Friedrich-Alexander-Universität

Pattern Recognition Lab

SMMetrDE: Segmentation-guided Monocular Metric Depth Estimation

Projektarbeit in Computer Science

submitted
by

Sandro Sage

born 25.12.2000 in Eggenfelden

Written at

Lehrstuhl für Mustererkennung (Informatik 5)
Department Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg.

Advisor: Dipl.-Inf. Hakan Calim

Started: 01.11.2023

Finished: 30.04.2024

Abstract

The project introduces a framework for combining segmentation and monocular metric depth estimation. Due to the challenges associated with acquiring dense metric depth, several state-of-the-art models have been incorporated in this work and optimized to not only perform relative depth, but more importantly metric depth estimation. A new method is proposed, that takes the segmentation masks of the semantic segmentation model and calculates the object-wise mean absolute distance. The experiments show that relative depth estimation can successfully be transformed into metric scale. Additionally, it is demonstrated that a trade-off between inference time, performance and model complexity has to be concluded. Fine-tuned and larger models tend to have a higher performance than other ones.

Contents

Chapter 1

Introduction

Estimating metric depth, or even relative depth, seems ill-posed without stereo vision to enable triangulation. In early work of computer vision Markov Random Fields (MRF) based formulations have been used to perform monocular depth estimation [Ran+22]. Other approaches have proposed methods for self-supervised monocular training, synchronized stereo pairs or monocular videos [God+19]. Newer methods use zero-shot cross-dataset transfer to achieve a higher generalization, but perform relative depth estimation [Ran+22]. If someone wants to estimate the depth to a specific object, it first has to be localized and detected and afterwards the absolute distance has to be calculated. Objects can be localized by using a detection model or semantic segmentation [Che+18]. This project proposes SMMetrDE - a framework for combining the monocular metric depth estimation with a semantic segmentation network. An automatic process for obtaining the mean absolute distance according to the localized object is established by having the opportunity to choose between various state-of-the-art (SOTA) depth estimators. Several depth estimation models have been considered and evaluated for obtaining metric depth prediction.

Chapter 2

Method

2.1 Segmentation

The segmentation part was already provided by the DeepLabv3+ model which is based on a ResNet50 architecture [Che+18]. The image was accordingly segmented and different segmentation masks depending on their class label were calculated. A deeper insight, on how these segementation masks are used, is shown in Section 2.3.

2.2 Metric Depth by Monocular Depth Estimation

As already discussed in Chapter 1, one has to distinguish between relative and metric depth. Since the goal of the project is to estimate metric depth some model configurations have been adjusted. For this project four different models have been tested and applied:

(1) MonoDepth2 (Metric Depth Estimation (MDE))

(2) MiDaS (Relative Depth Estimation (RDE))

(3) DPT (RDE)

(4) ZoeDepth (MDE)

2.2.1 Monodepth2

The Monodepth2 model consists of some newly invented features:

- minimum reprojection loss for robustly handling occlusions

- full-resolution multi-scale sampling method for reducing artifacts
- auto-masking loss for ignoring training pixels that violate camera motion assumptions

It performs dense depth prediction which means pixel-wise predictions are obtained. The network is based on a U-Net architecture with ResNet18 encoder and ImageNet pretrained weights. The results in the paper show that a combination of monocular and stereo training data leads to an increased accuracy [God+19].

