Lightweight Monocular Depth Estimation with an Edge Guided Network

Xingshuai Dong[1], Matthew A. Garratt[1], Sreenatha G. Anavatti[1], Hussein A. Abbass[1] and Junyu Dong[2]

***Abstract*— Monocular depth estimation is an important task that can be applied to many robotic applications. Existing methods focus on improving depth estimation accuracy via training increasingly deeper and wider networks, however these suffer from large computational complexity. Recent studies found that edge information are important cues for convolutional neural networks (CNNs) to estimate depth. Inspired by the above observations, we present a novel lightweight Edge Guided Depth Estimation Network (EGD-Net) in this study. In particular, we start out with a lightweight encoder-decoder architecture and embed an edge guidance branch which takes as input image gradients and multi-scale feature maps from the backbone to learn the edge attention features. In order to aggregate the context information and edge attention features, we design a transformer-based feature aggregation module (TRFA). TRFA captures the long-range dependencies between the context information and edge attention features through cross-attention mechanism. We perform extensive experiments on the NYU depth v2 dataset. Experimental results show that the proposed method runs about 96 fps on a Nvidia GTX 1080 GPU whilst achieving the state-of-the-art performance in terms of accuracy.**

I. INTRODUCTION

Depth estimation refers to estimating depth maps from RGB images, and has been widely explored in computer vision and robotics [1]. It is a fundamental perception component of robotic systems. Active sensors such as RGB-D cameras and LiDAR provide accurate depth perception. However, these devices are heavy and require high power consumption that cannot be deployed on resource constrained platforms. In contrast, monocular depth estimation (MDE) is inexpensive, and achievable in compact form factors with high energy efficiency.

Recent developed methods [2]–[6] learn deeper (more layers) and wider (more channels in each layer) models that produce better depth estimation performance. Particularly, these methods aim to improve depth estimation accuracy other than achieve real-time running speed. This makes them difficult to run in edge platforms where reaction time is crucial for operating safety such as for obstacle avoidance in small sized autonomous robots.

In order to solve the problem of running speed on embedded platforms, lightweight CNNs such as ERFNet [7] and MobileNets [8], [9] have been employed to design lightweight MDE networks [10], [11]. Moreover, Wofk et al. [11] applied network pruning technique to further reduce the number of parameters. Although these methods achieve real-time speed on embedded platforms, their accuracy are inferior to state-of-the-art methods.

In addition, the above discussed methods [10]–[12] employ encoder-decoder style network architectures. However the downsampling operations in the encoder network distort fine details in lower resolution layers, which leads to blurry results around object edges. To avoid the loss of spatial information, MobileXNet [13] stacked two relatively shallow encoder-decoder style subnetworks back-to-back in a unified framework. According to [14], [15], the edges in input images are important cue for CNNs to predict depth.

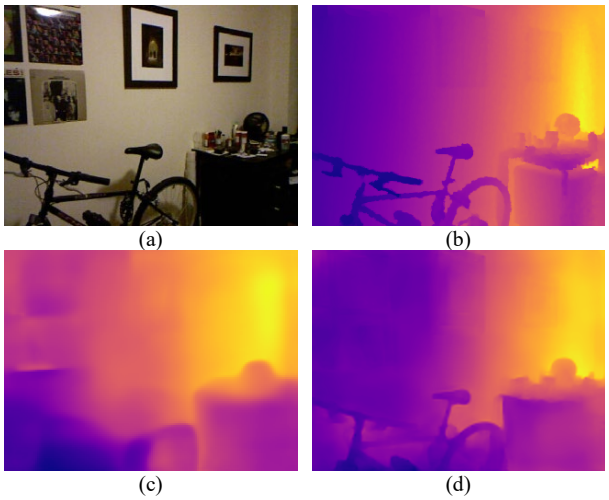


Fig. 1: Example comparison of estimated depth maps. (a) RGB image, (b) Ground-truth depth, (c) Wofk et al. [11] and (d) Our method.

