Uncertainty in Self-supervised Depth Estimation Using Multi-scale Decoders

Mayank Mali

CMU-CS-22-124

August 2022

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

Thesis Committee:
Dr. Jean Oh (advisor)
Dr. Ji Zhang

Submitted in partial fulfillment of the Masters degree in Computer Science

Keywords: Machine Learning, Depth Estimation, Self-Supervised, Uncertainty, Uncertainty Estimation, View Synthesis, Sparsification, Student-Teacher Framework, U-Net, Decoder, Multi-Scale, ResNet

For my family, friends, and faculty that offered endless support and feedback.

Abstract

Depth estimation is an image translation problem that predicts depth-maps for a given camera image, and has fostered research in various applications including self-driving vehicles. Self-supervised depth estimation methods are of particular interest since ground truth LIDAR depth is expensive to acquire and instead use view synthesis as weaker supervision. Generally, the produced depth maps to date are only point estimates of an underlying depth distribution due to randomness in model training, resulting in noisy depth estimates that can propagate errors and lead to inaccurate or fatal decisions in real-world applications. Recent interest has been sparked in reducing such noise by modeling the uncertainty of depth estimates. Empirical uncertainty strategies seek to predict uncertainty via statistical methods on treating independent models as black-box predictors. Of particular interest are predictive strategies that seek to learn the inherent uncertainty of a depth model. For example, student-teacher frameworks train one network to learn the depth output distribution of another. Such methods are desirable due to the advantage of requiring fewer training and space resources compared to other empirical methods. In this work, we study self-supervised depth models with a U-Net architecture that output depths at multiple scales. In particular, we explore a novel predictive uncertainty model that only has access to these scales and the U-Net bottleneck feature. We evaluate and discuss the novel method alongside other uncertainty strategies on the KITTI dataset.

Acknowledgments

Very special thanks to Dr. Soonmin Hwang for investing generous time in mentoring, teaching, and encouragement, extending the limits of my self-confidence and accelerating my understanding of depth estimation. Warm thanks to my advisor Dr. Jean Oh for much-needed guidance, direction, and support. Also thanks to Dr. Ji Zhang for giving time to be part of the thesis committee.

I also extend my gratitude to all my peers in the Bot Intelligence Group at CMU, especially Ingrid Navarro, for their valuable time and effort in providing feedback for this project.

Contents

1 Introduction **1**
- 1.1 Self-supervised Depth Estimation . . . 1
- 1.2 Uncertainty for self-supervised depth estimates . . . 2
  - 1.2.1 Why uncertainty in self-supervised depth estimation? . . . 2
  - 1.2.2 Problems with monodepth: uncertainty to the rescue? . . . 2

2 Related Works **3**
- 2.1 Self-supervised monodepth . . . 3
  - 2.1.1 View-synthesis pipeline . . . 3
  - 2.1.2 Problems with monodepth . . . 5
- 2.2 Predictive Uncertainty . . . 7
  - 2.2.1 Gray-box uncertainty estimation . . . 7
  - 2.2.2 Learning uncertainty . . . 7
- 2.3 Empirical uncertainty . . . 8
  - 2.3.1 Bootstrap ensembles . . . 8
  - 2.3.2 Snapshot ensembles . . . 9

3 Predicting Uncertainty with Depth Scales **11**
- 3.1 Empirical Uncertainty from depth scales . . . 11
- 3.2 Predictive Uncertainty from depth scales (ScaleDecoder) . . . 12
- 3.3 ScaleDecoder variants . . . 12

4 Experimental Results **13**
- 4.1 Metrics . . . 13
  - 4.1.1 Motivation . . . 13
  - 4.1.2 Depth Metrics . . . 13
  - 4.1.3 Uncertainty Metrics . . . 14
- 4.2 Data . . . 14
  - 4.2.1 Qualitative data . . . 15
  - 4.2.2 Quantitiative data . . . 15

5 Discussion **19**

6 Conclusion **21**

7 Future Works **23**

Bibliography **25**

List of Figures

1.1 **Monocular Depth Estimation.** Monodepth (monocular depth estimation) is the problem of predicting pixel-wise depth-maps given a single camera image. 1

2.1 **Model architecture for view synthesis.** Given two camera images, I_t, I_c, $\widetilde{I_c}$ is synthesized from I_t, predicted depth d_t of I_t, and transformation between the views $T_{t \to c}$. Photometric loss L_{photo} compares I_c and $\widetilde{I_c}$, jointly supervising d_t and $T_{t \to c}$ (if learned). 4

2.2 **Depth predictions on non-Lambertian surface.** The depthmap estimate (bottom) for the camera image (top) shows that depth estimates are inaccurate for the front window (non-Lambertian surface) of the vehicle, which illustrates the problem predicting depth with broken photometric assumptions. 6

2.3 **Flying points in 3D reconstruction.** Flying points in the reconstruction (right) can be seen corresponding to the boundary of the vehicle (top left). Depthmap is shown bottom left. 7

2.4 **Dropout sampling** Using random dropout to zero-out certain neurons. Uncertainty is taken as variance across N independent dropout forward passes with the *same* model after training. 8

2.5 **Bootstrap ensembles.** Using variance across bootstrap ensembles as uncertainty. 9

2.6 **Snapshot ensembles.** Using variance across snapshot ensembles as uncertainty. Snapshots of the model are taken at low points of a cyclic learning rate throughout a single training session. 10

3.1 **Basic ScaleDecoder architecture.** The ScaleDecoder (bottom) takes the four depth scales of the depth decoder (top) and the U-Net bottlenet to predict parameters $\mu(d_S), \sigma(d_S)$ for d_T's underlying distribution for the given input camera image I_t. 11

List of Tables

4.1 **Scale-variant depth metrics.** 13
4.2 **Metrics on uncertainty-based depth predictions.** The sections from top to bottom are raw depth, post-processed depth, ground-truth median-scaled depth, and ground-truth median-scaled depth, post processed depth. Best metrics are bolded and best metrics among the novel ScaleDecoder (SD) family are underlined. *All teaching variants have the same depth metrics. **The joint strategy reported is tested with a smaller posenet encoder. 16
4.3 **Metrics on uncertainty predictions.** The sections from top to bottom are raw depth, post-processed depth, ground-truth median-scaled depth, and ground-truth median-scaled post-processed depth. Best metrics are bolded. Best metric for novel ScaleDecoder (SD) among variants are underlined. *The joint strategy reported is tested with a smaller posenet encoder. 17

Chapter 1

Introduction

1.1 Self-supervised Depth Estimation

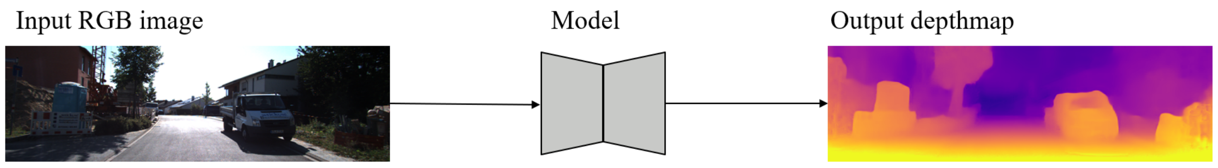

Figure 1.1: **Monocular Depth Estimation.** Monodepth (monocular depth estimation) is the problem of predicting pixel-wise depth-maps given a single camera image.

