Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields

Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid

Abstract—In this article, we tackle the problem of depth estimation from single monocular images. Compared with depth estimation using multiple images such as stereo depth perception, depth from monocular images is much more challenging. Prior work typically focuses on exploiting geometric priors or additional sources of information, most using hand-crafted features. Recently, there is mounting evidence that features from deep convolutional neural networks (CNN) set new records for various vision applications. On the other hand, considering the continuous characteristic of the depth values, depth estimation can be naturally formulated as a continuous conditional random field (CRF) learning problem. Therefore, here we present a deep convolutional neural field model for estimating depths from single monocular images, aiming to jointly explore the capacity of deep CNN and continuous CRF. In particular, we propose a deep structured learning scheme which learns the unary and pairwise potentials of continuous CRF in a unified deep CNN framework. We then further propose an equally effective model based on fully convolutional networks and a novel superpixel pooling method, which is about 10 times faster, to speedup the patch-wise convolutions in the deep model. With this more efficient model, we are able to design deeper networks to pursue better performance. Our proposed method can be used for depth estimation of general scenes with no geometric priors nor any extra information injected. In our case, the integral of the partition function can be calculated in a closed form such that we can exactly solve the log-likelihood maximization. Moreover, solving the inference problem for predicting depths of a test image is highly efficient as closed-form solutions exist. Experiments on both indoor and outdoor scene datasets demonstrate that the proposed method outperforms state-of-the-art depth estimation approaches.

Index Terms—Depth estimation, conditional random field (CRF), deep convolutional neural networks (CNN), fully convolutional networks, superpixel pooling

1 Introduction

Estimating depth information from single monocular images depicting general scenes is an important problem in computer vision. Many challenging computer vision problems have proven to benefit from the incorporation of depth information, to name a few, semantic labellings [1], pose estimations [2]. Although the highly developed depth sensors such as Microsoft Kinect nowadays have made the acquisition of RGBD images affordable, most of the vision datasets commonly evaluated among the vision community are still RGB images. Moreover, outdoor applications still rely on LiDAR or other laser sensors due to the fact that strong sunlight can cause infrared interference and make depth information extremely noisy. This has led to wide research interest on the topic of estimating depths from single RGB images. Unfortunately, it is a notoriously ill-posed problem, as one captured image scene may correspond to numerous real world scenarios [3].

Whereas for humans, inferring the underlying 3D structure from a single image is effortless, it remains a challenging task for automated computer vision systems to do so since no reliable cues can be exploited, such as temporal information in videos, stereo correspondences, etc. Previous work mainly focuses on enforcing geometric assumptions, e.g., box models, to infer the spatial layout of a room [4], [5] or outdoor scenes [6]. These models come with innate restrictions, which are limitations to model only particular scene structures and therefore are not applicable for general scene depth estimations. More recently, non-parametric methods [7] are explored, which consists of candidate images retrieval, scene alignment and then depth inference using optimizations with smoothness constraints. This is based on the assumption that scenes with semantically similar appearances should have similar depth distributions when densely aligned. However, this method is prone to propagate errors through the different decoupled stages and relies heavily on building a reasonable sized image database to perform the candidate retrieval. In recent years, efforts have been made towards incorporating additional sources of information, e.g., user annotations [8], semantic labellings [1], [9]. In the recent work of [1], Ladick et al. have shown that jointly performing depth estimation and semantic labelling can benefit each other. Nevertheless, all these methods use hand-crafted features.

In contrast to previous efforts, here we propose to formulate the depth estimation as a deep continuous Conditional Random Fields (CRF) learning problem, without relying on any geometric priors or any extra information. CRF [10] is a popular graphical model for structured output predictions. While extensively studied in classification (discrete) domains,

- *F. Liu is with the School of Computer Science, The University of Adelaide, Australia. E-mail: fayao.liu@adelaide.edu.au.*
- *C. Shen, G. Lin and I. Reid are with the ARC Centre of Excellence for Robotic Vision and the School of Computer Science, The University of Adelaide, Australia.*
  E-mail: {chunhua.shen, guosheng.lin, ian.reid}@adelaide.edu.au.

Manuscript received 2 Mar. 2015; revised 7 Oct. 2015; accepted 17 Nov. 2015. Date of publication 2 Dec. 2015; date of current version 12 Sept. 2016. Recommended for acceptance by A. Vedaldi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2505283

CRF has been less explored for regression (continuous) problems. One of the pioneer work on continuous CRF can be attributed to [11], in which it was proposed for global ranking in document retrieval. Under certain constraints, they can directly solve the maximum likelihood optimization as the partition function can be analytically calculated. Since then, continuous CRF has been successfully applied for solving various structured regression problems, e.g., remote sensing [12], [13], image denoising [13]. Motivated by these successes, we here propose to use it for depth estimation, given the continuous nature of the depth values, and learn the potential functions in a deep convolutional neural network (CNN).

Recent years have witnessed the prosperity of the deep CNN [14] since the breakthrough work of Krizhevsky et al. [15]. CNN features have been setting new records for a wide variety of vision applications [16]. Despite all the successes in classification problems, deep CNN has been less explored for structured learning problems, i.e., joint training of a deep CNN and a graphical model, which is a relatively new and not well addressed problem. To our knowledge, no such model has been successfully used for depth estimations. Here, we bridge this gap by jointly exploring CNN and continuous CRF, denoting this new method as a deep convolutional neural field (DCNF).

Fully convolutional networks have recently been studied for dense prediction problems, e.g., semantic labelling [17], [18]. Models based on fully convolutional networks have the advantage of highly efficient training and prediction. We here exploit this advance to speedup the training and prediction of our DCNF model. However, the feature maps produced by the fully convolutional models are typically much smaller than the input image size. This can cause problems for both training and prediction. During training, one needs to downsample the ground-truth maps, which may lead to information loss since small objects might disappear. In prediction, the upsampling operation is likely to bring in degraded performance at the object boundaries. We therefore propose a novel superpixel pooling method to address this issue. It jointly exploits the strengths of highly efficient fully convolutional networks and the benefits of superpixels at preserving object boundaries. We show some prediction examples in Fig. 1.

To sum up, we highlight the main contributions of this work as follows.

- We propose a deep convolutional neural field model for depth estimations by exploring CNN and continuous CRF. Given the continuous nature of the depth values, the partition function in the probability density function can be analytically calculated, therefore we can directly solve the log-likelihood optimization without any approximations. The gradients can be exactly calculated in the back propagation training. Moreover, solving the MAP problem for predicting the depth of a new image is highly efficient since closed form solutions exist.
- We jointly learn the unary and pairwise potentials of the CRF in a unified deep CNN framework, which is trained using back propagation.
- We propose a faster model based on fully convolutional networks and a novel superpixel pooling method, which results in ~ 10 times speedup while producing similar prediction accuracy. With this more efficient model, which we refer as DCNF-FCSP, we are able to design very deep networks for better performance.
- We demonstrate that the proposed method outperforms state-of-the-art results of depth estimation on both indoor and outdoor scene datasets.

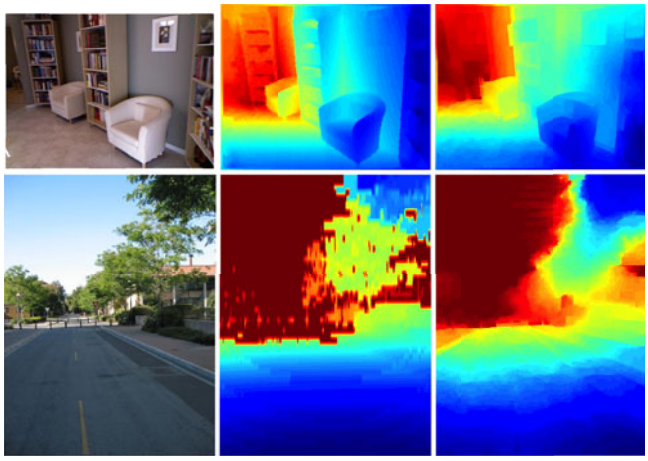


Fig. 1. Examples of depth estimation results using the proposed deep convolutional neural fields model. First row: NYU v2 dataset; second row: Make3D dataset. From left to right: input image, ground-truth, our prediction.

Preliminary results of our work appeared in Liu et al. [19].

1.1 Related Work

Our method exploits the recent advances of deep nets in image classification [15], [20], object detection [21] and semantic segmentation [17], [18], for single view image depth estimations. In the following, we give a brief introduction to the most closely related work.

Depth estimation. Traditional methods [9], [22], [23] typically formulate the depth estimation as a Markov Random Field (MRF) learning problem. As exact MRF learning and inference are intractable in general, most of these approaches employ approximation methods, e.g., multi-conditional learning (MCL) or particle belief propagation (PBP). Predicting the depths of a new image is inefficient, taking around 4-5 s in [23] and even longer (30 s) in [9]. To make things worse, these methods suffer from lacking of flexibility in that [22], [23] rely on horizontal alignment of images and [9] requires the semantic labellings of the training data available beforehand. More recently, Liu et al. [24] propose a discrete-continuous CRF model to take into consideration the relations between adjacent superpixels, e.g., occlusions. They also need to use approximation methods for learning and maximum a posteriori (MAP) inference. Besides, their method relies on image retrieval to obtain a reasonable initialization. In contrast, here we present a deep continuous CRF model in which we can directly solve the log-likelihood optimization without any approximations, since the partition function can be analytically calculated. Predicting the depth of a new image is highly efficient since a closed form solution exists. Moreover, we do not inject any geometric priors nor any extra information. On the other hand, previous methods [1], [7], [9], [23], [24] all use

hand-crafted features in their work, e.g., texton, GIST, SIFT, PHOG, object bank, etc. In contrast, we learn deep CNN for constructing unary and pairwise potentials of the CRF.

Recently, Eigen et al. [3] proposed a multi-scale CNN approach for depth estimation, which bears similarity to our work here. However, our method differs critically from theirs: they use the CNN as a black-box by directly regressing the depth map from an input image through convolutions; in contrast we use a CRF to explicitly model the relations of neighboring superpixels, and learn the potentials (both unary and binary) in a unified CNN framework.