This study aims to address the above problems by designing a novel lightweight network which adopts edge attention features to guide MDE. Specifically, we integrate the depth estimation branch and an edge guidance branch in a unified network, named Edge Guided Depth Estimation Network (EGD-Net). EGD-Net is built on top of a shallow (less layers) and narrow (less channels in each layer) encoder-decoder network and applies edge attention features to guide the depth estimation. The contributions of this study are attributed as follows: (1) we propose a novel lightweight monocular depth estimation network, named EGD-Net, which employs edge attention features to guide the task of depth estimation; (2) we design a channel attention-based feature fusion module; (3) we design a transformer-based feature aggregation module, which captures the long-range dependencies between the edge attention features and the context information; (4) we demonstrate the effectiveness of our designed method through extensive experiments.

II. RELATED WORK

A. Convolutional Neural Networks

Starting from the appearance of AlexNet [16], CNNs have significantly advanced the progress of computer vision. Over the past few years, many CNN architectures such as ResNet [17] and SENet [18] have been designed. The development

[1]Xingshuai Dong, Matthew A. Garratt, Sreenatha G. Anavatti and Hussein A. Abbass are with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2612, Australia

[2]Junyu Dong is with the School of Computer Science and Technology, Ocean University of China, Qingdao 266100, China

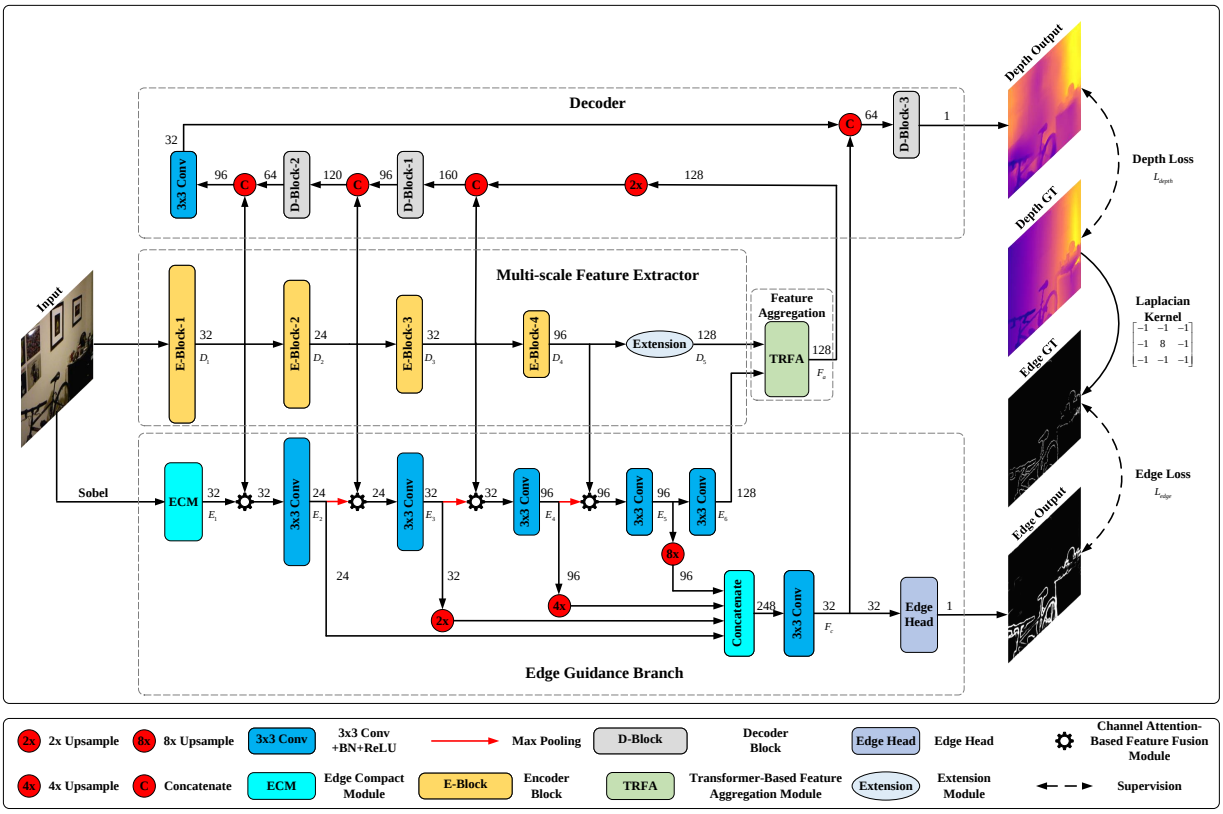


Fig. 2: Illustration of our EGD-Net architecture.

