# A CNN-based Depth Estimation Approach with Multi-scale Sub-pixel Convolutions and A Smoothness Constraint

Shiyu Zhao[1], Lin Zhang[1,*[0000−0002−4360−5523]], Ying Shen[1], and Yongning Zhu[2]

[1] School of Software Engineering, Tongji University, Shanghai 201804, China
{1731558,cslinzhang,yingshen}@tongji.edu.cn
[2] College of Arts and Media, Tongji University, Shanghai 201804, China
yzhu@tongji.edu.cn

**Abstract.** Depth estimation from a single image is of paramount importance in various vision tasks, such as obstacle detection, robot navigation, 3D reconstruction, *etc.* However, how to get an accurate depth map with clear details and a fine resolution remains an unresolved issue. As an attempt to solve this problem, we propose a novel CNN-based approach, namely $MSCN_{NS}$, which involves multi-scale sub-pixel convolutions and a neighborhood smoothness constraint. Specifically, $MSCN_{NS}$ makes use of sub-pixel convolutions which fuse multi-scale features from different branches of the network to retrieve a high resolution depth map with fine details of the scene. Furthermore, $MSCN_{NS}$ incorporates a neighborhood smoothness regularization term to make sure that spatially closer pixels with similar features would have close depth values. The effectiveness and efficiency of $MSCN_{NS}$ have been corroborated through extensive experiments conducted on benchmark datasets.

**Keywords:** Monocular depth estimation · Multi-scale feature fusion · Sub-pixel convolution · Neighborhood smoothness.

## 1 Introduction

Accurate depth information is vital to many computer vision tasks, such as scene understanding [5, 20, 3], 3D reconstruction [33], obstacle detection [28], *etc.* However, collecting depth is expensive or even impossible in some scenarios and in those cases, depth estimation is required. Generally, stereo vision approaches [7, 32, 24] are good solutions for this task. However, they require binocular images from two cameras and are very time consuming to get an accurate disparity map. Therefore, for data consisting of only monocular images, how to predict depth from a single still image becomes profoundly important. However, it is a very challenging task since one captured image may correspond to numerous real world scenes [2] and there are no reliable depth cues available, *e.g.* stereo correspondences or motions [16].

To handle such a problem, various solutions have been proposed in the literature. Primary methods [15, 22, 23] in this field usually formulated depth estimation as a markov random field (MRF) learning problem and resorted to hand-crafted features, such as SIFT, GIST, PHOG, *etc.* Later, data-driven approaches [9, 11] were explored. Those approaches made use of hand-crafted features to retrieve the most similar candidates in the training set for a given query image. And then, the corresponding depth candidates were warped and fused to produce the final prediction. However, all these methods were usually designed for specific conditions and thus could only achieve reluctantly acceptable results.

With the emergence and popularity of CNNs (Convolutional Neural Networks), recently, researchers have begun exploring CNNs in the context of depth estimation and preliminary better results in terms of both the efficiency and the accuracy have been achieved. Inspired by the great success already achieved along this direction, in this paper, we focus on how to further explore deep models for solving the depth estimation problem and propose a CNN-based approach with multi-scale sub-pixel convolutions and a neighborhood smoothness constraint, namely $MSCN_{NS}$ (Multi-scale Sub-pixel Convolutional Network with a Neighborhood Smoothness constraint). In our approach, we use multi-scale features from different branches of our network to get fine details in the prediction and further improve our model with the neighborhood smoothness constraint as a regularization term during the training phase.

The remainder of this paper is organized as follows. We first introduce the related work and our contributions in Section 2 and then present the proposed method $MSCN_{NS}$ in Section 3. After that, the experimental results and analysis are elaborated in Section 4. Finally, we conclude the paper in Section 5.

## 2 Related Work & Our Contributions

### 2.1 Related Work

In this paper, we focus on how to better explore deep models for solving the problem of depth estimation from monocular images. Some representative methods closely related to our study are briefly reviewed here.

The first depth estimation model exploiting CNN was proposed by Eigen *et al.* [2]. In Eigen *et al.*'s model, there were two paths, the coarse-scale one and the fine-scale one, mapping the input image to the target prediction. The coarse-scale path outputted a coarse depth map and the fine-scale path refined the output with more details of the scene. In their later work [1], Eigen and Fergus extended their model [2] using more paths to solve multiple tasks, including depth estimation, semantic segmentation and surface normal prediction. In [19], inspired by Eigen *et al.*'s work [2, 1], Mousavian *et al.* followed such a multi-path network to predict the semantic label and the depth value of each pixel jointly with shared representations. In Li *et al.*'s work [14], authors considered local details in the predictions and proposed a two-streamed CNN that could simultaneously predict depth and depth gradients, which were then fused together into an accurate and detailed depth map.

Different from Eigen *et al.*'s work [2, 1], other methods added more spices into CNNs for depth estimation. Liu *et al.* [16] assumed that pixels in one super-pixel own the same depth value and inferred the pixel-level depth through a conditional random field (CRF) whose unary and pairwise potentials were learned by a CNN. Later, they improved their model by introducing a super-pixel pooling operation [17] to remove redundant convolutions and reduce computation costs. Similarly, Li *et al.* [13] and Wang *et al.* [27] involved super-pixel-level predictions and then refined them to the pixel-level predictions via CRFs. Moreover, Roy *et al.* [21] combined CNNs with a regression forest, using shallow architectures at each tree node, to avoid the request of large datasets.

