Computational Photography Assignment #4 by Junekyo Jhung

Linearize Rendered Images

Description

HDR 이미징은 셔터 속도를 다양화하여 그에 따른 다양한 노출 시간을 통해 얻은 여러 장의 Low Dynamic Range 이미지를 병합하는 기법이다. 과제 지시 사항에서 언급한 것과 같이, dcraw를 이용하여 .NEF 이미지를 다음과 같은 환경 설정을 거친 후 .TIFF 이미지로 생성한다.

- Linear 16-bit
- White balancing using the camera's profile
- Demosaicing using high-quality interpolation.
- sRGB as the output color space.

위 4가지 사항을 충족하기 위해, 링크(https://dcraw.en.softonic.com/)에서 dcraw를 다운로드, 설치 및 실행한 뒤, dcraw documentation을 참고하여 conversion을 실시한다.

예) dcraw –w –o 1 –q 3 -4 –T exposure1.NEF

[Flags 정보]

-w: 카메라의 white balance 사용.

-o 1: 출력 colorspace를 sRGB로 설정.

-q 3: AHD(Adaptive Homogeneity-Directed) interpolation 적용.

-q 0: high-speed, low-quality bilinear interpolation.

-q 1: VNG(Variable Number of Gradients) interpolation.

-q 2: PPG(Patterned Pixel Grouping) interpolation.

-4: Linear 16-bit

-T: .TIFF 파일 쓰기.

dcraw documentation

```
-v        Print verbose messages
-c        Write image data to standard output
...
-w        Use camera white balance, if possible
-a        Average the whole image for white balance
-A <x y w h> Average a grey box for white balance
...
-t [0-7]  Flip image (0=none, 3=180, 5=90CCW, 6=90CW)
-o [0-6]  Output colorspace (raw,sRGB,Adobe,Wide,ProPhoto,XYZ,ACES)
-o <file> Apply output ICC profile from file
...
-q [0-3]  Set the interpolation quality
-h        Half-size color image (twice as fast as "-q 0")
...
-4        Linear 16-bit, same as "-6 -W -g 1 1"
-T        Write TIFF instead of PPM
```

Results

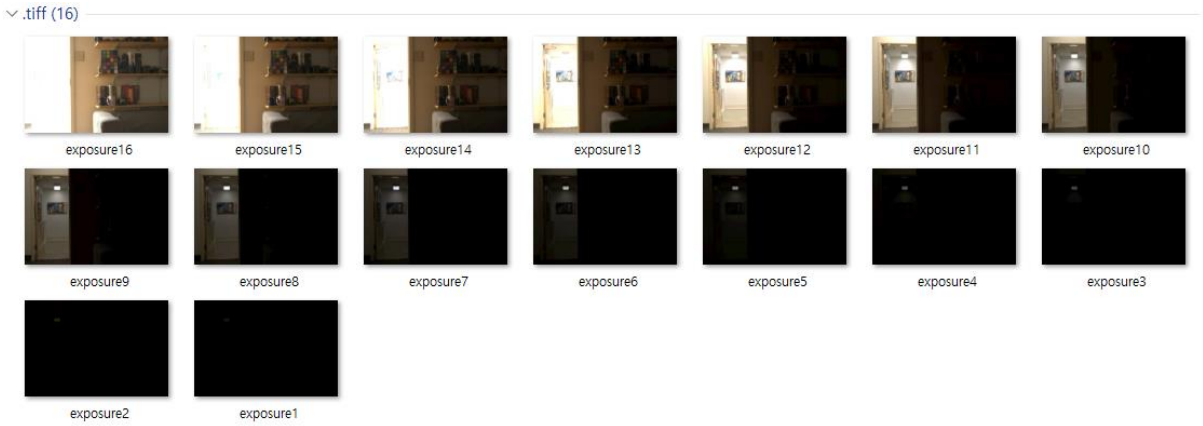

Figure 1. TIFF images after conversion from NEF using dcraw

Linearize Rendered Images

Description

이미지를 HDR로 병합하는 방법에는 크게 Linear Merging과 Logarithmic Merging 2가지가 있다.

$$I_{ij,HDR} = \frac{\sum_k w\left(I_{ij,LDR}^k\right)\frac{I_{ij}^k}{t^k}}{\sum_k w\left(I_{ij,LDR}^k\right)}$$

Figure 2. Linear Merging Eq.

$$\mathrm{I}_{\mathrm{ij,HDR}} = \exp\left(\frac{\sum_k w\left(I_{ij,LDR}^k\right)\left(I_{ij}^k - \log t^k\right)}{\sum_k w\left(I_{ij,LDR}^k\right)} \right)$$

Figure 3. Logarithmic Merging Eq.

여기에서 노출된 픽셀을 강조하는 값인 w(weight)를 적용하는 방법인 Weighting Scheme에는 아래 그림과 같이 Uniform, Tent, 그리고 Gaussian 총 3가지가 있다. 본 보고서에서는 (a) uniform과 (b) tent 2가지 weighting scheme을 사용한다.

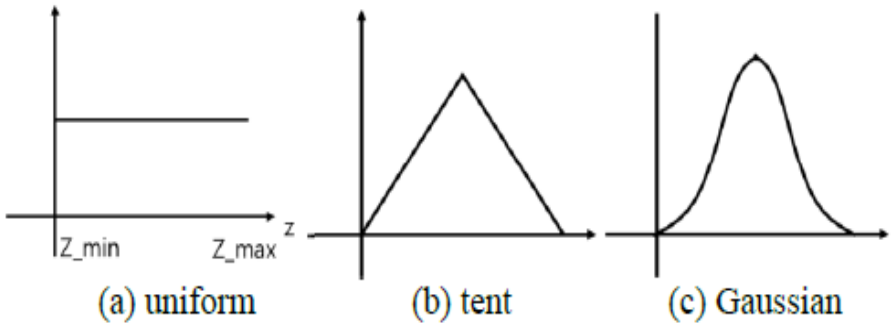

Figure 4. Weighting Scheme (a) uniform (b) tent (c) Gaussian

위 그림 Figure 2와 3의 공식을 알 수 있듯이 두 merging 방법의 큰 차이점은 바로 exposure time의 적용 방식이다. Linear와 달리 Logarithmic은 이름처럼 exposure time에 로그 함수를 씌워 이미지 픽셀에 뺀 뒤 exponential 값이 merged HDR이 됨을 확인할 수 있다.

