トーンマッピング その３

ヒストグラム平坦化
及び
画像間でのヒストグラムの交換

ヒストグラム平坦化

～逆光の補正に簡易的に利用できる～

ヒストグラム平坦化

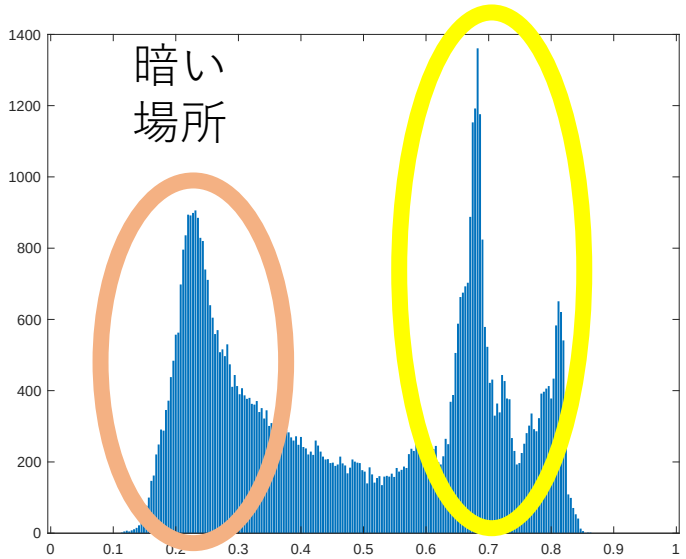
- 逆光時など，被写体によって，
  明るさにばらつきがある場合の補正処理
  - 暗い場所と明るい場所があると，
    ヒストグラムの山が分かれる．

明るい場所

暗い
場所

Sailbort.png

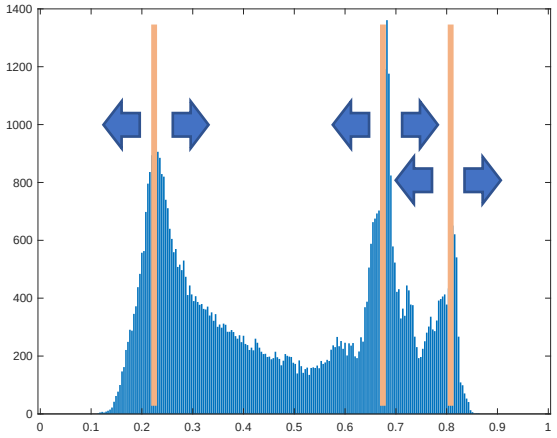
明るい場所



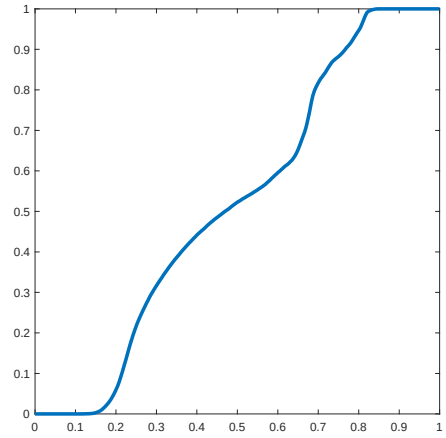
Sailbort の輝度値の分布

目的

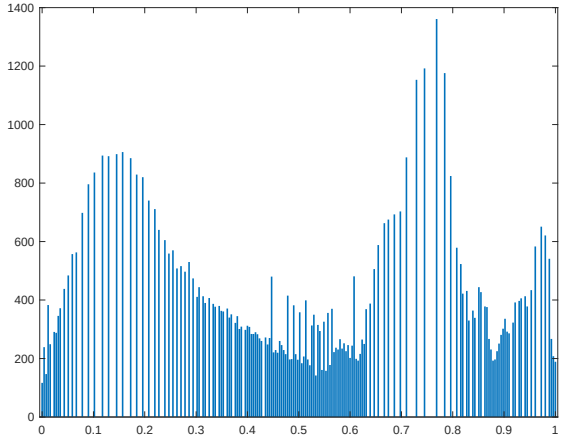
- 狭い輝度範囲に頻度が集中すると（山ができると）
  - 同一範囲内に属する他の画素との輝度差が小さくなり，区別がつきにくくなる．
  - **山を広げたい．**
  - ヒストグラムの山を広げるようなマッピング関数を自動的に生成したい．