More recently, deeper networks for this task have been exploited. Laina *et al.* [12] leveraged the residual learning from ResNet [6] and proposed a fully convolutional architecture with a novel up-sampling method called up-projection, while Xu *et al.* [30, 31] proposed two kinds of sequential deep networks, the cascade one and the unified graphical one, which fused complementary information derived from multiple side outputs of ResNet by the means of CRFs.

### 2.2 Our Motivations & Contributions

Having investigated the literature, we find that in the field of depth estimation based on CNNs there is still large room for further improvement. First, since CNN is usually to reduce feature maps' dimensions, many CNN-based works [2, 1, 12] can only generate low resolution outputs and adopt bilinear interpolation to restore the resolution, which leads to blur in the predictions. Second, even though multi-scale features are involved, there are still many details missing in current methods [2, 1, 19].

In this work, we attempt to solve the aforementioned problems to some extent by proposing a CNN-based approach with multi-scale sub-pixel convolutions and a neighborhood smoothness constraint, namely $MSCN_{NS}$. The advantages and novelties of $MSCN_{NS}$ are highlighted as follows:

(1) We formulate depth estimation problem as a super-resolution problem on depth and take a novel multi-scale target learning method which benefits large network training. And our model, $MSCN_{NS}$, is able to achieve the state-of-the-art results on popular datasets for monocular depth estimation, as well as running much faster than existing methods.

(2) For a fine prediction, $MSCN_{NS}$ makes a good use of multi-scale sub-pixel convolutions, which fuse multi-scale features derived from different branches of the network. Such structures are quite novel as well as reasonable. First, recovering the resolution of depth map can be naturally formulated as a super-resolution problem on depths and the sub-pixel convolution originally proposed by Shi *et al.* [25] is an efficient method to deal with RGB image super-resolution problem. Thus, the sub-pixel convolution is very likely to be effective on depth estimation. Second, many previous studies *e.g.* [2, 1, 19, 29] have shown that, for depth estimation as well as for other pixel-level classification or regression problems, more accurate predictions can be obtained by combining information from multiple scales. Based on those considerations, we propose the multi-scale sub-pixel

convolution as a first attempt to explore super-resolution techniques on depth estimation. More details concerning the sub-pixel convolution will be described in Section 3.3.

(3) Considering that adjacent pixels with similar features in an image should have close depth values, we have proposed a neighborhood smoothness constraint as a regularization term to train $MSCN_{NS}$. When constructing such a constraint, we adopt features learned from CNN automatically instead of traditional hand-crafted features to evaluate the similarity of adjacent pixels in order that it can be integrated with other parts of our model seamlessly. Additionally, our neighborhood smoothness constraint is highly explainable by regarding it as a pixel level conditional random field (CRF).

## 3 $MSCN_{NS}$: The Proposed Method

In this section, we present details of the proposed depth estimation approach $MSCN_{NS}$, including the problem formulation, the network architecture, the multi-scale sub-pixel convolution and the neighborhood smoothness constraint. Fig. 1 gives the outline of the model, which will be described later.

### 3.1 Problem Formulation & Overiew

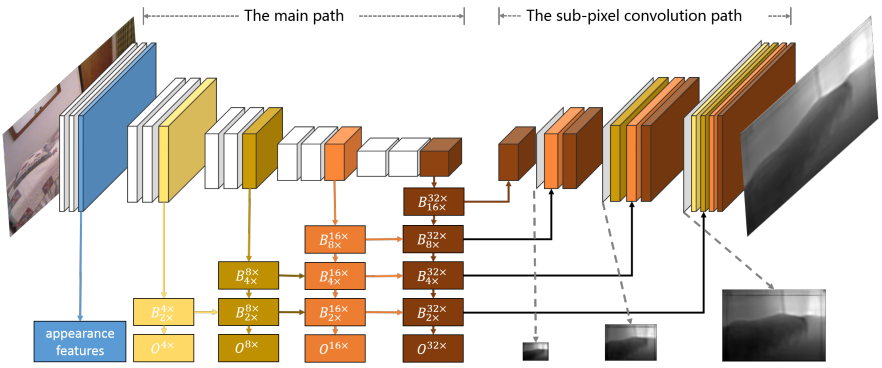



**Fig. 1.** The outline of our model. The blue objects represent the smoothness branch and other coloured objects connecting to the main path with vertical arrows are scale branches. Note that scale branches have no interaction with each other and the horizontal arrows just mean the features from a scale branch will be fused into the corresponding sub-pixel convolution.

Following previous works, we formulate the task of depth estimation from monocular RGB images as the problem of learning a non-linear mapping $F : \mathcal{I} \rightarrow \mathcal{D}$, which maps the image space $\mathcal{I}$ to the depth space $\mathcal{D}$. We denote the

training set as $\mathcal{T} = (X_i, Y_i), i \in \mathbb{N}$, where $X_i \in \mathcal{I}$ and $Y_i \in \mathcal{D}$ representing the corresponding depth map of $X_i$, and denote the test set by $\mathcal{Q}$. Our goal is to construct an $F$ which (a) minimizes the distance of $F(X_j)$ and $Y_j$, (b) ensures that the resolution of $F(X_j)$ is the same as $Y_j$'s and (c) preserves fine details of the scene in $F(X_j)$ for a given input $X_j \in \mathcal{Q}$.

