Abstract

In this paper we consider the task of image-guided depth completion where our system must infer the depth at every pixel of an input image based on the image content and a sparse set of depth measurements. We propose a novel approach that builds upon the strengths of modern deep learning techniques and classical optimization algorithms and significantly improves performance. The proposed method replaces the final convolutional layer employed in most depth completion networks with a least squares fitting module which computes weights by fitting the implicit depth bases to the given sparse depth measurements. In addition, we show how our proposed method can be naturally extended to a multi-scale formulation for improved self-supervised training. We demonstrate through extensive experiments on various datasets that our approach achieves consistent improvements over state-of-the-art baseline methods with small computational overhead.

1. Introduction

Deep convolutional networks have proven to be effective tools for solving deep regression problems like depth prediction and depth completion [7]. Most networks proposed for this regression task share a common structure where the penultimate features are reduced to single channel by a final convolutional layer. This final convolutional output is then passed through a nonlinear function to map it onto the range of acceptable depth values.

This observation motivates the main contribution of this paper: Instead of using a fixed set of weights in the final layer, we perform a least squares fit from the penultimate features to the sparse depths to get a set of data-dependent weights. The rest of the network parameters are still shared across input data and learned using stochastic gradient descent. From a regression point of view, the network that produces the penultimate layer of features is an adaptive basis function [2] and we refer to the features before the final layer as depth bases. We argue that explicitly carrying out a regression from the depth bases to the sparse depths allows the network to learn a different representation that better enforce its predictions to be consistent with the measurements, which manifests as significant performance gain.

To this end, we first demonstrate how one could circumvent the nonlinearity from the depth activation function by solving a linear least squares problem with transformed target sparse depths. We then address the full robustified nonlinear least squares problem in order to deal with noisy measurements and outliers in real-world data. Finally, to make our module truly a drop-in replacement for the final convolutional layer, we show how to adapt it to output predictions at multiple scales with progressively increased detail, which is a feature required by self-supervised training schemes.

2. Related Work

2.1. Depth Estimation

Supervised Learning. Estimating dense depths from a single image is a fundamentally ill-posed problem. Recent learning-based approaches try to solve this by leveraging the predictive power of deep convolutional neural networks (CNN) with strong regularization [7, 24, 9]. These works require dense or semi-dense ground truth annotations, which are costly to obtain in large quantities in practice. Synthetic data [33, 10, 35], on the other hand, can be generated more easily from current graphics systems. However, it is non-trivial to generate synthetic data that closely matches the appearance and structure of the real-world, thus the resulting networks may require an extra step of fine-tuning or domain adaptation [1].

Self-Supervised Learning. When ground truth depths are not available, one could instead seek to use view synthesis as a supervisory signal [39]. This so-called self-supervised training has gained popularity in recent years [27, 31, 44]. The network still takes a single image as input and predicts depths, but the loss is computed on a set of images. This is achieved by warping pixels from a set of source images to the target image using the predicted depths, along with estimated camera poses and camera intrinsics. Under various constancy assumptions [30], errors between target and syn-
Another version of self-supervision utilizes synchronized stereo pairs [12] during training. In this setting, the network predicts the depth for the left view and uses the known focal length and baseline to reconstruct the right view, and vice versa. A more complex form utilizes the motion in monocular videos [52]. In these approaches the network also needs to predict the transformation between frames. The biggest challenge faced by monocular self-supervision is handling moving objects. Many authors try to address this issue by predicting an explainability mask [52], motion segmentation [43], and joint optical-flow estimation [50]. We refer readers to [15] for a more detailed review.

