. To address this problem, we propose MonoViM, the first model integrating the Mamba to enhance the efficiency of self-supervised monocular depth estimation. Inspired by recent advancements in State Space Models (SSM), MonoViM integrates the SSM-based Mamba architecture into its encoder stage and employs a 2D selective scanning mechanism. This ensures that each image block acquires contextual knowledge through a compressed hidden state while maintaining a larger receptive field and reducing computational complexity from quadratic to linear. Comprehensive evaluations on the KITTI dataset, with fine-tuning and zero-shot on Cityscapes and Make3D, show that MonoViM outperforms current CNN-based and Transformer-based methods, achieving state-of-the-art performance and excellent generalization. Additionally, MonoViM demonstrates stronger ability in inference speed and GPU utilization than Transformer-based methods, particularly with high-resolution inputs. The code is available at https://github.com/aifeixingdelv/MonoViM.

**Keywords:** Monocular Depth Estimation · Self-Supervised Learning · State Space Models.

## 1 Introduction

Depth perception serves as the foundation for numerous advanced computer vision applications such as autonomous driving, robotics, and augmented reality [29]. In recent years, significant advancements have been made in depth learning methods, enabling supervised monocular depth estimation algorithms [34] to infer depth autonomously from a single RGB image. However, these methods require a large collection of images annotated with depth labels for effective training, which is costly. Consequently, there has been a surge of interest in utilizing

vast amounts of unlabelled real-world data, driving research into self-supervised methods [7,37] that use monocular video as input. These methods assume static scenes and reframe the depth estimation task as a cross-view consistency problem, where the difference between the current frame and its neighboring frames' reprojections serves as the image reprojection loss function.

Current self-supervised depth estimation networks predominantly utilize either CNN [16] architectures or Transformer [30] architectures. While CNNs are efficient and excel at capturing local features, their limited receptive field hampers understanding global context. On the other hand, Transformers, with their self-attention mechanism, can capture long-range dependencies and global context but suffer from high computational complexity and slow inference speeds, particularly when processing high-resolution images. These limitations impede real-time application of self-supervised depth estimation models and deployment on resource-constrained devices. Our research aims to design an efficient and robust self-supervised depth estimation model utilizing strengths of both architectures while mitigating respective weaknesses to address these challenges.

In this paper, we propose MonoViM, a novel model designed to enhance the efficiency of monocular self-supervised depth estimation algorithms. MonoViM incorporates the Mamba [12] module, inspired by recent advancements in efficient long-sequence modeling within the state space model (SSM) domain [11]. The visual Mamba [21,38] module significantly improves the visual backbone network's capability to handle long sequences by achieving efficient visual representation while maintaining linear computational complexity through the SSM and 2D selective scanning mechanism. By integrating the visual Mamba module, we enhance the overall accuracy and efficiency of the self-supervised monocular depth estimation model. Additionally, the visual Mamba module extends MonoViM's ability to process higher resolution input images.

The contributions of this study can be summarized as follows:

- We propose MonoViM, the first self-supervised monocular depth estimation model based on the SSM architecture. By integrating the SSM-based Mamba module, our model ensures a global receptive field while avoiding the quadratic computational complexity associated with Transformer models, thus improving real-time performance even achieving better depth estimation accuracy. Additionally, our model extends the capability to handle high-resolution input images.
- We conduct extensive evaluations of the MonoViM model on the KITTI dataset. Experimental results demonstrate that our model surpasses existing self-supervised monocular depth estimation methods based on CNNs and Transformers in terms of accuracy, achieving state-of-the-art performance. Furthermore, to validate the generalization capability of the model, we tested its fine-tuning and zero-shot transfer performance on the Cityscape and Make3D datasets.
- We also evaluate the inference speed and GPU utilization across various image resolutions between different architecture models. The results show that our model significantly outperforms Transformer-based methods, demonstrating its practicality and ability to handle high-resolution images.

## 2 Related Work

### 2.1 Self-supervised monocular depth estimation

Estimating depth from a single image is inherently complex and ambiguous. Despite significant advances from fully supervised deep learning techniques [23], obtaining precise depth labels in real-world scenarios remains difficult. Self-supervised methods offer a promising alternative by eliminating the need for hard-to-acquire ground truth annotations and reducing training costs. Building on Zhou et al.'s work [37], which made significant strides through joint optimization of depth and pose networks using image reconstruction loss, the field has advanced notably. The choice of backbone network architecture, whether CNN-based or Transformer-based, is critical to performance in monocular depth estimation.

In the realm of CNN-based structures, Gordon et al. [8] introduced two CNN-based Unet networks for single-image depth estimation, predicting ego-motion, object motion, and camera intrinsics, incorporating an occlusion-aware consistency loss. Lyu et al. [22] developed HR-Depth, an improved Unet for high-resolution monocular depth estimation, enhancing skip connections and incorporating a feature fusion Squeeze-and-Excitation module. Zhou et al. [36] presented DIFFNet, an encoder-decoder network utilizing HRNet to effectively utilize semantic information during downsampling and upsampling.

In the realm of Transformer-based structures, Han et al.'s TransDSSL [14] combine Transformers with self-supervised learning, introducing pixel-wise skip attention and self-distillation loss for improved detail and stability. Karpov et al.'s VTDepth [15] uses the Pyramid Vision Transformer for augmented reality applications. Zhao et al.'s MonoViT [35] blend CNN and Vision Transformer Hybrid architectures, achieving both local and global reasoning.

