Bayesian Deep Basis Fitting for Depth Completion with Uncertainty

Chao Qu  Wenxin Liu  Camillo J. Taylor
University of Pennsylvania
{quchao, wenxinl, cjtaylor}@seas.upenn.edu

Abstract

In this work we investigate the problem of uncertainty estimation for image-guided depth completion. We extend Deep Basis Fitting (DBF) [54] for depth completion within a Bayesian evidence framework to provide calibrated per-pixel variance. The DBF approach frames the depth completion problem in terms of a network that produces a set of low-dimensional depth bases and a differentiable least squares fitting module that computes the basis weights using the sparse depths. By adopting a Bayesian treatment, our Bayesian Deep Basis Fitting (BDBF) approach is able to 1) predict high-quality uncertainty estimates and 2) enable depth completion with few or no sparse measurements. We conduct controlled experiments to compare BDBF against commonly used techniques for uncertainty estimation under various scenarios. Results show that our method produces better uncertainty estimates with accurate depth prediction.

1. Introduction

As we seek to incorporate learned modules in safety critical applications such as autonomous driving, reliable uncertainty estimation becomes as critical as prediction accuracy [59]. Depth completion is one such task where well-calibrated uncertainty estimates can help to enable robust machine perception. Deep Convolutional Neural Networks (CNNs) are commonly used to solve structured regression problems like depth prediction due to their strong expressive power and inductive bias [12]. However, in its native form, a CNN only produces a point estimate, which offers few insights into whether or where its output should be trusted. Many probabilistic deep learning methods have been proposed to address this issue [44, 17], but they often fail to output calibrated uncertainty [23] or become susceptible to distributional shift [50]. Moreover, these methods can be expensive to compute due to the need for test time sampling [18] or inference over multiple models [36].

In this work, we propose a method for depth completion with uncertainty estimation that avoids the above limitations. Our approach builds on the idea of Deep Basis Fitting (DBF) [54]. DBF replaces the last layer of a depth completion network with a set of data-dependent weights. These weights are computed by a differentiable least squares fitting module between the penultimate features and the sparse depths. The network can also be seen as an adaptive basis function which explicitly models scene structure on a low-dimensional manifold [4, 60]. It can be used as a replacement to the final layer (with no change to the rest of the network or training scheme), which greatly improves depth completion performance.

We extend DBF by formulating it within a Bayesian evidence framework [3]. This is done by placing a prior distribution on the DBF weights and marginalizing it out during inference. Such last-layer probabilistic approach have been shown to be reasonable approximations to full Bayesian Neural Networks [34], while providing the advantage of tractable inference [48]. This is conceptually similar to...
Neural Linear Models (NLMs) [58] with the notable distinction that we perform Bayesian linear regression on each image as opposed to the entire dataset.

A Bayesian treatment also enables depth completion with highly sparse data. In DBF, when the number of sparse depths falls below the dimension of the bases, the underlying linear system becomes under-determined. We show that by learning a shared prior across images, our method is able to handle any number of sparse depth measurements.

We name our approach Bayesian Deep Basis Fitting (BDBF) and summarize its advantages: 1) It can be used as a drop-in replacement to the final layer of many depth completion networks and outputs uncertainty estimates (in the form of per-pixel variance). 2) Compared to other uncertainty estimation techniques, it produces higher quality uncertainty with one training session, one saved model and one forward pass, without needing extra parameters or modifications to the loss function. 3) It can handle any sparsity level, with performance degrading gracefully towards a pure monocular method when the number of depth measurements goes to zero.

2. Related Work

2.1. Uncertainty Estimation for Neural Networks

We start by reviewing uncertainty estimation techniques for neural networks. There are two types of uncertainty that one could model: epistemic (model), which describe the uncertainty in the model and aleatoric (data), which reflects the inherent noise in the data [33]. Modeling uncertainty in neural networks can be achieved by placing probabilistic distributions on network weights. Such networks are called Bayesian Neural Networks (BNN) [44].

