Monocular Depth Estimation of Tomato in Greenhouse Environment Using the Tomato-MDE Model

Chao Qi, Yawen Cheng, Qian Wu*

Institute of Agricultural Information, Jiangsu Academy of Agricultural Sciences, Nanjing 210031, China

Abstract: Currently, greenhouse tomato picking robots highly rely on stereo vision cameras to obtain the depth information of the objects. However, due to the inherent flaws of the stereo vision camera, the accuracy of the obtained depth value is extremely susceptible to environmental variations, thus affecting the stability of the picking. In this context, we propose a lightweight monocular depth estimation model for tomatoes (Tomato-MDE) that combines real and virtual datasets, capable of adapting to greenhouse environments with transparent objects. In the encoder, it comprises a simple stacking of the proposed fast self-attention module (FSAM) and depthwise convolution (DWConv), which effectively improves the inference speed of the model. In the decoder, the proposed Integrate module extracts the output of the encoder into image feature representations at different resolutions, and the proposed Merge module then gradually fuses these representations into the final dense depth prediction. The resulting model was tested on 3500 images in the collected greenhouse tomato test dataset, showing that the proposed Tomato-MDE achieved the Depth Error Metric (δ_1、δ_2、δ_3), Absolute Relative Error (Abs Rel) and Root Mean Squared Error (RMSE) of 0.883, 0.934, 0.949, 0.107 and 0.371, with an inference speed of 31.62 FPS under the NVIDIA Tesla V100 GPU environment. In the transparent dataset, Tomato-MDE reached the δ_1, δ_2, δ_3, Abs Rel and RMSE of 0.818, 0.887, 0.907, 0.177 and 0.45, respectively. In addition, this method could be further developed into a perception system for a greenhouse tomato picking robot in the future.

Keywords: Greenhouse tomato; Transparent object; Virtual dataset; Monocular depth estimation; Visual Transformer; Agricultural picking robots

1. Introduction

Greenhouse tomatoes play a crucial role in global agricultural vegetable production and trade. They not only have high yields and strong adaptability but also boast high nutritional value, making them highly favored by consumers (Zhang et al., 2024). At present, in greenhouses, tomatoes still depend on manual picking, and this is undoubtedly time-consuming and costly (Li et al., 2023). Agricultural picking robots are one of the best solutions for complex labour tasks such as greenhouse tomato picking, which have to be done manually at the moment. Globally, agricultural picking robots heavily rely on stereo vision cameras to capture both RGB images and depth images of the objects to recognise and locate the crops in three dimensions (3D) (Thakur et al., 2023). In contrast to RGB images, the quality of depth images is severely susceptible to environmental variations, and the accuracy of agricultural picking robots is critically based on the quality of the depth images acquired in real-time. This leads to poor environmental adaptability of the picking robots and unstable picking performance. With the rapid development of deep learning technology in recent years, MDE can quickly learn the most appropriate inductive bias based on the target features in an image, enabling deep mining of the dependencies between the RGB image features and allowing the estimation of crop depth information using only RGB images (Gasperini et al., 2023). This provides a highly robust and environmentally adaptive technological solution for the revolutionary upgrade of the vision system of picking robots in the future. Not only that, MDE has a broad application prospect in agricultural scenarios such as scene deployment, picking path planning, 3D reconstruction, and virtual reality.

Estimating scene depth from a single RGB image is a natural ability for humans, but for computational devices, developing models to accurately predict the depth map corresponding to a single RGB image is a challenging task. To date, MDE methods have been roughly divided into four categories (Arampatzakis et al., 2023), i.e., traditional digital image processing, traditional machine learning, convolutional neural network (CNN), zero-shot, and Visual Transformer (ViT). For traditional digital image processing methods, researchers mainly estimate depth maps based on depth cues (Criminisi et al., 2000) such as linear perspective, focusing, atmospheric scattering, shadows, textures, occlusions, relative heights and motion cues. However, these methods process images in real time by extracting shallow features of depth cues, and are mostly used in simple scenes. They are not suitable for complex, dense greenhouse tomato picking scenes, especially when predicting transparent objects (such as transparent plastic, glass, etc.), where accuracy tends to be relatively poor. For traditional machine leaning methods, many handcrafted features and probabilistic graphical models have been proposed (Saxena et al., 2008), such as scale-invariant feature transform, speeded-up robust features, oriented gradient histogram, conditional random field and Markov random field, for monocular depth map prediction in the process of parametric learning and non-parametric learning. Similar to methods based on traditional digital image processing, the versatility of handcrafted features is insufficient, making the method far from robust. The prediction accuracy in complex scenes is not ideal, and it is also poor for transparent objects. With the dramatic development of deep learning technology, CNN-based methods have come to the forefront (Eigen et al., 2015). These networks typically adopt an encoder-decoder architecture to learn the implicit relationship between color pixels and depth. How to effectively and reasonably extract low-level and high-level features has always been crucial for improving the

quality of depth maps. One solution is to enhance the global environmental modeling capabilities of CNN-based models. However, due to the limitations of the receptive field, it is imperfect. Therefore, various components have been proposed to improve the performance of CNN-based models, such as designing a multi-column structure using input images of different resolutions to extract features at different scales. Nevertheless, these methods suffer from inefficiency in their model structures, with many redundant blocks. Another method is to introduce auxiliary tasks into the encoder-decoder structure to better focus on the global semantic information, but at the cost of higher complexity and training time. In a word, the creative use of the encoder-decoder structure in CNN-based method substantially improves the accuracy and robustness of the MDE model, and significantly enhances the prediction accuracy for transparent objects. However, limited by the working mechanism of CNNs, the optimisation of local semantic information has become a bottleneck. In 2020, zero-shot methods began to emerge in MDE tasks (Ranftl et al., 2020), aiming to establish an MDE foundation model that can generate high-quality depth information for any image in any situation. The core of this method lies in collecting multiple unrelated unlabelled datasets, using affine invariant loss to ignore the potential different depth scales and shifts between different datasets, and training the model on a large scale using a ViT model (typically involving over 60 million images). The high computational cost associated with this approach makes zero-shot methods difficult to meet the real-time requirements of greenhouse tomato picking scenario. One year later, ViT was introduced into MDE models (Yang et al., 2021). Compared to CNNs, which can only operate locally, ViT is better at understanding the global structure. ViT excels at capturing long-range dependencies in an image, as the self-attention mechanism allows the model to integrate information across the entire image. Moreover, the

architecture of ViT reduces the convolutional inductive bias found in CNNs, providing greater learning flexibility during data training. However, the computational overhead of ViT is extremely high, and the computational complexity of the self-attention mechanism grows proportionally with the number of input image patches. As a result, when image resolution increases, both the computational and memory requirements grow rapidly. This makes ViT perform less well than lightweight CNNs in resource-constrained conditions. In short, compared with methods based on CNN, ViT-based methods improve the accuracy of MDE models, but the inference speed is slower, and there is no significant improvement in the prediction accuracy for transparent objects. To sum up, the following observations are proposed to inspire the research in this paper.

(1) Compared to traditional digital image processing methods, traditional machine learning methods, CNN methods, and zero-shot methods, ViT-based methods are currently the most suitable for greenhouse tomato-picking robot scenarios.

(2) ViT-based methods have high computational costs and slow inference speeds, making them difficult to practically apply to tasks such as environmental deployment, picking, picking path planning, and 3D reconstruction in the greenhouse tomato robot picking scenario.

(3) Existing MDE models have poor prediction accuracy for transparent objects, such as the glass commonly used in tomato greenhouses.

(4) There has been very few studies on MDE for agricultural scenarios, and almost no research on MDE for greenhouse tomatoes.

Therefore, this paper proposes a lightweight ViT-based MDE model for greenhouse tomatoes, aiming to identify principles for designing an efficient ViT architecture to quickly and accurately estimate monocular depth for greenhouse tomatoes in complex greenhouse environments, and to

improve the model's ability to predict transparent objects. To the best of the authors' knowledge, this paper is the first attempt to explore the capabilities of MDE in a greenhouse tomato picking scenario. The main contributions are summarised as follows.

1. A combined training approach using the proposed tomato dataset and the public virtual datasets is proposed to improve the prediction ability of the model for transparent objects.

2. A lightweight ViT-based MDE model is proposed to achieve fast and accurate MDE for greenhouse tomatoes.

3. The superiority of the proposed ViT-based model Tomato-MDE is verified by comparing it with the state-of-the-art MDE models based on CNN, zero-shot-based and ViT-based.

The organization of this paper is as follows. Section 2 describes the dataset, the construction of the Tomato-MDE, the evaluation metrics and the experimental setup. Section 3 shows a series of experimental results designed for the proposed model Tomato-MDE. Section 4 discusses the experimental results of Tomato-MDE and compares them with the related literature, illustrates the pros and cons of the proposed model, and gives solutions and future research directions. Section 5 concisely summarises the achievements of the paper.

2. Materials and methods

2.1 Dataset

2.1.1 Greenhouse tomato dataset

The tomato dataset was collected at the tomato smart greenhouse of the Jiangsu Academy of Agricultural Sciences in China from April to June 2023, October to December 2023, and April to June 2024. The acquisition platform was equipped with an Intel RealSense D435i RGB-D camera to simultaneously capture RGB images (1280×720) and depth images (1280×720) of greenhouse

tomatoes, totalling 35,000 pairs. The camera could be freely adjusted in height and angle to facilitate data acquisition from multiple angles. The entire dataset recorded three common types of greenhouse tomatoes, i.e., Globe tomatoes, Cherry tomatoes, and Cluster tomatoes, as shown in Fig. 1.

(a) Globe tomatoes (b) Cherry tomatoes (c) Cluster tomatoes

Fig. 1. Three common types of greenhouse tomatoes.

The original dataset (see Fig. 2 for an example) was randomly split into a training dataset (60%), a validation dataset (30%), and a test dataset (10%), with RGB images used as input data and depth images used as labels. Furthermore, to evaluate the prediction accuracy of the proposed model for transparent objects, we specifically selected 1000 tomato images containing transparent objects (transparent plastic, glass, etc.) from the original test dataset to form an additional transparent object test dataset.