For (a) and (b), we consider making use of CNN with a low resolution output and taking advantage of the sub-pixel convolution to recover the resolution. And for (c), we introduce multi-scale features into the sub-pixel convolution. Therefore, our method can be defined as,

$$P_{final} = F_1(X_j)$$

$$F_n(X_j) = SP_n(F_{2n}(X_j), \varphi_{2n}(X_j), \varphi_{2^2n}(X_j), ..., \varphi_{2^kn}(X_j)) \tag{1}$$

where $n, k \in \mathbb{N}$, $P_{final}$ refers to our final prediction, $n$ and $2^k n$ refer to the downscaling factors, $SP_n$ refers to a sub-pixel convolution which recovers the resolution to $1/n$ of the original input resolution and $\varphi_{2^t n}(\cdot)$ $(t = 1, 2, ..., k)$ refers to an function to get feature maps related to a downscaling factor of $2^t n$.

### 3.2 Network Architecture

We now describe the details of our model. As shown in Fig. 1, our network contains a main path, a sub-pixel convolution path, a smoothness branch and several scale branches. The main path follows the design of DenseNet [8] without the global pooling and the fully connection. We denote the features at a certain scale from the main path by $M_{n\times}$ where $n$ is the downscaling factor of the features whose dimensions are $1/n$ the size of the input. Note that $n \in \{4, 8, 16, 32\}$ which we call the scale set and denote by $\mathcal{S}$.

Each scale branch uses $M_{n\times}$ with a certain $n$, generates features of larger scales via a cascade of transposed convolutions and finally outputs a predicted depth map. We denote the scale branch using $M_{n\times}$ by $B^{n\times}$ and the prediction of $B^{n\times}$ by $O^{n\times}$. Note that $O^{n\times}$ has the same resolution as input's and $n$ in $B^{n\times}$ means that features with a downscaling factor of $n$ (i.e., $M_{n\times}$) from the main path are used. As for features at different scales generated by transposed convolutions of the branch, we denote them by $B^{n\times}_{m\times}$, respectively, where $m$ is the downscaling factor of the features and $m = \{x \mid x \in \{2, 4, 8, 16\}, x < n\}$. The smoothness branch takes $M_{2\times}$ (features coming from the main path with a downscaling factor of 2) as input and tries to learn features to measure the similarity of neighboring pixels.

The sub-pixel convolution path contains four multi-scale sub-pixel convolutions. Each sub-pixel convolution fuses $B^{n\times}_{m\times}$ $(\forall n \in \mathcal{S} \wedge n > m)$ with a certain $m$ and the output of the previous sub-pixel convolution and then generates a higher resolution depth map with a downscaling factor of $t$ $(t = m/2)$. We call this sub-pixel convolution $SP_{t\times}$ and denote its output by $P_{t\times}$. For illustration, $P_{4\times}$ can be presented as,

$$P_{4\times} = SP_{4\times}(P_{8\times}, B^{16\times}_{8\times}, B^{32\times}_{8\times}) \ . \tag{2}$$

During the training phase of our approach, we adopt a multi-scale target learning method to train the sub-pixel convolutions. That is, for a given sample $(X_i, Y_i) \in \mathcal{T}$ we minimize the $l_2$ distance between $Y_i$ and each of $1\times$ scale predictions which contain $P_{1\times}$ and $O^{n\times}$ ($n = 4, 8, 16, 32$, respectively). And then, we down-sample $Y_i$ to $Y_i^t$ where $t$ is the downscaling factor and minimize the $l_2$ distance of $Y_i^t$ and the corresponding $P_{t\times}$ to make sure that details of the scene are preserved. Moreover, we fuse $1\times$ scale predictions via weighted average and minimize the $l_2$ distance between $Y_i$ and the averaged prediction. Thus, our loss function can be defined as a sum of several $l_2$ distances,

$$L_2 = l_2(Y_i, P_{1\times}) + l_2(Y_i, \tilde{Y}_{fs}) + \lambda_1 \sum_{n\in\mathcal{S}} l_2(Y_i, O^{n\times}) + \sum_{t\in\{2,4,8\}} \lambda_2^t l_2(Y_i^t, P_{t\times}) \quad (3)$$

where $l_2(\cdot)$ refers to the $l_2$ distance, $\lambda_1$ and $\lambda_2^t$ are given constants, $\mathcal{S}$ is the scale set and $\tilde{Y}_{fs}$ refers to the weighted average prediction defined as,

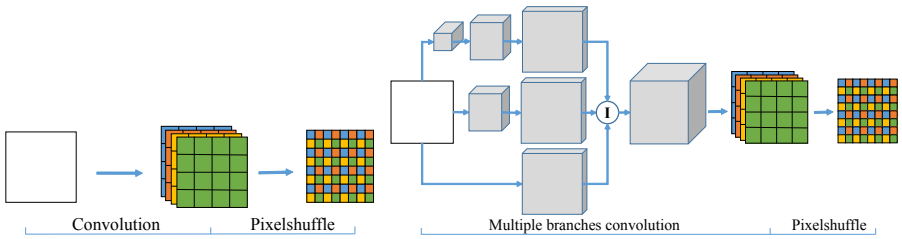
$$\tilde{Y}_{fs} = w_0 P_{1\times} + \sum_{i\in\mathcal{S}} w_i O^{i\times} \quad (4)$$

where $w_0$ and $w_i$ are weights for fusing $1\times$ scale predictions and those weights are learned automatically via CNN.

