Multi-View Depth Estimation by Fusing Single-View Depth Probability with Multi-View Geometry *Supplementary Material*

Gwangbin Bae   Ignas Budvytis   Roberto Cipolla
University of Cambridge
{gb585,ib255,rc10001}@cam.ac.uk

1. Mathematical Formulation

In this section, we provide the mathematical formulation for the proposed probabilistic depth sampling and consistency-weighted multi-view matching.

1.1. Probabilistic Depth Sampling

Suppose that, for pixel (u, v) in the reference image, D-Net estimates the single-view depth probability distribution (parameterized as a Gaussian) of mean $\mu_{u,v}$ and variance $\sigma^2_{u,v}$. We first define the search space $[\mu_{u,v} - \beta\sigma_{u,v}, \mu_{u,v} + \beta\sigma_{u,v}]$, where β is a hyper-parameter. Given the probability density function $p_{u,v}(x)$, the probability mass $P^*_{u,v}$ covered by the search space is

$$\begin{aligned} P^*_{u,v} &= \int_{\mu_{u,v}-\beta\sigma_{u,v}}^{\mu_{u,v}+\beta\sigma_{u,v}} p_{u,v}(x)dx \\ &= F_{u,v}(\mu_{u,v} + \beta\sigma_{u,v}) - F_{u,v}(\mu_{u,v} - \beta\sigma_{u,v}) \\ &= \text{erf}\left(\frac{\beta}{\sqrt{2}}\right), \end{aligned} \tag{1}$$

where $F_{u,v}(\cdot)$ and $\text{erf}(\cdot)$ are the cumulative distribution function and the error function. $P^*_{u,v}$ depends only on β and we can thus drop the subscripts. We then split the search space into N_s bins of equal probability mass, P^*/N_s, and select their mid-points as depth candidates. The k-th depth candidate, $d_{u,v,k}$, is thus defined as

$$\begin{aligned} d_{u,v,k} = \frac{1}{2}\Bigg[& F^{-1}_{u,v}\left(\frac{k-1}{N_s}P^* + \frac{1-P^*}{2}\right) \\ & + F^{-1}_{u,v}\left(\frac{k}{N_s}P^* + \frac{1-P^*}{2}\right)\Bigg], \end{aligned} \tag{2}$$

where $F^{-1}_{u,v}(\cdot)$ is the Gaussian quantile function (i.e. the inverse of $F_{u,v}(\cdot)$). Eq. 2 can be simplified into Eq. 4 in the paper by using the relation $F^{-1}_{u,v}(p) = \mu_{u,v} + \sigma_{u,v}\Phi^{-1}(p)$, where $\Phi^{-1}(p)$ is the quantile function of the standard normal distribution (i.e. the probit function).

1.2. Consistency-Weighted Multi-View Matching

In this section, we explain how the consistency-weighted multi-view matching score of each depth candidate can be calculated.

Step 1. Pixel coordinates of I_t → World coordinates. A 3D point $\mathbf{X}^c_t$ in the camera-centered coordinates of the reference image I_t at time t is projected to the pixel coordinates $\mathbf{w}_t$ via perspective projection. Given the camera calibration matrix $\mathbf{K}$, this can be written as

$$\begin{aligned} \tilde{\mathbf{w}}_t &= \mathbf{K}\mathbf{X}^c_t \\ \begin{bmatrix} su \\ sv \\ s \end{bmatrix} &= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^c_t \\ Y^c_t \\ Z^c_t \end{bmatrix}, \end{aligned} \tag{3}$$

where $\tilde{\mathbf{w}}_t$ is the homogeneous representation of the pixel coordinates $\mathbf{w}_t = (u, v)$. Suppose that, for pixel (u, v) in the reference image I_t, we have sampled N_s depth candidates $\{d_k\}^{N_s}_{k=1}$. Each depth candidate, together with the pixel coordinates, defines $\mathbf{X}^c_t$ as

$$\mathbf{X}^c_t = \begin{bmatrix} X^c_t \\ Y^c_t \\ Z^c_t \end{bmatrix} = \begin{bmatrix} \frac{u-u_0}{\alpha_u} \\ \frac{v-v_0}{\alpha_v} \\ 1 \end{bmatrix} \times d_k. \tag{4}$$

The world coordinates of $\mathbf{X}^c_t$ is then given as $\tilde{\mathbf{X}}^w_t = \mathbf{P}^{-1}_t\tilde{\mathbf{X}}^c_t$, where $\mathbf{P}_t$ is the rigid body transformation matrix (i.e. camera pose) of I_t, and $\tilde{\mathbf{X}}$ is the homogeneous representation of $\mathbf{X}$.