Sailbort の輝度値の分布



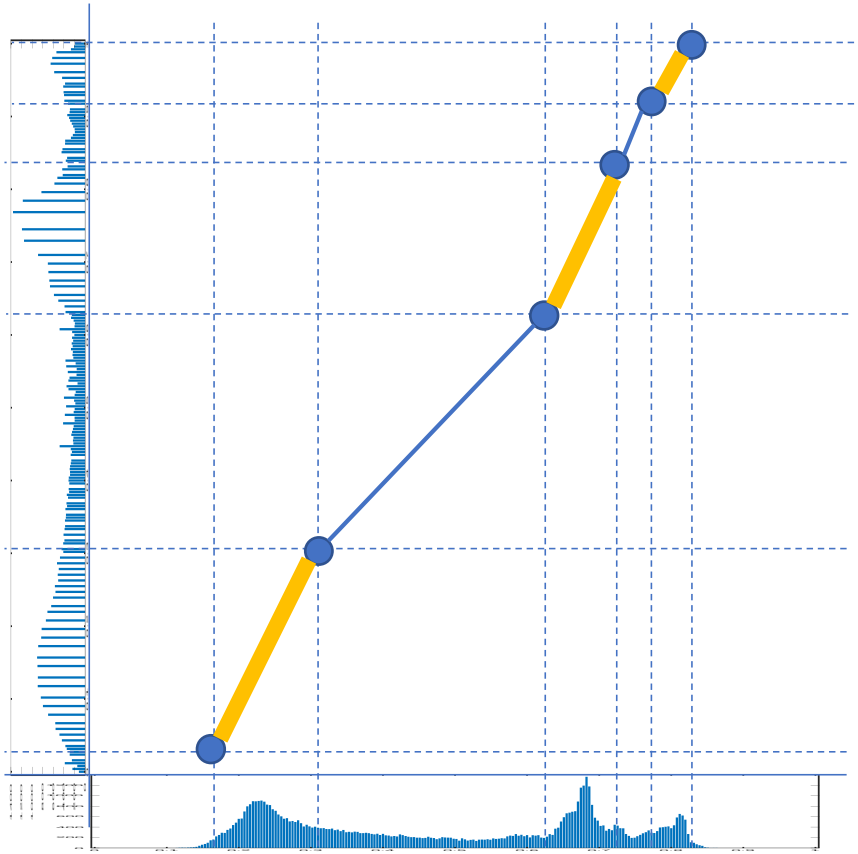
トーンマッピング関数



トーンマッピング後

どのような関数形状が必要か？

- 狭い輝度分布を広げたい
  - 広げたい場所は，
    関数の傾きを大きくする．
- ヒストグラムの山と
  関数の傾きの関係
  - ヒストグラムの山が大きい
    区間で
  - 関数の傾きを大きくしたい
  - 利用できないか？