### 2.2 State Space Models (SSM) Architecture

Vision Transformer (ViT) architectures have been widely adopted in visual tasks, but the quadratic computational complexity of Transformers can impact performance when handling high-resolution images or long input sequences. Recently, models based on structured State Space Models [10] have emerged as compelling alternatives to Transformers for managing long-range dependencies. For improved practical feasibility, the S4 [11] model further proposed normalizing the parameter matrix to a diagonal structure. Subsequently, various structured SSM models were developed with different architectural enhancements. Notably, the Mamba [12] architecture, incorporating selection mechanisms, combines significant advantages of both CNN and Transformer backbones, making it a highly promising model. In computer vision, Mamba-based backbone networks [21,38] have been introduced and can be broadly categorized as Pure Mamba or Hybrid Mamba. Many of these models have demonstrated effectiveness in downstream tasks like medical image segmentation, 3D reconstruction, and 3D object detection.

## 3 Method

In this section, we first introduce the algorithmic framework for self-supervised monocular depth estimation. Next, we provide a detailed description of the proposed MonoViM DepthNet and PoseNet. Finally, we present the loss functions used for the self-supervised training of the depth estimation framework.

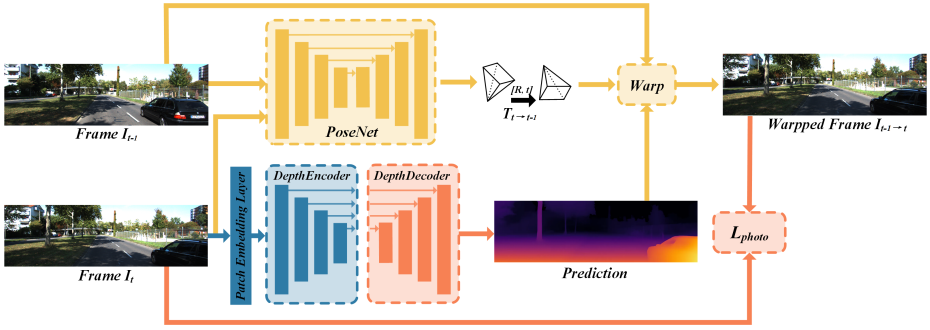


**Fig. 1.** Self-supervised monocular depth estimation algorithmic framework. The blue and orange blocks make up **DepthNet** and the yellow block represents **PoseNet**.

### 3.1 Self-Supervised Monocular Depth Estimation Framework

The self-supervised monocular depth estimation algorithm framework is shown in Fig. 1. Following the method proposed in [37], we use adjacent frames of keyframe $I_{t\pm1}$ to train the self-supervised depth network. This algorithm utilizes a depth estimation network to obtain the predicted depth information of the keyframe $I_t$ and a pose estimation network to estimate the 6-DOF pose changes between the keyframe $I_t$ and reference frame $I_{t\pm1}$. Subsequently, under the static scene assumption, the algorithm synthesizes the scene from the keyframe's perspective using the predicted depth map of the keyframe, the relative pose transformation of the reference frame, and the input image of the reference frame. Compared to many multi-frame self-supervised models, we only use the adjacent frames of the keyframe as reference frames. Based on structure-from-motion (SfM), the projection formula mapping each pixel of the keyframe $I_t$ to the previous and next frames is as follows:

$$p_{t\pm1} \sim KT_{t\to t\pm1}D_t(p_t)K^{-1}p_t \tag{1}$$

$$I_{t\pm1\to t}[p_t] = I_{t\pm1}\langle p_{t\pm1}\rangle \tag{2}$$

where <> represents the sampling operation, $K$ represents the camera intrinsics matrix, $p_{t\pm1}$ represents the pixel in the $I_{t\pm1}$ image, and $D_t(p_t)$ and $T$ respectively represent the predictions from the depth network and the pose network. Based on Equa.(1) and (2), we can obtain the synthesized keyframe $I_{t\pm1\to t}$ from the sampled $I_{t\pm1}$ by using the sampling operation.

At training time, both the DepthNet and PoseNet are optimized jointly by minimizing the photometric reprojection error $L_p$ (see Equa.(3)). More details on photometric error $pe$ can be found in Section 3.4.

$$L_p = \min_{t'} pe\left(I_t, I_{t\pm1\to t}\right) \tag{3}$$

### 3.2 MonoViM DepthNet

**MonoViM Overall Architecture.** MonoViM DepthNet details are shown as Fig. 2. During downsampling in the model, the input image (with dimensions $H \times W$) first passes through the conv-stem layer, transforming it into an embedding feature with dimensions $4H \times 4W$. Due to Mamba being more suitable for long sequence inputs [32], we apply a long sequence train strategy in which the feature size is set to $2H \times 2W$ for better model accuracy at relatively low input resolutions. The channel number changes from 3 to $C$ (typically set to 96) at this point. Subsequently, the embedding features are fed into the encoder to further extract features with larger local receptive fields and spatial dimensions.

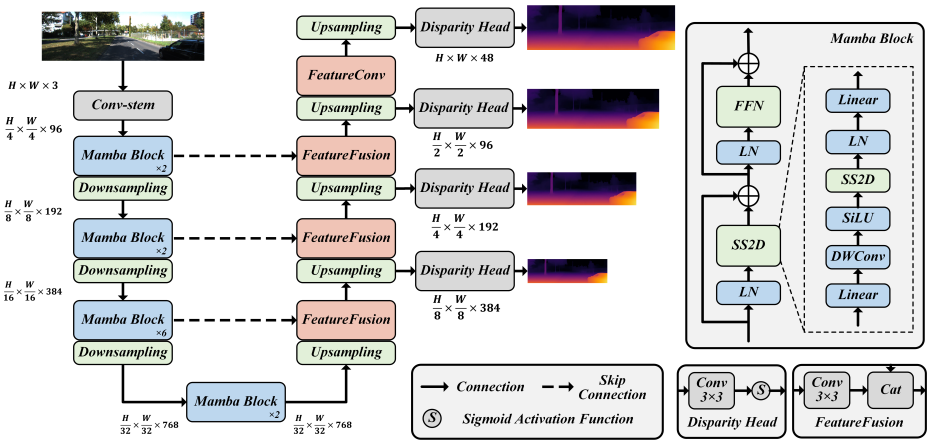


**Fig. 2. Details of MonoViM DepthNet. Left:** the U-Net architecture, starting with an input image and processing it through Conv-stem, Mamba Blocks, downsampling, and upsampling stages. **Right:** the details of three key modules: Disparity Head, Feature Fusion, and Mamba Block.

