An Evaluation of Objective Image Quality Assessment for Thermal Infrared Video Tone Mapping
[Supplementary Material]

Michael Teutsch[1], Simone Sedelmaier[1,2], Sebastian Moosbauer[1], Gabriel Eilertsen[3], Thomas Walter[2]

[1] Hensoldt Optronics GmbH, Oberkochen, Germany
{michael.teutsch, simone.sedelmaier, sebastian.moosbauer}@hensoldt.net
[2] Ulm University of Applied Sciences, Germany
thomas.walter@thu.de
[3] Linköping University, Sweden
gabriel.eilertsen@liu.se

Abstract

In this supplementary material, we show details about the systematic image degradation performed to quantitatively evaluate the IQA measures presented in the main manuscript. Furthermore, we present pseudo code for the four considered IQA measures to provide more insight into the evaluation procedure and the pre-processing. Finally, we show some details about training the deep learning based TMO approach.

1. Degraded Images for IQA Evaluation

An overview of the applied artificial image degrading techniques used to evaluate the IQA measures is given in Fig. 1 (a-e). For over-/underexposure, we use clipping in a way that we bring more and more pixels to saturation. Clipping for loss of contrast (Fig. 1 (c)) is used vice versa so that image contrast gets lost. The example images for blur and noise speak for themselves. The temporal coherence measure is evaluated using artificial flickering as shown in Fig. 1 (e). We clip the image with random clipping direction to either enforce more saturation or reduced contrast.

2. IQA Measures Pseudo Code

In this section, the pseudo code for the four considered evaluation measures *exposure*, *contrast*, *noise visibility* and *temporal incoherence* is presented.

2.1. Over-/Underexposure Measure

The tone mapped 8-bit images are read in as grayscale images. Algorithm 1 describes the histogram based calculation of the fraction of underexposed and overexposed pixels. Then, the values are averaged over all images and at the end over all sequences. On this basis, the standard deviation can then be calculated.

Algorithm 1: Exposure measure

Input: set of sequences with tone mapped images $s_T = \{s_{1_T}, s_{2_T}, s_{3_T}, ...\}$

Function *exposure*(s_T){
  for $i \leftarrow 1$ **to** $|s_T|$ **do**
    for $j \leftarrow 1$ **to** $|s_{i_T}|$ **do**
      // calculate histogram
      hist $\leftarrow$ histogram(T_j)
      // underexposed pixels
      under_exp_pix $\leftarrow$ under_exp_pix + $\frac{\sum hist[0:0.02]}{\sum hist} \cdot 100$
      // overexposed pixels
      over_exp_pix $\leftarrow$ over_exp_pix + $\frac{\sum hist[0.95:end]}{\sum hist} \cdot 100$
    end
    // calculate average
    under_exp[i] $\leftarrow \frac{under\_exp\_pix}{|s_{i_T}|}$
    over_exp[i] $\leftarrow \frac{over\_exp\_pix}{|s_{i_T}|}$
  end
  // average over all sequences
  under_exp_all $\leftarrow \frac{\sum_{i=1}^{|s_T|} under\_exp[i]}{|s_T|}$
  over_exp_all $\leftarrow \frac{\sum_{i=1}^{|s_T|} over\_exp[i]}{|s_T|}$
}
return under_exp_all, over_exp_all

(a) over-/underexposure - clipping (0, 0.15, 0.3, 0.45)

(b) loss of contrast - Gaussian blur (0.0, 1.1, 2.0, 3.0)

(c) loss of contrast - clipping (0, 0.15, 0.3, 0.45)

(d) noise visibility - noise (no noise, Poisson noise, Gaussian noise sigma 0.03 and 0.17)

(e) temporal coherence - flickering by random clipping with maximum value 0.3

Figure 1. Overview of the artificial image quality degrading techniques used to evaluate the IAQ measures.

