Probabilistic Depth

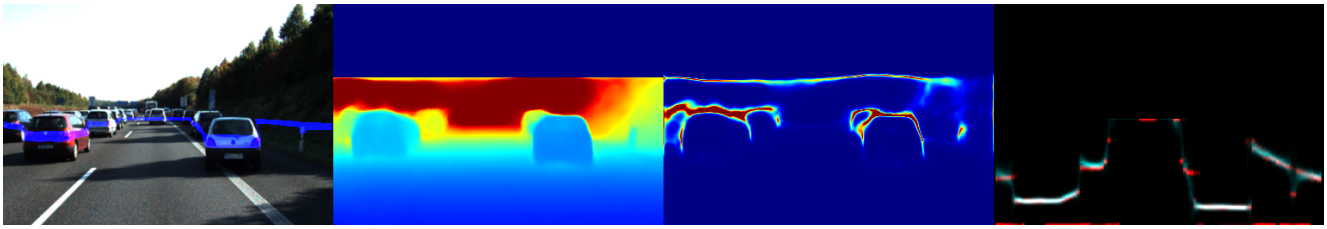

Figure 1: **(a)** RGB Image with blue region representing a slice above the road. **(b)** Expectation of Depth distribution yields a depth map. **(c)** We compute the variance of the distribution and display it. Note that the edges are more uncertain. **(d)** A top-down birds eye view of the Probabilistic Depth volume. Blue is the prediction and red is the ground truth

1. Introduction

To capture the error and uncertainty in RGB-only depth estimation, previous work had formulated that task as a probabilistic regression problem, by predicting per-pixel depth distributions via a Depth Probability Volume (DPV) [3] [1] [4]. The DPV provides both a Maximum Likelihood-Estimate (MLE) of the depth map, as well as the corresponding per-pixel uncertainty measure. We combine the ideas from this paper and present it in this clean easy to understand code base.

1.1. Representation

We wish to estimate the depth map $\mathbf{D} = \{d_{u,v}\}$ of the scene, which specifies the depth value $d_{u,v}$ for every camera pixel (u, v) at spatial resolution [H,W]. Since there is inherent uncertainty in the depth value at every pixel, we represent a *probability distribution* over depths for every pixel. Let us define $\mathbf{d}_{u,v}$ to be a *random variable* for depth predictions at the pixel (u, v). We quantize depth values into a set $\mathcal{D} = \{d_0, \ldots, d_{N-1}\}$ of N discrete, uniformly spaced depth values lying in $(d_{\text{min}}, d_{\text{max}})$. All the predictions $\mathbf{d}_{u,v} \in \mathcal{D}$ belong to this set. The output of our depth estimation method for each pixel is a probability distribution $P(\mathbf{d}_{u,v})$, modeled as a categorical distribution over $\mathcal{D}$. In this work, we use $N = 64$, resulting in a Depth Probability Volume (DPV) tensor of size [64, W, H]:

$$\mathcal{D} = \{d_0, \ldots, d_{N-1}\}; d_q = d_{\text{min}} + (d_{\text{max}} - d_{\text{min}}) \cdot q \quad (1)$$

$$\sum_{q=0}^{N-1} P(\mathbf{d}_{u,v} = d_q) = 1 \text{ (q is the quantization index)} \quad (2)$$

$$\text{Depth estimate } = \mathbb{E}[\mathbf{d}_{u,v}] = \sum_{q=0}^{N-1} P(\mathbf{d}_{u,v} = d_q) \cdot d_q \quad (3)$$

This DPV can be initialized using another sensor such as an RGB camera, or can be initialized with a Uniform or Gaussian distribution with a large σ for each pixel.

For easier visualization, we compress our DPV into a top-down an "Uncertainty Field" (UF) [4], by averaging the probabilities of the DPV across a subset of each column (Fig. 2). This subset considers those pixels (u, v) whose corresponding 3D heights $h(u, v)$ are between $(h_{\text{min}}, h_{\text{max}})$. The UF is defined for the camera column u and quantized depth location q as:

$$UF(u, q) = \frac{1}{|\mathcal{V}(u)|} \sum_{v \in \mathcal{V}(u)} P(\mathbf{d}_{u,v} = d_q)$$
$$\text{where } \mathcal{V}(u) = \{v \mid h_{\text{min}} \leq h(u, v) \leq h_{\text{max}}\} \quad (4)$$

We denote the categorical distribution of the uncertainty field on the u-th camera ray as:
$UF(u) = \text{Categorical}(d_q \in \mathcal{D} \mid P(d_q) = UF(u, q))$.