Step 2. World coordinates → Pixel coordinates of I_i. $\mathbf{X}^w_t$ is then projected to a neighboring image, I_i. The projection can be written as

$$\begin{aligned} \tilde{\mathbf{X}}^c_i &= \begin{bmatrix} X_{ik} \\ Y_{ik} \\ d_{ik} \\ 1 \end{bmatrix} = \mathbf{P}_i\tilde{\mathbf{X}}^w_t \\ \text{and} \quad \tilde{\mathbf{w}}_i &= \begin{bmatrix} su_{ik} \\ sv_{ik} \\ s \end{bmatrix} = \mathbf{K}\mathbf{X}^c_i. \end{aligned} \tag{5}$$

Method	NLL (lower the better)		
	ScanNet	7-Scenes	KITTI
D-Net (single-view)	2.235	2.794	2.644
Full pipeline (multi-view)	**0.145**	**1.561**	**1.805**

Table 1. Negative log-likelihood of the ground truth depth evaluated for D-Net (single-view) and the full pipeline.

In Eq. 5, $\mathbf{X}_i^c = (X_{ik}, Y_{ik}, d_{ik})$ is the camera-centered coordinates of image I_i and $\mathbf{w}_i = (u_{ik}, v_{ik})$ is the corresponding pixel coordinates. It is the projection of the 3D coordinates - defined by (u, v) and d_k in I_t - on I_i. This defines u_{ik}, v_{ik} and d_{ik} that appear in Eq. 3 and 5 in the paper.

Step 3. Calculating the matching score. The matching score for the k-th depth candidate is computed in terms of the dot product between the feature vectors - $\mathbf{f}_{u,v}(I_t)$ and $\mathbf{f}_{u_{ik},v_{ik}}(I_i)$. Since u_{ik} and v_{ik} are continuous values, $\mathbf{f}_{u_{ik},v_{ik}}(I_i)$ is obtained by bilinearly interpolating the 4-pixel neighbors - $(\lceil u_{ik} \rceil, \lceil v_{ik} \rceil)$, $(\lfloor u_{ik} \rfloor, \lceil v_{ik} \rceil)$, $(\lceil u_{ik} \rceil, \lfloor v_{ik} \rfloor)$ and $(\lfloor u_{ik} \rfloor, \lfloor v_{ik} \rfloor)$.

Step 4. Consistency-weighting. Similarly, $\mu_{u_{ik},v_{ik}}(I_i)$ and $\sigma_{u_{ik},v_{ik}}(I_i)$ can be bilinearly interpolated to evaluate the single-view depth probability of d_{ik} at (u_{ik}, v_{ik}), which gives the depth consistency weight (Eq. 5 in the paper).

2. Additional Quantitative Results

Negative log-likelihood. Depth metrics (e.g. RMSE) only reflect the accuracy of the predicted mean μ. In order to evaluate the accuracy of the distribution $\mathcal{N}(\mu, \sigma^2)$, we report the average negative log-likelihood (NLL) of the ground truth depth. Tab. 1 shows how NLL is reduced via multi-view matching.

3. Additional Qualitative Results

Update in the pixel-wise prediction. Fig. 1 shows how the pixel-wise prediction is updated throughout the pipeline. The initial D-Net prediction has high uncertainty, mainly due to the inherent ambiguity of single-view depth. This leads to large search space. After performing the multi-view matching for the sampled candidates, G-Net shifts the mean towards the candidate with high matching score. The variance is reduced so that the spacing between the candidates is smaller in the next iteration (i.e. better depth resolution).

Qualitative comparison against the state-of-the-art. Fig. 2 and Fig. 3 provide additional qualitative comparison against [4] (extension of Fig. 4 in the paper). Our network shows superior performance especially on textureless/reflective surfaces and moving objects.