2.2.2 MiDaS (v2.1)

The MiDaS model was trained based on zero-shot cross-dataset transfer, which means that the model was trained using certain datasets and tested on other never seen ones. This results in a improved generalization, since it's not a fine-tuned model as Monodepth2 is trained on KITTI. The effect of using different datasets is that the ground truth depth can be in form of absolute depth, depth up to an unknown scale (from Structure from Motion (SfM)) or even disparity maps [Ran+22]. This leads to the fact that inherently different representations of depth with scale and shift ambiguity exists. Therefore a scale-and shift-invariant loss was invented in the paper. Predictions are performed in disparity space (inverse depth up to scale and shift) and are no longer based on absolute/metric but on relative depth. So to get the metric depth one has to calculate the unknown shift and scale to get the aligned prediction and invert it from disparity space. [Ran+22] gives the equation for calculating the needed parameters:

$$\begin{gathered}(s,t) = \arg\min_{s,t} \sum_{i=1}^{M} (s\mathbf{d}_i + t - \mathbf{d}_\mathbf{i}^*)^2 \\ \hat{\mathbf{d}} = s\mathbf{d} + t, \quad \hat{\mathbf{d}}^* = \mathbf{d}^*\end{gathered} \tag{2.1}$$

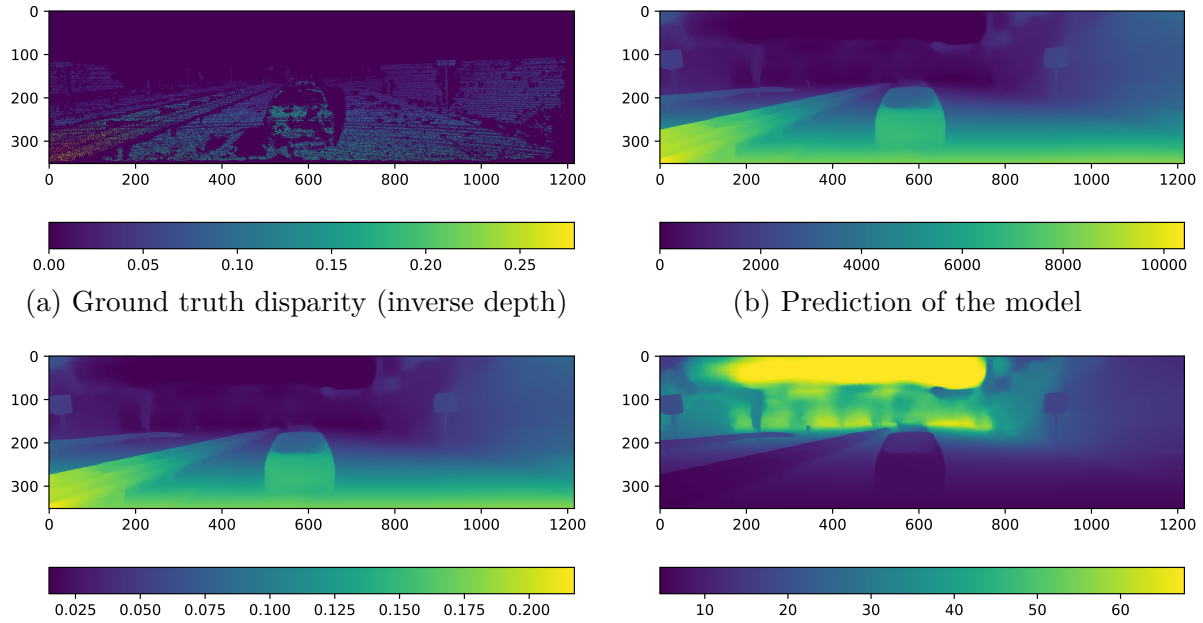
where $\hat{\mathbf{d}}$ and $\hat{\mathbf{d}}$ are the aligned prediction and ground truth, respectively. The factors s and t can be efficiently determined using the closed form from rewriting Equation (2.1):

$$h^{opt} = \left(\sum_{i=1}^{M} \vec{d}_\mathbf{i} \vec{d}_\mathbf{i}^\top \right)^{-1} \left(\sum_{i=1}^{M} \vec{d}_\mathbf{i} d_i^* \right) \tag{2.2}$$

where $h^{opt} = (s,t)^\top$ and $\vec{d}_i = (d_i, 1)^\top$. The models are build based on a ResNeXt-101-WSL encoder with ImageNet pretrained weights [Ran+22].