2. Neural Network

We use a Deep Learning based architecture to learn how to predict this DPV.

2.1. Structure of Network

The first step is to build a network (Fig. 3) that can generate DPV's from RGB images. We extend the Neural-RGBD [3] architecture to incorporate light curtain measurements. Multiple (N) images, usually two (I_0, I_1), are fed into shared encoders, and the features are then warped into different fronto-parallel planes of the reference image I_0 using pre-computed camera extrinsics $R^{I_1}_{I_o}, t^{I_1}_{I_o}$. Further convolutions are run to generate a low resolution DPV $dpv^{\text{l0}}_{\text{t}}$ [H/4, W/4] where the log softmax operator is applied and regressed on. The transformation between the cameras acts

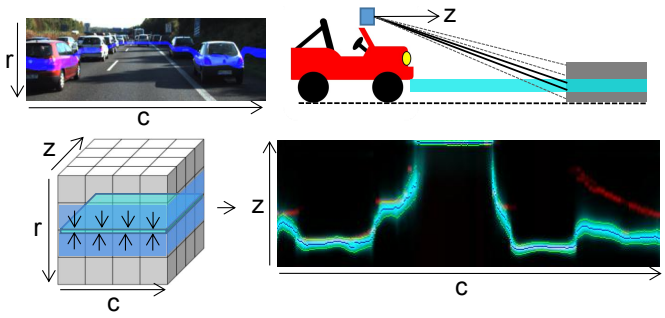

Figure 2: Our state space consists of a Depth Probability Volume (DPV) (left) storing per-pixel uncertainty distributions. It can be collapsed to a Bird's Eye Uncertainty Field (UF) (right) by averaging those rays in each row (blue pixels) of the DPV that correspond to a slice on the road parallel to the ground plane (right) (cyan pixels). Red pixels on UF represent the low resolution LIDAR ground truth.

as a constraint, forcing the feature maps to respect depth to channel correspondence. The add operator into a common feature map is similar to adding probabilities in log-space.

$dpv_{\mathrm{t}}^{\mathrm{l0}}$ is then fed into the DPV Fusion Network (a set of 3D Convolutions) that incorporate a downsampled version of $dpv_{\mathrm{t}-1}^{\mathrm{L}}$, and a residual is computed and added back to $dpv_{\mathrm{t}}^{\mathrm{l0}}$ to generate $dpv_{\mathrm{t}}^{\mathrm{l1}}$ to be regressed upon similarly. Finally, $dpv_{\mathrm{t}}^{\mathrm{l1}}$ is then passed into a decoder with skip connections to generate a high resolution DPV $dpv_{\mathrm{t}}^{\mathrm{L}}$. We repeat this process.

2.2. Loss Functions

Soft Cross Entropy Loss: We build upon the ideas in [5] and use a soft cross entropy loss function, with the ground truth LIDAR depthmap becoming a Gaussian DPV with σ_{gt} instead of a one hot vector. This way, when estimating $\mathbb{E}\left(dpv^{gt}\right)$ we get the exact depth value instead of an approximations limited by the depth quantization $\mathcal{D}$. We also make the quantization slightly non-linear to have more steps between objects that are closer to the camera:

$$l_{sce} = \frac{-\sum_i \sum_d \left(dpv^{\{\mathrm{l0},\mathrm{l1},\mathrm{L}\}} * log\left(dpv^{gt}\right)\right)}{n} \quad (5)$$

$$\mathcal{D} = \{d_0, \ldots, d_{N-1}\}; d_q = d_{\min} + (d_{\max} - d_{\min}) \cdot q^{pow} \quad (6)$$