2.2. Depth Completion

Depth completion is an extension to the depth estimation task where sparse depths are available as input. Uhrig et al. [42] propose a sparse convolution layer that explicitly handles missing data, which renders it invariant to different levels of sparsity. Ma et al. [26] adopt an early-fusion strategy to combine color and sparse depths inputs in a self-supervised training framework. On the other hand, Jaritz et al. [22] and Shivakumar et al. [37] advocate a late-fusion approach to transform both inputs into a common feature space. Zhang et al. [51] and Qiu et al. [32] estimate surface normals as a secondary task to help densify the sparse depths. Irman et al. [20] identify the cause of artifacts caused by convolution on sparse data and propose a novel scheme, Depth Coefficients, to address this problem. Eldesokey et al. [8] and Gansbeke [11] propose to use a confidence mask to handle noise and uncertainty in sparse data. Yang et al. [49] infer the posterior distribution of depth given an image and sparse depths by a Conditional Prior Network. While most of the above works deal with data from LiDARs or depth cameras, Wong et al. [48] design a system that works with very sparse data from a visual-inertial odometry system. Weeraskera et al. [47] attach a fully-connected Conditional Random Field to the output of a depth prediction network, which can also handle any input sparsity pattern.

Cheng et al. [4] propose a convolutional spatial propagation network that learns the affinity matrix to complete sparse depths. This is similar to a diffusion process and uses several iterations to update the depth map. Another iterative approach is described by Wang et al. [45], in which they design a module that can be integrated into many existing methods to improve performance of a pre-trained network without re-training. This is done by iteratively updating the intermediate feature map to make the model output consistent with the given sparse depths. Like [45], our approach could be readily integrated into many of the previously proposed depth completion networks.

In other related work Tang et al. [40], propose to parameterize depth map with a set of basis depth maps and optimize weights to minimize a feature-metric distance. In contrast, our bases are multi-scale by construction and are fit directly to the sparse depths.

3. Method

In this section, we describe our proposed method for the task of monocular image-guided depth completion1. Given an image \(X\) and a sparse depth map \(S\), we wish to predict a dense depth image \(D'\) from a depth estimation function \(f\) that minimizes some loss function \(L\) with respect to the ground truth depth \(D\). Typically, \(X\) is a color image, \(S\) the sparse depth map where invalid pixels are encoded by 0, and \(f\) a fully convolutional neural network whose parameters are denoted by \(\theta\). When ground-truth depth \(D\) is available, the learning problem is to determine \(\theta^*\) according to

\[
\theta^* = \arg\min_\theta L(f(X, S; \theta), D)
\]

For supervised training we choose \(L\) to be the L1 norm on depth and for self-supervised training we use a combination of L1+SSIM on the intensity values [46] coupled with an edge-aware smoothness term [15].

3.1. Linear Least-Squares Fitting (LSF) Module

Existing depth prediction networks usually employ a final convolutional layer to convert an \(M\)-channel set of basis features, \(B\), to a single-channel result, \(L\), which is sometimes referred to as the logits layer. The inputs to this final layer are allowed to range freely between \(-\infty\) and \(+\infty\) while the logit outputs are mapped to positive depth values by a nonlinear activation function, \(g\). Following common practice in the depth completion literature [15] we choose \(g\) as follows:

\[
g(x) = a/\sigma(x) = a(1 + e^{-x})
\]

1From now on we will refer to this task as depth completion.
where \( a \) is a scaling factor that controls the minimum depth and \( \sigma(\cdot) \) the sigmoid function. In this work, we set \( a = 1 \).

For simplicity we assume that the final convolution filter that maps the basis features, \( B \), onto the logits, \( L \), has a kernel size of \( 1 \times 1 \) with bias \( w_0 \), but one could easily extend our result to arbitrary kernel size. \( L \) is, therefore, an affine combination of channels in \( B \) and the predicted depth at pixel \( i \) is

\[
D'[i] = g(L[i]) = g \left( \sum_{j=0}^{M} w_j \cdot B_j[i] \right) = g(w^\top b_i) \tag{3}
\]

where \( w = (w_0, \ldots, w_M)^\top \) represents the combined filter weights and bias, and \( b_i \) the basis (feature) vector at pixel \( i \) with \( B_0[i] = 1 \), and \([\cdot]\) the pixel index operator. To simplify notations, we use lower case letters, e.g. \( b_i = B[i] \), to denote values at a particular pixel location. The weights \( w \) are updated via back-propagation [25] using stochastic gradient descent [3]. Once learned they are typically fixed at inference time.

