**<Autumn 2018>**

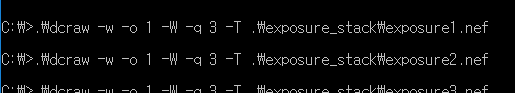
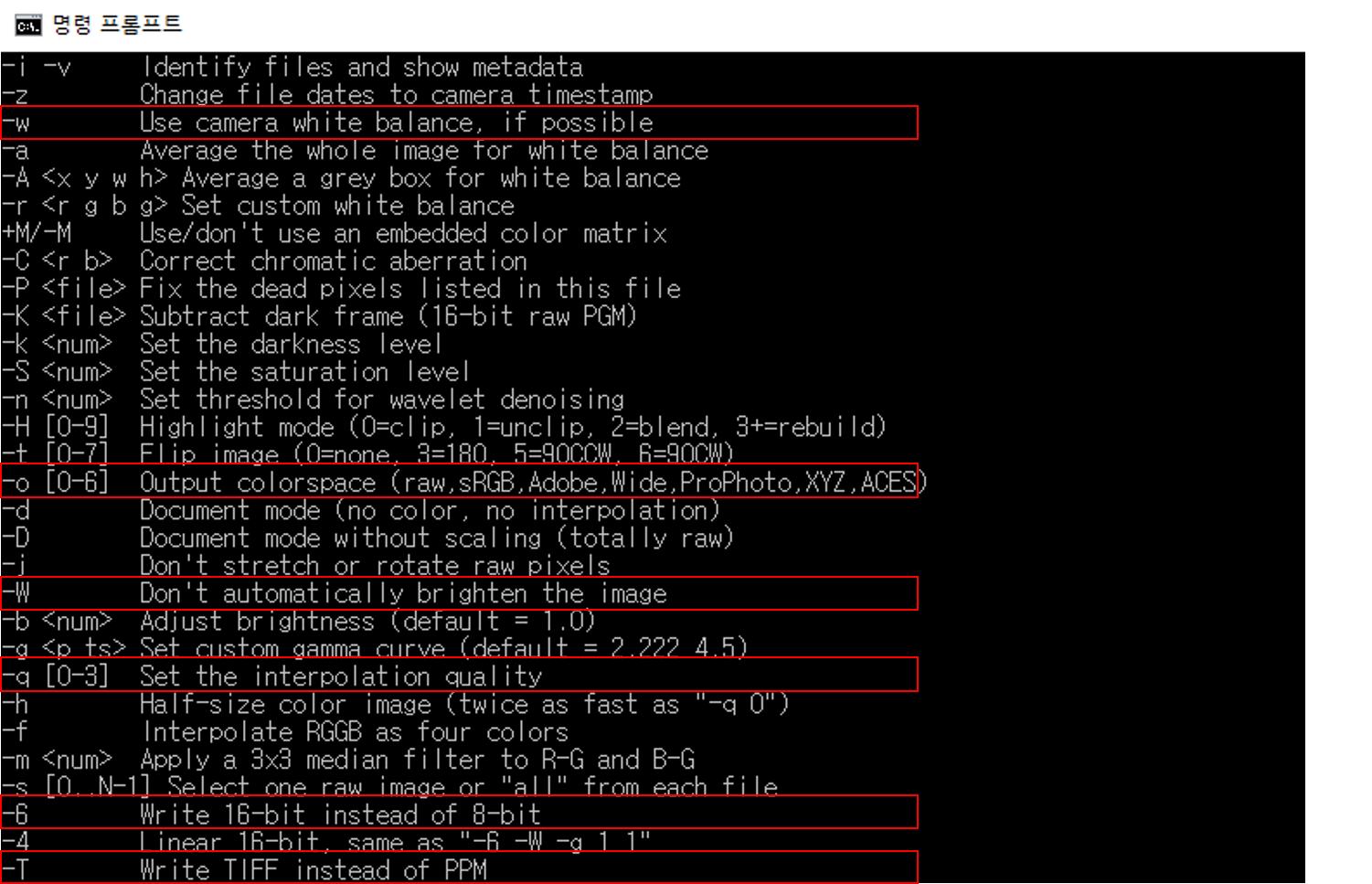
**계산영상시스템**

**Assignment #2**

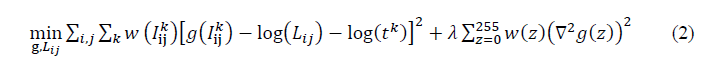
|  |
| --- |
| **심현정 교수님**  **글로벌융합공학부**  **2016314428 김진혁** |