The encoder is divided into four stages, each consisting of multiple visual Mamba blocks and a downsampling block (except for the lowest stage, which lacks a downsampling block). The number of visual Mamba blocks stacked per stage is [2, 2, 6, 2], with each stage having an input feature channel number of [$C$, $2C$, $4C$, $8C$]. Since the lowest stage lacks a downsampling block, its output feature channel number remains $8C$. The pyramid features obtained through the encoder are denoted as $x_1$, $x_2$, $x_3$, and $x_4$, with dimensions $H/8 \times W/8 \times 2C$, $H/16 \times W/16 \times 4C$, $H/32 \times W/32 \times 8C$, and $H/32 \times W/32 \times 8C$ respectively.

In the upsampling stage of the model, the final output feature $x_4$ from the encoder is fed into the lowest layer of the decoder. Similarly to the encoder, the decoder is divided into four stages, each consisting of feature fusion modules and upsampling modules. To simplify the structure of the model and enhance its scalability, the decoder uses the most simple convolution operations. The feature fusion modules are designed to receive skip-connected features and fuse them with the upsampled intermediate features. The number of input feature channels for each stage of the decoder is [$8C$, $4C$, $2C$, $C$].

Additionally, to obtain multi-scale depth predictions for supervision, the decoder includes four disparity heads, each composed of a $3 \times 3$ convolution and a Sigmoid activation function. These heads generate four different-sized single-channel depth prediction maps from the intermediate and final features of the four decoder stages, which are used for model training.

The skip connections in the model utilize the most simple addition operations, avoiding redundant additional parameters and reducing computational overhead, thereby improving inference speed.

**State Space Models Architecture.** State space models (SSM) originate from the Kalman filter and provide a method for describing dynamic systems. They use state equations and observation equations to represent the relationship between system state evolution and outputs. Continuous-time SSMs can be represented as linear ordinary differential equations, as shown in Equa.(4). The state equation describes the temporal evolution of the system state, while the observation equation describes the relationship between system output and state.

$$\begin{aligned} \mathbf{h}'(t) &= \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{u}(t). \end{aligned} \tag{4}$$

Subsequently, S4 [11] and S6 (Mamba) [12] discretize continuous-time SSMs using the zero-order hold method, as shown in Equa.(5). In this Equation, $\Delta$ represents the time scale parameter, while $\overline{A}$ and $\overline{B}$ denote the discretized weight parameters.

$$\begin{aligned} \overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}), \\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}, \\ h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{5}$$

Meanwhile, to effectively address the long-range dependency problem in sequence modeling within limited storage space, the Mamba model employs the HiPPO [9] function to approximate the optimal solution for generating the state matrix A. HiPPO, as a general framework, projects continuous signals and discrete-time sequences onto a polynomial basis, achieving online data compression (as shown in Equa.(6)).

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{everything below the diagonal} \\ n+1 & \text{the diagonal} \\ 0 & \text{everything above the diagonal} \end{cases} \tag{6}$$

**Visual Mamba Block.** Although the Mamba module's scanning order aligns with the temporal sequence of processing natural language data, in visual processing the model must handle spatial features of images. Therefore, as in other studies [21], we incorporate 2D selective scanning into the Mamba module, forming the SS2D block. The scanning algorithm captures key spatial information by scanning in four directions, improving the global receptive field of the visual Mamba module (shown in Fig. 3). Moreover, this algorithm can dynamically adjust scanning direction and region based on application scenarios for efficient resource allocation.

Building on the SS2D module, we integrate a feedforward neural network (FFN), layer normalization (LN), depthwise separable convolution (DWConv), activation function (SiLU), and residual connections to construct the visual Mamba block, as illustrated in Fig. 2. When the feature map enters the Mamba block, it first passes through a normalization layer for standardization. The normalized data are then fed into the SS2D module to extract feature information. Concurrently, residual connections allow the data to bypass the SS2D layer and combine with the SS2D output features before being input into the FFN. This mechanism alleviates the problem of vanishing gradients and improves training speed and performance. Finally, the visual Mamba block sums the output features of the FFN with the results of the residual connection, producing the final output of the module.

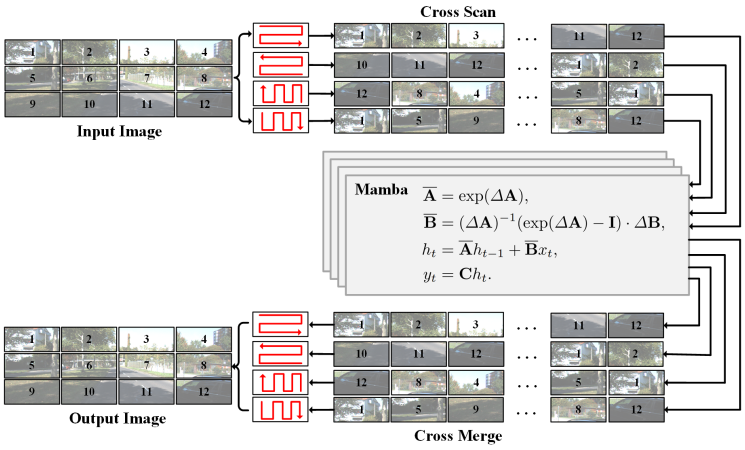


**Fig. 3. 2D Selective Scan Mechanism.** Input patches are traversed along four distinct scanning paths (Cross Scan), with each sequence independently processed by different Mamba blocks. The results are then combined to form a 2D feature map as the final output (Cross Merge).

### 3.3 Pose Estimation Network

Our pose estimation network employs the most simple, yet efficient implementation. Specifically, the PoseNet utilizes a lightweight ResNet18 architecture. It takes concatenated image pairs as input and outputs the 6-DoF relative pose between adjacent frames in an image sequence.