2.2.3 DPT (v3.0)

In the next version of MidaS, Dense Prediction Transformers (DPTs) were introduced. The models are no longer based on an convolutional approach, but they use encoder-decoder design where the backbone architecture is build on a Vision Transformer (ViT). These transformer-based approaches entail a global receptive field compared to locality. The networks are trained with an affine-invariant loss which was already developed in the MiDaS paper [Ran+21; Ran+22]. One can calculate the metric depth in the same way as described in Equation (2.2). There also exist fine-tuned models on KITTI and NYU using the DPT-hybrid model [Ran+21].



(a) Ground truth disparity (inverse depth)

(b) Prediction of the model

(c) Aligned prediction with s and t

(d) Metric depth

Figure 2.1: Visualization of steps to calculate the affine-transformation parameters (scale s, translation t) for the *MiDaS_dpt_beit_large_384* model. The inverted ground truth is used to align the prediction of the model. Afterwards the aligned disparity prediction is inverted to metric scale.

2.2.4 MiDaS (v3.1)

The newest version of MiDaS introduced new models based on different encoder architectures. The motivation for this was the success of transformers in computer vision. While v3.0 uses the vanilla ViT which was already discussed in the previous part, more newly invented

encoder types like Bidirectional Encoder representation from Image Transformers (BEiT), Swin, Swin2, Next-ViT and LeViT have been applied. These encoder types are incorporated into the existing MiDaS approach.The BEiT encoder provides the highest quality, whereas Swin achieves the second highest quality. For the project some of these model architectures have been applied and will be listed in the next section [Bir+23].

2.2.5 ZoeDepth

The ZoeDepth model is a two-stage framework where the first stage is the MiDaS depth estimation framework and the second one adds heads for metric depth estimation to the encoder-decoder design. This setup is then fine-tuned on metric depth datasets. The encoders are based on the DPT and BEiT architecture. In the paper a metric bins module is is introduced which is inspired by the LocalBins architecture. Heads are added by the metric bins module and four hierarchy levels of MiDaS decoder are hooked into it to perform metric depth estimation [Bha+23].

2.3 Implementation details

The project is organized in different components separated by their processing steps. Figure 2.2 shows the pipeline of processing the input data and getting the output results. The *InputHandler* deals with handling all the input- and output paths and prepares the information for the main component - the *Processor*. This component includes several subcomponents like the *Segmentator*, *DepthEstimator* and the *PerformanceTimer*. It operates as an interface between the subparts and performs the final output results. The input image is sequentially forwarded to the *Segmentator* and the *DepthEstimator*.

While the *Segmentator* predicts the pixel-wise segmentation labels, the *DepthEstimator* predicts the according pixel-wise metric depth of the input image. If the model is based on an RDE approach, it first has to calibrate itself and calculate the unknown shift and scale parameters by using Equation (2.2). Therefore the depth map has to be converted into disparity space (inverse depth) and it must be ensured that only depth values with known ground truth are considered for the alignment process. To obtain the metric depth prediction the aligned prediction (still in disparity space) has to be inverted. These steps are visually represented in Figure 2.1. Aligned depth values greater than 80 meters are set to this threshold. After the alignment process, the segmentation prediction is transformed