4. Network Architecture

Tab. 2, 3 and 4 show the architecture of D-Net, G-Net and the learned upsampling layer.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 6

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013. 5

[4] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 5

[5] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019. 6

[6] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, 2020. 6

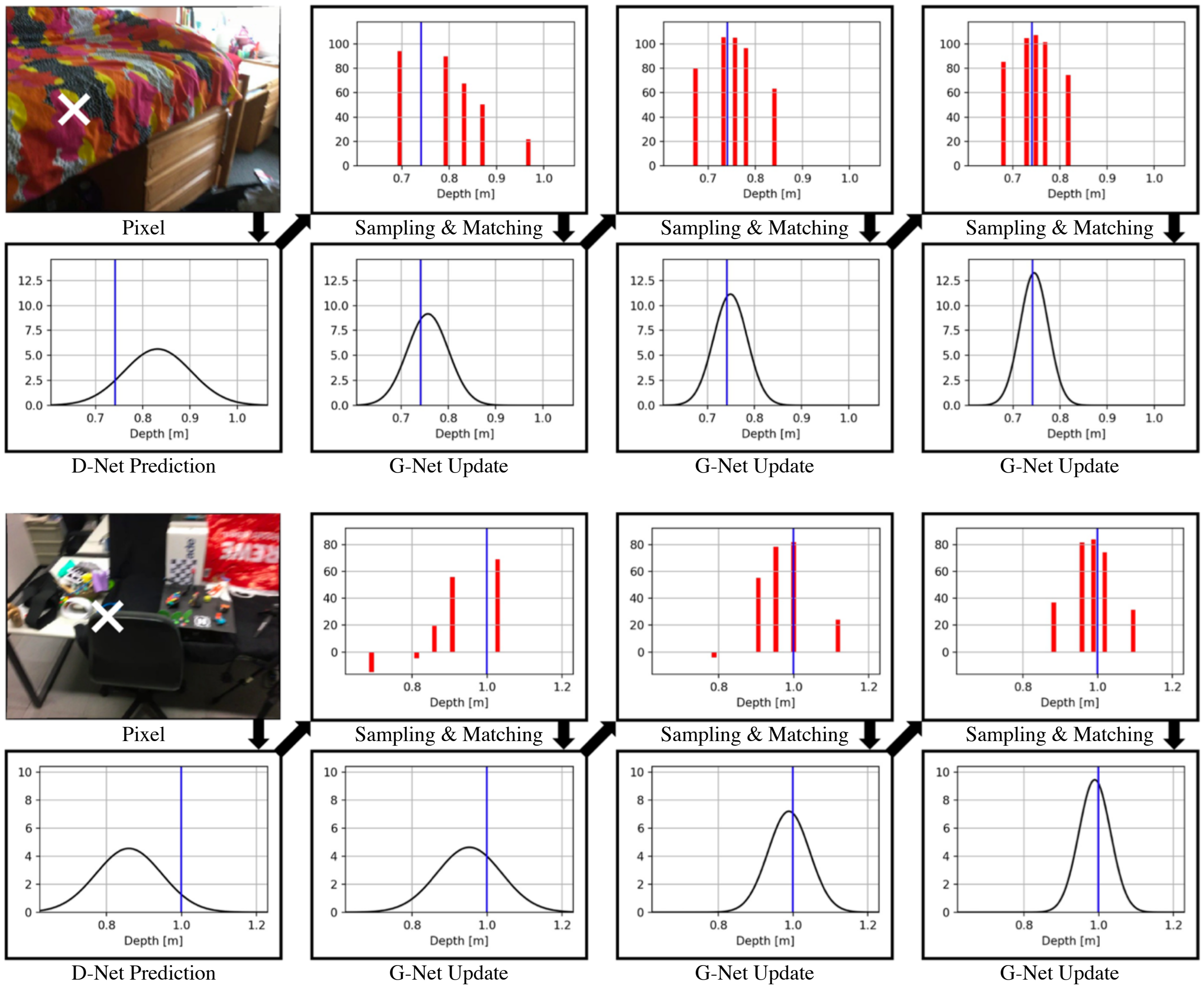

Figure 1. Iterative update in the pixel-wise depth probability distribution. The black curves and blue vertical lines are the estimated depth probability distributions and the ground truth depth, respectively. Red bar plots show the consistency-weighted multi-view matching scores measured at the sampled depth candidates. Throughout the iterative refinement, the distribution becomes more accurate and confident (i.e. σ is reduced).