MATLAB

```
% Tent weighting scheme
for i=1:256
    if i > 0.5 * (z_min + z_max)
        weights(i) = ((z_max - i) + 1);
    else
        weights(i) = ((i - z_min) + 1);
    end
end
...
z_r = zeros(N, count_exposure);
z_g = zeros(N, count_exposure);
z_b = zeros(N, count_exposure);

for i=1:count_exposure
    img = imread(img_list{i});
    r_ch = img(:,:,1);
    g_ch = img(:,:,2);
    b_ch = img(:,:,3);
    z_r(:,i) = r_ch(indices);
    z_g(:,i) = g_ch(indices);
    z_b(:,i) = b_ch(indices);
end

B = zeros(size(z_r,1) * size(z_r,2), count_exposure);

% Logarithmic merging
for i = 1:count_exposure
    B(:,i) = log(expo_list(i));
end

% gsolve
[g_r,lE_r]=gsolve(z_r, B, 50, weights);
[g_g,lE_g]=gsolve(z_g, B, 50, weights);
[g_b,lE_b]=gsolve(z_b, B, 50, weights);

img_hdr = HDR(img_list, g_r, g_g, g_b, weights, B);
figure;
imshow(img_hdr);
hdrwrite(img_hdr, 'Result_HDR.hdr');
```

