Machine Learning with Applications 7 (2022) 100218

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Edge loss functions for deep-learning depth-map

Sandip Paul [a,*], Bhuvan Jhamb [b], Deepak Mishra [c,*], M. Senthil Kumar [a]

[a] Space Applications Centre, Ahmedabad, Gujrat, India
[b] Motilal Nehru National Institute of Technology Allahabad, Uttar Pradesh, India
[c] Indian Institute of Space Science and Technology, Trivandrum, Kerela, India

ARTICLE INFO

Keywords:
Depth estimation
Deep learning
Transfer learning
Ground truth
Loss functions
Modified SSIM
Edge functions
Performance metrics

ABSTRACT

Depth computation from an image is useful for many robotic systems like obstacle recognition, autonomous navigation, and 3D measurements. The estimation is best solved with Deep Neural Networks (DNN) as these are non-linear and ill-posed problems. The network takes single-color images with corresponding ground truth to predict depth-map after training. The depth accuracy, here, is dependent on the quality of ground truth and training images. Images have inherent blurs, which impact depth prediction and accuracy. In our work, we study different combinations of loss functions involving various edge functions to improve the depth of images. We use DenseNet and transfer learning method for learning and prediction of depth. Our analysis shows improvement in performance parameters as well as in the visual depth-map. We achieve 85% $\delta 1$ accuracy and improve log_{10} error using NYU Depth V2 dataset.

1. Introduction

Depth information is required in modern-day applications like terrain mapping, autonomous navigation, aircraft landing, self-driving cars, robotic humanoid activity, rendering of 3D scenes, 2D-to-3D image conversions, human gestures, super-resolution, etc. This information is obtained from active and passive systems. Active systems like LIDAR, RADAR, and the like, require high power. Considering the minimal resources available in portable and mobile platforms, these are avoided. Passive systems consist of visible and infrared cameras. Depth estimation relies upon either stereo-vision using images from several cameras or from a single camera-based monocular-vision using a single or stack of images. Stereo-vision requires scale, translation, and gain matching among the cameras through the calibration process. Literature survey on monocular-vision yield methods like varying the focus (Ens & Lawrence, 1993), defocus (Subbarao & Surya, 1994; Xian & Subbarao, 2005) cue-based detectors, and using multiple apertures which are affected by object–image distance conjugate. Depth is also derived using a stack of images. Monocular methods avoid image correspondence issues, as faced by stereo-vision, but need to address contrast (defocus) or magnification (focal length) issues. The aperture can be physically coded with numerous shaped apertures or colored apertures to obtain two or more images simultaneously from a single camera (Lee et al., 2013). In Refs. He et al. (2018), Zhao et al. (2020) the information of depth is recovered using blur cue from single images. It is to be noted that methods based on such single images are robust to contrast, magnification and spectral sensitivity. Further, these methods are simple, consume minimum power and are presently the trend towards 3-D object detection and autonomous driving.

In computer vision, deriving depth from a single image can have multiple solutions. Machine learning methods like deep convolution neural networks (CNN) have successfully addressed this ill-posed problem (Anwar et al., 2017; Gur & Wolf, 2020). These networks take color images and reference data (ground truth) from big datasets to train models using defined loss functions. The trained network then can predict the depth from any image. Networks consist of Convolution layers, pooling functions, and activation layers. Here, feature maps are learned by the kernels from each layer using back-propagation. This feature map is enriched by additional layers of enlargement and deconvolution for better prediction. The training can be either supervised or unsupervised (Chi et al., 2019; Harsányi et al., 2019). Networks can be customized (Alhashim & Wonka, 2018; Shivakumar et al., 2019).

In an image, edges capture local features and these define contours, regions and features in an image (van Dijk & Croon, 2019). Edges are defined as the image positions where the intensity of two neighboring pixels is significantly different. Blurred edges lead to loss of object definitions and merge it with the background which can result in reduced dynamic range and loss of depth map details (Yue et al., 2020). Hence, edge enhancement improves accuracy of depth-map resolution (Xie

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding authors.
E-mail addresses: sandippaul@live.com (S. Paul), bhuvan@mnnit.ac.in (B. Jhamb), deepak.mishra@iist.ac.in (D. Mishra), msenthil@sac.isro.gov.in (M.S. Kumar).

https://doi.org/10.1016/j.mlwa.2021.100218
Received 30 June 2021; Received in revised form 10 November 2021; Accepted 12 November 2021
Available online 27 November 2021

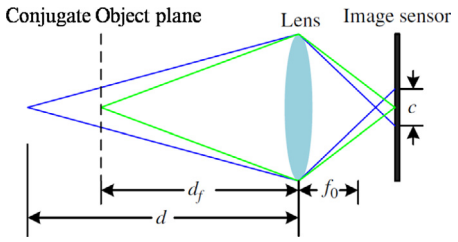

Fig. 1. Blur formation due to camera optics.

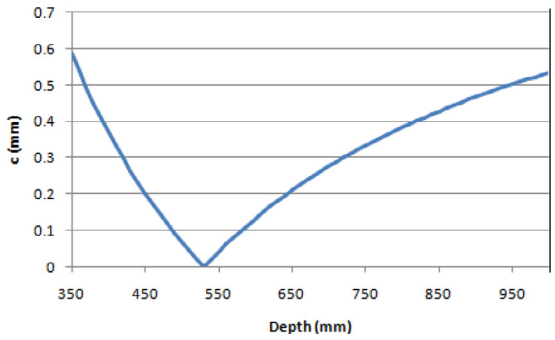

Fig. 2. The blur (COC) varies with the distance of the target object to the lens center (Eq. (1)). It is minimum when the object is in focus.

et al., 2016) estimations and lead to robust body parts (Zhang & Tian, 2013). In computer vision, gradients effectively detect the edges. Nonlinear filters with image smoothing are also applied to enhance edges as these produce less fragmentary edges.