The problem of monocular depth estimation is to predict depthmaps from a given camera image as shown in Figure 1.1, where the underlying premise is that single images of an indoor/outdoor 3D scene contains various depth cues (i.e. object spatial arrangement, textures) that encode the distance from the camera i.e. depth. Furthermore, if camera images are collected via video or as stereo image pairs, then motion parallax between frames or stereo parallax respectively can give further depth cues.

The significance of this challenge is that pixel-wise depthmaps are used to reconstruct and understand the 3D environment at the time of the camera image. Knowing the depths of an object and the geometry of the camera, allows a geometric approach to reconstruct the 3D scene. This task is particularly interesting for self-driving car applications [9].

However, ground truth depth points, usually from LIDAR (Light Detection and Ranging) hardware, are expensive to acquire. To address this problem, many works use view synthesis which is a geometry-based supervision. Sometimes called warping or reconstruction, this pipeline takes a camera image of a 3D scene from one viewpoint (i.e. camera pose) and predicts the image of the same scene for another viewpoint. For example, the KITTI dataset [5] offers stereo camera image pairs as the two views. Another option is to collect monocular video, and use different frames of time as the viewpoints [17]. The latter technique is called SfM (Structure from Motion) and involves using visual odometry to understand the transformation between viewpoints.

1.2 Uncertainty for self-supervised depth estimates

However, predicted depthmaps are point estimates No basis on whether it should be trusted [13] Underlying depth output distribution, over independent model training Depth noise from dataset (aleatoric) and model–inherent (epistemic) noises

Uncertainty := the variation for prediction estimates

The goal of the uncertainty task is to predict the variation in the depth estimate, dependent on the model.

Motivation: can we remove uncertain points by uncertainty-filtering? Hypothesis: If uncertainty sufficiently encodes errors, we can improve depth accuracy by filtering highest-uncertainty points

Importance: In self-driving deployment, uncertainty in predictions can lead to fatal decisions In robotics, keeping most certain points for successive downstream tasks

1.2.1 Why uncertainty in self-supervised depth estimation?

Uncertainty literature for self-supervised depth relatively new

Self-supervised loss based on view synthesis technique and photometric loss, based on reference and warped images, no ground truth.

Photometric loss based on structural similarity of images Does not penalize network as much for small warping mistakes

Takeaways: Self-supervised training is weaker supervision signal Does not directly supervise depth, only by photometric loss Creating a more accurate model by predicting uncertainty is cutting-edge in self-supervised depth estimation

1.2.2 Problems with monodepth: uncertainty to the rescue?

Depth continuities between objects

Infinitely far away objects (e.g. sky)

Scale ambiguity. Pose transformation $T_{t \to c}$ predicted

Nowhere in view synthesis pipeline are metric units introduced

Self-learned scale alleviated by median-scaling by ground truth

New prediction is $d_t^{scaled} = d_t \cdot \frac{median(d^*)}{median(d_t)}$

Still, could use mean-scaling, etc. [15].

What is needed for scale-awareness? Stereo, full-surround rigs have fixed, metric distances between cameras [10] I.e. known, metric camera extrinsics.

Chapter 2

Related Works

2.1 Self-supervised monodepth

Due to the high cost of LIDAR ground truth depths and ease of capturing many images via video, much attention has been paid to self-supervised methods that use warping, or view synthesis [1, 4, 6, 7, 8, 9, 10, 14, 16, 17], a geometric pipeline that can supervise depth by transforming one camera image into another.

During training, a pair of images I_t, I_c from two different camera poses are offered either as stereo camera pairs [4, 6], or as successive frames in a video [7, 8, 14, 17]. The supervision signal comes by using predicted depth-map d_t of I_t to transform (also called warping) pixels $p \in I_t$ into the pose of the I_c, and then comparing the warped image $\widetilde{I_c}$ with I_c using photometric loss. This is a window-based loss that penalizes structural differences of the two images $\widetilde{I_c}, I_c$, meaning if the same neighborhood of pixels for the two images have similar spatial arrangements, then the loss contribution from that region is smaller. Photometric loss is defined in terms of the Structural Similarity (SSIM) and L1 loss of $\widetilde{I_c}, I_c$:

$$\mathcal{L}_{photo} = \alpha \cdot \frac{1 - \mathrm{SSIM}(\widetilde{I_c}, I_c)}{2} + (1 - \alpha) \cdot |\widetilde{I_c} - I_c|_1,$$

where α is a relative weighting term.

2.1.1 View-synthesis pipeline

In the geometry-based view synthesis pipeline, pixels from I_t must be projected from pixel-space to 3D space, an operation called *unprojection* since projection is used to denote the reverse operation. The unprojection operation ϕ uses the predicted depth d_t and the camera's inverse intrinsics K^{-1}, where the intrinsic K is a matrix to project points onto the image of a pinhole camera [14]. K is determined by the focal lengths f_x, f_y and pixel center c_x, c_y of the camera geometry:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

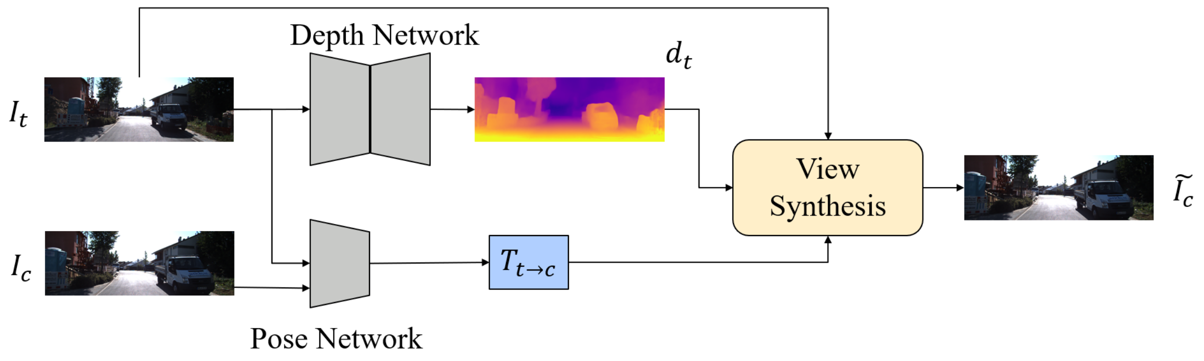

Figure 2.1: **Model architecture for view synthesis.** Given two camera images, I_t, I_c, $\widetilde{I}_c$ is synthesized from I_t, predicted depth d_t of I_t, and transformation between the views $T_{t\to c}$. Photometric loss L_{photo} compares I_c and $\widetilde{I}_c$, jointly supervising d_t and $T_{t\to c}$ (if learned).