When enough sparse depth measurements are available the weights \( w \) can instead be directly computed from data. Specifically, our weights are obtained through a least squares fit from the bases \( B \) to the sparse depths \( S \) at valid pixels, which can then be used in place of the final convolutional layer. An overview of our proposed method is shown in Figure 1.

The objective function we wish to minimize for the least squares problem is

\[
\min_w \frac{1}{2} \sum_{i=1}^{N} r^2 (w, b_i, s_i) \tag{4}
\]

with residual function

\[
r(w, b_i, s_i) = g \left( \sum_{j=0}^{M} w_j b_{ij} \right) - s_i = g(w^\top b_i) - s_i \tag{5}
\]

where \( s_i \) denotes an individual sparse depth measurement, \( N \) is the number of valid pixels in \( S \), \( M \) the number of channels in \( B \), and \( g(\cdot) \) a nonlinear activation function.

The residual function \( r(\cdot) \) is obviously nonlinear w.r.t. the weights \( w \) due to the nonlinearity in \( g(\cdot) \). A simple workaround is to transform the target variable \( s \) by \( g^{-1}(\cdot) \) to arrive at a new linear residual function

\[
r'(w, b_i, s_i) = w^\top b_i - g^{-1}(s_i) = w^\top b_i - t_i \tag{6}
\]

We can then rewrite the new objective function (4) in matrix form to obtain a linear least squares problem

\[
\min_w \frac{1}{2} \| Bw - t \|^2 \tag{7}
\]

where \( B \) denotes the \( N \times (M+1) \) matrix of stacked features \( b_i \) at valid pixel locations and \( t \) the corresponding transformed sparse depths vector. The solution to (7) is the well-known Moore–Penrose pseudo-inverse which can be further regularized with parameter \( \lambda \) [2].

\[
w^* = (\lambda I + B^\top B)^{-1} B^\top t \tag{8}
\]

Notice here that our weights \( w^* \) are calculated deterministically as a function of the bases \( B \) and the sparse depth \( S \), while the original convolution filter is independent of both. In practice, this problem is usually solved via LU or Cholesky decomposition both of which are differentiable [28]. Thus, the entire training process including our LSF module is differentiable which means that it can be trained in an end-to-end manner. This is an important point since we have found that retraining the network with this fitting module produces much better results than simply adding the fitting procedure to a pretrained network without retraining. Effectively the retraining allows the network to make best use of the new adaptive fitting layer.

### 3.2. Robustified Nonlinear Fitting

The linear LSF module is readily usable as a replacement for the final convolution layer in many depth prediction networks. One problem remains to be addressed, which is the fact that the original objective function in Equation 5 is nonlinear w.r.t. the weights \( w \). Although applying the inverse transformation \( g^{-1}(\cdot) \) to the sparse depths is a simple yet effective solution, we show that performing a full robustified nonlinear least squares fitting provides further performance improvements and outlier rejection at the cost of extra computation time.

Real-world data often contain noise and outliers that are hard to model or eliminate. Cheng et al. [5] point out that there exist many different types of noise in LiDAR data from the well-known KITTI [13] dataset. They propose a novel feedback loop that utilizes stereo matching from the network to clean erroneous data points in the sparse depths. Gansbeke et al. [11] let the network predict a confidence map to weight information from different input branches. To handle these cases, we employ M-estimators [18], which fit well within our least squares framework.

