Real-Time Restoration of Dark Stereo Images Supplementary

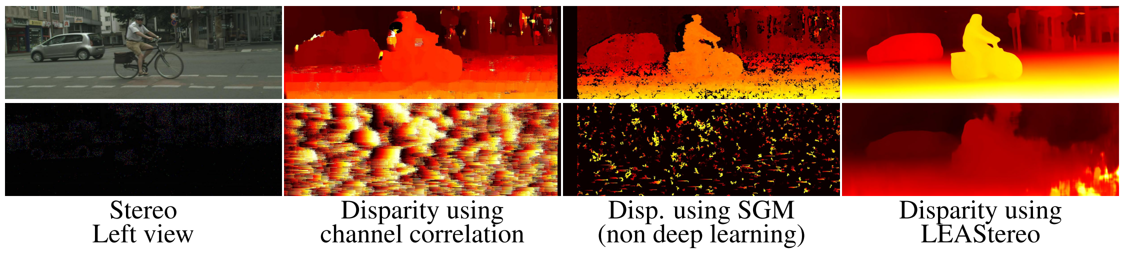

Figure 1: Top row: Disparity estimates for well lit stereo image. Bottom row: Disparity estimates for extreme low-light stereo image. Depth estimation done using (a) Simple pixel intensity correlation, (b) using non-deep learning based SGM method and (c) using deep learning based LEAstereo. Most stereo methods, which are designed for well-lit images, first estimate disparity and then use this information for the specific stereo task. However, we see that computing disparity for very noisy, low-contrast low light images could be erroneous.

1 Transforming well-lit images into low-light images

In real low-light conditions, the noise affects the image quality right from the point of image acquisition by the sensor. After the sensor acquisition, the image undergoes several transformations such as tone mapping, white balancing and gamma decompression to produce the final RGB image or more specifically the sRGB image. Thus, to transform well-lit camera images into low-light images we do not follow the crude approximation of simply darkening the sRGB image and adding White Gaussian noise. We rather follow the pipeline proposed in [1] to first transform the well-lit sRGB images into demosaiced Linear RGB images, then add the heteroscedastic noise and finally convert them back to sRGB images. As [1] was focused on well lit images, we present a brief overview of sensor acquisition to understand the nature of heteroscedastic noise to simulate low-light conditions.

Let ϕ be the number of photons striking the camera sensor. No sensor has 100% Quantum Efficiency (QE) and the number of electrons e generated are given as follows,

$$e \quad = \quad QE \cdot \phi. \tag{1}$$

The generated electrons then pass through an amplifier to convert electrons into pixel value p,

$$p \quad = \quad \frac{e}{gain}. \tag{2}$$

In low-light conditions since the number of generated electrons are less, a lower value of $gain$ is preferred, lest the camera exhibits poor sensitivity.