Recent work of [25] and [26] is relevant to ours in that they also perform depth estimation from a single image. The method of Su et al. [25] involves a continuous depth optimization step like ours, which also contains a unary regression term and a pairwise local smoothness term. However, these two works focus on 3D reconstruction of known segmented objects while our method targets at depth estimation of general scene images. Furthermore, the method of [25] relies on a pre-constructed 3D shape database of input object categories, and the work of [26] relies on class-specific object keypoints and object segmentations. In contrast, we do not inject these priors.

Combining CNN and CRF. In [27], Farabet et al. propose a multi-scale CNN framework for scene labelling, which uses CRF as a post-processing step for local refinement. In the most recent work of [28], Tompson et al. present a hybrid architecture for jointly training a deep CNN and an MRF for human pose estimation. They first train a unary term and a spatial model separately, then jointly learn them as a fine tuning step. During fine tuning of the whole model, they simply remove the partition function in the likelihood to have a loose approximation. In contrast, our model performs continuous variable prediction. We can directly solve the log-likelihood optimization without using approximations as the partition function is integrable and can be analytically calculated. Moreover, during prediction, we have closed-form solutions to the MAP inference problem. Although no convolutional layers are involved, the work of [29] shares similarity with ours in that both continuous CRF's use neural networks to model the potentials. Note that the model in [29] is not deep and only one hidden layer is used. It is unclear how the method of [29] performs on the challenging depth estimation problem that we consider here.

Fully convolutional networks. Fully convolutional networks have recently been actively studied for dense prediction problems, e.g., semantic segmentation [17], [18], image restoration [30], image super-resolution [31], depth estimations [3]. To deal with the downsampled output issue, interpolations are generally applied [3], [18]. In [32], Sermanet et al. propose an input shifting and output interlacing trick to produce dense predictions from coarse outputs without interpolations. Later on, Long et al. [17] present a deconvolution approach to put the upsampling into the training regime instead of applying it as a post-processing step. The CNN model presented in Eigen et al. [3] for depth estimation also suffers from this upsampling problem—the predicted depth maps of [3] is 1/4-resolution of the original input image with some border areas lost. They simply use bilinear interpolations to upsample the predictions to the input image size. Unlike these existing methods, we propose a novel superpixel pooling method to address this issue. It jointly exploits the strengths of highly efficient fully convolutional networks and the benefits of superpixels at preserving object boundaries.

2 Deep Convolutional Neural Fields

We present the details of our deep convolutional neural field model for depth estimation in this section. Unless otherwise stated, we use boldfaced uppercase and lowercase letters to denote matrices and column vectors respectively; $\mathbf{0}$ represents a column vector with all elements being 0.

2.1 Overview

The goal here is to infer the depth of each pixel in a single image depicting a general scene. Following the work of [9], [23], [24], we make the common assumption that an image is composed of small homogeneous regions (superpixels) and consider the graphical model composed of nodes defined on superpixels. Each superpixel is portrayed by the depth of its centroid. Let $\mathbf{x}$ be an image and $\mathbf{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ be a vector of continuous depth values corresponding to all n superpixels in $\mathbf{x}$. Similar to conventional CRFs, we model the conditional probability distribution of the data with the following density function:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathrm{Z}(\mathbf{x})}\exp(-E(\mathbf{y},\mathbf{x})), \tag{1}$$

where E is the energy function; Z is the partition function defined as:

$$\mathrm{Z}(\mathbf{x}) = \int_{\mathbf{y}} \exp\{-E(\mathbf{y},\mathbf{x})\}\mathrm{d}\mathbf{y}. \tag{2}$$

Here, because $\mathbf{y}$ is continuous, the integral in Eq. (1) can be analytically calculated under certain circumstances, which we will show in Section 2.3. This is different from the discrete case, in which approximation methods need to be applied. To predict the depth of a new image, we solve the following MAP inference problem:

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}). \tag{3}$$

We formulate the energy function as a typical combination of unary potentials U and pairwise potentials V over the nodes (superpixels) $\mathcal{N}$ and edges $\mathcal{S}$ of the image $\mathbf{x}$:

$$E(\mathbf{y},\mathbf{x}) = \sum_{p\in\mathcal{N}} U(y_p,\mathbf{x}) + \sum_{(p,q)\in\mathcal{S}} V(y_p,y_q,\mathbf{x}). \tag{4}$$

The unary term U aims to regress the depth value for a single superpixel. The pairwise term V encourages neighboring superpixels with similar appearances to take similar depths. We aim to jointly learn U and V in a unified CNN framework.

Fig. 2 sketches our deep convolutional neural field model for depth estimation. The whole network comprises a unary part, a pairwise part and a continuous CRF loss layer. For an input image, which has been over-segmented into n superpixels, we consider image patches centred around each superpixel centroid. The unary part then takes an image

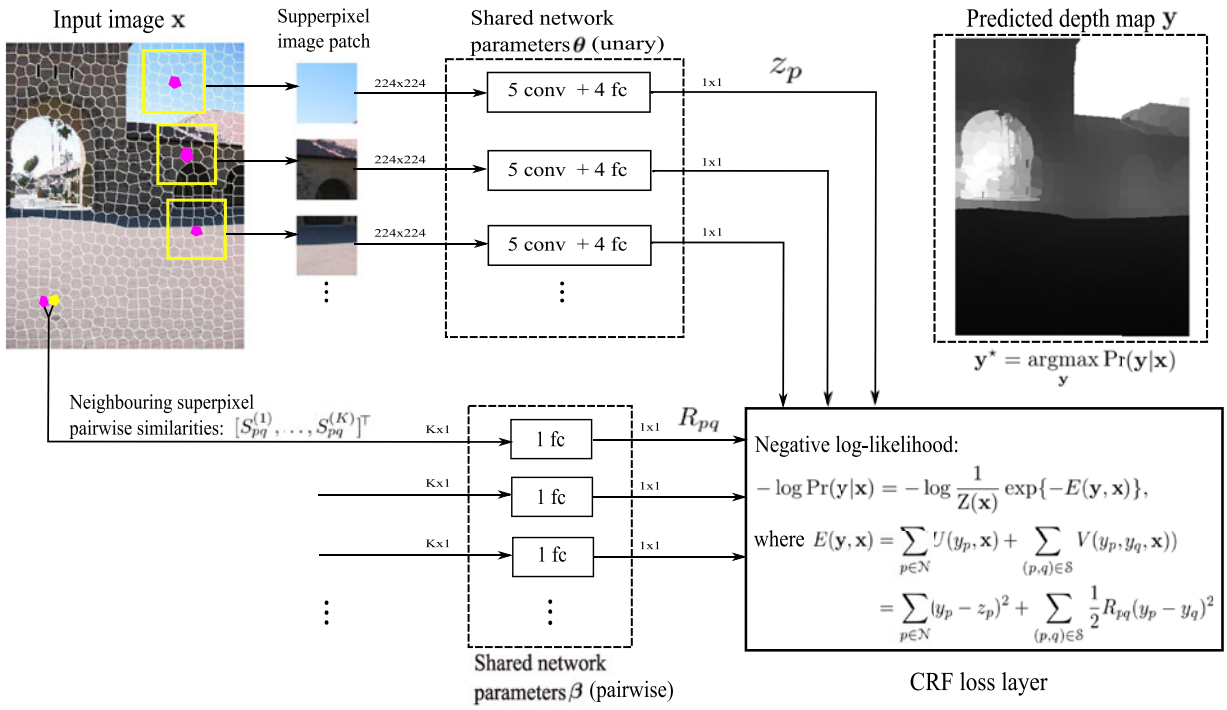


Fig. 2. An illustration of our DCNF model for depth estimation. The input image is first over-segmented into superpixels. In the unary part, for a superpixel p, we crop the image patch centred around its centroid, then resize and feed it to a CNN which is composed of five convolutional and four fully-connected layers (details refer to Fig. 3). In the pairwise part, for a pair of neighboring superpixels (p, q), we consider K types of similarities, and feed them into a fully-connected layer. The outputs of unary part and the pairwise part are then fed to the CRF structured loss layer, which minimizes the negative log-likelihood. Predicting the depths of a new image $\mathbf{x}$ is to maximize the conditional probability $\Pr(\mathbf{y}|\mathbf{x})$, which has closed-form solutions (see Section 2.3 for details).

patch as input and feeds it to a CNN whose output is a single number, the regressed depth value of the superpixel.

The network for the unary part is composed of five convolutional and four fully-connected layers with details in Fig. 3. Note that the CNN parameters are shared across all the superpixels. The pairwise part takes similarity vectors (each with K components) of all neighboring superpixel pairs as input and feeds each of them to a fully-connected layer (parameters are shared among different pairs), then outputs a vector containing all the one-dimensional similarities for each of the neighboring superpixel pairs. The continuous CRF loss layer takes the outputs from the unary and the pairwise terms to minimize the negative log-likelihood. Compared to the direct regression method in [3], our model possesses two potential advantages:

- We achieve translation invariance as we construct unary potentials irrespective of the superpixel's coordinate (shown in Section 2.2);
- We explicitly model the relations of neighboring superpixels through pairwise potentials.

In the following, we describe the details of potential functions involved in the energy function in Eq. (4).

2.2 Potential Functions

2.2.1 *Unary Potential*

The unary potential is constructed from the output of a CNN by considering the least square loss:

$$U(y_p, \mathbf{x}; \boldsymbol{\theta}) = (y_p - z_p(\boldsymbol{\theta}))^2, \quad \forall p = 1, ..., n. \tag{5}$$

Here z_p is the regressed depth of the superpixel p parametrized by the CNN parameters $\boldsymbol{\theta}$.

