1. Network Architecture

**Encoder:** Following the recent works [3, 7], we use a ResNet101 [6] as the encoder for the image. Each ResNet block consists of series of convolution operations with stride of 2 and pooling operations. The receptive field of the convolution is increased by decreasing resolution of the feature maps. This helps to capture more contextual information while compromising the feature map resolution. The final size of the feature map is usually 1/32 of the input image. The original ResNet is designed for the image classification task. To utilize it for a per-pixel prediction task, we remove the last 3 layers, i.e. pooling layer, fully-connected layer and the softmax layer. The ResNet encoder can be divided into 4 different blocks. Each block generates feature maps of different resolution (scales). These feature maps from different scales can be used as skip connections, i.e. fused with decoder outputs to integrate different level of semantic information. The output of the last encoder block is fetched to both decoder heads. Both decoder heads also receive the skip connection information.

**Decoder:** We base our decoder on [9] following [10]. We replace all ReLU operations with ELU [1] nonlinearities. The decoder is assembled from three modules: 1) **Feature fusion modules:** For each of these modules, residual convolution block is used to transform the skip connection feature map from the ResNet encoder. The output of the residual convolution block is fused with output of last feature fusion block using summation operation. Finally, the feature maps are upsampled to match the resolution of next layers input. 2) **Residual convolution modules:** This module is a series of two units of ELU and 3 × 3 convolution operations to merge the output of a previous decoder feature map output with a previous feature fusion module output 3) **Adaptive output module:** This is applied at the last stage to get the final output. It consist of two 3 × 3 convolution operation followed by up-sampling.

**Plane coefficient decoder:** The last layer of this decoder head is modified to output 4-channels for each planar coefficient instead of single channel depth.

**Offset Vector field decoder:** The last layer of this decoder head is modified to output 3-channels, i.e. two channels for the offset vector field and one for the confidence. The offset vector field is restricted by tanh layers and the confidence is generated through a sigmoid layer.

**Plane coefficient guidance:** This module is loosely based on [7]. The output of each decoder block is passed through the Plane coefficient guidance module to generate 4 channels of plane coefficients. The output size of the guidance module is up-sampled to match the input size of last decoder layer. At the end, these plane coefficients from each scale are converted into depth. All these depth maps are concatenated with feature map of the previous decoder layer passed to the last decoder layer.
2. Additional Results

KITTI Benchmark [5]: In this section we present the results of KITTI Benchmark server evaluation. Note that we train our model only on the KITTI Eigen split [2] training data. It can be seen in Table 1 that our results are on a par with SOTA methods and superior than the baseline. However, [7] performs better on this test set. In comparison with [11], we have a better absolute relative error and our performance is comparable to [11] in all other metrics. The drop in overall performance is expected considering the design of our method. Our method is specially designed to identify planar regions in the scene, to improve the depth quality. So, as the depth of the scene increases, the projections of distant parts of the scene get smaller. This causes difficulties in predicting offset vector field in these regions. We have already seen that our method produced the SOTA results on the Garg split [4], in which the maximum depth value is 50m. Due to the aforementioned reason, when tested on Eigen split [2] with max depth of 80m, we observe degradation in the performance. The KITTI Benchmark extends beyond that with 80m+ distances, thus affecting our results due to similar reasons.

Table 1: Results of KITTI Evaluation Server.

<table>
<thead>
<tr>
<th>Method</th>
<th>SIlog</th>
<th>sqErrorRel</th>
<th>absErrorRel</th>
<th>iRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official Baseline</td>
<td>18.19</td>
<td>7.32</td>
<td>14.24</td>
<td>18.50</td>
</tr>
<tr>
<td>VNL [11]</td>
<td>12.65</td>
<td>2.46</td>
<td>10.15</td>
<td>13.02</td>
</tr>
<tr>
<td>Ours</td>
<td>12.82</td>
<td>2.53</td>
<td>9.92</td>
<td>13.71</td>
</tr>
</tbody>
</table>

Qualitative Results: Here, we present additional qualitative results on both KITTI [5] and NYU Depth-v2 [8] datasets. We start with some examples from the KITTI dataset. We present some of the best cases along with the failure cases on this dataset. Additionally, we provide visualizations of the predicted depth maps and offset vector fields on NYU Depth-v2. Finally, we use the predicted depth maps to reconstruct the scenes and demonstrate quality in 3D. We observe that the predicted depth maps produce 3D reconstructions which are consistent with ground-truth point clouds and preserve the structure of the scene.

References


Figure 2: Visualization of predictions on KITTI dataset.

Figure 3: Visualization of some failure cases on KITTI dataset.
Figure 4: Visualization of predictions on NYU Depth-v2.
Figure 5: Additional reconstruction examples from NYU Depth-v2.