### 3.4 Self-Supervised Learning Loss Function

**Photometric Loss.** Following [7], we use a combination of the Structural Similarity (SSIM) loss $L_{ssim}$ and the photometric loss $L_1$ as our photometric loss function $L_{pe}$, as shown in Equa.(3) and (7). Here, $\alpha$ is a hyperparameter used to balance the photometric and structural losses.

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1 - \alpha)\|I_a - I_b\|_1 \tag{7}$$

**Gradient-Aware Smoothing Loss.** To regularize depth in textureless regions and smooth object edges, we incorporate a gradient-aware smoothness loss $L_s$ into the loss function, as shown in Equa.(8).

$$L_s = |\partial_x d_t^*|e^{-|\partial_x I_t|} + |\partial_y d_t^*|e^{-|\partial_y I_t|} \tag{8}$$

**Motion Object Masking Strategies.** In self-supervised monocular training, it is typically assumed that the camera is moving and the scene is static. However, if the camera is stationary or there are moving objects in the scene, performance may degrade significantly due to violation of this static assumption. During testing, this manifests as holes of infinite depth in the predicted depth map in regions with moving objects. Many studies [27] attempt to use instance segmentation models to construct masks of moving objects, but these methods consume substantial hardware resources during training, and false detections by the segmentation model can interfere with training. To keep the scalability of the model, we adopt the most simple yet effective automasking strategy [7] that filters out relatively stationary pixels and low-texture regions. This strategy introduces a binary mask $\mu$, representing pixel regions where the reprojection error is lower than the original unwarped source image error, as described by Equa.(9).

$$\mu = \left[\min_{t'} pe(I_t, I_{t\pm1\to t}) < \min_{t'} pe(I_t, I_{t\pm1})\right] \tag{9}$$

**Total Training Loss Function.** Based on the aforementioned moving object masking strategy, we calculate the photometric loss and gradient-aware smoothness loss between the keyframe input image $I_t$ and the reconstructed image $I_{t\pm1\to t}$ as the final supervised learning signal, as shown in Equa.(10). $\lambda$ is a hyperparameter used to balance the photometric and smoothing losses.

$$L = \mu L_p + \lambda L_s \tag{10}$$

## 4 Experiments

In this section, we report the experiment results of MonoViM across multiple datasets. Our findings clearly demonstrate that MonoViM not only achieves superior depth estimation accuracy but also exhibits enhanced applicability compared with different architecture models.

### 4.1 Implementation Details

In our experiment, we implement the MonoViM model using PyTorch. During training, we conduct 30 epochs on the KITTI dataset with a batch size of 12. We choose AdamW as the optimizer, setting the initial learning rates for PoseNet and Depth decoder to 1e-4, while the initial learning rate for SSM-based Depth encoder is set to 5e-5. The SSM-based Depth encoder contain 2, 2, 6, and 2 sequential Mamba blocks in its first to fourth stages, respectively. Both PoseNet and Depth encoder are pretrained on ImageNet [3]. Our low-resolution ($640 \times 192$) experiments are conducted on a single RTX 4090 GPU, whereas high-resolution ($1024 \times 320$ and $1280 \times 384$) experiments are performed on two RTX 4090 GPUs. Overall, network training takes approximately 20 hours. During the experiments, we use the same data augmentation methods detailed in [7,22].

### 4.2 Datasets and Evaluation Metrics

**KITTI[6].** The KITTI dataset provides stereo image sequences, commonly used for self-supervised monocular depth estimation. It comprises 61 scenes with typical image resolutions of $1242 \times 375$. Depth ground truth is obtained using LiDAR sensors mounted on a moving vehicle. We adopt the image split method by Eigen et al [4], including 39,810 monocular triplets for training and 4,424 for validation. To compare with existing solutions, we evaluate single-view depth performance on the original LiDAR test split (697 images) [4]. In our experiments, we train MonoViM on KITTI with minimal requirements, not using motion masks (only employing automasking [7]) and auxiliary information. To test performance, we retain the challenging setting of using only single-frame images as input, while other methods may use multi-frame images to enhance accuracy.

**Cityscapes[2].** The Cityscapes dataset is a challenging dataset with many moving objects. To evaluate our model's generalization, we fine-tune it on the Cityscapes dataset, starting from a model pretrained on KITTI. Notably, we don't use motion masks in our experiments, whereas many models did. For fairness and comparability, we use the data preprocessing scripts from [37], converting image sequences into triplets.

**Make3D[24].** The Make3D dataset contains outdoor environments and is commonly used to test the generalization performance of monocular depth frameworks. We conduct a zero-shot evaluation of the MonoViM model trained on the KITTI dataset, using the same image preprocessing steps and computing evaluation metrics as detailed in [7].

**Evaluation Metrics.** For evaluation, we follow the standard metrics proposed by Eigen et al [4]. It is important to note that for error metrics such as AbsRel, SqrRel, RMSE, and RMSE log, lower values indicate better performance. Conversely, for recall metrics like $\delta_{1.25}$ (i.e., $\delta < 1.25$), higher values indicate better performance.