Researchers have reported enhancement of the depth map edges leading to new application fields. Enhancing depth map is important as it improves quality. Deep learning methods use a loss function for edge enhancement or sharpening of depth maps. The loss function is effectively used for training a model. In our work, We implement various loss functions based on edge detectors to train a model. We study various edge operators and compare their impact on depth map quality. Further, we also implement various combinations of loss functions to analyze the performances. In this paper we discuss the experiments carried out and the results. Based on our literature survey, we feel this study is unique. Our main contributions are:

1. Improved SSIM loss function with sharpened depth map
2. Edge loss function with 5 different edge operators
3. Propose new loss function using improved SSIM loss, BerHu loss and Sobel loss
4. Analysis of quantitative performance and visual quality

2. Defocus model

Images are prone to aberration blurs, motion blurs, and depth of focus blurs. The ideal camera is a paraxial, thin lens system which is defined by lens-law as in Fig. 1. Model has f_0 as the focal length, d as the random object distance from lens center, d_f as the distance when the object is at focus and N as $optical\text{-}aperture/f_0$. The image of an object is formed on the sensor (image plane) by the collecting optics due to refraction. An object at d_f is only focused on the sensor while the other objects at d or behind d_f is defocused. These defocused objects create a blur called 'circle of confusion' (COC) on this plane. The COC has a diameter c which is related to d and f_0 as:

$$c = \frac{|d - d_f|}{d} \frac{f_0^2}{N(d_f - f_0)} \tag{1}$$

Image formation on the sensor depends on how focused or defocused a particular 3D world point is as presented in Fig. 2. The c varies with (a) the object distance d_f and (b) the focal length f_0. In a camera the focal length is usually fixed. Hence, the depth (object distance) d and d_f can be computed, for a given camera setting (focal length), from a known c. Similarly, a 2D depth-map can be constructed using the spatial blur in an image.

The camera-model adapted for our work (Fig. 1) assumes that:

1. Camera is focused on at least one object in the scene.
2. A blurred edge was a sharp edge in the scene.
3. The blur varies spatially and all blurs are modeled as single-disc PSFs.
4. The blur dimension is significantly smaller than the object.
5. Image does not have over/under-exposed pixels.

The blur is modeled as a power spectral function (PSF). The Gaussian PSF $h(x, y)$ is represented as.

$$h(x, y) = \frac{1}{2\pi\sigma^2} exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right) \tag{2}$$

Where σ = Spread parameter

We can express a defocused image g is formed from the sharp image f by convolution with PSF as:

$$g(x, y) = h(x, y) \otimes f(x, y) \tag{3}$$

The image blur can be reduced by deblurring the image with a filter kernel having inverse properties. If s is the deblurring kernel then the deblurred image is given by:

$$f_s(x, y) = s(x, y) \otimes g(x, y) \tag{4}$$

3. Description of datasets used in our work

NYU datasets (Nathan Silberman & Fergus, 2012) are popular for training and evaluating depth methods. This dataset is obtained from Kinect 3D camera (Microsoft) and provides high resolution labeled pairs of aligned 640 × 480 pixel color(RGB) and depth images. These indoor image sets comprise of 464 scenes and 407,024 unlabeled frames. The semi-synthetic depth maps are preprocessed with popular inpainting method to fill missing values due to shadows and noise from specular and low albedo surfaces. The maximum depth range is 10 m. The dataset does not provide pixel-level depth information. In our work, we use this dataset to have a fair comparison. Further, as these were used originally for training, any improvement employing the same DNN is an achievement. Standard practices are followed for data augmentation to improve generalization performance such as random mirroring and random color channel swapping of input images.

The NYU Depth-V2 dataset contains 120 K training samples and 654 testing samples. Authors (Alhashim & Wonka, 2018) have provided a dataset of 50 K for training. This is a subset from the NYU Depth-V2. This dataset is used.

4. Baseline deep network architecture for depth estimation

Deep networks automatically extract image features. The outputs are defined by loss functions (Janocha & Czarnecki, 2017). Training a network from scratch is effort-intensive. Transfer learning allows us to reuse an available trained model for a new related task, thus saving time and effort. One such recent method is presented in paper (Alhashim & Wonka, 2018). Their deep network architecture uses a DenseNet-169 as the baseline deep architecture. It uses a single encoder–decoder design along with skip connections. The input RGB image is encoded into a feature vector using the DenseNet-169 network pretrained on ImageNet. This vector is then fed to a decoder. The

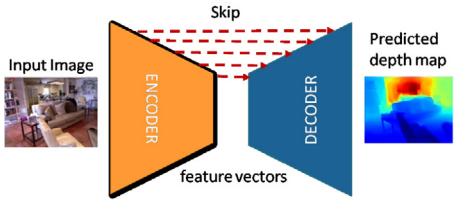

Fig. 3. Network architecture (Alhashim & Wonka, 2018).

decoder consists of successive series of $2x$ bilinear up-samplers, skip-connections, two (3×3) convolution layers and leaky ReLU (except the last block) in order to estimate the depth map. The decoder does not have Batch Normalization. Our network estimates depth map at half the resolution of input image (320×240). For training, the original ground truth depth images are also down sampled to 320×240. During test time, we compute the depth map prediction of the full test image and then up sample it by $2\times$ to match the ground truth resolution and evaluate on the predefined center cropping as defined by Eigen et al. At test time, we compute the final output by taking the average of an image's prediction and the prediction of its mirror image. The schematic of the network is shown in Fig. 3. We have extended their proposal by incorporating critical loss functions to apply variants of edge loss, SSIM and use systematically both MAE and MSE in the form of BerHu loss function. These improvements help us to estimate a better depth-map. The details of our proposal are described in the following sections.

5. Description of proposed loss functions for depth estimation

For training machine learning algorithms, the loss function formulation plays a crucial role. In our case, the ideal loss function should reduce edge distortions and reconstruct a depth map similar to the ground truth by forcing the model to converge and thereby minimize loss. An optimum loss function can train the model faster and yield good performance by balancing the penalizing and reconstruction recursive methods.