Direct inference in BNNs is intractable for continuous variables, and different approximation techniques have been explored [44, 29, 22, 5, 46, 45, 30]. However, they don’t scale well to large datasets and complex models, and are thus impractical for current vision tasks.

Gal et al. [18] proposed the use of dropout as an approximate variational inference method for BNNs. However, their method requires multiple forward passes to obtain Monte Carlo model estimates at test time. Another research direction is assumed density filtering (ADF) [49] which can be viewed as a single Expectation Propagation pass. Gast et al. [20] chose to propagate activation uncertainties without probabilistic weights in a lightweight manner, which requires modifying the layer operations based on moment matching.

Predictive methods directly output mean and variance of some parametric distribution by minimizing the negative-log likelihood (NLL) loss [47]. They only require small changes to the original network by adding a variance prediction head. This simplicity makes it a popular choice among recent works [31, 39].

Ensemble methods either train multiple models independently with different initializations (bootstrap) [36] or save several copies of weights at different stages during training (snapshot) [26]. These methods only model epistemic uncertainty but can be combined with a predictive one to model data uncertainty. They achieve good performances in various experimental settings [24, 52, 28], but still need multiple inference passes at test time, which makes them less suitable for resource constrained platforms.

Table 1 summarizes the aforementioned approaches and highlights the difference compared to ours. In Sec. 4.1 we describe in detail the methods that we evaluate against.

2.2. Uncertainty Estimation in Depth Completion

Great progress has been made in the past few years on depth completion ranging from high-density completion for RGB-D/ToF cameras [38, 15, 68, 70], to mid-density completion from LiDAR sensors [64, 43, 10, 61, 9, 8, 71, 37, 67, 69]. Recently, there has also been rising interests in low-density completion from map points generated by Visual-SLAM or Visual-Inertial Odometry [65, 66, 55, 72]. Unlike systems that are designed for a particular sparsity or sensing modality, our proposed method can be seen as a general component for depth completion similar to DBF [54].

A complete review of depth completion literature is out of the scope of this work, we instead focus on methods that also estimate uncertainty. Gansbeke et al. [19] predict depth and confidence weights for both color and depth branch and fuse them based on the confidence maps. Qiu et al. [53] adopt a similar strategy, but additionally guide the depth branch via surface normal prediction. Xu et al. [67] use a shared encoder and multiple decoders to predict surface normal, coarse depth and confidence, then use an anisotropic diffusion process to produce refined depth. Park et al. [51] instead use a single encoder-decoder network to predict initial depth, affinities and confidence be-

<table>
<thead>
<tr>
<th>Method</th>
<th>#T</th>
<th>#M</th>
<th>#F</th>
<th>Alea.</th>
<th>Epis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive [47, 31]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dropout (Predictive) [18, 31]</td>
<td></td>
<td></td>
<td>K</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Snapshot (Predictive) [26]</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bootstrap (Predictive) [36]</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Proposed (BDBF)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

16148
before applying a non-local spatial propagation to produce the final depth. Note that the uncertainties produced by the above methods are not calibrated and are only used internally. Therefore, they are not readily useful to downstream tasks that require probabilistic reasoning. This type of uncertainty estimation can also be seen as a simplified version of the predictive method in [47] without the NLL loss.

Few works have tried to evaluate the quality of depth completion uncertainty. Eldesokey et al. [13] present a probabilistic normalized convolution [14] that estimates confidence of both input sparse depths and output dense prediction, for unguided depth completion. Gustafsson et al. [24] compared several uncertainty estimation methods applied to depth completion in the same spirit as [52]. We follow their approach and provide a systematic comparison of our proposed method against the best performing schemes from [24, 52] and demonstrate superior performance and efficiency across a range of datasets.