### 4.3 Depth Estimation Performance

In this section, we verify the accuracy of the MonoViM model depth estimation by conducting precision tests on common architecture depth estimation models

**Table 1. Performance comparison on KITTI[6] eigen benchmark. Top:** 640 × 192 Resolution. **Middle:** 1024×320 Resolution. **Bottom:** 1280×384 Resolution. In the Data column, **S**: trained with synchronized stereo pairs, **M**: trained with monocular videos, **MS**: trained with monocular videos and stereo pairs. † stands for utilizing long sequence training strategy. ‡ stands for the novel results from the official Github repository, better than published ones. The best results are in **bold**, and the second best are underlined. All self-supervised methods use median-scaling to estimate the absolute depth scale.

| Method | Data | Resolution | lower is better | | | | higher is better | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | AbsRel | SqRel | RMSE | RMSElog | $\delta_1$ | $\delta_2$ | $\delta_3$ |
| Monodepth2 [7] | MS | 640×192 | 0.106 | 0.818 | 4.750 | 0.196 | 0.874 | 0.957 | 0.979 |
| HR-Depth [22] | MS | 640×192 | 0.107 | 0.785 | 4.612 | 0.185 | 0.887 | 0.962 | 0.982 |
| CADepth [31] | MS | 640×192 | 0.102 | 0.752 | 4.504 | 0.181 | 0.894 | 0.964 | 0.983 |
| DIFFNet‡ [36] | MS | 640×192 | 0.101 | 0.749 | 4.445 | 0.179 | 0.898 | 0.965 | 0.983 |
| Lite-Mono [33] | M | 640×192 | 0.101 | 0.729 | 4.454 | 0.178 | 0.897 | 0.965 | 0.983 |
| VTDepth [15] | M | 640×192 | 0.105 | 0.762 | 4.530 | 0.182 | 0.893 | 0.964 | 0.983 |
| TransDSSL [14] | M | 640×192 | 0.102 | 0.753 | 4.461 | 0.177 | 0.896 | 0.966 | **0.984** |
| MonoViT [35] | M | 640×192 | 0.099 | 0.708 | 4.372 | 0.175 | 0.900 | 0.967 | **0.984** |
| SENSE [19] | M | 640×192 | 0.104 | 0.693 | 4.294 | 0.177 | 0.894 | 0.965 | **0.984** |
| SwinDepth [25] | M | 640×192 | 0.106 | 0.739 | 4.510 | 0.182 | 0.890 | 0.964 | **0.984** |
| Dynamo-Depth [28] | M | 640×192 | 0.112 | 0.758 | 4.505 | 0.183 | 0.873 | 0.959 | **0.984** |
| AQUANet [1] | M | 640×192 | 0.105 | **0.621** | **4.227** | 0.179 | 0.889 | 0.964 | **0.984** |
| **MonoViM(ours)** | M | 640×192 | 0.106 | 0.743 | 4.548 | 0.181 | 0.89 | 0.964 | 0.983 |
| **MonoViM†(ours)** | M | 640×192 | **0.097** | 0.672 | 4.275 | **0.173** | **0.908** | **0.968** | **0.984** |
| Monodepth2 [7] | MS | 1024×320 | 0.106 | 0.806 | 4.63 | 0.193 | 0.876 | 0.958 | 0.98 |
| FeatDepth [26] | MS | 1024×320 | 0.099 | 0.697 | 4.427 | 0.184 | 0.889 | 0.963 | 0.982 |
| HR-Depth [22] | MS | 1024×320 | 0.101 | 0.716 | 4.395 | 0.179 | 0.899 | 0.966 | 0.983 |
| DIFFNet‡ [36] | M | 1024×320 | 0.097 | 0.722 | 4.345 | 0.174 | 0.907 | 0.967 | 0.984 |
| CADepth [31] | MS | 1024×320 | 0.096 | 0.694 | 4.264 | 0.173 | 0.908 | **0.968** | 0.984 |
| MonoViT [35] | M | 1024×320 | 0.096 | 0.714 | 4.292 | 0.172 | 0.908 | **0.968** | 0.984 |
| Lite-Mono [33] | M | 1024×320 | 0.097 | 0.710 | 4.309 | 0.174 | 0.905 | 0.967 | 0.984 |
| SENSE [19] | M | 1024×320 | 0.099 | **0.617** | **4.079** | 0.172 | 0.902 | **0.968** | **0.985** |
| **MonoViM(ours)** | M | 1024×320 | 0.095 | 0.658 | 4.263 | **0.171** | 0.909 | **0.968** | 0.984 |
| **MonoViM(ours)** | MS | 1024×320 | **0.093** | 0.65 | 4.234 | **0.171** | **0.913** | **0.968** | 0.984 |
| PackNet [13] | M | 1280×384 | 0.104 | 0.758 | 4.386 | 0.182 | 0.895 | 0.964 | 0.982 |
| HR-Depth [22] | M | 1280×384 | 0.104 | 0.727 | 4.41 | 0.179 | 0.894 | 0.966 | 0.984 |
| CADepth-Net [31] | M | 1280×384 | 0.102 | 0.715 | 4.312 | 0.176 | 0.9 | 0.968 | 0.984 |
| MonoViT [35] | M | 1280×384 | **0.094** | 0.682 | 4.2 | **0.17** | **0.912** | **0.969** | 0.984 |
| **MonoViM(ours)** | M | 1280×384 | 0.096 | **0.633** | **4.163** | **0.17** | 0.906 | **0.969** | **0.985** |

using KITTI, CityScapes, and Make3D datasets. Experimental results demonstrate that MonoViM exhibits outstanding performance across all datasets, achieving the level of existing state-of-the-art models.

**KITTI Results.** We utilize the standard KITTI Eigen split [4] to evaluate our model, which incorporates 697 images and original LiDAR scans. In light of monocular scale ambiguity inherent in monocular depth models trained on video sequences, the estimated depth is scaled by the median ground truth for each image [37]. Table 1 compiles results from current state-of-the-art self-supervised algorithms, including training effects at three different input image resolutions: 640×192 (low), 1024×320 (high), and 1280×384 (higher). For completeness, we report results for methods trained using monocular (M) and stereo (MS) inputs. MonoViM achieves state-of-the-art performance in all metrics, regardless of training resolution and setting. Notably, compared to baseline models based on ResNet and Transformer architectures, such as Monodepth2 [7] (ResNet based), TransDSSL [14] (Swin Transformer based), VTDepth [15] (Pyramid Vi-

