Monocular Depth Estimation
CS 7643 Project Report
Team: FADE Ain't Depth Estimation

Lixin Xu
Georgia Institute of Technology
lxu397@gatech.edu

Jiacheng Hou
Georgia Institute of Technology
jhou76@gatch.edu

Julong Li
Georgia Institute of Technology
jli3224@gatech.edu

Zheyang Liu
Georgia Institute of Technology
zliu891@gatech.edu

Abstract

Among different techniques for depth estimation, neural networks have proven their performance and exhibited great flexibility in various depth estimation settings. Currently, various learning-based depth estimation methods involve MiDaS, LeRes, SGR, etc. There are also boosting methods based upon these learning-based methods as backbones to improve the estimation performance. However, those approaches are subject to limitations. For instance, the depth maps inferred by boosting methods are not robust and sensitive to noise. To further identify this issue, we firstly aimed to explore the current boosting method based on MiDaS and LeRes, and evaluated its performance using NYU-v2 dataset both under noise and without noise. The experiment results are further analyzed using quantitative and qualitative approaches to identify the effect of noise on boosting performance, as a supplement to the current literature.

1. Introduction/Background/Motivation

1.1. Introduction

Depth estimation plays an important role in certain fields such as autonomous driving, environment perception, and reconstruction. According to different aspects, depth estimation is further divided into many sub-fields. In terms of cameras, there is monocular depth estimation based on single-lens cameras and stereo depth estimation using double-lens cameras. In terms of problem settings, there are image depth estimation and video sequence depth estimation. Various methods involved in depth estimation evolve from conventional methods to data-driven methods (see Section 2). For our project, we tried to investigate and rebuild a model based on existing work [12] to boost the performance of monocular depth estimation by utilizing models proposed in [14, 21].

1.2. Background

MiDaS [14] and LeRes [21] are currently state-of-the-art monocular depth estimation models. The boosting approach can choose either two models as backbones to the whole model [12].

Model	Description
MiDaS	Current model with SOTA performance by training with multi-source dataset, backbone to boosting
LeRes	Another SOTA model to infer depth and reconstruct 3D scenes, backbone to boosting
Boosting	A framework to boost the performance of current depth estimation models

Table 1. Comparison of Depth Estimation Models

According to [14], MiDaS is a way to train mixed data collected from multiple sources (see Section 1.4) to build a robust model that is able to generalize on unseen data. The method comes with specially designed loss functions that make the model invariant to changes in depth range and scale.

LeRes [21] is a method for the reconstruction of 3D scenes from a single monocular image, where the authors proposed a framework that firstly predicts depth value to an unknown scale and shift, and then predicts the lost depth shift and focal length to reconstruct a point cloud of 3D scene shape. The 3D reconstruction part is not used for our project, but rather the depth estimation from the LeRes

model.

The boosting method mentioned in [12] is not a direct network model for depth estimation itself, but rather a framework that uses MiDaS [14], SGR [18] or LeRes [21] as the backbone to boost the estimation performance. Besides, it involves modules for double estimation, patch estimation and Pix2Pix based merging network [5].

1.3. Motivation

Despite the achievement of the boosting method, as [12] stated, there are several limitations to this method. For instance, the current boosting method generates relative depth maps instead of absolute depth values. Besides, it is also stated that the model is sensitive to low-magnitude noise. Our motivation is based upon the impact of noisy images on the current boosting model as our quantitative experiment where we will calculate several indices to assess the impact, and we will also look into the effect of input resolutions where we will make modifications and assess the result qualitatively.

Besides, the boosting method is tested and validated on Middleblurry2014 and Ibims-1 dataset. We will further explore its generalization on other datasets such as NYU-v2.

Our work concerns the depth estimation community since this project is a supplement to the current literature [12]. The behavior of model under noise and corruption is discussed in details for the future reference of depth estimation field.

1.4. Datasets

There are various 3D vision datasets suitable for depth estimation tasks. They mainly consist of images with RGB values and corresponding depth annotations. These annotations are different in forms in terms of density (sparse or dense annotations) and types (absolute or relative depth annotations). In some of the datasets, depth annotation is acquired by sensors like Laser, ToF, or stereo cameras, others involve computed depth from motion (SfM, Structure From Motion) or synthesized depth data. There are also datasets with human depth annotations.