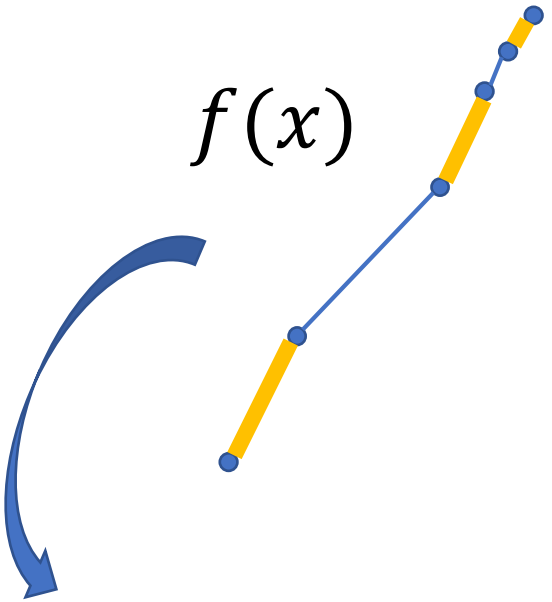


傾きが大きいとは？

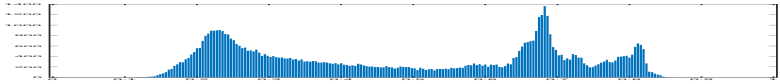
- 関数を微分したときの
  値が大きい（勾配が大きい）
- ２つの関数間の関係
  - トーンマッピング関数を $f(x)$,
  - ヒストグラム関数を $h(x)$ として
  - $\frac{d}{dx} f(x) = s\, h(x)$

  $s > 0$ は何らかの係数

  一般的には，$s = \frac{1}{\int h(x)\, dx}$



$h(x)$



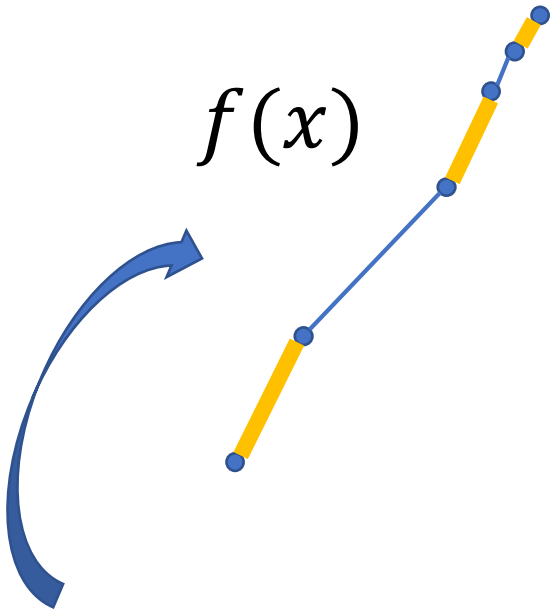
ヒストグラム関数の積分

- トーンマッピング関数 $f(x)$ を求めたい.
  - ヒストグラム関数 $h(x)$ を基にして
  - $f(x) = s \int h(x)dx$

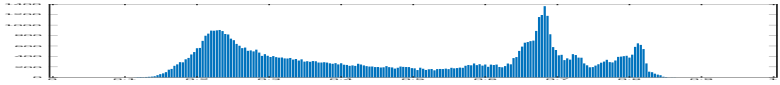
- 離散値であるので
  - 累積和を計算する
  - $f(x) = \sum_{i=0}^{x} h(i)$

$f(x)$

$\int h(x)dx$



$h(x)$



補足：離散信号の積分と微分

- 離散信号において
  - 微分：　となり合う値の引き算　　$h(i) = f(i) - f(i-1)$
  - 積分：　これまでの合計値に，
    現在の値を足す（累積和）　$f(i) = h(i) + \{h(i-1) + \ldots + h(0)\}$

- 簡単なプログラムで確認
  - ```
    N = 10;
    F = randi( N, [1,5] ) % 1～N までの 5 個の数値を生成
    dF = diff( [0, F] )  % 微分：隣同士を引く，最初に 0 を加える
    F_ = cumsum( dF )    % 積分：累積和を計算
    ```
    ```

プログラム（１／３）

```
I = im2double(imread('../images/Sailboat.png'));
Iycc = rgb2ycbcr( I );
G = Iycc(:,:,1);

% ヒストグラムの計算
N = 256;
X = (0:N-1)/(N-1);
H = hist( G(:), X );
H = H / sum(H);
% ヒストグラムの合計値で割る
% 同じ輝度値をもつ画素が，全画像中にどの程度の
% 割合で含まれるかを表すようになる．

figure(1), bar( X, H );
```

プログラム（２／３）

```
% ヒストグラム平坦化用のトーンマッピング関数の生成
% 累積和の計算
F = cumsum( H );

% 画像の輝度値を階調数へと変換
Tone = round( (N-1)*G ) + 1;

% トーンマッピング結果を出力画像に出力
G_out = G;
for i = 1:numel( G )
  tone = Tone( i );
  G_out( i ) = F( tone );
end
```

プログラム（３／３）

- % 入出力関係（トーンマッピング関数形状）の表示