2.1.2 Data scaling-up

At present, MDE models have poor perception of transparent objects. In a tomato greenhouse, there are usually many glass or transparent plastic materials, which can severely affect the accuracy of tomato depth estimation. Even with a significant expansion of the tomato dataset, the newly trained model still struggles to perceive transparent objects effectively. This prompts us to reconsider whether the depth inaccuracies stem from the prediction model or the data itself. Naturally, we wondered whether training with a virtual dataset containing complete depth information would have a positive impact on the model's learning. Thus, we extensively

researched two commonly used virtual datasets (see Fig. 2 for examples), Hypersim (Roberts et al., 2021) and Virtual KITTI (Cabon et al., 2020). The Hypersim virtual dataset is a virtual dataset introduced by Apple's machine learning team for comprehensive indoor scene understanding, aimed at addressing the challenges of understanding the complexity of indoor scenes in the computer vision community. The Hypersim dataset contains 77,400 color images and corresponding depth images from 461 indoor scenes. These images are generated from a large synthetic scene repository created by professional artists, featuring high-resolution textures and dynamic lighting. The Virtual KITTI virtual dataset was not developed by a specific company but was collaboratively created by researchers in the fields of computer vision and autonomous driving. It is a highly realistic virtual dataset that simulates real-world road environments and traffic scenes. The Virtual KITTI dataset contains 50 different virtual world scenes, 21,260 color images, and corresponding depth images. It leverages the powerful rendering capabilities of the Unity game engine to generate complex scenes containing multiple dynamic vehicles, pedestrians, traffic signs, and other elements. These scenes are not only visually realistic but also rich in physical and geometric information. It is important to note that both the Hypersim and Virtual KITTI datasets are open-source, and therefore do not raise any privacy or ethical concerns.

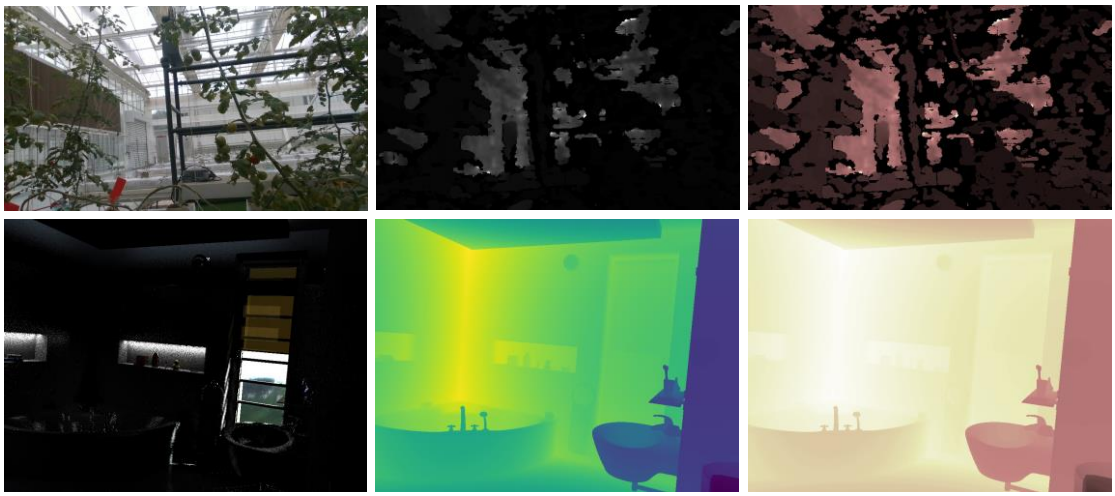

Fig. 2. The proposed tomato dataset and two virtual datasets. The first row shows an example from the proposed tomato dataset, the second row presents an example from the Hypersim virtual dataset, and the third row illustrates an example from the Virtual KITTI virtual dataset. The first column contains the original images, the second column displays the corresponding original depth maps, and the third column provides a visualization of the depth maps.

To combine the original tomato dataset with the two virtual datasets, the depth values were first converted to the disparity space and then normalized to 0~1 on each depth map. To realize joint training of multiple datasets, an affine invariant loss (Ranftl et al., 2020) was used to ignore the unknown scale and displacement of each sample:

$$\mathcal{L}_t = \frac{1}{HW}\sum_{i=1}^{HW} \gamma\,(d_i', d_i), \tag{1}$$

where d_i' and d_i are the predicted and ground truth values, respectively. γ is the affine invariant mean absolute error loss: $\gamma\,(d_i', d_i) = \left|\hat{d}_i' - \hat{d}_i\right|$, where $\hat{d}_i'$ and $\hat{d}_i$ are the scaled and translated versions of the predicted d_i' and ground truth values d_i, respectively.

$$\hat{d}_i = \frac{d_i - p(d)}{g(d)}, \tag{2}$$

where $p(d)$ and $g(d)$ are used to align the predicted value and the actual ground situation, so that they have zero translation and unit scale:

$$p(d) = mean(d)\,,\ g(d) = \frac{1}{HW}\sum_{i=1}^{HW} |d_i - p(d)|. \tag{3}$$

It is noteworthy that the validation and test datasets of the expanded dataset remain unchanged as in the case of the tomato dataset.

2.2 Tomato-MDE

Tomato-MDE is a lightweight MDE model jointly trained from labelled original real images

and virtual images with full depth information, which can adapt to complex smart greenhouse environments containing plenty of transparent objects. Tomato-MDE employs an encoder-decoder network structure as shown in Fig. 3. First, the expanded image resolution is uniformly adjusted to 518 × 518, and subsequently split into fixed-size image patches (14 × 14), and a certain proportion of image patches are randomly masked to enhance the learning ability of the model. Then, the corrupted image patches are linearly projected into a one-dimensional vector sequence and sent it to the encoder. The encoder mainly consists of repeated fast self-attention module (FSAM) and depthwise convolution (DWConv). Among them, the FSAM comprises feed forward neural network (FFN), DWConv, the proposed ReLU linear self-attention module (RLSAM), FFN, and DWConv in sequence (See Fig. 4), which can effectively improve the inference speed of the model. The decoder includes Integrate module and Merge module, the output of the encoder is efficiently extracted into image feature representations of different resolutions through the Integrate module, and then these feature representations are gradually fused into the final dense prediction through the Merge module, so as to realise the MDE work of the greenhouse tomato.

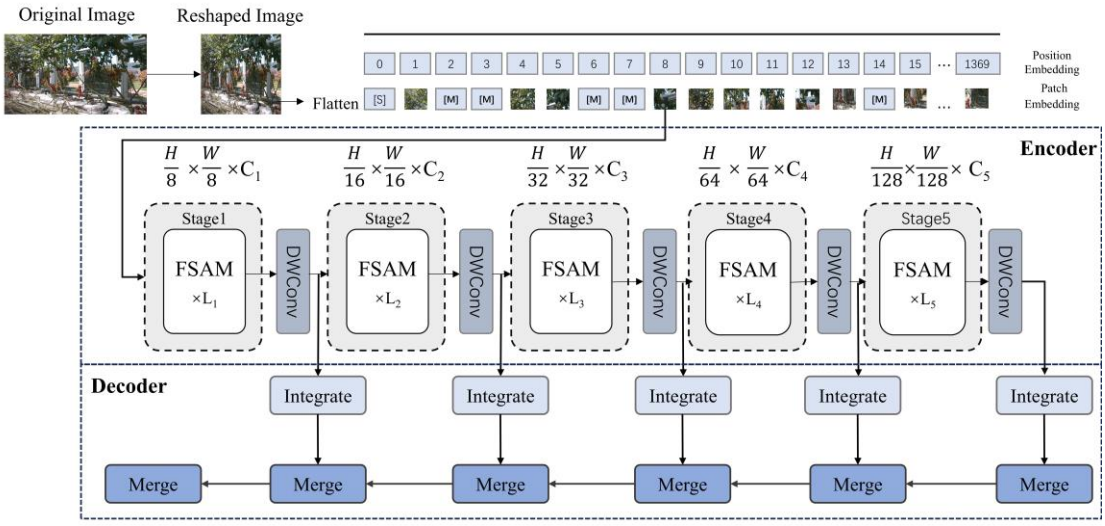

Fig. 3. Structure of the Tomato-MDE model.

2.2.1 Image input

The images are fed into the network, where *H*, *W*, and *C* are their height, width, and channel size, respectively. After that, the input image is uniformly segmented into $K \times K$ image patches, each of size $H/K \times W/K \times C$. These 3D image patches are then flattened into a one-dimensional sequence $x \in \mathbb{R}^{N \times D}$ by applying a learnable projection $f: x_i \to e_i \in \mathbb{R}^D (i = 1, .. N)$, where $N = \frac{HW}{K^2}$, $D = K \times K \times 3$, into the sequence $e \in \mathbb{R}^{N \times D}$. The projection operation converts the spatial and channel features of the image patches into a sequence of embedded features that serve as the input representation for the encoder. For our task, the original image resolution is adjusted to 518 × 518 and segmented into 1369 (518 × 518/14 × 14) image patches, each of which has a resolution of 14 × 14. ViT requires more training data than CNN, and to solve the data-starvation problem, a certain proportion of the image patches are randomly masked before they enter the encoder. The masking position is denoted as $P \in \{1, \dots, s\}^{0.3s}$, where *s* is the total number of image patches and 0.3 is the masking rate. Next, the masked image patch is replaced using a learnable embedding $e_{[P]} \in \mathbb{R}^D$, and then the image patch $x^P = \{x_i^k : i \notin P\}_{i=1}^s \cup \{e_{[p]} : i \in P\}_{i=1}^s$ is fed into the encoder. The final hidden vector $\{\boldsymbol{v}_i^k\}_{i=1}^s$ is considered as a coded representation of the image patch. For each masked position $\{\boldsymbol{v}_i^k : i \in P\}_{i=1}^s$, a softmax classifier is used to predict the corresponding visual labelling $p_M(z' \mid x^P) = \text{softmax}_{z'}\left(\boldsymbol{W}_c \boldsymbol{v}_i^k + \boldsymbol{b}_c\right)$, where x^P is the masked image patch, $\boldsymbol{W}_c \in \mathbb{R}^{|\mathcal{V}| \times D}$ and $\boldsymbol{b}_c \in \mathbb{R}^{|\mathcal{V}|}$. It is worth mentioning that the masked positions are not random, instead, they are masked in a clockwise direction, one image patch at a time.

