DEPTH ESTIMATION: FROM MONOCULAR TO MULTIPLE VIEWS

by

ZHUOFEI HUANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Since pixel-wise ground-truth depth data is difficult to obtain in real world, it is significant for researchers to develop self-supervised depth estimation frameworks. In recent years, self-supervised monocular depth estimation has shown impressive results where networks are trained to predict depth map for a single image frame by using adjacent frames as supervision signal during training period. Meanwhile, in many applications, information of video sequences are also available at test time. Many researchers found that multi-view stereo (MVS) depth estimation based on cost volume usually works better than monocular schemes except for moving objects and low-textured surfaces. Based on these facts, we hope to combine advantages of monocular and multi-view schemes and design a new integrated depth estimation framework with better performance.

In this paper, we first introduce several representative self-supervised depth estimation frameworks in recent years, including monocular and multi-view cases. Besides, to reduce the influence of observation noises (*e.g.*, occlusion and moving objects), we introduce the concept of Bayesian uncertainty and explain how to improve the depth accuracy with uncertainty estimation. Then we

will propose a multi-frame depth estimation framework where monocular depth map can be refined continuously by multi-frame sequential constraints, leveraging a Bayesian fusion layer within several iterations. Both monocular and multi-view networks can be trained with no depth supervision. Our method also enhances the interpretability when combining monocular estimation with multi-view cost volume.

CHAPTER 1

INTRODUCTION

Predicting scene depth from imagery is of central importance for applications ranging from autonomous driving, robot navigation, to augmented reality. As acquiring depth ground truth for each pixel is challenging, self-supervised learning method with relative pose estimation as supervised signal is promising in monocular depth estimation. Meanwhile, with pose provided, multi-view cost volume aggregation is also a practical method to obtain accurate depth. A good depth estimation method should be capable of generating both accurate and complete depth.

However, monocular depth is not intepretable and hard to generalize. Due to the large domain gap between training and testing data, many pretrained networks in monocular depth estimation [26, 7] may easily collapse in new scenes. Besides, monocular methods predict inaccurate depths for regions of large image gradient as shown in Figure 1.1. Actually, in many practical scenarios such as camera on moving vehicle or handheld mobile phone, more than one frame is available at test time. Such multi-frame information is not exploited by recent monocular methods. To make use of additional frames at both training and testing time, MVSNet [23] takes multiple images as input, and builds a cost volume with known camera poses, inferring depth in an end-to-end fashion. However, cost volume based methods degrade in moving object and untextured surfaces, resulting in incomplete depth maps as shown in Figure 1.1. Intuitively, combining both monocular and multi-frame methods should generate both accurate and complete depth.

One recent approach Manydepth [21] shows an implicit combination of monocular framework and cost volume. Different from MVSNet-like frameworks, it builds a cost volume concatenated with monocular feature map to regress final depths. It compares the depth represented by the $argmin$ of the cost volume with monocular depth to identify reliable pixels. However, this assumption is questionable for cost volume. In fact, [22] shows explicit representation of depth output: expectation of probability volume. Besides, the distribution of probability volume reflects the confidence of multi-view stereo matching, which serves as an important criteria to find reliable

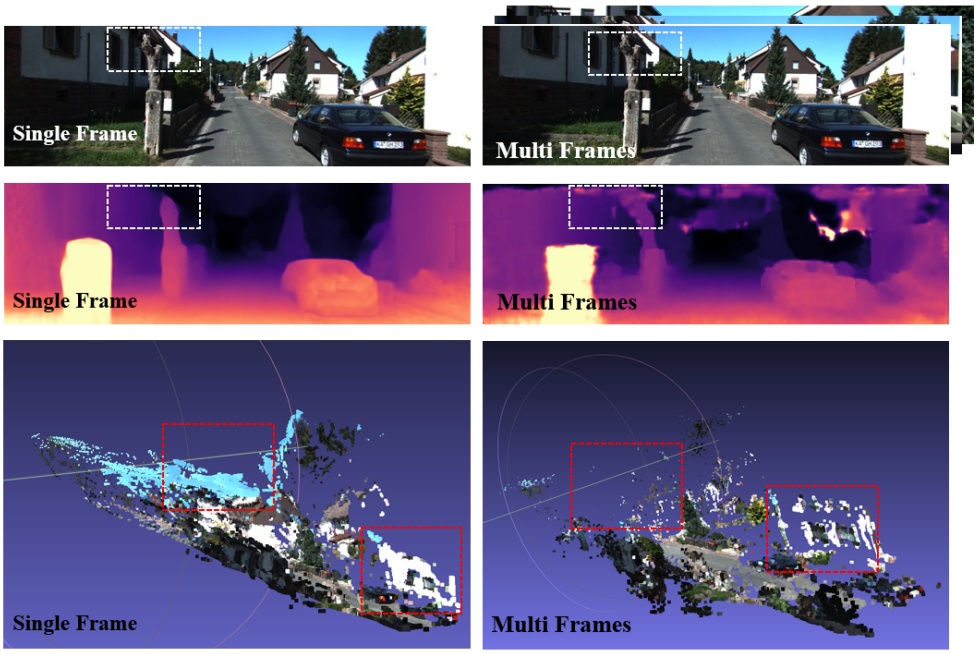


Figure 1.1: Depth predictions from single (left column) and multi frames (right column). Multi--view method can predict high quality depth in highly textured regions as in white dotted boxes, while failing in low textured cases as in red boxes. In the third line we generate 3D maps from monocular and multi-frame methods which run on the whole video sequence. We find that less depth can pass depth consistency check for multi-frame methods in those untextured regions.

pixels. Therefore, we design a system decoupling cost volume from monocular network, and show how to exploit more information from cost volume and provide a reliable sparse depth map for following fusion step, with monocular depth.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

In this chapter, we first give a brief introduction on the unsupervised depth estimation tasks, and then review the related work on monocular cases as well as on multi-frame cases.

2.1 Monocular Depth Estimation

The main purpose of monocular depth estimation is to predict the depth of each pixel in a single input image. Supervised framework exploit dense supervision from depth sensors during training, *e.g.* [5, 16]. Since supervised methods perform poor generalization and have difficulty of accessing ground truth depth value, researchers tried self-supervised monocular depth estimation in recent years.

Self-supervised methods usually train with image-reconstruction losses. [26] firstly proposes to adopt PoseCNN to estimate relative pose between each adjacent two frames. This work jointly updates depth and pose by minimizing image reconstruction loss. [1] presents scale-consistent depth estimation with geometry consistency loss and a self-discovered mask for handling dynamic scenes. [7] introduces auto-masking and min re-projection loss to solve the problems of moving objects and occlusion. By training with stereo pairs, [6] introduces a left-right depth consistency loss. [14] incorporates Mirrored Exponential Disparity probability volumes to regress the expectation of depths and Mirror Occlusion Module to solve occlusion cases. Besides, for robustness to illumination change and occlusions, [12, 22, 15] predict extra uncertainty map in self-supervised manner. Our monocular architecture is based on [7, 15], and provides initial guess of depth and uncertainty for all pixels. In contrast to previous monocular methods that directly regress final depth, monocular depth will be continuously refined by online learning during our training process.

2.2 Multi-frame Depth Estimation

In recent years, end-to-end frameworks that leverage temporal information from multiple views at test time grow fast to help improve the quality of predicted depths. One general version of multi-view methods is to adopt RNN or LSTM-based models. [14] uses the architecture of the ConvLSTM for real-time self-supervised monocular depth estimation and completion. [20] utilizes multi-view image reprojection and forward-backward flow consistency losses to train RNN model. Another version widely used as baseline is cost volume in multi-view stereo matching with known camera poses. MVSNet [23] starts to present an end-to-end deep learning architecture for depth map inference from multi-view images. [8] proposes a 3-stage 3D cost volume to generate multi-scale depth maps in a coarse-to-fine manner, where the range of depth interval reduces remarkably stage by stage. [25] regresses uncertainty map from probability volume to solve erroneous cost aggregation from occluded pixels. [9] trains the MVSNet in an unsupervised manner by combining pixel-wise and feature-wise loss function, which helps decrease mismatch errors in some untextured and texture repeat areas. These works utilize sequence information and show better performance of depth estimation for most static regions with textured surfaces than monocular frameworks.

In our designed system, MVSNet serves as real depth sensor with observation noises. Assuming that ground truth of camera poses are known, a pretrained MVSNet provides estimation of depth values with distribution around the ground truth of depths. For the same referenced view, when we select different source views as input for MVSNet, we will generate distinct observations for depth value of each pixel in referenced view.

2.3 Multi-frame Monocular Depth Fusion

Combining both privileges of monocular and multi-frame depth estimation is an attractive technique. In fact it is necessary to generate both accurate and complete depth in most 3D reconstruction tasks, where monocular brings complete and multi-frame brings accuracy. [21] predicts superior depths from a single image, or from multiple images when they are available, by concatenating monocular feature with cost volume. The network will select inliers by finding the $argmin$

of the cost volume, and encourage prediction of outliers to be similar to monocular estimation. [2, 24, 13] introduces self-supervised framework for jointly learning depth and optical flow with online refinement strategies.

CHAPTER 3

MONOCULAR DEPTH ESTIMATION

In this chapter, we present a general depth prediction network that takes a single color input and produces a depth map. Before that we will first introduce a typical unsupervised depth learning framework. In addition, geometry constraint is proposed to solve global scale problem. Finally we introduce the evaluation metrics of depth estimation and show results of partial representative methods in recent years.