3. Method

3.1. Problem Formulation

Let \( D = \{ (x_n, y_n) \}_{n=1}^{N_D} \) be a dataset containing \( N_D \) samples. We wish to learn a neural network \( f \) that maps \( x \) to \( y \). In depth completion, the input \( x \) is usually an image and sparse depth pair \((I, S)\), and the output \( y \) is the predicted depth map \( D \). We refer to \( f_\theta \) as the basis network and its output \( \Phi \) a set of depth bases [54]. \( \Phi \) is then reduced by a linear layer \( f_w \) to \( z \), which is then mapped to positive depth values via a nonlinear activation function \( g \).

\[
y = f(x) = g \circ f_w \circ f_\theta (x) = g \circ f_w (\Phi) = g(z)
\]

With a slight abuse of notation, we call \( z \) the latent variable and choose \( g \) to be the exponential function [12], so \( z \) effectively corresponds to log depth. An overview of our method is shown in Figure 2.

3.2. Bayesian Deep Basis Fitting

We choose to model the distribution of each pixel in the latent space \( z \) rather than in the target space \( y \), since depth is strictly positive and may span several orders of magnitude [57]. Assuming Gaussian noise in the latent space, we define our model to be

\[
z_i = w_\top \phi_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \beta^{-1})
\]

where \( \phi_i \) denotes the basis entries corresponding to the latent pixel value \( z_i \) and \( \beta \in \mathbb{R} \) is a precision parameter that corresponds to the inverse variance of the noise. Assuming that the errors at each pixel are independent, the likelihood function is

\[
p(z|x, w) = \mathcal{N}(z|\Phi w, \beta^{-1} I)
\]

here \( \Phi \) is the \( N \times M \) design matrix with \( N \) the number of sparse depths and \( M \) the dimension of \( w \). It is assembled by extracting the basis entries at the pixel locations specified by \( S \).

Given a suitable prior on the last-layer weights \( p(w) = \mathcal{N}(w_0, \alpha^{-1} \Sigma_0) \), where \( \alpha \in \mathbb{R} \) is a precision parameter to scale the covariance \( \Sigma_0 \), the posterior distribution of \( w \) can be computed analytically following Bayes’ rule [3]:

\[
p(w|x, z) = \mathcal{N}(w|\mathbf{m}, \Sigma) \propto \mathcal{N}(w|w_0, \alpha^{-1} \Sigma_0) \cdot \mathcal{N}(z|\Phi w, \beta^{-1} I)
\]

where the mean and covariance are given by

\[
\mathbf{m} = \Sigma (\alpha \Sigma_0^{-1} w_0 + \beta \Phi^\top z) \quad \Sigma = (\alpha \Sigma_0^{-1} + \beta \Phi^\top \Phi)^{-1}
\]

The latent predictive distribution for a pixel at test time is

\[
p(z|x, z) = \int p(z|x, w)p(w|x, z)dw = \mathcal{N}(z|m^\top \phi_*, \phi_*^\top \Sigma_*)
\]

The Gaussian assumption is made solely for the purpose of tractable inference. In practice, the shape of the predictive distribution depends heavily on the loss function. Since we use L1 loss for training, we employ a Laplace distribution as its parametric form for evaluating uncertainty [28, 4]. Additionally, a robust norm like Huber [27] can be applied if outliers are present in the target [63].

3.3. Training

**Loss Function.** The standard way of learning a Bayesian regressor is by maximizing the marginal likelihood function with respect to the parameters \( \theta \) of the basis function \( f_\theta \),

\[
\log p(z|x, \alpha, \beta) = \frac{1}{2} (N \ln \beta + M \ln \alpha - N \ln 2\pi - E(\mathbf{m}) + \ln |\Sigma| - \ln |\Sigma_0|)
\]

\[
E(w) = \beta \|z - \Phi w\|^2 + \alpha \|w - \mathbf{m}_0\|^2_{\Sigma_0}
\]
where \( \|v\|_A = v^T A^{-1} v \) is the Mahalanobis norm. This is known in literature as type 2 maximum likelihood [3].