Once in 3D space in I_t's camera reference frame, it is transformed to I_c's camera reference frame by either additionally predicting the transformation pose $T_{t\to c}$ between the frames in 3D space via a pose model that maps $(I_t, I_c) \mapsto T_{t\to c}$ or calculating it based on odometry hardware. In the case where I_t, I_c are from two separate and fixed cameras (e.g. stereo, full-surround rig [10]), camera extrinsics (the transformation from world to camera reference frame) are used instead of predicting the transformation between them. Finally, the 3D point in I_c's reference frame is projected to I_c's camera via its intrinsics K (often cameras are assumed to have identical pinhole geometries). The last step to transform pixels to their final location in the synthesized image is called projection and denoted π. The pipeline is summarized below:

1. Predict depthmap d_t from image I_t using a depth network.
2. Unproject points as homogenous coordinates $p = (u, v, 1)$ in I_t to I_c's 3D reference frame using d_t and camera inverse intrinsics K^{-1}:

$$\phi(p, d_t) = d_t K^{-1} I_t(p)$$

3. Transform 3D points $\phi(p, d_t)$ to I_c's camera reference frame.

$$P = T_{t\to c}\phi(p, d_t)$$

4. Project 3D points in I_c's reference frame to I_c's camera with camera intrinsics K:

$$\widetilde{I}_c(p) = \pi(P) = \frac{1}{P_z} K P = \frac{1}{P_z} K T_{t\to c} d_t K^{-1} I_t(p)$$

The prediction image $\widetilde{I_c}$ is compared with I_c using SSIM-based photometric loss.

It is important to note that due to the geometric grounding of the view synthesis pipeline, we can generalize this process to datasets with stereo video, and even videos from multi-camera rigs. Once unprojected via ϕ, we can chain transformations between cameras *and* between time frames simultaneously (e.g. via an additional matrix multiplication step). A viewpoint is generalized to any camera image in any time, and additional supervision comes from multiple photometric losses from reconstructions between these viewpoints.

Even more surprising is that some works have shown that we can predict the camera intrinsics K if unknown [1]. Furthermore, even with cameras without pinhole geometries (e.g. fish-eye camera) or with unknown geometries, a method called NRS (Neural Ray Surfaces) [14] is used to learn the unprojection and projection operations ϕ, π themselves.

During evaluation, the depth model is considered separately to make depth predictions for single-camera images (monodepth).

2.1.2 Problems with monodepth

Photometric assumptions. There are a variety of assumptions made about the scene that when broken, can affect the quality of depth estimates. One assumption is that the scene is rigid (no dynamic objects), so that the car's ego-motion is the only moving component of the scene. Some works have been able to address this limitation by predicting dynamic motion using optical flow [1] and by masking dynamic objects completely [8]. Either way, the view synthesis pipeline becomes increasingly complex. Another assumption is about how the scene objects reflect light. non-Lambertian surfaces like mirrors or glass affect the pixel brightness differently, and depth predictions for those objects are often incorrect, as shown in Figure 2.2.

Photometric loss. The tradeoff between ground-truth LIDAR points for self-supervised photometric loss is that the latter is a weaker training signal. Since photometric loss is based on structural similarity of points around the same neighborhoods of the two photos $\widetilde{I_c}, I_c$, view synthesis operations have more leniency with a less strict loss to warp all windows of the photo correctly. Therefore, self-supervised depth estimates suffer reduced accuracy and are amenable to possible post-processing.

Scale-ambiguity for monodepth. In the monodepth view synthesis pipeline as shown in Figure 2.1, the units of scale for depth estimates are inherent to the model since nowhere are metric units reinforced. Even the pose network estimate $T_{t\to c}$ is not supervised with metric units (unless ground-truth pose from odometry is used). In other words, there is some freedom with what scale the network learns to predict depth; such methods that don't constrain the depth scale are called scale-ambiguous. When depth estimates are in a metric scale, (when metric units enter the pipeline though known, fixed camera extrinsics in e.g. stereo, FSM), such methods are called scale-aware.

There are ways to alleviate scale-ambiguity. Median-scaling [15] is a method to at least match the median depth of the estimate with the median depth of ground truth: simply multiply ground truth estimates by a scale factor:

$$d_t^{scaled} = d_t \cdot \frac{\text{median}(d^*)}{\text{median}(d_t)}.$$

Another such method is mean-scaling, where the scale factor is $\mathrm{mean}(d^*)/\mathrm{mean}(d_t)$.

Still, ground truth depth are required to do median-scaling, and even with median-scaled estimates, the model may still produce different distributions of depth for an image than what is in ground truth.

3D reconstructions. With accurate depth estimates, reconstructing the scene in 3D can still reveal "flying points", where the estimated depth at that pixel does not unproject it to the correct object as shown in Figure 2.3. In essence, this reveals the problem of depth estimates along object boundaries. Without additional processing or semantic segmentation, such boundaries present problems without explicit handling. Some works add a smoothness loss to penalize depth gradients where image sharpness decreases with an "edge-aware" term [6, 7, 14, 16, 17]. Still, recognizing and explicitly handling such points is of crucial importance when the depth network is provided as a black-box for downstream applications.