The network architecture for the unary part is depicted in Fig. 3. Our CNN model in Fig. 3 is mainly inspired by the well-known network architecture of Krizhevsky et al. [15] with modifications. It is composed of five convolutional layers and four fully-connected layers. The input image is first over-segmented into superpixels, then for each superpixel, we consider the image patch centred around its centroid. Each of the image patches is resized to 224×224 pixels (other resolutions also work) and then fed to the convolutional neural network. Note that the convolutional and the fully-connected layers are shared across all the image patches of different superpixels. Rectified linear units (ReLU) are used as activation functions for the five convolutional layers

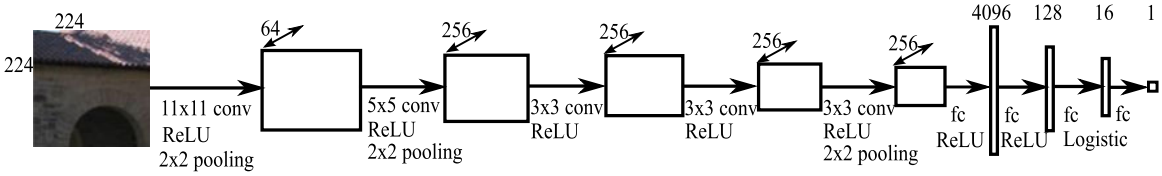


Fig. 3. Detailed network architecture of the unary part in Fig. 2.

and the first two fully connected layers. For the third fully-connected layer, we use the logistic function $f(x) = (1 + e^{-x})^{-1}$ as the activation function. The last fully-connected layer plays the role of model ensemble with no activation function followed. The output is an one-dimensional real-valued depth for a single superpixel.

2.2.2 Pairwise Potential

We construct the pairwise potential from K types of similarity observations, each of which enforces smoothness by exploiting consistency information of neighboring superpixels:

$$V(y_p, y_q, \mathbf{x}; \boldsymbol{\beta}) = \frac{1}{2} R_{pq}(y_p - y_q)^2, \ \forall p, q = 1, ..., n. \tag{6}$$

Here R_{pq} is the output of the network in the pairwise part (see Fig. 2) from a neighboring superpixel pair (p, q). We use a fully-connected layer here:

$$R_{pq} = \boldsymbol{\beta}^\top [S_{pq}^{(1)}, \ldots, S_{pq}^{(K)}]^\top = \sum_{k=1}^{K} \beta_k S_{pq}^{(k)}, \tag{7}$$

where $\mathbf{S}^{(k)}$ is the kth similarity matrix whose elements are $S_{pq}^{(k)}$ ($\mathbf{S}^{(k)}$ is symmetric); $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_k]^\top$ are the network parameters. From Eq. (7), we can see that we do not use any activation function. However, as our framework is general, more complicated networks may be seamlessly incorporated for the pairwise part. In Section 2.3, we will show that we can derive a general form for calculating the gradients with respect to $\boldsymbol{\beta}$ (see Eq. (19)). To ensure that $\mathrm{Z}(x)$ in Eq. (2) is integrable, we require $\beta_k \geq 0$ as in [11]. Note that this is a sufficient but not necessary condition.

Here we consider three types of pairwise similarities, measured by the colour difference, colour histogram difference and texture disparity in terms of local binary patterns (LBP) [33], which take the conventional form:

$$S_{pq}^{(k)} = e^{-\gamma \left\| s_p^{(k)} - s_q^{(k)} \right\|}, \ k = 1, 2, 3;$$

where $s_p^{(k)}$, $s_q^{(k)}$ are the observation values of the superpixel p, q calculated from colour, colour histogram and LBP; $\|\cdot\|$ denotes the ℓ_2 norm of a vector and γ is a constant. It may be possible to learn features for the pairwise term too. For example, the pairwise term can be a deep CNN with raw pixels as the input. A more sophisticated pairwise energy may further improve the estimation, especially for complex discrete labelling problems, with the price of increased computation complexity. For depth estimation, we find that our current pairwise energy already works very well.

2.3 Learning

With the unary and the pairwise potentials defined in Eqs. (5), (6), we can now write the energy function as:

$$E(\mathbf{y}, \mathbf{x}) = \sum_{p \in \mathcal{N}} (y_p - z_p)^2 + \sum_{(p,q) \in \mathcal{S}} \frac{1}{2} R_{pq}(y_p - y_q)^2. \tag{8}$$

For ease of expression, we introduce the following notation:

$$\mathbf{A} = \mathbf{I} + \mathbf{D} - \mathbf{R}, \tag{9}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix; $\mathbf{R}$ is the affinity matrix composed of R_{pq}; $\mathbf{D}$ is a diagonal matrix with $D_{pp} = \sum_q R_{pq}$. We see that $\mathbf{D} - \mathbf{R}$ is the graph Laplacian matrix. Thus $\mathbf{A}$ is the regularized Laplacian matrix. Expanding Eq. (8), we have:

$$E(\mathbf{y}, \mathbf{x}) = \mathbf{y}^\top \mathbf{A} \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{z}. \tag{10}$$

Due to the quadratic terms of $\mathbf{y}$ in the energy function in Eq. (10) and the positive definiteness of $\mathbf{A}$ (with all positive edges of the graph, the Laplacian matrix must be positive semidefinite and therefore $\mathbf{A}$ must be positive definite), we can analytically calculate the integral in the partition function (Eq. (2)) as:

$$\begin{aligned} \mathrm{Z}(\mathbf{x}) &= \int_{\mathbf{y}} \exp\Big\{-E(\mathbf{y}, \mathbf{x})\Big\} \mathrm{d}\mathbf{y} \\ &= \exp\{-\mathbf{z}^\top \mathbf{z}\} \int_{\mathbf{y}} \exp\Big\{-\mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y}\Big\} \mathrm{d}\mathbf{y} \\ &= \frac{(\pi)^{\frac{n}{2}}}{|\mathbf{A}|^{\frac{1}{2}}} \exp\{\mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} - \mathbf{z}^\top \mathbf{z}\}, \end{aligned} \tag{11}$$

From Eqs. (1), (10), (11), we can now write the probability distribution function as:

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{x})) = \frac{\exp\Big\{-\mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} - \mathbf{z}^\top \mathbf{z}\Big\}}{\frac{(\pi)^{\frac{n}{2}}}{|\mathbf{A}|^{\frac{1}{2}}} \exp\{\mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} - \mathbf{z}^\top \mathbf{z}\}} \\ &= \frac{|\mathbf{A}|^{\frac{1}{2}}}{\pi^{\frac{n}{2}}} \exp\Big\{-\mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} - \mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z}\Big\}, \end{aligned} \tag{12}$$

where $\mathbf{z} = [z_1, \ldots, z_n]^\top$; $|\cdot|$ denotes the determinant of a matrix, and $\mathbf{A}^{-1}$ the inverse of $\mathbf{A}$. Then the negative log-likelihood can be written as:

$$\begin{aligned} &-\log \Pr(\mathbf{y}|\mathbf{x}) \\ &= \mathbf{y}^\top \mathbf{A} \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} - \frac{1}{2} \log(|\mathbf{A}|) + \frac{n}{2} \log(\pi). \end{aligned} \tag{13}$$

During learning, we minimize the negative conditional log-likelihood of the training data. Adding regularization to $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, we then arrive at the final optimization:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\beta} \geq 0} \frac{\lambda_1}{2} \|\boldsymbol{\theta}\|_2^2 + \frac{\lambda_2}{2} \|\boldsymbol{\beta}\|_2^2 - \sum_{i=1}^{N} \log \Pr(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}, \boldsymbol{\beta}), \tag{14}$$

where $\mathbf{x}^{(i)}$, $\mathbf{y}^{(i)}$ denote the ith training image and the corresponding depth map; N is the number of training images; λ_1 and λ_2 are weight decay parameters.

2.3.1 Optimization

We use stochastic gradient descent (SGD) based back propagation to solve the optimization problem in Eq. (14) for learning all parameters of the whole network. We project the solutions to the feasible set when the bounded constraints $\beta_k \geq 0$ is violated. In the following, we calculate the

partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to the network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.

For the unary part, here we calculate the partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to θ_l (one element of $\boldsymbol{\theta}$). Recall that $\mathbf{A} = \mathbf{I} + \mathbf{D} - \mathbf{R}$ (Eq. (9)); $\mathbf{A}^\top = \mathbf{A}$; $(\mathbf{A}^{-1})^\top = \mathbf{A}^{-1}$; $|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}$, we have:

$$\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \theta_l} = \frac{\partial\{-2\mathbf{z}^\top\mathbf{y} + \mathbf{z}^\top\mathbf{A}^{-1}\mathbf{z}\}}{\partial \theta_l} \tag{15}$$

$$= 2(\mathbf{A}^{-1}\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial \theta_l}. \tag{16}$$

For the pairwise part, we calculate the partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to β_k as:

$$\begin{aligned}
&\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \beta_k} \\
&= \frac{\partial\{\mathbf{y}^\top\mathbf{A}\mathbf{y} + \mathbf{z}^\top\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\log(|\mathbf{A}|)\}}{\partial \beta_k} \\
&= \mathbf{y}^\top \frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{y} - \mathbf{z}^\top\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\frac{1}{|\mathbf{A}|}\frac{\partial\{|\mathbf{A}|\}}{\partial \beta_k}, \\
&= \mathbf{y}^\top \frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{y} - \mathbf{z}^\top\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\right).
\end{aligned} \tag{17}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix. We here introduce matrix $\mathbf{J}$ to denote $\frac{\partial \mathbf{A}}{\partial \beta_k}$. Each element of $\mathbf{J}$ is:

$$\begin{aligned}
J_{pq} &= \frac{\partial A_{pq}}{\partial \beta_k} \\
&= \frac{\partial\{D_{pq} - R_{pq}\}}{\partial \beta_k} \\
&= -\frac{\partial R_{pq}}{\partial \beta_k} + \delta(p = q)\sum_q \frac{\partial R_{pq}}{\partial \beta_k},
\end{aligned} \tag{18}$$

where $\delta(\cdot)$ is the indicator function, which equals 1 if $p = q$ is true and 0 otherwise. According to Eq. (17) and the definition of $\mathbf{J}$ in (18), we can now write the partial derivative of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to β_k as:

$$\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \beta_k} = \mathbf{y}^\top\mathbf{J}\mathbf{y} - \mathbf{z}^\top\mathbf{A}^{-1}\mathbf{J}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{-1}\mathbf{J}\right). \tag{19}$$

From Eqs. (19), (18), we can see that our framework is general and more complicated networks for the pairwise part can be seamlessly incorporated. Here, in our case, with the definition of R_{pq} in Eq. (7), we have $\frac{\partial R_{pq}}{\partial \beta_k} = S_{pq}^{(k)}$.

