A Multi-Feature-Based Multi-Task Loss Approach for Enhanced Monocular Depth Estimation

YeaShuan Huang
CSIE
Chung-Hua University
Hsinchu, Taiwan
yeashuan@chu.edu.tw

Yuxiang Chen
CSIE
Chung-Hua University
Hsinchu, Taiwan
tp655998@gmail.com

Chang Wu Yu
CSIE
Chung-Hua University
Hsinchu, Taiwan
cwyu@chu.edu.tw

Abstract—This paper proposes a novel approach for monocular depth estimation that leverages deep networks and employs three types of features, namely depth map, point cloud, and virtual normal, to compute a multi-task loss function. Our experimental evaluations employ one cutting-edge monocular depth estimation model with distinct convolutional network architectures, trained and tested on a public indoor depth estimation NYU dataset. The experimental results demonstrate the consistent improvement in accuracy of depth estimation achieved by minimizing the proposed multi-task loss function, thus validating its effectiveness.

Keywords—depth estimation, deep learning, multi-task loss function

I. INTRODUCTION

Depth estimation has long been recognized as a crucial computer vision technology that enhances the perception and understanding of real 3D scenes, with potential applications in robot navigation, autopilot, and virtual reality [1,2,3]. Historically, researchers have estimated depth maps based on depth cues such as vanishing points [4], focus and defocus [5], and shadows [6]. Binocular camera depth estimation methods typically involve stereo matching and triangulation to obtain the disparity between two 2D images captured by binocular cameras [7], followed by calculation of the depth map. However, binocular depth estimation methods require two fixed cameras to capture sufficient features for matching in the image and can be challenging in scenes with low or no texture. As a result, researchers have shifted their focus to monocular depth estimation, which only requires a single camera to capture images or video sequences, without the need for additional complex equipment and specialized calibration techniques, making it highly applicable in various scenarios. However, due to the lack of reliable stereo vision relationships in monocular images, depth regression estimation is inherently ill-posed [8], and obtaining accurate depth estimation is challenging, necessitating continuous research and development efforts.

In recent times, there has been significant progress in the integration of artificial intelligence (AI) into smart vehicles, which are defined as movable vehicles equipped with sensory and intelligent capabilities. Similar to human perception through visual cues, smart vehicles utilize cameras to perceive the internal and external environment of the vehicle, understand road conditions, and driving status, thereby improving driving safety. The intelligent features provided by cameras for vehicles include lane detection, pedestrian detection, vehicle detection, collision prevention, and road-sign recognition, among others. The reliability of intelligent assisted driving in vehicles can be enhanced through big data analysis and deep learning, leading to advancements in autonomous driving technology and the early maturity of the self-driving car industry. This paper focuses on investigating monocular depth estimation using image processing and deep learning techniques to predict pixel-level depth values from a single two-dimensional image. The combination of this technology with mature object detection techniques in the future has the potential to provide safe and reliable driving services for the self-driving car industry, such as collision prevention, speed recognition, and blind spot detection.

II. LITERATURE REVIEW

Eigen et al. [9] first proposed a coarse-to-fine framework in 2014, where a coarse network learns the global depth of the entire image to obtain a rough depth map, and a fine network learns local features to refine the depth map. Essentially, the framework for monocular depth estimation with deep learning consists of an encoder-decoder network, where the input is an RGB image and the output is a depth map. In general, Monocular depth estimation methods can be attributed into three categories: supervised, unsupervised, and semi-supervised training approaches.

The supervised monocular depth estimation, as described in [10], estimates depth by learning scene structural information from the ground-truth (GT) depth map. However, obtaining GT depth maps is costly, and therefore, some monocular depth estimation networks are trained with fewer GT or without GT, resulting in unsupervised and semi-supervised learning methods. Supervised learning, as reported in [11], has the highest estimation accuracy, but it heavily relies on GT depth maps. On the other hand, unsupervised monocular depth estimation is usually trained with stereo pair-wise images [12] or monocular image sequences [13], and tested on monocular images or sequences, which are trained with scene geometric constraints. Semi-supervised learning methods rely on auxiliary information such as virtual data [14], sparse data [15], and surface normal [16], which are easier to obtain than GT depth maps.