Results

HDRs

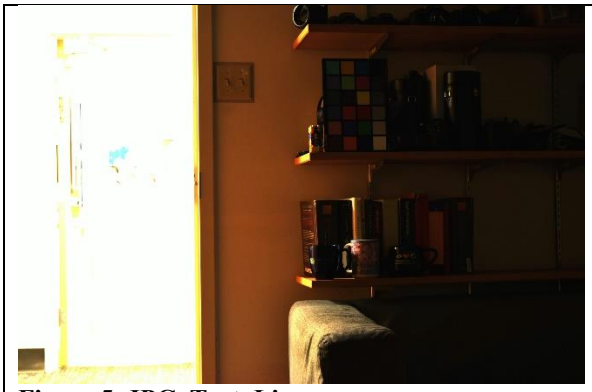

Figure 5. JPG, Tent, Linear

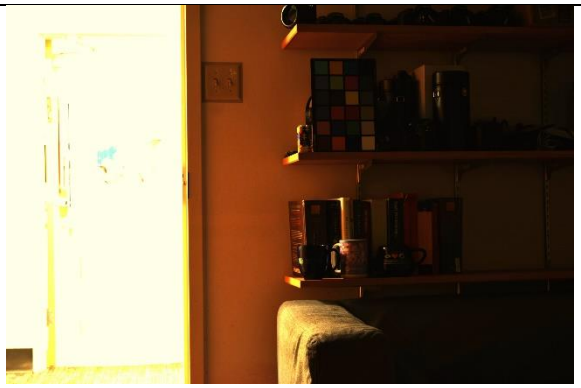

Figure 6. TIFF, Tent, Linear

Figure 7. JPG, Tent, Log

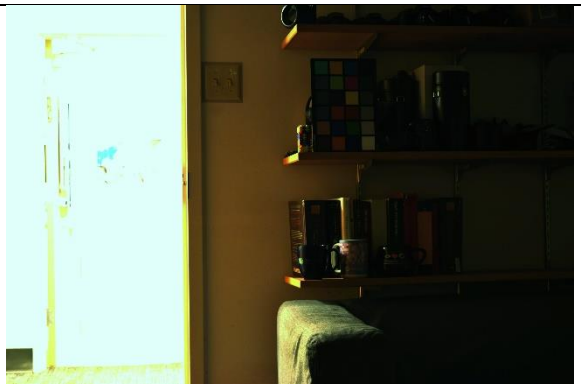

Figure 8. TIFF, Tent, Log

Figure 9. JPG, Uniform, Linear

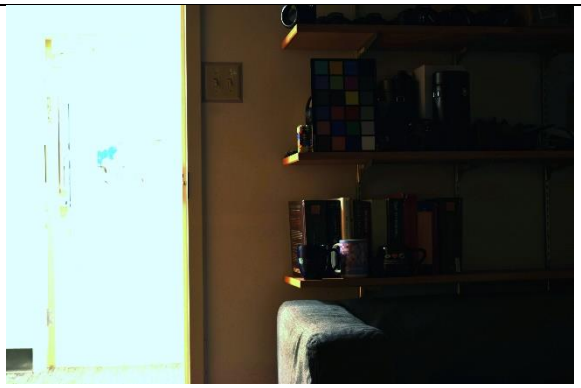

Figure 10. TIFF, Uniform, Linear

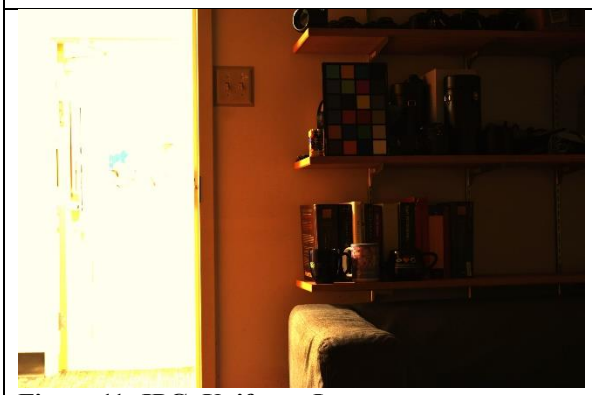

Figure 11. JPG, Uniform, Log

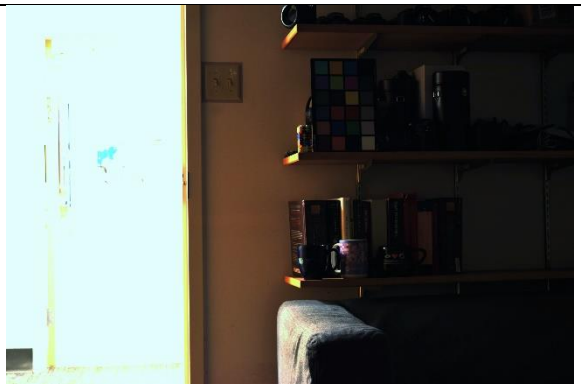

Figure 12. TIFF, Uniform, Log

Weighting Schemes

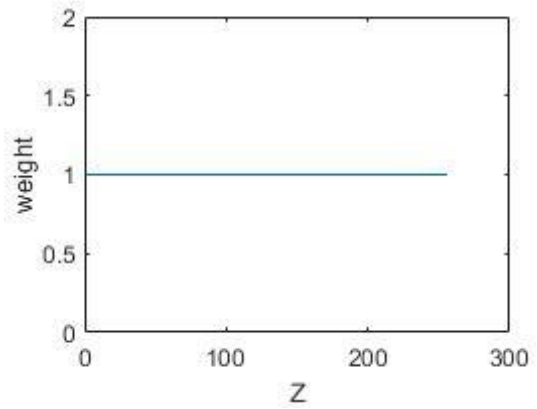

Figure 13. Z vs w with Uniform Weighting Scheme

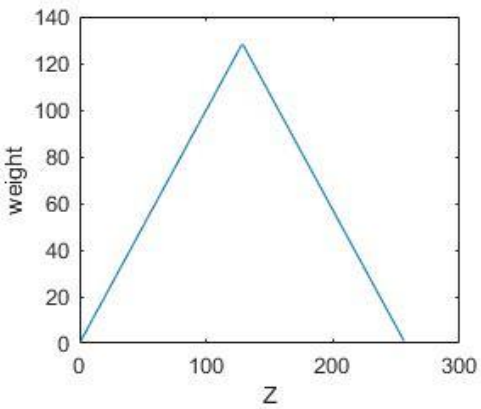

Figure 14. Z vs w with Tent Weighting Scheme

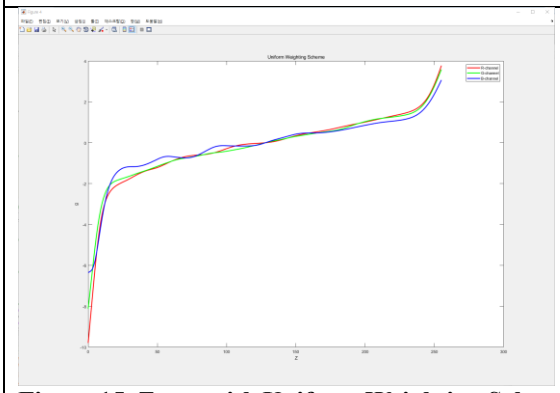

Figure 15. Z vs g with Uniform Weighting Scheme

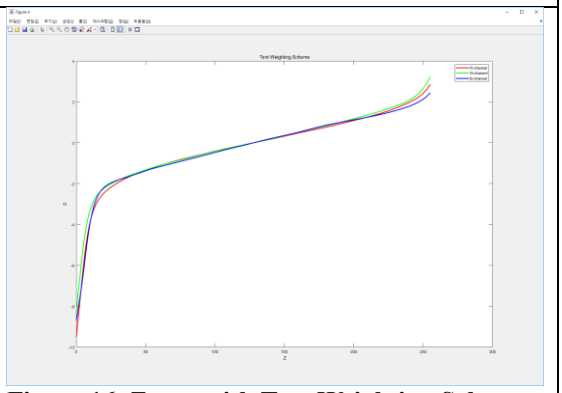

Figure 16. Z vs g with Tent Weighting Scheme

** Color Evaluation을 위해 rgb2xyz 함수에서 color space인 srgb, adobe-rgb-1998, linear-rgb 중 linear-rgb를 사용하는 것이 요구되었으나 matlab에서 해당 명령어에서 오류가 발생하여 진행하지 못하였다.

Photographic Tonemapping

Description

Reinhard가 제안한 Photographic Tonemapping을 구현하기 위해 다음과 같은 공식을 적용한다.

$$I_{\mathrm{ij,Tonemapping}} = \frac{\tilde{I}_{\mathrm{ij,HDR}}\left(1+\frac{\tilde{I}_{\mathrm{ij,HDR}}}{\tilde{I}^2_{\mathrm{white}}}\right)}{1+\tilde{I}_{\mathrm{ij,HDR}}}$$

$$\tilde{I}_{\mathrm{white}} = \mathrm{B} \cdot \max_{i,j}(\tilde{I}_{\mathrm{ij,HDR}})$$

$$\tilde{I}_{\mathrm{ij,HDR}} = \frac{K}{I_{\mathrm{m,HDR}}} I_{\mathrm{ij,HDR}}$$

$$I_{\mathrm{m,HDR}} = \exp\left(\frac{1}{N}\sum_{i,j}\log\left(I_{ij,HDR}+\varepsilon\right)\right)$$

Figure 17. Equations Proposed by Reinhard

이를 적용하는 방법은 크게 두가지 방법으로 RGB와 Luminance 맵핑이 있다. Luminance의 경우 RGB HDR 이미지를 xyY로 변환하여 Y 값을 변환한 뒤, RGB로 재변환하면 된다. 또한, 위 공식에서 볼 수 있듯이 밝기(K, key)와 대조(B, contrast)를 조정하여 결과물을 다양하게 얻을 수 있다. 사용된 원본 HDR 이미지는 JPG 이미지를 Tent Weighting Scheme과 Logarithmic Merging을 통해 생성된 이미지를 사용하였으며 아래의 그림과 같다.