```
figure(2), plot( X, F );
xlim([0,1]);  ylim([0,1]);  axis equal; % 軸の調整

% トーンマッピング後の輝度画像のヒストグラム
H_out = hist( G_out(:), X );
figure(3), bar( X, H_out );

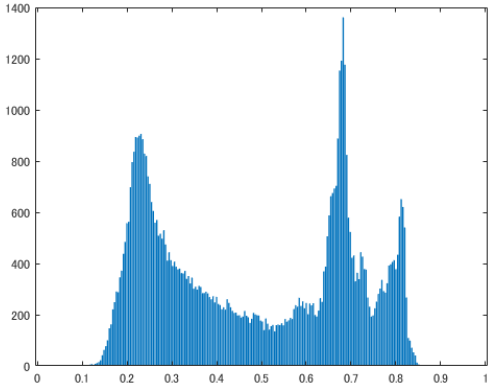
% 結果を出力
Jycc = Iycc;
Jycc(:,:,1) = G_out;
J = ycbcr2rgb( Jycc );

figure(4), imshow( [I,J] );
```

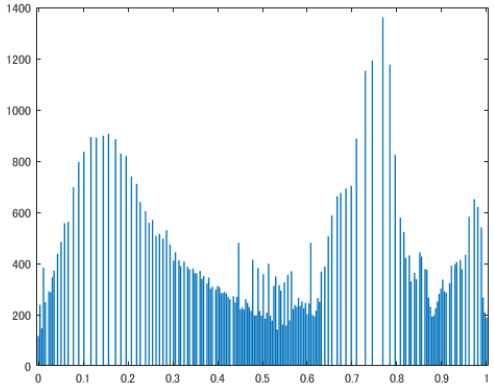
実行結果

入力画像

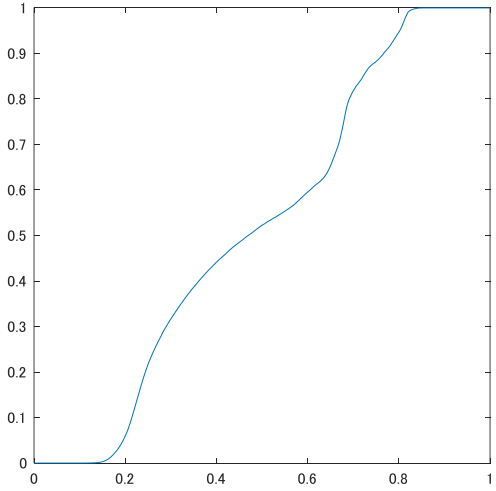
出力画像



入力画像の輝度の
ヒストグラム



出力画像の輝度の
ヒストグラム



トーンマップ関数