3.1 Overview of Monodepth2

Given one target image $I \in \mathcal{R}^{3\times H\times W}$, a monocular depth network Φ as shown in Figure 3.1 infers depth values for each pixel, *i.e.*, $D = \Phi(I)$. Φ is designed as a skip-connection encoder-decoder architecture similar to Monodepth2 [7]. Due to the lack of ground-truth depth values, we consider the target frame I_t together with two adjacent source views $I_s \in \{I_{t-1}, I_{t+1}\}$ to optimize Φ by warping pixels from source to target view and minimizing the photometric re-projection error:

$$r(I_t, I_{t'}) = \frac{\alpha}{2}(1 - SSIM(I_t, I_{t'})) + (1-\alpha)\|I_t - I_{t'}\| \tag{3.1}$$

where I'_t is the warped image from source view I_s to target I_t, and $SSIM$ calculates structural similarity of patches. Similar to [7] we follow the form of per-pixel minimum loss to tackle potential occlusion issues. The photometric loss is denoted as:

$$\mathcal{L}_{pho} = \min_t r(I_t, I_{t'}) \tag{3.2}$$

To remove discontinuity of the learned depth in low-texture regions and resolve the gradient-locality issue, we propose an edge-aware smoothness term as in [6]:

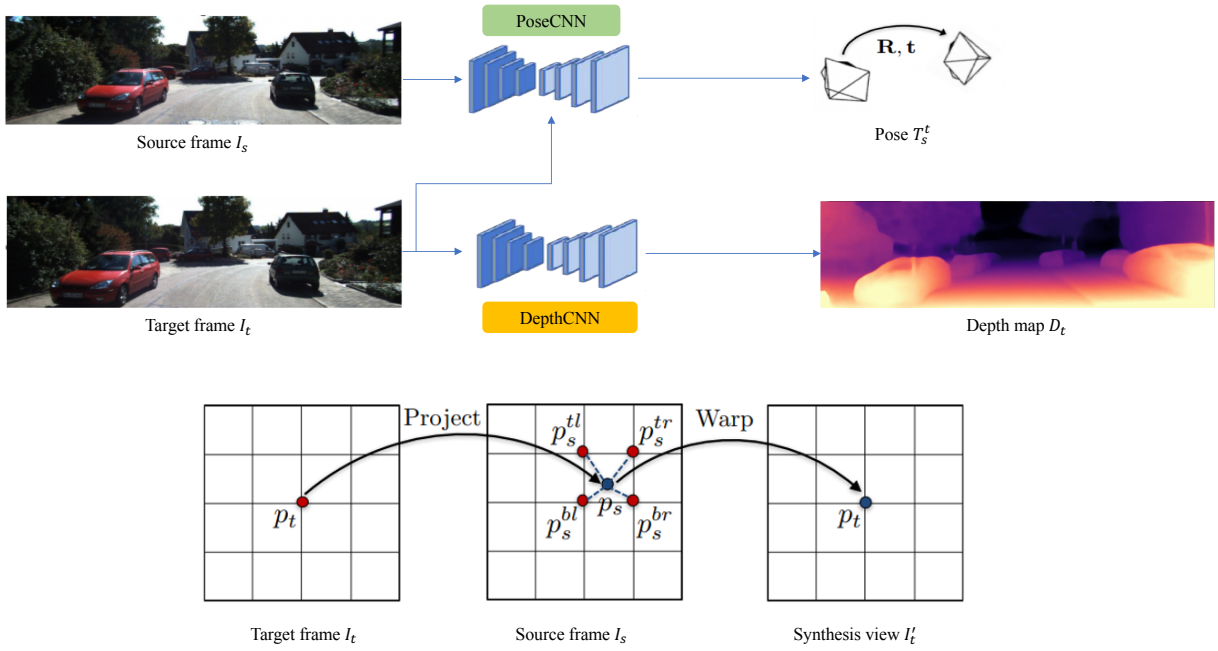


Figure 3.1: Universal framework of self-supervised depth estimator. It uses two standard fully convolutional U-Net structures to predict depth and relative pose T_s^t. The output result T_s^t from PoseCNN will be used to warp target view I_t to source view I_s on each pixel, and the synthesis view I_t' is used to calculate photometric loss.

$$\mathcal{L}_{smooth} = |\delta_x D_t| e^{-\delta_x I_t} + |\delta_y D_t| e^{-\delta_y I_t} \tag{3.3}$$

where δ_x and δ_y represents gradients of depth and RGB images. The final loss of unsupervised depth estimation is:

$$\mathcal{L}_{mono} = \frac{1}{s} \sum_{i}^{s} (\mathcal{L}_{pho} + \lambda \mathcal{L}_{smooth}) \tag{3.4}$$

where s is the number of scales and λ the weight of smooth term.

3.2 Geometry Consistency

Since Monodepth2 [7] only uses information of 3 adjacent views as supervision signals, we fail to keep global scale consistency for pose estimation network (PoseCNN) in long a video sequence. Since PoseCNN and DepthCNN share the same scale in every sliding window with 3 views, it also means that we will not ensure global scale consistency for depth estimation. To solve the global scale problem, [1] proposes a geometry consistency constraint to enforce the scale-consistency of depth and pose networks, leading to a globally scale-consistent depth estimator in a long video sequence. Given two adjacent frames I_a and I_b, we first estimate their depth maps D_a, D_b and relative pose P_{ab} using the monocular estimation network as in Figure 3.1, and then we apply the re-projection operation to get a warped depth map D_{ba} by converting D_a to 3D space and projecting to the image plane of I_b using their relative pose P_{ab}. Finally we use the inconsistency between D_{ba} and the D'_b interpolated from D_b as the geometric consistency loss to supervise the network training. With this constraint, we compute the depth inconsistency map D_{diff}. For each pixel $p \in V$, it is defined as:

$$D_{diff}(p) = \frac{|D_{ba}(p) - D'_b(p)|}{D_{ba}(p) + D'_b(p)} \tag{3.5}$$

where V represents all image pixels which are successfully projected from I_a to I_b using relative pose P_{ab}, and $|V|$ is the number of points in V. With the inconsistency map, the geometry consistency loss $\mathcal{L}_{geo}$ and final loss $\mathcal{L}_{gc}$ can be simply defined as:

$$\begin{aligned} \mathcal{L}_{geo} &= \frac{1}{|V|} \sum_{p \in V} D_{diff}(p) \\ \mathcal{L}_{gc} &= \alpha \mathcal{L}_{mono} + \beta \mathcal{L}_{geo} \end{aligned} \tag{3.6}$$

where α and β are dynamic weighted terms. For two training batches with data intersection, *e.g.*, (I_t, I_{t-1}, I_{t+1}) and (I_{t+1}, I_t, I_{t+2}), where the overlapped frame is I_{t+1}, with consideration of geometry consistency constraint, the overlapped frame I_{t+1} will keep the same scale in these two training batches. And so on, by minizing final loss $\mathcal{L}_{gc}$, the training steps can propagate to the

entire video sequence and the global scale-consistency will be successfully kept among the whole video sequence.

3.3 Evaluation Metrics

To evaluate the quality of our estimated depth maps, we use scale-invariant metrics in [3] as our depth error measurement. The metrics include Absolute Relative Difference (Abs Rel), Squared Relative Difference (Sq Rel), Root Mean Square Error (RMSE), Logarithm of Root Mean Square Error (RMSE Log), and threshold accuracy. The explanation of these metrics are shown in Equation 3.7:

$$
\begin{aligned}
&Abs\ Rel : \frac{1}{|\mathcal{I}|}\sum_{\mathcal{I}}\frac{|d_{ij}^{pred}-d_{ij}^{gt}|}{d_{ij}^{gt}}, Sq\ Rel : \frac{1}{|\mathcal{I}|}\sum_{\mathcal{I}}\frac{||d_{ij}^{pred}-d_{ij}^{gt}||^2}{d_{ij}^{gt}} \\
&RMSE : \frac{1}{|\mathcal{I}|}\sum_{\mathcal{I}}\sqrt{||d_{ij}^{pred}-d_{ij}^{gt}||^2}, RMSE\ Log : \frac{1}{|\mathcal{I}|}\sum_{\mathcal{I}}\sqrt{||\log d_{ij}^{pred}-\log d_{ij}^{gt}||^2} \\
&Accuracy : percent\ of\ d_{ij}^{pred}\ s.t.\ max(\frac{d_{ij}^{pred}}{d_{ij}^{gt}}, \frac{d_{ij}^{gt}}{d_{ij}^{pred}}) = \delta < 1.25, 1.25^2, 1.25^3
\end{aligned}
\tag{3.7}
$$

where $|\mathcal{I}|$ is the total number of pixels in image $\mathcal{I}$. Obviously, we will find that for the first 4 metrics (Abs Rel, Sq Rel, RMSE, RMSE Log), their values follow "the lower the better", meanwhile values of threshold accuarcy follow "the higher the better". Using above metrics calculation methods, we list depth evaluation results of several self-supervised depth estimation variants in Table 3.1. All experiments are done at 640×192 resolution on KITTI Eigen split set [3], where we use 39810 monocular triplets (I_t, I_{t-1}, I_{t+1}) for training and 4424 for validation. Among four compared methods, Monodepth2 [7] shows the best performance under all 7 evaluation indexes.

It is worth noting that for all of above monocular frameworks, we will predict depth maps with no certain metric scale. So before we use Equation 3.7 to calculate evaluation metrics, we scale the estimated depth map by the ratio of the median of the ground-truth and estimated depth map as in Equation 3.8. After scaling, we replace d_{ij}^{pred} with $\hat{d_{ij}}^{pred}$ in Equation 3.7 for final

	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Sfm-learner [26]	0.183	1.595	6.709	0.270	0.734	0.902	0.959
DF-Net [27]	0.150	1.124	5.507	0.223	0.806	0.933	0.973
SC [1]	0.137	1.089	5.439	0.217	0.830	0.942	0.975
Monodepth2 [7]	**0.115**	**0.903**	**4.863**	**0.193**	**0.877**	**0.959**	**0.981**

Table 3.1: Quantitative results of depth estimation on KITTI Eigen split set [3] for distance up to 80m. For error evaluating indexes, Abs Rel, Sq Rel, RMSE and RMSE Log, lower is better, and for accuracy evaluating indexes, $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$, higher is better. Note that among these compared methods, SC and Monodepth2 are previously introduced in Section 3.2 and 3.1 respectively.

metrics calculation. Besides, such median scaling operation gives an explanation that even we add geometry consistency constraint in final loss, the evaluation metrics will not be better due to the uncertain metric scale calculation (as shown in the last two lines of Table 3.1, comparison of SC [1] and Monodepth2 [7]).

$$\hat{d_{ij}}^{pred} = d_{ij}^{pred} \cdot \frac{median(d^{gt})}{median(d^{pred})} \tag{3.8}$$

CHAPTER 4

BAYESIAN DEEP LEARNING FOR UNCERTAINTY

In this chapter, we introduce the basic theory of uncertainty in Bayesian deep learning, and then apply a universal Laplacian model in uncertainty estimation based on our previous depth estimation tasks. Finally, we show evaluation metrics for uncertainties.

4.1 Uncertainties Theory

In many computer vision tasks, in addition to predict a result, we also hope to know about the confidence level of our predictions as in Figure 4.1. In many cases, we hope that our deep model can autonomously estimate uncertainty maps. For wrong prediction results and out-of-distribution sample inputs beyond the expected range, we hope that the model can give a higher uncertainty. In [11], the author proposes two types of uncertainties: epistemic uncertainty and aleatoric uncertainty.