2.2. Loss Of Contrast

Algorithm 2: Local contrast measure

Input: set of sequences with HDR images $s_L = \{s_{1_L}, s_{2_L}, s_{3_L}, ...\}$ and tone mapped images $s_T = \{s_{1_T}, s_{2_T}, s_{3_T}, ...\}$

Function *contrast_local*(s_L, s_T){
  for $i \leftarrow 1$ **to** $|s_T|$ **do**
    for $j \leftarrow 1$ **to** $|s_{i_T}|$ **do**
      // local contrast LDR
      $c_{1_T} \leftarrow F_{bi}(T_j)$
      $c_{2_T} \leftarrow T_j - c_{1_T}$
      $c_{3_T} \leftarrow c_{3_T} + \text{mean}(L_j \cdot |c_{2_T}|)$
      // local contrast HDR
      $c_{1_L} \leftarrow F_{bi}(L_j)$
      $c_{2_L} \leftarrow L_j - c_{1_L}$
      $c_{3_L} \leftarrow c_{3_L} + \text{mean}(L_j \cdot |c_{2_L}|)$
    end
    // sequence's mean values
    $c_T[i] \leftarrow \frac{c_{3_T}}{|s_{i_T}|}$
    $c_L[i] \leftarrow \frac{c_{3_L}}{|s_{i_T}|}$
    temp[i] $\leftarrow c_T[i] - c_L[i]$
  end
  // average over all sequences
  measure $\leftarrow \frac{\sum_{i=1}^{|s_T|} temp[i]}{|s_T|}$
}
return measure

We first discuss the different pre-processing steps depending on the input format. Considering HDR images in EXR format, they are read in as grayscale images with any bit depth. Afterwards, these images are converted to 32-bit single precision floating point format. The next step is to remove all negative and zero values of the HDR image, so that a transformation into the log domain can be applied. If the HDR images are in TIFF format, the pre-processing also includes reading in as grayscale images. The images are then available in 16-bit unsigned integer format. Afterwards, a conversion into the 32-bit single precision floating point format takes place and the images are normalized. For 16-bit images, the normalization is done by a division with $2^{16} = 65,535$ and for 14-bit images with $2^{14} = 16,383$. Finally, all negative and zero values are removed before the images are transformed into the log domain.

The tone mapped images are read in as grayscale images and are then available in 8-bit unsigned integer format, since they are stored as TIFF or PNG. The conversion of the images is also done in the 32-bit single precision floating point format, so that the HDR and tone mapped LDR images are in the same format before they get processed with the contrast measure. Subsequently, the tone mapped images are normalized with $2^8 = 255$ as they are saved as 8-bit images. The last step involves removing all negative and zero values, so that the transformation into the log domain can take place. In addition, a gamma correction with $\gamma = 2.2$ is performed for the tone mapped images.

Algorithm 3: Global contrast measure

Input: set of sequences with HDR images $s_L = \{s_{1_L}, s_{2_L}, s_{3_L}, ...\}$ and tone mapped images $s_T = \{s_{1_T}, s_{2_T}, s_{3_T}, ...\}$

Function *contrast_global*(s_L, s_T){
  for $i \leftarrow 1$ **to** $|s_T|$ **do**
    for $j \leftarrow 1$ **to** $|s_{i_T}|$ **do**
      // global contrast LDR
      $g_{1_T} \leftarrow F_g(T_j)$
      $g_{2_T} \leftarrow g_{1_T} \cdot g_{1_T}$
      $g_{3_T} \leftarrow F_g(T_j \cdot T_j)$
      $g_{4_T} \leftarrow \sqrt{g_{3_T} - g_{2_T}}$
      $g_T \leftarrow g_T + \text{mean}(g_{4_T})$
      // global contrast HDR
      $g_{1_L} \leftarrow F_g(L_j)$
      $g_{2_L} \leftarrow g_{1_L} \cdot g_{1_L}$
      $g_{3_L} \leftarrow F_g(L_j \cdot L_j)$
      $g_{4_L} \leftarrow \sqrt{g_{3_L} - g_{2_L}}$
      $g_L \leftarrow g_L + \text{mean}(g_{4_L})$
    end
    // sequence's mean values
    $c_T[i] \leftarrow \frac{g_T}{|s_{i_T}|}$
    $c_L[i] \leftarrow \frac{g_L}{|s_{i_T}|}$
    temp[i] $\leftarrow c_T[i] - c_L[i]$
  end
  // average over all sequences
  measure $\leftarrow \frac{\sum_{i=1}^{|s_T|} temp[i]}{|s_T|}$
}
return measure

The implementation design of the local contrast measure is represented in Algorithm 2 using the bilateral filter function F_{bi}. For the diameter of each pixel neighborhood used during filtering, a diameter is calculated from the value of $\sigma_{sp} = 10$. σ_c of the bilateral filter is set to 0.2. The loss of local contrast is finally the average of all images in a sequence and then averaged over all sequences.