2.2.2 Encoder

Recent studies have shown that operations with high computational complexity are mainly located in the self-attention layer, rather than in the FFN layer (Dong et al., 2021). However, most

existing ViTs use an equal number of these two layers. Hence, we propose an efficient encoder that can use less memory-constrained self-attention layer and more memory-efficient FFN layer for channel communication. The overall computational flow of the encoder is as follows:

$$x_{k+1} = \prod^{N} \Phi_k^{\text{FFN+DWConv}} \left(\Phi_k^{\text{A}} \left(\prod^{N} \Phi_k^{\text{FFN+DWConv}} (x_k) \right) \right) \tag{4}$$

where x_k is the complete input features of the kth patch, the patch x_k will be converted to x_{k+1}, Φ_k^{A} represents the proposed RLSAM. In the RLSAM, given an input feature map $X \in \mathbb{R}^{N \times D}$, where N is the number of image patches and D is the feature dimension, the computational complexity of the RLSAM is $O(N^2 \cdot D)$ and the size of the RLSAM matrix is $N \times N$, storing these values consumes a lot of memory. Especially at high resolution images, this quadratic growth in memory requirement can become a bottleneck. Unlike the RLSAM, the FFN usually consists of two linear layers and a nonlinear activation function with a computational complexity of $O(N \times D^2)$, and the memory requirement of the FFN is linearly related to the number of image patches, so the memory consumption of the FFN is relatively low at high resolution inputs. Specifically, in the encoder, a single RLSAM is preceded and followed by two FFNs. with this design, the model can rely on the FFN layer for efficient channel communication without requiring a large number of RLSAMs. Linear computation of the FFN layer reduces the memory footprint while preserving the information exchange across channels. On the other hand, RLSAM is used to capture global features, but its memory consumption is significantly reduced by reducing it to a single feature. It is worth highlighting that an additional DWConv is applied before each FFN. Compared to standard convolution operations, DWConv has significantly reduced computational intensity while maintaining the ability to extract local features. The computational complexity of deep convolution is $O(K^2 \cdot D \cdot N)$, where K is the size of the

convolution kernel, *D* is the number of channels, and *N* is the spatial size of the input. Compared to standard convolution (complexity of $O(K^2 \cdot D^2 \cdot N)$), DWConv effectively reduces the parameters and computational cost. Therefore, it can introduce local structure awareness to the model and improve its performance without significantly increasing the computational burden.

For RLSAM (refer to Fig. 4), the redundancy of attention heads is a serious issue, directly leading to inefficient computation. As a result, we propose a feature segmentation strategy that splits the complete feature differently for each head, explicitly decomposing attention computation across heads, as shown in the following equations.

$$\tilde{x}_{km} = Attn\left(X_{km}W_{km}^{\mathrm{Q}}, X_{km}W_{km}^{\mathrm{K}}, X_{km}W_{km}^{\mathrm{V}}\right) \tag{5}$$

$$\tilde{x}_{k+1} = Concat[\tilde{x}_{km}]_{m=1:h}W_{k}^{\mathrm{P}},$$

where the *m*th head computes the self-attention of x_{km}, x_{km} is the *m*th segmentation of the input features x_k, i.e. $x_k = [X_{k1}, X_{k2}, \dots, X_{kh}]$, $1 \le m \le h$, and h is the total number of heads, W_{km}^{Q}, W_{km}^{K} and W_{km}^{V} are projection layers that map the input feature segmentations to different subspaces, and W_{k}^{P} is a linear layer that projects the tandem output features to dimensions consistent with the input.

Although using feature segmentation for each head instead of the complete feature is more efficient and saves computational overhead, the capacity can be further improved by encouraging the *Q*, *K*, and *V* layers to learn feature projections with richer information. We compute the attention maps for each head in a cascaded manner, as shown in Fig. 4, where the output of each head is added to the subsequent head, progressively enhancing the feature representation:

$$x'_{km} = X_{km} + \tilde{X}_{k(m-1)}, \ 1 \le m \le h, \tag{6}$$

where x'_{km} is the summation of the *m*th input segmentation point x_{km} with the (*m*-1)th head

output $\tilde{x}_{k(m-1)}$. It replaces x_{km}, when computing self-attention, as the new input feature for the *m*th head. Also, DWConv is applied after the Q projection, allowing the self-attention mechanism to jointly capture both local and global relationships, further enhancing the feature representation. This cascaded design has two advantages. First, assigning different feature segmentations to each head increases the diversity of the attention maps. Second, cascading attention heads increases the network depth, thereby further improving the model's capacity without introducing any additional parameters. Since the attention map computation for each head uses a smaller QK channel dimension, it only incurs a small delay overhead.

Besides, the RLSAM uses ReLU linear attention (Niu et al., 2021) to enable a global receptive field, instead of relying on heavy softmax attention. Specifically, given the input $x \in \mathbb{R}^{N\times g}$, the generalised form of softmax attention can be written as:

$$O_k = \sum_{m=1}^{N} \frac{S(Q_k,K_m)}{\sum_{m=1}^{N} S(Q_k,\mathrm{m})} V_m \tag{7}$$

where $Q = xW_Q,\ K = xW_K,\ V = xW_V$, $W_Q/W_K/W_V \in \mathbb{R}^{g\times d}$ is the learnable linear projection matrix. O_k denotes the *k*th row of the matrix *O*. $S()$ is the similarity function. When the similarity function $S(Q,K) = \exp\left(\frac{QK^T}{\sqrt{d}}\right)$ is used, Eq. (7) becomes the original softmax attention. In addition to $\exp\left(\frac{QK^T}{\sqrt{d}}\right)$, ReLU linear attention is used to achieve global receptive fields and linear computational complexity. In ReLU linear attention, the similarity function is defined as:

$$S(Q,K) = \mathrm{ReLU}(Q)\mathrm{ReLU}(K)^T \tag{8}$$

Since $S(Q,K) = \mathrm{ReLU}(Q)\mathrm{ReLU}(K)^T$, Eq. (7) can be rewritten as:

$$O_k = \sum_{m=1}^{N} \frac{\mathrm{ReLU}(Q_k)\mathrm{ReLU}(K_m)^T}{\sum_{m=1}^{N}\mathrm{ReLU}(Q_k)\mathrm{ReLU}(K_m)^T} V_m = \frac{\sum_{m=1}^{N}(\mathrm{ReLU}(Q_k)\mathrm{ReLU}(K_m)^T)V_m}{\mathrm{ReLU}(Q_k)\sum_{m=1}^{N}\mathrm{ReLU}(K_m)^T} \tag{9}$$

And then, the associative property of matrix multiplication is exploited to reduce the memory footprint and computational complexity from quadratic to linear without changing its functionality:

$$O_k = \frac{\sum_{m=1}^{N} [ReLU(Q_k)ReLU(K_m)^T]V_m}{ReLU(Q_k)\sum_{m=1}^{N} ReLU(K_m)^T} = \frac{ReLU(Q_k)(\sum_{m=1}^{N} ReLU(K_m)^T V_m)}{ReLU(Q_k)(\sum_{m=1}^{N} ReLU(K_m)^T)} \quad (10)$$

As shown in Eq. (10), $((\sum_{m=1}^{N} \mathrm{ReLU}(K_m)^T V_m) \in \mathbb{R}^{d\times d}$ and $((\sum_{m=1}^{N} \mathrm{ReLU}(K_m)^T) \in \mathbb{R}^{d\times 1}$ need to be computed only once and they can be reused in each query, thus requiring only O(N) computational cost and memory.

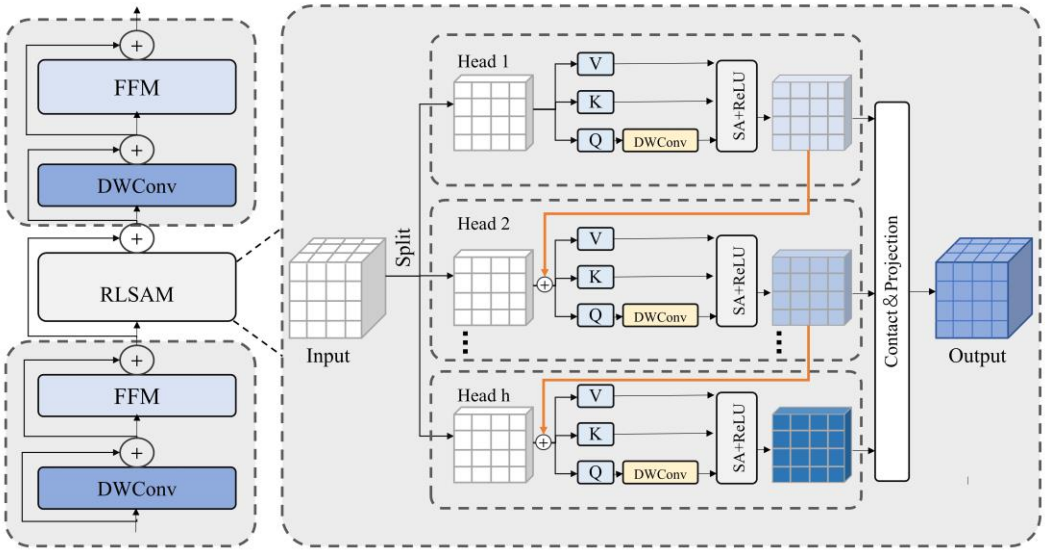

Fig. 4. Structure of the FSAM.