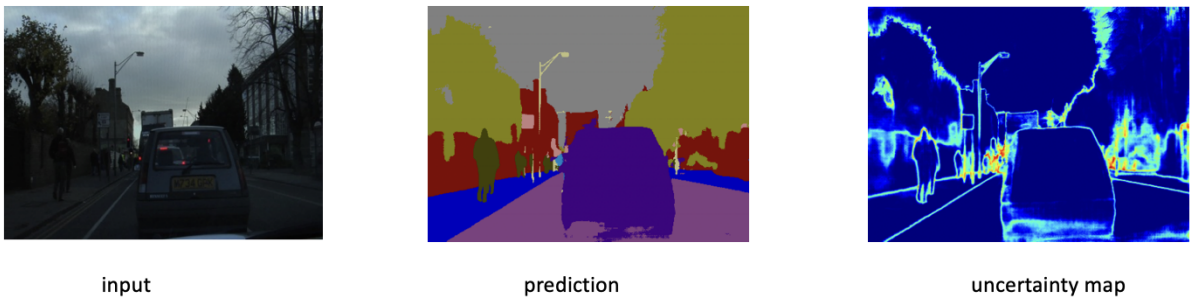


Figure 4.1: Example of uncertainty estimation in semantic segmentation. In the estimated uncertainty map, the bright areas means that our predictions are close to ground-truth, meanwhile the dark areas means that our predictions are far from ground-truth.

Epistemic uncertainty. It is the uncertainty of parameters in Bayesian theory, which is also known as model uncertainty. For the same data, we may train neural networks with different

weights, but all of them can solve the problem. Models can have different perceptions of data. Epistemic uncertainty is usually due to lack of knowledge, so that it needs more data helps, *i.e.*, data argumentation will help decrease epistemic uncertainty as shown in Figure 4.2.

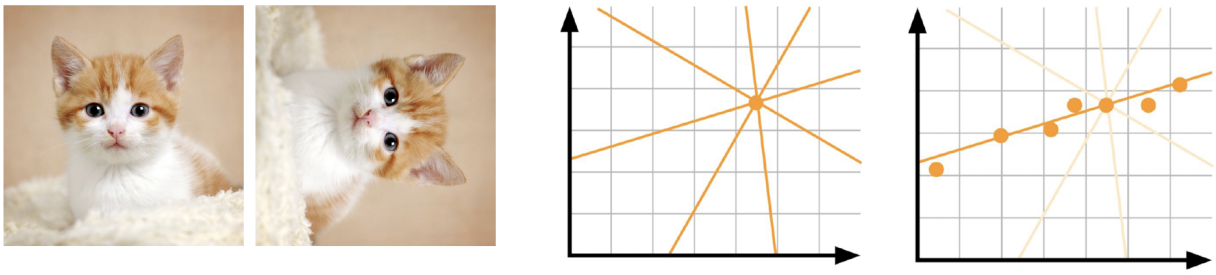


Figure 4.2: Explanation of decreasing epistemic uncertainty. On the left, we hope that our model can handle epistemic uncertainty, that is, when we input an image containing a cat, whatever its pose is, our model can distinguish that it is a cat. This is an example of using data augmentation to reduce epistemic uncertainty, *i.e.*, add the rotated image into the training data. On the right, it shows that when we enlarge our dataset, our model will trend to be unique with lower epistemic uncertainty.

Aleatoric uncertainty. It is the noise inherent in the oberservation. In other words, it captures the inherent uncertainty in the data, such as the inherent error of the depth sensor. Figure 4.3 explains a case of aleatoric uncertainty in a semantic segmentation task. Aleatoric uncertainty is stochastic and irreducible, and different from epistemic uncertainty, more data input will not help solve aleatoric uncertainty.

4.2 Uncertainty for Depth Estimation

As in [12], for a prediction task $\hat{y} = \Phi(x)$, where x is the input data and $\hat{y}$ is the predicted result, the parameters of estimation network Φ would be optimized to minimize the simple L_1 loss function $l = |\hat{y} - y|$, where y is the corresponding ground-truth label. However, if there exists non-negligible noises in ground truth data y (*e.g.*, the inaccurate annotations in Figure 4.3), we may down-weight our loss as l/σ with a proper coefficient σ in order to prevent our model from heavily influenced by such noises. The core idea in uncertainty estimation is to find out a suitable value of σ.

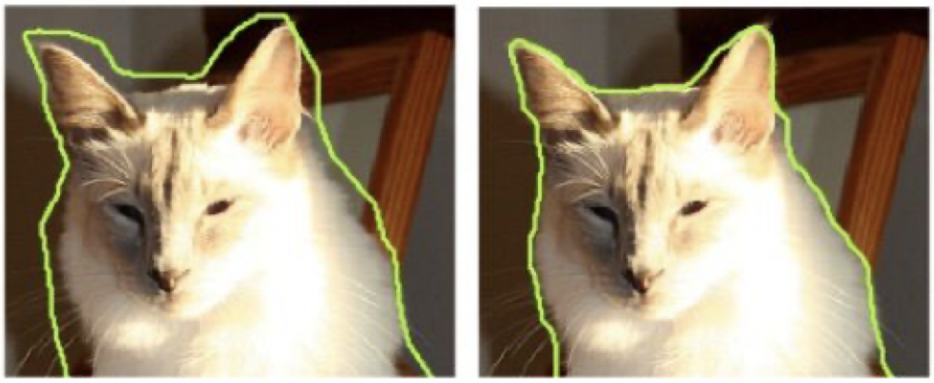


Figure 4.3: Example of aleatoric uncertainty in a semantic segmentation task. It shows that if data annotations are inaccurate, it may bring aleatoric uncertainty which will not be reduced by more data input. In this case, the wrong data annotations built by human is regarded as noise inherent.

Consider a simple supervised version of depth estimation task where ground-truth of depth maps are provided, for each single image frame I_t and model Φ, we have the prediction result $\hat{d} = \Phi(I_t)$. Simply we apply L_1 training loss $|\hat{d} - d|$ where d is the ground-truth depth label. Now our purpose is to model the confidence level for our prediction result $\hat{d}$, which stands for uncertainty of our model. In general cases, we will have assumptions that the distribution of prediction result $\hat{d}$ will stay around the ground-truth value d, and $\hat{d}$ will be far away from ground-truth d with low probability (refer to the distribution curves in Figure 4.4). Based on these assumptions, we can apply Laplace's distribution to measure the uncertainty. For each single image I, let the neural network output the parameters $(\hat{d}, \sigma) = \Phi(I)$ of a posterior probability distribution $p(d|\hat{d}, \sigma)$ over possible ground-truth labels d. For example, using Laplace's distribution:

$$p(d|\hat{d}, \sigma) = \frac{1}{2\sigma} exp(-\frac{|d - \hat{d}|}{\sigma}) \tag{4.1}$$

Usually we will consider the negative log-likelihood:

$$-\log p(d|\hat{d}, \sigma) = \frac{|d - \hat{d}|}{\sigma} + \log \sigma + const \tag{4.2}$$

The neural network that minimizes this quantity will try to guess $\hat{d}$ as close as possible to

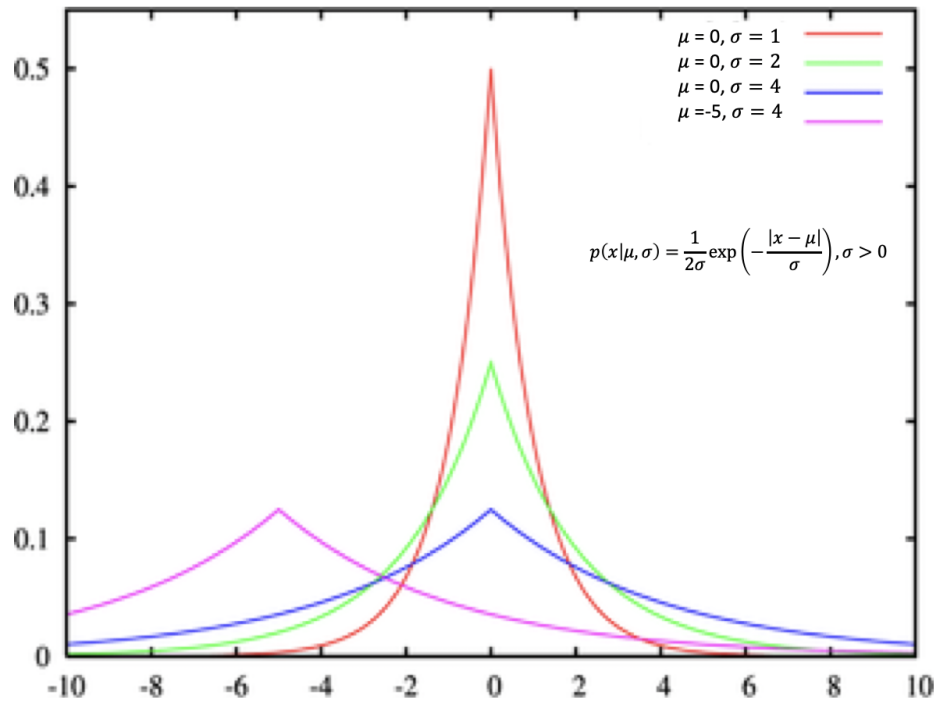


Figure 4.4: Plot for Equation (4.1), where ground-truth μ stands for the value on the x (horizontal) axis and the posterior probability $p(x|\mu, \sigma)$ stands for the value on the y (vertical) axis. We can easily derive that for a fixed μ, $p(x|\mu, \sigma)$ is maximized when $\sigma = |x - \mu|$.

ground-truth d. At the same time, it will try to set σ to the fitting error it expects. In fact, it is easy to see that, for a fixed $\hat{d}$, the loss is minimized when $\sigma = |d - \hat{d}|$.

Such method has been applied in [22] to estimate the photometric uncertainty in self-supervised depth estimation task. In this case the ground-truth d are unknown. Recall that in our previous unsupervised monocular scheme [7], we minimize the loss function as Equation (3.4) to estimate depth maps. However, Figure 4.5 shows that such loss function will not work fine when occlusion cases exist. Based on the brightness constancy assumption, the key idea is to replace the ground-truth d with the photometric re-projection error L_{pho}. The negative log-likelihood to be minimized is:

$$\mathcal{L}_{self} = \frac{1}{|V|} \sum_{p \in V} \frac{\mathcal{L}_{pho}}{\sigma_t} + \log \sigma_t \tag{4.3}$$

where σ_t is the uncertainty map of frame I_t and $\mathcal{L}_{pho}$ is calculated by Equation (4.3). The total loss function is the summation of negative log-likelihood losses and the smooth term from Equation (3.3) on multi-scale images:

$$\mathcal{L}_{total} = \frac{1}{s}\sum_{s}(\mathcal{L}^{s}_{self} + \lambda\mathcal{L}^{s}_{smooth}) \tag{4.4}$$

where $s = 4$ is the number of scales. To summarize, the proposed structure of uncertainty network is just slightly different from Figure 3.1. We add an extra channel on the last layer of DepthCNN, and this channel outputs the uncertainty map. By minimizing the loss in Equation (4.4), the network will finetune the parameters to fix the regions affected by noises, such as occlusion and moving objects.