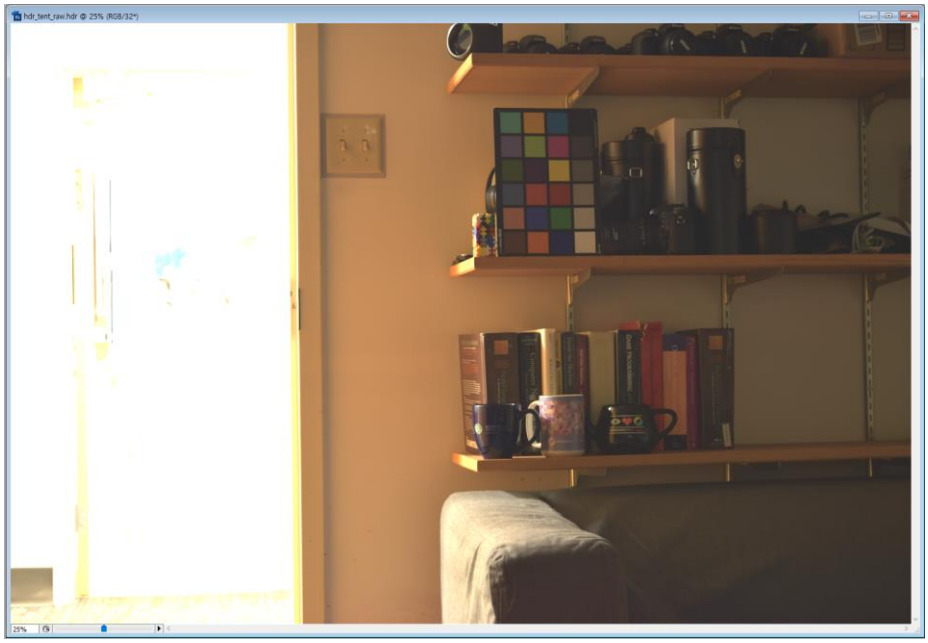

Figure 18. Original HDR Image used for Tonemapping (JPG, tent weighting scheme, logarithmic merging)

MATLAB

```
function [ img_PM_RGB, img_PM_LUM ] = PM(K, B, h, w)
hdr_img = hdrread('Result_HDR.hdr');
%% RGB
img_PM_RGB = zeros(h, w, 3);
eps = 1e-10;

for i = 1:3
    img_o = hdr_img(:, :, i);
    img_m = exp((1 / (h * w)) * sum(log(img_o(:) + eps)));
    img_ij = (K / img_m )* img_o;
    img_w = B * max(img_ij(:));
    img_tm = img_ij .* (1 + (img_ij ./ (img_w ^ 2))) ./ (1 + img_ij);
    img_PM_RGB(:, :, i) = img_tm(:, :);
end

%% Luminance
img_PM_LUM = zeros(h, w, 3);
% xyY
hdr_xyY = zeros(h, w, 3);
hdr_xyz = rgb2xyz(hdr_img, 'ColorSpace', 'srgb');
hdr_xyY(:, :, 1) = hdr_xyz(:, :, 1) ./ (hdr_xyz(:, :, 1) + hdr_xyz(:, :,
2) + hdr_xyz(:, :, 3));
hdr_xyY(:, :, 2) = hdr_xyz(:, :, 2) ./ (hdr_xyz(:, :, 1) + hdr_xyz(:, :,
2) + hdr_xyz(:, :, 3));
hdr_xyY(:, :, 3) = hdr_xyz(:, :, 2);

img_o = hdr_xyY(:, :, 3);
img_m = exp((1 / (h * w)) * sum(log(img_o(:) + eps)));
img_ij = (K / img_m )* img_o;
img_w = B * max(img_ij(:));
img_tm = img_ij .* (1 + (img_ij ./ (img_w ^ 2))) ./ (1 + img_ij);

% RGB
img_PM_LUM(:, :, 1) = img_tm(:, :) .* hdr_xyY(:, :, 1) ./ hdr_xyY(:, :,
2);
img_PM_LUM(:, :, 2) = img_tm(:, :);
img_PM_LUM(:, :, 3) = img_tm(:, :) .* (1 - hdr_xyY(:, :, 1) -
hdr_xyY(:, :, 2)) ./ hdr_xyY(:, :, 2);

img_PM_LUM = xyz2rgb(img_PM_LUM);

end
```

Results

논문에서 제안한 파라미터 값은 K = 0.15 와 B = 0.95이며, RGB와 Luminance의 결과물은 다음과 같다.