2.2.3 Decoder

After refining massive multi-scale feature information, the decoder is used to progressively reconstruct the high-dimensional compressed features output from the encoder into a depth map with the same resolution as the input image. The decoder continues the encoder's lightweight design, consisting of two simple proposed modules (the Integrate module and the Merge module) stacked together, as shown in Fig. 5. The Integrate module combines tomato features from the encoder at various stages and resolutions, blending high-level semantic features at lower resolutions with low-level features at higher resolutions. Subsequently, a projection is used as reading operations to generate feature maps of different dimensions for the Merge module. The

Merge module uses DWConv to assemble the tomato features extracted from different network stages and upsamples the feature representations at each fusion stage. Finally, with task-specific output headers, the final dense tomato prediction results are generated.

Specifically, the decoder first inputs the image feature representations from different network stages of the encoder into multiple Integrate modules, to recover the class image representations from the encoder's output tokens, as follows:

$$\text{Integrate}_{r}^{\hat{C}}(i) = (\text{Resample}_{s} \circ \text{Concat} \circ \text{Read})(i) \tag{11}$$

where r denotes the ratio of the output size of the recovered representation relative to the input image , and $\hat{C}$ denotes the output feature dimension.

The N+1 tokens are mapped to a set of N tokens that can be spatially connected and mapped to an image-like representation:

$$\text{Read} : \mathbb{R}^{N+1\times D} \rightarrow \mathbb{R}^{N\times D} \tag{12}$$

This operation is primarily responsible for appropriately handling the readout tokens. Although the role of the readout tokens in dense prediction tasks is not significant, they still have potential utility in capturing and distributing global information. Therefore, we elaborate on three different operations for this mapping:

$$\text{Read}_{\text{ignore}}(i) = \{i_1, \dots, i_N\} \tag{13}$$

It will directly ignore the readout tokens,

$$\text{Read}_{add}\ (i) = \{i_1 + i_0, \dots, i_N + i_0\} \tag{14}$$

Pass information from the readout tokens to all other tokens by adding their representations. Then, project the resulting representation to the original feature dimension C using linear layers followed by GELU non-linear layers.

$$\text{Read}_{\text{proj}}\ (i) = \{\text{mlp}\ (\text{Concat}(i_1, i_0)), \dots,$$

$$\text{mlp}(\text{Concat}(i_N, i_0))\} \tag{15}$$

After a Read block, each token can be reshaped into an image-like representation based on the location of the initial patch in the image, resulting in *N* tokens. Formally, a spatial join operation is employed to obtain a feature map of size $\frac{H}{p} \times \frac{W}{p}$ with *C* channels:

$$\text{Concat} : \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times C} \tag{16}$$

This representation is transferred to the spatial resampling layer, which scales the representation to size $\frac{H}{r} \times \frac{W}{r}$, with $\hat{C}$ feature a per pixel:

$$\text{Resample}_{r} : \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times C} \rightarrow \mathbb{R}^{\frac{H}{r} \times \frac{W}{r} \times \hat{C}} \tag{17}$$

In the Integrate module, a non-linear activation function is used to improve the quality of tomato feature reading during the Read operation. The Concat operation merges multiple feature channels, helping the model process richer information at lower dimensions, making the network structure more concise and the data flow through the model more direct. The Resample operation adjusts the tomato feature size, reducing the amount of data entering the model. By combining these three simple operations, the efficiency of the Integrate module is ensured.

After that, in the Merge module, lightweight DWConv is used to gradually fuse feature representations, followed by the Resample operation to adjust the size of the feature representations. Finally, a 1×1 convolution is applied to project inputs of different resolutions, generating fine-grained predictions and completing the lightweight decoding task. The Merge module combines DWConv, Resample operations, and 1×1 convolution, making the module structure simple and efficient.

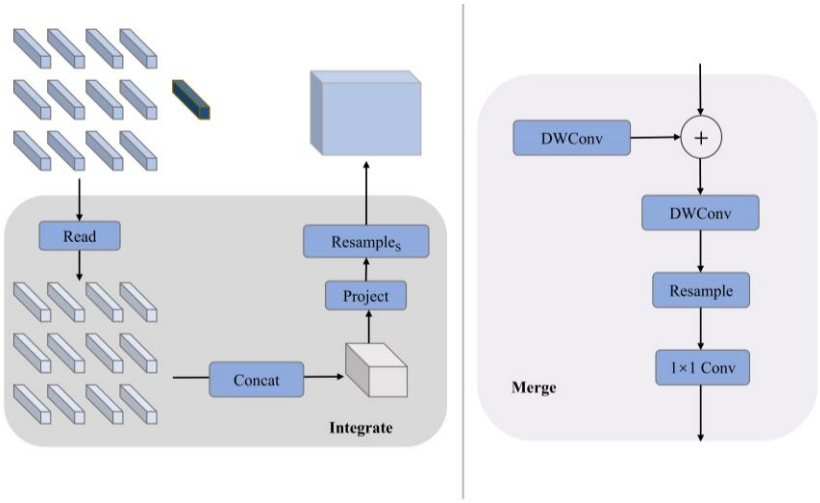

Fig. 5. Structure of the decoder.

2.3 Loss function

In the tomato greenhouse, depth data is densely collected from the near area, while data collected from farther distances is very sparse due to the limitations of the Intel RealSense D435i RGB-D camera. To mitigate this problem, data loss L_d is introduced in the adopted square root loss function to compute the difference between the predicted and true depth values:

$$L_d(y, y^*) = \sqrt{\frac{1}{n}\sum_{i\in V}^{N_V} d_i^2 - \frac{\lambda}{n^2}\left(\sum_{i\in V}^{N_V} d_i\right)^2} \tag{18}$$

where $d_i = log\, y_i - log\, y_i^*$. V is the depth map of a set of effective pixels. N_V denotes the total number of effective pixels. λ represents the balance factor.

For our task, data loss is often concerned with minimising the overall error, which may cause the model to underperform in capturing high-frequency details of the image. These regions are visually important because they define the structure and spatial layout of the object. In addition, data loss tends to produce excessively smooth depth maps because it penalizes large prediction errors, causing the model to sacrifice small-scale high-frequency information for an overall low error. As such, for the loss function L_t in our task, a gradient loss L_g is added to the data loss L_d, aiming to enhance the model's ability to perceive and predict boundary and detail features in the

image. The gradient loss equation is as follows:

$$L_g(y, y^*) = \frac{1}{N}\left|y_{h,i} - m(y^*)_{h,i}\right| + \left|y_{v,i} - m(y^*)_{v,i}\right| \tag{19}$$

where $m(\cdot)$ denotes the interpolation function. $y_{h,i}$ and $m(y^*)_{h,i}$ indicate the ith gradient value of the estimated depth map and the interpolated true value in the horizontal direction , respectively. Similarly, $y_{v,i}$ and $m(y^*)_{v,i}$ are defined in the vertical direction. N is the total number of pixels contained in the estimated depth map.

In summary, the loss function of Tomato-MDE is given as follows:

$$L_t = \begin{cases} \alpha L_d(y, y^*), \text{if epoch} < 30, \\ \alpha L_d(y, y^*) + \beta L_g(y, y^*), \text{otherwise} \end{cases} \tag{20}$$

where y and y^* denote the predicted depth map and the true depth map, respectively. α and β are the balancing factors for L_d and L_g, used to adjust the relative importance of these two losses in the total loss function, ensuring that the model effectively learns these key features during training. Note that the gradient loss is computed after 30 epochs. During the first 30 epochs of training, only data loss is used to bootstrap the model's basic MDE capability. This helps the model stably learn global depth features without being affected by local gradient errors. Once the model achieves better stability in the MDE task, gradient loss is gradually introduced. This allows the already relatively accurate MDE results to refine the model's ability to handle edges and details, further improving the quality and accuracy of the depth map.

2.4 Evaluation metrics

The metrics proposed by Eigen et al. (Eigen et al., 2015) were used for the performance evaluation of Tomato-MDE as follows:

$$\delta = max\left(\frac{y}{y^*}, \frac{y^*}{y}\right) < t \tag{21}$$

$$Abs\ Rel = \frac{1}{|T|}\sum_{y \in T}|y - y^*| / y^* \tag{22}$$

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{y \in T} y - y^{*2}} \quad (23)$$

where y and y^*refer to the predicted depth map and the real depth map, respectively. The threshold t is set to 1.25, 1.25^2 and 1.25^3, corresponding to δ_1, δ_2 and δ_3, respectively. T is the total number of valid pixels in the real depth map.

Moreover, Frames Per Second (FPS) is a common indicator for evaluating the inference speed of the visual models. It represents the number of times the graphics processor renders frames per second, i.e., the number of frames displayed within one second. The equation is as follows:

$$FPS = 1/t \quad (24)$$

where t refers to the inference time, which is calculated by inferring a set of input images and averaging the time.

2.5 Experimental Setup

All experiments were conducted on Ubuntu 20 operating system using PyTorch on 4 Tesla V100 GPUs. The core framework was Tomato-MDE. During training, the key hyperparameters were set to epoch = 1800, learning rate = 0.0002, and batch size = 64. In the loss function, the equilibrium factors α and β were set to 2 and 0.2, respectively, and the balance factor λ was set to 0.85. The optimizer was AdamW (Loshchilov et al., 2017).

3. Results

3.1 Impact of dataset size on the Tomato-MDE

To explore the optimal dataset setup, the size of the tomato dataset needs to be determined first. Concretely, Images were randomly selected from the training dataset of the tomato dataset to build a new training dataset, and the validation dataset and test dataset were kept unchanged.

There are seven new training datasets used for here, each containing 1000, 4000, 7000, 10000, 13000, 16000 and 20000 images, respectively. The results are shown in Table 1.