2.3.2 Depth Prediction

Predicting the depths of a new image is to solve the MAP inference in Eq. (3), which writes as:

$$\begin{aligned}
\mathbf{y}^\star &= \underset{\mathbf{y}}{\mathrm{argmax}} \log \Pr(\mathbf{y}|\mathbf{x}) \\
&= \underset{\mathbf{y}}{\mathrm{argmax}} -\mathbf{y}^\top\mathbf{A}\mathbf{y} + 2\mathbf{z}^\top\mathbf{y}.
\end{aligned} \tag{20}$$

With the definition of $\mathbf{A}$ in Eq. (9), $\mathbf{A}$ is symmetric. Then by setting the partial derivative of $-\mathbf{y}^\top\mathbf{A}\mathbf{y} + 2\mathbf{z}^\top\mathbf{y}$ with respect to $\mathbf{y}$ to $\mathbf{0}$, we have

$$\begin{aligned}
&\frac{\partial\{-\mathbf{y}^\top\mathbf{A}\mathbf{y} + 2\mathbf{z}^\top\mathbf{y}\}}{\partial \mathbf{y}} = \mathbf{0} \\
&\Rightarrow -(\mathbf{A} + \mathbf{A}^\top)\mathbf{y} + 2\mathbf{z} = \mathbf{0} \\
&\Rightarrow \mathbf{y} = \mathbf{A}^{-1}\mathbf{z}.
\end{aligned} \tag{21}$$

It shows that closed-form solutions exist for the MAP inference in Eq. (20):

$$\mathbf{y}^\star = \mathbf{A}^{-1}\mathbf{z}. \tag{22}$$

If we discard the pairwise terms, namely $R_{pq} = 0$, then Eq. (22) degenerates to $\mathbf{y}^\star = \mathbf{z}$, which is a plain CNN regression model (we will report the results of this method as a baseline in the experiment). Note that we do not need to explicitly compute the matrix inverse $\mathbf{A}^{-1}$ which can be expensive (cubic in the number of nodes). Instead we can obtain the value of $\mathbf{A}^{-1}\mathbf{z}$ by solving a linear system.

2.4 Speeding Up Training Using Fully Convolutional Networks and Superpixel Pooling

Thus far, we have presented our DCNF model for depth estimations based on image superpixels. From Fig. 2, we can see that for constructing the unary potentials, we are essentially performing patchwise convolutions (similar operations are performed in the R-CNN [21]). A major concern of the proposed method is its computational efficiency and memory consumption, since we need to perform convolutions over hundreds or even thousands (number of superpixels) of image patches for a single input image. Many of those convolutions are redundant due to significant image patch overlaps.

Naturally, a promising direction for reducing the computation burden is to perform convolutions over the entire image once, and then obtain convolutional features for each superpixel. However, to find the convolutional features of the image superpixels from the obtained convolution maps, one needs to establish associations between these two. Therefore, we here propose an improved model, which we term as DCNF-FCSP (DCNF with Fully Convolutional networks and Superpixel Pooling), based on fully convolutional networks and a novel superpixel pooling method, to address this issue. As we will show in Section 3.2, this new model significantly speeds up the training and prediction while producing almost the same prediction accuracy. Most importantly, with this more efficient model, we are able to design deeper networks to achieve better performance.

2.4.1 DCNF-FCSP Overview

In comparison to the DCNF model, our new DCNF-FCSP model mainly improves the unary part while keeping the same pairwise network architecture as in Fig. 2. We show the model architecture of the unary part in Fig. 4. Specifically, the input image is fed to a fully convolutional network (introduced in the sequel). The outputs are convolutional feature maps of size $h \times w \times d$ (d is the dimension of the obtained convolutional feature vector, i.e., the number of channels of the last convolutional layer). The size of the convolution maps $h \times w$ are typically smaller than the input image size. Each convolutional feature vector in the output convolution maps corresponds to a patch in the input

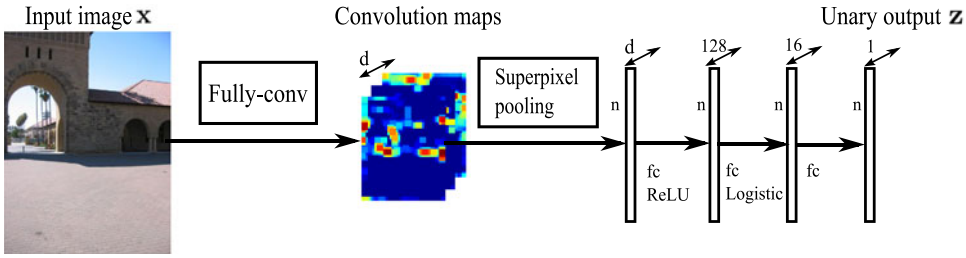


Fig. 4. An overview of the unary part of the DCNF-FCSP model. For the unary part, the input image is fed into a fully-convolutional network to produce convolution maps (d is the number of filters of the last fully-convolutional layer). The obtained convolution maps, together with the superpixel segmentation over the original input image, are fed to a superpixel pooling layer. The outputs are $n \times 1$ d dimensional feature vectors for each of the n superpixels, which are then followed by three fully-connected layers to produce the unary output $\mathbf{z}$. The pairwise part are omitted here since we use the same network architecture as in the DCNF model (Fig. 2). The unary output $\mathbf{z}$ and the pairwise output $\mathbf{R}$ are used as input to the CRF loss layer, which minimizes the negative log-likelihood (See Section 2.4 for details).

image. We propose a novel superpixel pooling method (described in the following) to associate these outputs back to the superpixels in the input image. Specifically, the convolutional feature maps are used as inputs to a superpixel pooling layer to obtain n superpixel feature vectors with d dimensions (n is the number of superpixels). The n superpixel feature vectors are then fed to three fully-connected layers to produce the unary output $\mathbf{z}$. We use the same pairwise network architecture as depicted in Fig. 2, which we do not show here. With the unary output $\mathbf{z}$ and the pairwise output $\mathbf{R}$, we construct potential functions according to Eqs. (5) and (6) and optimize the negative log-likelihood.

Compared to the original proposed DCNF method, this improved DCNF-FCSP model only needs to perform convolutions over the entire image once, rather than hundreds of superpixel image patches. This significantly reduces the computation and GPU memory burden, bringing around 10 times training speedup while producing almost the same prediction accuracy, as we demonstrate later in Section 3.2. We next introduce the fully convolutional networks and the superpixel pooling method in detail.

2.4.2 Fully Convolutional Networks

Typical CNN models, including AlexNet [15], vggNet [20], [34], etc., are composed of convolution layers and fully connected layers. They usually take standard sized images as inputs, e.g., 224×224 pixels, to produce nonspatial outputs. In contrast, a fully convolutional network can take as inputs arbitrarily sized images, and outputs convolutional spatial maps. It has therefore been actively studied for dense prediction problems [3], [17], [18], [31] very recently. We here exploit this new development trend in CNN to speedup the patch-wise convolutions in the DCNF model. We illustrate the fully convolutional network architecture that we use in Fig. 5. As shown, the network is composed of seven convolution layers, with the first five layers transferred from the AlexNet [15]. We then add two more convolution layers with 3×3 filter size and 512 channels each. The network takes as input images of arbitrary size and outputs convolution maps of channel $d = 512$. Note that we can design deeper networks here for pursuing better performance. We will demonstrate in Section 3.3 the benefits of using deeper models.

2.4.3 Superpixel Pooling

After the input images go through the fully convolutional networks, we acquire convolution maps. To obtain superpixel features, we need to associate these convolutional feature maps back to the image superpixels. Thus we here propose a novel superpixel pooling method. An illustration of this method is shown in Fig. 6, which mainly consists of the convolution maps upsampling and superpixel average pooling. In general, this superpixel pooling layer takes the convolution maps as input and outputs superpixel features. Specifically, the obtained convolution maps are first upsampled to the original image size by nearest neighbor interpolation. Note that other interpolation methods such as linear interpolation may be applicable too. However, as discussed below, nearest neighbor interpolation makes the implementation much easier and computation faster.

Then the superpixel masking is applied and average pooling is performed within each superpixel region. The outputs are pooled superpixel features, which are used for constructing the unary potentials. In the sequel, we describe this method in detail.

In practice there is no need to explicitly upsample the convolution maps. Instead, we count the frequencies of the convolutional feature vectors that fall into each superpixel region. We denote the convolution maps as $\mathbf{C} \in \mathbb{R}^{h \times w \times d}$, with each element being C_{ijk} $(i = 1, \ldots, h; j = 1, \ldots, w; k = 1, \ldots, d)$. We represent the tth superpixel feature as a d-dimensional column vector $\mathbf{h}_t$ $(t = 1, \ldots, n)$, with elements h_{tk} $(k = 1, \ldots, d)$. $\mathbf{W}_t \in \mathbb{R}^{h \times w}$ is a frequency weighting matrix associated to the tth superpixel, with elements being W_{ijt}. W_{ijt} represents the weight of (i, j)th feature vector in

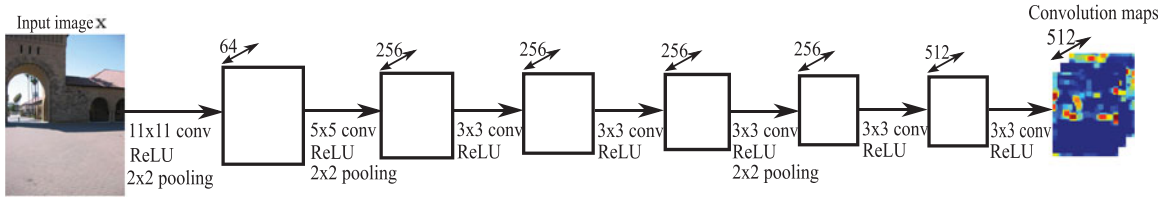


Fig. 5. The fully convolutional network architecture used in Fig. 4. The network takes input images of arbitrary size and output convolution maps.