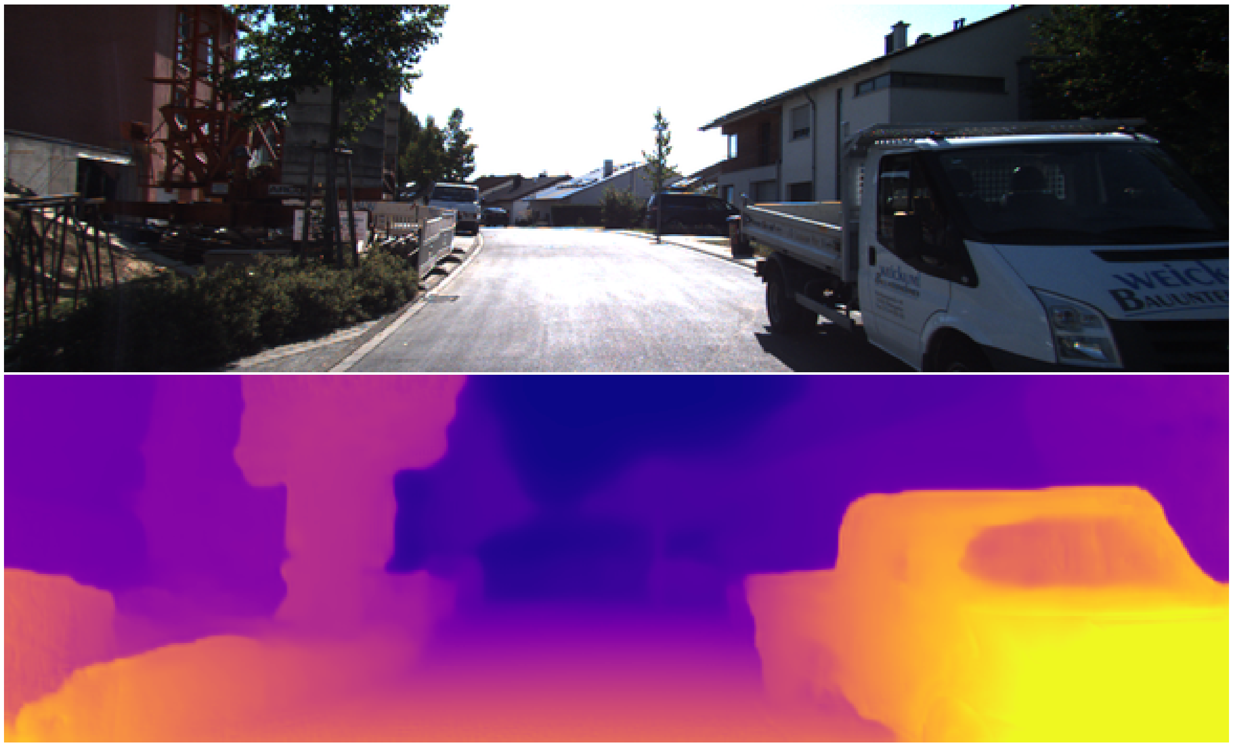

Figure 2.2: **Depth predictions on non-Lambertian surface.** The depthmap estimate (bottom) for the camera image (top) shows that depth estimates are inaccurate for the front window (non-Lambertian surface) of the vehicle, which illustrates the problem predicting depth with broken photometric assumptions.

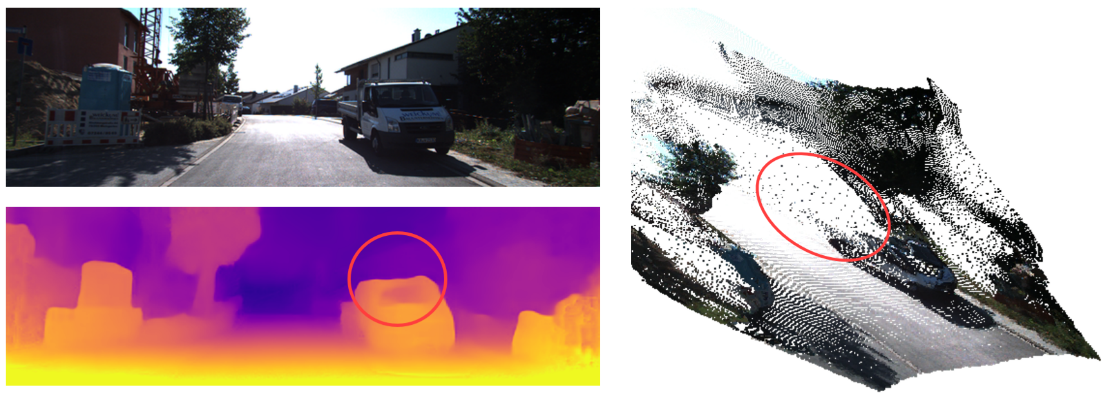

Figure 2.3: **Flying points in 3D reconstruction.** Flying points in the reconstruction (right) can be seen corresponding to the boundary of the vehicle (top left). Depthmap is shown bottom left.

2.2 Predictive Uncertainty

2.2.1 Gray-box uncertainty estimation

Black Box Uncertainty Estimation (BBUE) strategies, like empirical strategies, only have access to the model as a black-box. An example of an BBUE strategy is to perturb the input image and observe the change in output depth. Some perturbation T (rotation, flip, translation, etc.) is applied to input I_t, then the inverse transformation T^{-1} is applied to output. Many observations by randomizing the perturbation can be made this way, and the variance along the un-transformed outputs is taken. Still, N observations cost N forward passes of the same model with a random transformation T, T^{-1}.

Gray Box Uncertainty Estimation (GBUE) strategies have access to intermediate layers of network (but not the parameters). **Feature noise**: During a forward pass for depth estimation, Gaussian noise is added to the feature maps of network's intermediate layers. In this case, each depth observation costs one forward pass, and each observation

Dropout: Randomly dropout neurons for each of multiple forward passes as shown in Figure 2.4.

2.2.2 Learning uncertainty

Self-teaching. Assume output d_T as underlying distribution with pixel-wise mean, standard deviation for a given input I. Uncertainty network S estimates $\mu(d_S), \sigma(d_S)$ from I.