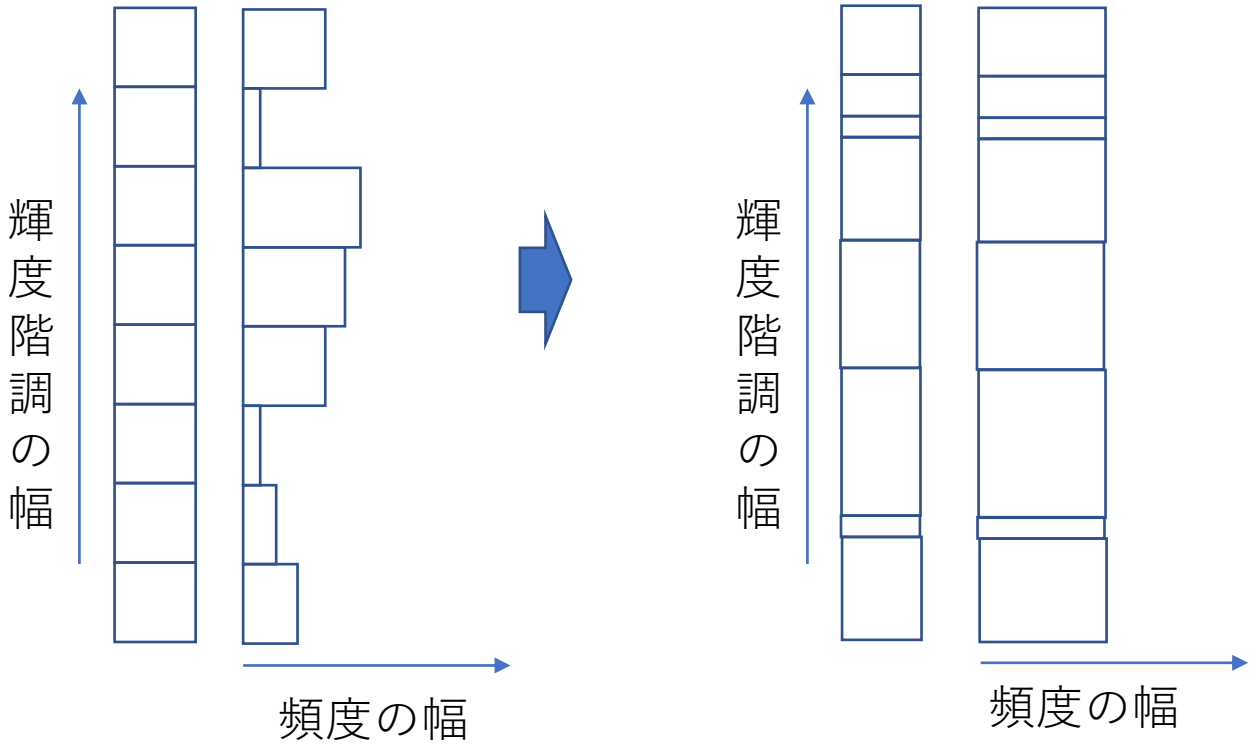
補足：ここまでの計算の意味

- 入力と出力で，輝度階調の幅と頻度の幅を交換している．
  - 入力側のヒストグラムが各階調で持っていた頻度の粗密さが，
    出力側では階調方向に広がる．
    そして，その幅の異なるビンで頻度を計算すると，
    同一の頻度をもつようになる

入力画像では，

輝度階調が
等間隔で
区切られ

それぞれの
区間で
異なる幅の
頻度をもつ



他の画像との
ヒストグラムの交換

～カラーグレーディング処理への応用～

他の画像のヒストグラムの反映

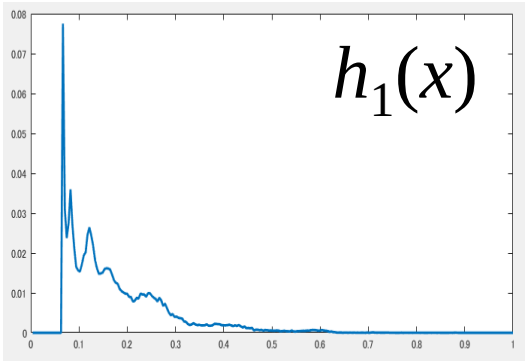
- 他の画像の輝度値や色の分布を反映する
  - 色を補正する処理，Color Grading にも利用できる．

処理対象
（destination）

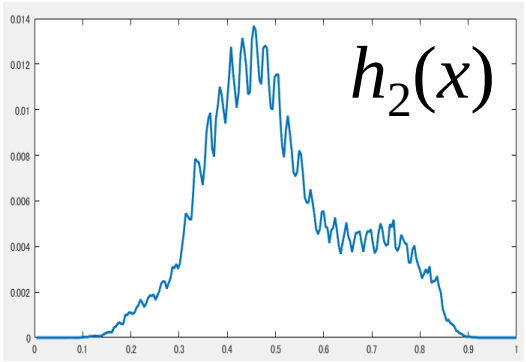
反映

似せる

参照画像