5.1. Mean absolute error loss

A conventional loss function typically used is the Mean Absolute Error Loss (MAE) or L1 loss function. Authors (Alhashim & Wonka, 2018) have also used MAE, as part of their loss function, in their proposal. The MAE loss function has a constant and large loss gradient which is robust to data outlier. (Fig. 4, orange curve). Furthermore, the MAE computes pixel-wise errors and is represented as:

$$L_{pix_1} = \frac{\sum_{i=1}^{n} |Y_i - Y_{pred_i}|}{N} \tag{5}$$

where Y_i is a pixel value in the ground truth depth-map, pixel Y_{pred_i} is from the predicted output depth-map and the total number of pixels amount to N in the depth-map.

5.2. Reversed Huber loss function

MAE loss has some limitations since it is linear and this property restricts estimating a better quality depth. This property gives equal weight to lower and higher values leading to fewer predictions. In order to explore the impact of the loss function in our work, we have improved upon the loss function of authors (Alhashim & Wonka, 2018) by combining MAE and with mean square error (MSE) also known as reversed Huber loss function or BerHu loss. This BerHu Loss has an advantage here, as it attaches better weight to pixels with higher residuals by selecting MSE (or L2) loss. Simultaneously, it permits lower residuals to have a wider effect on the gradients as MAE loss. This

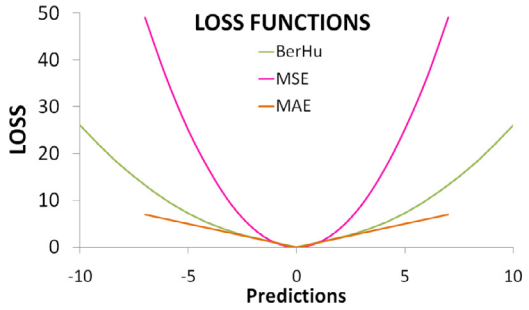

Fig. 4. A comparison between MAE, MSE and BerHu functions.

applies a trade-off among L1 loss and L2 loss about a threshold t_h. In our experiment, for a batch, a threshold ($t_h = 0.2$) is considered to decide the loss function that has to be active while training the deep neural network architecture. We use the following equation to express the loss function in our training:

$$L_{pix_2} = \begin{cases} |Y_{pred_i} - Y_i| & |Y_{pred_i} - Y_i| \le t_h, \\ ((Y_{pred_i} - Y_i)^2 + t_h^2)/2t_h & |Y_{pred_i} - Y_i| > t_h \\ t_h = 0.2 max_i(|Y_{pred_i} - Y_i|)\} & \end{cases} \tag{6}$$

A comparison of MAE(L1), MSE(L2) and BerHu functions are plotted in Fig. 4 with simulated loss. It is observed here that the BerHu loss function utilizes both MAE and MSE in an elegant manner, thus benefiting the overall training procedure.

5.3. Structural similarity index loss function

To enhance the perceptual quality of the depth-map, another loss function is considered. This loss function utilizes the input image structure and associated features using SSIM (Structural Similarity Index) loss function in conjunction with the BerHu loss function. The SSIM loss measures the perceptual difference between two similar images. Structural loss details come from spatially adjacent pixels that shows a stronger relation. These pixels hold crucial information about the structure of the objects in the visual scene (Gur & Wolf, 2020). When SSIM is 1, the images are similar to each other, with the same structure. The loss function is expressed as:

$$L_{SSIM} = 1 - SSIM(Y_i, Y_{pred_i}) \tag{7}$$

Maximum advantage is harnessed from SSIM by utilizing an edge enhanced image. This replaces the original ground truth image in SSIM. The following expression (improved SSIM) is used in this research work:

$$L_{SSIM} = 1 - SSIM(Y_i', Y_{pred_i}) \tag{8}$$

The improved loss function uses an edge enhanced sharpened ground truth image i.e. Y_i' in place of Y_i. This improves the construction details of the predicted depth map. The image is sharpened by a second order derivative Laplacian Filter. The filter highlights both the positive and negative edges of the feature boundaries in an image. This processed image is added to the original image. The following filter has been used to obtain the edge enhanced image:

$$s = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The final loss function that we consider in conjunction with BerHu and improved SSIM loss is called the edge loss function. This is used to improve edge related features and high frequency details of the structure from the training image. The edge detection approaches studied are Gradient, Sobel, Laplacian, Laplacian of Gaussian (LOG) and Difference of Gaussian (DOG). The application of these different

edge detection operators improves the depth estimation performance and provides a sharper depth map. We train with each type of edge operator, one at a time, for our studies. A total of five training sessions with different operators were completed. In our experiments the Sobel edge detection operator turns out to be the best. The details of the experiments will be discussed in section 8. The following subsections elucidate the description of various edge detection operators that we use in our study.

5.4. Gradient edge detector

This simple edge detection operator is based on computing the gradient of the neighboring pixels in an image (depth map). The intensity gradients using finite difference method for the horizontal and vertical directions of an image Y is expressed as:

$$\begin{aligned} \frac{\partial Y}{\partial x} &= Y(x+1,y) - Y(x-1,y), \\ \frac{\partial Y}{\partial y} &= Y(x,y+1) - Y(x,y-1) \end{aligned} \tag{9}$$

The gradient edge loss function L_{edges_1} is then expressed as:

$$L_{edges_1} = mean\left(\left|\frac{\partial Y_{pred}}{\partial y} - \frac{\partial Y}{\partial y}\right| + \left|\frac{\partial Y_{pred}}{\partial x} - \frac{\partial Y}{\partial x}\right|\right) \tag{10}$$

Although the gradient edge operator is the simplest edge detection method, it produces dual edges because of the first derivative operator. Further, proper thresholds are necessary to remove small peaks or noise.

5.5. Sobel edge detector

The Sobel edge operator is less sensitive to noise. This operator uses one 3×3 kernels each for horizontal and vertical directions to map the 2-D spatial gradient measurements. The kernels are represented as:

$$Sb_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; Sb_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The gradient components from x and y directions are utilized to compute the approximate gradient magnitude L_{mag} as:

$$|L_{mag}| = |Sb_x| + |Sb_y| \tag{11}$$

The Sobel edge loss function L_{edges_2} based on Sobel operator is then expressed as:

$$L_{edges_2} = mean(|I_{pred} * Sb_x - I * Sb_x| + |I_{pred} * Sb_y - I * Sb_y|) \tag{12}$$

Here the loss depends on the mean of the gradient magnitudes computed from the difference of depth values (predicted and ground truth). This loss penalizes errors due to large differences between the two values. Sobel operators approximate the first order derivatives in only x and y directions and may need a proper threshold to reduce noise.