III. THE PROPOSED METHOD

This paper proposes a monocular depth estimation method with a semi-supervised multi-task loss. This loss utilizes three kinds of features: depth, point cloud, and virtual normal, to calculate a multi-task loss function. The depth loss evaluates the difference between the true and estimated depth values, while the point cloud loss evaluates the difference between the true and estimated point cloud coordinates, and the virtual normal loss measures the difference between real and estimated virtual normal vectors. In the following section, we will present the overall structure of the proposed method,

depth estimation, point cloud transformation, virtual normal calculation, and the proposed multi-task loss function.

A. Overall structure

Figure 1 illustrates the overall structure of the proposed monocular depth estimation method. It resizes an input image to half its original size and uses a depth estimation network, Adabins [17], to estimate depth. The depth map is then used to compute 3D point cloud coordinates, which are further used to obtain pixel-level surface normals. Prior to network training, the point cloud coordinates and virtual normal vectors for each training image are computed from the depth map. During network training, a multi-task loss function, incorporating three kinds of features: depth map, point cloud, and virtual normal, is calculated. The network parameters are then trained to minimize this multi-task loss function. After the training process is completed, an input test image will be fed into the depth estimation network, which then can produce an estimated depth map as its output.

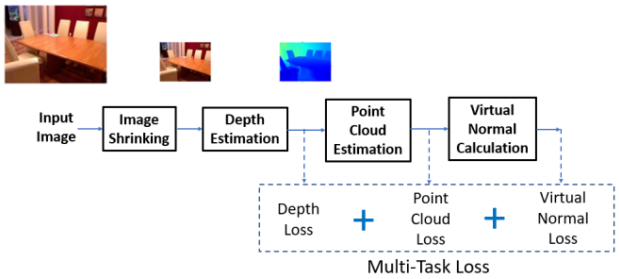

Fig. 1. The overall structure of the proposed method

B. Adabins depth estimation deep network

Adabins [17] emphasizes the importance of global feature processing at high resolution, and it passes the decoder's high-resolution features through mViT (Mini Vision Transformer) to extract a more expansive and holistic range of global information. Adabins employs a classification-based approach to estimate the precise depth value by predicting the corresponding depth interval (bins) for each input image scene. This allows for a more accurate regression of the depth information. Fig. 2 shows the architecture of Adabins, which consists of two primary components: a standard encoder-decoder and an Adabins module. The first component employs an EfficientNet-B5 [18] as the encoder to extract features from the input image. These features are then progressively convolved by the decoder to reconstruct an initial depth map. The second component involves processing the high-resolution features generated by the decoder through mViT, and yields two distinct outputs: the range-attention map (R) with dimensions $h \times w \times C$, and the bin widths (b), represented as an N-dimensional vector, where N corresponds to the total number of depth intervals. To obtain the final depth map, a hybrid regression approach consists of four steps that leverages both R and b. Firstly, the channel number of R is transformed to N through a 1×1 convolution. Secondly, a softmax operation is applied to generate N probability scores $(p_1, p_2, \dots, p_N)$, where p_i is the probability of each pixel belonging to the i-th depth interval. Thirdly, the centers of N depth intervals, denoted $c(b_1),\ c(b_2), \dots, c(b_N)$, are calculated using the following formula:

$$c(b_i) = d_{min} + (d_{max} - d_{min})\left(\frac{b_i}{2} + \sum_j^{i-1} b_j\right) \quad (1)$$

where, d_{min} represents the minimum depth value, d_{max} represents the maximum depth value, and b_i represents the width of the i-th depth interval. Finally, the final estimated depth value, $\tilde{d}$, is computed by combining the Softmax scores $\{p_1, p_2, \dots, p_N\}$ and the depth bin centers $\{c(b_1), c(b_2), \dots, c(b_N)\}$ in a linear manner as