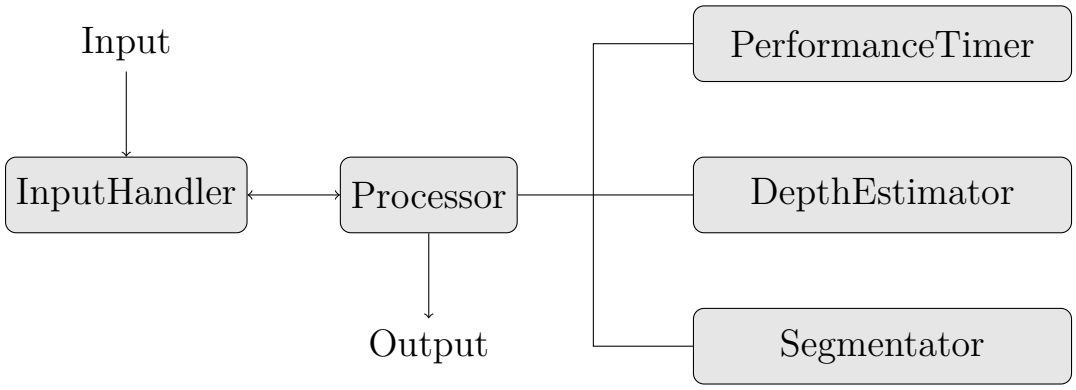


Figure 2.2: Structure and processing pipeline of the project. The *Processor* has several subcomponents like *PerformanceTimer*, *DepthEstimator* and *Segmentator*. The *InputHandler* provides the *Processor* with all the necessary informations.

into segmentation masks. Only segmentation classes apparent in the image are regarded for this transformation. The segmentation masks are used to extract the relevant depth pixels in the metric depth prediction of the *DepthEstimator*. Afterwards the mean of the masked depth pixels is calculated and concatenated with the segmentation class label. These results are stored in a csv-File to easily adapt it into other applications. Meanwhile in the background, the *PerformanceTimer* is measuring the inference time of the *Segmentator* and the *DepthEstimator*. A feature of the *PerformanceTimer* is that it calibrates itself according to the measured value of the first inference process of the model. If the measured value is smaller than 1000 milliseconds, the unit remains milliseconds, otherwise it is transformed into seconds. After processing all the input images some generic features, e.g. mean, standard deviation and max, of the recorded inference times are calculated for inspection. All the results of the *PerformanceTimer* are visualized and stored.