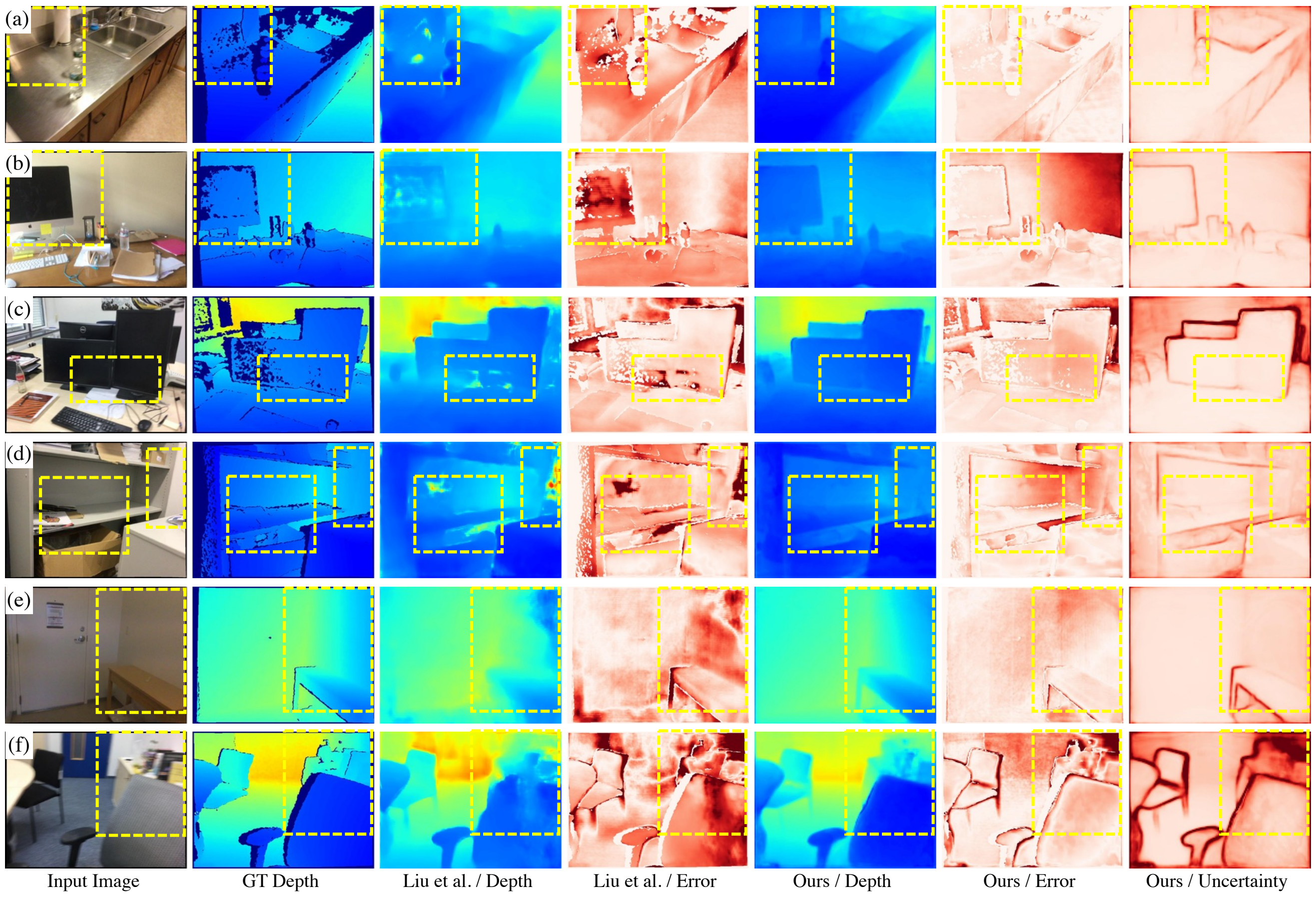

Figure 2. Qualitative comparison against [4] on ScanNet [2]. With the proposed fusion of single-view depth probability (probabilistic sampling and consistency weighting), MaGNet can make accurate prediction for reflective surfaces (a-c) and texture-less surfaces (d-e). If a depth candidate is occluded in a particular neighboring frame, the corresponding single-view depth probability estimated from that view will be low. In such case, MaGNet ignores that view by setting the depth-consistency weight to zero. This improves the prediction near occlusion boundaries (f).