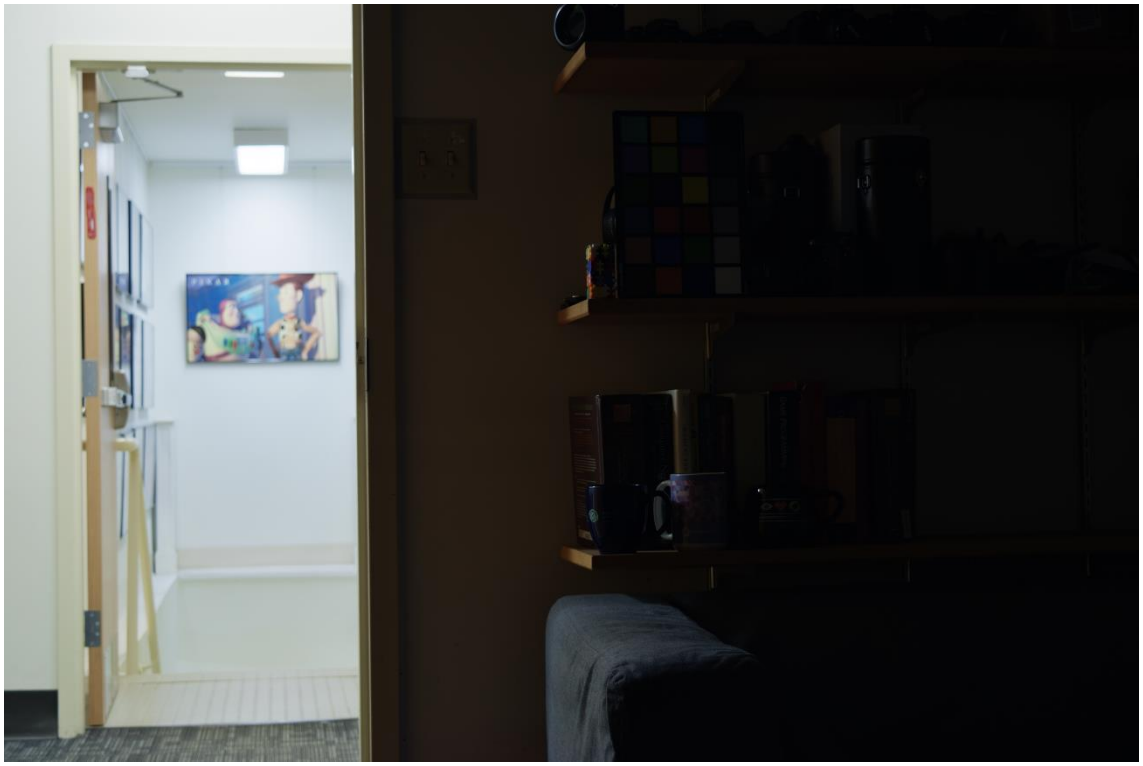

Figure 19. RGB Tonemapped HDR Image with K=0.15 & B=0.95

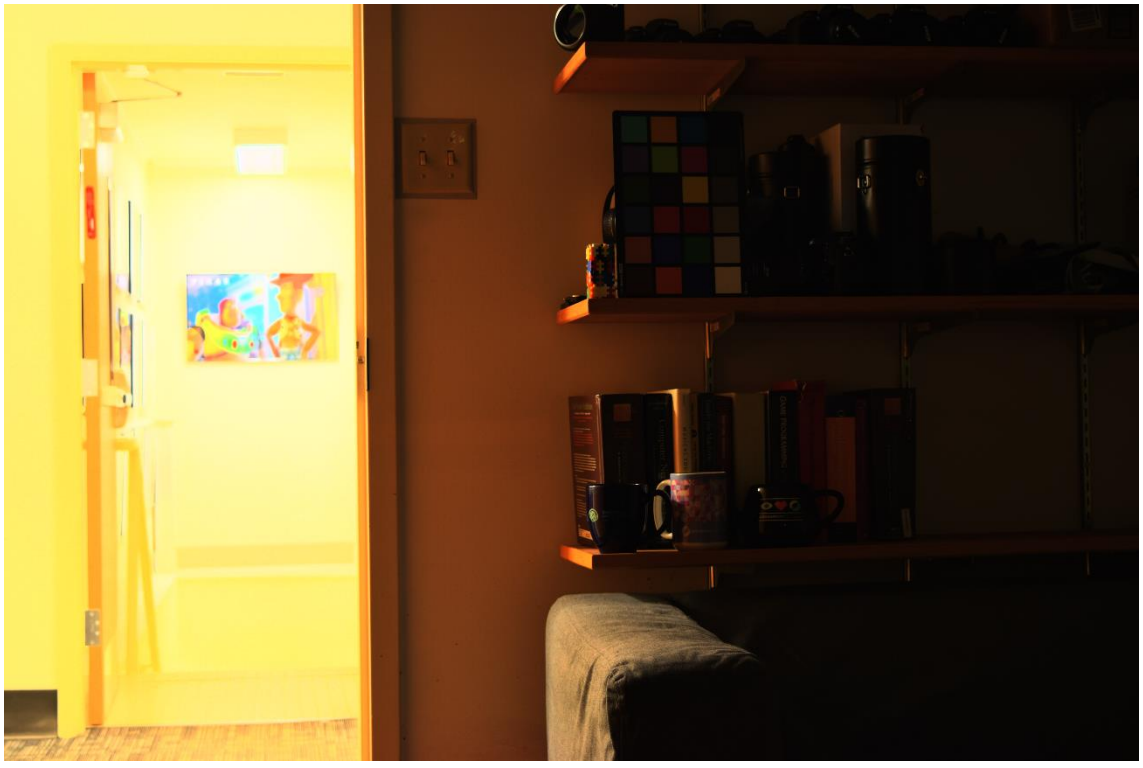

Figure 20. Luminance Tonemapped HDR Image with K=0.15 & B=0.95

변수 K와 B 중 B 값을 0.95로 고정한 후 K 값만을 변화한 결과는 다음과 같다.

RGB (B = 0.95, constant)