(source)



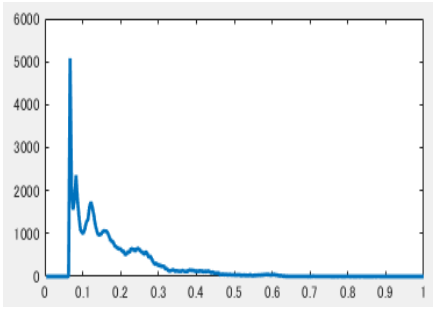
ヒストグラム（偏りあり）



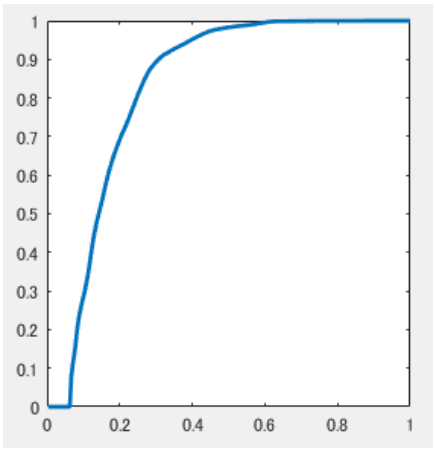
ヒストグラム

積分関数の対応付け（１／３）

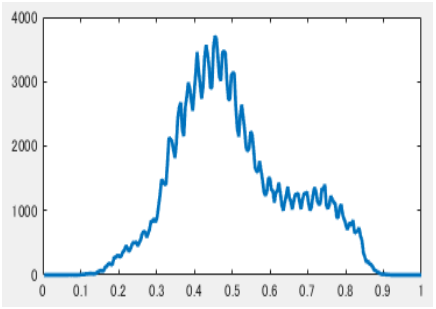
- 積分前の関数の特徴，及び，積分後の関数の特徴
  - 積分前：　正の値を持ち，負の値を持たない．
  - 積分後：　単調に増加する．



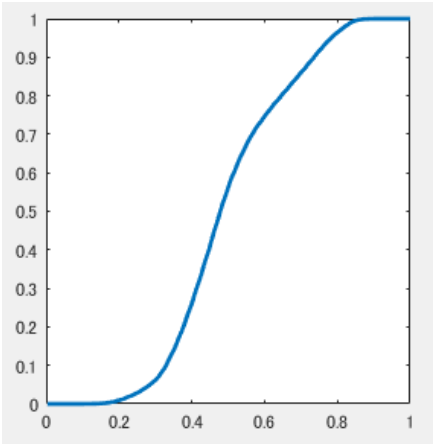
$h_1(x)$



$H_1(x)$



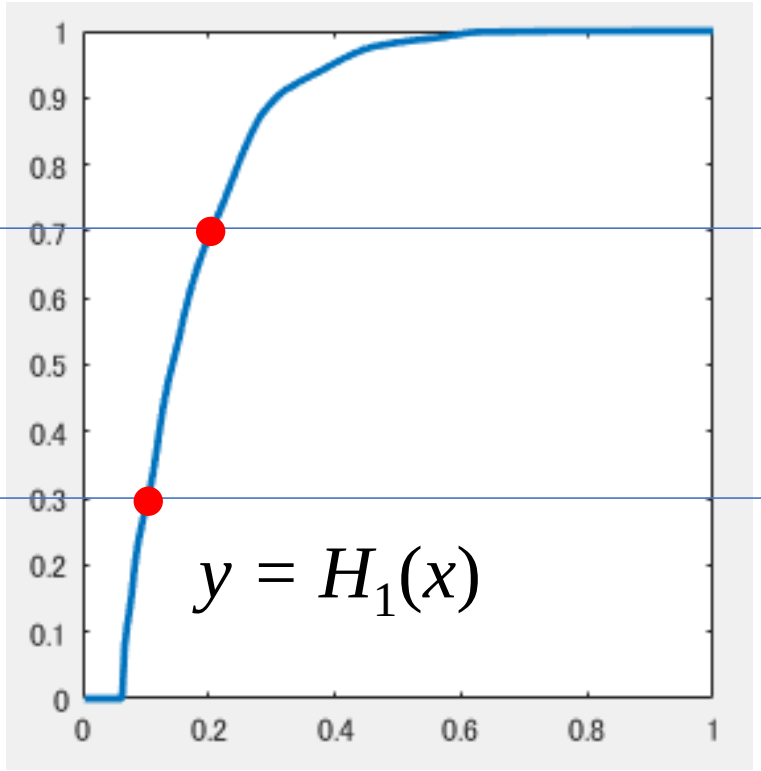
$h_2(x)$

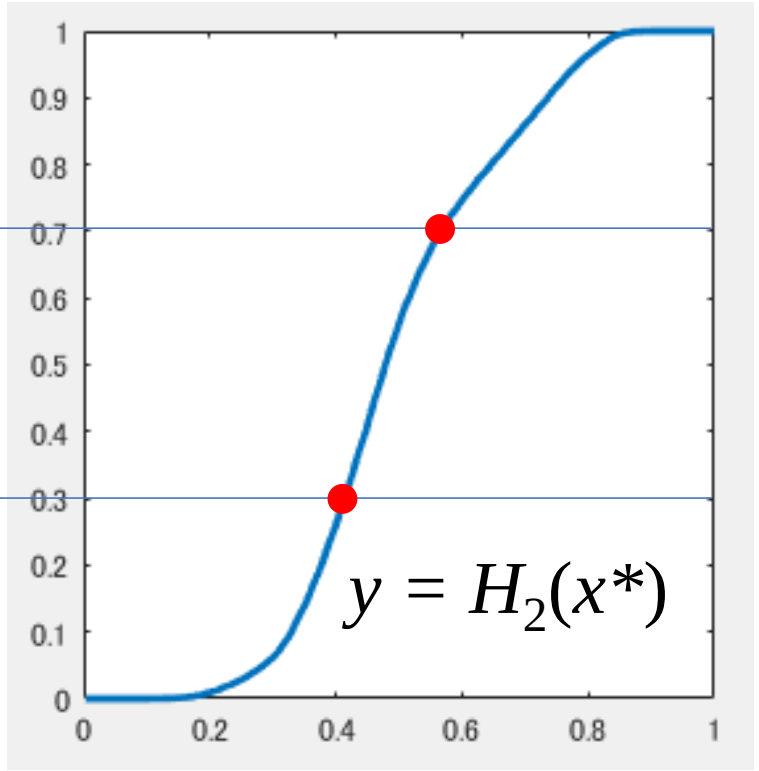


$H_2(x)$

積分関数の対応付け（２／３）

- 単調増加を利用
  - 縦軸方向に，２つの積分関数の値は１対１で対応付が可能.