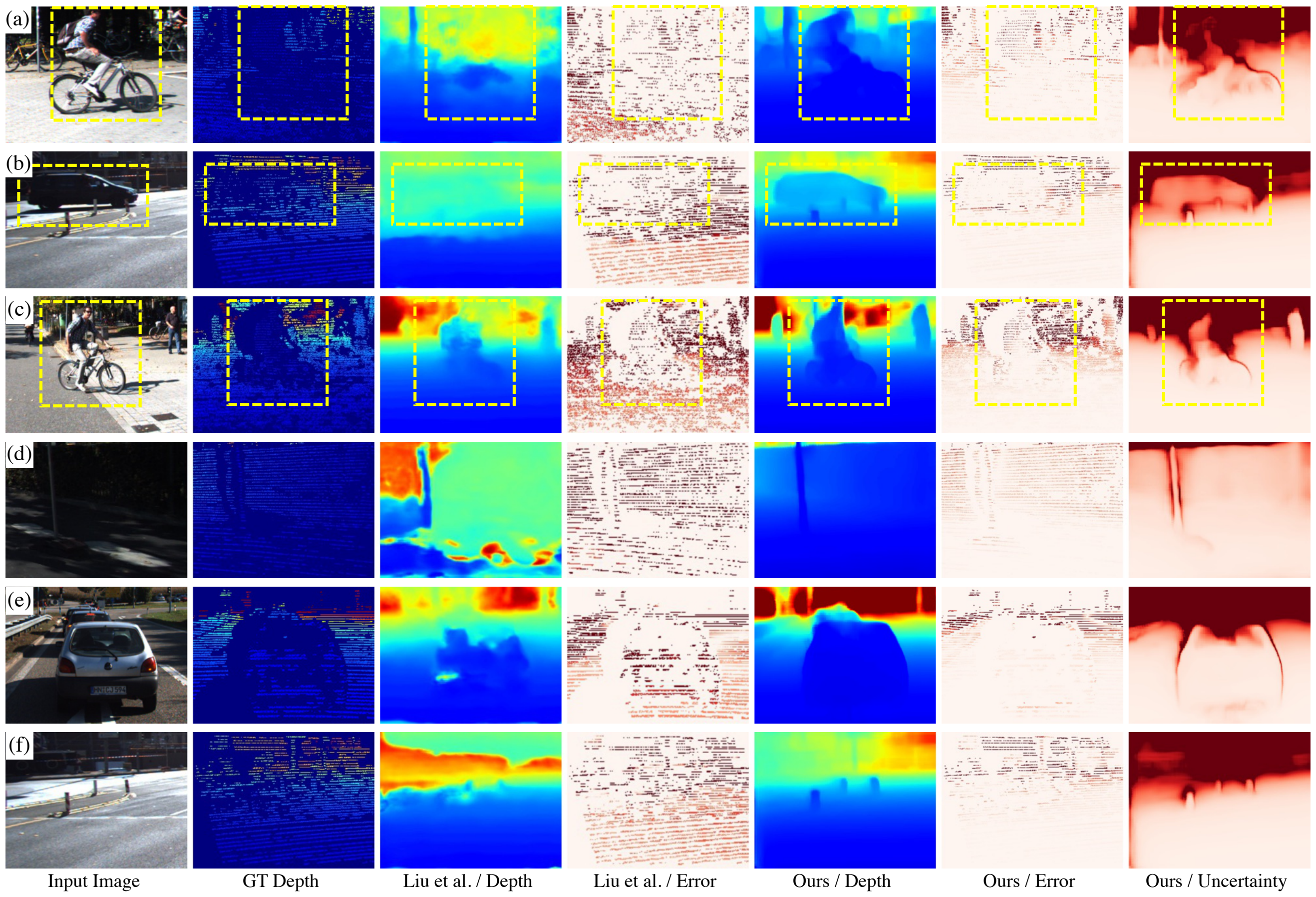

Figure 3. Qualitative comparison against [4] on KITTI [3]. Examples (a-c) show dynamic objects, for which multi-view consistency assumption is violated. Examples (d-f) show images where the camera was static within the local window. In such case, all depth candidates lead to same matching score (i.e. ambiguous matching). In both scenarios, MaGNet can make accurate predictions. This is because the proposed fusion enforces the final output to be consistent with the single-view depth probability distributions.

Input	**Layer**	**Output**	**Output Dimension**
image	-	-	$H \times W \times 3$
Encoder			
image	EfficientNet B5 [5]	$FEAT_4$ $FEAT_8$ $FEAT_{16}$ $FEAT_{32}$	$H/4 \times W/4 \times 40$ $H/8 \times W/8 \times 64$ $H/16 \times W/16 \times 176$ $H/32 \times W/32 \times 2048$
Decoder			
$FEAT_{32}$	Conv2D(ks=1, C_{out}=2048, padding=0)	*x_d0*	$H/32 \times W/32 \times 2048$
upsample(*x_d0*) + $FEAT_{16}$	$\begin{pmatrix}\text{Conv2D(ks=3, } C_{\text{out}}\text{=1024, padding=1),} \\ \text{BatchNorm2D(),} \\ \text{LeakyReLU()}\end{pmatrix} \times 2$	*x_d1*	$H/16 \times W/16 \times 1024$
upsample(*x_d1*) + $FEAT_8$	$\begin{pmatrix}\text{Conv2D(ks=3, } C_{\text{out}}\text{=512, padding=1),} \\ \text{BatchNorm2D(),} \\ \text{LeakyReLU()}\end{pmatrix} \times 2$	*x_d2*	$H/8 \times W/8 \times 512$
upsample(*x_d2*) + $FEAT_4$	$\begin{pmatrix}\text{Conv2D(ks=3, } C_{\text{out}}\text{=256, padding=1),} \\ \text{BatchNorm2D(),} \\ \text{LeakyReLU()}\end{pmatrix} \times 2$	*x_d3*	$H/4 \times W/4 \times 256$
x_d3	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=2, padding=0)	*out*	$H/4 \times W/4 \times 2$