$$\tilde{d} = \sum_{k=1}^{N} c(b_k) p_k \quad (2)$$

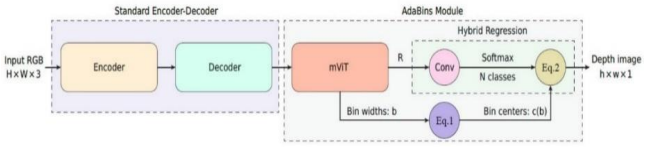

Fig. 2. Adabins comprises a standard encoder-decoder and an AdaBins module, and it takes an RGB color image as input and produces a depth map of pixel-level resolution as output.

C. Features

This subsections briefly describes the applied two kinds of features, i.e. point cloud and virtual normal.

- Point cloud

A point cloud refers to a collection of data points that represent the spatial attributes of each image pixel, including its depth value d, pixel coordinate (u,v), and camera coordinate (X,Y,Z). To obtain accurate spatial information from a depth map, a two-step process is required. Firstly, a pixel coordinate is transformed into an image coordinate. Secondly, an image coordinate is further transformed into a camera coordinate. The pixel coordinate system takes the upper-left corner of the image as the origin, while the image coordinate system takes the center of the image as the origin. Assuming (u,v) is the pixel coordinate of a pixel P in the image, (x,y) is the corresponding image coordinate of P, and (u_0,v_0) is the pixel coordinate of the center of the image, then

$$\begin{cases} x = u - u_0 \\ y = v - v_0 \end{cases} \quad (3)$$

Suppose f_x and f_y are the horizontal and vertical focal lengths of the camera in pixel units, and d is the depth value, which is the actual distance between the camera and the target point. The transformation formula between image coordinates and camera coordinates is as follows:

$$\begin{cases} X = \frac{x}{f_x} Z \\ Y = \frac{y}{f_y} Z \\ Z = \frac{d}{\sqrt{(\frac{x}{f_x})^2 + (\frac{y}{f_y})^2 + 1}} \end{cases} \quad (4)$$

By utilizing equations (3) and (4), the transformation from a given pixel depth (with known values of u, v, and d) to its corresponding point cloud can be expressed as

$$\begin{cases} X = \frac{Z(u-u_0)}{f_x} \\ Y = \frac{Z(v-v_0)}{f_y} \\ Z = \frac{d}{\sqrt{(\frac{u-u_0}{f_x})^2 + (\frac{v-v_0}{f_y})^2 + 1}} \end{cases} \quad (5)$$

- Virtual normal

A virtual normal vector represents the normal vector of a virtual surface of one pixel and can be constructed from a depth map. A virtual normal (*VN*) loss function was proposed in [19] to conditionally select *N* groups of points (each group consisting of three point cloud coordinates *A*, *B*, and *C*) randomly from the point cloud transformed from the depth map to calculate the error of the virtual normal vectors. A *VN* loss provides a remote three-dimensional geometric constraint for the depth-estimation model. The selected *N* groups of points,$\{(P_A^k, P_B^k, P_C^k)|k = 1 \dots N\}$, are subject to the following two constraints:

1. Non-collinear: The angle between any two vectors in a triangle must be between $30^\circ$and 120°, that is
$$30^\circ \le \angle\left(\overrightarrow{P_A^k P_B^k}, \overrightarrow{P_A^k P_C^k}\right) \le 120^\circ,$$
$$30^\circ \le \angle\left(\overrightarrow{P_B^k P_C^k}, \overrightarrow{P_B^k P_C^k}\right) \le 120^\circ$$
$$30^\circ \le \angle\left(\overrightarrow{P_C^k P_B^k}, \overrightarrow{P_C^k P_A^k}\right) \le 120^\circ$$
2. Long distance: The distance between any two points in a triangle must be greater than 0.6 meters, that is
$$\left\|\overline{P_A^k P_B^k}\right\| > 0.6\,, \left\|\overline{P_A^k P_C^k}\right\| > 0.6 \text{ and } \left\|\overline{P_B^k P_C^k}\right\| > 0.6$$