L/R Consistency Loss: We train on both the Left and Right Images of the stereo pair whose Projection matrices P_l, P_r are known [2]. We enforce predicted Depth and RGB consistency by warping the Left Depthmap into the Right Camera and vice-versa, and minimize the following metric:

$$D_l = \mathbb{E}\left(dpv_l^L\right) \qquad D_r = \mathbb{E}\left(dpv_r^L\right) \quad (7)$$

$$l_{dcl} = \frac{1}{n}\sum_i \left(\frac{\left|D_{\{l,r\}} - w\left(D_{\{r,l\}}, P_{\{l,r\}}\right)\right|}{D_{\{l,r\}} + w\left(D_{\{r,l\}}, P_{\{l,r\}}\right)}\right) \quad (8)$$

$$l_{rcl} = \frac{1}{n}\sum_i \left(||I_{\{l,r\}} - w\left(I_{\{r,l\}}, D_{\{l,r\}}, P_{\{l,r\}}\right)||_1\right) \quad (9)$$

Edge aware Smoothness Loss: We ensure that neighbouring pixels have consistent surface normals, except on the edges/boundaries of objects with the Sobel operator S_x, S_y via the term:

$$l_s = \frac{1}{n}\sum_i \left(\left|\frac{\partial I}{\partial x}\right| e^{-|S_x I|} + \left|\frac{\partial I}{\partial y}\right| e^{-|S_y I|}\right) \quad (10)$$

3. Experiments

We train and validate our algorithms on the KITTI dataset.

For evaluation, we consider the RMSE metric against the entire depthmap as opposed to just the Uncertainty Field (UF) as $\sqrt{\sum_{i=1}^{n} \frac{(\mathbb{E}(\mathbf{d}_{u,v}) - \mathbf{d_{gt}}(u,v)_i)^2}{n}}$ against our ground truth.

Effects of our loss function: We do some simple experiments, training the task of monocular depth estimation, and we explore the effects of enabling various loss functions:

Parameters	RMSE/m
$\sigma_{gt} = 0.05$	3.24
$\sigma_{gt} = 0.2$	3.16
$\sigma_{gt} = 0.3$	3.06
$\sigma_{gt} = 0.3$ with l_{dcl}, l_{rcl}	2.93
$\sigma_{gt} = 0.3$ with l_{dcl}, l_{rcl}, l_s	2.90

Table 1: Effects of Soft Cross Entropy (σ_{gt}), Left/Right Consistency (l_{dcl}, l_{rcl}), Smoothness losses (l_s) on Monocular Depth Estimation.

Table 1 shows successively improving performance as we increase σ_{gt}, with poorer performance when the depth is effectively encoded as a one-hot vector (eg. $\sigma_{gt} = 0.05$), since the depth was more likely to be forced into one of the categories in $\mathcal{D}$. Adding in l_{dcl}, l_{rcl} and l_s improved performance further, but we did not see any major performance improvement in varying q^{pow} (depth quantization $\mathcal{D}$ varying non-linearly) when σ_{gt} was increased.

Effect of Stereo Inputs: We experiment with a monocular pair at times $t, t-1$, or a stereo pair at time t as input, with extrinsics known in both cases. We note significantly better performance with Stereo input. Note that our method can generalize to any N camera setup (Fig. 5).

Effect of DPV Fusion Network: We consider the task of LIDAR Upsampling. The Velodyne LIDAR in the KITTI dataset, can be converted into a low resolution depthmap, and consequently a low-res DPV we call dpv_t^{gt}. We could then fuse both dpv_t^{l0} and dpv_t^{gt} to generate dpv_t^{l1} using Bayesian inference. Alternatively, we could feed both of those inputs into our DPV Fusion Network. We note improved performance in this upsampling task when using this approach as seen in Fig. 5.