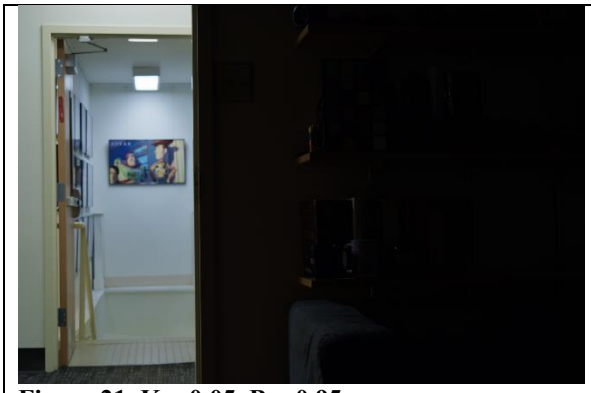

Figure 21. K = 0.05, B = 0.95

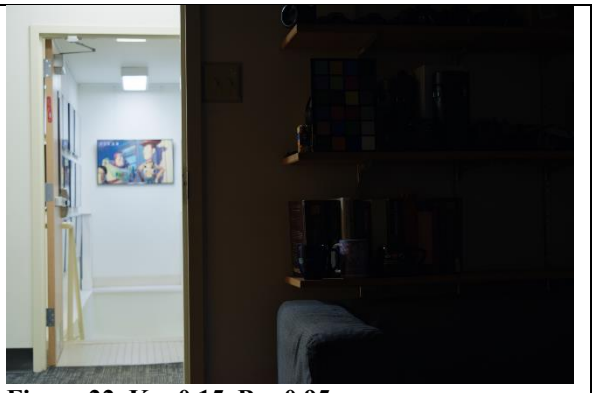

Figure 22. K = 0.15, B = 0.95

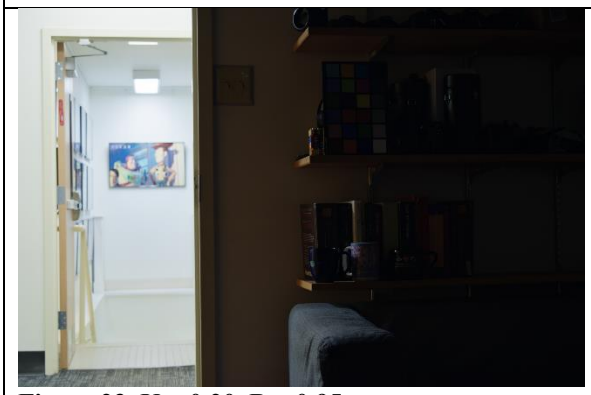

Figure 23. K = 0.20, B = 0.95

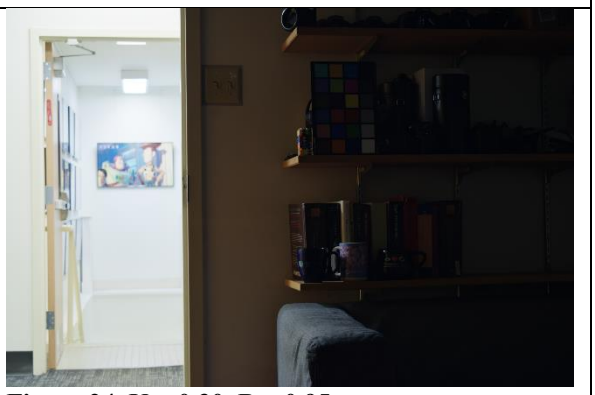

Figure 24. K = 0.30, B = 0.95

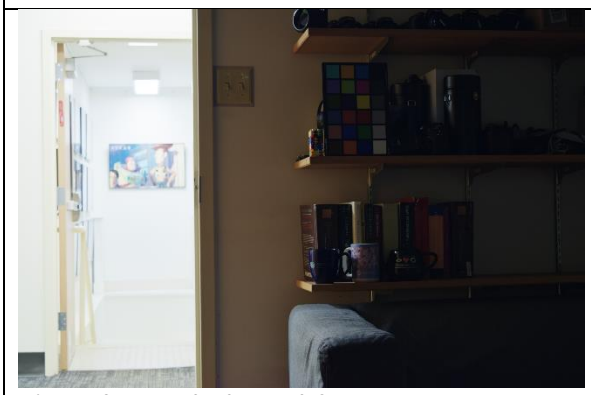

Figure 25. K = 0.50, B = 0.95

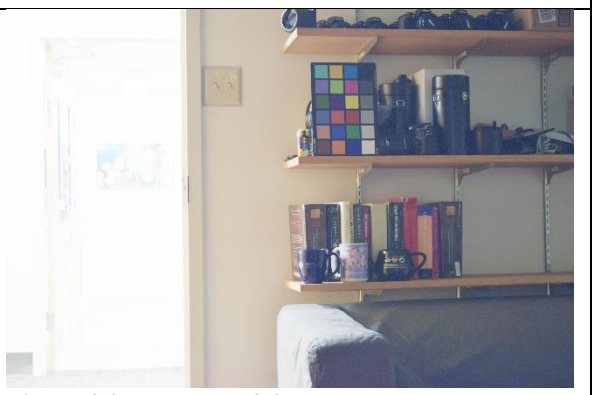

Figure 26. K = 5, B = 0.95

Luminance (B = 0.95, constant)