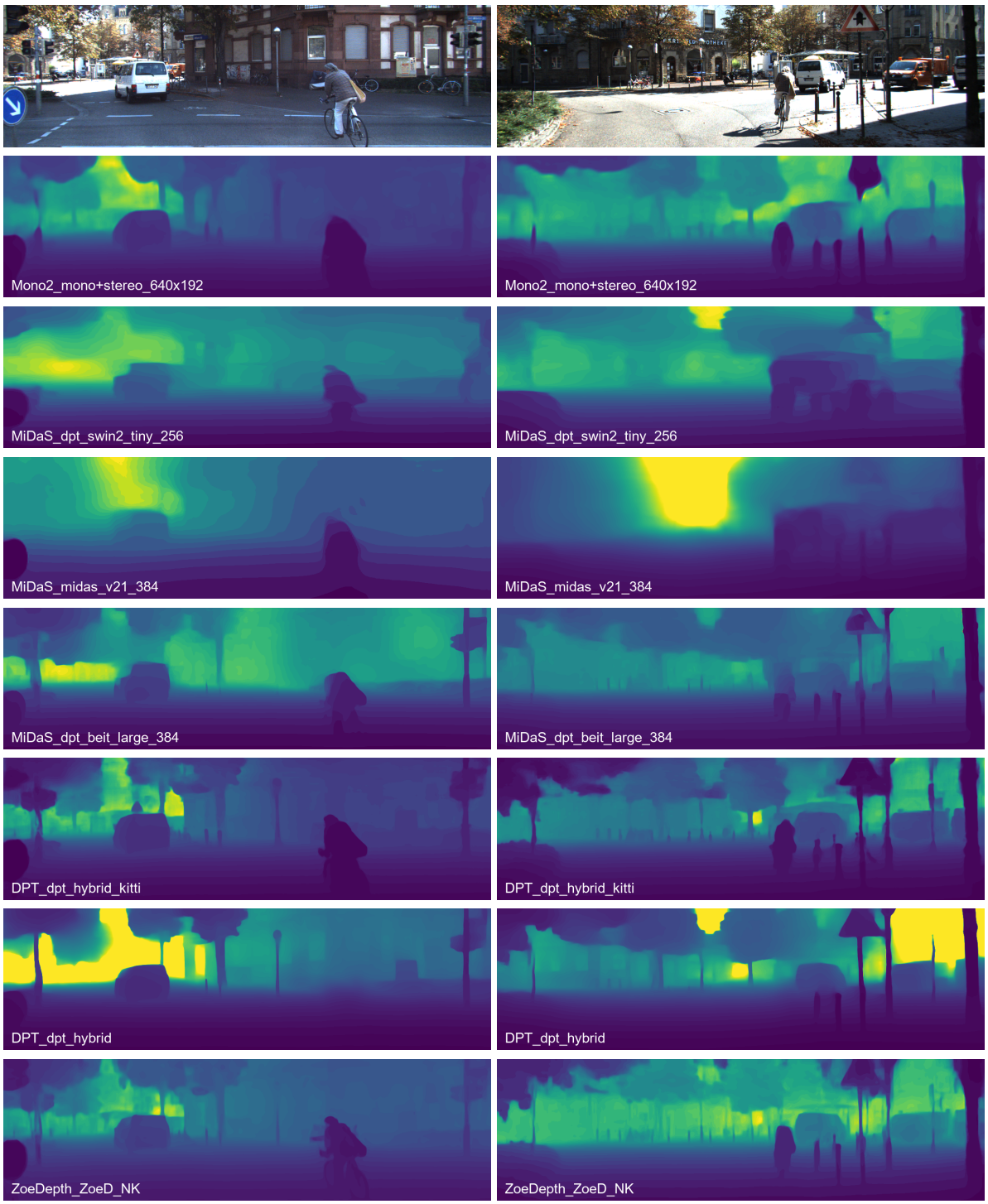


Figure 2.3: Comparison of metric depth prediction of evaluated models. The model type can be identified by the small annotation in the left bottom corner. The top image is the input and the ones below are the metric depth predictions of the models. One row represents the correspondence to the same model type.

Chapter 3

Experiments

For the setup of this project a AMD Ryzen 5 5500U with Radeon Graphics, 2100 MHz, 6 cores and 12 logical processors was used. The dataset for the evaluation is based on the manually selected validation and test sets of KITTI Depth Prediction Evaluation [Uhr+17]. A subset of 200 images and ground truth depth maps was used for the evaluation process. An illustration of two example images of the dataset and their predictions according to the model type is represented in Figure 2.3. The calculation of the affine-transformation parameters is based on one image of the previously mentioned dataset (2011_10_03_drive_0047_sync_image_0000000791_image_03.png). The crop region for the predictions and depth maps is obtained by using the Garg-crop [Gar+16]. In addition, only pixels with valid ground truth depth are extracted by using a mask over the predictions and the ground truth depth maps. For the evaluation the minimum depth is set to 10^{-3} meters and the maximum depth is limited by 80 meters.

For the Monodepth2 model only one type was evaluated. Since the aim is to perform a metric depth prediction, the implementation with the best performance for this use case is the *Mono2_mono+stereo_640x192* [God+19]. For the MiDaS depth estimators, 3 different model types were considered. One baseline model *MiDaS_midas_v21_384* from MiDaS (v2.1) and two newer models (*MiDaS_dpt_swin2_tiny_256*, *MiDaS_dpt_beit_large_384*) with updated encoder backbones, that are no longer convolutional, were evaluated. Only the *DPT_dpt_hybrid_kitti* and the *DPT_dpt_hybrid* were taken into account for the DPT-approach, due to inference time. *ZoeDepth_ZoeD_NK* was also considered as an alternative implementation of the MiDaS baseline, since it directly performs metric depth estimation. The abbreviation *NK* stands for training on NYU and KITTI dataset [Bha+23]. The models

Table 3.1: Comparison of evaluated models. Best results in each category are in bold; second best are underlined. In the most right column the Mean Inference Time (MIT) is indicated according to seconds (s) and milliseconds (ms). The asterisk * refers to unit in seconds (s).