The formula for calculating the normal vector of the *k*-th group of points is:

$$n_k = \frac{\overrightarrow{P_A^k P_B^k} \times \overrightarrow{P_A^k P_C^k}}{||\overrightarrow{P_A^k P_B^k} \times \overrightarrow{P_A^k P_C^k}||} \dots\dots\dots (k = 1, \dots, N) \quad (6)$$

D. Multi-task Loss Function

One of the main contribution of this paper is to propose a novel multi-task loss function$L_{MultiTask}$ with three different losses (depth loss, point cloud loss, and virtual normal loss). The depth loss L_{Depth} calculates the difference between the true and estimated depth values, the point cloud loss $L_{PointCloud}$ calculates the difference between the true and estimated point cloud coordinates, and the virtual normal loss $L_{VirtualNormal}$ calculates the vector difference between the true and estimated virtual normals. $L_{MultiTask}$ is calculated as follows:

$$L_{MultiTask} = \alpha_1 \times L_{Depth} + \alpha_2 \times L_{CloudPoint} + \alpha_3 \times L_{VirtualNormal} \quad (7)$$

here, α_1, α_2 and α_3 are the weights of the depth loss, point cloud loss, and virtual normal loss, respectively, which adjust the contribution of the three losses in the multi-task loss function.

- Depth Loss (L_{Depth})

Let d_i and d_i' be respectively the true and estimated depth values of the *i*-th pixel in an image, and *T* be the total number of valid depth point in this image[1], then

$$L_{Depth} = \frac{1}{T}\sum_{i=1}^{T} \frac{|d_i - d_i'|}{d_i} \quad (8)$$

3.5.2 Point Cloud Loss ($L_{PointCloud}$)

According to Eq. (5), the corresponding pixel coordinate (*u*,*v*) can be reconstructed by a point cloud coordinate (*X*, *Y*, *Z*) as

$$\begin{cases} u = \frac{Xf_x}{Z} + u_0 \\ v = \frac{Yf_y}{Z} + v_0 \end{cases} \quad (9)$$

Let (*u*, *v*) be the pixel coordinate of an image pixel, (*X*, *Y*, *Z*) and (X', Y', Z') be its corresponding point cloud coordinates derived respectively from the ground-truth and estimated depth maps. To calculate $L_{PointCloud}$, *X* and *Y* are deliberately paired with Z', and a distinct pixel coordinate (u',v') can be derived from the specified point cloud coordinate (*X*, *Y*, Z') as

$$\begin{cases} u' = \frac{Xf_x}{Z'} + u_0 \\ v' = \frac{Yf_y}{Z'} + v_0 \end{cases} \quad (10)$$

Conceptually, if the estimated depth d' is close to the ground-truth depth *d*, then *Z* and Z' will be close too, leading to (*u*,*v*) and (u',v') being close. However, if the difference between d' and *d* is large, then *Z* and Z' will also differ a lot, causing (*u*,*v*) and (u',v') to differ greatly. Therefore, the proposed $L_{PointCloud}$ provides a consistency constraint between the depth map and the point cloud for the depth estimation model, and the resulting pixel coordinates can be used to calculate the point cloud loss. As a result, $\sqrt{(u-u')^2 + (v-v')^2}$ can be utilized in calculating and provides a consistency constraint between the depth map and the point cloud for the depth estimation model. However, Equation (9) denotes the same amount of deviation ($Z - Z'$) in the optical axis direction, both deviation ($u - u'$) and ($v - v'$) will increase proportionally with the values of *X* and *Y*. It means the pixels away from the image center are prone to have larger deviation for both $u - u'$and $v - v'$ than those near the image center. Hence, $L_{PointCloud}$ should be normalized as $\sqrt{\left(\frac{u_i - u_i'}{u_i - u_0}\right)^2 + \left(\frac{v_i - v_i'}{v_i - v_0}\right)^2}$. Furthermore, to avoid the abnormality of dividing by zero, the pixels with a u_0 horizontal coordinate or a v_0 vertical coordinate should have a different normalization design. Let A be the set of points with both non u_0 horizontal coordinates and non v_0 vertical coordinates, *B* be the set of points with u_0 horizontal coordinates and non v_0 vertical coordinates, and *C* be the set of points with non