Maximum likelihood estimation (MLE) finds most probable parameters of some distribution family given data. Equivalent problem: minimize negative log likelihood

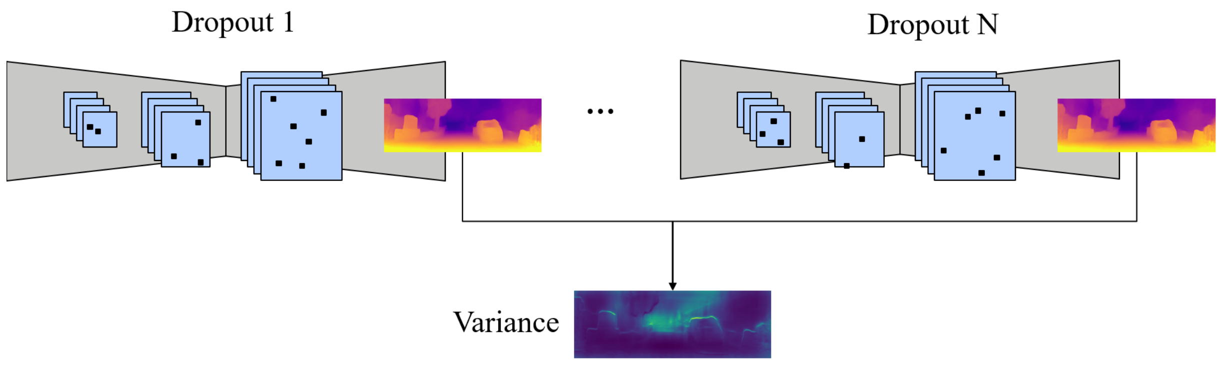

Figure 2.4: **Dropout sampling** Using random dropout to zero-out certain neurons. Uncertainty is taken as variance across N independent dropout forward passes with the *same* model after training.

Assume depth distribution of d_t is Laplacian distribution. Want to find $\mu(d_S), \sigma(d_S)$ of distribution that minimizes L1 loss [11]

$$\mathcal{L}_{Self} = \frac{|\mu(d_S) - d_T|}{\sigma(d_S)} + \log \sigma(d_S).$$

Assuming Gaussian distribution, use L2 loss.

With uncertainty outputs $\mu(d_S), \sigma(d_S)$ we can train uncertainty network to model predicted depth distribution of d_T.

Significance: Normally ground truth depth d^* used instead of estimate d_T [12].

2.3 Empirical uncertainty

Empirical methods use many models as black-box predictors, and use the mean and variance of the predictions to form parameter estimates of the depth distribution. We will describe two that we use, bootstrap ensembles and snapshot ensembles [12].

2.3.1 Bootstrap ensembles

One straightforward way to use black-box models, is to independently train N depth networks. The uncertainty is defined as the variance of the independent depth estimates on a single image

2.5. However, this method requires training and storing N independent models and performing N forward passes, which is time and resource intensive.

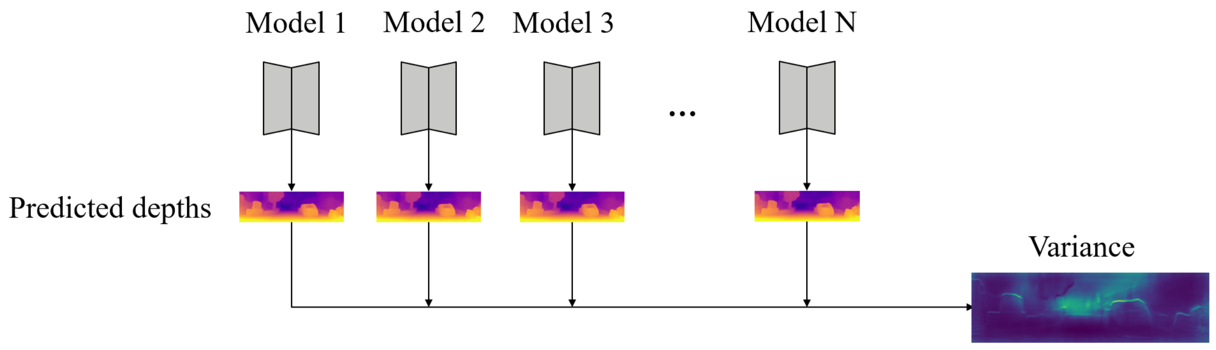

Figure 2.5: **Bootstrap ensembles.** Using variance across bootstrap ensembles as uncertainty.

2.3.2 Snapshot ensembles

Training N independent models is resource and time intensive. Instead, snapshot ensembles [12] trains a single depth network under a cyclic learning-rate throughout training[1]. Every time the learning-rate hits its lower bound, the model weights are saved (a single snapshot). The uncertainty is defined as the variance across depth estimates of these snapshots, as shown in Figure 2.6. It is important to note that in this strategy, model independence is an approximation, which is traded for performing a single training cycle instead of N. Still, N forward passes are needed.

[1]As a technical detail, the learning rate scheduler is updated every batch instead of every epoch.

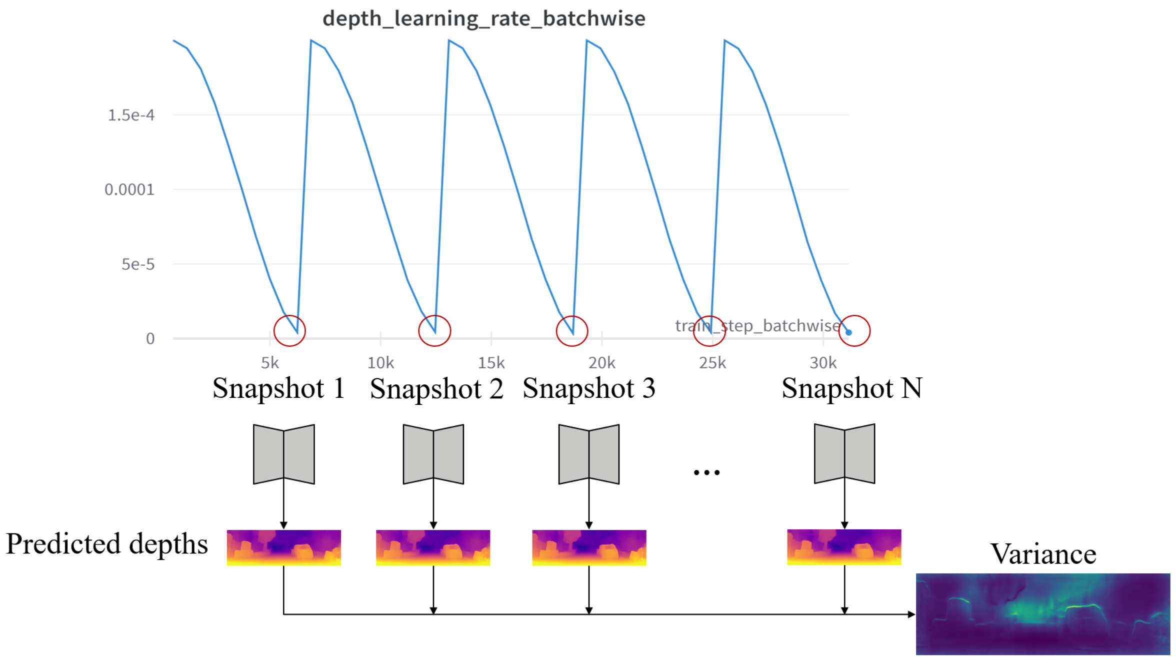

Figure 2.6: **Snapshot ensembles.** Using variance across snapshot ensembles as uncertainty. Snapshots of the model are taken at low points of a cyclic learning rate throughout a single training session.