  - 例
    - 縦軸 0.3 と 0.7 と２つの関数の交点は，２つの関数において，ともに１つ

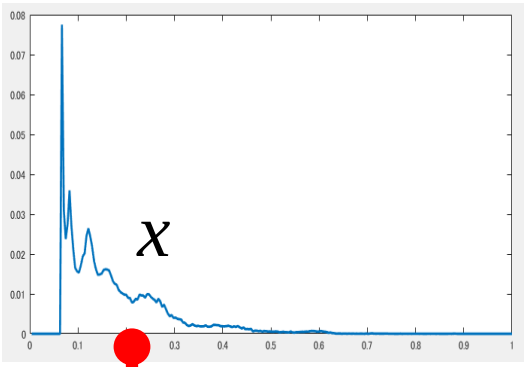




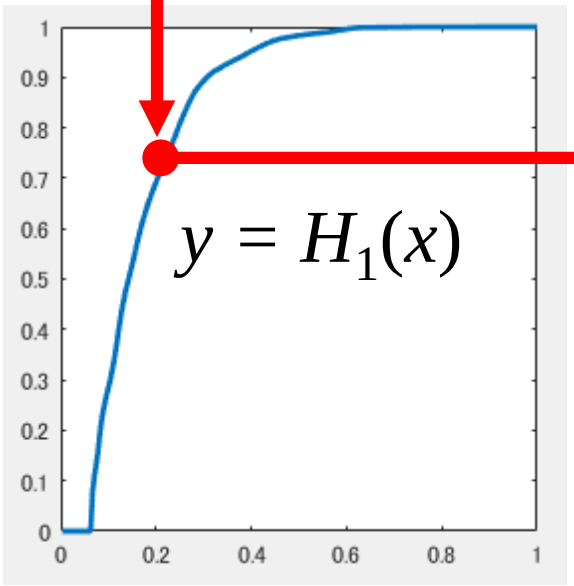
積分関数の対応付け（１／３）

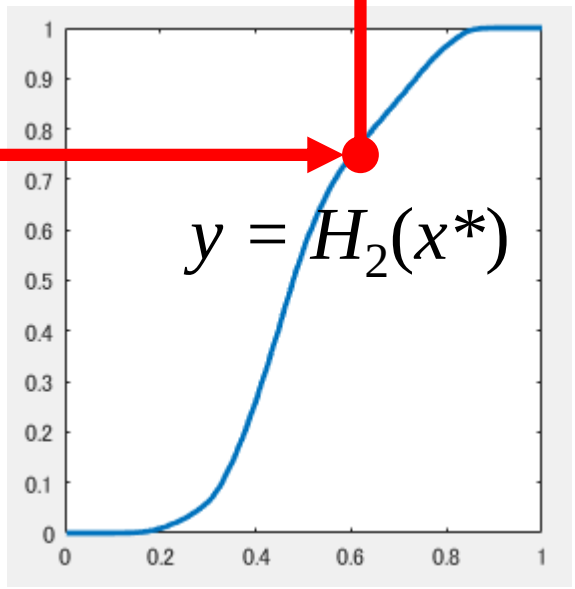
- 関数間の対応関係　$x^* = \mathrm{ToneMap}(x) = H_2^{-1}(H_1(x))$

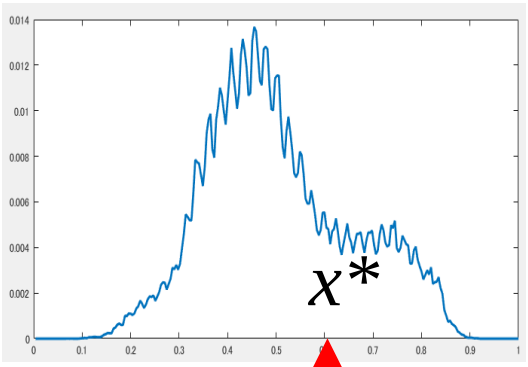
Step 1.
0.2 が入力される.



Step 2.
0.2 に対応する出力輝度値 0.75 を取得







Step 4.
探索により得られた 0.6 が出力される.

Step 3.
別の画像からの出力輝度値 0.75 に対応する入力輝度値を探索

プログラム（１／３）