Recall the objective function in equation (4), taking the derivative with respect to \( w \), setting it to zero and ignoring higher-order terms yields the following linear equation (Gauss-Newton approximation)

\[
J^\top J \Delta w = -J^\top r \tag{9}
\]

where \( J \) is the Jacobian matrix that is formed by stacking Jacobians \( J_s(w, b_i, s_i) = \partial r(w, b_i, s_i)/\partial w \), and \( r \) is the residual vector formed by stacking \( r_i(w, b_i, s_i) \). Following standard practice in Triggs et al. [41], we minimize the
iteratively calculate \( \Delta \) which is also known as an \( \text{ents} \) [17], we adopt the fixed-iteration approach used in [40], but to alleviate the problem of vanishing or exploding gradients with \( w \) [19] and \( \rho \) proportional to the noise in each measurement, which we waste of parameters.

Lower resolution depth maps are used in training/inference time and reduced memory consumption, which is often desirable in robotic systems with limited computational resources. As discussed in earlier Section 3.1, solving a linear system like equation (10) via Cholesky decomposition is differentiable, thus optimizing this non-linear objective function by performing a fixed number of Gauss-Newton steps maintains the differentiability of the entire system.

3.3. Multi-scale Prediction for Self-supervision

Self-supervised training formulates the learning problem as novel view synthesis, where the network predicted depth is used to synthesize a target image from other viewpoints. To overcome the gradient locality problem of the bi-linear sampler [21] during image warping, previous works [14, 52] adopt a multi-scale prediction and image reconstruction scheme by predicting a depth map at each decoder layer’s resolution. According to Godard et al. [15], this has the side effect of creating artifacts in large texture-less regions in the lower resolution depth maps due to ambiguities in photometric errors. They later improved upon this multi-scale formulation by upsampling all the lower resolution depth maps to the input image resolution.

This technique greatly reduces various artifacts in the final depth prediction, but it still has one undesired property, namely, depth maps predicted at each scale are largely independent. Lower resolution depth maps are used in training phase, but are discarded during inference, resulting in a waste of parameters.

Theoretically, one should repeat this until convergence, but to alleviate the problem of vanishing or exploding gradients [17], we adopt the fixed-iteration approach used in [40], which is also known as an incomplete optimization [6]. Despite its limitations, it has the advantage of having a fixed training/inference time and reduced memory consumption, which is often desirable in robotic systems with limited computational resources. As discussed in earlier Section 3.1, solving a linear system like equation (10) via Cholesky decomposition is differentiable, thus optimizing this non-linear objective function by performing a fixed number of Gauss-Newton steps maintains the differentiability of the entire system.

3.3. Multi-scale Prediction for Self-supervision

Self-supervised training formulates the learning problem as novel view synthesis, where the network predicted depth is used to synthesize a target image from other viewpoints. To overcome the gradient locality problem of the bi-linear sampler [21] during image warping, previous works [14, 52] adopt a multi-scale prediction and image reconstruction scheme by predicting a depth map at each decoder layer’s resolution. According to Godard et al. [15], this has the side effect of creating artifacts in large texture-less regions in the lower resolution depth maps due to ambiguities in photometric errors. They later improved upon this multi-scale formulation by upsampling all the lower resolution depth maps to the input image resolution.

This technique greatly reduces various artifacts in the final depth prediction, but it still has one undesired property, namely, depth maps predicted at each scale are largely independent. Lower resolution depth maps are used in training phase, but are discarded during inference, resulting in a waste of parameters.

Theoretically, one should repeat this until convergence, but to alleviate the problem of vanishing or exploding gradients [17], we adopt the fixed-iteration approach used in [40], which is also known as an incomplete optimization [6]. Despite its limitations, it has the advantage of having a fixed training/inference time and reduced memory consumption, which is often desirable in robotic systems with limited computational resources. As discussed in earlier Section 3.1, solving a linear system like equation (10) via Cholesky decomposition is differentiable, thus optimizing this non-linear objective function by performing a fixed number of Gauss-Newton steps maintains the differentiability of the entire system.