**LINEARIZE RENDERED IMAGES**

**Code**



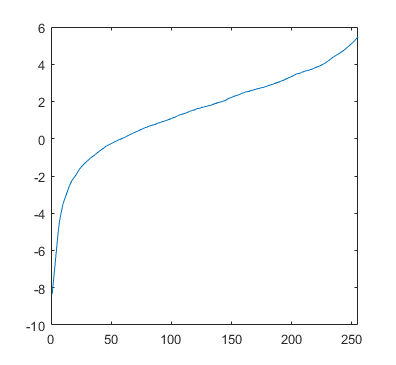
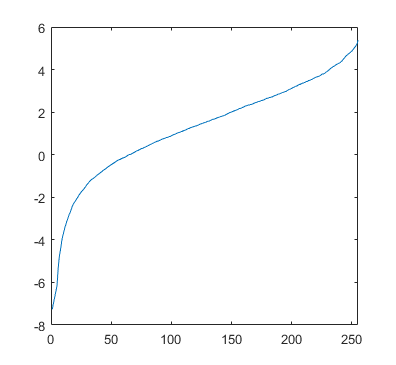
DCraw 프로그램을 다운받아 def 파일을 tiff 파일로 변환하였다. 사용한 버전은 9.27이며, 사용한 명령어는 위 사진에 표시한 6개이다.

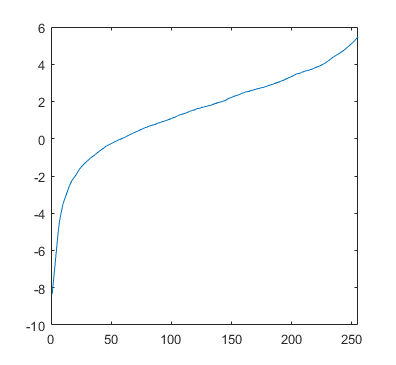


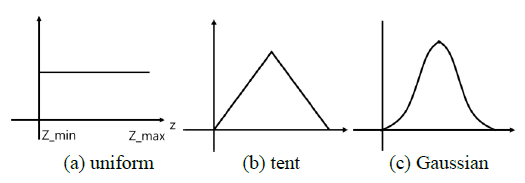
문제에서 주어진 공식을 사용하여 의 형태로 정리한 후, least-square를 취하며 최적화하면 값을 얻을 수 있다.

|  |
| --- |
| % Linearize Rendered images  image = {};  dir = 'exposure\_stack\\exposure';  for i = 1:16  image{end+1} = imread(strcat(dir, int2str(i), '.tiff'));  image{end} = im2uint8(imresize(image{end}, 0.125));  end  image\_jpg = {};  for i = 1:16  image\_jpg{end+1} = imread(strcat(dir, int2str(i), '.jpg'));  image\_jpg{end} = im2uint8(imresize(image\_jpg{end}, 0.125));  end    for c = 1:3  sample = zeros(32000, 1, 'uint8');  for i = 0:3999  x = randi(500); y = randi(750);  for j = 1:16  sample(i\*16 + j) = image\_jpg{j}(x, y, c);  end  end    image\_g = zeros(32256, 256);  b = zeros(30255, 1);  for i = 0:1999  for j = 1:15  image\_g((i\*16 + j), sample((i\*16 + j)) + 1) = 1;  b(i\*15 + j) = log(2) \* getw(sample(i\*16 + j + 1));  end  image\_g((i\*16 + 16), sample((i\*16 + 16)) + 1) = 1;  end    for i = i:256  image\_g(32000+i, i) = 1;  end  b(30255) = log(255);    diff = zeros(30255, 32256);  for i = 0:1999  for j = 1:15  diff(i\*15 + j, i\*16 + j) = -b(i\*15 + j);  diff(i\*15 + j, i\*16 + j+1) = b(i\*15 + j);  end  end  for i = 1:254  diff(30000+i, 32000+i) = 1;  diff(30000+i, 32000+i+1) = -2;  diff(30000+i, 32000+i+2) = 1;  end  diff(30255, 32256) = 1;    A = diff\*image\_g;  g(c, :) = A \ b;  end |