Table 1. Impact of tomato dataset size on Tomato-MDE.

dataset size	δ_1	δ_2	δ_3	Abs Rel	RMSE
1000	0.184	0.324	0.426	0.865	2.749
4000	0.311	0.514	0.628	0.526	2.409
7000	0.554	0.767	0.859	0.433	0.927
10000	0.685	0.859	0.917	0.246	0.720
13000	0.763	0.887	0.926	0.196	0.648
16000	0.794	0.917	0.950	0.178	0.601
20000	0.791	0.906	0.939	0.181	0.612

When the size of the tomato dataset was less than 10,000 images, the performance of Tomato-MDE increased rapidly (δ_1 increased from 0.184 to 0.685, Abs Rel decreased from 0.865 to 0.246 and RMSE decreased from 2.749 to 0.720). When the size of the tomato dataset exceeded 13,000 images, the model performance increased slowly (δ_1 increased from 0.763 to 0.794, Abs Rel decreased from 0.196 to 0.178 and RMSE decreased from 0.648 to 0.601). When the size of the tomato dataset reached 16,000 images, the model tended to converge (δ_1=0.794, δ_2=0.917, δ_3=0.950, Abs Rel=0.178, RMSE=0.601). Therefore, the optimal size of the tomato dataset was set to 16,000 images.

Then, the Tomato-MDE, trained on the optimal tomato dataset, was tested on the transparent object test dataset (see Section 2.1.1), to evaluate the prediction performance of the proposed model for transparent objects, and the results are given in Table 2.

Table 2. Impact of Tomato-MDE trained on the Tomato dataset for transparent objects.

Dataset	δ_1	δ_2	δ_3	Abs Rel	RMSE
Tomato dataset	0.565	0.754	0.829	0.481	0.790

As can be seen from Table 1, Tomato-MDE did not perform well in the test dataset of the tomato dataset (δ_1=0.794, δ_2=0.917, δ_3=0.950, Abs Rel=0.178, RMSE=0.601), and its prediction ability for transparent objects was even worse (δ1 decreased by 40.43%, δ2 decreased by 21.62%,

δ3 decreased by 14.6%, Abs Rel increased by 170.22%, and RMSE increased by 31.45%). To enhance the comprehensive performance of Tomato-MDE, the dataset scaling-up approach proposed in Section 2.1.2 was introduced. Images were randomly selected from the Hypersim virtual dataset and Virtual KITTI virtual dataset and combined them with the training dataset of the tomato dataset (see Section 2.1.2). The validation dataset and test dataset remained unchanged. The number of virtual images extracted was 10,000, 20,000, 30,000, 40,000, 50,000, 60,000, 70,000, 80,000 and 90,000 respectively, and the results are presented in Table 3.

Table 3. Impact of virtual dataset size on Tomato-MDE.

Virtual dataset size	δ_1	δ_2	δ_3	Abs Rel	RMSE
10000	0.682	0.842	0.897	0.245	0.822
20000	0.695	0.848	0.897	0.238	0.690
30000	0.724	0.884	0.931	0.211	0.773
40000	0.776	0.914	0.947	0.184	0.638
50000	0.807	0.904	0.932	0.175	0.575
60000	0.813	0.911	0.937	0.158	0.526
70000	0.832	0.920	0.944	0.146	0.572
80000	0.883	0.934	0.949	0.107	0.371
90000	0.881	0.923	0.934	0.111	0.395

From Table 3, when the size of the virtual dataset was less than 30,000 images, the performance of Tomato-MDE decreased markedly, with the lowest values of δ1=0.682, δ2=0.842, δ3=0.897, Abs Rel=0.245 and RMSE=0.822. This may be because as the proportion of heterogeneous data increased, the domain gaps in the data caused the model failing to immediately adapt to the new data distribution, resulting in negative transfer, making the model perform degraded on the new dataset. As the size of the virtual dataset further enlarged, when it was larger than 30,000 images, the performance of the model began to gradually improve. When the virtual dataset size reached 80,000 images, the model tended to saturate (δ1=0.883, δ2=0.934, δ3=0.949, Abs Re=0.107, RMSE =0.371). This indicated that as more virtual data was added, a stronger relationship was established between different data distributions. The model progressively boosted

its cross-domain generalisation capabilities, thus improving the understanding of scenes and objects, which is also a strength of ViT. When the size of the virtual dataset was over 80000 images, more data failed to further improve the performance of the model, revealing that the model had reached its upper performance limit under the current structure and hyper-parameter settings, and the model's learning capability had been fully utilized. Based on the above results, the optimal size of the virtual dataset was set to 80,000 images. To validate the prediction ability of Tomato-MDE for transparent objects after adding the virtual dataset, we tested the retrained Tomato-MDE on the transparent object test dataset, and the results are listed in Table 4.

Table 4. Impact of Tomato-MDE trained on the Tomato dataset and virtual dataset for transparent objects.

Dataset	δ_1	δ_2	δ_3	Abs Rel	RMSE
Tomato dataset + virtual dataset	0.818	0.887	0.907	0.177	0.450

Compared with just using the tomato dataset, the prediction ability of Tomato-MDE for transparent objects has significantly improved after incorporating the virtual dataset (δ1 improved by 44.78%, δ2 improved by 17.64%, δ3 improved by 9.41%, Abs Re decreased by 171.75% and RMSE decreased by 75.56%). Surprisingly, the virtual dataset not only increased the prediction ability of Tomato-MDE for transparent objects, it also enormously improved the overall depth estimation performance of the model, with δ1 improving by 11.21%, Abs Rel decreasing by 66.36%, and RMSE decreasing by 61.99%. Fig. 6 exhibits some examples of monocular depth estimation for transparent objects.

Fig. 6. Examples of monocular depth estimation for transparent objects. The first column is the original image, the second column is the estimated depth map, and the third figure is the heat map of the estimated depth map. The estimated window outline is clear. In the first row, the dense glass could be relatively well represented. In the second row, the glass could still be estimated when it was blocked. In the third row, the proposed model could distinguish the glass and the foreground.

3.2 Ablation experiments

The number of FSAM in the encoder, the RLSAM, and the decoder were the crucial components in building Tomato-MDE. Accordingly, ablation experiments were designed for these three components. For the number of FSAM modules, it was set to 3, 4, 5, 6, and 7, respectively. For the RLSAM, the feature segmentation strategy and the ReLU linear attention mechanism were verified. For the decoder, the proposed decoder was compared with the decoders of fifteen state-of-the-art MDE models (FastDepth, SDC-Depth, LapDepth, CADepth, AdaBins, LocalBins, GfD, DPT-Hybrid, PixelFormer, URCDC-Depth, NDDepth, BinsFormer, ECoDepth, UniDepth, Depth Anything). The results are illustrated in Table 5.

Table 5. Results of ablation experiments. FSS stands for feature segmentation strategy, and ReLU represents ReLU linear attention mechanism.

Strategy	FPS	δ_1	δ_2	δ_3	Abs Rel	RMSE
FSAM=3	41.35	0.807	0.901	0.927	0.183	0.507

FSAM=4	35.18	0.853	0.929	0.948	0.131	0.521
FSAM=5	31.62	0.883	0.934	0.949	0.107	0.371
FSAM=6	27.39	0.887	0.938	0.951	0.104	0.444
FSAM=7	24.81	0.892	0.936	0.948	0.102	0.433
Ours-FSS	26.14	0.886	0.939	0.953	0.112	0.458
Ours-ReLU	30.02	0.881	0.928	0.941	0.116	0.420
FastDepth	36.37	0.811	0.833	0.904	0.177	0.492
SDC-Depth	24.12	0.852	0.916	0.934	0.145	0.498
LapDepth	23.25	0.855	0.922	0.941	0.141	0.501
CADepth	28.03	0.856	0.921	0.940	0.143	0.493
AdaBins	20.35	0.882	0.935	0.949	0.113	0.463
LocalBins	22.45	0.864	0.918	0.933	0.139	0.484
GfD	29.58	0.851	0.915	0.934	0.132	0.504
DPT-Hybrid	18.11	0.889	0.937	0.951	0.115	0.446
PixelFormer	15.29	0.868	0.914	0.926	0.134	0.447
URCDC-Depth	30.26	0.854	0.913	0.930	0.128	0.480
NDDepth	27.71	0.848	0.911	0.931	0.139	0.499
BinsFormer	17.07	0.878	0.923	0.936	0.122	0.468
ECoDepth	32.68	0.831	0.906	0.930	0.157	0.505
UniDepth	33.22	0.847	0.912	0.932	0.153	0.515
Depth Anything	26.53	0.873	0.920	0.934	0.122	0.451
Tomato-MDE	31.62	0.883	0.934	0.949	0.107	0.371

For the FSAM module in the encoder, when the number of FSAM modules was set to 7, Tomato-MDE achieved the optimal prediction accuracy (δ1=0.892, δ2=0.936, δ3=0.948, Abs Rel=0.102, RMSE=0.433), but the inference speed was slow and did not reach real-time inference speed (FPS=24.81). When the number of FSAM modules was set to 3, Tomato-MDE had the fastest inference speed, reaching 41.35, but the prediction accuracy was noticeably low (δ1=0.807, δ2=0.901, δ3=0.927, Abs Rel=0.183, RMSE=0.507). Considering both prediction accuracy and inference speed, the Tomato-MDE used five FSAM modules, and the prediction accuracy was only slightly lower than the optimal prediction accuracy (δ1 decreased by 1.02%, δ2 decreased by 0.21%, δ3 improved by 0.11%, Abs Rel improved by 4.9% and RMSE decreased by 16.71%), but the inference speed was obviously faster (FPS increased by 6.81). In the tomato-picking robot scenario, balancing prediction accuracy and inference speed have to be taken into account. For the RLSAM, it was observed that using the feature segmentation strategy and the ReLU linear attention mechanism can improve the inference speed (the FPS increased by 5.48 and 1.6, respectively) while maintaining almost the same prediction accuracy of Tomato-MDE. Based on

the description of the solution to improve computational efficiency from a mathematical and logical perspective in Section 2.2.2, the experimental results intuitively showed the effectiveness of the proposed feature segmentation strategy and ReLU linear attention mechanism. For the decoder, the prediction accuracy of the proposed decoder was slightly lower than that of the optimal encoder, DPT-Hybrid (δ1 was 0.006 lower, δ2 was 0.003 lower, δ3 was 0.002 lower, Abs Rel was 0.008 lower and RMSE was 0.075 lower), but the inference speed enjoyed a tremendous improvement (FPS is 13.51 higher). Sacrificing negligible accuracy for huge inference speed was undoubtedly worthwhile.