Chapter 3

Predicting Uncertainty with Depth Scales

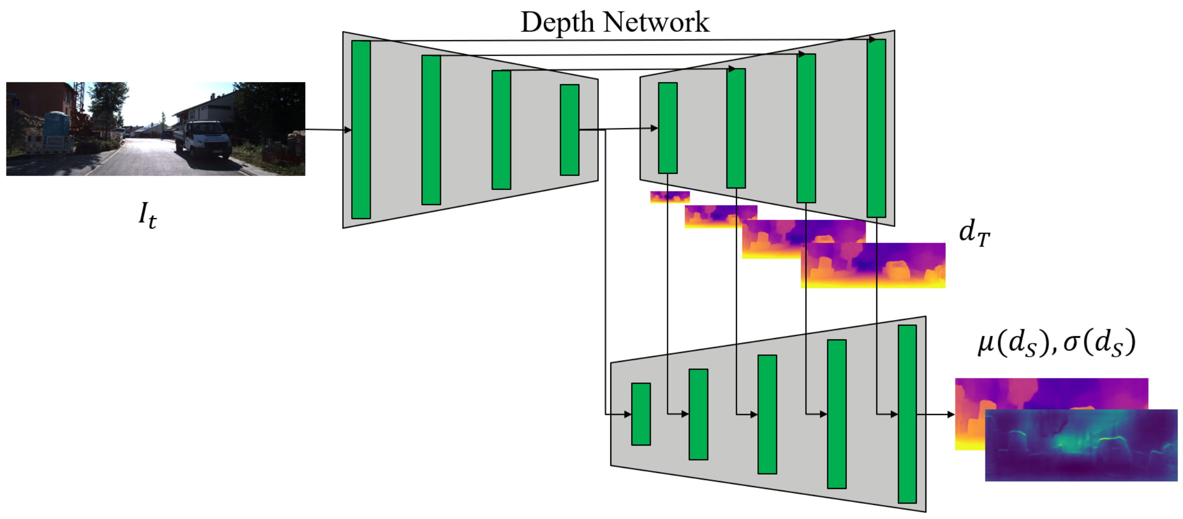

Figure 3.1: **Basic ScaleDecoder architecture.** The ScaleDecoder (bottom) takes the four depth scales of the depth decoder (top) and the U-Net bottlenet to predict parameters $\mu(d_S), \sigma(d_S)$ for d_T's underlying distribution for the given input camera image I_t.

3.1 Empirical Uncertainty from depth scales

Baseline depth network uses modified Monodepth2 U-Net architecture with ResNet18 backbone

Since decoder outputs at multiple scales, treat each scale as independent prediction. Uncertainty := variance

3.2 Predictive Uncertainty from depth scales (ScaleDecoder)

Intuition: using scales seems to be right idea, but has high-signal noise/artifacts. Need to process scales to get more useable uncertainty. Key idea: Feed depth scales to another decoder learn uncertainty from depth scales concatenated with previous layer Use ResNet18 bottleneck

ScaleDecoder Hypothesis: Depth decoder scales + bottleneck can sufficiently learn depth distribution (uncertainty)

Experiment: Do learned uncertainties encode error?

Uncertainty Depth Scale decoder (ScaleDecoder) Use uncertainty network to predict mean and standard deviation at multiple scales Training: single training cycle Evaluation: single forward pass Like self teaching, but uncertainty network has access to bottleneck + depth scales Access to bottleneck = access to global latent code for image

3.3 ScaleDecoder variants

Training scheme Teach – frozen pretrained depth net, only uncertainty net learns Detach – depth+uncertainty nets learn together, uncertainty gradients do not backprop to depth net Joint - depth+uncertainty nets learn together, all gradients backprop

Uncertainty network final nonlinearity NOTE: predicted $\log \sigma(d)$ instead of $\sigma(d)$ numerical stability, always positive $\sigma(d)$ Default: $\log \sigma(d) = 6 \tanh(\cdots) - 3$.

Adaptive: $\log \sigma(d) = a \tanh(\cdots) - b$, where a, b are network parameters.

Number of uncertainty output scales Default: decoder outputs one scale $\sigma(d), \mu(d)$. Multi-scale: decoder outputs four scales $\sigma_1(d), \mu_1(d), \sigma_2(d), \mu_2(d), \cdots, \sigma_4(d), \mu_4(d)$.

Chapter 4

Experimental Results

4.1 Metrics

4.1.1 Motivation

Uncertainty is supposed to encode the errors: pixels ordered by uncertainty and error should be roughly the same.

No "ground-truth uncertainty" for evaluation. If uncertainty encodes the errors (requires ground-truth depth) via sparsification-based metrics (AUSE, AURG).

What are we evaluating when we look at Depth/Uncertainty metrics? What are we looking for?

4.1.2 Depth Metrics

For evaluation, LIDAR ground truth depthmap d^* is compared against the predicted depthmap d_t only on valid ground-truth LIDAR pixels P. Table 4.1.2 shows the definitions for the metrics AbsRel (Absolute Relative), SqRel (Squared Relative), RMSE (Root Mean Squared Error), and $RMSE_{log}$ (Root Mean Squared of Log Error). During evaluation, these instance metrics are then averaged over the entire KITTI eigen test split [3].