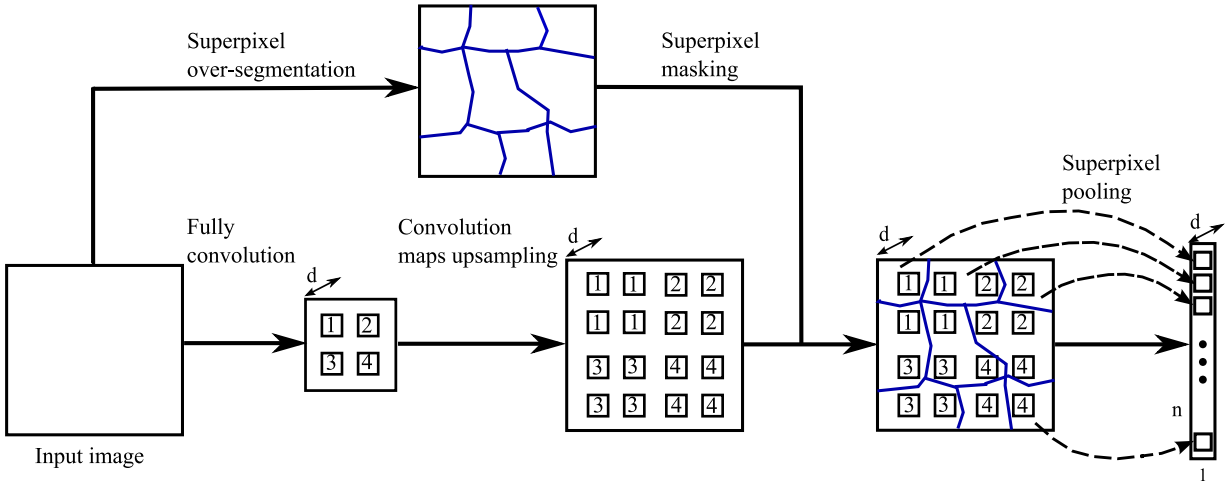


Fig. 6. An illustration of the superpixel pooling method, which mainly consists of convolution maps upsampling and superpixel pooling. The convolution maps are upsampled to the original image size by nearest neighbor interpolations, over which the superpixel masking is applied. Then average pooling is performed within each superpixel region, to produce the n convolution features. n is the number of superpixels in the image. d is the number of channels of the convolution maps.

the convolution maps that associated to the tth superpixel. To calculate W_{ijt}, we simply count the occurrences of the (i, j)th convolutional feature vector that appear in the tth superpixel region, and do L_1 normalization for each $\mathbf{W}_t$. By constructing this frequency matrix $\mathbf{W}_t$, we avoid the explicit upsampling operation. Then the superpixel pooling can be represented as:

$$h_{tk} = \sum_{(i,j)\in\mathcal{R}_t} W_{ijt} \cdot C_{ijk}, \tag{23}$$

where $(i, j) \in \mathcal{R}_t$ denotes the (i, j)th convolutional feature vector in the convolution maps being associated to the tth superpixel.

During the network forward pass, the superpixel pooling layer performs a linear transformation in Eq. (23) to output $\mathbf{h}_t$ from the input $\mathbf{C}$. For the network backward, the gradients can be easily calculated since we have:

$$\frac{\partial h_{tk}}{\partial C_{ijk}} = \begin{cases} W_{ijt} & \text{if } (i,j) \in \mathcal{R}_t \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

Thus far, we have successfully established the associations between the convolutional feature maps and the image superpixels. It should be noted that although simple as it is, the proposed superpixel pooling method jointly exploits the benefits of fully convolutional networks and superpixels. It provides an efficient yet equally effective approach to the patchwise convolutions used in the DCNF model, as we demonstrate in Section 3.2.

2.5 Implementation Details

We implement the network training based on the CNN toolbox: VLFeat MatConvNet[1] with our own modifications. Training is done on a standard desktop with an NVIDIA GTX 780 GPU with 6 GB memory. The DCNF model, with the network design in Fig. 3, has approximately 40 million parameters. The DCNF-FCSP model, with the network design in Fig. 5, has around 5.8 million parameters. With the very deep network architecture, the DCNF-FCSP model has around 20 million parameters. The huge amounts of parameters for the DCNF model mainly comes from the 4,096 fully connected layer in Fig. 3, which is removed in the DCNF-FCSP model. Next we present the implementation details of the two proposed models in the following.

DCNF We initialize the first six layers of the unary part in Fig. 3 using a CNN model trained on the ImageNet from [34]. First, we do not back propagate through the previous six layers by keeping them fixed and train the rest of the network (we refer this process as pre-train) with the following settings: momentum is set to 0.9, and weight decay parameters λ_1, λ_2 are set to 0.0005. During pre-train, the learning rate is initialized at 0.0001, and decreased by 40 percent every 20 epoches. We then run 60 epoches to report the results of pre-train (with learning rate decreased twice). During pre-training, it takes less than 0.1 s for one network forward pass to do depth predictions. Then we train the whole network with the same momentum and weight decay. Training the whole network takes around 16.5 hours on the Make3D dataset, and around 33 hours on the NYU v2 dataset. For this fine-tune model, it takes ~ 1.1 s for one network forward pass to do depth predictions.

DCNF-FCSP We initialize the first five layers in Fig. 5 with the same model trained on the ImageNet from [34]. The momentum and weight decay parameters are set the same as in the DCNF model. We also use the same training protocol as in the DCNF model, i.e., first pre-train and then fine-tune the whole model.

3 Experiments

We organize our experiments into the following three parts: 1) We compare our DCNF model with several baseline methods to show the benefits of jointly learning CNN and CRF; 2) We perform comparisons between the two proposed models, i.e., DCNF and DCNF-FCSP, to show that the DCNF-FCSP model is equally effective while generally being ~ 10 times faster; 3) We compare our DCNF-FCSP model using deeper network design with state-of-the-art methods to show that our model performs significantly

1. VLFeat MatConvNet: http://www.vlfeat.org/matconvnet/

TABLE 1
Baseline Comparisons on the NYU v2 Dataset

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
SVR	0.313	0.128	1.068	0.490	0.787	0.921
SVR (smooth)	0.290	0.116	0.993	0.514	0.821	0.943
Unary only	0.295	0.117	0.985	0.516	0.815	0.938
Unary only (smooth)	0.287	0.112	0.956	0.535	0.828	0.943
DCNF (pre-train)	0.257	0.101	0.843	0.588	0.868	0.961
DCNF (fine-tune)	**0.230**	**0.095**	**0.824**	**0.614**	**0.883**	**0.971**

Our method with the whole network training performs the best.

better. We evaluate on three popular datasets which are available online: the indoor NYU v2 Kinect dataset [35], the outdoor Make3D range image dataset [23] and the KITTI dataset [36]. Several measures commonly used in prior works are applied here for quantitative evaluations:

- average relative error (rel): $\frac{1}{T}\sum_p \frac{|d_p^{gt} - d_p|}{d_p^{gt}}$;
- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_p (d_p^{gt} - d_p)^2}$;
- average $\log_{10}$ error (log10): $\frac{1}{T}\sum_p |\log_{10} d_p^{gt} - \log_{10} d_p|$;
- accuracy with threshold thr: percentage (%) of d_p s.t. : $\max(\frac{d_p^{gt}}{d_p}, \frac{d_p}{d_p^{gt}}) = \delta < thr$;

where d_p^{gt} and d_p are the ground-truth and predicted depths respectively at pixel indexed by p, and T is the total number of pixels in all the evaluated images.

We use SLIC [37] to segment the images into a set of non-overlapping superpixels. For the DCNF model, we consider the image patch within a rectangular box centred on the centroid of each of the superpixels, which contains a large portion of its background surroundings. More specifically, we use a box size of 168×168 pixels for the NYU v2 and 120×120 pixels for the Make3D dataset. Following [3], [9], [23], we transform the depths into log-scale before training. For better visualizations, we apply a cross-bilateral filter [38] for inpainting using the provided toolbox [35] after obtaining the superpixel depth predictions. Our experiments empirically show that this post-processing has negligibly impact on the evaluation performance.

3.1 Baseline Comparisons

To demonstrate the effectiveness of the proposed method, we first conduct experimental comparisons against several baseline methods:

- SVR: We train a support vector regressor using the CNN representations from the first six layers of Fig. 3;
- SVR (smooth): We add a smoothness term to the trained SVR during prediction by solving the inference problem in Eq. (22). As tuning multiple pairwise parameters is not straightforward, we only use color difference as the pairwise potential and choose the parameter β by hand-tuning on a validation set;
- Unary only: We replace the CRF loss layer in Fig. 2 with a least-square regression layer (by setting the pairwise outputs $R_{pq} = 0, p, q = 1, ..., n$), which degenerates to a deep regression model trained by SGD.
- Unary only (smooth): We add similar smoothness term to our unary only model, as did in the SVR (smooth) case.

3.1.1 NYU v2 Data

The NYU v2 dataset consists of $1{,}449$ RGBD images of indoor scenes, among which 795 are used for training and 654 for test (we use the standard training/test split provided with the dataset). We report the baseline comparisons in Table 1. From the table, several conclusions can be made: 1) When trained with only unary term, deeper network is beneficial for better performance, which is demonstrated by the fact that our unary only model outperforms the SVR model; 2) Adding smoothness term to the SVR or our unary only model helps improve the prediction accuracy; 3) Our DCNF model achieves the best performance by jointly learning the unary and the pairwise parameters in a unified deep CNN framework. Moreover, fine-tuning the whole network yields further performance gain. These well demonstrate the efficacy of our model.

3.1.2 Make3D Data

The Make3D dataset contains 534 images depicting outdoor scenes, with 400 for training and 134 images for test. As pointed out in [23], [24], this dataset is with limitations: the maximum value of depths is 81 m with far-away objects are all mapped to the one distance of 81 meters. As a remedy, two criteria are used in [24] to report the prediction errors: (C_1) Errors are calculated only in the regions with the ground-truth depth less than 70 meters; (C_2) Errors are calculated over the entire image. We follow this protocol to report the evaluation results in Table 2. As we can see, our full DCNF model with the whole network training performs the best among all the compared baseline methods. Using deeper networks and adding smoothness terms generally help improve the performance.

3.2 DCNF versus DCNF-FCSP

In this section, we compare the performance of the proposed DCNF and DCNF-FCSP in terms of both prediction accuracy and computational efficiency. The compared prediction performance are reported in Tables 3 and 4. We can see that the proposed DCNF-FCSP model performs very close to the DCNF model. Next, we compare the

TABLE 2
Baseline Comparisons on the Make3D Dataset

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
SVR	0.433	0.158	8.93	0.429	0.170	15.29
SVR (smooth)	0.380	0.140	**8.12**	0.384	0.155	15.10
Unary only	0.366	0.137	8.63	0.363	0.148	14.41
Unary only (smooth)	0.341	0.131	8.49	0.349	0.144	14.37
DCNF (pre-train)	0.331	0.127	8.82	0.324	0.134	13.29
DCNF (fine-tune)	**0.314**	**0.119**	8.60	**0.307**	**0.125**	**12.89**

Our method with the whole network training performs the best.