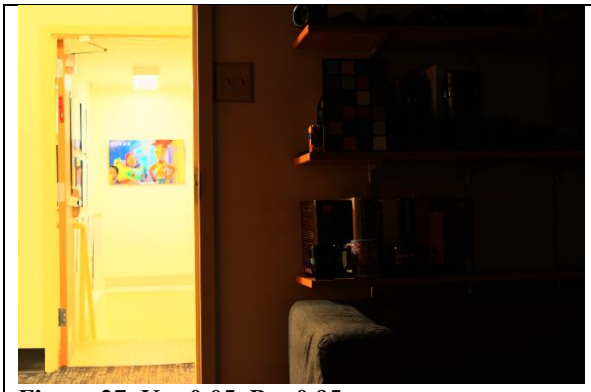

Figure 27. K = 0.05, B = 0.95

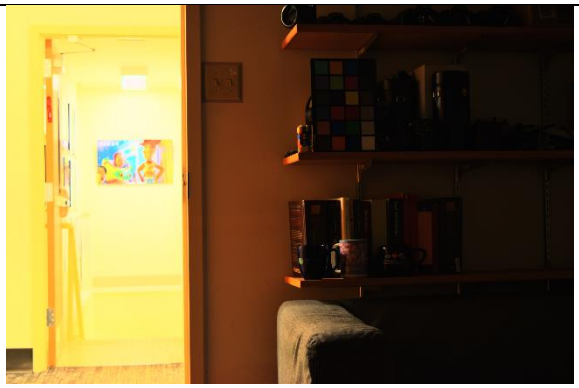

Figure 28. K = 0.15, B = 0.95

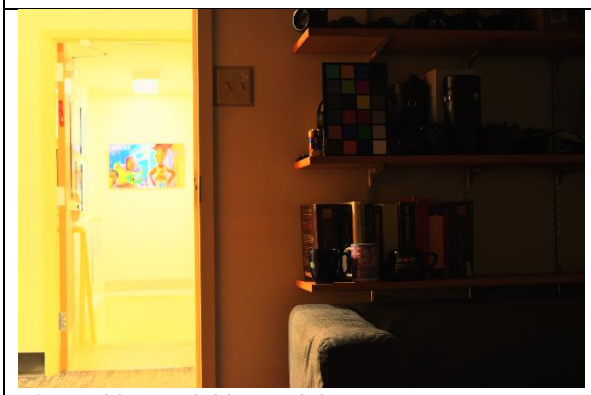

Figure 29. K = 0.20, B = 0.95

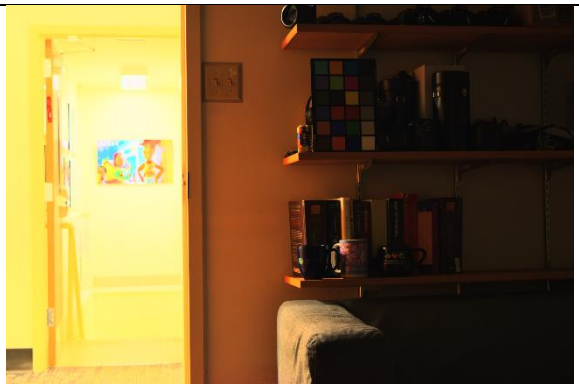

Figure 30. K = 0.30, B = 0.95

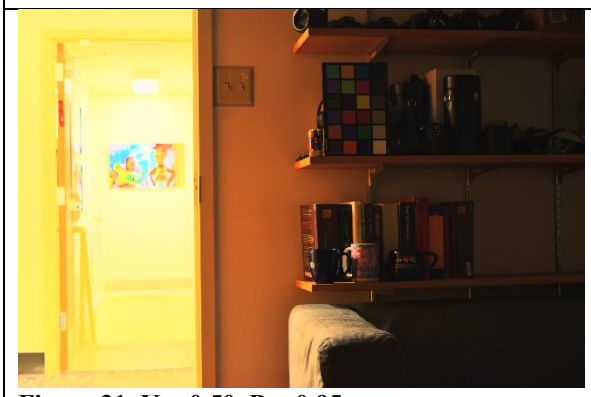

Figure 31. K = 0.50, B = 0.95

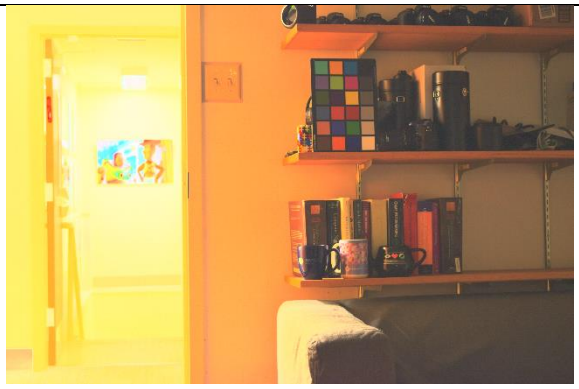

Figure 32. K = 5, B = 0.95

정의된 것과 같이 이미지의 밝기를 결정하는 parameter인 K(key) 값이 증가함에 따라 이미지의 밝기가 전체적으로 증가하는 것을 볼 수 있다. B = 0.95로 고정 값일 경우, RGB에선 K = 0.2, Luminance에선 K = 0.15 일 때 결과 이미지가 가장 자연스러운 것으로 판단된다.

반면, B 값의 변화가 100 단위의 scale factor가 아닌 경우엔 큰 차이가 없어 논문에서 제안한 0.95를 최종 값으로 사용하기로 판단했다.

Tonemapping using Bilateral Filtering

Description

Bilateral Filtering을 이용한 Tonemapping을 위해서는 다음과 같은 절차를 진행한다.

1. Compute the log intensity, $L_{ij} = \log(I_{ij,HDR})$.
2. Compute the base layer using bilatering filtering, $B_{ij} = bilatering\ filter(L_{ij})$.
3. Compute the detail layer, $D_{ij} = L_{ij} - B_{ij}$.
4. Apply an offset and a scale S to the base, $B'_{ij} = \mathrm{S} \cdot \left(B_{ij} - \max_{i,j}(B_{ij})\right)$.
5. Reconstruct the intensity, $I_{ij,TM} = \exp(B'_{ij} + D_{ij})$.

Figure 33. Algorithm of Tonemapping using Bilateral Filter

본 Tonemapping에서 사용된 원본 HDR Image는 위 Photographic Tonemapping에서 사용한 동일한 이미지를 사용하였다. 변수는 S(scale factor), W(kernel size), Sigma(space & range parameters) 등이 있다.

(현재 보고서에 많은 이미지 파일들로 인해 워드 작성에 지속적인 차질이 생겨 이 부분엔 이미지 파일을 최소화하기 위해 RGB 결과만을 삽입하였습니다. Luminance 결과는 제출 압축 파일에 첨부하였습니다.)