So far, we have only looked at metrics that are scale-variant, in that the metrics depend on the scale learned by a monodepth network. We can partially address this problem by median-scaling the depth predictions by ground truth (predictions are multiplied by $\mathrm{median}(d^*)/\mathrm{median}(d_t)$), so

AbsRel↓	SqRel↓	RMSE↓	$\mathrm{RMSE}_{log}\downarrow$
$\frac{1}{\lvert P\rvert}\sum_P \frac{\lvert d_t-d^*\rvert}{d^*}$	$\frac{1}{\lvert P\rvert}\sum_P \frac{\lvert\lvert d_t-d^*\rvert\rvert^2}{d^*}$	$\sqrt{\frac{1}{\lvert P\rvert}\sum_P \lvert\lvert d_t-d^*\rvert\rvert^2}$	$\sqrt{\frac{1}{\lvert P\rvert}\sum_P \lvert\lvert\log(d_t)-\log(d^*)\rvert\rvert^2}$

Table 4.1: **Scale-variant depth metrics.**

$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
$\max(\frac{d_t}{d^*}, \frac{d^*}{d_t}) < 1.25$	$\max(\frac{d_t}{d^*}, \frac{d^*}{d_t}) < 1.25^2$	$\max(\frac{d_t}{d^*}, \frac{d^*}{d_t}) < 1.25^3$

that the new predicted median depth is the same as the ground-truth median depth[1]. Considering a maximum scale factor between prediction d^* and ground truth d_t as $\delta = \max(d_t/d^*, d^*/d_t)$, we can evaluate a depthmap for scale accuracy as well. The notation $\delta < c$ represents the percentage of pixels p that have predicted depth $d_t(p)$ within a maximum scale factor of c with respect to ground truth $d^*(p)$. In self-supervised depth estimation literature, c is chosen as $1.25, 1.25^2, \text{ and } 1.25^3$ for three accuracy metrics.

4.1.3 Uncertainty Metrics

Uncertainty: (predictive) predicted depthmap standard deviation $\sigma(d)$. (empirical) variance across many depthmaps

Given: depth error $d_t - d^*$.

Want to see how well uncertainty $\sigma(d)$ encodes depth error $d_t - d^*$. If pixels ordered by $\sigma(d)$ and $d_t - d^*$ are "roughly same."

How to quantify "roughly same" order? (Sparsification)

Model sparsification: Given error metric ϵ (e.g. RMSE) on a set of points.

Choose pixel order from uncertainty model (e.g. by decreasing $\sigma(d)$)

Remove a constant fraction of highest ordered pixels (e.g. 2%) Plot error ϵ averaged on remaining points. Repeat until no points left

Oracle sparsification: pixels with highest ϵ error removed first; perfect error-encoding

Random Sparsification: pixels sorted in random order (random variance)

[TODO show example sparsification plots and errors]

Example plot ϵ = RMSE Blue is method sparsification Green is oracle sparsification Red is random sparsification

Best method sparsification is: Closest to oracle (green) Farthest from random (red)

Equivalently… Model sparsification error: Model sparsification minus oracle sparsification

AUSE == area under model sparsification error

AURG == area between model sparsification error and random sparsification error

Key takeaway: AUSE$\uparrow$, AURG$\downarrow$ tell us how well uncertainty encodes errors.

4.2 Data

Show frame results from video

[1]As an implementation detail, we need to multiply the depths, not the inverse depth output, as the network provides.

Camera image I_t.
3D reconstruction Predicted depth d_t
Uncertainty $\sigma(d), \sigma^2(d)$
Ground truth depth d^* Filtered 3D reconstruction Error $(d_t - d^*) * *2$
uncertainty-filtered depth

4.2.1 Qualitative data

4.2.2 Quantitiative data

	Model	AbsRel↓	SqRel↓	RMSE↓	$\mathrm{RMSE}_{log} \downarrow$	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
depth	boot	0.973	14.849	18.939	3.633	–	–	–
	scale	0.964	14.652	18.825	3.350	–	–	–
	snap	0.974	14.963	19.063	3.688	–	–	–
	SD teach*	0.963	14.567	18.757	3.348	–	–	–
	MS SD detach	0.957	14.376	18.627	3.189	–	–	–
	MS SD detach ad.	0.956	14.366	18.638	3.163	–	–	–
	MS SD joint**	$\underline{\mathbf{0.922}}$	$\underline{\mathbf{13.428}}$	$\underline{\mathbf{18.034}}$	$\underline{\mathbf{2.620}}$	–	–	–
depth-pp	boot	0.973	14.849	18.939	3.633	–	–	–
	scale	0.964	14.656	18.828	3.351	–	–	–
	snap	0.974	14.962	19.063	3.688	–	–	–
	SD teach*	0.964	14.571	18.760	3.348	–	–	–
	SD detach MS	0.957	14.388	18.639	3.192	–	–	–
	MS SD detach ad.	0.956	14.374	18.649	3.162	–	–	–
	SD joint MS**	$\underline{\mathbf{0.923}}$	$\underline{\mathbf{13.455}}$	$\underline{\mathbf{18.064}}$	$\underline{\mathbf{2.620}}$	–	–	–
depth-gt	boot	0.126	**0.849**	**4.716**	**0.200**	0.855	**0.955**	**0.981**
	scale	**0.123**	1.048	5.087	0.204	**0.866**	**0.955**	0.978
	snap	0.189	1.549	7.681	0.287	0.691	0.888	0.954
	SD teach*	$\underline{0.133}$	1.166	$\underline{5.226}$	$\underline{0.216}$	$\underline{0.852}$	$\underline{0.949}$	$\underline{0.975}$
	SD detach MS	0.145	1.243	5.444	0.223	0.828	0.943	$\underline{0.975}$
	MS SD detach ad.	0.144	$\underline{1.162}$	5.287	0.224	0.826	0.942	0.974
	SD joint MS**	0.168	1.304	5.577	0.244	0.773	0.929	0.971
depth-pp-gt	boot	0.126	0.844	**4.704**	0.200	0.855	0.955	**0.981**
	scale	**0.119**	**0.939**	4.883	**0.199**	**0.870**	**0.957**	0.979
	snap	0.188	1.544	7.680	0.287	0.692	0.889	0.954
	SD teach*	$\underline{0.129}$	1.052	$\underline{5.035}$	$\underline{0.211}$	$\underline{0.855}$	$\underline{0.952}$	$\underline{0.977}$
	SD detach MS	0.144	1.126	5.283	0.220	0.825	0.947	$\underline{0.977}$
	MS SD detach ad.	0.141	$\underline{1.026}$	5.166	0.219	0.826	0.946	$\underline{0.977}$
	SD joint MS**	0.159	1.146	5.348	0.235	0.784	0.939	0.975

Table 4.2: **Metrics on uncertainty-based depth predictions.** The sections from top to bottom are raw depth, post-processed depth, ground-truth median-scaled depth, and ground-truth median-scaled depth, post processed depth. Best metrics are bolded and best metrics among the novel ScaleDecoder (SD) family are underlined. *All teaching variants have the same depth metrics. **The joint strategy reported is tested with a smaller posenet encoder.