TABLE 3
Performance Comparisons of DCNF and DCNF-FCSP on the NYU v2 Dataset

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
DCNF (pre-train)	0.257	0.101	0.843	0.588	0.868	0.961
DCNF (fine-tune)	0.230	0.095	0.824	**0.614**	0.883	**0.971**
DCNF-FCSP (pre-train)	0.261	0.100	0.842	0.583	0.869	0.964
DCNF-FCSP (fine-tune)	0.237	**0.082**	**0.822**	0.608	**0.889**	0.969

The two models show comparable performance.

TABLE 4
Performance Comparisons of DCNF and DCNF-FCSP on the Make3D Dataset

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
DCNF (pre-train)	0.331	0.127	8.82	0.324	0.134	13.29
DCNF (fine-tune)	0.314	0.119	**8.60**	0.307	0.125	**12.89**
DCNF-FCSP (pre-train)	0.323	0.127	9.01	0.318	0.136	13.89
DCNF-FCSP (fine-tune)	**0.312**	**0.113**	9.10	**0.305**	**0.120**	13.24

The two models perform on par in general.

computational efficiency of these two models. Specifically, we report the training time (network forward + backward) of one image for both whole models in terms of different numbers of superpixels we use per image. The comparison is conducted on the NYU v2 dataset and is shown in Fig. 11. As demonstrated, the DCNF-FCSP model is generally orders of magnitude faster than the DCNF model. Moreover, the speedup becomes more significant with the number of superpixels increases. We also compare the network forward time of whole models during depth predictions, and plot the results in Fig. 12. The shown time is for processing one image. We can see that the DCNF-FCSP model is much faster as well as more scalable than the DCNF model. Most importantly, with this more efficient DCNF-FCSP model, we can design deeper network for better performance, as we will show in the sequel in Section 3.3.

3.3 State-of-the-Art Comparisons

Recent studies have shown that very deep networks can significantly improve the image classifications performance [20], [39]. Thanks to the speedup brought about by the superpixel pooling method, we are now able to design deeper networks in our framework. We transfer the popular VGG-16 net trained on the ImageNet from [20]. Specifically, we replace the AlexNet part (the first first convolutional layers in Fig. 5) with all the convolutional layers (including the fifth pooling layer) in VGG-16. These layers are followed by the two newly added convolutional layers with 512 channels each, and then the three fully connected layer to construct the unary potentials. We follow the same training protocol, i.e., first pre-train the remaining layers by fixing the transferred layers (the VGG-16 net part) and then fine-tune the whole network.

3.3.1 NYU v2 Data

In Table 5, we report the results compared to several popular state-of-the-art methods on the NYU v2 dataset. As can be observed, our method outperforms classic methods like Make3d [23], DepthTransfer [7] with large margins. Most notably, our results are significantly better than that of [1], which jointly exploits depth estimation and semantic labelling. Compared to the recent work of Eigen et al. [3], our method also exhibits better performance in terms of all metrics. Note that, to overcome overfit, they [3] have to collect

TABLE 5
State-of-the-Art Comparisons on the NYU v2 Dataset

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
DepthTransfer [7]	0.35	0.131	1.2	-	-	-
Discrete-continuous CRF [24]	0.335	0.127	1.06	-	-	-
Ladicky et al. [1]	-	-	-	0.542	0.829	0.941
Eigen et al. [3]	0.215	-	0.907	0.611	0.887	0.971
DCNF-FCSP (pre-train)	0.234	0.095	0.842	0.604	0.885	0.973
DCNF-FCSP (fine-tune)	**0.213**	**0.087**	**0.759**	**0.650**	**0.906**	**0.976**

Our method performs the best in most cases. Note that the results of Eigen et al. [3] are obtained by using extra training data (in the millions in total) while ours are obtained using the standard training set.

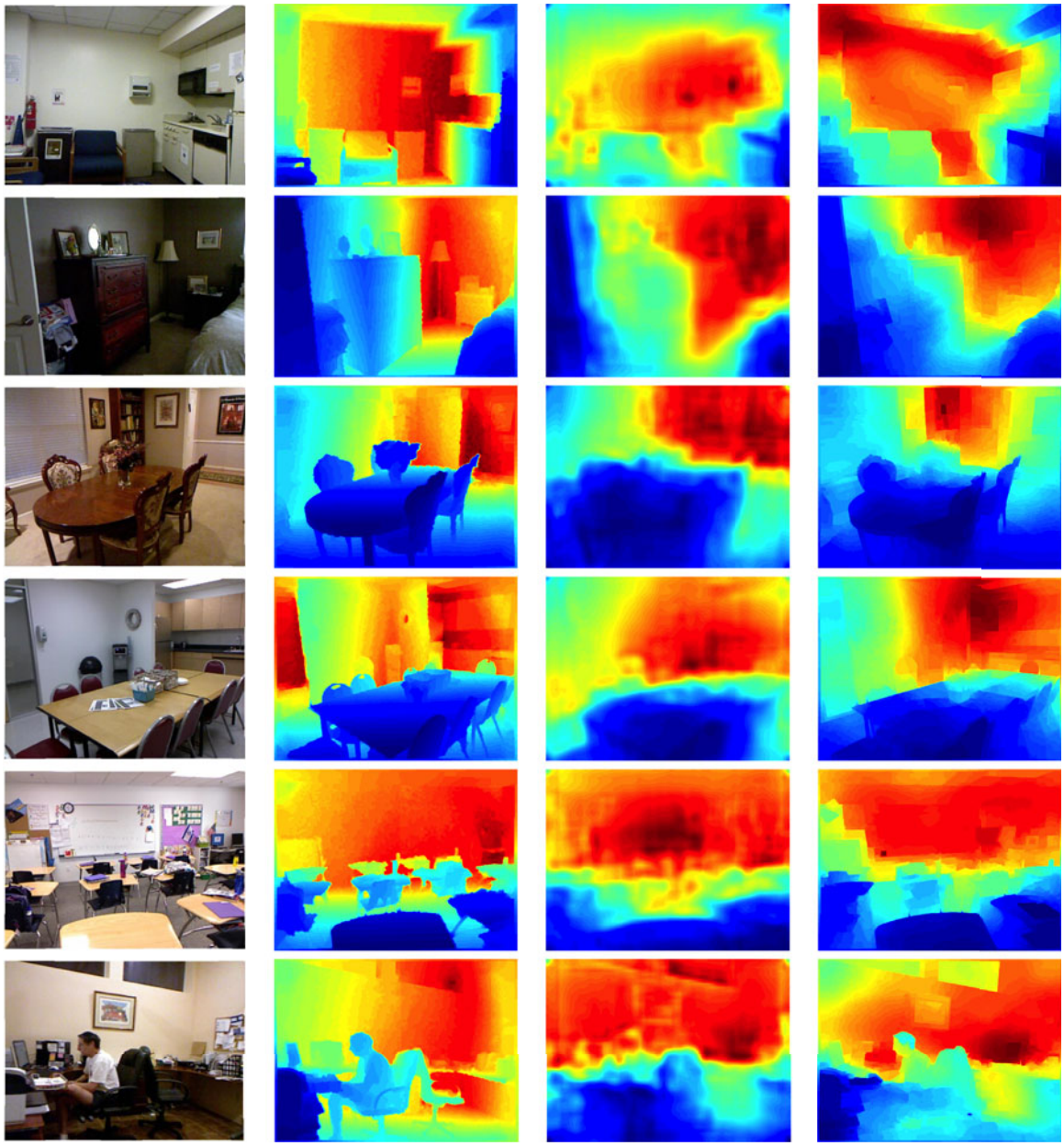


Fig. 7. Examples of qualitative comparisons on the NYUD2 dataset (Best viewed on screen). Color indicates depths (red is far, blue is close). Our method yields visually better predictions with sharper transitions, aligning to local details.

TABLE 6
State-of-the-Art Comparisons on the Make3D Dataset

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
Saxena et al. [23]	-	-	-	0.370	0.187	-
Semantic Labelling [9]	-	-	-	0.379	0.148	-
DepthTransfer [7]	0.355	0.127	9.20	0.361	0.148	15.10
Discrete-continuous CRF [24]	0.335	0.137	9.49	0.338	0.134	**12.60**
DCNF-FCSP (pre-train)	0.331	0.119	7.77	0.330	0.133	14.46
DCNF-FCSP (fine-tune)	**0.287**	**0.109**	**7.36**	**0.287**	**0.122**	14.09

Our method performs the best. Note that the C2 errors of the Discrete-continuous CRF [24] are reported with an ad-hoc post-processing step (train a classifier to label sky pixels and set the corresponding regions to the maximum depth).

millions of additional labelled images to train their model. In contrast, we only use the standard training sets (795) without any extra data, yet we achieve better performance. Fig. 7 illustrates some qualitative evaluations of our method compared against Eigen et al. [3] (We download the predictions of [3] from the authors' website.). Compared to the coarse predictions of [3], our method yields better visualizations with sharper transitions, aligning to local details. To better illustrate how our predictions deviate from the ground-truth depths, we plot the absolute depth error maps and the pixel-wise error histograms in Fig. 10. Specifically, the absolute error maps are shown in meters, with the color bar shown in the last row. For the error histogram plot, the horizontal axis shows the prediction error in meters (quantized into 20 bins), and the vertical axis shows the percentage of pixels in each bin. As we can see, our predictions are mostly well aligned to the ground truth depth maps.