Model	δ_1	δ_2	δ_3	AbsRel	RMSE	MIT / ms
Mono2_mono+stereo_640x192	0.668	0.828	0.908	0.228	7.596	**149.99**
MiDaS_dpt_swin2_tiny_256	<u>0.673</u>	0.849	0.929	**0.218**	**6.747**	489.06
MiDaS_midas_v21_384	0.238	0.492	0.701	0.532	13.331	<u>374.10</u>
MiDaS_dpt_beit_large_384	0.600	**0.854**	<u>0.930</u>	0.236	<u>6.862</u>	29.59*
DPT_dpt_hybrid_kitti	**0.674**	0.830	0.913	<u>0.223</u>	7.410	6.78*
DPT_dpt_hybrid	0.629	0.828	0.910	0.318	10.937	8.66*
ZoeDepth_ZoeD_NK	0.640	<u>0.852</u>	**0.935**	0.279	7.181	56.75*

were evaluated against several performance indicator like the absolute value of the relative error (AbsRel), the percentage of bad depth pixels δ with different thresholds, the root mean square error (RMSE) and the MIT. The best results in each category are written in bold, whereas the second best are underlined. As visualized, there is a big differentiation between the MIT, since smaller models, that are mostly based on a convolutional encoder, tend to have a smaller inference time than others. *ZoeDepth_ZoeD_NK* has the highest inference time and is therefore not suited for applications that require a relatively low execution time. The model that performs the worst is the *MiDaS_midas_v21_384*. There is no clearly recognisable winner, but larger models that area also fine-tuned on the KITTI dataset tend to have better results. In general, one also has to keep in mind that some of the models are only trained on a specific dataset, whereas others are trained on a variety of datasets and partially fine-tuned.

Chapter 4

Conclusion

The project has presented various types of SOTA-models and how to evaluate them based on monocular metric depth estimation. It also gives a framework for integrating all the models and how to combine them with a semantic segmentation model to obtain absolute depth distance to localized objects. For monocular depth estimation, it is believed that existing datasets are still insufficient and likely constitute the limiting factor of robust depth estimation [Ran+22]. Possible further experiments could be a fine-tuned training on a self-recorded dataset for the specific application, a more general evaluation on various datasets or the consideration of newly proposed approaches like Depth Anything [Yan+24]. The code for this project is available at https://github.com/sandrosage/Monocular_Depth_Estimation.

List of Abbreviations

DPT Dense Prediction Transformer

RDE Relative Depth Estimation

MDE Metric Depth Estimation

ViT Vision Transformer

SfM Structure from Motion

MIT Mean Inference Time

SOTA state-of-the-art

MRF Markov Random Fields

BEiT Bidirectional Encoder representation from Image Transformers

List of Figures

List of Tables

Bibliography

[Bha+23] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller. Zoedepth: zero-shot transfer by combining relative and metric depth, 2023. arXiv: 2302.12288 [cs.CV] (cited on pp. 6, 9).

[Bir+23] R. Birkl, D. Wofk, and M. Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023 (cited on p. 6).

[Che+18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018. arXiv: 1802.02611 [cs.CV] (cited on pp. 1, 3).

[Gar+16] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: geometry to the rescue, 2016. arXiv: 1603.04992 [cs.CV] (cited on p. 9).

[God+19] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth prediction, 2019 (cited on pp. 1, 4, 9).

[Ran+21] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. *ICCV*, 2021 (cited on p. 5).

[Ran+22] 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*, 44(3), 2022 (cited on pp. 1, 4, 5, 11).

[Uhr+17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017 (cited on p. 9).

[Yan+24] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: unleashing the power of large-scale unlabeled data, 2024. arXiv: 2401.10891 [cs.CV] (cited on p. 11).