trend is to increase the depth and width of CNNs to enhance the accuracy. However, these advances neglect the need to make networks more efficient with respect to the size and speed required for some real-time applications. Howard et al. [8] designed the first lightweight CNN, MobileNet, for mobile and embedded vision applications. MobileNet is built on top of depthwise separable convolutions, which decompose a regular convolution into a depthwise convolution and a 1×1 pointwise convolution. Later, Sandler et al. [9] extended [8] through a novel efficient convolutional block with inverted residual and linear bottleneck. ShuffleNet [19] applied channel shuffle operators in the channel dimension of feature maps to make cross-group information flow for group convolution layers. Tan et al. [20] developed an EfficientNet-B0 baseline network by using neural architecture search and scaled it up to get a set of models, named EfficientNets.

B. Monocular Depth Estimation

Monocular depth estimation (MDE) is a task that estimates dense depth maps from single RGB images. Motivated by the success of CNNs in image classification, Eigen et al. [21] designed the first MDE network. Later, Laina et al. [2] designed a fully convolutional network which includes an encoder and a decoder. Inspired by [2], Hu et al. [5] combined the encoder-decoder network with a multi-scale feature fusion module and a refinement module. Later, Chen et al. [4] exploited multiple scale scene structure information to estimate depth maps. Ye et al. [6] designed a dual branch network architecture for MDE, named DPNet. DPNet incorporates a spatial branch to retain spatial details and produce high resolution features. The produced high resolution features are fused with features from the contextual branch in a refinement module. Recently, Chang et al. [22] designed a hybrid network consisting of a Transformer-based encoder and a CNN-based decoder to solve the MDE problem. The above discussed methods focus more on depth estimation accuracy and use large backbones or complex architectures to achieve performance increase which lead to increased computational complexity.

Spek et al. [10] designed a lightweight depth estimation network on the basis of "non-bottleneck-1D" block [7]. Although the designed network achieves real-time speed on the Nvidia TX2 GPU, its accuracy is inferior. Wofk et al. [11] introduced a lightweight encoder-decoder network for MDE. Besides, they employed network pruning method to further reduce the amount of parameters. Dong et al. [13] introduced a real-time MDE network which stacks two simple encoder-decoder style network in a unified framework. Existing methods employ encoder-decoder network to aggregate high-level and low-level features to regress depth. Researches in [14], [15] demonstrated that the edges in input images are important cue for CNNs to predict depth. In addition, Fan et al. [23] fused the learned detail information and contextual features to perform semantic segmentation. Inspired by the above described methods, we design a novel lightweight network which integrates an edge guidance branch to produce edge attention features to guide the task of depth estimation.

III. Methodology

This study proposes a novel lightweight monocular depth estimation network, EGD-Net. Fig. 2 illustrates the architecture of EGD-Net, which is composed of four parts: multi-scale feature extractor, edge guidance branch, feature aggregation module and decoder.

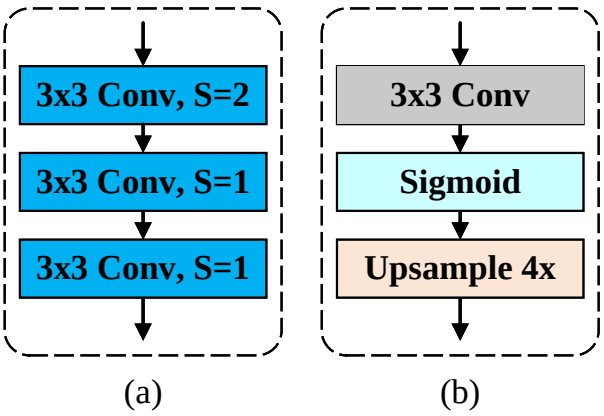


Fig. 3: (a) The edge compact module, where S denotes stride. (b) The edge head, where "3 × 3 Conv" has no batch normalization and ReLU.