4. Experiments

4.1. Implementation Details

Network Architecture. All networks and training are implemented in PyTorch To investigate the effectiveness of the proposed LSF module, we adopt the network used in Ma et al. [26] as our main baseline. The network is a symmetric encoder-decoder [34] with skip connections. We make the following modifications for better training: 1) transposed
Table 1: A summary of all datasets used. Cap indicates the maximum depth being used for sampling sparse depths as well as in computing various error metrics. Resolution is the image resolution that we use in our experiments, which we downsample from the original one if necessary.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Resolution</th>
<th># Train</th>
<th># Val</th>
<th>Cap [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>KITTI [13, 42]</td>
<td>375 × 1242</td>
<td>38412</td>
<td>3347</td>
<td>80</td>
</tr>
<tr>
<td>V-KITTI [10]</td>
<td>188 × 621</td>
<td>5156</td>
<td>837</td>
<td>130</td>
</tr>
<tr>
<td>Synthia [35]</td>
<td>304 × 512</td>
<td>3634</td>
<td>901</td>
<td>130</td>
</tr>
<tr>
<td>NYU-V2 [38]</td>
<td>480 × 640</td>
<td>1086</td>
<td>363</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: A summary of all datasets used. Cap indicates the maximum depth being used for sampling sparse depths as well as in computing various error metrics. Resolution is the image resolution that we use in our experiments, which we downsample from the original one if necessary.

4.2. Datasets

A summary of all datasets we evaluate on is shown in Table 1.

KITTI Depth Completion. We evaluate on the newly introduced KITTI depth completion dataset [42] and follow the official training/validation split. The ground truth depth is generated by merging several consecutive LiDAR scans around a given frame and refined using a stereo matching algorithm. The sparse depth map is generated by projecting LiDAR measurements onto the closest image, which occupies on average 4% of the image resolution. We use all categories from the KITTI raw dataset [13] except for Person as it contains mostly static scenes with moving objects, which is not suitable for self-supervised training.

Virtual KITTI. The Virtual KITTI (V-KITTI) dataset is a synthetic video dataset [10], which contains 50 monocular videos generated with various simulated lighting and weather conditions with dense ground truth annotations. We adopt an out-of-distribution testing scheme for this dataset. Specifically, we use sequences 1, 2, 6, 18 with variations clone, morning, overcast and sunset for training, and sequence 20 with variation clone for validation. Thus the testing sequence is never seen during training. The sparse depths are generated by randomly sampling pixels that have a depth value less than 130 meters. We intentionally increase the depth cap to 130 meters for all synthetic datasets since recent LiDAR units can easily achieve this range.

Synthia. Synthia [35] is another synthetic dataset in urban settings with dense ground truth. We use the SYNTHIA-Seqs version which simulates four video sequences acquired from a virtual car across different seasons. Following the training protocol in V-KITTI, we use sequences 1,2,5,6 for training and sequence 4 for validation, all under the summer variation. We include this dataset because it has simulated stereo images, which serves as a complement to the monocular only V-KITTI. Again ground truth and sparse depths are capped at 130 meters.

NYU Depth V2. In addition to all the outdoor datasets, we also validate our approach on NYU Depth V2 (NYU-V2) [38], which is an indoor dataset. We use the 1449 densely labeled pairs of aligned RGB and depth images instead of the full dataset which is comprised of raw image and depth data as provided by the Kinect sensor. The dataset is split into approximately 75% training and 25% validation. We use the same strategy as above for sampling sparse depths but put no cap on the maximum depth.

4.3. Results

We evaluate performance using standard metrics in the depth estimation literature. Note that for accuracy ($\delta$ threshold) [7] we only report $\delta_1 < 1.25$, due to space limitations and the fact that the $\delta_2$ and $\delta_3$ are typically 99% for our experiments and thus provide limited insights. Following [45], we group results based on input modalities, where rgb denotes a network that only takes a color image as input. In contrast rgbd indicates a network that takes both the color image and the sparse depths as inputs.