5.6. Laplacian edge detector

The Laplacian operators give the second spatial derivative values and detect rapid changes with zero crossings. The Laplacian $L(x,y)$ of an image with pixel values $Y(x,y)$ is expressed as:

$$L(x,y) = \frac{\partial^2 Y(x,y)}{\partial x^2} + \frac{\partial^2 Y(x,y)}{\partial y^2} \tag{13}$$

We use the Laplacian kernel which is advantageous for matrix operations. The kernel is represented as:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The Laplacian edge loss function L_{edges_3}, based on this operator, is then expressed as:

$$L_{edges_3} = mean(|Y_{pred} * L - Y * L|) \tag{14}$$

We like to highlight here, that as the Laplacian operators are based on a single kernel, these are computationally faster. Furthermore, Laplacian operators are insensitive to orientation. However, these second order derivative operators are super sensitive to noise.

5.7. Laplacian of Gaussian edge detector

The Laplacian operator is normally used with a smoothing Gaussian filter. This filter is used on depth map before edge detection operators to reduce the sensitivity to noise. The smoothing 2D filter is expressed as:

$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{15}$$

Alternately, we can also take the Laplacian of Gaussian (LOG) and apply the same on the depth-map. LOG combines the Laplacian and Gaussian expressions to obtain a unified equation as:

$$LOG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{16}$$

The edge is also defined by detecting zero crossing. The implemented 3×3 Gaussian kernel in our work is:

$$Gaussian_1 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The LOG edge loss function L_{edges_4}, based on LOG operator, is then defined as:

$$L_{edges_4} = mean(|Y_{pred} * (Gaussian_1 * L) - Y * (Gaussian_1 * L)|) \tag{17}$$

The LOG operator is useful for noisy images. In image processing, images are usually smoothed before any use. This operator, in such cases, have the advantage of computational speed, since only the Laplacian operation is required. The other advantages of Laplacian operator discussed above are also valid here. This operator may miss the boundaries in the background due to loss of high-frequency components after smoothing.

5.8. Difference of Gaussian edge detector

Similar to LOG, an edge detector can be based on the difference of smoothed images using two Gaussian filters with different scales (σ). This two stage filter detects the edge by zero crossing. The DOG operator is defined as:

$$DOG \triangleq \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2}\right) \tag{18}$$

The DOG operation is independent of the derivative and is computationally simple. In our work, We use two scales of Gaussian filters, a 3×3 kernel as described above and a second 5×5 kernel defined as:

$$Gaussian_2 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

The DOG edge loss function L_{edges_5}, based on DOG operator is then expressed as:

$$\begin{aligned} L_{edges_5} = mean(|&(Y_{pred} * Gaussian_1 - Y_{pred} * Gaussian_2) \\ &- (Y * Gaussian_1 - Y * Gaussian_2)|) \end{aligned} \tag{19}$$

This operator is beneficial for noisy images and rapid varying contrast in images like shadows. The detected edges are weak due to loss

of high-frequency components after smoothing. This will be prominent for blurred (out of focus) background features. Similar to LOG operator, as most images are already filtered, computational efforts are only required for operation with the second smoothing filter.

5.9. Proposed loss-function

Here, we explain the details of proposed overall loss-function. The loss-function proposed for our work is the weighted combination of MAE, BerHu, SSIM and one of the various edge loss functions from above sub-sections. This loss function is expressed as:

$$L_{total}(Y_i, Y_{pred_i}) = \lambda_1 L_{pix_l}(Y_i, Y_{pred_i}) + \lambda_2 LSSIM(Y_i, Y_{pred_i}) + \lambda_3 L_{edges_k}(Y_i, Y_{pred_i}) \quad (20)$$

Please note that the subscript l here refers to the MAE function or BerHu function based loss. The loss function L_{pix_l} is defined by Eqs. (1) and (1). This loss-function estimates the similarity among the actual and the predicted depth-map utilizing either MAE and BerHu. Similarly, subscript k here refers to the various edge loss functions (described in sub Section 5.1 to Section 5.5). The improved SSIM loss function (Eq. (1)) is considered here to benefit the perceptual quality of the estimated depth-map.

The weighting factor λ_1 is selected based on trial and error method. A value of 0.1 for λ_1 is found to be the most appropriate when MAE loss is active and a value of 1 for λ_1 when BerHu loss is active. The values for λ_2 and λ_3 were kept as 1 as this was found to be optimum.

6. Results and experiments

In order to train the deep learning model as shown in Fig. 3, the total loss function given in Eq. (20) is used. In the subsequent sections, the description of various performances, characteristics and experiments are given.

6.1. Standard performance metrics

The performance evaluation metrics to rate the performances of our proposed models with some of the recent methods (Alhashim & Wonka, 2018; Eigen et al., 2014; Lubor Ladicky & Pollefeys, 2014) are presented below:

- Root-Mean-Squared-Error (RMSE):

$$RMS = \sqrt{\frac{1}{N}\sum_{i \in N}(Y_i - Y_{pred_i})^2} \quad (21)$$

- Average-relative-error (REL):

$$REL = \frac{1}{N}\sum_{i \in N}\frac{|Y_i - Y_{pred_i}|}{Y} \quad (22)$$

- Average (log_{10}) error:

$$log_{10} = \frac{1}{N}\sum_{i \in N}|log_{10}(Y_i) - log_{10}(Y_{pred_i})| \quad (23)$$

- Threshold accuracy(δ_η) gives the percentage of pixels having relative error below a defined threshold (1.25, 1.25^2, 1.25^3) It is based on the maximum ratio of predicted pixels and the ground truth pixels (Lubor Ladicky & Pollefeys, 2014). This metric is represented as % of Y_i s.t.

$$max\left(\frac{Y_{pred_i}}{Y_i}, \frac{Y_i}{Y_{pred_i}}\right) = \delta_\eta < \eta, \quad \eta = 1.25, 1.25^2, 1.25^3 \quad (24)$$

Where Y is the average pixel value in the actual (ground truth) depth image.