**Result**

****

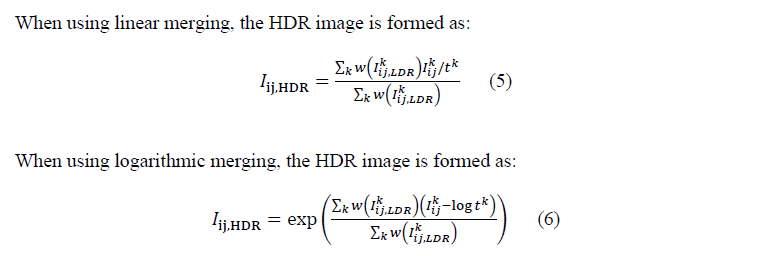


****

**MERGE EXPOSURE STACK INTO HDR IMAGE**

**Code**

|  |
| --- |
| for c = 1:3  merge = 0;  for i = 1:500  for j = 1:750  weight = 0;  v = 0;  for f = 1:16  v = v+getw(image{f}(i, j, c))\*double(image{f}(i, j, c))\*2^(-f);  %if ( image{f}(i, j, c) > 0 )  % v = v+getw(image{f}(i, j, c))\*(log(double(image{f}(i, j, c)))-log(2)\*f);  %end  %v = v+getw(image\_jpg{f}(i, j, c))\*exp(g(c, image\_jpg{f}(i, j, c)+1));  %v = v+getw(image\_jpg{f}(i, j, c))\* (g(c, image\_jpg{f}(i, j, c)+1)-log(2)\*f);  weight = weight+getw(image\_jpg{f}(i, j, c));  end  if weight ~= 0  v = v/weight;  end    %v = exp(v);    image\_\_(i, j, c) = v;  if ( merge < v )  merge = v;  end  end  end  image\_\_(:, :, c) = image\_\_(:, :, c) ./ m;  end |



문제에서 주어진 공식을 사용하여 linear merging과 logarithmic merging을 수행하였다. 각 주석의 line들은 raw/render 및 linear/logarithmic에 맞게 사용되었다.

**Result**

|  |  |  |
| --- | --- | --- |
| **UNIFORM** | LINEAR | LOGARITHMIC |
| RAW |  |  |
| RENDERED |  |  |

|  |  |  |
| --- | --- | --- |
| **GAUSSIAN** | LINEAR | LOGARITHMIC |
| RAW |  |  |
| RENDERED |  |  |

**EVALUATION**

**Code**

|  |
| --- |
| image\_eval = rgb2xyz(image\_\_, 'ColorSpace', 'srgb');  y\_ = zeros(6, 1);    for f = 1:6  for i = fp(f, 1) - 3 : fp(f, 1) + 3  for j = fp(f, 2) - 3 : fp(f, 2) + 3  y\_(f) = y\_(f) + image\_eval(j, i, 2);  end  end  y\_(f) = log(y\_(f) / 49);  end    err = 0;    for f = 2:5  err = err + ((y\_(1) + (y\_(6) - y\_(1)) \* (f-1) / 5) - y\_(f)) ^ 2;  end |

위 문제에서 수행한 8장의 HDR image에 대한 least-square error를 계산하였다.

**Result**

|  |  |  |
| --- | --- | --- |
| **UNIFORM** | LINEAR | LOGARITHMIC |
| RAW | 0.03 | 0.0188 |
| RENDERED | 0.0498 | 0.0067 |

|  |  |  |
| --- | --- | --- |
| **GAUSSIAN** | LINEAR | LOGARITHMIC |
| RAW | 0.1347 | 0.0326 |
| RENDERED | 0.0549 | 0.0511 |