Performance of Linear Fitting. Table 2 shows quantitative comparisons between our proposed linear LSF module from Section 3.1 and the baseline under supervised training. We see consistent improvements of our linear LSF module
Table 2: Quantitative results of supervised training on various datasets. **conv** denotes the baseline network, **pnp** denotes running the PnP [45] module on the trained **conv** network without re-training, **lsf** - indicates adding a linear LSF module to the pre-trained **conv** network without re-training for 5 iterations, and **lsf2** is our linear fitting module (re-trained). Percentage values listed under the Sparse column indicates sparse depths percentage of image resolution. Best results in each category are in **bold**.

<table>
<thead>
<tr>
<th>Input</th>
<th>Method</th>
<th>Sparse</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td><strong>conv</strong></td>
<td>-</td>
<td>0.6244</td>
<td>0.8693</td>
<td>58.44</td>
<td>6.9998</td>
<td>14.653</td>
<td>66.43</td>
<td>2.3911</td>
<td>6.3915</td>
<td>76.09</td>
<td>1.8915</td>
<td>4.1164</td>
<td>86.24</td>
</tr>
<tr>
<td>rgb</td>
<td>pnp</td>
<td>0.2%</td>
<td>0.5517</td>
<td>0.7976</td>
<td>64.23</td>
<td>6.4701</td>
<td>13.990</td>
<td>70.18</td>
<td>2.1716</td>
<td>6.0804</td>
<td>81.37</td>
<td>1.6851</td>
<td>3.8019</td>
<td>88.67</td>
</tr>
<tr>
<td>rgb</td>
<td>lsf</td>
<td>0.2%</td>
<td>0.4081</td>
<td>0.6124</td>
<td>77.86</td>
<td>5.8379</td>
<td>12.712</td>
<td>71.62</td>
<td>2.4089</td>
<td>6.2520</td>
<td>78.49</td>
<td>1.7033</td>
<td>3.5986</td>
<td>91.80</td>
</tr>
<tr>
<td>rgb</td>
<td>lsf2+</td>
<td>0.2%</td>
<td><strong>0.1826</strong></td>
<td><strong>0.3165</strong></td>
<td><strong>96.11</strong></td>
<td><strong>4.5122</strong></td>
<td><strong>9.7933</strong></td>
<td><strong>77.18</strong></td>
<td><strong>2.0104</strong></td>
<td><strong>5.6285</strong></td>
<td><strong>84.37</strong></td>
<td><strong>0.7716</strong></td>
<td><strong>2.0808</strong></td>
<td><strong>97.69</strong></td>
</tr>
</tbody>
</table>

Table 3: Quantitative results of supervised training with noisy data and outliers. For all datasets except KITTI, noise is additive Gaussian with standard deviation of 0.05m. We randomly sample 30% of sparse depths to be outliers. **conv** denotes the baseline network, **pnp** denotes running the PnP [45] module on the trained **conv** network without re-training, **lsf** is our linear fitting module, **lsf2** is our nonlinear fitting module with 2 iterations, and **lsf2+** is **lsf2** with robust norm (Huber). Best results in each category are in **bold**.