4.3 Self-Teaching Scheme

In order to decouple depth and pose when modeling uncertainty, [15] propose to source a direct form of supervision from the learned model itself. By training a first network in a self-supervised manner, it builds a network instance *T* producing a noisy distribution d_T. Then, we train a second instance of the same model, namely *S*, to mimic the distribution sourced from *T*. Typically, teacher-student frameworks applied to monocular depth estimation deploy a complex architecture to supervise a more compact one. In contrast, in our approach the teacher T and the student S share the same architecture and for this reason we refer to it as Self-Teaching. By assuming an L_1 loss, we can model for instance negative log-likelihood minimization as:

$$L_{self} = \frac{|D^{T}_{t} - D^{S}_{t}|}{\sigma_t} + \log\sigma_t \tag{4.5}$$

With this strategy, we obtain a network S more accurate than T and in case of monocular supervision, we can decouple depth from pose and achieve a much more effective uncertainty estimation. In the next section we will introduce evaluation metrics for uncertainty estimation,

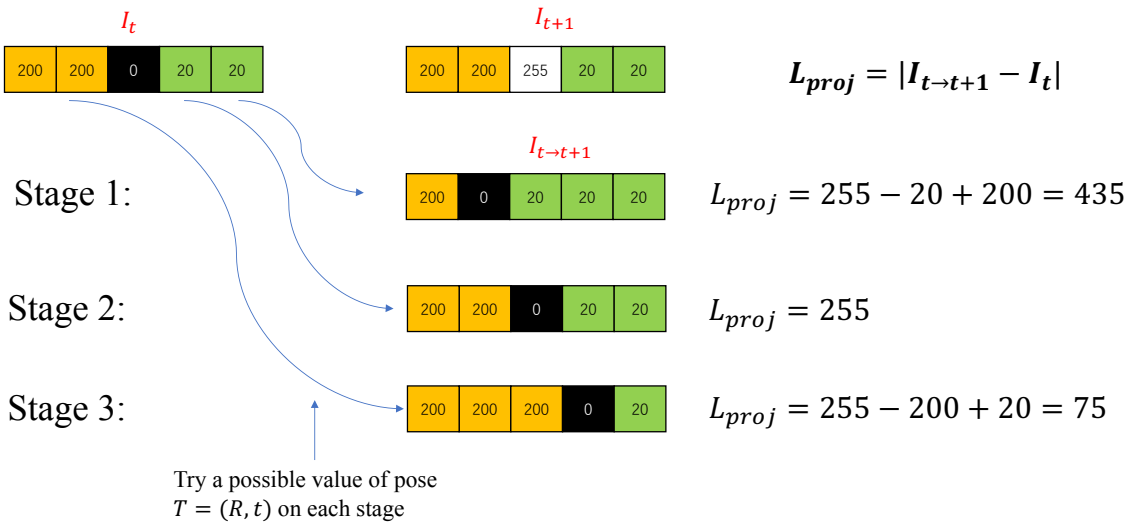


Figure 4.5: **Occlusion case without uncertainty estimation**. For adjacent views I_t and I_{t+1}, assuming that an occlusion occurs at the center of the image. In Monodepth2 scheme, we will try to obtain a proper value of relative pose between two frames from PoseCNN, in order to minimize the reprojection error L_{proj}. The figure shows that when we move I_t to left in 1 unit per stage, we will get different reprojection errors. Obviously at stage 2 we have tried the best value of pose which is equal to groundtruth, but we find the lowest reprojection error at stage 3. It reveals that the simple re-projection loss function cannot work fine in special cases.

and show that the self-teaching scheme performs better than other frameworks both on depth and uncertainty metrics.

4.4 Uncertainty metrics

To evaluate the significance of our estimated uncertainties, we use sparsification curves as in [10]. Recall the metrics in section 3.3, given an error metric ε, we sort all pixels in each depth map in order of descending uncertainty. Then, we iteratively extract a subset of pixels (usually 2%) and compute ε on the remaining to plot a curve (as shown in the red curve in the upper row of Figure 4.7), that is supposed to shrink if the uncertainty properly encodes the errors in the depth map.

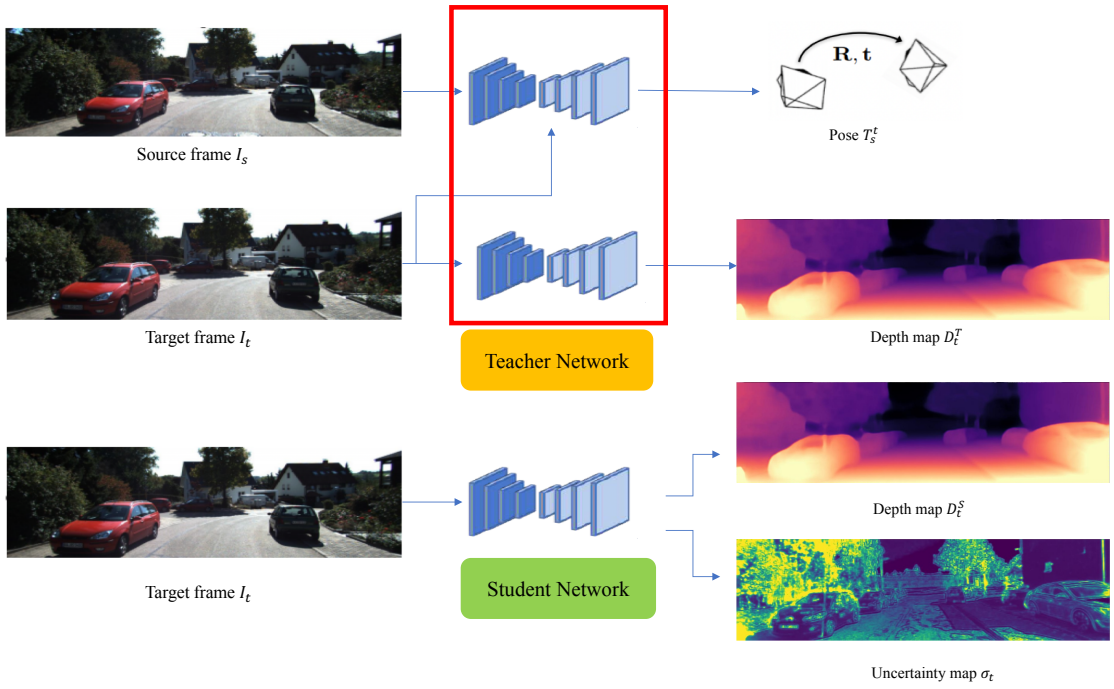


Figure 4.6: **Self-Teaching scheme**. A teacher network T is trained in unsupervised fashion, on monocular sequences $[t-1, t]$. A new instance of S the same is trained on D_t^T output of T. This scheme decouple depth uncertainty from pose network through setting a student network with only depth network.

There is a fact that if our estimated variance is a good representation of the model uncertainty, and the pixels with the highest variance are removed gradually, the error value ε should decrease monotonically. An ideal sparsification (oracle) is obtained by sorting pixels in descending order of the ε magnitude, as shown in the blue curve in the upper row of Figure 4.7. In contrast, a random uncertainty can be modelled as a constant, giving no information about how to remove erroneous measurements and, thus, a flat curve as shown in the green curve.

By plotting the difference between estimated and oracle sparsification, we can measure the Area Under the Sparsification Error (AUSE, the lower the better). Subtracting estimated sparsification from random one enables computing the Area Under the Random Gain (AURG, the higher the better). The former quantifies how close the estimate is to the oracle uncertainty, the latter how better (or worse, as we will see in some cases) it is compared to no modelling at all. We assume Abs Rel, RMSE or $\delta \geq 1.25$ (since $\delta < 1.25$ defines an accuracy score) as ε. The plot of AUSE

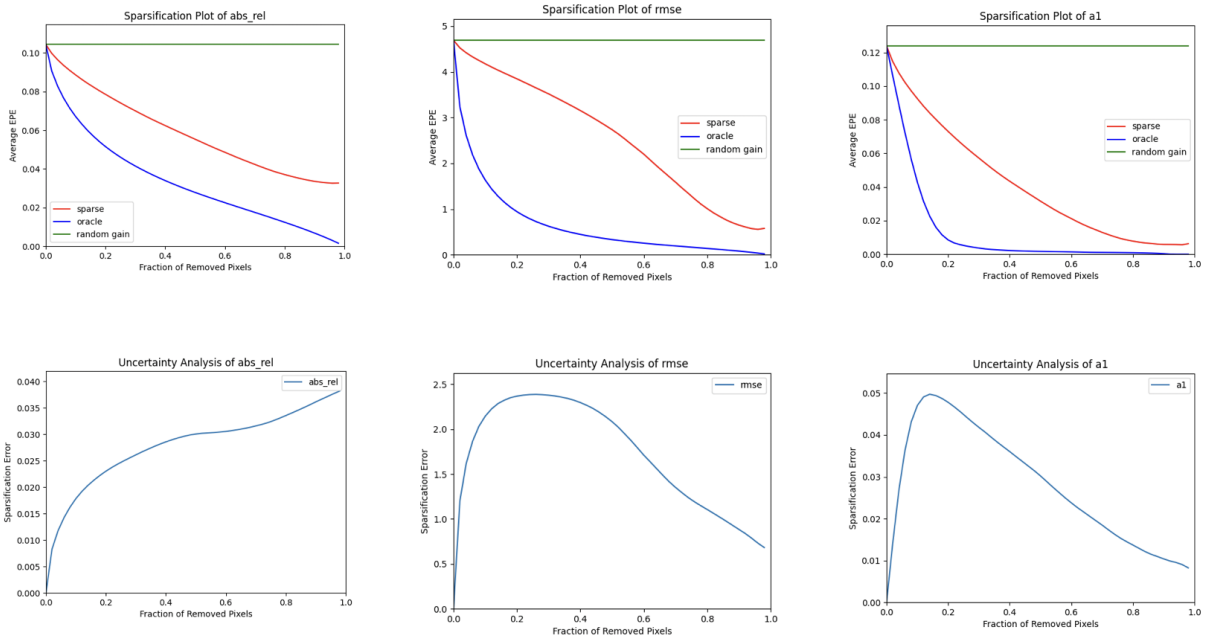


Figure 4.7: **Uncertainty metrics**. The error metrics includes absolute relative difference (Abs Rel), root mean square error (RMSE) and $\delta \geq 1.25$ (a^1). The first row shows sparsification, oracle and random curves. The second row shows the plots of AUSE, which means the difference between sparse and oracle curves.

are shown in the second row of Figure 4.7. The lower AUSE value reveals that our uncertainty estimation is closer to the oracle. We list the depth and uncertainty evaluation results in Table 4.1 and 4.2 respectively among three schemes: Monodepth2 [7], Monodepth2-Log (introduced in Section 4.2), and Monodepth2-Self (introduced in Section 4.3). All experiments are done at 640×192 resolution on KITTI Eigen split set [3], which is the same as in Section 3.3. Among three compared methods, self-teaching scheme shows the best performance both on depth and uncertainty estimation, under all evaluation metric indexes.