Our project is mainly based upon MiDaS [14] as our backbone model. According to [14], one of the major challenges of training is those ground truths have different forms across different datasets. This is because each dataset has its own characteristics, their corresponding data scale, image quality, depth accuracy, and corresponding camera parameters are different. Good results can be achieved on the same kind of test set when training on a single dataset. However, the generalization ability of other datasets is limited. Therefore, MiDaS adopts multi-dataset training to improve the generalization ability of the model.

In the training phase, five complementary datasets are used in MiDaS to train the model.

1. ReDWeb (RW) [17]: This dataset is a small dataset characterized by dynamic scenarios. That was acquired with a relatively large stereo baseline.

2. MegaDepth (MD) [9]: a large dataset displaying static scenes. Wide-baseline multi-view STEREO reconstruction was used for acquisition

3. WSVD (WS) [16]: consists of stereo videos obtained from the network.

4. DIML indoor (DL) [6]: AN RGB-D dataset of static indoor scenes was captured by Kinect V2.

5. 3D Movie [14]: a dynamic dataset composed of high-quality video frames.

However, the deep learning model used is large in scale, and datasets involved are complicated. In [14], the authors even spent six GPU months of computation. Consider that we as students do not have access to advanced hardware, and the project span is relatively short. Moreover, current models are well-trained in depth estimation tasks. As a result, we found that further training is not as useful, and decided to use a pre-trained MiDaS model for further evaluation. Therefore, we mainly put our effort into the evaluation phase, where we tested the model performance with different $\mathcal{R}_x$ values (mentioned in Section 3) and tested under noisy conditions.

In the evaluation phase, we decided to use NYU Depth Dataset V2 (NYU-v2) [13] to assess the model performance.

NYU-v2 is firstly proposed by Silberman et al. [13] to interpret major surfaces and objects of indoor scenes. It is widely used in the training and evaluation of depth estimation and semantic segmentation models. The dataset is composed of 1449 annotated RGB and depth images collected from 464 scenes of 3 cities. The annotations of images include labeled class and instance number. The evaluation result of NYU-v2 is further analyzed in Section 4.

2. Related Works

With the rise of deep learning, an increasing number of data-driven methods emerge and achieve impressive results in monocular estimation. These methods can generally be categorized into two main classes: supervised methods and self-supervised methods. Earlier works are mainly based on supervised learning, where they model the dataset distribution from the given data. A typical example can be found in [2], in which a global coarse-scale convolutional network and a local fine-scale convolutional network are collaborated to achieve a global-to-fine depth estimation from a single image. This approach is subsequently improved for multiple times by adopting the Conditional Random Fields