To assess the quality of the obtained depth, the performance indicator with a lower value of RMSE, a lower of REL and a smaller value of log_{10} error is preferred. However on the other-hand, a higher value of δ_1, δ_2 and δ_3 is preferred.

6.2. Hardware and software configuration

We trained the model on Google CoLaboratory. Single GPU (Tesla T4, Tesla P100 or V100 GPU) with CUDA 11.2 version was used. Most of the training session was limited to 20 epochs as the improvements beyond were insignificant. We used a batch size of 6 and 8 depending on GPU and memory availability. The weights for the decoder are randomly initialized, and an ADAM optimizer is used. During training, ground truth depth maps were normalized between 0.01 and 1 to avoid any numerical issues. It was reported by the authors in paper Wang (2020), that the prediction error in the estimated depth-maps increase for the distant targets/features compared to the ground-truth map. To overcome this limitation, we consider the reciprocal of ground truth depth ($10/Y_i$ for prediction, where 10 m is the maximum depth in the NYU dataset) in our proposal.

We used NYU-V2 datasets (4.1 GB) downloaded from Nathan Silberman and Fergus (2012). Subsequently we have fine-tuned our model using the additional dataset available from Alhashim and Wonka (2018). Furthermore, we match our results with the reference paper Alhashim and Wonka (2018). The detailed analysis is shown in Table 3. In this table we also present the performance obtained by the proposed models after considering several improved loss functions. The 'Baseline model' tabulates the results published in the said paper.

It is evident from Table 3, that the proposed improvement (BerHu + Sobel + SSIM–Sharpened) turned out to be the overall best among all other loss functions. However, in terms of δ_1 accuracy the proposed model with Sobel Loss function gave the best result. The same can be verified from Fig. 7 to Fig. 10.

For a fair evaluation, the test dataset of NYU V2 is never used for training the model. To examine the perceptual quality of our result, random RGB images were segregated from the NYU V2 test dataset. We would like to highlight here, that these are complex indoor images having good depths in large rooms, small simple rooms with less furniture and features, varying lights and with a mix of textures like furniture, ladder, door, rack, wall corners, roof, etc. at various places in the rooms.

The nine estimated depth maps from all the models are concatenated and presented in Fig. 5 with false color. This figure compares the visual quality performance of the models. Here, the blue color represents the shortest distance while dark red is the largest distance. We also computed the difference map from the actual and the estimated depths to highlight the improvements. These are plotted for the proposed model to understand the performance at the edge locations and at sudden depth variations (Fig. 6). Here, the color Cyan has the lowest error while pink has the highest error. The more the cyan color, the better the match of predicted depth map to ground truth.

6.3. Ablation study and analysis of results obtained via various models

We implement different loss functions for training based on Eq. (1). The implementation of the loss functions for training the models is in the following format:

1. SSIM + Gradient + MAE (SGM)
2. SSIM–Sharpened + Gradient + MAE (SSIM')
3. BerHu + SSIM + Gradient (BerHu)
4. SSIM–Sharpened + BerHu + Gradient (SSIM'+B)
5. SSIM–Sharpened + BerHu (BSSIM')
6. Sobel + Gradient + MAE (Sobel)
7. Laplacian + Gradient + MAE (Laplacian)
8. LOG + Gradient + MAE (LOG)
9. DOG + Gradient + MAE (DOG)
10. BerHu + Sobel + SSIM (B+Sobel)
11. BerHu + Sobel + SSIM–Sharpened (B+S+SSIM')

Table 1
Performance of weights for B+S+SSIM' model.

λ_1	λ_2	λ_3	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL ↓	RMSE ↓	log10 ↓
0.1	1	1	0.811	0.971	0.994	0.133	0.606	0.06
1	0.1	1	0.842	0.972	0.9941	0.128	0.539	0.054
1	1	0.1	0.848	0.973	0.9939	0.126	0.534	0.054
1	1	1	0.845	0.973	0.9939	0.124	0.524	0.053

Table 2
Performance comparison of trained models.

Model	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL ↓	RMSE ↓	log10 ↓
DenseNet169	0.8453	0.9732	0.9939	0.1238	0.5242	0.053
DenseNet202	0.8495	0.9736	0.9941	0.1233	0.5264	0.0528

The model-B+S+SSIM' is trained with two different encodes (a) DenseNet 169 and (b) DenseNet 202. DenseNet 202 fares better.

Table 3
Performance comparison of trained models.

Model	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL ↓	RMSE ↓	log10 ↓
Sobel	**0.8526**	0.9709	0.9919	0.1253	0.5585	0.0529
BerHu	0.851	0.972	0.993	0.126	0.552	0.053
BSSIM'	0.8505	0.9727	0.9932	0.1265	0.5367	0.0533
Laplacian	0.8504	0.9716	0.9926	0.1255	0.5441	0.0531
B+S+SSIM'	0.8494	0.9727	0.9940	0.1231	0.5234	**0.0528**
B+Sobel	0.8480	0.9725	0.9937	0.1259	0.5388	0.0534
SSIM'	0.847	0.9725	0.9929	0.1237	0.5429	0.054
SSIM'+B	0.8448	0.9706	0.9934	0.1275	0.5513	0.0539
LOG	0.8316	0.9669	0.9926	0.135	0.579	0.0565
DOG	0.8034	0.9588	0.9898	0.146	0.6032	0.0617
Baseline (Alhashim & Wonka, 2018)	0.846	0.974	0.994	0.123	0.465	0.053

Note: The models are ranked as per δ_1, with the largest on top. The best results are in bold. Overall best performance is met by model-B+S+SSIM'.

Here, the short form of the loss function is given inside the brackets. All the models estimated depth maps. These models ran evaluation dataset. The weights for loss functions were tuned to get the optimal values. The results for model B+S+SSIM' are tabulated in Table 1. The overall best performance is achieved for λ_1, λ_2 and $\lambda_3 = 1$. These weights are used for our work.