A. Multi-scale Feature Extractor

The multi-scale feature extractor (MSFE) consists of a backbone (E-Block-1, 2, 3, 4) and an extension module. We adopt a lightweight CNN, MobileNetV2 [9] as the backbone. To adapt MobileNetV2 to the MDE task, we remove the layers after the fourth convolutional stage. Thus, the final features from the backbone are 1/16 size of the input image. To further enhance the representation ability of produced features, we add an extension module after the backbone. The extension module is comprised of six dilated Inverted Residual Blocks (IRBs) and each IRB has different dilation rate to capture multiple scales context information. Specifically, the dilation rates are set as 1, 2, 3, 1, 2 and 3. Furthermore, the expansion factor and stride values in IRB are set as 4 and 1 respectively. For convenience, we define the output feature maps from the MSFE as D_1, D_2, D_3, D_4, D_5, with strides of $2^1, 2^2, 2^3, 2^4, 2^4$, respectively.

B. Edge Guidance Branch

Edge information are important cues for CNNs to predict depth [14], [15]. To model the edge attention features for guiding the task of depth estimation, we design an edge guidance branch (EGB) which is composed of a few convolutional layers interleaved with channel attention-based feature fusion modules (CAFF). EGB takes image gradients as well as the intermediate features from the MSFE as input and outputs edge images and high resolution feature maps. We first extract image gradients in x and y directions from the input image with the Sobel operator. Image gradients are processed by an edge compact module (Fig. 3 (a)) to generate features (E_1) at $1/2$ size of the input image. The generated features (E_1) are fused with D_1 through the CAFF.

As shown in Fig. 4, intermediate feature maps generated by the MSFE (D_i) and EGB (E_i) are concatenated along the channel dimension. The concatenated features are passed through a 1×1 Conv+BN+ReLU layer to reduce the channel dimension to half size. Next, we pool the reduced features to a feature vector, which is then processed by 1×1 convolutions, ReLU and Sigmoid operators to get an attention map. The attention map is multiplied with D_i and E_i respectively. These multiplied features are added element-wise to build a fused feature. The fused feature is then convolved with

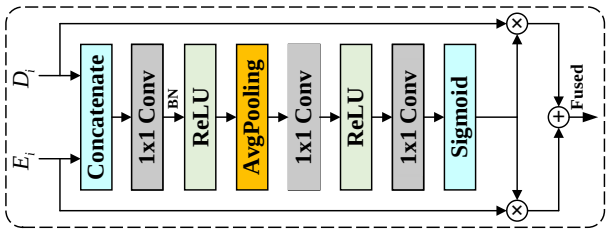


Fig. 4: Illustration of the channel attention-based feature fusion module (CAFF), where "1 × 1 Conv" has no batch normalization and ReLU, "×" and "+" represent multiply and add operations respectively.

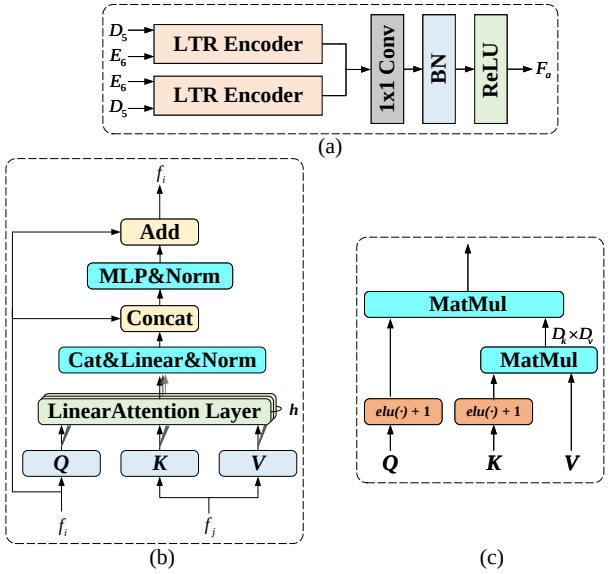


Fig. 5: (a) The transformer-based feature aggregation module (TFAM), where LTR represents linear transformer. (b) LTR encoder layer, h means the multiple heads of attention, which is set as 4 in this study. (c) Linear attention layer.

a 3×3 Conv-BN-ReLU layer. We denote the feature maps generated by the EGB as $E_1, E_2, E_3, E_4, E_5, E_6$, with strides of $2^1, 2^1, 2^2, 2^3, 2^4, 2^4$, respectively.