	Model	AUSE↓	AURG↑
depth	boot	14.856	-4.971
	scale	2.851	6.923
	snap	**0.759**	**9.257**
	SD teach	7.312	2.483
	SD teach adaptive	6.923	2.873
	SD teach adaptive MS	6.157	3.640
	SD detach MS	7.896	1.828
	SD detach adaptive MS	4.322	5.425
	SD joint*	<u>3.239</u>	<u>6.211</u>
depth-pp	boot	15.155	-5.262
	scale	2.339	7.430
	snap	**0.739**	**9.280**
	SD teach	7.119	2.676
	SD teach adaptive	6.735	3.060
	SD teach adaptive MS	5.960	3.837
	SD detach MS	7.529	2.204
	SD detach adaptive MS	4.108	5.648
	SD joint*	<u>3.233</u>	<u>6.243</u>
depth-gt	boot	4.379	-0.462
	scale	1.981	2.322
	snap	**1.290**	**5.035**
	SD teach	2.452	1.947
	SD teach adaptive	2.300	2.099
	SD teach adaptive MS	1.972	2.422
	SD detach MS	2.550	1.984
	SD detach adaptive MS	1.522	2.860
	SD joint*	<u>1.470</u>	<u>3.028</u>
depth-pp-gt	boot	4.365	-0.466
	scale	1.736	2.377
	snap	**1.265**	**5.052**
	SD teach	2.299	1.926
	SD teach adaptive	2.151	2.073
	SD teach adaptive MS	1.828	2.390
	SD detach MS	2.360	1.991
	SD detach adaptive MS	<u>1.416</u>	2.835
	SD joint*	1.440	<u>2.867</u>

Table 4.3: **Metrics on uncertainty predictions.** The sections from top to bottom are raw depth, post-processed depth, ground-truth median-scaled depth, and ground-truth median-scaled post-processed depth. Best metrics are bolded. Best metric for novel ScaleDecoder (SD) among variants are underlined. *The joint strategy reported is tested with a smaller posenet encoder.

Chapter 5

Discussion

DEPTH Depth metrics are evaluated with uncertainty depth estimate ScaleDecoder: $\mu(d)$ Empirical methods: statistical mean of depth estimates ScaleDecoder depths as good as other empirical methods, with fewer resources Scale depth is from depth net Teaching under single-scale loss is best among ScaleDecoder variants

Depth Metrics Joint (worst): uncertainty loss pollutes depth network Changes d_t's underlying distribution! Detach: d_t's underlying distribution not yet converged Teach (best): pretrained depth network frozen

UNCERTAINTY Empirical snap is best

No clear winner among ScaleDecoder groups

Curve with least area wins (AUSE).

Dashed lines – ScaleDecoders Solid lines – empirical

No clear winner among ScaleDecoder groups

Uncertainty Metrics ScaleDecoder as good as empirical methods Expected self-teaching strategies to out-perform empirical ones [12]. However, ScaleDecoder is small, single forward pass, possible substitute for empirical methods.

Vast amount of predicted uncertain points not shared with valid LIDAR ground truth points By design, uncertainty cannot be evaluated on invalid LIDAR pixels "Sky" regions (infinite depth) non-Lambertian surfaces (glass, mirrors, etc.)

What if ScaleDecoder is incorrect with high certainty? More likely with predictive uncertainty Keep and promote erroneous points -¿ road disaster

What if depth estimates predicted as certain but not accurate? $\rightarrow$ most dangerous kind of estimate (false certainty).

Chapter 6

Conclusion

ScaleDecoder Hypothesis: Depth decoder scales + bottleneck can sufficiently learn depth distribution (uncertainty)

ScaleDecoder performs as well as empirical methods, with fewer resources. Only one decoder is needed vs. entire U-Net in self-teaching (Poggi et. al.) [12]

Uncertainty works best if learned from distribution of trained, frozen depth net Cannot train depth+uncertainty together (joint, detach)

Hypothesis: If uncertainty sufficiently encodes errors, we can improve depth accuracy by filtering highest-uncertainty points

Qualitative: flying points removed around objects

What makes architecture of depth decoder better/worse than baseline methods.

How to explain performance of SD using the ablation and difference between other methods.

TODO: [1, 2, 4, 5, 5, 6, 7, 8, 9, 10, 12, 14, 16, 17]

Chapter 7

Future Works

Major What if ScaleDecoder takes all intermediate features Take features of depth encoder as well

How to evaluate uncertainty on pixels without LIDAR?

Ablation for all combinations of (teach — joint — detach) x (adaptive — fixed) x (single-scale — multi-scale).

Minor Proper hyperparameter search for uncertainty loss weights (loss + regularization). Understand effect of gt-scaling for empirical uncertainty models before/after statistical mean/variance. Compare image flipping and dropout uncertainty predictions. AUSE + AURG use the $\epsilon = RMSE$. We can use $\epsilon = AbsRel, SqRel$,etc.

Bibliography

[1] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072, 2019. 2.1, 2.1.1, 2.1.2, 6

[2] Jaehoon Choi, Dongki Jung, Donghwan Lee, and Changick Kim. Safenet: Self-supervised monocular depth estimation with semantic-aware feature extraction. *arXiv preprint arXiv:2010.02893*, 2020. 6

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014. 4.1.2

[4] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer, 2016. 2.1, 6

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1.1, 6

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 2.1, 2.1.2, 6

[7] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 2.1, 2.1.2, 6

[8] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 2.1, 2.1.2, 6

[9] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020. 1.1, 2.1, 6

[10] Vitor Guizilini, Igor Vasiljevic, Rares Ambrus, Greg Shakhnarovich, and Adrien Gaidon. Full surround monodepth from multiple cameras. *IEEE Robotics and Automation Letters*,

7(2):5397–5404, 2022. 1.2.2, 2.1, 2.1.1, 6

[11] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018. 2.2.2

[12] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020. 2.2.2, 2.3, 2.3.2, 5, 6

[13] Chao Qu, Wenxin Liu, and Camillo J Taylor. Bayesian deep basis fitting for depth completion with uncertainty. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16147–16157, 2021. 1.2

[14] Igor Vasiljevic, Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Wolfram Burgard, Greg Shakhnarovich, and Adrien Gaidon. Neural ray surfaces for self-supervised learning of depth and ego-motion. In *2020 International Conference on 3D Vision (3DV)*, pages 1–11. IEEE, 2020. 2.1, 2.1.1, 2.1.1, 2.1.2, 6

[15] Lijun Wang, Yifan Wang, Linzhao Wang, Yunlong Zhan, Ying Wang, and Huchuan Lu. Can scale-consistent monocular depth be learned in a self-supervised scale-invariant manner? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12727–12736, 2021. 1.2.2, 2.1.2

[16] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 340–349, 2018. 2.1, 2.1.2, 6

[17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017. 1.1, 2.1, 2.1.2, 6