**PHOTOGRAPHIC TONEMAPPING**

**Code**

|  |
| --- |
| image\_hdr = hdrread('tonemapping.hdr');    K = 0.5; B = 0.95; image\_tone = zeros(500, 750, 3);    for c = 1:3  var = exp(1/(500\*750) \* sum(log(image\_hdr(:, :, c) + 1e-15)));  var2 = image\_hdr(:, :, c) \* K / var;  var\_white = B \* max(var2(:));  tone = var2 .\* (var2./(var\_white^2) + 1) ./ (var2+1);  image\_tone(:, :, c) = tone(:, :);  end    imwrite(image\_tone, strcat('RGB\_',int2str(K),'\_',int2str(B),'.png'));    image\_hdr\_ = rgb2xyz(image\_hdr, 'ColorSpace', 'srgb');  hdr = zeros(500, 750, 3);  hdr(:, :, 1) = image\_hdr\_(:, : ,1) ./ (image\_hdr\_(:, : ,1) + image\_hdr\_(:, : ,2) + image\_hdr\_(:, : ,3));  hdr(:, :, 2) = image\_hdr\_(:, : ,2) ./ (image\_hdr\_(:, : ,1) + image\_hdr\_(:, : ,2) + image\_hdr\_(:, : ,3));  hdr(:, :, 3) = image\_hdr\_(:, : ,2);    var = exp(1/(500\*750) \* sum(log(image\_hdr(:, :, c) + 1e-15)));  var2 = image\_hdr(:, :, c) \* K / var;  var\_white = B \* max(var2(:));  tone = var2 .\* (var2./(var\_white^2) + 1) ./ (var2+1);    image\_tone(:, :, 1) = tone(:, :).\*hdr(:, :, 1)./hdr(:, :, 2);  image\_tone(:, :, 2) = tone(:, :);  image\_tone(:, :, 3) = tone(:, :).\*(1-hdr(:, :, 1)-hdr(:, :, 2))./hdr(:, :, 2);    image\_tone = xyz2rgb(image\_tone);  imwrite(image\_tone, strcat('Luminance\_',int2str(K),'\_',int2str(B),'.png')); |

문제에서 주어진 힌트대로 K=0.15, B=0.95에서 시작하여 최적의 값을 찾아나갔다. 우선, K 값을 먼저 찾고, 그 다음 B 값을 찾는 식으로 진행하였다. 진행한 결과, K=0.35, B=0.1에서 최적의 결과를 얻을 수 있었다.

**Result**

|  |  |  |
| --- | --- | --- |
|  | RGB | Luminance |
| K=0.35  B=0.95 |  |  |
| K=0.35  B=0.1 |  |  |

**TONEMAPPING USING BILATERAL FILTERING**

**Code**

|  |
| --- |
| image\_hdr = hdrread('tonemapping\_.hdr');    S=0.3; W=1.; sig=[5.,1.0]; image\_tone = zeros(500, 750, 3);    for c = 1:3  image\_L = log(image\_hdr(:, :, c));  image\_B = bfilter2(image\_L, W, sig);  image\_D = image\_L-image\_B;  image\_B = (image\_B-max(image\_B(:))) .\* S;  image\_DB = exp(image\_D+image\_B);  image\_tone(:, :, c) = image\_DB(:, :);  end    imwrite(image\_tone, strcat('RGB\_',int2str(S),'\_',int2str(W),'\_',int2str(sig),'.png'));    image\_hdr\_ = rgb2xyz(image\_hdr, 'ColorSpace', 'srgb');  hdr = zeros(500, 750, 3);  hdr(:, :, 1) = image\_hdr\_(:, : ,1) ./ (image\_hdr\_(:, : ,1) + image\_hdr\_(:, : ,2) + image\_hdr\_(:, : ,3));  hdr(:, :, 2) = image\_hdr\_(:, : ,2) ./ (image\_hdr\_(:, : ,1) + image\_hdr\_(:, : ,2) + image\_hdr\_(:, : ,3));  hdr(:, :, 3) = image\_hdr\_(:, : ,2);    image\_L = log(hdr(:, :, 3));  image\_B = bfilter2(image\_L, W, sig);  image\_D = image\_L-image\_B;  image\_B = (image\_B - max(image\_B(:))) .\* S;  image\_DB = exp(image\_D + image\_B);  image\_tone(:, :, 1) = image\_DB(:, :) .\* hdr(:, :, 1) ./ hdr(:, :, 2);  image\_tone(:, :, 2) = image\_DB(:, :);  image\_tone(:, :, 3) = image\_DB(:, :) .\* (1 - hdr(:, :, 1) - hdr(:, :, 2)) ./ hdr(:, :, 2);    image\_tone = xyz2rgb(image\_tone);  imwrite(image\_tone, strcat('Luminance\_',int2str(S),'\_',int2str(W),'\_',int2str(sig),'.png')); |

위 문제와 마찬가지 방법으로 W, S, σ 값을 변화시키며 최적의 값을 찾아나갔다. 최적의 값으로는 W=5, S=0.2, σ[5, 1.0]을 얻어내었다.

**Result**

|  |  |  |
| --- | --- | --- |
| **W=5** | RGB | Luminance |
| S=0.5 |  |  |
| S=0.2 |  |  |

|  |  |  |
| --- | --- | --- |
|  | W=1 | W=10 |
| **S=0.2** |  |  |

|  |  |  |
| --- | --- | --- |
|  | σ[1, 1.0] | σ[5, 1.0] |
| **S=0.2 W=5** |  |  |