Monocular Depth Estimation Using Conditional FRAME Model

Hao Xin

April 2019

1 Introduction

Monocular depth estimation is an important problem in scene understanding, depth information from single RGB image can be useful for 3D reconstruction. One fundamental question of monocular depth learning is that whether there is indeed rich enough depth information contained in a single RGB image. If the answer is yes, then it would be possible for a well proposed model to learn depth in an unsupervised manner. Otherwise, supervised learning scheme should be preferred.

In the context of supervised learning, similar to semantic segmentation, the prediction task is pixel-wised. Thus, the problem can be seen as a pixel-wised regression problem. A lot of CNN based models have been proposed so far. Some observations from computer vision researchers shows some guideline on designing deep convolutional neural networks for depth learning are:

1. Global spatial information (usually captured by large receptive field) is beneficiary.
2. Combining imagenet pre-trained models (such as VGG and ResNet) helps improve performance.
3. Skip connections with middle level information could help with dense prediction.

2 Statistics perspectives

2.1 Model for Depth Map

1. FRAME (Markov random field) model.
2. CRF (Conditional random filed) model.

A non-trivial problem for random field model is to define neighborhood relation on image pixels. FRAME model, though based on Markov random field model, bypass clique potentials used in traditional MRF by using expressive filters to capture graph relations.

Let $\mathbf{D}$ be the depth map defined on spatial domain $\mathcal{D}$ and $\mathbf{X}$ be the input RGB image. A conditional random field model defines a probability measure $P_\theta(\mathbf{D}|\mathbf{X}) : \mathbb{R}_+^{|\mathcal{D}|} \to [0,1]^{|\mathcal{D}|}$, where θ are unknown underlying model parameters. For a given CRF model, one would want to find the maximum likelihood estimate that

$$\hat{\mathbf{D}} = \arg\max_{\mathbf{D}} P_\theta(\mathbf{D}|\mathbf{X})$$

The non-stationary inhomogeneous FRAME model on texture can be generalized to a conditional FRAME model on depth map in the following form:

$$P(\mathbf{D};\lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K}\sum_{y\in\mathcal{D}} \lambda_{k,y}\big([F_k * \mathbf{D}](y), \mathbf{X}\big)\right] q(\mathbf{D}) \tag{1}$$

where function $\lambda_{k,y}(\cdot, \mathbf{X})$ depend on spatial position y and $\{F_k : k = 1, \ldots, K\}$ is a bank of filters. $q(\mathbf{D})$ works as a reference distribution here, in Bayesian inference $q(\mathbf{D})$ is the prior distribution on $\mathbf{D}$. For simplifying learning, we use

$$\lambda_{k,y}(r, \mathbf{X}) = \omega_{k,y}(\mathbf{X}) \max(0, r)$$

It also make sense to consider

$$\lambda_{k,y}(r, \mathbf{X}) = \omega_{k,y}(\mathbf{X}) \cdot r$$

which will all different pixels contribute. With a with a slight abuse of notation, we will denote both the above two function as the same, so we get the conditional FRAME model for depth:

$$P_\theta(\mathbf{D}|\mathbf{X}) = \frac{1}{Z(W,F)} \exp\left[\sum_{k=1}^{K}\sum_{y\in\mathcal{D}} \omega_{k,y}(\mathbf{X})[F_k * \mathbf{D}](y)\right] q(\mathbf{D}) \tag{2}$$

W can be learned by maximum likelihood estimation criteria. Given training data $\{(\mathbf{X}_i, \mathbf{D}_i)\}_{i=1}^m$ the gradient of log-likelihood for energy based model (EBM) follows:

$$L'(\theta) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial\theta} \log P_\theta(\mathbf{D}_i|\mathbf{X}_i) = \mathbb{E}_{p_{data}}\left[\frac{\partial}{\partial\theta} \log P_\theta(\mathbf{D}|\mathbf{X})\right] \tag{3}$$

$$= \mathbb{E}_{p_{data}}\left[\frac{\partial}{\partial\theta} f_\theta(\mathbf{D}|\mathbf{X}) - \frac{\partial}{\partial\theta} \log Z(\theta|\mathbf{X})\right] \tag{4}$$

$$= \mathbb{E}_{p_{data}}\left[\frac{\partial}{\partial\theta} f_\theta(\mathbf{D}|\mathbf{X})\right] - \mathbb{E}_{p_\theta}\left[\frac{\partial}{\partial\theta} f_\theta(\mathbf{D}|\mathbf{X})\right] \tag{5}$$

$$\approx \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial\theta} f_\theta(\mathbf{D}_i|\mathbf{X}_i) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial}{\partial\theta} \tag{6}$$

Due to the intractable normalizing constant $Z(\theta|\mathbf{X})$ during computation, we need to use MCMC samples $\tilde{\mathbf{D}}$ from $P_\theta(\mathbf{D}|\mathbf{X})$ to approximate the gradient update for learning $\theta = W$.

Following previous work (learning FRAME with CNN filters) we have the following learning algorithm:

Algorithm 1: Learning conditional FRAME model.

input : Initial weight θ_0, training steps T, training data $\{(\mathbf{X}_i, \mathbf{D}_i)\}_{i=1}^N$, batch size m, variance of noise σ^2, Langevin descretization δ steps n, learning rate η

output: θ_{T+1}

for $t = 0{:}T$ **do**

1. Draw batch samples $\{(\mathbf{X}_i, \mathbf{D}_i)\}$;

2.**for** i=1 : m **do**

Draw n samples $\{\tilde{\mathbf{D}}_i\}$ from $P_{\theta_t}(\mathbf{D}|\mathbf{X}_i)$ using Langevin dynamics:

$$\tilde{\mathbf{D}}_{\tau+1} = \tilde{\mathbf{D}}_\tau + \frac{\delta}{2} f'_{\theta_t}(\mathbf{D}|\mathbf{X}_i) + \sqrt{\delta} U_\tau$$

where $U_\tau \sim N(0, \sigma^2 I)$ is Gaussian noise.

end

3. Update θ_t by

$$\theta_{t+1} = \theta_t + g(\Delta(\theta_t), \eta, t)$$

where $\Delta(\theta_t)$ is the approximate gradient from (6).

end

2.2 Convolution Operation

Large receptive field of CNN filters has been shown to be effective on capturing global features in practice. However, large filter size would bring in computation burden especially for deep network structures. Atrous convolution, also known as dilated convolution solves the computational difficulty for large filters by employ a sparse convolution with input feature maps. Is there exist other way of convolution can effectively capture global features for depth learning?

Another issue is the pooling operation combined with convolution layers. Pooling is way of down sampling output feature maps further for computation efficiency. However, this can be inappropriate for depth learning problem, for example, max-pooling could cause ignoring useful local information which could be important for depth prediction. Should we still keep pooling for convolutional neural networks?

References