Additionally, we formulate the neighborhood smoothness constraint as a regularization term, $L_{smooth}$, for training our model, which will be described concretely in Section 3.4. Finally, our final loss function can be defined as,

$$L = L_2 + L_{smooth} \ . \quad (5)$$

### 3.3 Multi-scale Sub-pixel Convolution




(a) Sub-pixel convolution (b) Multi-scale sub-pixel convolution

**Fig. 2.** (a) illustrates the original sub-pixel convolution proposed by [25]. (b) outlines our multi-scale sub-pixel convolution, which fuses more features from different scale branches. The circle with I in it means concatenating features from different branches. And the number of branches before concatenation may be less or more than 3, depending on the output scale of the multi-scale sub-pixel convolution.

Sub-pixel convolution is first proposed by Shi *et al.* [25] as an efficient up-sampling method for RGB image super-resolution problem. It is originally defined as,

$$I^{SR} = f^{L}(I^{LR}) = PS(W_L * f^{L-1}(I^{LR}) + b_L) \tag{6}$$

where $I^{LR}$ refers to a low resolution RGB image, $I^{SR}$ refers to the high resolution RGB image to be recovered, $f^{L-1}$ refers to a neural network with *L-1* layers, $W_L$ and $b_L$ are parameters of Layer $L$ and $PS(\cdot)$ is a shuffling operator that rearranges the elements of an $H \times W \times C \cdot r^2$ tensor to a tensor of the shape $rH \times rW \times C$ where $r$ is the upscaling factor.

Inspired by this, we formulate predicting depth map of a fine resolution as a multi-stage super-resolution problem on depths. In each stage, the predicted depth map is up-sampled by the factor of 2. Therefore, different scale targets should be considered during training and our multi-scale target learning strategy is born. Furthermore, considering that features from early layers of a CNN carry more detail information of the input image and the information is very likely to benefit preserving more details of the scene in the prediction, we construct several branches called scale branches in our network and fuse features of different scales from those branches into the sub-pixel convolution. Finally, suppose that the resolution of the original input image is $H \times W$ and our multi-scale sub-pixel convolution can be defined as,

$$P_{n\times} = PS(H([P_{2n\times}, B_{2n\times}^{2^2 n\times}, B_{2n\times}^{2^3 n\times}, ..., B_{2n\times}^{32\times}])), n \in \{1, 2, 4, 8\} \tag{7}$$

where $[\cdot]$ refers to the concatenation of input feature maps, $H(\cdot)$ refers to a series of convolutional layers which output a tensor with the shape of $H/(2n) \times W/(2n) \times 1 \cdot 4$ and $PS(\cdot)$ represents a pixelshuffle operation which shuffles the output of $H(\cdot)$ to a predicted depth map of resolution $H/n \times W/n$. Note that the aforementioned notation $SP_n(\cdot)$ in (1) is equivalent to $PS(H(\cdot))$ in (7). Fig. 2 gives illustrations of the original sub-pixel convolution and our multi-scale sub-pixel convolution.

### 3.4 Neighborhood Smoothness

The neighborhood smoothness constraint is based on the prior that neighboring pixels with similar appearances in an image are likely to correspond to close depth values. In Liu *et al.*'s work [16], it is presumed that pixels within a superpixel have the same depth value and superpixels with similar appearances have closer depth values. And they make use of hand-crafted features to measure the similarity of adjacent superpixels. Different from them, we use features learned by CNN to measure the similarity of neighboring pixels and the features of each pixel are extracted in a patch whose center is this pixel. And then, we use the weighted average of features of a pixel to represent the appearance of the pixel and define a constraint term in the loss function, $L_{smooth}$, as,

$$L_{smooth} = \frac{\lambda_3}{2} \sum_{j-i=1} (y_i - y_j)^2 e^{-t(r_i - r_j)^2} \tag{8}$$

where $\lambda_3$ and $t$ are positive constants, $i$ and $j$ are the location of two adjacent pixels in a row or a column, $r_i$ and $r_j$ refer to appearances of the two pixels, and $y_i$ and $y_j$ refer to predicted depth values. For better comprehensibility, we will explain why we choose such a formulation and how we implement it, together with why it should work in the following.

As (8) shows, there are two parts, namely, $L_2$ and the weight part. $L_2$ is used to make the predictions of adjacent pixels closer. However, not all adjacent pixels should own close depth. So weights are introduced for such situations. Adjacent pixels which look "different" should own a smaller weight whereas similar pixels should own a larger weight. Therefore, we choose $e^x$ as weights and $x$ should be related to the similarity. We expect to use low level features to formulate the similarity and the first several layers of the CNN are able to learn various low level features. Therefore, we feed $2\times$ feature maps ($M_{2\times}$, blue one in Fig. 1) from the main path to one DenseBlock with 128 output channels. A convolution layer with a kernel size of $5 \times 5$ follows this DenseBlock and generates 4 channel output. The 4 channel output is then shuffled into one channel to up-sample its dimension to the original input's. Finally, each value in the output is regarded as the feature of the pixel in the same location (e.g. $r_i$ or $r_j$ in (8)).

Actually, our neighborhood smoothness constraint can be explained as a conditional random field (CRF) with pairwise potentials defined as,

$$\begin{aligned} \varphi(y_i, y_j, r_i, r_j) &= \mu\left(y_i, y_j\right) k\left(r_i, r_j\right) \\ \mu\left(y_i, y_j\right) = \frac{1}{2}(y_i - y_j)^2 &,\ k\left(r_i, r_j\right) = \lambda e^{-t(r_i - r_j)^2} \end{aligned} \tag{9}$$

where $\mu(y_i, y_j)$ represents the compatibility function between pixel $i$ and $j$ and $k(r_i, r_j)$ represents a self-defined weight kernel.