Directly maximizing (10) has two practical issues. First, one needs to estimate the hyperparameters \( \alpha \) and \( \beta \), which adds large overhead during training. Second, gradients need to be back-propagated through costly operations like matrix inversion and determinant. Together, they pose challenges to the training phase and often produce empirically similar results to a point estimate [58]. We avoid these issues by assuming sufficient sparse points in training \( (N \gg M) \). This renders the linear system over-determined, which allows us to treat the prior \( p(w) \) as infinitely broad. The solution in (6) therefore reduces to a maximum likelihood (ML) one which can be computed efficiently in one pass [3].

\[
w_{\text{ML}} = (\Phi^T \Phi)^{-1} \Phi^T z, \quad \beta_{\text{ML}}^{-1} = \frac{1}{N} \|z - \Phi w_{\text{ML}}\|^2
\]  

(12)

The predicted mean and variance of \( z \) at training time are given by the following according to (9)

\[
\mu = w_{\text{ML}} \phi, \quad \sigma^2 = \beta_{\text{ML}}^{-1} \Phi (\Phi^T \Phi)^{-1} \phi
\]  

(13)

Given the above results, we can minimize the Negative Log-Likelihood Loss (NLL) assuming a Laplace distribution [28], which is defined for a single pixel as

\[
-\log p(z|\mu, b) \propto \frac{1}{b} \left( \frac{\mu - z}{b} + \log b \right) + 2b^2 = \sigma^2
\]  

(14)

However, this would still involve back-propagation through a costly matrix inversion as in (12). We find in our experiments that this sometimes causes numerical instability during training and incurs a visible decrease in prediction accuracy. Therefore, we opt to minimize directly the L1 loss for supervised learning. This allows our network to be trained in its original form without suffering from the performance drop caused by the NLL loss [40].

**Uncertainty Calibration.** Not using the likelihood loss comes with the risk of over-confidence in uncertainty estimation, since there is no explicit penalty on the variance prediction. As the number of parameters in \( \theta \) is usually on the order of millions, the noise variance \( \beta^{-1} \) will be pushed towards zero [48]. One solution is to regularize \( \theta \) using an L2 regularization term in the optimizer. This introduces an extra hyperparameter to tune: a small regularization would not prevent overfitting, while a large one will render the feature bases inexpressive [62]. We notice empirically that the amount of overconfidence in our method is consistent during training and validation. Therefore, we take a pragmatic approach and propose to solve this problem in terms of estimator consistency [2], measured by normalized estimation error squared (NEES). For a Laplace distribution, NEES is defined as \( \varepsilon = (\mu - z)^2 / b^2 \). We record the average NEES \( \bar{\varepsilon} \) at training time for the last epoch, and use it to scale the variance accordingly during inference with \( \sigma^2 = \bar{\varepsilon} \sigma^2, \) which attempts to make the final prediction consistent. Note that the scaling factor is computed entirely at training without any additional data and NEES is not used as a loss function.

**Shared Prior.** Although the prior \( p(w) \) is not used in training, we still need it for inference. Rather than estimating a different prior for each image, we make another simplifying assumption that there exists a shared prior for the entire dataset. This aligns with our observation from experiments that \( p(w) \) shows a relatively sharp peak. Given our training strategy, we adopt a frequentist approach and collect all ML solutions of weights \( w_{\text{ML}} \) within one training epoch. The mean, \( m_0 \), and covariance, \( \Sigma_0 \), can then be computed from this set. Having a shared prior enables robust depth completion from a few sparse depth measurements.

### 3.4. Inference

Inference follows the standard evidence framework [3]. We use EM [11] to estimate the hyperparameters \( \alpha \) and \( \beta \). The re-estimation equations are obtained by maximizing the expected complete-data log likelihood with respect to \( \alpha, \beta \)

\[
\alpha^{-1} = \frac{1}{M} \left( \|m - m_0\|^2 \Sigma_0^{-1} + \text{tr}(\Sigma_0^{-1} \Sigma) \right)
\]  

(15)

\[
\beta^{-1} = \frac{1}{N} \left( \|z - \Phi m\|^2 + \text{tr}(\Phi^T \Phi \Sigma) \right)
\]  