Input	Operation	Output
	——— Encoder ———	
l, r: $512 \times 512 \times 3$	*conv*(in=3, out=6, stride=1)	l, r: $512 \times 512 \times 6$
l, r: $512 \times 512 \times 6$	Pixel UnShuffle	l, r: $256 \times 256 \times 24$
l, r: $256 \times 256 \times 24$	*conv*(in=24, out=12, stride=1)	l, r: $256 \times 256 \times 12$
l, r: $256 \times 256 \times 12$	Pixel UnShuffle	l, r: $128 \times 128 \times 48$
l, r: $128 \times 128 \times 48$	*conv*(in=48, out=24, stride=1)	l, r: $128 \times 128 \times 24$
l, r: $128 \times 128 \times 24$	Pixel UnShuffle	l, r: $64 \times 64 \times 96$
$64 \times 64 \times 192$	*conv*(in=192, out=64, stride=1)	$64 \times 64 \times 64$
$64 \times 64 \times 64$	Pixel UnShuffle	$32 \times 32 \times 256$
$32 \times 32 \times 256$	*conv*(in=256, out=128, stride=1)	$32 \times 32 \times 128$
$32 \times 32 \times 128$	*conv*(in=128, out=128, stride=2)	$16 \times 16 \times 128$
$16 \times 16 \times 128$	*conv*(in=128, out=256, stride=1)	$16 \times 16 \times 256$
	——— Decoder ———	
$16 \times 16 \times 256$	Pixel Shuffle	$32 \times 32 \times 64$
$32 \times 32 \times 192$	*conv*(in=192, out=128, stride=1)	$32 \times 32 \times 128$
$32 \times 32 \times 128$	Pixel Shuffle	$64 \times 64 \times 32$
$64 \times 64 \times 96$	*ResBlock*(in=96, out=96)	$64 \times 64 \times 96$
$64 \times 64 \times 96$	Pixel Shuffle	$128 \times 128 \times 24$
l, r: $128 \times 128 \times 48$	*conv*(in=48, out=48, stride=1)	l, r: $128 \times 128 \times 48$
l, r: $128 \times 128 \times 48$	Pixel Shuffle	l, r: $256 \times 256 \times 12$
l, r: $256 \times 256 \times 24$	*conv*(in=24, out=12, stride=1)	l, r: $256 \times 256 \times 12$
l, r: $256 \times 256 \times 12$	Bilinear Upsampling	l, r: $512 \times 512 \times 12$
l, r: $512 \times 512 \times 18$	*conv*(in=18, out=36, stride=1)	l, r: $512 \times 512 \times 36$
l, r: $512 \times 512 \times 36$	*conv*(in=36, out=3, stride=1)	l, r: $512 \times 512 \times 3$

Table 1: Feature maps propagating through our Hybrid U-net network architecture shown in Fig.1 of the main paper. *conv:* 2D Convolution; *ResBlock:* Residual block; *Pixel Shuffle*: Upsampling operation introduced in [4]; *Pixel UnShuffle*: Reverse Pixel Shuffle operation for downsampling; *l/r*: left/right view features.

There are two main sources of noise for p: a) Photon noise which is due to randomness in photon arrival and b) Read Noise which is due to the imperfections in the electron read-out circuit. The arriving photons follow a poisson distribution and so the mean and variance are both given by ϕ. Now using Eq. 1 and Eq. 2, the variance due to photon noise in the pixel domain is given by,

$$\left(\frac{QE}{gain}\right)^2 \cdot \phi \quad = \quad \frac{QE}{gain} \cdot p. \tag{3}$$

The read-out does not depend on incoming electrons and generally modelled as gaussian noise with 0 mean and fixed variance $erms$. $erms$ is quite low for modern cameras. Due to the independent nature of photon noise and read-out noise, the noisy image can be modelled as the following heteroscedastic distribution,

$$\mathcal{N}\left(\text{mean} = p, \text{variance} = erms + \frac{QE}{gain} \cdot p\right) \tag{4}$$