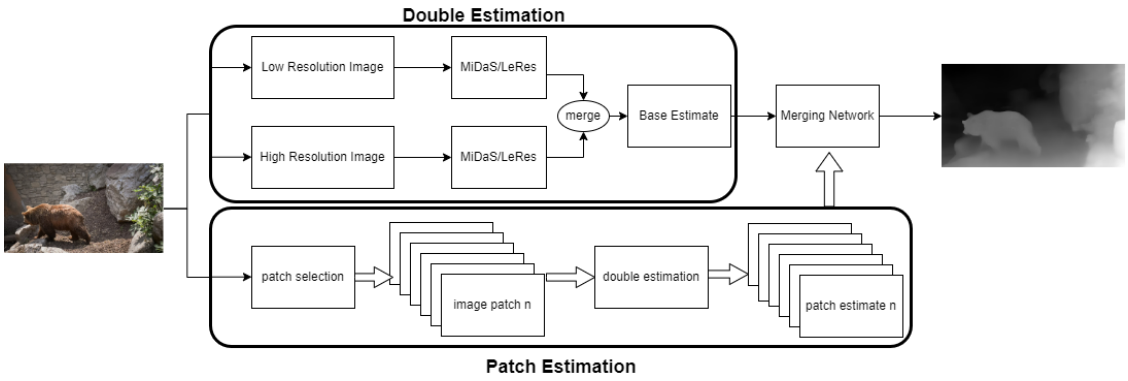

Figure 1. Structure of Our Project

(CRF) with vanilla form [10], hierarchical variant [8] and multi-scale variant [20]. On the other hand, some works place their attention on using better loss functions to stimulate the network performance including the reverse Huber (Berhu) loss and original regression loss. Currently, more and more researchers realize the importance of self-supervised and even unsupervised methods because the data label becomes more and more expensive while the network becomes more and more hungry owing to its increasing depth. Garg et al. [3] utilize the auto-encoder structure for depth map prediction from a single view image. In particular, the encoder is aimed at learning depth prediction, and the decoder learns to recover the depth map to the raw image. Zhou et al. [22] simultaneously estimate the depth and pose from the image and construct additional loss on the estimated pose to improve the network accuracy. Kuznietov et al. [7] adopt a part of ground-truth depth for supervised learning and then train the deep network using a direct image alignment loss.

3. Approach

In this project, we have re-implemented the model in [12]. In addition, we investigate the influence of different hyper-parameters $\mathcal{R}_x$ on the structural consistency of the estimated depth map. Besides, we also focus on the influence of noise even corruption of the raw image on the estimated depth map.

In the baseline approach [12], the influence of the image resolution to the estimated depth image has been explored. Specifically, there exists the structural inconsistency problem. *From our observation, when the input resolution is high, the structural inconsistency cannot be well preserved. When the image resolution is low, high-frequency details will be lost.* To solve this *problem*, we decide to try different resolutions and different adjustable hyper-parameters in [12] to alleviate the inconsistency in structure and loss in details. Although, in [12], it has already been found the resolution parameter $\mathcal{R}_x$ is influential, where x denotes the percentage of pixels not receiving contextual cues. It has not fully exploited how the $\mathcal{R}_x$ affects the estimated depth map and just set it as the default value. Thus, in our approach, we are going to explore different $\mathcal{R}_x$.

It is anticipated that probably there are other hyper-parameters are also influential, or $\mathcal{R}_x$ should be co-adjusted with other hyper-parameters, or the network should be re-trained again. In addition, since our approach is mainly based upon [12] to further improve the performance of MiDaS [14] and LeRes [21]. Thus, we can start by introducing the MiDaS network.

3.1. MiDaS

The network structure of MiDaS can be denoted as Figure 2, where the input RGB image is processed with 3 stages to obtain the estimated depth map.

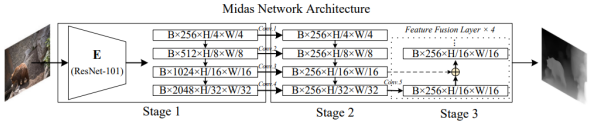

Figure 2. Structure of MiDaS

Stage 1. In stage 1, the encoder is simple, where the ResNet-101 [4] is directly employed as the backbone. Specifically, the pre-trained weight and several layers are used for feature representation, which is denoted as stage 1 as the Fig. 2 shows.

Stage 2. After stage 1, there are four feature maps with different sizes, and they are processed with four different convolutional layers respectively to unify the number of channels in each feature map. As shown in Fig. 2, despite the same number of channels, the size is still various, and this multi-scale feature representation is expected to help the model to learn features in various scale regions for better prediction.

Stage 3. As shown in Fig. 2, in MiDaS network, it decodes the feature maps in a special way. In particular, the

decoding starts at the smallest feature map, and then this feature map is doubled in shape in order to match the feature map of its upper layer in stage 2. Here, a residual-like connection is added to connect features in different feature maps. The whole process is described as the Feature Fusion Layer. In stage 3, there are 4 such layers to fuse feature maps in different sizes, and the feature map is gradually enlarged at the same time. Finally, the predicted depth map can be obtained when the size reaches the specified one.

Feature fusion layer fuses the outputs from convolution layers and produces resulting depth maps.

Loss Function. The choice of loss function is vital for this structure. Because MiDaS uses a variety of datasets, the problem to be dealt with in training is the diversity of label formats.

According to [14], there are three difficulties : 1) Different datasets represent different depths in different ways. 2) Scale ambiguity: for some data sources, depth is only given up to an unknown scale. 3) Shift for ambiguity: some datasets provide disparity only up to an unknown scale.