3.3 Comparison with state-of-the-art models

To fully validate the performance of the proposed method, we compared Tomato-MDE with fifteen state-of-the-art MDE models (Table 6). The first seven of these models are based on CNN, the eighth to tenth are based on zero-shot, and the last five are based on ViT. Our proposed model is based on ViT.

Table 6. Comparison with state-of-the-art monocular depth estimation models.

Method	Architecture	FPS	δ_1	δ_2	δ_3	Abs Rel	RMSE
FastDepth (Wofk et al., 2019)	CNN	32.53	0.745	0.866	0.899	0.203	0.583
SDC-Depth (Wang et al., 2020)	CNN	21.86	0.861	0.921	0.937	0.132	0.457
LapDepth (Song et al., 2021)	CNN	35.42	0.807	0.880	0.904	0.177	0.513
CADepth (Yan et al., 2021)	CNN	28.01	0.825	0.895	0.917	0.156	0.531
AdaBins (Bhat et al., 2021)	CNN	24.33	0.828	0.897	0.918	0.149	0.520
LocalBins (Bhat et al., 2022)	CNN	25.57	0.831	0.902	0.921	0.155	0.482
GfD (Haji-Esmaeili et al., 2024)	CNN	25.68	0.839	0.901	0.920	0.158	0.492
ECoDepth (Patni et al., 2024)	zero-shot	5.39	0.838	0.907	0.927	0.154	0.509
UniDepth (Piccinelli et al., 2024)	zero-shot	5.67	0.844	0.908	0.926	0.150	0.518
Depth Anything(Yang et al., 2024)	zero-shot	4.48	0.832	0.907	0.928	0.157	0.474
DPT-Hybrid (Ranftl et al., 2021)	ViT	25.99	0.845	0.915	0.935	0.148	0.503
PixelFormer (Agarwal et al., 2023)	ViT	23.12	0.841	0.915	0.933	0.149	0.430
URCDC-Depth (Shao et al., 2023)	ViT	24.16	0.816	0.913	0.937	0.166	0.497
NDDepth (Shao et al., 2023)	ViT	21.25	0.829	0.924	0.948	0.142	0.502
BinsFormer (Li et al., 2024)	ViT	26.71	0.831	0.918	0.941	0.154	0.475
Tomato-MDE	ViT	31.62	0.883	0.934	0.949	0.107	0.371

The proposed Tomato-MDE achieved the best prediction accuracy (δ1=0.883, δ2=0.934, δ3=0.949, Abs Rel=0.107, RMSE=0.371) and the third fastest inference speed among all models

(FPS=31.62). Considering the excellent performance of the proposed model, sacrificing few inference speeds was acceptable. Notably, the top two models in terms of inference speed were both CNN-based, which showed that CNN-based models had a unique advantage in computation cost. Moreover, the models based on zero-shot showed promising performance, especially for UniDepth ($\delta1$=0.844, $\delta2$=0.908, $\delta3$=0.926, RMSE=0.15, RMSE=0.518). There is nothing related to tomatoes in the training dataset for the zero-shot models, which indicates that the foundation model built with sufficient data possessed the potential to estimate any object. But it should be emphasised that although the zero-shot models are currently gaining attention from researchers, the intolerably slow inference speed (the FPS of the optimal zero-shot model was 5.67) makes it difficult to deploy them in agricultural picking robots. Fig. 7 illustrates the training process of Tomato-MDE, and Fig. 8 shows the qualitative results of Tomato-MDE.

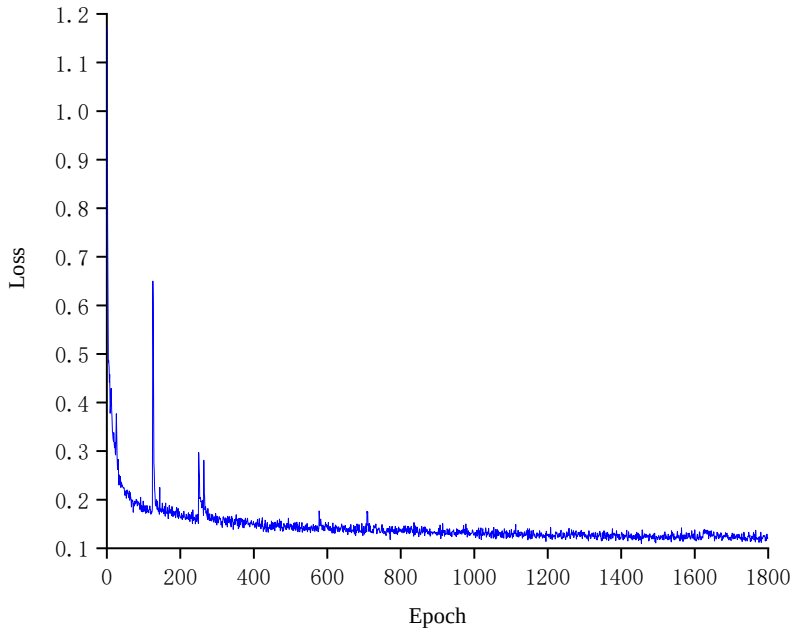

Fig. 7. The training process of the Tomato-MDE.

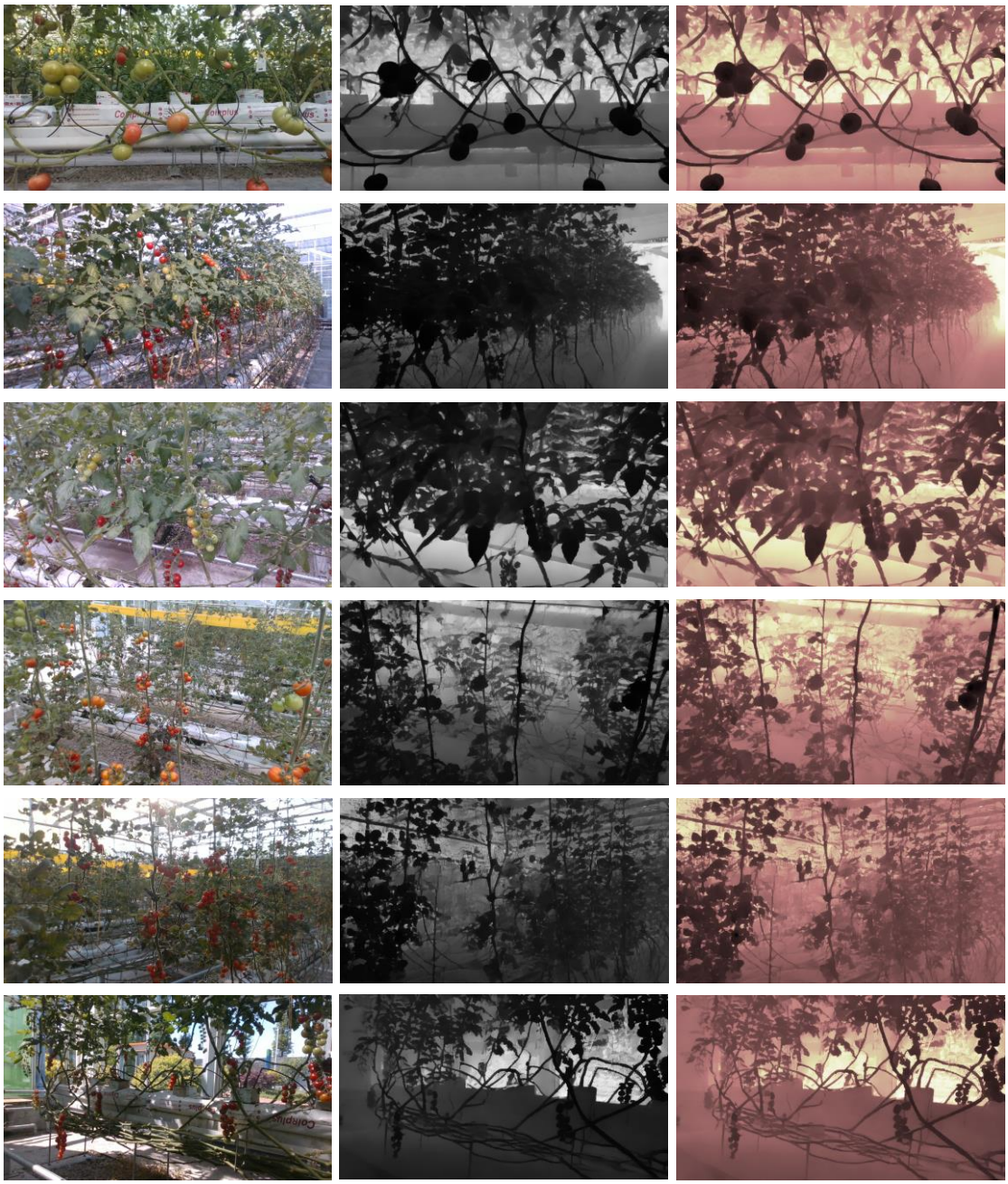

Fig. 8. Qualitative results of our method. The first column represents the original image, the second column shows the estimated depth map, and the third column displays the heatmap of the estimated depth map. The proposed Tomato-MDE model can monocularly estimate depth maps containing complex backgrounds, including tomato plants (globe tomatoes, cherry tomatoes, and cluster tomatoes), glass, planting slots, insect traps, close-ups, and long shots, etc.