Feature maps from EGB (E_3, E_4, E_5) are resized to 1/2 size of the input image and concatenated with E_2 and then pass through a 3×3 Conv-BN-ReLU layer. As shown in Fig. 2, the output feature (F_c) from the 3×3 Conv-BN-ReLU layer is used as the input of two directions. The first direction is the edge head, which produces edge maps for supervision. The second direction is passed to the decoder to act as the edge guidance feature. In addition, E_5 is convolved with a 3×3 Conv-BN-ReLU layer and generates E_6, which is fed to the TRFA module to aggregate with the context rich feature from the MSFE. In this study, the edge detection is modeled as a binary segmentation task. To obtain the ground-truth binary edge images, we adopt the Laplacian operator to extract edge maps from the ground-truth depth maps. The edge guidance branch is directly supervised by the binary edge labels. Thus, it learns edge attention features.

C. Transformer-Based Feature Aggregation Module

In order to combine the edge attention features from the EGB and context rich features from the MSFE to produce

high resolution depth maps, we design a transformer-based feature aggregation module (TRFA). TRFA consists of two linear transformer encoder layers [24] and a 1×1 Conv-BN-ReLU layer. The core element of the linear transformer encoder layer is that the linear attention layer computes the attention between a set of query vectors (Q) and key vectors (K) using dot-product similarity, which is then used to weigh a set of value vectors (V). Thus, the attention computation selects the relevant information through measuring the similarity between the query vector and each key vector.

Inspired by [25], we adopt the linear transformer encoder layer to capture the long-range dependencies (or global context) between the edge and context features through cross-attention in two directions. As shown in Fig. 5, the input features to linear transformer encoders are (D_5, E_6) and (E_6, D_5) respectively. Then, features from the linear transformer encoder layers are concatenated and passed through the 1×1 Conv-BN-ReLU layer to aggregate them together.

The output features from TRFA are first upsampled to two times in size and then fed to the decoder. The decoder is composed of a 3×3 Conv-BN-ReLU layer and three decoder blocks (D-Block-1, 2, 3), each decoder block includes a 3×3 Conv-BN-ReLU layer and a bilinear interpolation with a scale factor of 2. The decoder incorporates features (F_a) from the TRFA, low-level features (D_1, D_2, D_3) from the MSFE, and the high resolution features (F_c) from the EGB to produce depth maps with the same size as input images.

D. Loss Function

The designed network includes two branches that output depth maps and edge maps respectively. For the task of monocular depth estimation, we adopt the loss function proposed in [13]. This loss stacks the regular L_1 loss: $L_1(d, d^*) = \frac{1}{N}\sum_i^N |d_i - d_i^*|$ and the image gradient based L_1 loss: $L_{grad}(d, d^*) = \frac{1}{N}\sum_i^N |\bigtriangledown_x(d_i, d_i^*)| + |\bigtriangledown_y(d_i, d_i^*)|$, where N is the total number of pixels being considered, d and d^* are the predicted and ground-truth depth, $\bigtriangledown_x$ and $\bigtriangledown_y$ are the spatial derivatives in x and y directions. The depth estimation loss can be written as: $L_{depth} = L_1 + L_{grad}$. For edge detection, we employ the standard binary cross-entropy (BCE) loss L_{edge}, which is defined as:$L_{edge} = -\sum_i (e_i^* log e_i + (1 - e_i^*) log(1 - e_i))$, where e_i and e_i^* are the detected and ground-truth edges. Finally, the whole loss function is formulated as: $L = \lambda_1 L_{depth} + \lambda_2 L_{edge}$, where λ_1 and λ_2 are the hyper-parameters, we empirically set λ_1 = 1 and λ_2 = 20.

IV. Experiments

To demonstrate the effectiveness of our proposed EGD-Net, we evaluate it on the NYU depth v2 dataset [26].

A. Implementation Details

The designed network is implemented in PyTorch. A workstation with a single Nvidia RTX 3090 GPU is used for training and testing. The weights of the backbone of the MSFE are initialized with the weights pre-trained on ImageNet. The other layers are randomly initialized. The training is optimized by using the SGD optimizer, and the batch size is set as 8. We train the network for 25 epochs. The poly learning rate policy is adopted, the learning rate for the n^{th} epoch is $init\_lr \times (1 - \frac{n}{max\_epoch})^{power}$, where the $init\_lr$ and power are set as 0.01 and 0.9 respectively.