Therefore, if all datasets are trained, the loss function should be scale-and shift-invariant. The ideas provided in MiDaS are as follows:

Firstly, the paper adopts the following more flexible expression of the loss function

$$L_{ssi}(\hat{d}, \hat{d}^*) = \frac{1}{2M} \sum_{i=1}^{M} \rho(\hat{d}_i - \hat{d}_i^*) \tag{1}$$

where $\hat{d}$ and $\hat{d}^*$ are the scaled and shifted versions of the predictions and ground truth [14]. ρ represents the type of loss function, where we can select mean-squared error (MSE) in this case. As can be seen, L_{ssi} considers multiple scaled and shifted versions of predictions and ground truth, as it compensates for the incompatibility between different datasets.

Moreover, regularization is proposed as a matching term in the disparity space [9].

$$L_{reg}(\hat{d}, \hat{d}^*) = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{M} (|\nabla_x R_i^k| + |\nabla_y R_i^k|) \tag{2}$$

where $R_i = \hat{d}_i - \hat{d}_i^*$, and R_k is the difference of disparity maps at scale k.

When combined together, the final loss function can be written as:

$$L_l = \frac{1}{N_l} \sum_{n=1}^{N_l} L_{ssi}(\hat{d}, \hat{d}^*) + \alpha L_{reg}(\hat{d}, \hat{d}^*) \tag{3}$$

where N_l is the training set size, and α is selected as 0.5.

We can also use LeRes [21] as a backbone to this project. Refer to Appendix B for further details due to the limitation of pages.

3.2. Double Estimation

The dilemma of low-frequency structural consistency and high-frequency details of estimates shows a result of contradiction. Double estimation is first introduced in [12] to turn this contradiction into supplementary components by taking full advantage of this duality.

To reasonably find out images with two appropriate resolutions for double estimation, [12] discussed the concept of conceptual cues, which can be described as the most relevant pixels for monocular depth estimation in an image. Following their discussions, we find that the edges of images are related to contextual cues. Therefore, we can threshold the RGB gradients to get an approximate edge map for further inference.

Once we get the edge map, we are able to define the maximum resolution $\mathcal{R}_0$ by ensuring that every pixel is within half of the receptive size of the contextual cues. Estimations with resolutions higher than that maximum resolution will lose structural consistency but will have more high-frequency details.

Empirically, we can choose a resolution of image that loses 20 percent of contextual information by the above definition as our high-resolution input. Percentages higher than 20 are seen as harmful to the merging of double estimation. Then we choose an image resolution that is identical to the receptive field size by which the model can better process the global structure of the image as our low-resolution input. These two inputs are merged using Pix2Pix-based merge net which is discussed in Section 3.4.

3.3. Patch Estimation

Following double estimation, patch estimation is another approach defined in [12] to further boost the estimation performance. Usually, the densities of the contextual cues are different across the whole image, and lower contextual cue density dominates the maximum resolution using double estimation. Therefore, for regions with high contextual cue density, we need to use patch estimation to refine the details.

During this process, we go through phases of base estimation, patch selection, patch estimation, and base adjustment. The base estimation is the result of double estimation. In the following phases, the base estimation is modified and merged to generate a final depth map. The patch selection is to select patches with higher contextual cue density from image patches with sizes equal to the receptive field and a 1/3 overlap [12]. The patch estimation is to perform double estimation on selected patches. The estimation results are merged into the based estimation in the base adjustment.

3.4. Merge Net

In the workflow presented in Figure 1, we have to merge two depth estimations. The first one is a lower resolution depth map from a lower resolution image through a depth prediction model. The second one is either a higher resolution image (as described in Section 3.2) or patches (as described in Section 3.3). Through a merge network modified from Pix2Pix model [5] for learning a mapping from input images to output images given paired data. The standard Pix2Pix [5] is modified with a 10-layer U-net [15] to raise the inference resolution to 1024 by 1024. For this project, we used a pre-trained network of the above structure to add details from the high-resolution input to the low-resolution input to generate depth maps of high-frequency details with consistent low-frequency structures.

4. Experiments and Results

In this section, we mainly investigate the influence of image corruption to the monocular depth estimation as a supplementary work to [12]. Therefore, we measure our success from two aspects. The first one is that all of the codes can be successfully re-implemented, and another one is that we can verify the negative influence of noise on the model performance via both quantitative and qualitative experiments.