<table>
<thead>
<tr>
<th>Input</th>
<th>Method</th>
<th>Sparse</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td><strong>pnp</strong></td>
<td>0.2%</td>
<td>0.5587</td>
<td>0.8019</td>
<td>63.66</td>
<td>6.3099</td>
<td>14.018</td>
<td>69.86</td>
<td>2.2044</td>
<td>6.0268</td>
<td>80.89</td>
<td>1.0571</td>
<td>3.8019</td>
<td>88.67</td>
</tr>
<tr>
<td>rgb</td>
<td>lsf</td>
<td>0.2%</td>
<td>0.2439</td>
<td>0.3815</td>
<td>92.93</td>
<td>5.2670</td>
<td>10.696</td>
<td>65.00</td>
<td>2.2179</td>
<td>5.9136</td>
<td>78.34</td>
<td>0.7716</td>
<td>2.0808</td>
<td>97.69</td>
</tr>
<tr>
<td>rgb</td>
<td>lsf2</td>
<td>0.2%</td>
<td>0.2304</td>
<td>0.3519</td>
<td>92.70</td>
<td>6.0025</td>
<td>10.768</td>
<td>51.01</td>
<td>3.2160</td>
<td>7.2096</td>
<td>59.68</td>
<td>1.0111</td>
<td>2.4547</td>
<td>95.88</td>
</tr>
<tr>
<td>rgb</td>
<td>lsf2+</td>
<td>0.2%</td>
<td><strong>0.1880</strong></td>
<td><strong>0.3217</strong></td>
<td><strong>94.97</strong></td>
<td><strong>4.6786</strong></td>
<td><strong>9.7402</strong></td>
<td><strong>70.16</strong></td>
<td><strong>2.1032</strong></td>
<td><strong>5.7685</strong></td>
<td><strong>79.00</strong></td>
<td><strong>0.6775</strong></td>
<td><strong>1.9651</strong></td>
<td><strong>98.28</strong></td>
</tr>
</tbody>
</table>

Table 4: Quantitative results of self-supervised training on various datasets. The densely labeled NYU-V2 is random and monocular, thus is excluded from this experiment. Here **conv-ms** is the baseline multi-scale prediction, **lsf** is our proposed method with linear fitting and multi-scale basis. Best results in each category are in **bold**.

<table>
<thead>
<tr>
<th>Input</th>
<th>Method</th>
<th>Sparse</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
<th>MAE</th>
<th>RMSE</th>
<th>$\delta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td><strong>conv-ms</strong></td>
<td>4%</td>
<td>2.9904</td>
<td>7.4517</td>
<td>88.87</td>
<td>3.0191</td>
<td>9.1076</td>
<td>66.43</td>
<td>1.3498</td>
<td>5.8643</td>
<td>92.78</td>
<td>0.6295</td>
<td>2.0950</td>
<td>99.00</td>
</tr>
<tr>
<td>rgb</td>
<td><strong>lsf</strong></td>
<td>4%</td>
<td>2.3804</td>
<td>6.7326</td>
<td>93.76</td>
<td>1.4564</td>
<td>4.6260</td>
<td>91.76</td>
<td>0.8619</td>
<td>3.9523</td>
<td>96.30</td>
<td>0.5820</td>
<td>1.7370</td>
<td>98.79</td>
</tr>
</tbody>
</table>

Note that for rgb input only, the baseline doesn’t use any sparse depth information at all. Thus the large improvement achieved by our fitting method using depth measurements for only 0.2% of the pixels is quite significant. For the rgbd case, although the sparse depth map is already used as the input to the baseline network, adding our fitting module better constrains the final prediction to be in accordance with the measurements and improves the baseline network. Since we use the L1 norm as our loss function, the improvement in MAE is bigger than that in RMSE. Examples of depth prediction are shown in Figure 3 for qualitative comparisons.
We also perform experiments in which we take a pre-trained baseline method, replace the final convolutional layer with our LSF module and evaluate without re-training. This is denoted by \textit{lsf-}. Results show that re-training a baseline network with the LSF module allows it to achieve significantly better performance.

Additionally, we compare with PnP [45], which is a similar method that can be used on many existing networks to improve performance (see Table 2 and 3). The main difference is that PnP does not require re-training. We use the author’s official implementation on our baseline network by updating the output of the encoder and run for 5 iterations with update rate 0.01 as suggested in the paper. We found that although PnP has the advantage no re-training, it takes much longer to run, uses a large amount of memory and yields a smaller improvement compared to ours. Comparisons of runtime are provided in the supplementary material.

Table 5 compares our results to those achieved with CSPN[4]. The numbers for the CSPN system are taken directly taken from their paper and the official KITTI depth completion benchmark. For NYU-V2 we use the same data split they used and sample 500 sparse depths. These results show the improvement afforded by our method.