## 4 Experiments

### 4.1 Experimental Protocol

We evaluate our method on two popular datasets which are publicly available and widely used in the area of depth estimation, Make3D Range Image Dataset [23] and NYU Depth Dataset V2 [26]. The same as prior works, four criteria were considered for quantitative evaluation.

- Average relative error (rel): $\frac{1}{T}\sum_i \frac{|y_i - y_i^*|}{y_i^*}$ .
- Root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_i (y_i - y_i^*)^2}$ .
- Average log10 error (log10): $\frac{1}{T}\sum_i |log_{10} y_i^* - log_{10} y_i|$ .
- Accuracy with threshold (thr): percentage (%) of $y_i$ $s.t.: \max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \delta < thr$ .

where $y_i$ and $y_i^*$ are the predicted depth and the ground-truth depth of the pixel $i$, respectively, and $T$ is the total number of pixels in the evaluated image.

To clearly validate the effectiveness of the multi-scale target learning, the multi-scale sub-pixel convolution and the neighborhood smoothness constraint, we considered the following four baselines:

- **DenseNet-TC** (DenseNet with Transposed Convolution): We trained a DenseNet without the global pooling and the fully connection for depth estimation and used transposed convolution to get a fine resolution of the predicted depth map.
- **DenseNet-MT** (DenseNet with Multi-scale Targets): We trained a network similar as DenseNet-TC and added the multi-scale target learning to it. Specially, each transposed convolution was followed with an extra convolution layer to get an output of a certain scale.
- **DenseNet-SC** (DenseNet with Sub-pixel Convolution): We replaced the transposed convolution in DenseNet-MT with the original sub-pixel convolution in [25].
- **MSCN** (multi-scale sub-pixel convolution network): The proposed approach without the neighborhood smoothness constraint.

### 4.2 Implementation Details

We implement our method on the popular CNN platform, PyTorch [3]. Training is done on Ubuntu 16.04 with an NVIDIA Titan X Pascal GPU. We choose DenseNet-121 as the main path. We reference the implementation of DenseNet-121 in the vision project [4] and use the pre-trained model to initialize parameters of the four denseblocks. Other layers of the network are initialized by the mean of xavier [4]. We use Adam strategy with $weight\_decay = 0.0005$ and set $\lambda_1 = 0.5$, $\lambda_2^i = 1/i, (i = 2, 4, 8)$, $\lambda_3 = 0.01$ and $t = 2$. The batch size is set to 16 due to the memory limitation. The learning rate is set to 0.001 at the very beginning and reduced 70% every $M$ epochs. The value of $M$ depends on the size of the dataset. Generally, we set it to 6$\sim$10 for NYU Depth v2 and 20$\sim$60 for Make3D.

For data augmentation, we follow the strategies proposed in [2] and the parameters are described here. The scaling factor $s \in \{1, 1.2, 1.5\}$, the rotation factor $r \in [-5, 5]$, and the color scaling factor $c \in [0.85, 1.15]$. Moreover, translation and flipping are conducted over all image pairs.

### 4.3 Ablation study on components

We compare the proposed method $MSCN_{NS}$ with several aforementioned baselines to validate the effectiveness of the multi-scale target learning, the multi-scale sub-pixel convolution and the neighborhood smoothness constraint, respectively. The results are shown in Table 1, from which we could have the following

[3] http://pytorch.org/
[4] https://github.com/pytorch/vision

**Table 1.** Baseline comparison on NYU Depth v2. DenseNet-TC and DenseNet-MT verify multi-scale target learning. DenseNet-SC and MSCN verify the multi-scale sub-pixel convolution. MSCN and $MSCN_{NS}$ verify the neighborhood smoothness constraint.

| Method | Error (lower is better) | | | Accuracy (higher is better) | | |
|---|---|---|---|---|---|---|
| | rel | log10 | rms | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
| DenseNet-TC | 0.248 | 0.101 | 0.786 | 0.585 | 0.870 | 0.962 |
| DenseNet-MT | 0.240 | 0.097 | 0.761 | 0.601 | 0.879 | 0.965 |
| DenseNet-SC | 0.227 | 0.094 | 0.720 | 0.613 | 0.885 | 0.969 |
| MSCN | 0.154 | 0.068 | 0.569 | 0.755 | 0.941 | 0.986 |
| $MSCN_{NS}$ | **0.128** | **0.059** | **0.523** | **0.813** | **0.964** | **0.992** |

**Table 2.** Intermediate output comparison on NYU Depth v2. $O^{4\times} \sim O^{32\times}$ are outputs of corresponding scale branches. $P_{8\times} \sim P_{1\times}$ are outputs of corresponding sub-pixel convolutions. Note that $P_{1\times}$ is the final output of our method.