[1] It is related to the device, such as using lidar to obtain depth values. In certain areas such as sky and the dark regions that absorb light completely, distance values cannot be acquired. Consequently, these areas generate missing values, rendering them invalid points.

u_0 horizontal coordinates and v_0 vertical coordinates. Finally, $L_{PointCloud}$ is designed as

$$L_{PointCloud} = \frac{1}{T}\left(\sum_{i\in A}\sqrt{\left(\frac{u_i-u_i'}{u_i-u_0}\right)^2+\left(\frac{v_i-v_i'}{v_i-v_0}\right)^2}+\sum_{i\in B}\left|\frac{v_i-v_i'}{v_i-v_0}\right|+\sum_{i\in C}\left|\frac{u_i-u_i'}{u_i-u_0}\right|\right) \quad (11)$$

- Calculation of virtual normal loss ($L_{VirtualNormal}$)

Assuming that n_i and n_i' are respectively the ground-truth and estimated virtual normal vectors of the i-th pixel in the image, then $L_{VirtualNormal}$ is computed as

$$L_{VirtualNormal} = \frac{1}{T}\sum_{i=1}^{N}||n_i - n_i'||_1 \quad (12)$$

IV. Experiment

In this section, we evaluate the performance of combining different loss functions with scale-invariant loss functions. We train and test these models using the NYU-Mini datasets. NYU Depth v2 [20] is an indoor dataset that contains RGB images and depth maps. The data was captured using Microsoft Kinect in various indoor scenes, with a uniform image resolution of 480 × 640. The dataset contains 120K training samples and 654 test samples, with a maximum valid depth value of 10 meters. Due to experimental cost considerations, we proportionally cropped the data from each scene in the NYU-Depth-v2 dataset to create a subset of 1,597 images. We used this subset, named NYU-Mini, for training in this paper. For testing, we did not make any adjustments and used the original 654 test samples. Figure 3 shows three examples of NYU Depth v2. The first row shows RGB images, and the second row shows their corresponding depth maps.

Fig. 3. Example images from NYU Depth v2. The first row shows RGB images, and the second row shows their corresponding depth maps.

Table 7 compares the performance of depth estimation using different loss functions on NYU-Mini Dataset, including scale-invariant loss (SI), point cloud conversion loss (PCL), and virtual normal loss (VNL). SI was proposed by Eigen [9] and has become a commonly used loss function in the field of depth estimation. In our experiments, SI is used to compute the depth loss. The weight values for the loss functions are shown as $\alpha_1 = 10$, $\alpha_2 = 8$ and $\alpha_3 = 4$ for Eq. (7). The results demonstrate that the proposed multi-task loss function effectively improves the accuracy of depth estimation. Figure 3 shows two estimated depth maps using different loss functions. The red-box regions highlight the places where our method performs better.

Table 7: Performance Comparison with Different loss functions on the NYU-Mini Dataset. SI is scale-invariant loss, PTL is point cloud loss, and VNL is virtual normal loss.

SI	0.7653	0.1687
SI+PCL	0.7786	0.1659
SI+VNL	0.7817	0.1616
SI+PCL+VNL	0.7850	0.1633

V. Conclusion

This paper introduces a novel semi-multi-task loss function aimed at enhancing the performance of monocular depth estimation. Our experimental results clearly demonstrate a significant performance boost when utilizing this proposed loss function. Our comprehensive approach maximizes the adaptability of point clouds. In addition to generating the fundamental depth map, it provides supplementary supervision by incorporating point clouds and surface normals. Notably, this method can be computed without the need for training additional subnetworks, thus demonstrating high generality and practical utility. In the future, we plan to extend the application of this proposed loss function to various other depth estimation modules to assess its overall effectiveness.