**Dealing with Noise and Outliers.** To verify the effectiveness of our proposed robustified nonlinear fitting module, we inject additive Gaussian noise with a standard deviation

![Figure 3: Qualitative results of supervised learning on various datasets. Sparse depths are dilated for visualization purpose (4% of image resolution). Artifacts in the upper part of depth prediction from outdoor datasets are due to lack of supervision.](image)

![Figure 4: When using a robust norm, outliers from the input sparse depths can be identified. For KITTI dataset, these outliers usually occur at object boundaries, which we highlight a few in rectangles. Best view when zoomed in.](image)
of 0.05 meters to sparse depths from NYU-V2, V-KITTI, and Synthia. We then randomly select 30% of the available sparse depths to be outliers and set them to random values drawn uniformly from a range between $0.5 \times 1.5 \times$ of the true depth value. We left KITTI untouched as it already contains noise and outliers [5]. All nonlinear variants of LSF runs for 2 iterations, which we empirically found to achieve a good balance between performance and efficiency. We refer the reader to our supplementary material for further discussion on the number of iterations. We then train various models with different configurations using the corrupted data, which are also grouped by input modalities. Quantitative results are shown in Table 3.

For the rgb case, we ignore the baseline conv as it doesn’t use sparse depths and is, therefore, unaffected by noise. We again see consistent improvements in all metrics across all datasets. Notice that for our nonlinear fitting without Huber loss (lsf2), we get worse numbers on some datasets compared to our linear variant (lsf). This is because least squares fitting is sensitive to outliers without a robust norm. There are also some models in the rgb case where the robustified version (lsf2+) doesn’t outperform the linear and nonlinear ones. We hypothesize this to be caused by using the corrupted sparse depths as network input which degrades the networks performance early on. We show in Figure 4 that our proposed method is able to identify outliers in the sparse depths and downplay them during fitting.

These results can also be cross-compared with those in Table 2, which are all trained on clean data. Clearly, models trained with clean data outperform those trained with corrupted ones with the same configuration. But ours with nonlinear fitting and Huber loss (lsf2+) can sometimes reach similar performance to those trained with clean data even when significant noise and outliers are present.

**Self-supervised Training with Multi-scale Prediction.** Table 4 shows quantitative comparisons between our linear LSF module with multi-scale basis and the baseline network under both monocular and stereo self-supervised training. In this case, the baseline network has more parameters because it needs to predict depths at different scales independently. We again witness consistent improvement in all metrics across all datasets except for $\delta_1$ in KITTI. Qualitative results are shown in Figure 5. For all self-supervised training, we use the same hyper-parameters on photometric and smoothness loss as in [15], where $\lambda_p = 1.0$ and $\lambda_y = 0.001$. Note in monocular training, we use the ground truth poses directly, as opposed to having a dedicated pose network.

**5. Conclusions**

In this paper we propose a novel approach to the depth completion problem that augments deep convolutional networks with a least squares fitting procedure. This method allows us to combine some of the best features of modern deep networks and classical regression algorithms. This scheme could be applied to a number of proposed depth completion networks or other regression problems to improve performance. Our proposed module is differentiable which means the modified networks can still be trained from end to end. This is important because retraining the networks allows them to make better use of the new fitting layer and allows them to produce better depth bases from the input data. We then describe how a linear least squares fitting scheme could be extended to incorporate robust estimation to improve resilience to noise and outliers which are common in real world data. We also show the method can be employed in self-supervised settings where no ground truth is available. We validate our fitting module on a state-of-the-art depth completion network with various input modalities, training frameworks, and datasets. One limitation of our approach is that it is unable to handle extremely sparse points, which creates an underdetermined linear system and can only be solved by adding strong regularization. In future work, we propose to handle this case by adopting a full bayesian approach.

**Acknowledgement**

We gratefully acknowledge the support of Novateur Research Solutions, and NVIDIA through the NVAIL grant.
References