During training, we employ data augmentation approaches to increase the diversity of training samples. Data augmentations are applied to each RGB and ground-truth depth image pair in an online fashion:

- Random Flips: RGB and ground-truth depth image pairs are horizontally flipped at a probability of 0.5.
- Random Rotation: RGB and ground-truth depth image pairs are randomly rotated by a degree of $r \in [-5, 5]$.
- Color Jitter: the brightness, contrast and saturation values of the RGB images are randomly scaled by a factor of $c \in [0.6, 1.4]$.

B. Dataset and Evaluation Metric

We evaluate the proposed method on the commonly used NYU depth v2 dataset [26], which was collected in real-world indoor surroundings with a Microsoft Kinect camera. The original images have a resolution of 640×480 pixels. In this work, we train our method on the training set proposed by Hu et al. [5] and evaluate it on the offical testing set including 654 RGB and depth image pairs. Each image pair is downsampled to 342×256 and then center cropped to 320×240. We first compare our proposed method with state-of-the-art methods and then perform ablation experiments to validate the contribution of each component of the proposed network.

We adopt three widely used metrics to evaluate the proposed method. Since our aim is to estimate depth maps from RGB images, only the error and accuracy metrics of depth estimation are compared. Let N denotes the total number of valid pixels, d_i and d_i^* represent the estimated and ground-truth depth values at the pixel indexed by i, respectively. The error metrics are defined as follows:

- Root Mean Square Error (RMSE): $\sqrt{\frac{1}{N}\sum_i^N |d_i - d_i^*|^2}$.
- Mean Relative Error (REL): $\frac{1}{N}\sum_i^N \frac{|d_i - d_i^*|}{d_i}$.
- δ_i Accuracy: % of d_i s.t. $max(\frac{d^*}{d}, \frac{d}{d^*}) < \delta_i, \delta_i = 1.25^i$.

C. Comparison with State-of-the-art

In this subsection, we compare the performance of our EGD-Net with state-of-the-art methods [4], [5], [11]–[13], [27] in terms of the amount of network parameters (Params., million), error (RMSE and REL) and accuracy (δ_1, δ_2 and δ_3) metrics. Quantitative results of our proposed method and state-of-the-art methods are listed in Table I. For [4], [5] we report the corresponding results from their papers. The results of [12], [27] are reported in [12]. We retrained [11], [13] with the same training and testing procedure as described in Section IV-A. To compare fairly, [11], [13] are supervised by the depth loss described in Section III-D.

It can be observed that: (1) among all methods, EGD-Net has the lowest amount of parameters. In particular, EGD-Net has $>2.5\times$ fewer parameters than [12], [27], $>11\times$ fewer parameters than MobileXNet [13], $>71\times$ fewer parameters than Hu et al. [5], and $>95\times$ fewer than Chen et al. [4]; (2) with much fewer network parameters, EGD-Net generates the best RMSE performance while its δ_2 and δ_3 metrics are very close to Hu et al. [5] and Chen et al. [4]; (3) EGD-Net outperforms [11]–[13], [27] in terms of all error and accuracy metrics.

TABLE I: Comparison of performances on the NYU depth v2 dataset [26]. ↑ means higher is better, ↓ means lower is better. The red and **bold** values indicate the best results.

Method	Backbone	Params. ↓	RMSE ↓	REL ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Hu et al. [5]	SENet-154	157.0 M	0.530	0.115	0.866	0.975	0.993
Chen et al. [4]	SENet-154	210.3 M	0.514	**0.111**	**0.878**	**0.977**	**0.994**
Wofk et al. [11]	MobileNet	20.67 M	0.529	0.155	0.789	0.950	0.987
Tu et al. [27]	MobileNetV2	5.7 M	0.531	0.147	0.801	0.956	0.989
Rudolph et al. [12]	DDRNet-23-slim	5.8 M	0.501	0.138	0.823	0.961	0.990
MobileXNet [13]	MobileNet	24.95 M	0.507	0.149	0.807	0.953	0.989
Ours	MobileNetV2	**2.21 M**	**0.486**	0.136	0.825	0.960	0.990

TABLE II: Ablation study on different feature aggregation methods. ↑ means higher is better, ↓ means lower is better. "w CAFFM" means the EGB includes the CAFF module. "w/o CAFFM" indicates the EGB does not include the CAFF module. The red and **bold** values indicate the best results.