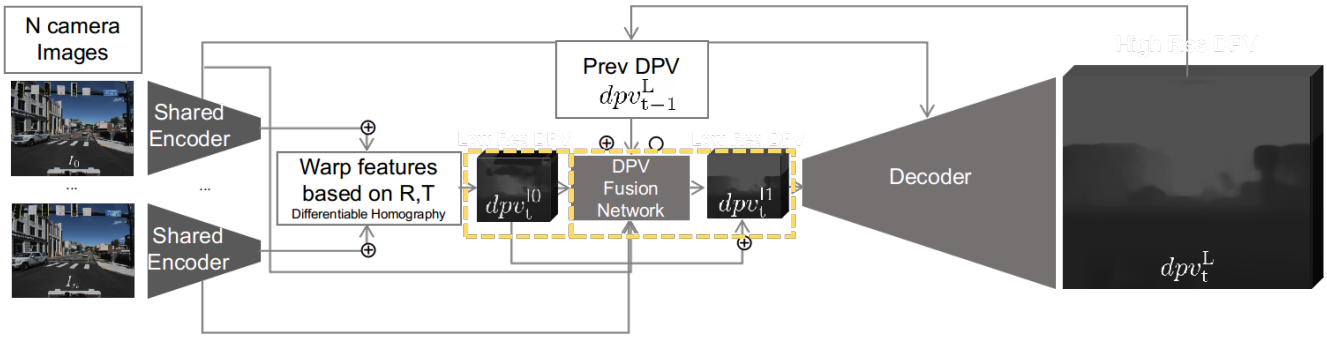

Figure 3: Our network takes in RGB images to generate a Depth Probability Volume (DPV)

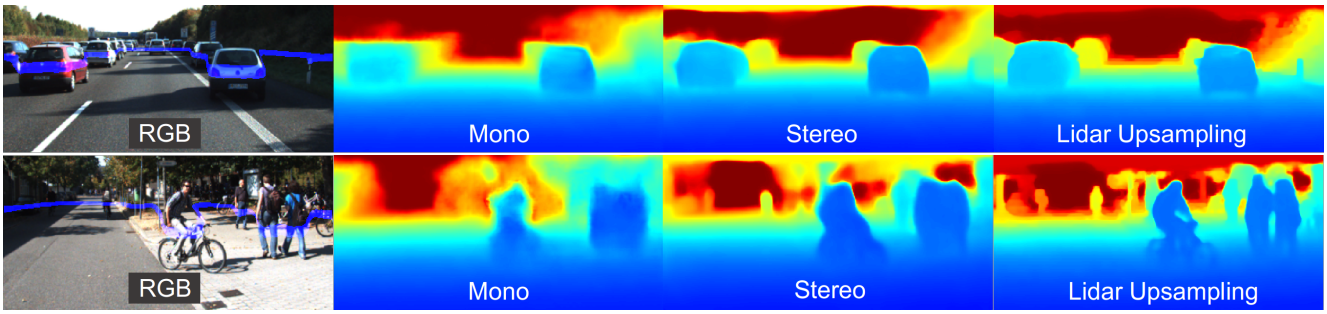

Figure 4: Monocular, Stereo and Lidar Upsampling from RGB data

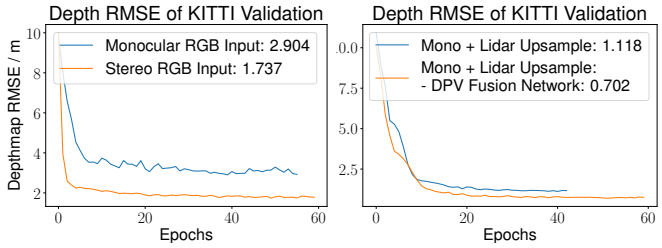

Figure 5: **Left:** Stereo pair at t instead of Monocular pair at $t, t-1$ input to the network. **Right:** Fusing the GT LIDAR data with dpv_t^{l0} to generate dpv_t^{l1} and dpv_t^{L} with Bayesian inference vs DPV Fusion Network.

References

[1] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 1

[2] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency, 2017. 2

[3] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa Narasimhan, and Jan Kautz. Neural rgb-¿d sensing: Depth and uncertainty from a video camera, 2019. 1

[4] Gengshan Yang, Peiyun Hu, and Deva Ramanan. Inferring distributions over depth from a single image, 2019. 1

[5] Gengshan Yang, Peiyun Hu, and Deva Ramanan. Inferring distributions over depth from a single image. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2019. 2