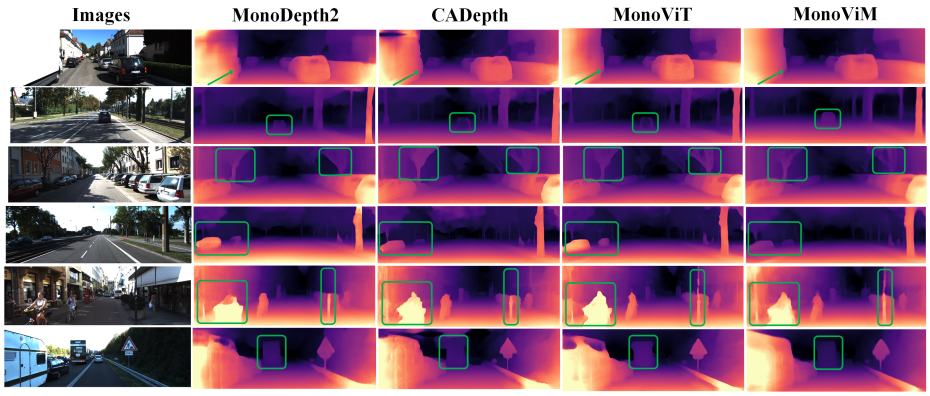


**Fig. 4. Qualitative results on KITTI.** First column is input images. Then, predictions by SoTA methods (MonoDepth2, CADepth, MonoViT) and MonoViM (Ours).

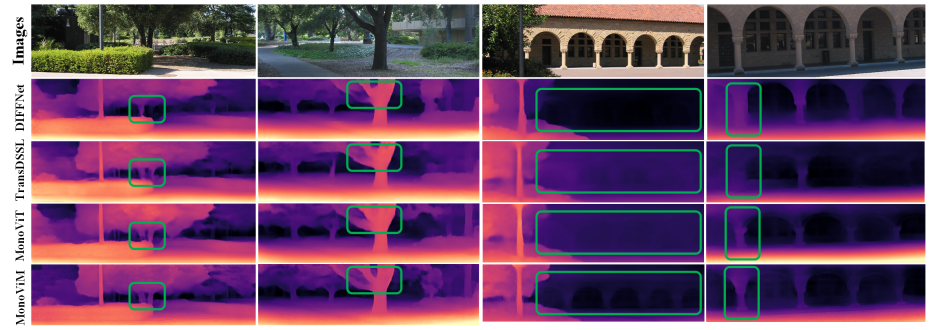


**Fig. 5. Qualitative results on Make3D.** Top row is input images. Then, predictions by SoTA methods (DIFFNet, TransDSSL, MonoViT) and MonoViM (Ours).

sion Transformer based), and MonoViT [35] (MPViT based), the Mamba based model achieves higher accuracy. This indicates Mamba is more suitable for depth estimation, which is a type of long-sequence prediction task.

Fig. 4. shows a qualitative comparison of predictions from MonoViM and other models. MonoViM achieves more complete, linear, continuous depth predictions for coplanar planes (e.g., truck containers), as seen in rows 1 and 6 of Fig. 4. It also avoids black hole situations when predicting vehicle depth, as in row 2. The KITTI dataset has many scenes with significant light variation from environmental lighting, affecting autonomous driving perception algorithms. In these high dynamic range scenes, our model provides more accurate, detailed depth predictions, as shown by the shaded vehicle in row 4 and the illuminated pedestrian in row 5.

**Cityscapes Results.** To evaluate the generalization capability of MonoViM, we fine-tune the KITTI pretrained model on the CityScapes dataset. The results are presented in Table 2. Because CityScapes contains many moving objects, significantly impacting model training, most of the models in Table 2 employ more effective motion masks to handle these objects. However, even with basic masking, MonoViM demonstrates superior performance.

**Table 2. Performance comparison on Cityscapes[2] dataset.** All the other baselines are trained from scratch on Cityscapes. **K** for KITTI, **C** for Cityscapes, **K→C** for pretrained on KITTI and fine-tuned on Cityscapes. **MMask** means using motion mask to deal with moving objects (extremely important for training on Cityscapes), – for no motion mask.

| Method | Train | Resolution | lower is better | | | | higher is better | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | AbsRel | SqRel | RMSE | RMSElog | $\delta_1$ | $\delta_2$ | $\delta_3$ |
| Monodepth2 [7] | -, C | 416×128 | 0.129 | 1.569 | 6.876 | 0.187 | 0.849 | 0.957 | 0.983 |
| Videos in the Wild [8] | MMask, C | 416×128 | 0.127 | 1.330 | 6.960 | 0.195 | 0.830 | 0.947 | 0.981 |
| Li et al. [20] | MMask, C | 416×128 | 0.119 | 1.29 | 6.98 | 0.19 | 0.846 | 0.952 | 0.982 |
| Lee et al. [17] | MMask, C | 832×256 | 0.116 | 1.213 | 6.695 | 0.186 | 0.852 | 0.951 | 0.982 |
| InstaDM [18] | MMask, C | 832×256 | **0.111** | 1.158 | **6.437** | 0.182 | 0.868 | 0.961 | 0.983 |
| **MonoViM(ours)** | -,K→C | 416×128 | 0.117 | **1.154** | 6.609 | **0.176** | **0.869** | **0.967** | **0.99** |

**Make3D Results.** To further evaluate the generalization capability of MonoViM, we perform zero-shot evaluation on the Make3D dataset [24] using weights pretrained on KITTI. Following the same evaluation setup as in [5], we test on center-cropped images with a 2×1 ratio. The results, as shown in Table 3 and Fig. 5, indicate that MonoViM outperforms the baseline models, producing sharper depth maps with more accurate scene details, particularly when handling structured buildings and tree trunks. This demonstrates the superior zero-shot generalization ability of our model.