Method	Params. ↓	RMSE ↓	REL ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Baseline	**1.56 M**	0.520	0.149	0.796	0.955	0.988
Baseline + EGB	2.01 M	0.514	0.149	0.808	0.959	**0.990**
Baseline + TAFM	2.19 M	0.506	0.144	0.810	0.955	0.986
Baseline + EGB (w CAFF) + TAFM	2.21 M	**0.486**	**0.136**	**0.825**	**0.960**	**0.990**
Baseline + EGB (w/o CAFF) + TAFM	2.16 M	0.489	0.144	0.817	**0.960**	0.989

TABLE III: Ablation study on different backbones. ↑ means higher is better, ↓ means lower is better. The red and **bold** values indicate the best results.

Backbone	Params. ↓	RMSE ↓	REL ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
ResNet-18 [17]	6.43 M	0.489	0.144	0.818	**0.960**	**0.990**
EffcientNet-B0 [20]	2.63 M	0.555	0.161	0.771	0947	0.988
ShuffleNetV2 [19]	3.37 M	0.514	0.155	0.798	0.949	0.986
MobileNetV2 [9]	**2.21 M**	**0.486**	**0.136**	**0.825**	**0.960**	**0.990**

Regarding the running speed, EGD-Net runs about 96 fps (GPU reference time is 10.4 ms) on a Nvidia GTX 1080 GPU (2580 CUDA cores and 8GB memory), which is adequate for real-time robotic applications. We present the qualitative comparison of our results with Wofk et al. [11] and MobileXNet [13] in Fig. 6. It can be observed that our proposed method can predict finely detailed object boundaries while [11], [13] cannot predict clearly.

D. Ablation Experiments and Analyses

To analyze the contribution of each component of the designed network, we perform experiments with different deployments on the NYU Depth v2 dataset [26]. The training and testing strategies are kept the same as Section IV-A.

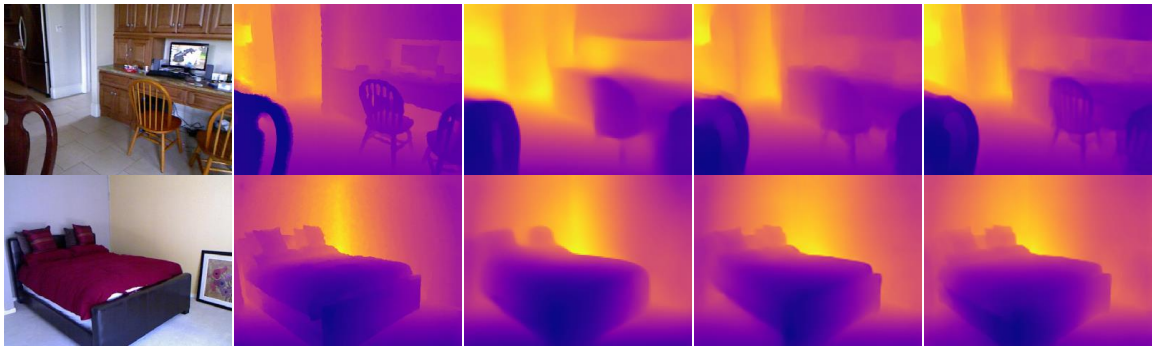
1) Contribution of Different Components: We setup a baseline network that is consisted of the multi-scale feature extractor and the decoder shown in Fig. 2. The baseline method is trained with the depth loss described in Section III-D. The edge guidance branch and transformer-based feature aggregation module are added to the baseline step by step.

As show in Table II, the baseline has the lowest amount of parameters, while it yields the worst results. When we combine the proposed EGB with the baseline, it outperforms the baseline in terms of RMSE, δ_1, δ_2 and δ_3. To explore the influence of the proposed TRFA, we append it to the baseline network and train it with the depth loss. With the same training procedure, it outperforms the baseline and the combination of "Baseline + EGB". Finally, our whole EGD-Net (line 4, Table II), with both EGB and TRFA and trained with the whole loss function, yields the best performance in terms of all metrics. To evaluate the contribution of the proposed CAFF, we design a variant by replacing CAFF with pixel-wise addition (Baseline + EGB (w/o CAFF) + TAFM). According to the last two lines, the CAFF improves the performance of EGD-Net in term of all error and accuracy metrics.