4.1. Implementation Details

We implement our project with Pytorch framework, and the main learnable component is the Merge Net to adaptively merge two depth estimations between different resolution estimations. Since we aim at investigating the network performance on different hyper-parameters, resolutions, noise and corruption on the NYU-v2 dataset, so we do not train the network or fine-tune the network on the NYU-v2 dataset. We adopt the pre-trained model in [12], where the network is trained with the Adam optimizer at initial learning as 1×10^{-5} with 100 epochs.

4.2. Quantitative Experiments

In quantitative comparison, we adopt three popular metrics to measure the quantitative performance. The first metric we employ is the absolute relative difference (AbsRel) that can be computed by:

$$\text{AbsRel} = \frac{1}{N}\sum \frac{|d_i - d_i^*|}{d_i}, \tag{4}$$

where d_i, d_i^* and N represent the ground-truth depth map, estimated depth map and the number of pixels in the depth map. The second metric we adopt is the root mean square error (RMSE), and it is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum (d_i - d_i^*)^2}. \tag{5}$$

These two metrics directly measure the average L1 and L2 distance between the ground-truth and the estimated depth map. The lower indicates the estimated one is closer to the ground-truth one. In addition, we also use the $\delta_{1.25}$ that is defined as:

$$\text{per. } \max\{\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\} < 1.25, \tag{6}$$

which measures the percentage of pixels ratio between d_i and d_i^* over the threshold, 1.25. Hence, the larger $\delta_{1.25}$ shows more pixels in the estimated one are closer to the ground-truth one.

Figure 3. Visual comparison of estimated depth map under different types of noises.

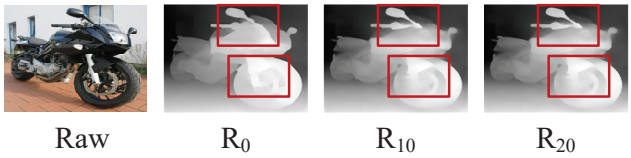

Figure 4. Visual comparison of depth map under different $\mathcal{R}_x$.

Figure 5. The estimated depth maps with/out missing areas.

In our experiments, we adopt the Gaussian noise instead of the Gaussian blurriness used in [12] to further investigate the influence of image corruption on the depth estimation. Since the authors in [12] only consider the corruption as blurriness, which motivates us to try other types of corruption like the Gaussian noise as the Fig. 3 shows. Random noisy pixels are put on the raw image to simulate the corrupted image. The experimental results are reported in the

Models	AbsRel ↓	RMSE ↓	$\delta_{1.25}$ ↓
Boosting MiDaS	1.1800	0.3117	0.8816
Boosting MiDaS with noise (var.=1e-3)	1.3551	0.3321	0.8909
Boosting LeRes	**0.9237**	**0.2699**	**0.8143**
Boosting LeRes with noise (var.=1e-3)	1.1645	0.3164	0.8421

Table 2. Quantitative comparison between the model with/out the Gaussian noise. ↓ indicates lower is better. Var. denotes the variance of the Gaussian noise.

Table 2, where with the Gaussian noise, the performance on two different models drops obviously on all three metrics. This verifies the monocular depth estimation is largely restricted by noise and corruption. Next, we further verify this hypothesis through visual comparison.

4.3. Qualitative Experiments

In this part, we mainly compare the estimated depth maps with various random noises, $\mathcal{R}_x$, and even corruption. The first experiment is established on three different types of noise with a variance of 0.01. The qualitative results are presented in the Fig. 3, from which we could observe the Gaussian noise impacts the depth estimation the most, followed by the speckle noise and salt noise separately. We attribute this result to Gaussian noise replacing the raw pixels with random pixels and damaging the semantics of the raw image largely rather than just putting pure black or white noisy points.

Furthermore, we even simulate a special situation, where free-form masks [1] are generated and masked on the raw image as shown in the Fig. 5. We denoted the masked images as corrupted images, and the corruption is considered as coming from the occlusion in the real-world. From the Fig. 5, we can observe the estimated depth on the occlusion is totally wrong, which means the model fails to estimate the depth of occlusion and even cannot recognize the occlusion. In [12], it has been mentioned one of the limitations of monocular depth estimation is the model can only produce the relative depth instead of the absolute depth. Thus, even though the area is missing, the model still considers the depth to be the same as its neighbors and therefore produces misleading results. This motivates us to make some improvements on the future works.