**Table 3. Performance comparison on Make3D[24] dataset.**

| Method | Data | lower is better | | | |
|---|---|---|---|---|---|
| | | AbsRel | SqRel | RMSE | RMSElog |
| Monodepth2 [7] | M | 0.322 | 3.589 | 7.417 | 0.163 |
| CADepth [31] | M | 0.312 | 3.086 | 7.066 | 0.159 |
| HR-Depth [22] | M | 0.305 | 2.944 | 6.857 | 0.157 |
| DIFFNet [36] | M | 0.298 | 2.901 | 6.753 | 0.153 |
| TransDSSL [14] | M | 0.289 | 3.061 | 7.071 | 0.151 |
| MonoViT [35] | M | 0.286 | **2.758** | **6.623** | 0.147 |
| Lite-Mono [33] | M | 0.305 | 3.060 | 6.981 | 0.158 |
| **MonoViM(ours)** | M | **0.275** | 2.806 | 6.655 | **0.142** |

### 4.4 Inference Speed and Resource Utilization Analysis

In this section, we further verify MonoViM's practicality by comparing its inference speed and GPU utilization against baseline depth estimation models with common architectures. Experimental results (Fig. 6) show MonoViM excels in both speed and GPU usage, outperforming existing Transformer-based models [14,15,35]. Related experiments are conducted on a single RTX 4090 GPU.

**Inference Speed Analysis.** MonoViM demonstrates superior computational efficiency compared to existing models, as shown in Fig. 6 (Right). It outperforms previous Transformer-based models at low-resolution inputs and significantly surpasses them at high-resolution inputs. Specifically, at resolutions of $1280 \times 384$, $1536 \times 480$, and $2048 \times 640$, MonoViM's inference speed is approximately 14.35%, 84.82%, and 141% faster than VTDepth [15], and about 118.54%, 199.45%, and 215.87% faster than MonoViT [35]. These results indicate that MonoViM maintains efficient computational performance with high-resolution

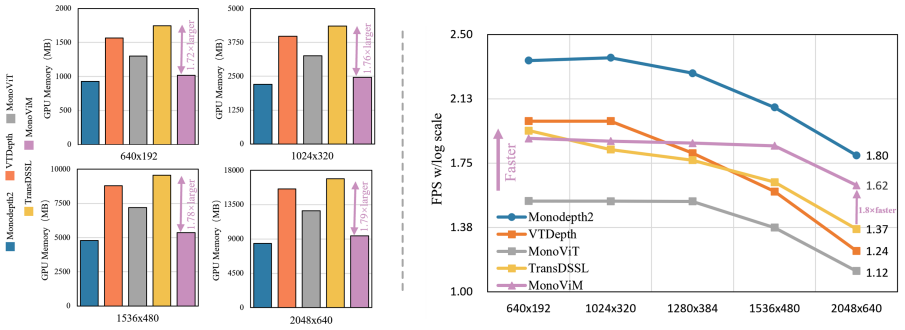


**Fig. 6. GPU Memory Usage and Inference Speed Comparison. Left:** shows the GPU memory usage comparison for five baseline models, Monodepth2(Res50, 34.57M), VTDepth(PVT-small, 28.85M), MonoViT(MPViT, 27.87M), TransDSSL(Swin-Tiny, 41.68M) and MonoViM(Mamba, 38.68M), across four resolutions: 640×192, 1024×320, 1536×480 and 2048×640. Each bar chart represents the GPU memory usage consumption (in MB) for each model at the given resolution. **Right:** shows the inference speed comparison for the same five models across the same resolutions. The line graph displays the frames per second (FPS) on a logarithmic scale, indicating how the inference speed of each model varies with different resolutions.

inputs, underscoring its practicality and effectiveness in real-world applications, particularly for high-resolution input scenarios.

**GPU Memory Usage Analysis.** The results, as shown in Fig. 6 (Left), demonstrate that MonoViM consistently uses less GPU memory usage than Transformer models across all resolutions, while being comparable to ResNet models, indicating high resource efficiency. At resolutions of 640×192, 1024×320, 1536 × 480, and 2048 × 640, MonoViM's GPU memory usage is approximately 41.74%, 43.36%, 44.02%, and 44.29% lower than that of TransDSSL [14], and about 35.01%, 38.05%, 39.12%, and 39.52% lower than VTDepth [15], respectively. Notably, as the input image resolution increases, MonoViM's advantage in GPU utilization becomes more pronounced, highlighting its superior practicality over Transformer-based models for high-resolution inputs.

### 4.5 Ablation Study

In our KITTI dataset experiments, we analyze the impact of SSM and ImageNet pretraining on MonoViM model accuracy, as shown in Table 4. We replace the SSM structure with convolutional and direct connection structures, finding SSM essential for higher accuracy. Comparing models with and without ImageNet pretrained weights, the results show that stronger pretrained weights significantly improve performance, highlighting their importance.

## 5 Conclusion

This paper proposed MonoViM, a novel model that significantly enhances the efficiency and accuracy of self-supervised monocular depth estimation algorithms

**Table 4. Ablation study for the core component of MonoViM. w/o** for without. **SSM→Conv** means using Conv to replace SSM.

| Method | Data | Resolution | lower is better | | | | higher is better | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | AbsRel | SqRel | RMSE | RMSElog | $\delta_1$ | $\delta_2$ | $\delta_3$ |
| **MonoViM** | M | 1024×320 | **0.095** | **0.658** | **4.26** | **0.171** | **0.909** | **0.968** | **0.984** |
| w/o SSM | M | 1024×320 | 0.116 | 0.803 | 4.680 | 0.194 | 0.869 | 0.959 | 0.981 |
| SSM→Conv | M | 1024×320 | 0.113 | 0.837 | 4.600 | 0.185 | 0.882 | 0.963 | 0.983 |
| w/o Pretrained | M | 1024×320 | 0.103 | 0.847 | 4.595 | 0.186 | 0.887 | 0.963 | 0.983 |

through the integration of the Mamba module. By incorporating the Mamba module with 2D selective scanning mechanism, MonoViM ensures efficient visual representation and linear computational complexity, extending the model's capability to process high-resolution images. Experimental results demonstrate that MonoViM achieves outstanding depth estimation accuracy across multiple datasets and outperforms Transformer architectures in terms of applicability. Additionally, MonoViM exhibits higher computational efficiency and GPU utilization when handling high-resolution inputs. In summary, MonoViM provides a more efficient, accurate, and practical solution for monocular self-supervised depth estimation.