	Abs Rel ↓	RMSE ↓	$\delta < 1.25$ ↑
Monodepth2	0.090	3.942	0.914
Monodepth2-Log	0.091	4.052	0.910
Monodepth2-Self	**0.087**	**3.826**	**0.920**

Table 4.1: Quantitative results of depth estimation on KITTI Eigen split set [3] for distance up to 80m. Different from Section 3.3, here we use improve ground-truth data in [18]. For error evaluating indexes, Abs Rel, Sq Rel, RMSE and RMSE Log, lower is better, and for accuracy evaluating indexes, $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$, higher is better.

	Abs Rel		RMSE		$\delta \geq 1.25$	
	AUSE ↓	AURG ↑	AUSE ↓	AURG ↑	AUSE ↓	AURG ↑
Monodepth2	0.044	0.012	2.864	0.412	0.056	0.022
Monodepth2-Log	0.039	0.020	2.562	0.916	0.044	0.038
Monodepth2-Self	**0.030**	**0.026**	**2.009**	**1.266**	**0.030**	**0.045**

Table 4.2: Quantitative results of uncertainty estimation on KITTI Eigen split set [3] for distance up to 80m. For error evaluating indexes, AUSE, lower is better, and for AURG, higher is better.

CHAPTER 5

MULTI-VIEW DEPTH ESTIMATION

In this chapter, we extend our depth estimation task to multi-view cases, that is, to use a sequence of frames as input instead of single frame during inference time. We start with the methodology of stereo matching and multi-view stereo algorithms, and then introduce the architecture of Adaptive Cost Volume in an unsupervised multi-view learning scheme [21]. For generalization, we apply strategy of Dropout [17] to deal with special cases when the camera does not move or depth map of first frame is being estimated.

5.1 Problem Setup

In multi-view depth estimation problem, the aim is still to predict a depth map D_t for the input image I_t. Different from monocular cases in Chapter 3, during inference period, our neural model Φ accepts N previous frames from I_t as input: $D_t = \Phi(I_t, I_{t-1}, ..., I_{t-N})$. Specially, despite the model D_t accepts multiple frames as input, we still hope that it works when only single frame is available during test time, *i.e.*, it can be run at the single starting frame of the video sequence. Thus, the model Φ also accepts single frame as input: $D_t = \Phi(I_t)$.

5.2 Building Cost Volume

We first introduce a basic architecture of cost volume named MVSNet [23, 8] applied in depth estimation tasks. Although it is a supervised depth learning scheme with multiple views as input, we can easily extract its core components and apply them in unsupervised cases. Given multiple input frames I_t, I_{t-1}, I_{t+1}, the MVSNet extracts 2D features for each view. We then warp the feature maps from source (I_{t-1}, I_{t+1}) to target view (I_t) using each of determined discrete hypothesis depth planes $\{d_j\}_{j=1}^{N_d}$, with the known camera intrinsics and relative poses. All warped features

from source views are aggregated together into a single 3D probability volume via the *soft-argmax* operation. Finally we regress the depth map D from the probability volume $\{P_{ij}\}_{j=1}^{N_d}$ for each pixel i, which follows the below:

$$D_i = \sum_{j=1}^{N_d} d_j P_{ij} \tag{5.1}$$

where depth value D_i is computed as the expectation of distribution, and N_d is number of hypothesis depth planes. Notice that the probability distribution $\{P_{ij}\}_{j=1}^{N_d}$ reflects the confidence of depth value for each pixel, which helps measure the similarity between corresponding image patches and determine whether they are matched.

In conclusion, cost volume is used to measure the geometric compatibility between the pixel depth of the current frame and that of the adjacent frame. That is, when the depth of a certain pixel in an adjacent frame is a certain value d, what possibility is it to obtain an image like the current frame. However, in unsupervised depth estimation tasks, we cannot apply the architecture from MVSNet [8, 23] directly because relative poses between input frames are unknown. Such issue will cause that we are not able to warp the feature maps from source (I_{t-1}, I_{t+1}) to target view (I_t), so that we fail to build our 3D cost volume directly. Inspired from Monodepth2 [7], we can apply an extra pose estimation network, *i.e.*, PoseCNN, and then use the estimated relative pose from PoseCNN to build cost volume.

5.3 Adaptive Cost Volume

Cost volume can aggregate information from multiple views. However, without known depth range i.e. d_{min} and d_{max}, it will bring high difficulty in building a significant cost volume. To solve this problem, [21] proposed a novel architecture named Adaptive Cost Volume, which allows values of depth range to be learnt from data. Based on the final output depth map D_t in Figure 5.1, we compute the average min and max of each output D_t over a training batch, noted as D_t^{max} and D_t^{min}. We regard d_{min} and d_{max} as dynamic parameters needed to be finetuned, while processing each training batch, we iteratively update d_{min} and d_{max} as in Equation (5.2):

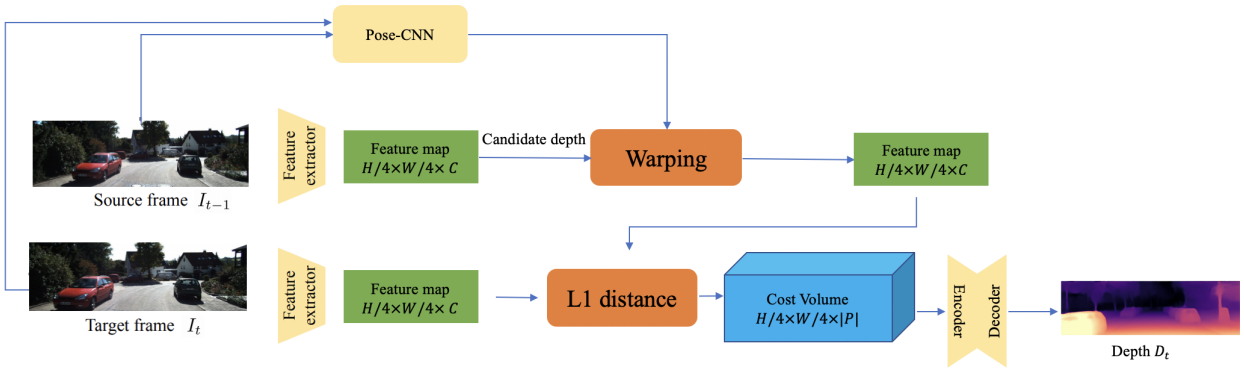


Figure 5.1: Network architecture of unsupervised depth estimation with cost volume. It contains three main components: a feature extractor, an pose network, and a depth encoder-decoder in U-Net structure. Our pose network estimates the relative pose between pairs of images, which is then used to build a cost volume in the reference frame of the target image I_t by warping features extracted from images at different time points. The encoder and depth decoder processes the cost volume to produce a depth image for I_t.

$$\begin{aligned} d_{max} &= 0.99 \cdot d_{max} + 0.01 \cdot D_t^{max} \\ d_{min} &= 0.99 \cdot d_{min} + 0.01 \cdot D_t^{min} \end{aligned} \tag{5.2}$$

After training, final values of d_{min} and d_{max} are saved along with the model weights, and then we will keep them fixed during inference period. Using this scheme, our model will continuously learn prior knowledge of depth range in the scene of our training data, which help us build a better structure of cost volume with more accurate range of hypothesis depth planes.

5.4 Cost Volume Overfitting

The authors of [23, 8] proposed that if the world is not static, that is, the scenery contains moving objects, the output depth map generated from cost volume will include large 'holes'. In these special cases, our network will be too dependent on information of cost volume. In a nutshell, the moving objects in consequent frames may bring a match in the cost volume at an incorrect depth plane and it will correspond to a low re-projection loss. [21] applied Knowledge Distillation to

solve above issues of overfitting. Similar to section 4.3, it proposed a teacher-student framework, where the teaching process is to use the output of the teacher network as the ground-truth of the student network. In this scheme, when there exists non-static scenes or moving objects, we hope that our network will discard the information from cost volume, and then simulate the learning process of monocular depth estimation as mentioned in section 3. To implement these, we use the monocular framework in [7] as teacher, and framework in figure 5.1 with cost volume as student. The teacher and student share the same PoseCNN to ensure scale-consistency. [21] propose an extra loss function $\mathcal{L}_{consistency}$:

$$\mathcal{L}_{consistency} = \sum M|D_t - \hat{D}_t| \tag{5.3}$$

where $\hat{D}_t$ and D_t are output depth maps from teacher and student network respectively, and M is a binary mask. The significance of mask M is to identify unreliable pixels of the target frame, where these pixels may cause that the cost volume fails to work. In masked regions we encourage the final predictions from student network to be similar to $\hat{D}_t$ (output of teacher). There is a valid assumption that for a region where the cost volume is reliable, the output depth map of teacher $\hat{D}_t$ will be similar to the depth map represented by the $argmin$ of the cost volume (we call D_{cv} below). In other words, only when we find that the output of monocular teacher network obviously differs from that from cost volume, we will discard the output result from student network and set M as 1. So that we can derive the equation of mask M:

$$M = max(\frac{D_{cv} - \hat{D}_t}{\hat{D}_t}, \frac{\hat{D}_t - D_{cv}}{D_{cv}}) > 1 \tag{5.4}$$

The loss function $\mathcal{L}_p$ of student network is similar to the Equation (3.4) in monocular cases. It is conceivable that when the cost volume is unreliable ($M = 1$), we should obviously decrease $\mathcal{L}_{consistency}$ while ignoring $\mathcal{L}_p$ (because $\mathcal{L}_{mono}$ is used to measure the performance of the student network, that is, the performance of cost volume). So the final loss is as follows, where $\mathcal{L}_{smooth}$ is the regular depth smoothing loss term in Equation (3.3):

$$\mathcal{L}_{mvs} = (1 - M)\mathcal{L}_p + \mathcal{L}_{consistency} + \mathcal{L}_{smooth} \tag{5.5}$$

5.5 Static cameras and first camera frame

There exists two major challenges when using multiple frames during test time in above method. The first issue is when the camera keeps static between frame I_{t-1} and I_t, the cost volume will also fail to work. The second issue arises when we estimate the first frame of a video sequences, I_{t-1} does not exist so that we cannot build cost volume. To address these two problems, we use the idea of Dropout [17] which means we replace part of our original modules with others with assigned probability. For the first case, in training process, we replace the frame I_{t-1} input to cost volume with a color-augmented version of I_t with probability p_1, but still calculate loss $\mathcal{L}_p$ with real I_{t-1}, I_{t+1}. This scheme will encourage network to predict significant depth maps when cost volume is built from two static frames with no camera baseline. For the second case, we can replace the cost volume with a tensor of zeros during training process with probability p_2. For these images, since they lack information of last frame, the scheme will encourage network to learn features directly from current frame I_t. When testing, we can simply replace the cost volume of I_{t-1} with zero tensors when dealing with the first frame of a sequence.