(16)

where \( \text{tr}(\cdot) \) is the matrix trace operator. The re-estimated \( \alpha \) and \( \beta \) are then plugged back into (6) and (7) to recompute \( m \) and \( \Sigma \). We initialize this process empirically with \( \alpha = 1 \) and \( \beta = \sqrt{N} \) and set the maximum number of iterations to 8. In practice we reach convergence within 2 to 3 iterations when \( N \gg M \), thus incurring only a small computation overhead. In the extreme case when \( N \to 0 \), we rely on the shared prior alone for a pure monocular prediction.

\[
\mu = m_0 \phi, \quad \sigma^2 = \phi^T \Sigma_0 \phi
\]  

(17)

### 4. Evaluation

In this section, we show results on various datasets that our method outperforms baseline approaches in uncertainty estimates with accurate depth prediction, and remains resilient to sparsity change and domain-shift.

#### 4.1. Baselines

We describe three baselines for uncertainty estimation that we compare to, which are shown to have strong performance [52, 24] and can be evaluated under controlled settings. As discussed in Section 3.2, all methods output mean and variance of the latent prediction before \( g(\cdot) \), and are trained with either L1 loss or its NLL variant (14).

For empirical methods, we choose Snapshot Ensemble [26] (snap) which can be completed in one training session.
such that all methods have the same training budget. The mean and variance are computed using $K$ snapshots.

$$\mu = \frac{1}{K} \sum_{i=1}^{K} \mu_i, \quad \sigma^2 = \frac{1}{K} \sum_{i=1}^{K} (\mu_i - \mu)^2 \quad (18)$$

For predictive methods (log) [47], we attach a variance prediction head parallel to the depth prediction and train with the NLL loss. Finally, we combine the above two (snap+log) [31] to form a predictive ensemble method.

$$\mu = \frac{1}{K} \sum_{i=1}^{K} \mu_i, \quad \sigma^2 = \frac{1}{K} \sum_{i=1}^{K} ((\mu_i - \mu)^2 + \sigma_i^2) \quad (19)$$

4.2. Datasets

Virtual KITTI 2. The VKITTI2 dataset [6] is an updated version to its predecessor [16]. We use sequences 2, 6, 18 and 20 with variations clone, morning, overcast and sunset for training and validation, and clone in sequence 1 for testing. This results in 6717 training and 447 testing images. The sparse depths are generated by randomly sampling pixels that have a depth less than 80m [24]. Ground truth depths are also capped to 80m following common evaluation protocols. All images are downsampled by half.

NYU Depth V2. The NYU-V2 [56] dataset is comprised of various indoor scenes recorded by an off-the-shelf RGB-D camera. We use the 1449 densely labeled pairs of aligned RGB and depth images, and split it into approximately 75% training and 25% testing. The same depth sampling strategy is adopted as above. Note that we intentionally choose this small dataset (as opposed to the full) to evaluate uncertainty estimation under data scarcity [62].

KITTI Depth Completion. We also evaluate on the KITTI depth completion dataset [64] following its official train/val split. For all experiments other than the official submission, we down sample both the image and depth by half.

4.3. Implementation Details

Network architecture. We use the same basis network for all methods, which is an encoder-decoder architecture with skip connection similar to that in [54]. We use a MobileNetV2 [57] pretrained on ImageNet [35]. The decoder outputs a set of multi-scale bases [54], which are then upsampled to the input resolution and concatenated together to form a final 63-dimensional basis. For baseline methods we initialize the bias of depth prediction head with the average log depth of the dataset and let the variance head predict an initial variance of 1. Our method, however, requires no initialization. When using sparse depths as network input, we adopt the two-stage approach from [66] which first scaffolds the sparse depths by interpolation and then fuses it with the first layer of the encoder via convolution. Note that this depth pre-processing step is orthogonal to the uncertainty estimation techniques and we choose this approach for its simplicity and applicability to both mid- and low-sparsity. All networks use the same setup unless otherwise stated.