## References


1. Bello, J.L.G.e.a.: Self-supervised monocular depth estimation with positional shift depth variance and adaptive disparity quantization. IEEE Transactions on Image Processing (2024)
2. Cordts, M.e.a.: The cityscapes dataset for semantic urban scene understanding. In: Proc. CVPR. pp. 3213–3223 (2016)
3. Deng, J.e.a.: Imagenet: A large-scale hierarchical image database. In: Proc. CVPR. pp. 248–255. IEEE (2009)
4. Eigen, D.e.a.: Depth map prediction from a single image using a multi-scale deep network. Advances in neural information processing systems **27** (2014)
5. Garg, R.e.a.: Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: Proc. ECCV. pp. 740–756. Springer (2016)
6. Geiger, A.e.a.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
7. Godard, C.e.a.: Digging into self-supervised monocular depth estimation. In: Proc. ICCV. pp. 3828–3838 (2019)
8. Gordon, A.e.a.: Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In: Proc. ICCV. pp. 8977–8986 (2019)
9. Gu, A.e.a.: Hippo: Recurrent memory with optimal polynomial projections. Advances in neural information processing systems **33**, 1474–1487 (2020)
10. Gu, A.e.a.: Combining recurrent, convolutional, and continuous-time models with linear state space layers. Advances in neural information processing systems (2021)
11. Gu, A.e.a.: Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396 (2021)
12. Gu, A.e.a.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
13. Guizilini, V.e.a.: 3d packing for self-supervised monocular depth estimation. In: Proc. CVPR. pp. 2485–2494 (2020)

14. Han, D.e.a.: Transdssl: Transformer based depth estimation via self-supervised learning. IEEE Robotics and Automation Letters **7**(4), 10969–10976 (2022)
15. Karpov, A.e.a.: Exploring efficiency of vision transformers for self-supervised monocular depth estimation. In: Proc. ISMAR. pp. 711–719. IEEE (2022)
16. LeCun, Y.e.a.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
17. Lee, S.e.a.: Attentive and contrastive learning for joint depth and motion field estimation. In: Proc. ICCV. pp. 4862–4871 (2021)
18. Lee, S.e.a.: Learning monocular depth in dynamic scenes via instance-aware projection consistency. In: Proc. AAAI. vol. 35, pp. 1863–1872 (2021)
19. Li, G.e.a.: Sense: Self-evolving learning for self-supervised monocular depth estimation. IEEE Transactions on Image Processing (2023)
20. Li, H.e.a.: Unsupervised monocular depth learning in dynamic scenes. In: Proc. CoRL. pp. 1908–1917. PMLR (2021)
21. Liu, Y.e.a.: Vmamba: Visual state space model (2024)
22. Lyu, X.e.a.: Hr-depth: High resolution self-supervised monocular depth estimation. In: Proc. AAAI. vol. 35, pp. 2294–2301 (2021)
23. Piccinelli, L.e.a.: idisc: Internal discretization for monocular depth estimation. In: Proc. CVPR. pp. 21477–21487 (2023)
24. Saxena, A.e.a.: Make3d: Learning 3d scene structure from a single still image. IEEE transactions on pattern analysis and machine intelligence **31**(5), 824–840 (2008)
25. Shim, D.e.a.: Swindepth: Unsupervised depth estimation using monocular sequences via swin transformer and densely cascaded network. In: Proc. ICRA (2023)
26. Shu, C.e.a.: Feature-metric loss for self-supervised learning of depth and egomotion. In: Proc. ECCV. pp. 572–588. Springer (2020)
27. Sun, L.e.a.: Sc-depthv3: Robust self-supervised monocular depth estimation for dynamic scenes. IEEE Trans. Pattern Anal. Mach. Intell. (2023)
28. Sun, Y.e.a.: Dynamo-depth: fixing unsupervised depth estimation for dynamical scenes. Advances in Neural Information Processing Systems **36** (2024)
29. Tang, Y.e.a.: Perception and navigation in autonomous systems in the era of learning: A survey. IEEE Transactions on Neural Networks and Learning Systems **34**(12), 9604–9624 (2022)
30. Vaswani, A.e.a.: Attention is all you need. Advances in neural information processing systems **30** (2017)
31. Yan, J.e.a.: Channel-wise attention-based network for self-supervised monocular depth estimation. In: Proc. 3DV. pp. 464–473. IEEE (2021)
32. Yu, W.e.a.: Mambaout: Do we really need mamba for vision? arXiv preprint arXiv:2405.07992 (2024)
33. Zhang, N.e.a.: Lite-mono: A lightweight cnn and transformer architecture for self-supervised monocular depth estimation. In: Proc. CVPR. pp. 18537–18546 (2023)
34. Zhao, C.e.a.: Monocular depth estimation based on deep learning: An overview. Science China Technological Sciences **63**(9), 1612–1627 (2020)
35. Zhao, C.e.a.: Monovit: Self-supervised monocular depth estimation with a vision transformer. In: Proc. 3DV. pp. 668–678. IEEE (2022)
36. Zhou, H.e.a.: Self-supervised monocular depth estimation with internal feature fusion. arXiv preprint arXiv:2110.09482 (2021)
37. Zhou, T.e.a.: Unsupervised learning of depth and ego-motion from video. In: Proc. CVPR. pp. 1851–1858 (2017)
38. Zhu, L.e.a.: Vision mamba: Efficient visual representation learning with bidirectional state space model. arXiv preprint arXiv:2401.09417 (2024)