4. Discussion

This paper presents a lightweight MDE model, called Tomato-MDE, which is suitable for greenhouse environments containing transparent objects. To achieve the MDE task, the impact of dataset settings on Tomato-MDE was specifically explored in a pilot study. Most MDE models use labeled original real images in their datasets, as researchers typically assume that real images better reflect the data distribution of the real world, offer richer scene coverage, provide more fine-grained and comprehensive supervisory information, and enable deep learning models to efficiently learn depth representations from labeled original real images. However, models trained on labeled original real images show lower prediction accuracy for transparent objects (see Table 2). For tomato picking robotic operations, the presence of transparent glass is unavoidable. Depth datasets are mainly created by obtaining depth data from sensors, stereo matching or structure from motion (Arampatzakis et al., 2023), which leads to imprecise original depth values in the datasets, with significant errors in some cases. Common post-processing methods, such as filtering and threshold filling, cannot solve this issue. Real-world datasets also suffer from this inherent flaw, which leads us to question whether virtual datasets can address this problem. We used the Hypersim and Virtual KITTI virtual datasets to train Tomato-MDE, randomly selecting 10,000, 20,000, 30,000, 40,000, 50,000, 60,000, and 70,000 virtual images as training datasets. The validation and test datasets followed the scheme outlined in Section 3.1.1. The results are demonstrated in Tables 7 and 8, the performance of the proposed model is dramatically improved compared with using only the tomato dataset. But there is still a considerable room for improvement in the model performance.

Table 7. Impact of virtual dataset size on Tomato-MDE.

Virtual dataset size	δ_1	δ_2	δ_3	Abs Rel	RMSE
10000	0.682	0.842	0.897	0.245	0.822
20000	0.695	0.848	0.897	0.238	0.690
30000	0.724	0.884	0.931	0.211	0.773
40000	0.776	0.914	0.947	0.184	0.638
50000	0.807	0.904	0.932	0.175	0.575
60000	0.813	0.911	0.937	0.158	0.526
70000	0.832	0.920	0.944	0.146	0.572

Table 8. Impact of Tomato-MDE trained on a virtual dataset on transparent objects.

Dataset	δ_1	δ_2	δ_3	Abs Rel	RMSE
virtual dataset	0.818	0.887	0.907	0.177	0.450

We repeatedly observed the characteristics of the virtual datasets and found that the style and colour distribution of the virtual datasets are distinctly different from those of the real datasets. The current graphics engine strives to achieve a realistic effect, causing the virtual images to be too clean in terms of colour and too ordered in terms of layout, while the real images contain more randomness. In addition, virtual images have a limited coverage of scenes, they are iteratively sampled from graphics engines with predefined fixed scene types, which are completely different from tomato scenes. Considering the above factors, we directly added the virtual datasets to the tomato dataset for training. As a result, we found that the model's prediction accuracy started to decline. Even worse, the model experienced continuous oscillations during training, as shown in Fig. 9, making it difficult to converge. It is unwise to sacrifice the overall prediction accuracy of the model in order to improve its performance on transparent objects. Furthermore, the oscillations prevent the model from gradually converging to an optimal solution, thus failing to obtain reliable results.

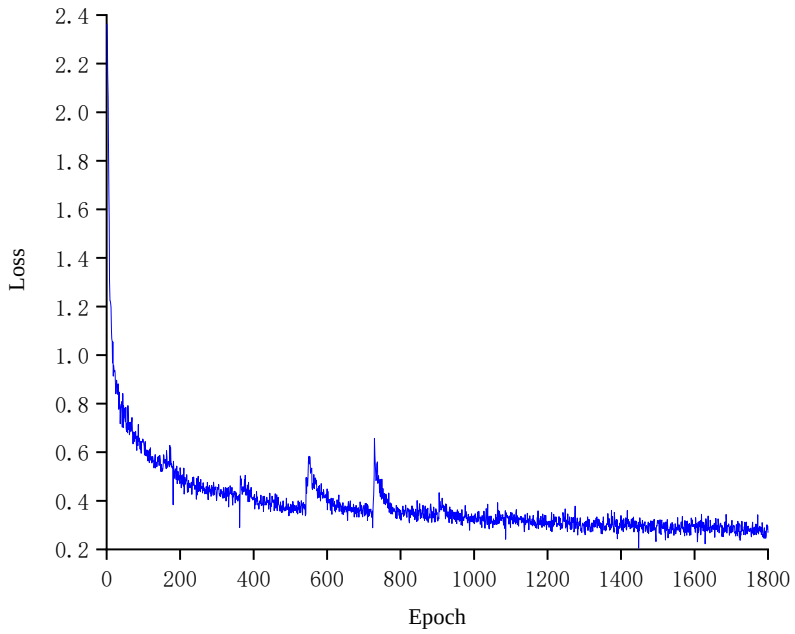

Fig. 9. The training process that directly combined the tomato dataset with the virtual datasets.

This led us to revisit the entire dataset and found that the tomato dataset generally has small depth values, while the virtual dataset has relatively large depth values. Mixing them together for training may cause the model to become overly sensitive to local features in the data, ignoring global patterns and trends, and making it difficult to adapt to data with different scales, displacements, or noise. We therefore used a depth map alignment and normalisation based approach to concatenate the datasets, being able to ignore the unknown scales and displacements of the samples, thus enabling joint training of multiple datasets. However, it is puzzling that the prediction accuracy is still unsatisfactory when using 30,000 virtual datasets. Through further pilot studies, we found that that since the texture and illumination features of the virtual data are different from the real scene, the model needs time to adjust and relearn new depth features, especially in response to the fact that the geometry of the virtual images is completely different from the tomato dataset. In view of this, we kept expanding the virtual dataset and the accuracy starts to pick up slowly, and the model started to learn more generalised depth estimation features

from different scenes and data sources, especially outdoor geometries (e.g., driveways, roads, buildings) in Virtual KITTI and complex indoor layouts (e.g., furniture, walls, etc.) in Hypersim, which enable the model to understand more diverse depth information. We noticed a huge improvement in the prediction ability of the trained Tomato-MDE for transparent objects (see Table 4). Of note, our dataset did not undergo any data enhancement strategies. During the pilot studies, we observed that when the virtual dataset was increased to 80,000, it was difficult to further improve the prediction accuracy of the model. Naturally, we thought of using data augmentation strategies to obtain more powerful visual representation capabilities by learning perturbed semantic constraints. Moreover, using data augmentation strategies can create a more challenging optimisation target for training the model, forcing the model to actively seek additional visual knowledge, which helps the proposed model to more robustly handle the target. Specifically, we introduced colour distortions (Colour Jitter and Gaussian Blur) and spatial distortions (Cutout) (Qi et al., 2022), but did not observe any significant improvement in the prediction accuracy of the proposed model, thus we abandoned the use of the data enhancement strategy.

So far, there are few studies on MDE for agricultural scenarios, and we carefully searched several major databases (Elsevier, SpringerLink, Wiley Online Library, IEEE Xplore, and Google Scholar) for relevant literatures, only nine publications were found, as summarised in Table 9. The model we proposed cannot be directly compared with these monocular depth estimation models in the literatures, because the differences in research objects, test scenarios, and scene complexity, such a comparison would be unfair. However, we can notice that the research on MDE models for agricultural scenarios started in 2021, and so far the scenario complexity in the literatures is low,

and the objects used for depth estimation are typically only one or a few with a single background. In contrast, the background of our tomato dataset is rich, and common objects in greenhouse scenarios are included, such as tomato plants, glass, planting tanks, insect-trap, greenhouse environment control system, etc. It can be seen that the proposed model Tomato-MDE can achieve good performance in real smart greenhouse environments, which is more powerful and practical compared to previous studies.

Table 9. Related works on monocular depth estimation in agriculture.

literature source	research object	performance	test scenario	scenario complexity
(Lu et al., 2021)	plant root system	Variance=9.23	indoor	Simple. single plant root system, single background
(Zhao et al., 2022)	Four plants	estimation errors= 12.948%	field	Simple. Single plant, single background.
(Bellocchio et al., 2022)	mango tree	RMSE= 69.79	field	Normal. Few mango trees, single background
(Coll-Ribes et al., 2023)	table grapes	Abs Rel=2.66	Field, within 1m.	Normal. Few table grapes, single background
(Liu et al., 2024)	watermelon, pear, pomelo	Between 0.8 m and 1.5 m, Abs Rel=0.05	Indoor, square chessboard and round chessboard	Simple. Single fruit, single background
(Kong et al., 2023)	citrus	Between 0.3m and 1.5m，RMSE= 2.05	Field, between 0.3m and 1.5m	Normal. Few citrus, single background
(Divyanth et al., 2023)	apple	Abs Rel=0.035，RMSE= 1.833	field	Normal. Massive apples, single background
(Febriana et al., 2024)	coffee bean	δ_1=0.289，δ_2=0.499，δ_3=0.63，Abs Rel=2.36，RMSE=0.229	indoor	Simple. Single coffee bean, single background
(Magalhaes et al., 2024)	tomato	Abs Rel=0.02	laboratory bench	Simple, few tomatoes, single background