The final experiment is made on the variable $\mathcal{R}_x$. The experimental results are reported in Fig. 4, where it is obvious that when x increases the details fail to be preserved. That is because the percentage of available contextual information becomes less, and the high-frequency details are lost as a consequence. But, the benefit is the running time becomes less as well. From the observation and considering the results in [12], we consider $\mathcal{R}_x = 20$ can achieve a good trade-off between the running time and the performance.

5. Conclusion

In this report, we re-implement the work in [12] and further explore the influence of different resolution images to the depth estimation. Besides, we also focus on the monocular estimation on corrupted images. We adopt three different types of noises for testing the model and find the monocular depth estimation is largely affected by the Gaussian noise owing to its damage to image semantics. Furthermore, we use random shape masks to simulate the real-world occlusion, and we find the monocular depth model even produces misleading predictions, which is attributed to its heavy reliance on relative cues. Based on our observation, we are going to develop a monocular depth estimation model with strong robustness to noise, corruption, and real-world occlusion, which is assumed as beneficial for the future 3D vision application by providing more reliable depth information even under extreme environments.

6. Work Division

Four people in our team contributed to this project. We sincerely thank the people of our team for their work in this project, from project proposal and data collection, to modeling, evaluation and paper writing. During this whole process, we have had fruitful discussions and come up with many inspirational ideas.

Among our teammates, Lixin Xu chose this topic of depth estimation and contribute constantly to this project, from problem formulation, data collection (NYU-v2), model selection (Boosting MiDaS and LeRes, Pix2Pix merge net), to implementation, parameter-tuning and coordination. Jiacheng Hou provided us with testing ideas of applying corruption and noise to dataset, implemented our testing code and tested our model with NYU-v2 dataset, and analyzed the effect of noise. Julong Li did a thorough research in MiDaS, and analyzed the model structure, selection of loss function and effects of hyper-parameters. Zheyang Liu studied the structure of LeRes, and analyzed the model structure, selection of loss function and effects of hyper-parameters.

We specially thank Professor Zsolt Kira and TAs for their assistance and suggestions during this course.

Student Name	Contributed Aspects	Details
Lixin Xu	Data Collection, Implementation and Analysis	Collected the dataset and pre-trained model for this project, explored and tried different parameters in double estimation and patch selection, and coordinated teamwork.
Jiacheng Hou	Implementation, Evaluation and Experiments	Implemented testing codes, evaluated the model with dataset via experiments and analyzed the behavior of model under noise.
Julong Li	Implementation and Analysis	Refined model structure and analyzed structure, hyperparameters and loss function of MiDaS model.
Zheyang Liu	Implementation and Analysis	Refined model structure and analyzed hyperparameters and loss function of LeRes model.

Table 3. Contributions of team members.

References

[1] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. Free-form video inpainting with 3d gated convolution and temporal patchgan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9066–9075, 2019. 6

[2] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *MIT Press*, 2014. 2

[3] R. Garg, V. K. Bg, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *Springer, Cham*, 2016. 3

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 8

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2, 5, 8

[6] Youngjung Kim, Hyungjoo Jung, Dongbo Min, and Kwanghoon Sohn. Deep monocular depth estimation via integration of global and local predictions. *IEEE transactions on Image Processing*, 27(8):4131–4144, 2018. 2

[7] Y. Kuznietsov, Jrg Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. *IEEE Computer Society*, 2017. 3

[8] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Computer Vision Pattern Recognition*, 2015. 3

[9] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018. 2, 4, 9

[10] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 38(10):2024–2039, 2015. 3

[11] Y Liu, B. Zhuang, C. Shen, H. Chen, and W. Yin. Training compact neural networks via auxiliary overparameterization. 2019. 8

[12] Smh Miangoleh, S. Dille, M. Long, S. Paris, and Y. Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. 2021. 1, 2, 3, 4, 5, 6, 8

[13] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[14] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1, 2, 3, 4, 8