We studied the performances of our loss function- B+S+SSIM' with two different encoders viz. DenseNet169 and DenseNet201. DenseNet201 has 201 layers with 20M parameters, thus more accurate. we ran for 10 epochs in each case. The performance of DenseNet201 is better. the same is presented in Table 2.

6.4. Performance analysis of proposed loss functions

While comparing pixel wise loss for various combinations of loss functions, as explained in Section 6.3, almost all of our models perform better than the baseline model in terms of δ_1 performance (Please refer to Table 3 and Fig. 5). Among the pixel-wise loss functions (Section 6.3), BerHu loss out-performs MAE loss (Please refer Table 3 and Fig. 8). This is expected as BerHu is suitable for long-tailed data such as depth-maps.

Among all the loss functions studied, δ_1 performance metrics are the highest for Sobel edge loss (85.26%). Table 3 presents model ranking w.r.t. δ_1 performance. Model with the highest δ_1 is kept as first row.

The B+SSIM' performance was second best for δ_1. This implies that we can get good performance with reduced computation complexity for any image as edge computation is an expensive operation.

Overall, the performance of models using loss functions based on BerHu, SSIM', Sobel and Laplacian (except LOG) performed better than reference paper Alhashim and Wonka (2018). Further, improvements are also seen for log_{10} error on models based on Sobel and B+S+SSIM' loss functions.

Table 4
Performance of others.

Authors	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL ↓	RMSE ↓	log10 ↓
Swami et al. (2020)	0.87	0.97	0.99	0.115	0.528	0.049
Xu et al. (2017)	0.81	0.95	0.98	0.12	0.58	0.051
Yue et al. (2020)	0.86	0.97	0.99	0.12	0.48	0.051
Yang et al. (2017)	0.725	0.906	0.963	1.481	6.5	0.267
Ours	0.853	0.971	0.992	0.125	0.559	0.053

6.5. Quality and depth range

Model trained with Sobel edge function provide better perceptual depth map with more false colors (Fig. 5). The accuracy is higher, and the depth range is increased. This is more profound for images 3, 6, 7 and 8. Image 7 shows the high depth range for Sobel up to the end of the corridor. However, nonuniform lighting in this corridor leads to wrong depth information.

This ranking is followed by models — BerHu, SSIM', SSIM'+B and LOG. The other models are poorer in such aspects.

The performance of Sobel for near objects are poor. Fig. 9 shows the missing feature. Here the model ranking is BerHu followed by Laplacian, Sobel and DOG. This aspect is more prominent in image 4. In all cases the errors were low for near objects compared to distant features.

Mid range performance is good for models trained with BerHu and DOG. This factor can be seen in image 6 in terms of better details.

All loss functions performed poorly with poor scene lighting. Sobel gives wrong depth (notice board in image 7).

For visual inference and comparison, the difference map of predicted and the ground truth is presented in Fig. 6 for SGM, BSSIM', B+Sobel and B+S+SSIM'. The overall best performance is given by model based on B+S+SSIM'. This is prominent in image 7 and 9 (Fig. 6) for the wall and sofa.

Similar edge based depth-map work using NYU dataset by other researchers are also compared with our work. The same is summarized in Table 4.

7. Conclusion

We studied the performances of various loss functions on the quality of depth-maps on NYU datasets using transfer learning method and tailored improvements to the existing loss function. The loss functions studied were SSIM with sharpened image, reverse Huber loss, MAE loss and various combinations. The performance of edge loss function was evaluated through training with gradients, Laplacian, LOG, and DOG based operator loss functions. We believe that the work carried out in this research is significant to the computer vision community and provides one of its kind a comprehensive comparison of various combinations of loss functions for monocular depth map estimation.

Performance parameter δ_1 showed improvement for methods with BerHu, SSIM with sharpened depth map, Sobel and Laplacian. Our results are compared with the results from the original paper. The performance with edge function, followed by BerHu. The best δ_1 achieved was 85.26%. The combination of only BerHu and improved SSIM (BSSIM') was ranked as the third. Overall best performance was achieved with the loss models based on BerHu, Sobel and improved SSIM loss (δ_1: 84.94% and lower log_{10}:0.0528).

We conclude that our improved SSIM' loss is very effective for detailed depth map. We also can conclude that choice of edge functions has an impact on depth map performance. Similarly derivative and differentiation-based edge detectors perform better than smoothing-based edge detectors. Finally, the idea of using a systematic combination of loss function will be beneficial for other computer vision problems as well.