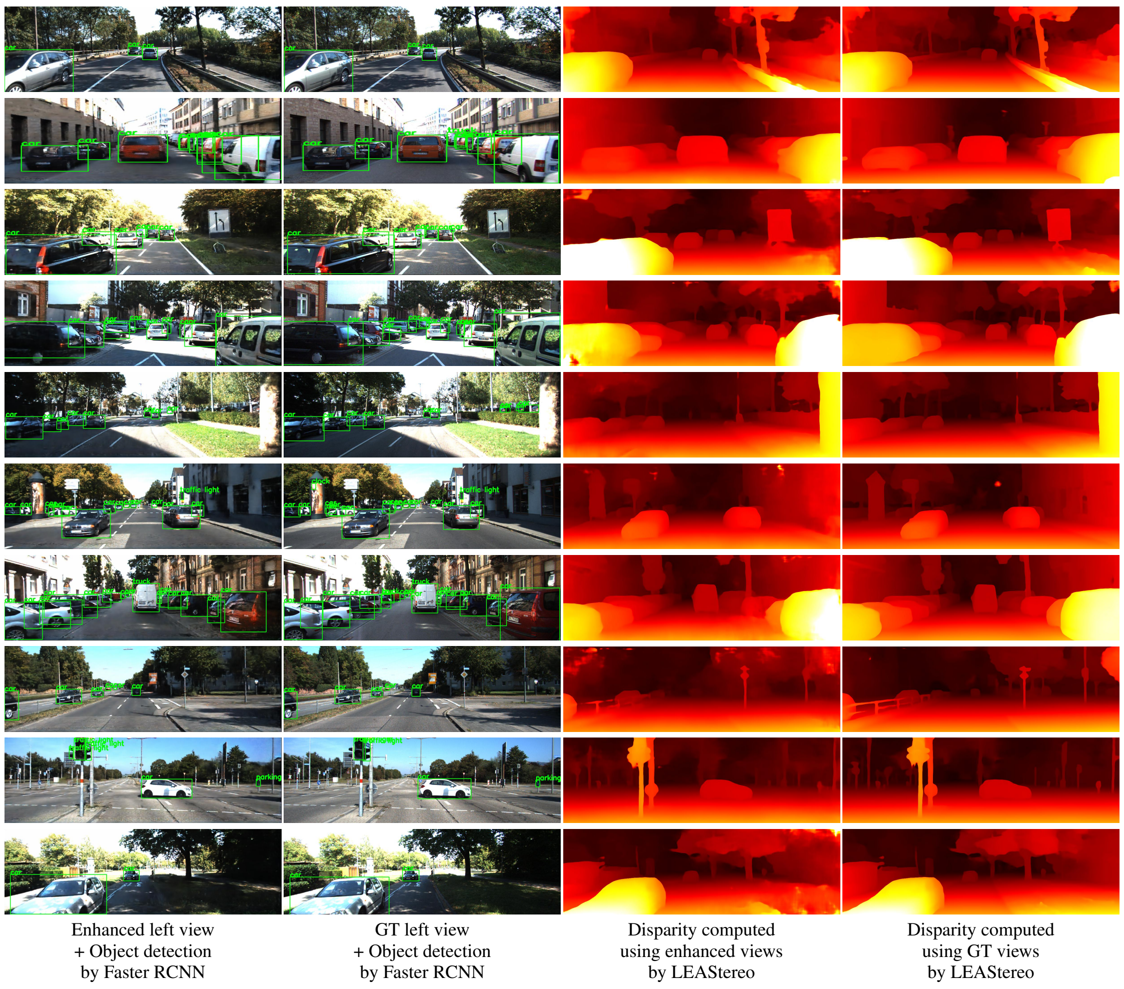

Figure 2: More qualitative results by our method on the KITTI dataset. Apart from showing enhancement results we also show the disparity computed using left/right enhanced views using LEAStereo (pre-trained on well-lit KITTI images) and object detection by passing the enhanced left view through Faster RCNN [3] (pre-trained on well-lit COCO dataset [2].)

In Eq. 4, p is the pixel value in the raw image. Most cameras demosaic this raw image using simple interpolation formulations to get the Linear RGB image. Since, the KITTI and CityScape meta data does not contain the type of interpolation done, nor could we find it from the datasheet, we approximate the raw image with the Linear RGB image only. Finally, in Eq. 4, p belongs to low-light image, but our Linear RGB is obtained from well-lit sRGB images. We thus first scale down the Linear RGB image and then sample the noisy low-light pixels from the following distribution,

$$\mathcal{N}\left(\frac{x}{scale}, erms + \frac{QE}{gain} \cdot \frac{x}{scale}\right) \tag{5}$$

where x denotes the pixels of Linear RGB image obtained from well-lit images. The image thus obtained is then converted back into sRGB image. As the lux value on a clear sunny day is atleast 500 lux and we aim for extreme low-light conditions like something around 20 lux, we use large $scale$ factor. For example in the main paper we showed results for $scale = 40$ and in Fig. 2 of this supplementary we show results when our model is trained for low-light images generated using $scale = 20$.

References

[1] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *CVPR*, 2019.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[4] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.