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[16] Chaoyang Wang, Simon Lucey, Federico Perazzi, and Oliver Wang. Web stereo video supervision for depth prediction from dynamic scenes. In *2019 International Conference on 3D Vision (3DV)*, pages 348–357. IEEE, 2019. 2

[17] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2018. 2

[18] K. Xian, J. Zhang, O. Wang, L. Mai, and Z. Cao. Structure-guided ranking loss for single image depth prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 8

[19] S. Xie, R. Girshick, P Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *IEEE*, 2016. 8

[20] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. *IEEE*, 2017. 3

[21] W. Yin, J. Zhang, O. Wang, S. Niklaus, and C. Shen. Learning to recover 3d scene shape from a single image. 2020. 1, 2, 3, 4, 8

[22] Z. Zhang, C. Zhen, C. Xu, Z. Jie, L. Xiang, and Y. Jian. Joint task-recursive learning for semantic segmentation and depth estimation. In *European Conference on Computer Vision*, 2018. 3

A. Project Code Repository

We used the original code implementation of Boosting method [12] as a starting point, and made some modifications to the resolution selection part and refined the code structure. We combined code implementations of MiDaS [14] as the backbone and Pix2Pix [5] as the merge network togerther with the boosting implementation. In addition, we implemented a script to apply noise and corruptions to NYU-v2 dataset, and evaluated the model performance on that dataset.

The link to Boosting Monocular Depth Estimation is at https://github.com/compphoto/BoostingMonocularDepth

The link to MiDaS is at https://github.com/isl-org/MiDaS

The link to Pix2Pix is at https://github.com/phillipi/pix2pix

The repository for our final project is at https://github.com/DavidLXu/CS7643_Project

B. LeRes Model Applied to Boosting

Encoder. In order to restore the shape of a real 3D scene, LeRes uses a 3D point cloud encoder to predict the missing depth transfer and focal length. The point cloud reconstruction module uses the point cloud encoder network to predict the adjustment factors of shift and focal length through the initial value in point cloud reconstruction. These point cloud encoders can be well extended to invisible datasets.

Decoder. The depth prediction architecture [18] used by DPM (depth prediction module) first includes a standard backbone feature extraction (for example, resnet50 [4] or resnext101 [19]), followed by a decoder. The decoder outputs the depth map. In addition, a lightweight auxiliary path [11] is added to the decoder to output the reverse depth. Different losses are enforced on these two branches.

Loss Function. LeRes [21] proposed two new loss functions: image level normalized regression loss (ILNR) and a normal geometric loss (PWN) to enhance the robustness of the depth prediction model trained on the mixed dataset.

Image level normalized regression (ILNR) loss is mainly used to solve the problem that datasets have different depth ranges and web stereo datasets contain unknown depth scales and displacements. The ILNR loss function is as follows:

$$L_{ILNR} = \frac{1}{N}\sum_{i}^{N} |d_i - \overline{d}_i^*| + |\tanh\frac{d_i}{100} - \tanh\frac{\overline{d}_i^*}{100}| \quad (7)$$

In this equation, d is the predicted depth, and d^* is the ground truth depth map [21]. $\overline{d}_i^* = (d_i^* - \mu_{trim})/\sigma_{trim}$, where μ_{trim} and σ_{trim} are the average value and standard deviation of the depth map.

Paired normal regression (PWN) loss is used to improve the geometry of the predicted depth map, this loss takes into account both local and global geometry. The PWN loss function is as follows:

$$L_{PWN} = \frac{1}{N}\sum_{i}^{N} |n_{A_i} \cdot n_{B_i} - n^*_{A_i} \cdot n^*_{B_i}| \tag{8}$$

where n^* denotes ground truth surface normals.

Finally, LeRes also use a multi-scale gradient loss [9]:

$$L_{MSG} = \frac{1}{N}\sum_{i}^{K}\sum_{i}^{N} |\nabla^k_x d_i - \nabla^k_x \overline{d}^*_i| + |\nabla^k_y d_i - \nabla^k_y \overline{d}^*_i| \tag{9}$$

Hence, the final loss function of LeRes is:

$$L = L_{PWN} + \lambda_a L_{ILNR} + \lambda_g L_{MSG} \tag{10}$$

Empirically, λ_a and λ_g can be selected as 1 and 0.5 respectively.