Training parameters. For training we use the Adam optimizer [32] with an initial learning rate of 2e-4 and reduce it by half every 5 epochs following [42, 54]. We train our method for 20 epochs and all others for 30. This is to account for the increased training time using our method. For Snapshot Ensemble, we follow the original paper [26] and use the cyclic annealing scheduler from [41] with the same initial learning rate as before. we train for 5 epochs per cycle, and discard the worst snapshot, which leaves us with 5 snapshots. All training is carried out on a single Tesla V100 GPU with the same batch size and random seed. For data augmentation we apply a random horizontal flip with a probability of 0.5 and a small color jitter of 0.02.

4.4. Metrics

Depth prediction metrics. We evaluate depth completion performance using standard metrics [12]. Specifically, we report MAE, RMSE and accuracy ($\delta$-threshold) on depth. Due to space limitation, we only report $\delta_1 < 1.25$.

Uncertainty estimation metrics. Unlike depth prediction which can be compared to ground truth, the true probability density function of depth is not available. This makes evaluating uncertainty estimates a difficult task in itself. Here,
we describe three popular metrics commonly found in literature for uncertainty estimates. Note that each metric has its own advantages and drawbacks, we seek to provide a more comprehensive evaluation by reporting all three.

1) Area Under the Sparsification Error curve (AUSE). Sparsification plots [1] are commonly used for measuring the quality of uncertainty estimates. Given an error metric (e.g., MAE), we sort the prediction errors by their uncertainty in descending order and compute the error metric repeatedly by removing a fraction (e.g., 1%) of the most uncertain subset. An oracle sparsification curve is obtained by sorting using the true prediction errors. AUSE is the area between the sparsification curve and the oracle curve. This normalizes the oracle out and can be used to compare different methods [28]. Note that AUSE is a relative measure of uncertainty quality, since its computation relies on the order of predicted uncertainties.

2) Area Under the Calibration Error curve (AUCE). For an absolute measure of uncertainty estimation quality, [24] proposes to generalize the Expected Calibration Error (ECE) [23] metric to regression. For Laplace distributions, given mean $\mu$ and variance $\sigma^2$, we construct prediction intervals $\mu \pm \Psi^{-1}(1-\frac{p}{2})b$ for $p \in (0, 1)$, where $\Psi$ is the CDF of the unit Laplace distribution. For each value of $p$, we compute the proportion of pixels $\hat{p}$ for which the true target falls within the predicted interval. For a well-calibrated model, $\hat{p}$ should closely match $p$. The Calibration Error curve is defined as $|\hat{p} - p|$, and AUCE is the area under this curve. Like ECE, AUCE is not a proper scoring rule [50], as there exists trivial solutions which yield perfect scores.

3) Negative Log-likelihood (NLL). NLL (14) is commonly used to evaluate the quality of model uncertainty on a held-out dataset [50]. It is a proper scoring rule [21], but over-emphasizes tail probabilities [7] and cannot fully capture posterior in-between uncertainty [62].

4.5. Results

Mid-density Depth Completion. In this setting, we train all methods with 5% sparsity. Table 2 shows quantitative results when testing on the same dataset under the same sparsity level. This is considered an in-distribution test. We see significant improvements of our method in almost all metrics. Figure 3 shows qualitative results of one sample from the VKITTI2 test set. Compared to others, bdhf not only predicts higher quality depths but also sharper uncertainties that closely match the true prediction errors. This indicates that the learned depth bases in ours is expressive for predicting both depth and uncertainty. Note that when there’s enough data, bdhf will reduce to dbf as expected, but the Bayesian formulation allows for smooth transition to a low sparsity level by incorporating a learned prior.