Algorithm 3 describes the implementation of the global contrast measure using the Gaussian filter function F_g. The kernel size is defined with a width and height of 9 and the standard deviation in horizontal direction is set to 3. Consequently, the standard deviation in vertical direction is also 3. As with local contrast, all images in a sequence are aver-

aged and finally all sequences are averaged.

2.3. Noise Visibility Measure

Algorithm 4: Noise visibility measure

Input: set of sequences s_L, s_T, $s_{\hat{L}}$, $s_{\hat{T}}$
Function *noise_visibility*(s_L, s_T, $s_{\hat{L}}$, $s_{\hat{T}}$){
  for $i \leftarrow 1$ **to** $|s_T|$ **do**
    for $j \leftarrow 1$ **to** $|s_{i_T}|$ **do**
      if *HDR image format* $\widehat{=}$ *exr* **then**
        Do read reference HDR image L_j
        $L \leftarrow \frac{L_j}{\max(L_j)}$
        Do read test HDR image $\hat{L}_j$
        $\hat{L} \leftarrow \frac{\hat{L}_j}{\max(\hat{L}_j)}$
      end
      if *HDR image format* $\widehat{=}$ *tif* **then**
        Do read reference HDR image L_j
        $L \leftarrow \frac{L_j}{65535}$
        Do read test HDR image $\hat{L}_j$
        $\hat{L} \leftarrow \frac{\hat{L}_j}{65535}$
      end
      Do read reference tone mapped image T_j
      $T \leftarrow \frac{T_j}{255}$
      Do read test tone mapped image $\hat{T}_j$
      $\hat{T} \leftarrow \frac{\hat{T}_j}{255}$
      `// get metric` Θ
      $\Theta_{HDR} \leftarrow$ hdrvdp($30 \cdot \hat{L}$, $30 \cdot L$, 'luminance', 30)
      $\Theta_{TM} \leftarrow$ hdrvdp($30 \cdot \hat{T}$, $30 \cdot T$, 'luminance', 30)
      `// noise visibility`
      $n \leftarrow n + (\Theta_{HDR} - \Theta_{TM})$
    end
    temp$[i] \leftarrow \frac{n}{|s_{i_T}|}$
  end
  measure $\leftarrow \frac{\sum_{i=1}^{|s_T|} temp[i]}{|s_T|}$
}
return measure

Algorithm 4 describes the implementation of the noise visibility measure. If the input HDR images are available in the EXR format, they can be read in for example with the help of the HDR Toolbox provided by Banterle et al. [1]. Afterwards, the images are normalized with their individual maximum value. For HDR images in the TIFF format, a standard function can be used to read them in. The HDR images get converted from 16-bit unsigned integer to double and normalized with 65,535 or 16,383 depending on their bit depth. In the same way, the tone mapped images are converted from 8-bit unsigned integer to double and normalized with 255, so that the values are between 0 and 1. The HDR-VDP metric offers several options as to how the input images can be interpreted. For grayscale images, the option `luminance` is chosen, which identifies images that contain absolute luminance values and exactly one color channel. Other options are for example `sRGB-display`, which should be used if the color encoding is adapted for standard LDR color images, where the maximum pixel value must be 1 and not 256. Furthermore, the option `rgb-bt.709` is used to adjust the color encoding for HDR images. After the noise visibility has been calculated, the mean value of all images in a sequence and finally of all sequences is calculated.