MATLAB

```
hdr_img = hdrread('Result_HDR.hdr');
%% RGB
img_o = (hdr_img(:, :, 1) + hdr_img(:, :, 2) + hdr_img(:, :, 3)) / 3;
img_r = hdr_img(:, :, 1) ./ img_o;
img_g = hdr_img(:, :, 2) ./ img_o;
img_b = hdr_img(:, :, 3) ./ img_o;
...
img_B = bilateralFilter(intensity, 5, [sigma_d, sigma_r]);
img_D = intensity - img_B;
offset = max(img_B(:));
...
img_BF_RGB(:, :, 1) = img_r;
img_BF_RGB(:, :, 2) = img_g;
img_BF_RGB(:, :, 3) = img_B;
img_BF_RGB = img_BF_RGB .^ gamma;

%% Luminance
% xyY
hdr_xyY = zeros(4000, 6000, 3);
hdr_xyz = rgb2xyz(hdr_img, 'ColorSpace', 'srgb');
hdr_xyY(:, :, 1) = hdr_xyz(:, : ,1) ./ (hdr_xyz(:, : ,1) +
hdr_xyz(:, : ,2) + hdr_xyz(:, : ,3));
hdr_xyY(:, :, 2) = hdr_xyz(:, : ,2) ./ (hdr_xyz(:, : ,1) +
hdr_xyz(:, : ,2) + hdr_xyz(:, : ,3));
hdr_xyY(:, :, 3) = hdr_xyz(:, : ,2);

img_L = log(hdr_xyY(:, :, 3));
img_B = bilateralFilter(intensity, 5, [sigma_d, sigma_r]);
D_img = img_L - img_B;
img_B = (img_B - max(img_B(:))) .* s;
img_tm = exp(D_img + img_B);

% RGB
img_BF_LUM(:, :, 1) = img_tm(:, :) .* hdr_xyY(:, :, 1) ./ hdr_xyY(:, :,
2);
img_BF_LUM(:, :, 2) = img_tm(:, :);
img_BF_LUM(:, :, 3) = img_tm(:, :) .* (1 - hdr_xyY(:, :, 1) -
hdr_xyY(:, :, 2)) ./ hdr_xyY(:, :, 2);

img_BF_LUM = xyz2rgb(img_BF_LUM);
```

Results

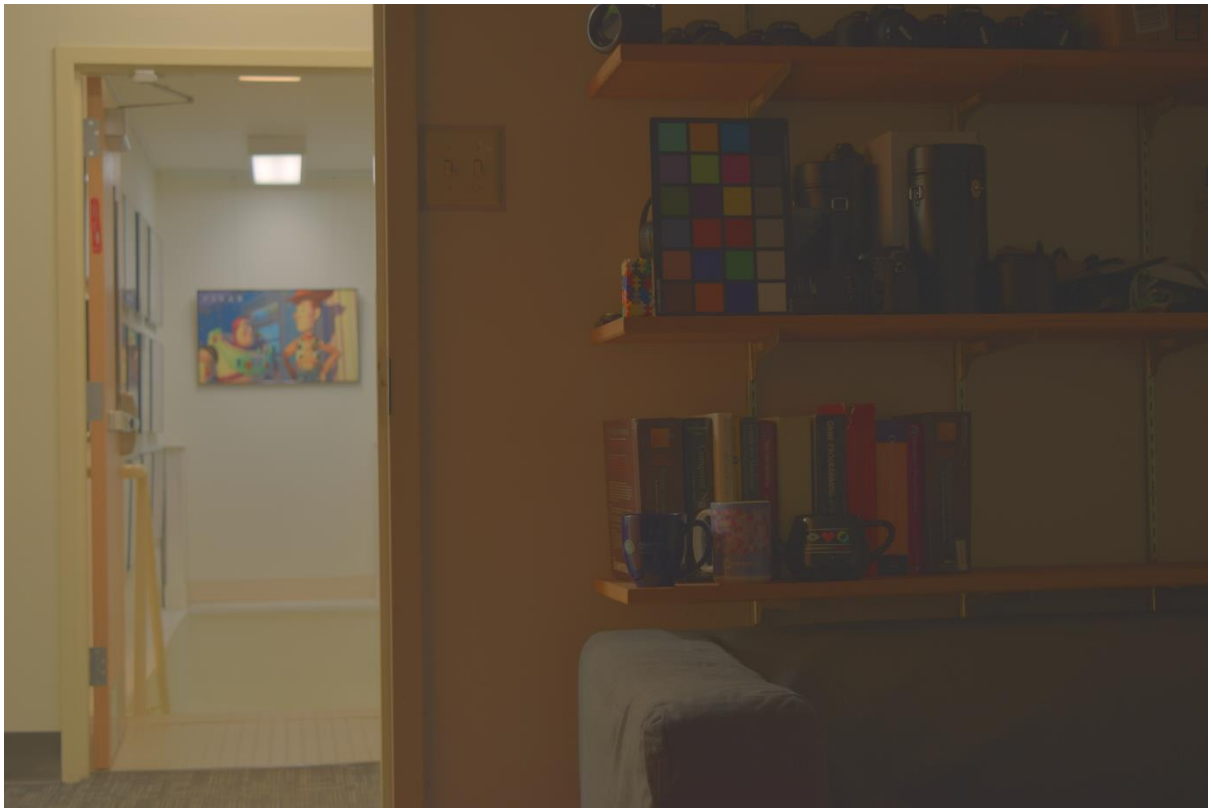

Figure 34. Result Image with Tonemapping using Bilateral Filter

변수에 따른 결과값의 차이는 S가 증가함에 따라 전체적인 밝기가 어두워지는 것을 확인할 수 있으며, W와 sigma 값이 증가함에 따라 이미지의 detail texture들이 sharpen 해지는 것을 볼 수 있다.

Figure 35. S increase from 0.3 to 1.0

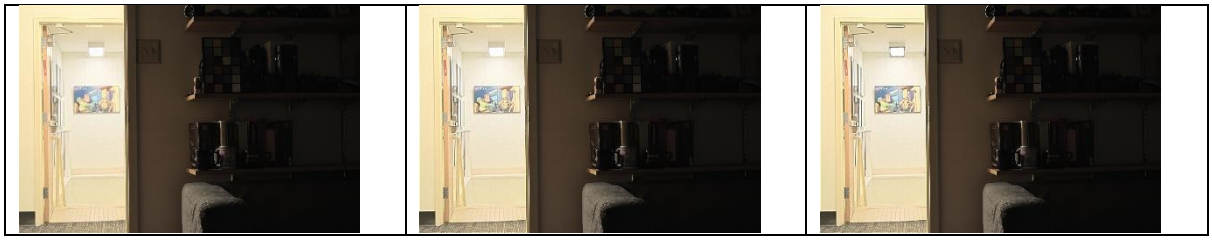

Figure 36. W and sigma increase

결과를 바탕으로 Bilater Filtering에선 S=0.3, W = 0.8, sigma = [0.4, 0.85]가 자연스러운 이미지를 출력하는 것으로 보인다.