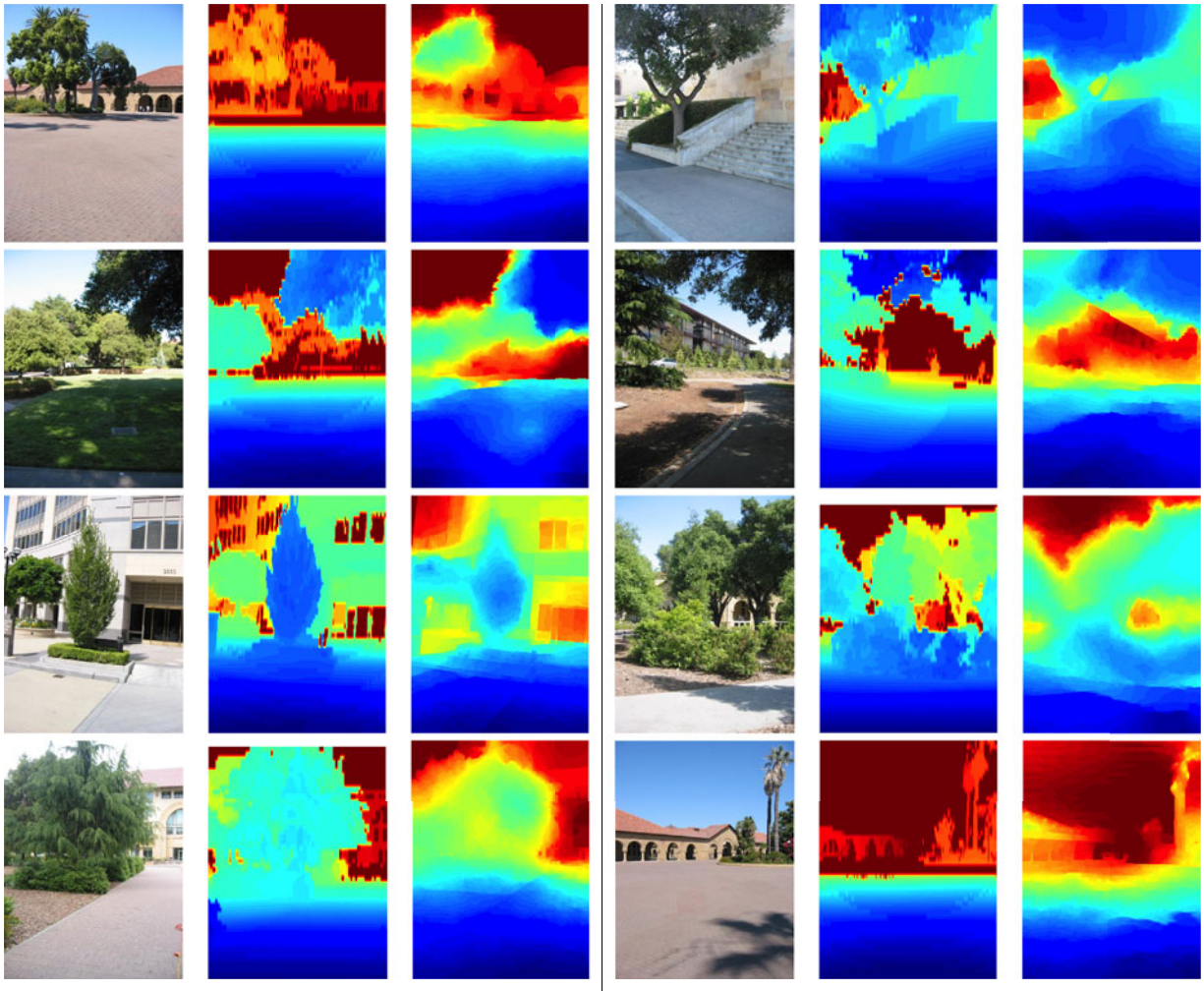


Fig. 8. Examples of depth predictions on the Make3D dataset (Best viewed on screen). Depths are shown in log scale and in color (red is far, blue is close).

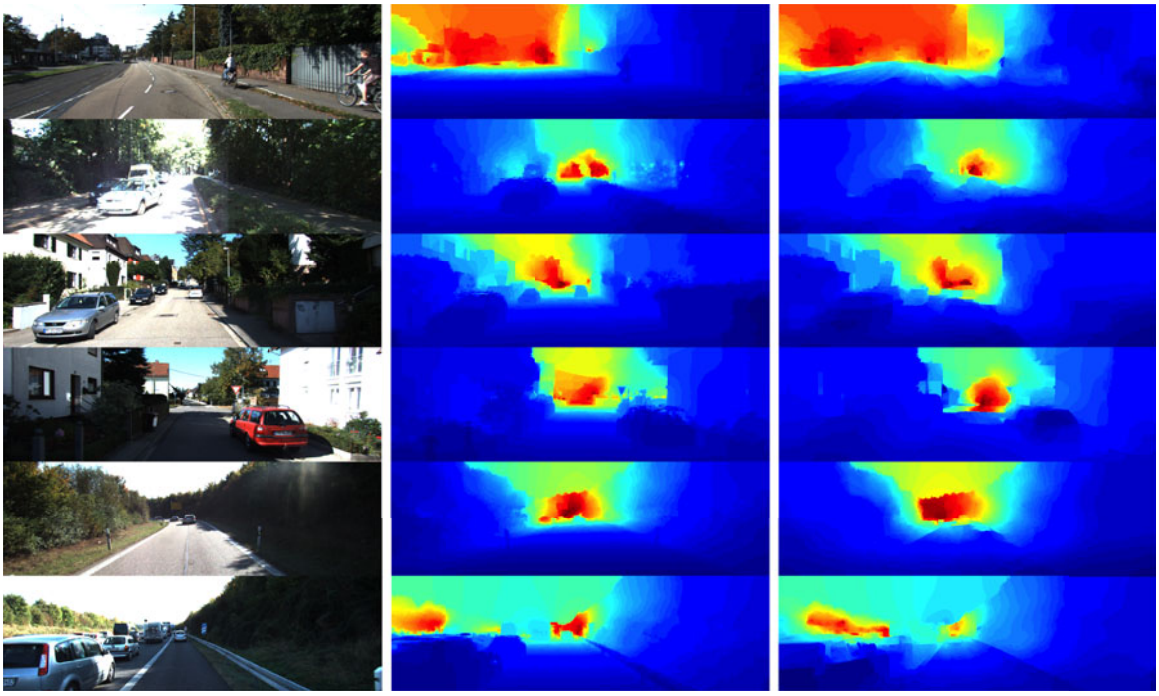


Fig. 9. Examples of depth predictions on the KITTI dataset (Best viewed on screen). Depths are shown in log scale and in color (red is far, blue is close).

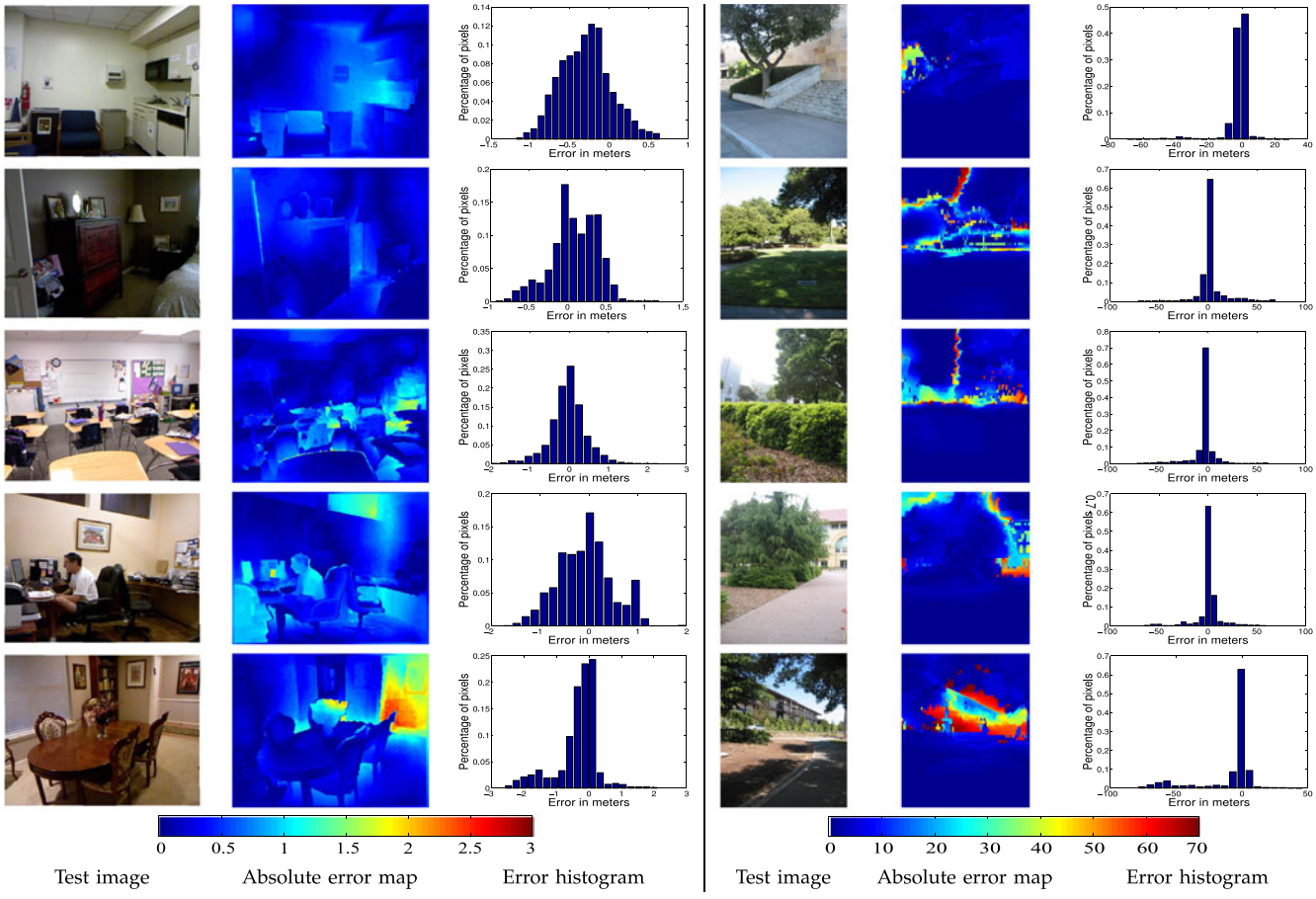


Fig. 10. An illustration of the absolute error maps and the pixel-wise error histograms of our predictions (Left: NYU v2; Right: Make3D). The absolute error maps are shown in meters, with the color bar shown in the last row. For the error histogram plot, the horizontal axis shows the prediction error in meters (quantized into 20 bins), and the vertical axis shows the percentage of pixels in each bin.