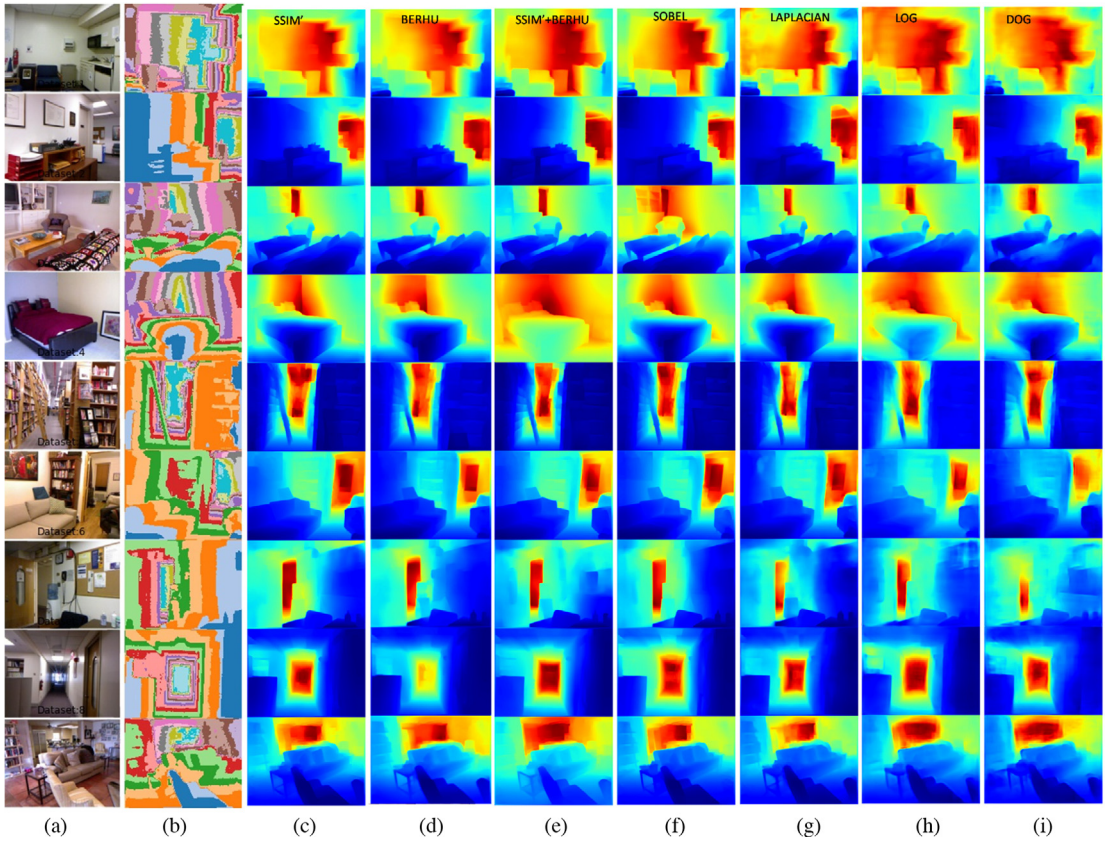

Fig. 5. Depth map prediction after training, a visual comparison (Dark blue is nearest and dark red is farthest). Modified loss functions: (a) Original image (NYU Dataset) (b) False color ground truth (c) SSIM', (d) BerHu, (e) SSIM'+B, (f) Sobel, (g) Laplacian, (h) LOG, (i) DOG. Here Sobel performs the best for δ_1 followed by BerHu. The details of the depth map are better than that of SGM model depth map.

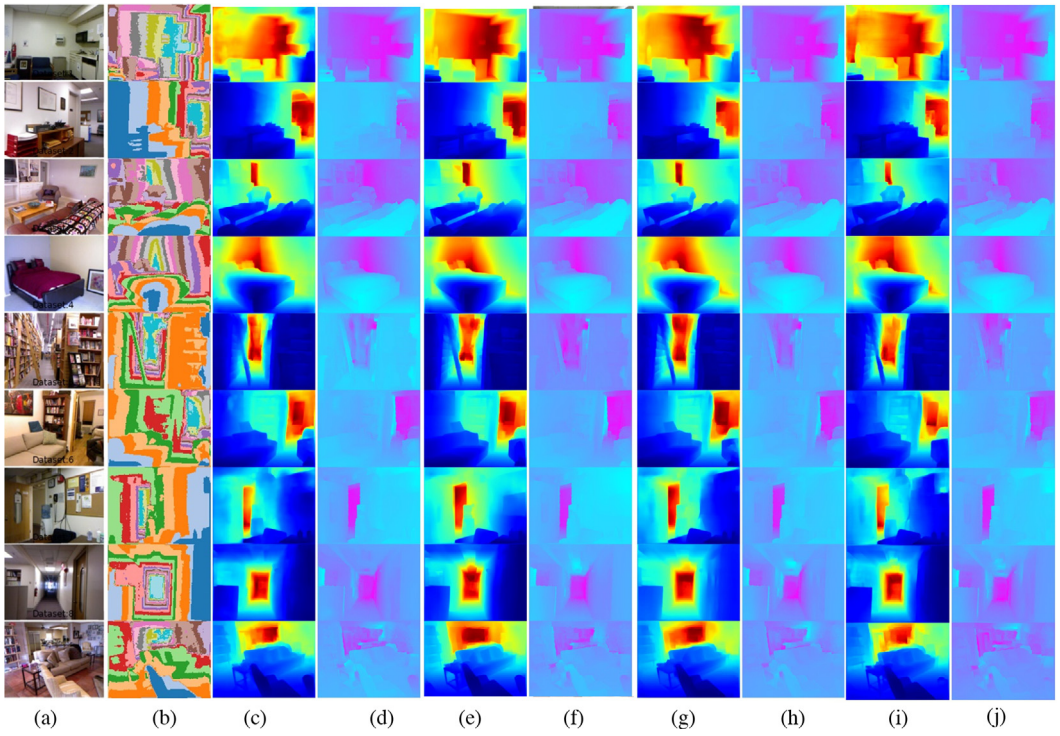

Fig. 6. (a) Original image (b) Ground truth (c) Predicted depth map (Loss: SGM), (d) Error map of ground truth and predicted (SGM), (e) Predicted depth map (B+Sobel), (f) Error map of ground truth and predicted (B+Sobel), (g) Predicted depth map (B+S+SSIM'), (h) Error map of ground truth and predicted (B+S+SSIM'), (i) Predicted depth map (BSSIM'), (j) Error map of ground truth and predicted (BSSIM'). The error map lower values are cyan and higher are pink. Here, it is evident that errors are high for distant features. Model B+S+SSIM' ranks the best w.r.t. depth errors.

Fig. 7. (a) Original image (b) Ground truth (c) Predicted depth-map (Alhashim & Wonka, 2018).

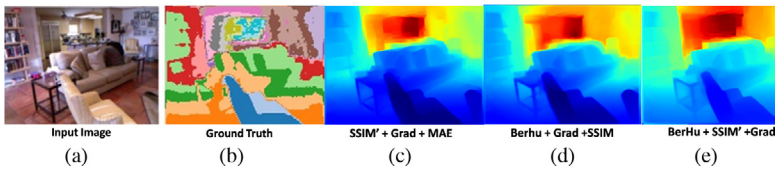

Fig. 8. Predicted depth-map with modified loss (a) Original image (b) Ground truth (c) SSIM sharpened, (d) BerHu, (e) SSIM sharpened and BerHu.