2) Comparison of Different Backbones: In this subsection, we investigate the influence of adopting different backbones in the MSFE. We compare MobileNetV2 [9] with three CNNs, ResNet-18 [17], EfficientNet-B0 [20] and ShuffleNetV2 [19]. Specifically, ResNet-18 [17] is a general CNN, ShuffleNetV2 [19] and EfficientNet-B0 [20] are lightweight CNNs. The weights of all backbones are initialized from the pre-trained models on ImageNet. To make the backbones compatible with the fixed feature aggregation module, we fixed the channel dimension of the final feature maps from the MSFE (D_5) and EGB (E_6) to 128. We report the results of four backbones in Table III. As can be observed, when using MobileNetV2 [9] as backbone the proposed method achieves the best trade-off between accuracy and computation complexity. In particular, it has the lowest amount of parameters and yields the best performance in terms of both error and accuracy metrics.

V. Conclusion

In this paper, we introduced a novel lightweight monocular depth estimation network, named EGD-Net. Specifically, we designed an Edge Guidance Branch to detect edges and produce edge attention features that contain edge information. Moreover, a transformer-based feature aggregation module has been designed to learn the long-range dependencies between the edge and context features and aggregate them together. Extensive experiments on the NYU depth v2 dataset demonstrated the effectiveness of our proposed network. In future work, we will extend our approach to real-world and simulated outdoor environments.



(a) (b) (c) (d) (e)

Fig. 6: Qualitative results from the NYU depth v2 dataset. (a) RGB image, (b) Ground-truth depth, (c) Wofk et al. [11], (d) MobileXNet [13] and (e) Our results. Color represents depth (yellow is far, blue is close).

REFERENCES

[1] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, "Towards real-time monocular depth estimation for robotics: A survey," *IEEE TITS*, 2022.

[2] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3DV*, 2016, pp. 239–248.

[3] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[4] X. Chen, X. Chen, and Z.-J. Zha, "Structure-aware residual pyramid network for monocular depth estimation," *arXiv preprint arXiv:1907.06023*, 2019.

[5] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *WACV*, 2019, pp. 1043–1051.

[6] X. Ye, S. Chen, and R. Xu, "DPNet: Detail-preserving network for high quality monocular depth estimation," *PR*, vol. 109, p. 107578, 2021.

[7] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE TITS*, vol. 19, no. 1, pp. 263–272, 2017.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[10] A. Spek, T. Dharmasiri, and T. Drummond, "CReaM: Condensed real-time models for depth prediction using convolutional neural networks," in *IROS*, 2018, pp. 540–547.

[11] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "FastDepth: Fast monocular depth estimation on embedded systems," in *ICRA*, 2019, pp. 6101–6108.

[12] M. Rudolph, Y. Dawoud, R. Güldenring, L. Nalpantidis, and V. Belagiannis, "Lightweight monocular depth estimation through guided decoding," *arXiv preprint arXiv:2203.04206*, 2022.

[13] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, "MobileXNet: An efficient convolutional neural network for monocular depth estimation," *arXiv preprint arXiv:2111.12334*, 2021.

[14] T. v. Dijk and G. d. Croon, "How do neural networks see depth in single images?" in *ICCV*, 2019, pp. 2183–2191.

[15] J. Hu, Y. Zhang, and T. Okatani, "Visualization of convolutional neural networks for monocular depth estimation," in *ICCV*, 2019, pp. 3869–3878.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImagenNet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[18] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[19] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *ECCV*, 2018, pp. 116–131.

[20] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019, pp. 6105–6114.

[21] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, 2014, pp. 2366–2374.

[22] W. Chang, Y. Zhang, and Z. Xiong, "Transformer-based monocular depth estimation with attention supervision," in *BMVC*, 2021.

[23] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *CVPR*, 2021, pp. 9716–9725.

[24] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *ICML*, 2020, pp. 5156–5165.

[25] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: Detector-free local feature matching with transformers," in *CVPR*, 2021, pp. 8922–8931.

[26] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012, pp. 746–760.

[27] X. Tu, C. Xu, S. Liu, R. Li, G. Xie, J. Huang, and L. T. Yang, "Efficient monocular depth estimation for edge devices in internet of things," *IEEE TII*, vol. 17, no. 4, pp. 2821–2832, 2020.