| Output | Error (lower is better) | | | Accuracy (higher is better) | | |
|---|---|---|---|---|---|---|
| | rel | log10 | rms | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
| $O^{4\times}$ | 0.289 | 0.136 | 1.049 | 0.426 | 0.742 | 0.911 |
| $O^{8\times}$ | 0.203 | 0.098 | 0.772 | 0.596 | 0.870 | 0.966 |
| $O^{16\times}$ | 0.156 | 0.076 | 0.634 | 0.719 | 0.914 | 0.978 |
| $O^{32\times}$ | 0.154 | 0.075 | 0.635 | 0.726 | 0.915 | 0.978 |
| $P_{8\times}$ | 0.174 | 0.087 | 0.748 | 0.656 | 0.902 | 0.974 |
| $P_{4\times}$ | 0.154 | 0.075 | 0.663 | 0.723 | 0.932 | 0.982 |
| $P_{2\times}$ | 0.137 | 0.064 | 0.573 | 0.781 | 0.954 | 0.990 |
| $P_{1\times}$ | **0.128** | **0.059** | **0.523** | **0.813** | **0.964** | **0.992** |

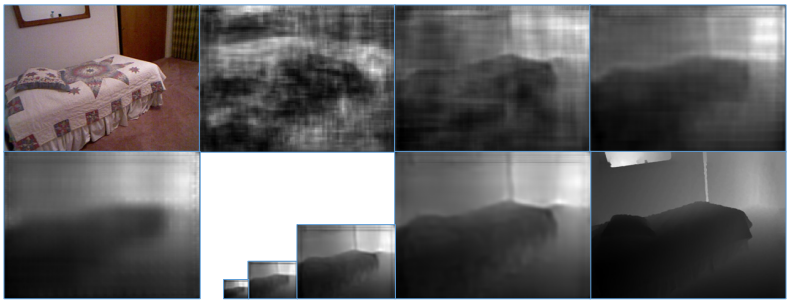


**Fig. 3.** One test sample in NYU Depth V2. The first row (from left to right) is the RGB image, $O^{4\times}$, $O^{8\times}$ and $O^{16\times}$ and the second row (from left to right) is $O^{32\times}$, $P_{8\times}$, $P_{4\times}$, $P_{2\times}$, $P_{1\times}$ (the final prediction) and the ground-truth depth map.

**Table 3.** Performance evaluation of different methods on NYU Depth v2.

| **Method** | **Error (lower is better)** | | | **Accuracy (higher is better)** | | |
|---|---|---|---|---|---|---|
| | rel | log10 | rms | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
| Saxena *et al.* [23] | 0.349 | - | 1.214 | 0.447 | 0.745 | 0.897 |
| Karsch *et al.* [10] | 0.350 | 0.131 | 1.2 | - | - | - |
| Liu *et al.* [18] | 0.335 | 0.127 | 1.06 | - | - | - |
| Eigen *et al.* [2] | 0.215 | - | 0.907 | 0.611 | 0.887 | 0.971 |
| Liu *et al.* [17] | 0.234 | 0.095 | 0.842 | 0.604 | 0.885 | 0.973 |
| Mousavian *et al.* [19] | 0.200 | - | 0.816 | 0.568 | 0.856 | 0.956 |
| Roy and Todorovic [21] | 0.187 | 0.078 | 0.744 | - | - | - |
| Eigen and Fergus [1] | 0.158 | - | 0.641 | 0.769 | 0.950 | 0.988 |
| Li *et al.* [14] | 0.143 | 0.063 | 0.635 | 0.788 | 0.958 | 0.991 |
| Laina *et al.* ($l_2$) [12] | 0.138 | 0.060 | 0.592 | 0.785 | 0.952 | 0.980 |
| $MSCN_{NS}$ (Ours) | **0.128** | **0.059** | **0.523** | **0.813** | **0.964** | **0.992** |

**Table 4.** Time evaluation on NYU Depth v2. All methods are evaluated using the same computer with an NVIDIA Titan X (Pascal) GPU.

| **Method** | **Time** |
|---|---|
| Karsch *et al.* [10] | 60s |
| Eigen *et al.* [2] | 2.091s |
| Eigen and Fergus [1] | 5.622s |
| Liu *et al.* [17] | 1.291s |
| Laina *et al.* [12] | 0.257s |
| $MSCN_{NS}$ (Ours) | **0.019s** |

**Table 5.** Performance evaluation of different methods on Make3D.

| **Method** | **C1 error(lower is better)** | | | **C2 error(lower is better)** | | |
|---|---|---|---|---|---|---|
| | rel | log10 | rms | rel | log10 | rms |
| Saxena *et al.* [23] | - | - | - | 0.370 | 0.187 | - |
| Karsch *et al.* [10] | 0.355 | 0.127 | 9.20 | 0.361 | 0.148 | 15.10 |
| Liu *et al.* [18] | 0.335 | 0.137 | 9.49 | 0.338 | 0.134 | 12.60 |
| Liu *et al.* [16] | 0.314 | 0.119 | 8.60 | 0.307 | 0.125 | 12.89 |
| Zhang *et al.* [34] | 0.374 | 0.127 | 9.18 | 0.364 | 0.141 | 14.11 |
| Liu *et al.* [17] | 0.312 | 0.113 | 9.10 | 0.305 | 0.120 | 13.24 |
| Roy and Todorovic [21] | - | - | - | 0.260 | 0.119 | 12.40 |
| Li *et al.* [13] | 0.278 | 0.092 | 7.19 | 0.279 | 0.102 | 10.67 |
| $MSCN_{NS}$ (Ours) | **0.254** | **0.080** | **6.92** | **0.249** | **0.088** | **10.47** |