CHAPTER 6

DESIGN OF FUSIONDEPTH

In this chapter, we design a multi-frame monocular fusion system **FusionDepth**, which integrates some of components and ideas in previous chapters. The system overview is illustrated in Figure 6.1. Specifically, the goal of our system is to estimate a pixel-aligned depth map D_t with adjacent N frames from video sequence. First, we apply a single-image depth estimation as introduced in Monodepth2 [7], with an extra 2D-Convolution layer to predict depth uncertainty map σ_t. Then we use a representative MVSNet which contains 3-stage cascade cost volume to generate multi-view depth maps. Finally, combining uncertainty map of single frame with consistency checks of depths map from MVSNet, we adopt a Bayesian fusion module to refine depth values obtained from monocular depth maps.

6.1 Multi-Stage Cascade Cost Volume

To exploit multiple input frames in depth estimation, we build a 3-stage cascade cost volume [8] which generates depth map in a coarse-to-fine manner. Given multiple input frames I_t, I_{t-1}, I_{t+1}, the Cascade MVSNet extracts 2D features for each view. We then warp the feature maps from source to target view I_t using each of determined discrete hypothesis depth planes $\{d_j\}_{j=1}^{N_d}$, with the known camera intrinsics and relative poses. All warped features from source views are aggregated together into a single 3D probability volume via the *soft-argmax* operation. Finally we regress the depth map D and uncertainty map U from the probability volume $\{P_{ij}\}_{j=1}^{N_d}$ for each pixel i, which follows the below:

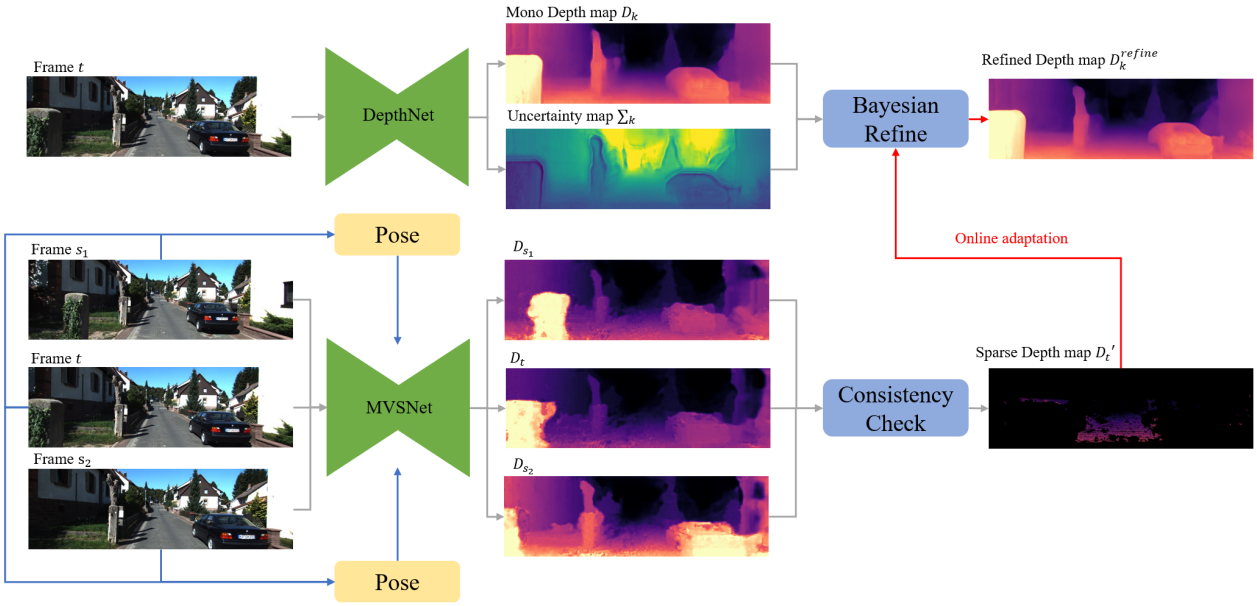


Figure 6.1: The overview of our multi-view monocular depth fusion framework. Our final purpose is to estimate depth of frame t. First, DepthNet predicts single-view depth map D_k and uncertainty map Σ_k as initial guess. Then we select several source views $s_1, s_2, ...$, and feed them together with reference frame t into MVSNet, which predicts depth map D_{s_1}, D_t, D_{s_2} for each view. As we select different set of source views, depth result D_t will differ. We take a multi-view depth consistency check on D_{s_1}, D_t, D_{s_2} to retain inliers, and obtain the sparse projected depth D_t' on frame t. Finally, D_t' is used to refine initial monocular depth D_k and uncertainty Σ_k during an online Bayesian refinement module. So we obtain the final depth estimation D_k^{refine} for frame t.

$$
\begin{aligned}
D_i^{mvs} &= \sum_{j=1}^{N_d} d_j P_{ij} \\
U_i^{mvs} &= f_u(\sum_{j=1}^{N_d} -P_{ij} \log P_{ij})
\end{aligned}
\tag{6.1}
$$

where depth value D_i^{mvs} is computed as the expectation of distribution and uncertainty U_i^{mvs} [22] as the cross entropy over the probability distribution followed by a 2D CNN function layer f_u. Notice that the probability distribution $\{P_{ij}\}_{j=1}^{N_d}$ reflects the confidence of depth value for each pixel, which helps measure the similarity between corresponding image patches and determine whether they are matched.

MVSNet with Uncertainty. Following the strategy in [22, 15], we jointly learning the depth D_t^{mvs} and its uncertainty U_t^{mvs} by minimizing the negative log likelihood of regular photometric loss:

$$\mathcal{L}_{multi} = \frac{1}{V} \sum_{p_i \in V} \frac{\mathcal{L}_{pho}(I_i, D_i^{mvs})}{U_i^{mvs}} + \log U_i^{mvs} \tag{6.2}$$

where uncertainty U_i^{mvs} allows our network to adapt illumination change and occlusions [22]. In most cases, MVSNet exploits sequence information so that we will get more accurate estimation of depth values than DepthNet, except the area of moving objects or low-textured surfaces.

6.2 Depth Consistency Check

Considering the predicted depth of target view D_t^{mvs} is not reliable for all pixels, we adopt a simple multi-view depth consistency check strategy to select confident depth values. Given target view I_t and N source views $\{I_{s_i}\}_{i=1}^{N}$, we first compute depth maps of each source view by selecting it as target, while all other frames as source, and then feed them into cost volume to regress the depth maps $D_{s_i}^{mvs}$. Taking each source view and the target view as a pair $\{I_{s_i}, I_t\}$, with estimated depths and known relative pose, we can check the geometric consistency by considering two criteria: reprojection error and relative depth error. We first project all pixels from I_t to I_{s_i}, and then re-project them back to I_t with sampled source view depths. Besides, we warp the source depth D_{s_i} to the target view as $D_{s_i \to t}^{mvs}$. We define the distance between reprojected pixel and the original one as E_{dist}, and the difference between reprojected depth map $D_{s_i \to t}^{mvs}$ and target depth map D_t^{mvs} as E_{diff}. We consider depth value to be more confident if pixels in the target view satisfying $E_{dist} < e_1$ and $E_{diff} < e_2$, by setting thresholds e_1 and e_2 as 1 and 0.001 respectively. In the following section, we will show how to apply E_{dist} and E_{diff} as the variance of a new observation for different inputs of MVSNet.