ACKNOWLEDGEMENT

This work is supported by the National Science Council of Taiwan with grant no. NSTC 112-2221-E-216-004.

References

[1] M. Alam, M.D. Samad, L. Vidyaratne, A. Glandon, K.M. Iftekharuddin, Survey on deep neural networks in speech and vision systems, Neurocomputing 417, pp. 302–321, 2020.

[2] J. Valentin, A. Kowdle, J.T. Barron, N. Wadhwa, M. Dzitsiuk, M. Schoenberg, V. Verma, A. Csaszar, E. Turner, I. Dryanovski, et al., Depth from motion for smartphone, ACM Trans. Graph. (TOG) 37, pp. 1–19 , 2018.

[3] X. Yang, H. Luo, Y. Wu, Y. Gao, C. Liao, K.T. Cheng, Reactive obstacle avoidance of monocular quadrotors with online adapted depth prediction network, Neurocomputing 325, pp. 142–158, 2019.

[4] Y.M. Tsai, Y.L. Chang, L.G. Chen, Block-based vanishing line and vanishing point detection for 3d scene reconstruction, in: 2006 International Symposium on Intelligent Signal Processing and Communications, IEEE, pp. 586–589, 2006.

[5] C. Tang, C. Hou, Z. Song, Depth recovery and refinement from a single image using defocus cues, J. Mod. Opt. 62, pp. 441-448, 2015.

[6] R. Zhang, P.S. Tsai, J.E. Cryer, M. Shah, Shape-from-shading: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 21, pp. 690-706, 1999.

[7] P. Zhang, J. Liu, X. Wang, T. Pu, C. Fei, Z. Guo, Stereoscopic video saliency detection based on spatiotemporal correlation and depth confidence optimization, Neurocomputing 377, pp. 256–268, 2020.

[8] R. Mur-Artal, J.M.M. Montiel, J.D. Tardos, Orb-slam: a versatile and accurate monocular slam system, IEEE Trans. Robot. 31, pp. 1147–1163, 2015.

[9] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, Adv. Neural Inf. Process. Syst., pp. 2366–2374, 2014.

[10] J.M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, J. Civera, Camconvs: Camera-aware multi-scale convolutions for single-view

depth, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 11826–11835, 2020.

[11] E. Ricci, W. Ouyang, X. Wang, N. Sebe, et al., Monocular depth estimation using multi-scale continuous crfs as sequential deep networks, IEEE Trans. Pattern Anal. Mach. Intell. 41, pp. 1426–1440, 2018.

[12] C. Godard, O. Mac Aodha, G.J. Brostow, Unsupervised monocular depth estimation with left-right consistency, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 270–279, 2017.

[13] X. Ye, X. Ji, B. Sun, S. Chen, Z. Wang, H. Li, Drm-slam: towards dense reconstruction of monocular slam with scene depth fusion, Neurocomputing 396, pp. 76–91, 2020.

[14] A. Tonioni, M. Poggi, S. Mattoccia, L. Di Stefano, Unsupervised domain adaptation for depth prediction from images, IEEE Trans. Pattern Anal. Mach. Intelligence, 42, pp. 2396-2409, 2020.

[15] X. Fei, A. Wong, S. Soatto, Geo-supervised visual depth prediction, IEEE Robot. Autom. Lett. 4, pp. 1661–1668, 2019.

[16] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, M. Pollefeys, Deeplidar: deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3313–3322, 2019.

[17] S. Bhat, I. Alhashim, P. Wonka, Adabins: Depth estimation using adaptive bins. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4009-4018, 2021.

[18] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning, PMLR, pp. pp. 6105-6114, 2019.

[19] W. Yin, Y. Liu, C. Shen, Y. Yan, Enforcing geometric constraints of virtual normal for depth prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5684-5693, 2019.

[20] Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012, October). Indoor segmentation and support inference from rgbd images. In European conference on computer vision (pp. 746-760). Springer, Berlin, Heidelberg.