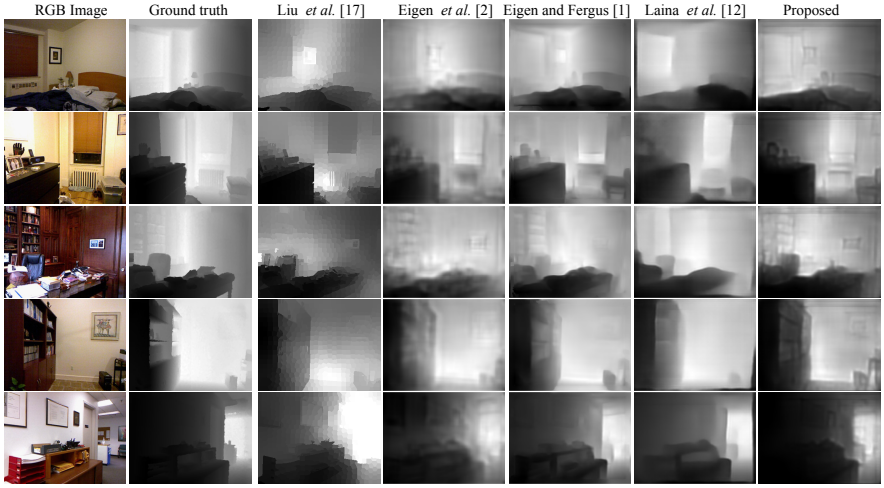



**Fig. 4.** Qualitative evaluations on NYU Depth v2. Note that the output resolution of the proposed method is the same as input's, whereas Eigen *et al.*'s [2] is 1/4 of the input's, Eigen and Fergus's [1] is 1/2 of the input's and Laina *et al.*'s [12] is 1/2 of the input's.


**Fig. 5.** Qualitative evaluations of our approach on Make3D. Fine boundaries of objects in different environments are provided in our predictions.

findings. First, all approaches with the multi-scale target learning perform better than DenseNet-TC, indicating that our multi-scale target learning is very effective for depth estimation. Second, DenseNet-SC outperforms DenseNet-MT and MSCN outperforms both DenseNet-SC and DenseNet-MT with a great increase on performance. Such a result clearly demonstrates that sub-pixel convolution is more suitable than the widely used transposed convolution for this task and the proposed multi-scale sub-pixel convolution is very powerful for improving the predictions. This result is quite reasonable, since multi-scale information benefits keeping details of the scene in the prediction. Third, compared with MSCN, $MSCN_{NS}$ gains obvious performance increases on all criteria. Considering that the only difference between them is that the former is trained with the neighborhood smoothness regularization term, we can conclude that our neighborhood smoothness constraint further improve the results.

Since there are multiple outputs ($O^{4\times} \sim O^{32\times}$ and $P_{8\times} \sim P_{1\times}$) in our network, we display the qualitative results of them in Fig. 3 and present the quantitative comparisons in Table 2, from which several interesting conclusions can be drawn. First, the performances from $O^{4\times}$ to $O^{32\times}$ increase sequentially whereas the figures become less sharp. Such a phenomenon clearly illustrates that the early layers of the network provide more information about the contour of the object and deeper layers provide more useful information for depth values. Thus, there is a trade-off between sharpness and accuracy using single scale features. Second, $P_{8\times} \sim P_{1\times}$ are nearly the same irrespective of the resolution, which indicates that our multi-scale sub-pixel convolution overcomes this trade-off and is able to provide a sharp and accurate prediction.

### 4.4 Evaluation on NYU Depth Dataset v2

The NYU Depth Dataset V2 [26] is an indoor scene RGB-D dataset. Following Laina *et al.*'s work [12], we sample frames with a fixed step out of each training sequence and acquire approximately 12k images. After the offline data augmentation, we finally get around 72k samples for training. For test, we use the standard test set, which contains 694 images with filled-in depth values, to compare with previous works. During our experiment, all images with the corresponding depth maps are down-sampled to $320 \times 240$.

To show the superiority of $MSCN_{NS}$, we compare it with the state-of-the-art methods in terms of the prediction accuracy and the inference speed. The results are listed in Table 3 and Table 4. It can be seen that our method outperforms other methods on all criteria. And it runs much faster and only consumes 1/13 the time of the runner-up, meaning that it can meet the requirement of real-time application. Moreover, our training set with 72K images is smaller than Laina *et al.*'s [12] with 95K images, as well as Eigen and Fergus's [1] with 120K images. Besides, the size of our model is approximately 100 M, which is 40% that of Laina *et al.*'s and 1/8 that of Eigen and Fergus's. For further comparison, we provide qualitative results of several methods in Fig. 4.

### 4.5 Evaluation on Make3D

The Make3D dataset [23] is an outdoor scene RGB-D dataset, containing 534 images with the resolution of $1704 \times 2272$. Officially, images of this dataset are split into 400 images for training and 134 images for test. We get 9.6K images via offline data augmentation and resize all images to $345 \times 460$. Following Laina *et al.*'s work [12], we further reduce the resolution of the images by half as the input of our network. Following previous works, we report our results using two kinds of criteria, C1 error and C2 error, as previous works used. C1 errors are calculated only in the regions with the ground-truth less than 70 meters while C2 errors are calculated over the entire images.