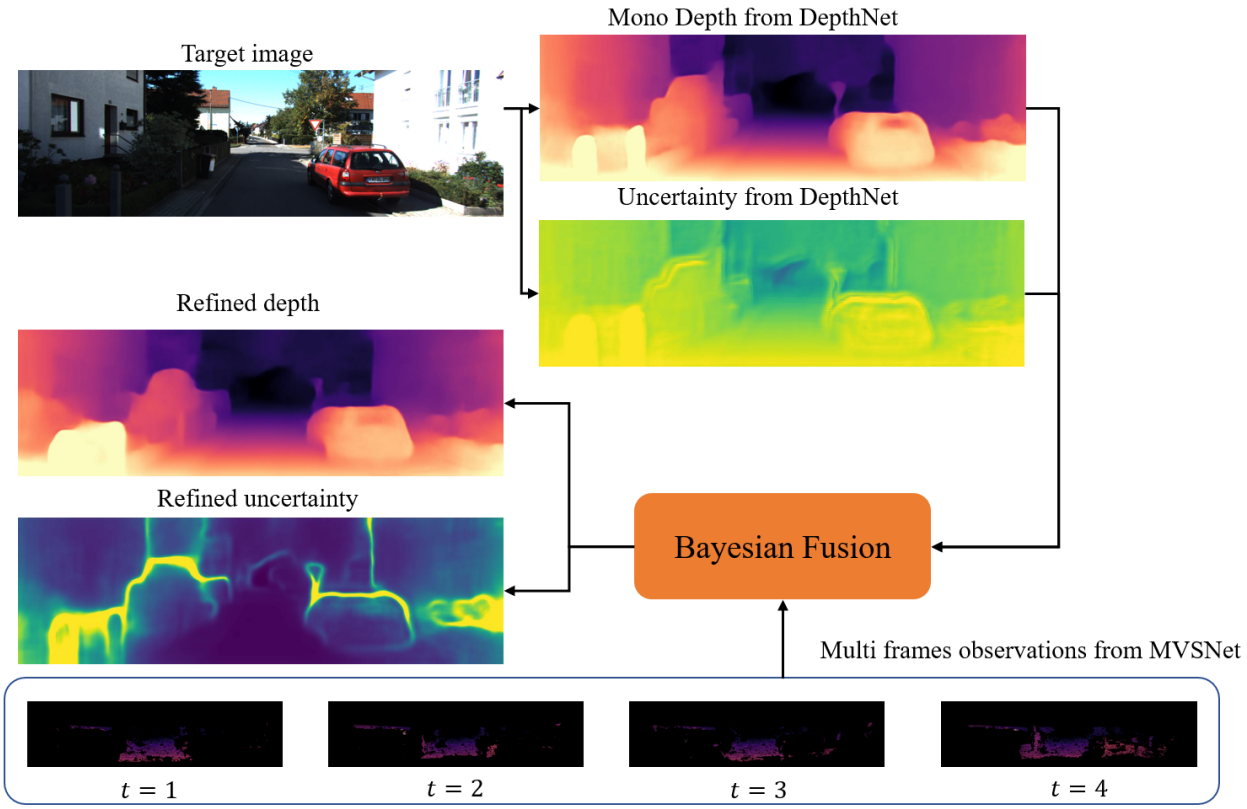


Figure 6.2: Bayesian fusion process. When tested on KITTI, the pretrained DepthNet predicts an initial guess of depth and uncertainty. With incoming observations from MVSNet, both depth and uncertainty will be continuously refined in Bayesian fusion module, where depths become more accurate with decreasing uncertainty.

6.3 Bayesian Online Adaptation

Since [13] proposes a Bayesian online refinement module, we try to apply it to refine the monocular depth map with filtered sparse depth maps from MVSNet [23, 8]. It models the depth estimation as inverse depth as in SVO [4], which is more robust for distant objects. The measurement of inverse depth $z_i = \frac{1}{d_i}$ is modeled with a mixture model of Gaussian and Uniform distribution [4], where a good measurement is normally distributed around the ground-truth z_i and an outlier measurement arises from a uniform distribution in the interval $[z_i^{min}, z_i^{max}]$. For the same reference view I_r, if we choose different source views together with I_r and pass them through MVSNet, we will regress distinct depth maps. Each depth inference can be treated as *new observation* for depth values in

reference view I_r. For every new observation z_i^t:

$$p(z_i^t|z_i, \rho_i^t) = \rho_i^t \mathcal{N}(z_i^t|z_i, \tau_i^2) + (1 - \rho_i^t)\mathcal{U}(z_i^t|z_i^{min}, z_i^{max}) \tag{6.3}$$

where ρ_i^t is the inlier probability which is modelled as Beta distribution and τ_i^2 the variance of a good observation that can be computed geometrically as in our depth consistency check.

The main idea of Bayesian fusion is to find the Maximum A Posteriori (MAP) estimation of z_i^t for each observation, which can be approximated by the product of a Gaussian distribution for z and a Beta distribution for inlier ratio ρ:

$$q(z, \rho|a_t, b_t, \mu_t, \sigma_t^2) = \textit{Beta}(\rho|a_t, b_t)\mathcal{N}(z_t|\mu_t, \sigma_t^2) \tag{6.4}$$

As shown in Figure 6.2, our Bayesian fusion module will update 4 parameters: $\mu_t, \sigma_t^2, a_t, b_t$, within N iterations where N is the number of new observations. The single-view depth estimation d_k provides depth prior and inverse depth uncertainty u_k from monocular *DepthNet*.

The mean and variance of inverse depth can be updated by:

$$\begin{aligned} \mu_t &= C_1' m + C_2' \mu_{t-1} \\ \sigma_t^2 &= C_1'(s^2 + m^2) + C_2'(\sigma_{t-1}^2 + \mu_{t-1}^2) - \mu_{t-1}^2 \end{aligned} \tag{6.5}$$

where C_1', C_2', s, m are computed as in [19]. As seen in Equation (6.5), each iteration will refine the inverse depth values of inliers μ_t with decreasing uncertainty σ_t^2. The inverse depth z_t will tend to be updated close to real inverse depth z_i once the uncertainty value σ_t^2 is lower than a threshold. The parameters are initialized as follows:

$$\begin{aligned} &\mu_i^0 = \frac{1}{d_k}, \qquad \sigma_i^0 = u_k, \qquad z_i^{max} = \mu_i^0 + \sigma_i^0 \\ &z_i^{min} = max(\mu_i^0 - \sigma_i^0, 10^{-6}) \end{aligned} \tag{6.6}$$

After N iterations in Bayesian fusion, we obtain the final refined depth map $D^r = \frac{1}{\mu_i^{N-1}}$. As

Algorithm 1 Bayesian Updating Algorithm

Input: Sparse inverse depths $\{z_i^t\}_{t=0}^{N-1}$ of N observations
Parameter: $\mu_i, \sigma_i^2, a_i, b_i$

1: Initialize parameters using (6.6).
2: **while** ${\sigma_i^t}^2$ not converges **do**
3: Choose a new observation z_t and obtain its variance τ by adopting depth consistency check.
4: **for** every pixel i **do**
5: **if** pixel i is an inlier **then**
6: Update $\mu_i^t, {\sigma_i^t}^2$ using (6.5)

our monocular DepthNet predicts depth D_t and uncertainty σ_t, we adopt negative log-likelihood loss together with photometric and smooth loss in Equation (4.4) as the total self-supervised loss:

$$\mathcal{L}_{total} = \frac{|1/D^r - 1/D_t|}{\sigma_t} + \log \sigma_t + \mathcal{L}_{mono} \tag{6.7}$$

Our training process contains two stages: In the first stage, we minimize $\mathcal{L}_{mono}$ in Equation (3.4) to pretrain monocular DepthNet and $\mathcal{L}_{multi}$ in Equation (6.2) for MVSNet as in unsupervised learning of depth estimation. In the second stage, for every time step when we obtain a new observation from MVSNet, we online re-train DepthNet by minimizing Equation (6.7) to refine depth of inliers, while $\mathcal{L}_{mono}$ retains the photometric consistency and smoothness. In the next chapter, we will show some of evaluation results and give some analysis about them.

CHAPTER 7

EXPERIMENTAL EVALUATION

In this chapter, we evaluate depth results of our system **FusionDepth** using metrics introduced in Section 3.3. We also evaluate the performance of uncertainty estimation on our online refinement module, using sparsification curves in Section 4.4.

7.1 Experimental Setup

For monocular architecture, we adopt [7] and add a 3×3 convolution layer at the output to generate depth uncertainty map. The backbone of our MVSNet is a cascade architecture as in [8] and we add a shallow 2D CNN to convert the entropy map to uncertainty map. Regarding numerical stability [11], both uncertainty outputs are modelled as the log-variance in order to avoid zero values of the variance.

We use training-time color and flip augmentations on images being fed to the depth networks. We implement our models on PyTorch and train them on 4 Telsa V100 GPUs. The images are resized to 192×640 for KITTI dataset. The monocular DepthNet and MVSNet are both pretrained first in a self-supervised manner for 10^5 iterations by minimizing Equation (4.4) and (6.2). We use the Adam Optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. We train both models for 20 epochs with a batch size of 12. The initial learning rate is 10^{-4} and decreased to 10^{-5} after 15 epochs. We set the $SSIM$ weight as $\alpha = 0.85$ and smooth loss weight as $\lambda = 10^{-3}$. In Bayesian Fusion, we choose 4 source views for each target I_t, includes I_{t-1}, I_{t-2}, I_{t+1}, I_{t+2}. For each measurement, we select 2 of them as source views, together with target I_t as inputs of MVSNet. The threshold of Depth Filter e_1 and e_2 are set as 1 and 0.001 respectively. The fusion step takes 4 iterations, with source views of $\{I_{t-1}, I_{t+1}\}, \{I_{t-2}, I_{t+1}\}, \{I_{t-1}, I_{t+2}\}, \{I_{t-2}, I_{t+2}\}$ for each iteration respectively.

	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Sfm-learner	0.327	3.113	9.522	0.403	0.423	0.701	0.848
SC	0.163	0.964	4.913	0.224	0.776	0.932	0.977
Monodepth2	0.120	0.779	4.611	0.178	0.846	0.962	0.988
Ours (p.d.)	0.118	0.627	4.111	0.171	0.850	0.967	0.991
Ours (p.k.)	**0.100**	**0.473**	**3.520**	**0.149**	**0.885**	**0.976**	**0.994**

Table 7.1: Quantitative results of depth estimation on KITTI odometry set 09, 10 for distance up to 80m. For error evaluating indexes, Abs Rel, Sq Rel, RMSE and RMSElog, lower is better, and for accuracy evaluating indexes, $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$, higher is better. Sfm-learner: [26], SC: [1], Monodepth2: [7]. **p.d.**: MVSNet pretrained on DTU [23]. **p.k.**: MVSNet pretrained on KITTI.

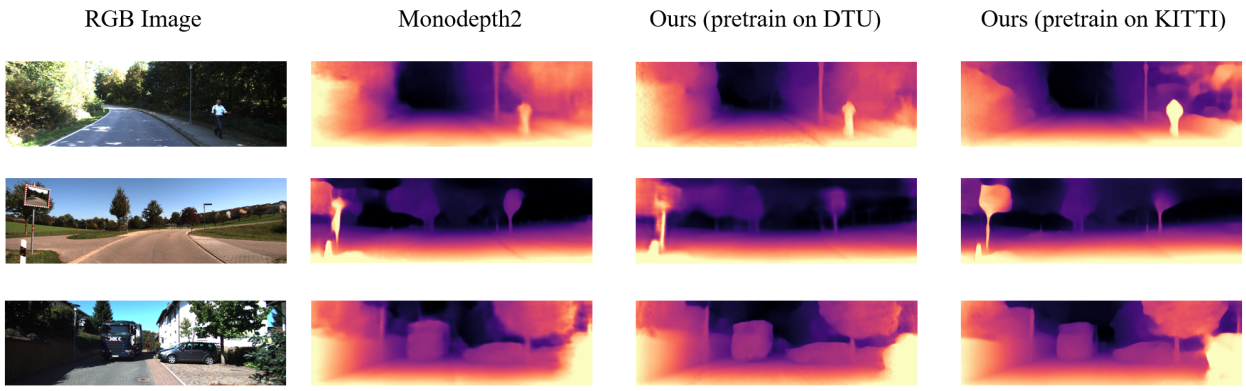


Figure 7.1: **Qualitative monocular depth estimation performance** comparing our methods with previous monocular SOTA. We pretrain our network on DTU and KITTI, and test on KITTI. The results show that when adopting Bayesian fusion of our method to combine multi-view information, it will predict higher quality depths especially at edges as we apply Bayesian refinement even though the MVSNet model is pretrained in an unseen scene.