Table 2. D-Net architecture. In each convolutional layer, "ks" means the kernel size and C_{out} is the number of output channels. $FEAT_N$ represents the feature-map of resolution $H/N \times W/N$. $X + Y$ means that the two tensors are concatenated, and upsample$(\cdot)$ is bilinear upsampling. Note that a different activation function is applied to each channel of the final output, as explained in Sec. 3.1 of the paper.

Input	**Layer**	**Output**	**Output Dimension**
cost-volume + *x_d3*	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=2, padding=0)	*out*	$H/4 \times W/4 \times 2$

Table 3. G-Net architecture. Using the cost-volume and the feature-map from D-Net as input, G-Net updates the initial depth probability distribution by estimating $\Delta\mu/\sigma$ and $\sigma^{\text{new}}/\sigma$. Similar to D-Net, we use linear activation for $\Delta\mu/\sigma$, and the modified ELU function [1], $f(x) = \text{ELU}(x) + 1$ for $\sigma^{\text{new}}/\sigma$ to ensure positive variance and smooth gradient.

Input	**Layer**	**Output**	**Output Dimension**
x_d3	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=4×4×9, padding=0)	*out*	$H/4 \times W/4 \times (4 \times 4 \times 9)$

Table 4. Architecture of the learned upsampling layer. The output, which has the shape of $H/4 \times W/4 \times (4 \times 4 \times 9)$ is reshaped into $H \times W \times 9$, and softmax is applied to the last channel. Then, the depth of each pixel in the full resolution is calculated as the weighted sum of the 3×3 grid of its coarse resolution neighbors [6].