2.4. Temporal Incoherence Measure

Again, we first discuss the different pre-processing steps depending on the input format. If the HDR image format is the EXR format, they can be read with the help of the HDR Toolbox provided by Banterle et al. [1]. Images with the standard formats like TIFF can be read in with a standard function. Subsequently, the HDR images are converted to double, normalized to an interval of $[0, 1]$ and converted to grayscale images. The tone mapped images are read in like the HDR images, but an additional gamma correction is performed. By definition, the input of the temporal incoherence measure is a stack of images. For this evaluation framework, the temporal radius of the neighborhood of each image pair to be considered is set to 5. Therefore, the inputs of the measure are 11 image pairs. As a second pre-processing step, the images are transformed into the log domain. To do this, all negative and zero values of the images have to be removed as otherwise the logarithm is not defined. The image stacks are then the basis to calculate the global and local temporal incoherence measure independently of each other.

Algorithm 5 shows the calculation of the global incoherence coefficient. The input is a stack of HDR images and their corresponding tone mapped images. If each stack contains eleven images, the variable d is defined with 5. For the local incoherence measure the same succession is performed, but the calculations are done elementwise as described above.

3. Training the TMO DCNN

The described deep learning framework based on the Context Aggregation Network (CAN24_AN) architecture and model provided by Chen *et al.* [2] is pre-trained only with 8-bit visual-optical images. Therefore, additional training with IR data is necessary. We use the original training code provided with the paper. However, it has to be no-

Algorithm 5: Global temporal incoherence

Input: paths to HDR images and tone mapped images

Function *global_incoherence_measure(HDR_stack, TM_stack)*{

$d \leftarrow \lfloor \frac{stack\_size}{2} \rfloor$

$$\boldsymbol{\mu_L} \leftarrow \begin{pmatrix} \mu_{L_{-d}} \\ \vdots \\ \mu_{L_d} \end{pmatrix}, \qquad \boldsymbol{\mu_T} \leftarrow \begin{pmatrix} \mu_{T_{-d}} \\ \vdots \\ \mu_{T_d} \end{pmatrix}$$

$$\boldsymbol{X} \leftarrow \begin{pmatrix} -d & \cdots & d \end{pmatrix}$$

$$w_L \leftarrow \frac{\sum_{k=t-d}^{t+d} \boldsymbol{\mu_L^T} \cdot \boldsymbol{X}}{\boldsymbol{X} \cdot \boldsymbol{X}^T}, \qquad w_T \leftarrow \frac{\sum_{k=t-d}^{t+d} \boldsymbol{\mu_T^T} \cdot \boldsymbol{X}}{\boldsymbol{X} \cdot \boldsymbol{X}^T}$$

$$\boldsymbol{K} \leftarrow \begin{pmatrix} \frac{1}{2d+1} \\ \vdots \\ \frac{1}{2d+1} \end{pmatrix}$$

$$\boldsymbol{y_L} \leftarrow w_L \cdot \boldsymbol{X} + \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot \boldsymbol{\mu_L}, \quad \boldsymbol{y_T} \leftarrow w_T \cdot \boldsymbol{X} + \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot \boldsymbol{\mu_T}$$

$$\hat{\boldsymbol{t}}_{\boldsymbol{L}} \leftarrow \boldsymbol{\mu_L} - \boldsymbol{y_L^T}, \qquad \hat{\boldsymbol{t}}_{\boldsymbol{T}} \leftarrow \boldsymbol{\mu_T} - \boldsymbol{y_T^T}$$

$$\sigma_L^2 \leftarrow \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot (\hat{\boldsymbol{t}}_{\boldsymbol{L}})^2, \quad \sigma_T^2 \leftarrow \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot (\hat{\boldsymbol{t}}_{\boldsymbol{T}})^2$$

$$\tilde{\boldsymbol{t}}_{\boldsymbol{L}} \leftarrow \frac{\hat{\boldsymbol{t}}_{\boldsymbol{L}} \cdot \sigma_T}{\sigma_L}$$

$$\boldsymbol{t_L} \leftarrow 0.25 \cdot \boldsymbol{X}^T + \tilde{\boldsymbol{t}}_{\boldsymbol{L}}, \quad \boldsymbol{t_T} \leftarrow 0.25 \cdot \boldsymbol{X}^T + \hat{\boldsymbol{t}}_{\boldsymbol{T}}$$

$$q_1 \leftarrow \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot (\boldsymbol{t_L})^2$$

$$q_2 \leftarrow \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot (\boldsymbol{t_T})^2$$

$$q_3 \leftarrow \sum_{k=t-d}^{t+d} \boldsymbol{K} \cdot \boldsymbol{t_L} \cdot \boldsymbol{t_T}$$

$$cf_{LT}^{global} \leftarrow 1 - \max\left(0, \frac{q_3}{\sqrt{q_1 \cdot q_2}}\right)$$