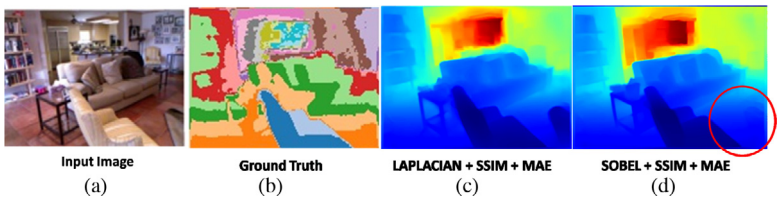

Fig. 9. Predicted depth-map with various edge loss functions (a) Original image (b) Ground truth, (c) Laplacian, (d) Sobel. The red circle refers to the feature missing here compared to Laplacian based depthmap.

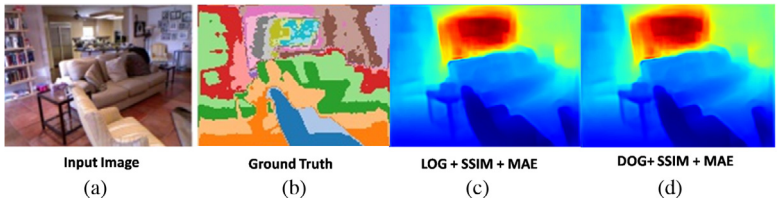

Fig. 10. Predicted depth-map with smoothing edge loss functions (a) Original image (b) Ground truth, (c) LOG, (d) DOG.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Alhashim, I., & Wonka, P. (2018). High quality monocular depth estimation via transfer learning. arXiv e-prints abs/1812.11941. http://arxiv.org/abs/1812.11941 [arXiv:1812.11941].

Anwar, S., Hayder, Z., & Porikli, F. (2017). Depth estimation and blur removal from a single out-of-focus image. *British Machine Vision Conference*.

Chi, J., Gao, J., Qi, L., Zhang, S., Dong, J., & Yu, H. (2019). Depth estimation of a single RGB image with semi-supervised two-stage regression. In *Proceedings of the 5th international conference on communication and information processing*. ACM, http://dx.doi.org/10.1145/3369985.3370004.

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. arXiv:1406.2283.

Ens, J., & Lawrence, P. (1993). An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(2), 97–108. http://dx.doi.org/10.1109/34.192482.

Gur, S., & Wolf, L. (2020). Single image depth estimation trained via depth from defocus cues. arXiv:2001.05036.

Harsányi, K., Kiss, A., Majdik, A., & Sziranyi, T. (2019). A hybrid CNN approach for single image depth estimation: A case study. In K. Choroś, M. Kopel, E. Kukla, & A. Siemiński (Eds.), *Multimedia and network information systems* (pp. 372–381). Cham: Springer International Publishing.

He, L., Wang, G., & Hu, Z. (2018). Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, *27*(9), 4676–4689. http://dx.doi.org/10.1109/tip.2018.2832296.

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. arXiv:1702.05659.

Lee, S., Hayes, M. H., & Paik, J. (2013). Distance estimation using a single computational camera with dual off-axis color filtered apertures. *Optics Express*, *21*(20), 23116–23129. http://dx.doi.org/10.1364/OE.21.023116, URL http://www.opticsexpress.org/abstract.cfm?URI=oe-21-20-23116.

Lubor Ladicky, J. S., & Pollefeys, M. (2014). Pulling things out of perspective. In *CVPR '14: proceedings of the 2014 IEEE conference on computer vision and pattern recognition* (pp. 89–96). USA: IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2014.19.

Nathan Silberman, P. K., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *ECCV*.

Shivakumar, S. S., Nguyen, T., Miller, I. D., Chen, S. W., Kumar, V., & Taylor, C. J. (2019). DFuseNet: Deep fusion of RGB and sparse depth information for image guided dense depth completion. arXiv:1902.00761.

Subbarao, M., & Surya, G. (1994). Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, *13*(3), 271–294. http://dx.doi.org/10.1007/BF02028349, URL https://doi.org/10.1007/BF02028349.

Swami, K., Bondada, P. V., & Bajpai, P. K. (2020). AcED: Accurate and edge-consistent monocular depth estimation. arXiv:2006.09243.

van Dijk, T., & Croon, G. C. (2019). How do neural networks see depth in single images? arXiv:1905.07005.

Wang, Y. (2020). MobileDepth: Efficient monocular depth prediction on mobile devices. arXiv:2011.10189.

Xian, T., & Subbarao, M. (2005). Performance evaluation of different depth from defocus (DFD) techniques. In *SPIE, vol. 6000* (pp. 87–99). http://dx.doi.org/10.1117/12.629611, URL https://doi.org/10.1117/12.629611.

Xie, J., Feris, R. S., & Sun, M.-T. (2016). Edge-guided single depth image super resolution. *IEEE Transactions on Image Processing*, *25*(1), 428–438. http://dx.doi.org/10.1109/TIP.2015.2501749.

Xu, D., Ricci, E., Ouyang, W., Wang, X., & Sebe, N. (2017). Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 161–169). http://dx.doi.org/10.1109/CVPR.2017.25.

Yang, Z., Wang, P., Xu, W., Zhao, L., & Nevatia, R. (2017). Unsupervised learning of geometry with edge-aware depth-normal consistency. arXiv:1711.03665.

Yue, H., Zhang, J., Wu, X., Wang, J., & Chen, W. (2020). Edge enhancement in monocular depth prediction. In *2020 15th IEEE conference on industrial electronics and applications* (pp. 1594–1599). http://dx.doi.org/10.1109/ICIEA48937.2020.9248336.

Zhang, C., & Tian, Y. (2013). Edge enhanced depth motion map for dynamic hand gesture recognition. In *2013 IEEE conference on computer vision and pattern recognition workshops* (pp. 500–505). http://dx.doi.org/10.1109/CVPRW.2013.80.

Zhao, C., Sun, Q., Zhang, C., Tang, Y., & Qian, F. (2020). Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, *63*(9), 1612–1627. http://dx.doi.org/10.1007/s11431-020-1582-8, URL http://dx.doi.org/10.1007/s11431-020-1582-8.