Although Tomato-MDE has achieved promising results, there is still room for further improvement. First, using the virtual dataset can substantially improve the model's prediction accuracy for transparent objects, but there is still significant potential for further performance improvement (δ1=0.818, δ2=0.887, δ3=0.907, Abs Rel=0.177, RMSE=0.45). Solutions to this problem are considered in two ways. One is to further explore virtual datasets, the mechanism of combining real datasets with virtual datasets is not clear at present, and better dataset combination schemes could be sought through extensive experiments. The second is to combine optical simulations (e.g., refractive index, optical properties of light propagation paths and object boundaries) to create a comprehensive loss function for transparent objects, so that the model can

better capture the depth characteristics of these objects, as it is difficult to accurately determine the depth information of an object with simple geometric information (e.g., texture, shape, etc.). In addition, how to further improve the inference speed of Tomato-MDE is an issue worth focusing on. This is because MDE models, in the practical use of agricultural harvesting robots, inevitably need to be combined with object detection models to quickly and accurately determine the 3D spatial position of the target tomatoes. At present, our own object detection model for greenhouse tomatoes, under the NVIDIA Tesla V100 GPU environment, achieves an accuracy of over 90% while maintaining an inference speed of over 200 FPS. However, the inference speed of the proposed MDE model is only 31.62 FPS. A reasonable approach in the design of MDE models is to incorporate pruning (Saleh et al., 2024) and knowledge distillation (Ke et al., 2024). Pruning reduces the complexity and number of parameters of the model by removing unimportant neurons or connections, thereby accelerating inference speed while maintaining performance as much as possible. Knowledge distillation involves transferring the knowledge of a large, high-performance "teacher" model to a smaller "student" model, enabling the student model to closely match the performance of the teacher model while maintaining a lower computational cost. The combination of these two techniques can effectively reduce model inference latency and improve the feasibility of real-time applications. Also, the proposed Tomato-MDE can only perform monocular depth estimation for common greenhouse tomatoes (Globe tomatoes, Cherry tomatoes, and Cluster tomatoes). Developing a separate model for each crop is undoubtedly inefficient and lacks generality. In the Introduction section, we mentioned the zero-shot based approach that emerged in 2020. Zero-shot based MDE models aim to infer depth information of a scene from a single image without relying on traditional training data, and they can adapt to different scenes through

knowledge transfer. This is an emerging and promising direction, representing a potential solution. However, zero-shot methods have very high data requirements. MDE models trained on datasets containing millions of images still struggle to achieve true zero-shot performance. Furthermore, the vast datasets lead to large model sizes, often reaching hundreds of megabytes or even gigabytes, making it difficult to deploy these models in agricultural harvesting robots. In future work, we will explore how to efficiently compress Tomato-MDE without significantly compromising or only slightly compromising its performance, and we will investigate the potential of highly compressing zero-shot-based methods. Additionally, the proposed MDE model, after further compression, will be deployed on our self-developed greenhouse tomato robot, as shown in Fig. 1. By integrating with the object detection algorithm, it will enable fast and accurate tomato picking in the greenhouse, relying solely on an industrial camera.

5. Conclusions

In this paper, we propose a lightweight ViT model, Tomato-MDE, for MDE of greenhouse tomatoes in a complex smart greenhouse environment containing transparent objects. We collected 16,000 tomato images and 80,000 public virtual images as the original dataset. Tomato-MDE achieves δ_1, δ_2, δ_3, Abs Rel and RMSE of 0.883, 0.934, 0.949, 0.107 and 0.371, respectively, with an inference speed of 31.62 FPS, under 4 NVIDIA Tesla V100 GPUs. Compared with the latest CNN-based, zero-shot-based, and ViT-based MDE models, the proposed model attains the best prediction accuracy. Although its inference speed is not the fastest, it is still acceptable for practical deployment. We also found that, compared to the Tomato-MDE model without virtual images, δ_1 increased by 11.21%, Abs Rel decreased by 66.36%, and RMSE decreased by 61.99%, which significantly enhances the model's ability to predict transparent objects. In the future, it can

be further developed as a perception system for agricultural picking robots.

CRediT authorship contribution statement

Chao Qi: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Yawen Cheng:** Writing - review & editing. **Qian Wu:** Supervision, Project administration, Funding acquisition, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they do not have any financial or non-financial conflict of interests.

Acknowledgments

This research was founded by Jiangsu Agriculture Science and Technology Independent Innovation Fund (grant agreement number CX(24)3128).

References

Agarwal A, Arora C. Attention attention everywhere: Monocular depth prediction with skip attention[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023: 5861-5870.

Arampatzakis V, Pavlidis G, Mitianoudis N, et al. Monocular depth estimation: A thorough review[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

Bellocchio E, Crocetti F, Costante G, et al. A novel vision-based weakly supervised framework for autonomous yield estimation in agricultural applications[J]. Engineering Applications of Artificial Intelligence, 2022, 109: 104615.

Bhat S F, Alhashim I, Wonka P. Adabins: Depth estimation using adaptive bins[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 4009-4018.

Bhat S F, Alhashim I, Wonka P. Localbins: Improving depth estimation by learning local

distributions[C]//European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022: 480-496.

Cabon Y, Murray N, Humenberger M. Virtual kitti 2[J]. arXiv preprint arXiv:2001.10773, 2020.

Coll-Ribes G, Torres-Rodríguez I J, Grau A, et al. Accurate detection and depth estimation of table grapes and peduncles for robot harvesting, combining monocular depth estimation and CNN methods[J]. Computers and Electronics in Agriculture, 2023, 215: 108362.

Criminisi A, Reid I, Zisserman A. Single view metrology[J]. International Journal of Computer Vision, 2000, 40: 123-148.

Divyanth L G, Rathore D, Senthilkumar P, et al. Estimating depth from RGB images using deep-learning for robotic applications in apple orchards[J]. Smart Agricultural Technology, 2023, 6: 100345.

Dong Y, Cordonnier J B, Loukas A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth[C]//International Conference on Machine Learning. PMLR, 2021: 2793-2803.

Eigen D, Fergus R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture[C]//Proceedings of the IEEE international conference on computer vision. 2015: 2650-2658.

Febriana A, Farady I, Lin P C, et al. Pseudo-LiDAR Meets Agriculture: Leveraging 3D Monocular Point Cloud Processing for Coffee Beans[C]//2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE). IEEE, 2023: 84-89.

Gasperini S, Morbitzer N, Jung H J, et al. Robust monocular depth estimation under challenging

conditions[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2023: 8177-8186.

Haji-Esmaeili M M, Montazer G. Large-scale monocular depth estimation in the wild[J]. Engineering Applications of Artificial Intelligence, 2024, 127: 107189.

Ke B, Obukhov A, Huang S, et al. Repurposing diffusion-based image generators for monocular depth estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 9492-9502.

Kong D, Wang J, Zhang Q, et al. Research on Fruit Spatial Coordinate Positioning by Combining Improved YOLOv8s and Adaptive Multi-Resolution Model[J]. Agronomy, 2023, 13(8): 2122.

Li T, Sun M, He Q, et al. Tomato recognition and location algorithm based on improved YOLOv5[J]. Computers and Electronics in Agriculture, 2023, 208: 107759.

Li Z, Wang X, Liu X, et al. Binsformer: Revisiting adaptive bins for monocular depth estimation[J]. IEEE Transactions on Image Processing, 2024.

Liu J, Zhou D, Wang Y, et al. A Distance Measurement Approach for Large Fruit Picking with Single Camera[J]. Horticulturae, 2023, 9(5): 537.

Loshchilov I. Decoupled weight decay regularization[J]. arXiv preprint arXiv:1711.05101, 2017.

Niu Z, Zhong G, Yu H. A review on the attention mechanism of deep learning[J]. Neurocomputing, 2021, 452: 48-62.

Patni S, Agarwal A, Arora C. ECoDepth: Effective Conditioning of Diffusion Models for Monocular Depth Estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 28285-28295.

Lu Y, Wang Y, Parikh D, et al. Simultaneous direct depth estimation and synthesis stereo for single

image plant root reconstruction[J]. IEEE Transactions on Image Processing, 2021, 30: 4883-4893.

Magalhaes S A C, dos Santos F N, Moreira A P, et al. MonoVisual3DFilter: 3D tomatoes' localisation with monocular cameras using histogram filters[J]. Robotica, 2024: 1-20.

Piccinelli L, Yang Y H, Sakaridis C, et al. UniDepth: Universal Monocular Metric Depth Estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 10106-10116.

Qi C, Gao J, Pearson S, et al. Tea chrysanthemum detection under unstructured environments using the TC-YOLO model[J]. Expert Systems with Applications, 2022, 193: 116473.

Ranftl R, Bochkovskiy A, Koltun V. Vision transformers for dense prediction[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 12179-12188.

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

Roberts M, Ramapuram J, Ranjan A, et al. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 10912-10922.

Shao S, Pei Z, Chen W, et al. Nddepth: Normal-distance assisted monocular depth estimation[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 7931-7940.

Shao S, Pei Z, Chen W, et al. Urcdc-depth: Uncertainty rectified cross-distillation with cutflip for monocular depth estimation[J]. IEEE Transactions on Multimedia, 2023.

Song M, Lim S, Kim W. Monocular depth estimation using laplacian pyramid-based depth residuals[J]. IEEE transactions on circuits and systems for video technology, 2021, 31(11): 4381-4393.

Thakur A, Venu S, Gurusamy M. An extensive review on agricultural robots with a focus on their perception systems[J]. Computers and Electronics in Agriculture, 2023, 212: 108146.

Saleh S, Hasan R, Battseren B, et al. Optimizing Monocular Depth Estimation for Real-Time Edge Computing Platforms[C]//2024 International Symposium ELMAR. IEEE, 2024: 127-131.

Saxena A, Sun M, Ng A Y. Make3d: Learning 3d scene structure from a single still image[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 31(5): 824-840.

Wang L, Zhang J, Wang O, et al. Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 541-550.

Wofk D, Ma F, Yang T J, et al. Fastdepth: Fast monocular depth estimation on embedded systems[C]//2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 6101-6108.

Yan J, Zhao H, Bu P, et al. Channel-wise attention-based network for self-supervised monocular depth estimation[C]//2021 International Conference on 3D vision (3DV). IEEE, 2021: 464-473.

Yang G, Tang H, Ding M, et al. Transformer-based attention networks for continuous pixel-wise prediction[C]//Proceedings of the IEEE/CVF International Conference on Computer vision. 2021: 16269-16279.

Yang L, Kang B, Huang Z, et al. Depth anything: Unleashing the power of large-scale unlabeled

data[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 10371-10381.

Zhang J, Xie J, Zhang F, et al. Greenhouse tomato detection and pose classification algorithm based on improved YOLOv5[J]. Computers and Electronics in Agriculture, 2024, 216: 108519.

Zhao G, Cai W, Wang Z, et al. Phenotypic parameters estimation of plants using deep learning-based 3-D reconstruction from single RGB image[J]. IEEE Geoscience and Remote Sensing Letters, 2022, 19: 1-5.