We evaluate our method using the official test set with 134 images and list the comparison results with several state-of-the-art approaches in Table 5. It can be observed that our method outperforms all competitors on all criteria. Moreover, it should be noted that Make3D is an outdoor depth dataset. And thus the range of its depth values is much larger than that of indoor dataset like NYU Depth v2, which brings more challenges for the prediction. Nevertheless, our method makes a great improvement on the metric log10 which refers to an absolute error on logarithm. Therefore, it can be concluded that our method is superior to predict relatively accurate depth value in spite of a large range of ground truth value.

We also provide the qualitative evaluations on Make3D, which are presented in Fig. 5. Since the depth range is large, we provide the reverse depth map for better visualization. As is shown, our method provides fine boundaries of objects even in complex environments involving buildings, shrubs and trees.

## 5 Conclusion

In this paper, we have presented a CNN-based approach for depth estimation from a single monocular image. In our model, there is a main path, a sub-pixel convolution path, a smoothness branch and four scale branches. The main path and scale branches generate features of different scales. And the sub-pixel convolution path leverages the multi-scale sub-pixel convolutions to get an accurate depth map. During training, a novel multi-scale target learning strategy is adopted to train the sub-pixel convolutions. Moreover, we have proposed a novel neighborhood smoothness constraint that makes use of features from the smoothness branch to further improve the performance. Our approach is able to achieve the state-of-the-art results, as well as running much faster. Additionally, it can provide more details of the scene and high resolution depth maps.

## 6 Acknowledgement

This work was supported in part by the Natural Science Foundation of China under Grant 61672380, in part by the Fundamental Research Funds for the Central Universities under Grant 2100219068, and in part by the Shanghai Automotive Industry Science and Technology Development Foundation under Grant 1712.

## References


1. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. pp. 2650–2658 (2015)
2. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS. pp. 2366–2374 (2014)
3. Eitel, A., Springenberg, J.T., Spinello, L., Riedmiller, M., Burgard, W.: Multimodal deep learning for robust RGB-D object recognition. In: IROS. pp. 681–687 (2015)
4. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. pp. 249–256 (2010)
5. Gupta, S., Arbeláez, P., Girshick, R., Malik, J.: Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation. IJCV pp. 133–149 (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
7. Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: CVPR. pp. 807–814 (2005)
8. Huang, G., Liu, Z., Maaten, L.V.D., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 2261–2269 (2017)
9. Karsch, K., Liu, C., Kang, S.B.: Depth extraction from video using non-parametric sampling. In: ECCV. pp. 775–788 (2012)
10. Karsch, K., Liu, C., Kang, S.B.: Depth transfer: Depth extraction from video using non-parametric sampling. IEEE Trans. PAMI pp. 2144–2158 (2014)
11. Konrad, J., Wang, M., Ishwar, P.: 2D-to-3D image conversion by learning depth from examples. In: CVPRW. pp. 16–22 (2012)
12. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3DV. pp. 239–248 (2016)
13. Li, B., Shen, C., Dai, Y., van den Hengel, A., He, M.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In: CVPR. pp. 1119–1127 (2015)
14. Li, J., Klein, R., Yao, A.: A two-streamed network for estimating fine-scaled depth maps from single RGB images. In: CVPR. pp. 3372–3380 (2017)
15. Liu, B., Gould, S., Koller, D.: Single image depth estimation from predicted semantic labels. In: CVPR. pp. 1253–1260 (2010)
16. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: CVPR. pp. 5162–5170 (2015)
17. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Trans. PAMI pp. 2024–2039 (2016)
18. Liu, M., Salzmann, M., He, X.: Discrete-continuous depth estimation from a single image. In: CVPR. pp. 716–723 (2014)
19. Mousavian, A., Pirsiavash, H., Košecká, J.: Joint semantic segmentation and depth estimation with deep convolutional networks. In: 3DV. pp. 611–619 (2016)
20. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: Features and algorithms. In: CVPR. pp. 2759–2766 (2012)
21. Roy, A., Todorovic, S.: Monocular depth estimation using neural regression forest. In: CVPR. pp. 5506–5514 (2016)
22. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: NIPS. pp. 1161–1168 (2006)
23. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D scene structure from a single still image. IEEE Trans. PAMI pp. 824–840 (2009)

24. Seki, A., Pollefeys, M.: Sgm-nets: Semi-global matching with neural networks. In: CVPRW. pp. 21–26 (2017)
25. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR. pp. 1874–1883 (2016)
26. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: ECCV. pp. 746–760 (2012)
27. Wang, P., Shen, X., Lin, Z., Cohen, S., Price, B., Yuille, A.L.: Towards unified depth and semantic prediction from a single image. In: CVPR. pp. 2800–2809 (2015)
28. Wang, Z., Liu, H., Wang, X., Qian, Y.: Segment and label indoor scene based on RGB-D for the visually impaired. In: MMM. pp. 449–460 (2014)
29. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV. pp. 1395–1403 (2015)
30. Xu, D., Ricci, E., Ouyang, W., Wang, X., Sebe, N.: Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation. In: CVPR. pp. 161–169 (2017)
31. Xu, D., Ricci, E., Ouyang, W., Wang, X., Sebe, N.: Monocular depth estimation using multi-scale continuous crfs as sequential deep networks. IEEE Trans. PAMI (Early Access) (2018)
32. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. JMLR pp. 2287–2318 (2016)
33. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In: CVPR. pp. 199–208 (2017)
34. Zhang, Y., Ji, R., Fan, X., Wang, Y., Guo, F., Gao, Y., Zhao, D.: Search-based depth estimation via coupled dictionary learning with large-margin structure inference. In: ECCV. pp. 858–874 (2016)