7.2 Depth Estimation Performance

Since MVSNet strongly requires accurate camera poses to build cost volume, instead of testing on KITTI Eigen Split [3], we adopt the KITTI odometry dataset which contains 11 driving sequences with ground-truth poses and depth available, except for 03 where ground-truth depth is not acquirable. We train the model on sequence 00-08 and evaluate the monocular depth estimation performance on 09 and 10. For 00-08, 35186 images are used for training and 3912 for validation. For 09-10, 2778 images are used for test. We compare our method with [26], [1], [7] on the same learning setting and dataset, and show the results in Table 7.1. All groundtruth depth maps

are capped at 80 meters. The result shows that our method with pretrained MVSNet on DTU and KITTI both outperforms SOTA monocular methods.

7.3 Ablation Studies

To further explore the performance improvements that our network provides, we take some extra ablative analysis on the different components.

Uncertainty Metrics. Since we have refined uncertainty map in our model, we try to evaluate whether our uncertainty estimation can properly reveals the real error of predicted depth maps. When we plot the curve with original total mean metrics, as seen in green curves named *Random Curve*, we plot all of the three curves for each depth metric: (Abs Rel, RMSE, $\delta \geq 1.25$), before and after Bayesian refinement, as seen in Figure 7.2. In the upper row, an initial guess of uncertainty before refinement is of random high value for each pixel, so uncertainty does not show the similar distribution to depth metrics and *Sparsification Curve* (red) is far away from *Oracle* (blue). In the lower row, after Bayesian refinement, uncertainty of many inliers decrease to a proper threshold, thus *Sparsification Curve* seems closer to *Oracle* (blue), which proves a conclusion that Bayesian fusion module can make our predicted uncertainty map more intepretable due to the consistency between uncertainty and real depth error.

Generalization Distinct from monocular depth estimation learning with photometric error [7], learning-based multi-view stereo matching methods encourage network to match high-level features and show stronger robustness when inferring depth maps in an unseen scene. Thus, it is possible for us to pretrain MVSNet on another unseen indoor scene with ground truth depth provided, and then test on KITTI while taking Bayesian fusion steps. Since during the whole training process, we do not use any ground truth depth of KITTI, our training is still self-supervised. By adopting the pretrained model from DTU [23] where MVSNet converges smoothly, we take multi-view depth consistency check on the output depth maps when testing on KITTI, the result is shown in Fig 7.3. The model pretrained on DTU provides roughly half inliers of that pretrained on KITTI, the former number of inliers is still large for refinement. As seen in Table 7.1 again, our method with pretrained model on DTU still outperforms the monocular ones.

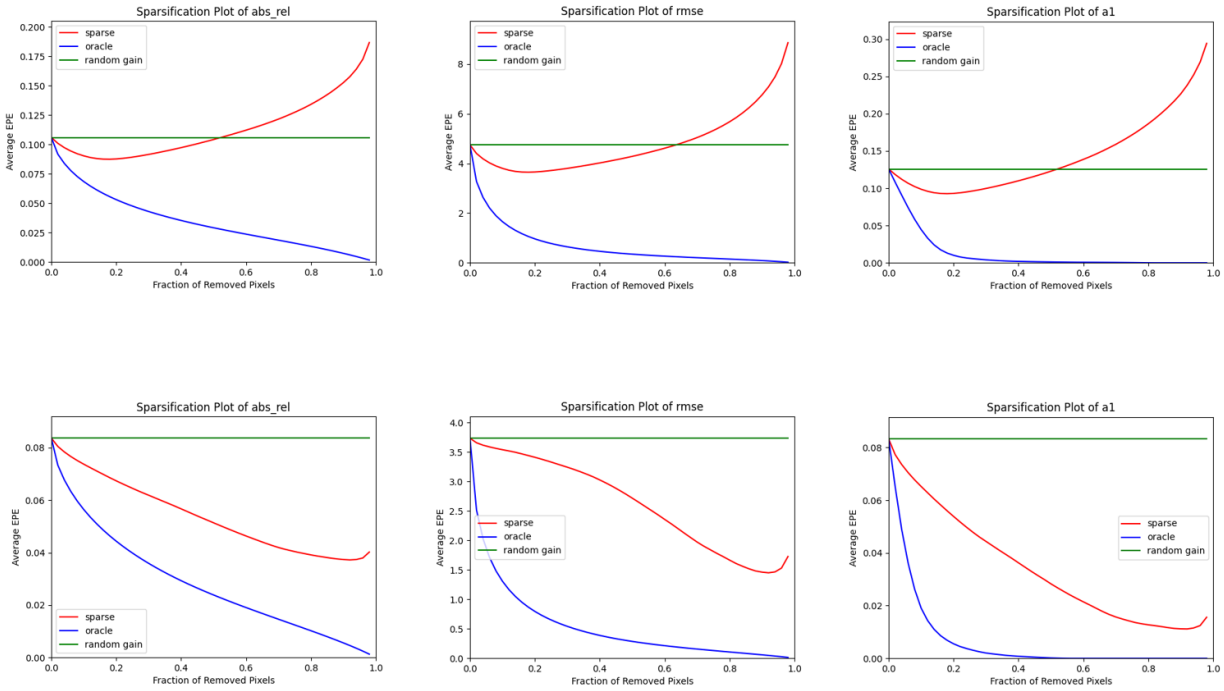
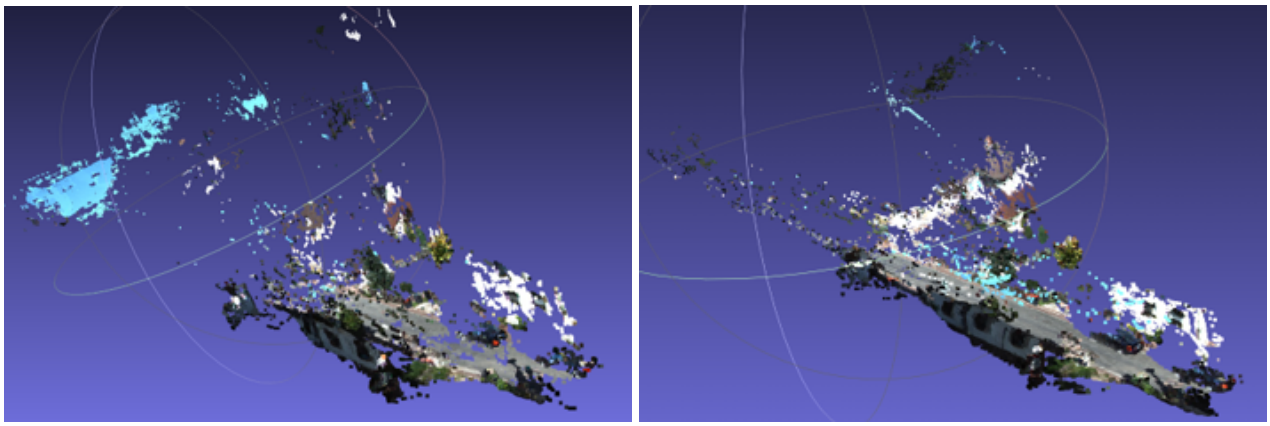


Figure 7.2: **Sparsification curves** of predicted uncertainty before and after Bayesian refinement. After refinement, we find that the Sparsification Curves are highly revised and moved close to Oracle ones, which means that the estimated uncertainty values become more intepretable.



(a) Pretrained on DTU: 153276 inliers (b) Pretrained on KITTI: 276464 inliers

Figure 7.3: MVSNet pretrained on DTU [23] and KITTI. A pretrained MVSNet model in unseen environment can generate sufficient inliers for Bayesian fusion even if less than the one pretrained in KITTI, after multi-view depth consistency check.

CHAPTER 8

CONCLUSION

We analyze several self-supervised depth estimation frameworks in recent years in this thesis, and inspired from them, we propose an online learning multi-frame monocular combined system for depth estimation with the help of multi-view stereo matching and Bayesian fusion. The predicted single-view depth is continuously refined with incoming output from cost volume with different source views as input. The updating process follows Bayesian formula and our final refined depth will converge to the ground truth once the refined uncertainty decreases to a threshold. Our system leverages the generalization ability of cost volume to make up for the shortage of monocular depth estimation, so that we achieve the purpose of generating both accurate and complete depth.

REFERENCES

[1] Jia Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, 2019.

[2] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[3] Christian Puhrsch David Eigen and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, 2014.

[4] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *ICRA*, 2014.

[5] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. In *CVPR*, 2018.

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017.

[7] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[8] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020.

[9] Baichuan Huang, Hongwei Yi, Can Huang, Yijia He, Jingbin Liu, and Xiao Liu. M3VSNet: Unsupervised multi-metric multi-view stereo network. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3163–3167. IEEE, 2021.

[10] Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*, 2018.

[11] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *NeurIPS*, 2017.

[12] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: Learning SFM from SFM. In *ECCV*, 2018.

[13] Shunkai Li, Xin Wu, Yingdian Cao, and Hongbin Zha. Generalizing to the Open World: Deep Visual Odometry with Online Adaptation. In *CVPR*, 2021.

[14] Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. Don't Forget The Past: Recurrent Depth Estimation from Monocular Video, 2020.

[15] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *CVPR*, 2020.

[16] Michaël Ramamonjisoa, Yuming Du, and Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *CVPR*, 2020.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[18] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity Invariant CNNs, 2017.

[19] George Vogiatzis and Carlos Hernández. Video-based, real-time multi-view stereo. *Image and Vision Computing*, 29(7):434–441, 2011.

[20] Rui Wang, Stephen M. Pizer, and Jan-Michael Frahm. Recurrent Neural Network for (Un-)supervised Learning of Monocular VideoVisual Odometry and Depth, 2019.

[21] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *CVPR*, 2021.

[22] Nan Yang, Lukas Von Stumberg, Rui Wang, and Daniel Cremers. D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, 2020.

[23] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[24] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, Ravi Garg, and Ian Reid. DF-VO: What Should Be Learnt for Visual Odometry? In *ICRA*, 2020.

[25] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network, 2020.

[26] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[27] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency. In *ECCV*, 2018.