![Figure 4: (a) Qualitative results of bdhf on one test image from NYU-V2. (b) Absolute log depth error (blue line) and 3b bounds (blue shades) for a single row of pixels (red line) from the image. (c) Normalized error density of the entire image compared to a unit Laplace distribution (red line). All axes are of the same scale within each column.](image1)

![Figure 5: (a) Sparsification error, (b) calibration error and (c) sparsity change plots of NLL vs. MAE with 5% sparsity on VKITTI2 (top) and NYU-V2 (bottom). Sparsification and calibration plots are generated using 5% test sparsity. Sparsity change plots are generated with varying test sparsity from 5% to 1%.](image2)
distribution. \textit{log} and \textit{snap+log} produce decent relative uncertainties (AUSE) but are not well-calibrated (AUCE). Figure 5 (a) (b) show sparsification error and calibration plots for the in-distribution test on both datasets, which are used to compute AUSE and AUCE respectively. We see that predictive method (\textit{log}) performs similar to its ensemble variant \textit{snap+log} and both are better than the pure ensemble method, \textit{snap}. This is consistent with findings in [28, 52].

We also evaluate all methods under the effect of distributional (dataset) shift [50]. Here we mainly focus on the following two aspects: \textbf{sparsity change} and \textbf{domain shift}. For sparsity change within mid-density, we take models trained on 5\% sparsity and test on varying sparsity level from 5\% to 1\%. Results are shown in Figure 5 (c). Note that these plots reflect how each method performs on two axes in terms of uncertainty estimation (NLL) and depth prediction (MAE), where better methods should reside closer to the lower left corner. We see that performance of all methods degrade in a similar manner with deceasing sparsity, which is largely due to the sparse depth scaffolding approach we choose. However, \textit{bdbf} stands out with its 1\% result better than the 5\% results of its competitors. We refer the reader to our supplementary material for results on domain shift.

\textbf{Low-density Depth Completion.} In this setting, we train all methods with 500 sparse points, which is roughly 0.5\% sparsity given our image size. We also introduce a slight variation of our method \textit{bdbf(rgb)} which only uses sparse depths at the fitting stage (not as network input). Because at very low sparsity levels (e.g. 50 points), the scaffolding method we use for depth interpolation [66] struggles to recover the scene structure which impacts the performance of all \textit{rgbd} methods.

The top half of Table 3 shows the in-distribution test of all methods. Among the four \textit{rgbd} methods, \textit{bdbf} again outperforms the rest by a large margin. \textit{bdbf(rgb)}, despite not utilizing the rich information provided by the interpolated depths, performs on-par with the baselines. The real advantage of this approach is that it does not suffer from the artifacts caused by poor depth interpolation in the very low sparsity regime, which makes it sparsity-invariant. This claim is verified in Figure 6, which shows how each

<table>
<thead>
<tr>
<th>Table 2: Quantitative results of all methods trained and tested with 5% sparsity on VKITTI2 and NYU-V2.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
</tbody>
</table>

Table 3: Quantitative results of all methods trained and tested with 500 sparse depths and our proposed method tested with no sparse depths compared to a monocular depth prediction baseline (†). \textit{rgbd} under the input column indicates the basis network uses the sparse depths scaffolding approach from [66], whereas \textit{rgb} uses color image as basis network input only.

<table>
<thead>
<tr>
<th>Table 3: Quantitative results of all methods trained and tested with 500 sparse depths and our proposed method tested with no sparse depths compared to a monocular depth prediction baseline (†). rgbd under the input column indicates the basis network uses the sparse depths scaffolding approach from [66], whereas rgb uses color image as basis network input only.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgbd</td>
</tr>
<tr>
<td>rgb†</td>
</tr>
<tr>
<td>rgb†</td>
</tr>
</tbody>
</table>

Figure 6: Sparsity change plots of all methods trained with 500 sparse depths and test with various sparsity from 500 to 50. The small subplots show how the performance of each method changes with decreasing sparsity w.r.t. the performance of their in-distribution test (500). Shorter lines indicate better sparsity-invariance.
method’s performance deteriorates with decreasing sparsity. It is shown in the small subplots that bdbf(rgb) is able to maintain good performance even with as few as 50 points.