TABLE 7
State-of-the-Art Comparisons on the KITTI Dataset

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]	0.280	-	8.734	0.601	0.820	0.926
Eigen et al. [3]	**0.190**	-	7.156	**0.692**	**0.899**	**0.967**
DCNF-FCSP (pre-train)	0.236	0.101	7.421	0.613	0.858	0.949
DCNF-FCSP (fine-tune)	0.217	**0.092**	**7.046**	0.656	0.881	0.958

Our method achieves the best RMS error. Note that the results of Eigen et al. [3] are obtained by using extra training data (in the millions in total) while ours are obtained using 700 training images. The results of Saxena et al. [23] are reproduced from [3].

3.3.2 Make3D Data

We show the compared results on the Make3D dataset in Table 6. We can see that our DCNF-FCSP model with the whole network training ranks the first in overall performance, outperforming the compared methods by large margins. Note that the C2 errors of [24] are reported with an ad-hoc post-processing step, which trains a classifier to label sky pixels and set the corresponding regions to the maximum depth. In contrast, we do not employ any of those heuristics to refine our results, yet we achieve better results in terms of relative error. Compared to the results of the DCNF-FCSP model using smaller net in Table 4, we get better $C1$ error but degraded $C2$ error. This can be explained from the limitations of this dataset that the depths of all far away objects are all set to one maximum depth value. Some examples of qualitative evaluations are shown in Fig. 8. By jointly learning the unary and pairwise potentials, our DCNF-FCSP model produce predictions that well capture local details. The absolute depth error maps and the pixel-wise error histograms are shown in the right part of Fig. 10. As can be observed, our predictions are mostly well aligned to the ground truth depth maps, with most of the prediction errors are on the boundary regions which show extremely large depth jumps. In these cases, our predictions exhibit relatively mild depth transitions.

3.3.3 KITTI Data

We further perform depth estimation on the KITTI dataset [36], which consists of videos taken from a driving vehicle with depths captured by a LiDAR sensor. We use the same test set, i.e., 697 images from 28 scenes, as provided by Eigen [3]. As for the training set, we use the same 700 images that Eigen et al. [3] used to train the method of Saxena et al. [23]. Since the ground-truth depths of the KITTI dataset are scattered at irregularly spaced points, which only consists of ~ 5 percent pixels of each image, we extract the ground-truth depth closest to each superpixel centroid as the superpixel depth label. We then construct our CRF graph only on those superpixels that have ground-truth labels. The compared results are presented in Table 7. In summary, Eigen et al. [3] have achieved slightly better performance on this dataset by leveraging large amounts of training data (in the millions). This can be explained by the fact that the highly sparse ground-truth depth maps lessened the benefits of the pairwise term in our model. Fig. 9 shows some prediction examples.

4 Conclusion

We have presented a deep convolutional neural field model for depth estimation from a single image. The proposed method combines the strength of deep CNN and continuous CRF in a unified CNN framework. We show that the log-likelihood optimization in our method can be directly solved using back propagation without any approximations required. Predicting the depths of a new image by solving the MAP inference can be efficiently performed as closed-form solutions exist. We further propose an improved model that is based on fully convolutional networks and a novel superpixel pooling method. We experimentally demonstrate

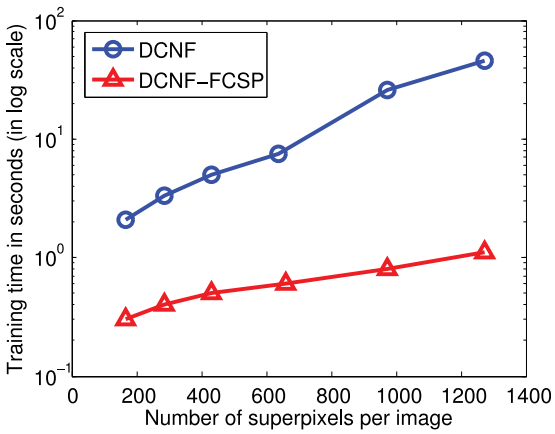


Fig. 11. Comparison of the whole model training time (network forward + backward) in seconds (in log scale) for one image on the NYU v2 dataset with respect to different numbers of superpixels per image. The DCNF-FCSP model is orders of magnitude faster than the DCNF model.

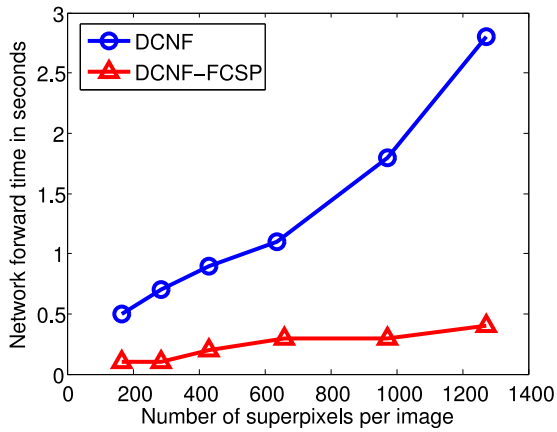


Fig. 12. Comparison of the network forward time of the whole model during depth prediction (in seconds) for one image on the NYU v2 dataset with respect to different numbers of superpixels per image. The DCNF-FCSP model is significantly faster than the DCNF model.

that it is equally effective while brings orders of magnitude faster training speedup, which enables the use of deeper networks for better performance. Experimental results demonstrate that the proposed method outperforms state-of-the-art methods on both indoor and outdoor scene datasets.

The main limitation of our method is that it does not exploit any geometric cues, which can be explored in the future work. Given the general learning framework of our method, it is also possible to be applied to other vision applications with minimum modification, e.g., image denoising, and deblurring. Another potential working direction is to apply our depth estimation models for benefiting other vision tasks, e.g., semantic segmentation, object detection, on traditional vision datasets where ground-truth depths are not available.

Acknowledgments

C. Shen's participation was in part support by ARC Future Fellowship. The authors would like to thank NVIDIA for GPU donation. C. Shen is the corresponding author.

References

[1] L. Ladick, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 89–96.

[2] J. Shotton, R. B. Girshick, A. W. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2821–2840, Dec. 2013.

[3] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2366–2374.

[4] V. Hedau, D. Hoiem, and D. A. Forsyth, "Thinking inside the box: Using appearance models and context based on room geometry," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 2240–237.

[5] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1288–1296.

[6] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 482–496.

[7] K. Karsch, C. Liu, and S. B. Kang, "Depthtransfer: Depth extraction from video using non-parametric sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2144–2158, Nov. 2014.

[8] B. C. Russell and A. Torralba, "Building a database of 3d scenes from user annotations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 2711–2718.

[9] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 1253–1260.

[10] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 282–289.

[11] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li, "Global ranking using continuous conditional random fields," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1281–1288.

[12] V. Radosavljevic, S. Vucetic, and Z. Obradovic, "Continuous conditional random fields for regression in remote sensing," in *Proc. Eur. Conf. Artif. Intell.*, 2010, pp. 809–814.

[13] K. Ristovski, V. Radosavljevic, S. Vucetic, and Z. Obradovic, "Continuous conditional random fields for efficient regression in large fully connected graphs," in *Proc. AAAI Conf. Artif. Intell.*, 2013, pp. 840–846.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, pp. 541–551, 1989.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[16] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, Jun. 2014, pp. 512–519.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3431–3440.

[18] M. Cogswell, X. Lin, S. Purushwalkam, and D. Batra. (2014). Combining the best of graphical models and convnets for semantic segmentation [Online]. Available: http://arxiv.org/abs/1412.4313

[19] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5162–5170.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 580–587.

[22] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1161–1168.

[23] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3d scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2009.

[24] M. Liu, M. Salzmann, and X. He, "Discrete-continuous depth estimation from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 716–723.

[25] H. Su, Q. Huang, N. J. Mitra, Y. Li, and L. Guibas, "Estimating image depth using shape collections," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–11, 2014.

[26] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1966–1974.

[27] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[28] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1799–1807.

[29] T. Baltrušaitis, L.-P. Morency, and P. Robinson, "Continuous conditional neural fields for structured regression," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 593–608.

[30] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 633–640.

[31] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2014.

[33] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in *Proc. IEEE Int. Conf. Pattern Recog.*, 1994, pp. 582–585.

[34] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014.

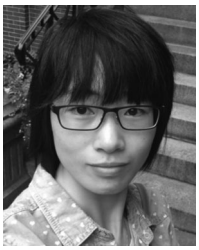
[35] P. K. Nathan Silberman, D. Hoiem, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[36] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, pp. 1231–1237, 2013.

[37] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

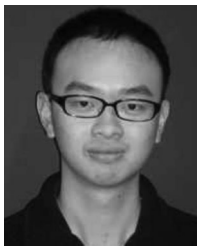
[38] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol 21, pp. 257–266, 2002.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1–9.



Fayao Liu received the BEng and MEng degrees from the School of Computer Science, National University of Defense Technology, Hunan, China, in 2008 and 2010, respectively. She is currently working toward the PhD degree at the University of Adelaide, Adelaide, Australia. Her current research interests include machine learning and computer vision.

Chunhua Shen studied at Nanjing University, Australian National University, and received the PhD degree from the University of Adelaide. He is a professor of computer science at the University of Adelaide. His research interests are in the intersection of computer vision and statistical machine learning. Before he moved back to the University of Adelaide in 2011, he was with the computer vision program at NICTA (National ICT Australia), Canberra Research Laboratory for about six years. In 2012, he was awarded the Australian Research Council Future Fellowship.



Guosheng Lin received the bachelor and master degrees from the South China University of Technology in computer science in 2007 and 2010, respectively, and the PhD degree in computer vision from the University of Adelaide in 2015. He is a research fellow at the School of Computer Science at the University of Adelaide. His research interests are on computer vision and machine learning.

Ian Reid received the BSc degree in computer science and mathematics with first class honors from the University of Western Australia in 1987 and was awarded a Rhodes Scholarship in 1988 in order to study at the University of Oxford, where he received the DPhil degree in 1991. He is a professor of computer science at the University of Adelaide. His research interests include active vision, visual navigation, visual geometry, human motion capture, and intelligent visual surveillance, with an emphasis on real-time aspects of the computations.