}

return cf_{LT}^{global}

ticed that this network shows boundary artifacts (Fig. 3 after epoch 10), which is also illustrated in the paper of Wu *et al*. [5]. A further development of the CAN24_AN model is provided by Wu *et al*., but for the operator of multiscale tone manipulation (which is the considered task closest to tone mapping) their network shows worse results compared to the CAN24_AN, which is why it is not further investigated in this paper.

Table 1. FLIR dataset [4] split into training, validation, and test.

Set	#Frames	Resolution
Train	8,862	640×512
Val	1,366	640×512
Test	4,224	640×512

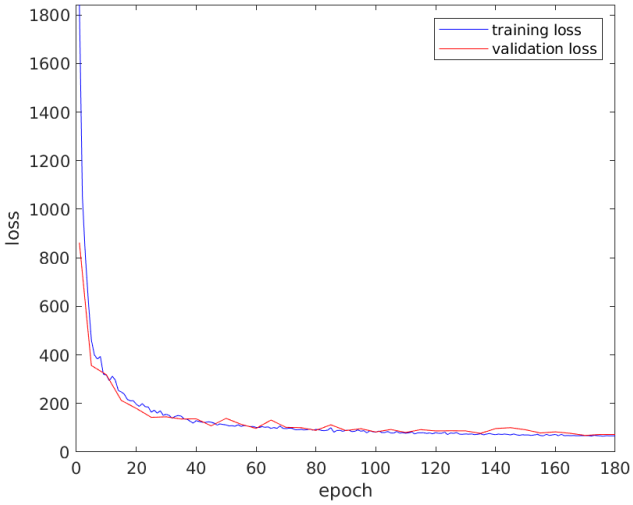

Figure 2. Loss curve of the training and the validation loss for the DCNN based approach [2] trained on the FLIR training subdataset [4]. Both curves converge well. This shows that our DCNN does not suffer from overfitting even after epoch 180.

The FLIR dataset with the training, validation and test split described in Table 1 is used for training the CAN24_AN model [2] with a learning rate of 0.001. We train for 180 epochs and the loss curve is shown in Fig. 2. Figure 3 shows example images after epoch 10, after epoch 125, and after epoch 180, and processed with the reference TMO [3]. After epoch 10, boundary artifacts and halos are visible which disappear after epoch 125. Noise is further reduced between epoch 125 and 180. In general, the example image is still noisy after epoch 180, but this is also the case with the image processed with the reference TMO. The contrast enhancement between the epochs can be seen on the wheels close to the right image border.

It could be expected that the processing time decreased when using the DCNN for tone mapping instead of the reference TMO as stated by Chen *et al*. [2] or Wu *et al*. [5]. However, we did not evaluate the runtime and we did not try to optimize the DCNN in any way using quantization or pruning techniques for example.

Figure 3. Evolution of the DCNN's [2] TMO quality during transfer learning along the epochs. The clearly visible halos after epoch 10 disappear after epoch 125. However, noise is still more visible compared to the reference TMO. This gets better after epoch 180. Training is stopped after epoch 180.

References

[1] Francesco Banterle, Alessandro Artusi, Kurt Debattista, and Alan Chalmers. *Advanced high dynamic range imaging*. AK Peters/CRC Press, 2017. 4

[2] Qifeng Chen, Jia Xu, and Vladlen Koltun. Fast image processing with fully-convolutional networks. In *IEEE ICCV*, 2017. 4, 5, 6

[3] Gabriel Eilertsen, Rafał K. Mantiuk, and Jonas Unger. Real-time noise-aware tone mapping. *ACM Transactions on Graphics (TOG)*, 34(6):198:1–198:15, 2015. 5

[4] FLIR Systems. FREE FLIR Thermal Dataset for Algorithm Training. https://www.flir.com/oem/adas/adas-dataset-form/. [Accessed 8 March 2020]. 5

[5] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *IEEE CVPR*, 2018. 5