Finally, we test bdbf(rgb) with no sparse depths, which relies only on the shared prior to make a prediction. We ignore all rgbd methods because with nothing to interpolate the network outputs poor solutions. We thus only compare to another baseline log(rgb), which is trained for monocular depth prediction with NLL loss. Note that bdbf(rgb) and log(rgb) have exactly the same architecture (except for the last layer) and number of parameters. We see that bdbf(rgb) produces sharper depth than the baseline as shown in Figure 7. Quantitative results can be found in the last two rows of Table 3. The difference in performance of our method between two datasets is due to the distribution of the data: VKITTI2 contains mainly sequential driving videos, which gives a sharply peaked prior; whereas data from NYU-V2 are taken from a wide variety of scenes with different viewing angles, hence a less informative one. These results show that our learned depth bases and shared prior contain geometric information about the scene conditioned on the image and can be used under extreme conditions without catastrophic failure.

<table>
<thead>
<tr>
<th>Method</th>
<th>iRMSE</th>
<th>iMAE</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2D [42]</td>
<td>2.80</td>
<td>1.21</td>
<td>814.73</td>
<td>249.95</td>
</tr>
<tr>
<td>Gansbeke [19]</td>
<td>2.19</td>
<td>0.93</td>
<td>772.87</td>
<td>215.02</td>
</tr>
<tr>
<td>DepthNormal [67]</td>
<td>2.42</td>
<td>1.13</td>
<td>777.05</td>
<td>235.17</td>
</tr>
<tr>
<td>DeepLiDAR [53]</td>
<td>2.56</td>
<td>1.15</td>
<td>758.38</td>
<td>226.50</td>
</tr>
<tr>
<td>FuseNet [8]</td>
<td>2.34</td>
<td>1.14</td>
<td>752.88</td>
<td>221.19</td>
</tr>
<tr>
<td>CSPN++ [9]</td>
<td>2.07</td>
<td>0.90</td>
<td>743.69</td>
<td>209.28</td>
</tr>
<tr>
<td>NLSPN [51]</td>
<td>1.99</td>
<td>0.84</td>
<td>741.68</td>
<td>199.59</td>
</tr>
<tr>
<td>GuideNet [61]</td>
<td>2.25</td>
<td>0.99</td>
<td>736.24</td>
<td>218.83</td>
</tr>
<tr>
<td>bdbf (ours)</td>
<td>2.37</td>
<td>0.89</td>
<td>900.38</td>
<td>216.44</td>
</tr>
</tbody>
</table>

Table 4: Comparison with selected methods on the official KITTI depth completion test set.

Further Comparisons. While our focus is on evaluating the quality of our uncertainty estimation scheme, we also evaluate depth completion performance for completeness. We trained our method with a ResNet34 encoder [25] and applied it to the KITTI depth completion benchmark with results shown in Table 4. We compare our relatively simple Bayesian filtering scheme to SOTA methods that utilize either iterative refinement [9, 51] or sub-networks with extra constraints [67, 53]. Our method compares favorably on all measures except RMSE and we observe that this difference is due to a small number of mis-attributed pixels near depth discontinuities and the use of L1 loss only. This suggests that these methods could be further improved by predicting initial depth and uncertainty estimates with our module.

We also compare with pNCNN [13] as it is the only work that provides a quantitative evaluation of predicted uncertainties for depth completion. Unfortunately, they only evaluate using a single metric, AUSE, which we argue cannot completely capture the true quality of the uncertainty estimate. Results are shown in Table 5, note that pNCNN is un-guided and evaluation is done on the KITTI validation set, as groundtruth is required to compute uncertainty metrics.

5. Conclusions

In this paper, we extend Deep Basis Fitting for depth completion under a principled Bayesian framework that outputs uncertainty estimates alongside depth prediction. Compared to commonly used uncertainty estimation techniques, our integrated approach is able to produce better uncertainty estimates while being data- and compute-efficient. The benefit of being Bayesian is also demonstrated by the ability to handle very low-density sparse depths, a situation where the original DBF method struggles. Our work allows a depth completion network to be further integrated into robotics systems, where Bayesian sensor fusion is the dominant approach.

Acknowledgement

This work was supported in part by C-BRIC and DCIST, both sponsored by DARPA. We acknowledge the support of NVIDIA Corporation with the donation of the DGX Station used for this research and Novateur Research Solutions.
References


