**Q1. HDR IMAGING**

**Description**

Dave Coffin의 dcraw 프로그램의 소스코드를 받아 컴파일하여 사용하였다.



사용한 파라미터는 다음과 같다.

-o 1 : 출력물의 색공간을 sRGB로 한다.

-w : white balancing을 수행한다.

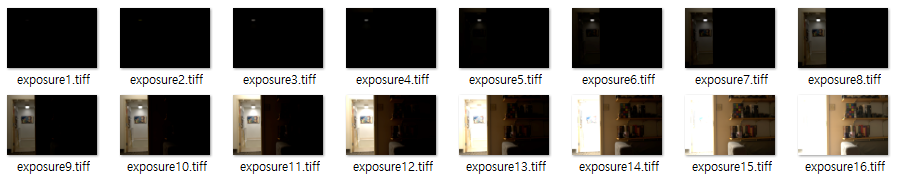
-q 3 : 최대한 퀄리티가 높은 방법으로 interpolation을 수행한다.

-6 : 16bit color로 출력한다.

-W : 이미지를 자동으로 밝게 하지 않는다.

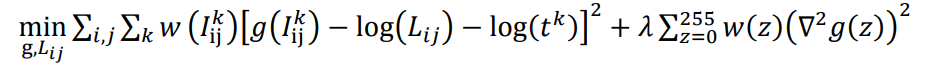
-T : 출력물을 tiff로 한다.

**Result**



**Q2. LINEARIZE RENDERED IMAGES**

**Description**



주어진 공식을 보면 이 공식은 pixel의 위치와는 independent한 식임을 알 수 있다. 따라서 픽셀의 일부를 샘플링하여 실행 속도를 높였다.

가 한 픽셀에서는 상수라는 점에서, 이라는 L과 상관없는 식을 만들 수 있다.

다음으로 에서만 1이고 나머지는 0인 256-D 벡터 를 정의하면, = 로 적을 수 있고 이를 펼치면 하나의 픽셀에 대하여

라는 식을 만들 수 있다. 이 벡터의 1차 미분은 모두 1이다. weight까지 고려하면 하나의 픽셀에 대하여

을 근사적으로 얻을 수 있다.

이제 형태의 식을 얻었으므로 샘플링한 모든 픽셀과 라플라시안 매트릭스까지 더하여 least-squares optimization을 풀면 된다.

이때 모든 등식을 미분 형태로만 구성하였으므로 constant shift가 생길 수 있다. 따라서 g(255) = log(255)로 고정하였다.

**Code**

|  |
| --- |
| % Load image as reduced size for reduce memory usage.  image\_origin = {};  for i = 1:16  image\_origin{end+1} = imread(strcat('res\_stack\\exposure',int2str(i),'.tiff'));  image\_origin{end} = im2uint8(imresize(image\_origin{end}, 0.125));  end    for chan = 1:3  % sample 2000 pointe  samples = zeros(32000, 1, 'uint8');  for i = 0:1999  x = randi(750);  y = randi(500);  for f = 1:16  samples(i \* 16 + f) = image\_jpg{f}(y, x, chan);  end  end  lambda = 1.0;  % construct G matrix and the differance value b  getg = zeros(32256, 256);  b = zeros(30255, 1);  for i = 0:1999  for j = 1:15  getg((i \* 16 + j), samples((i \* 16 + j)) + 1) = 1;  b(i \* 15 + j) = 0.693147180 \* getw(samples(i \* 16 + j + 1));  end  getg((i \* 16 + 16), samples((i \* 16 + 16)) + 1) = 1;  end    % append original g for create laplacian  for i = 1:256  getg(32000 + i, i) = 1;  end    % set constant (for preserve 255)  b(30255) = log(255);      % set first derivative matrix  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  % set second derivative (laplacian) matrix  for i = 1:254  diff(30000 + i, 32000 + i) = 1 \* lambda;  diff(30000 + i, 32000 + i + 1) = -2 \* 1 \* lambda;  diff(30000 + i, 32000 + i + 2) = 1 \* lambda;  end  % set constant (for preserve 255)  diff(30255, 32256) = 1;      A = diff \* getg;  g(chan, :) = A \ b;  end |

**Result**

|  |  |  |
| --- | --- | --- |
| w = uniform | w = tent | w = gaussian |
|  |  |  |

**Q2. MERGE EXPOSURE STACK INTO HDR IMAGE & Q3. EVALUATION**

**Description**

linear merging공식으로

logarithmic merging 공식으로

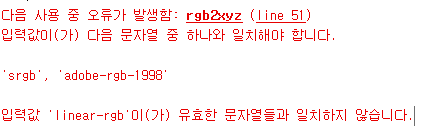
을 사용하였다.

또한 Evaluation을 위해 원본 이미지에 있는



6개 점의 luminacity를 XYZ 색공간에서 비교하여 얼마나 linear한지 확인하는 방법을 사용하였다.

다만 색공간 변환 과정에서



오류가 발생하여 srgb로 비교하였다.

**Code**

|  |
| --- |
| for chan = 1:3  m = 0;  for i = 1:500  for j = 1:750  wsum = 0;  v = 0;  for f = 1:16  % Descripted below  % /--  % --  % --  % --/  wsum = wsum + getw(image\_jpg{f}(i, j, chan));  end  if wsum ~= 0  v = v / wsum;  end  % For Logarithm merging  % /--  v = exp(v);  % --/  image\_res(i, j, chan) = v;  if (m < v)  m = v;  end  end  end  image\_res(:, :, chan) = image\_res(:, :, chan) ./ m;  end  % Image Evaluation  image\_xyz = rgb2xyz(image\_res, 'ColorSpace', 'srgb');  lum = 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  lum(f) = lum(f) + image\_xyz(j, i, 2);  end  end  lum(f) = log(lum(f) / 49);  end  lumerr = 0;  for f = 2:5  lumerr = lumerr + ((lum(1) + (lum(6) - lum(1)) \* (f-1) / 5) - lum(f)) ^ 2;  end |

**For Raw, Linear merging**

|  |
| --- |
| v = v + getw(image\_tiff{f}(i, j, chan)) \* ...  double(image\_tiff{f}(i, j, chan)) \* 2 ^ (-f); |

**For Raw Lorarithm merging**

|  |
| --- |
| if(image\_tiff{f}(i, j, chan) > 0)  v = v + getw(image\_tiff{f}(i, j, chan)) \* ...  (log(double(image\_tiff{f}(i, j, chan))) - 0.69314718 \* f);  end |

**For Rendered, Linear merging**

|  |
| --- |
| v = v + getw(image\_jpg{f}(i, j, chan)) \* ...  exp(g(chan, image\_jpg{f}(i, j, chan) + 1)) \* 2 ^ (-f); |

**For Rendered, Lorarithm merging**

|  |
| --- |
| v = v + getw(image\_jpg{f}(i, j, chan)) \* ...  (g(chan, image\_jpg{f}(i, j, chan) + 1) - 0.69314718 \* f); |

**Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | x5 | x125 | x625 |  |
| raw  uniform  linear |  |  |  | 0.03 |
| raw  uniform  logarithm |  |  |  | 0.0167 |
| raw  gaussian  linear |  |  |  | 0.1133 |
| raw  gaussian  logarithm |  |  |  | 0.0325 |
| rendered  uniform  linear |  |  |  | 0.0467 |
| rendered  uniform  logarithm |  |  |  | 0.0053 |
| rendered  gaussian  linear |  |  |  | 0.0510 |
| rendered  gaussian  logarithm |  |  |  | 0.0488 |