```
I_dst = im2double( imread('../images/couple.png') ); % 処理画像
I_src = im2double( imread('../images/family.png') ); % 参照画像

Iycc_dst = rgb2ycbcr( I_dst );
Iycc_src = rgb2ycbcr( I_src );

G_dst = Iycc_dst(:,:,1);
G_src = Iycc_src(:,:,1);

N = 256;
X = (1:(N-1))/(N-1);

H_dst = hist( G_dst(:), X ) / numel( G_dst ); % ヒストグラム生成
H_src = hist( G_src(:), X ) / numel( G_src ); % 頻度から割合に変換

figure(5);
subplot(2,2,1), imshow( G_dst );
subplot(2,2,2), imshow( G_src );
subplot(2,2,3), plot( X, H_dst, 'LineWidth', 2 );
subplot(2,2,4), plot( X, H_src, 'LineWidth', 2 );
```

プログラム（２／３）

```
F_dst = cumsum( H_dst );
F_src = cumsum( H_src );

figure(6);
subplot(1,2,1), plot( X, F_dst, 'LineWidth', 2 );
subplot(1,2,2), plot( X, F_src, 'LineWidth', 2 );

% 入出力関係を探索
Y = ones( size(X) );  % 同サイズの配列を用意
for i = 1:N-1
  f_dst = F_dst( i );
  i_src = max( 1, find( F_src >= f_dst, 1 ) - 1 );
  if ~isempty( i_src )   % 見つからなかったら，1 とする
    Y(i) = X( i_src );
  end
end
figure(7), plot( X, Y, 'LineWidth', 2 );
```

プログラム（３／３）

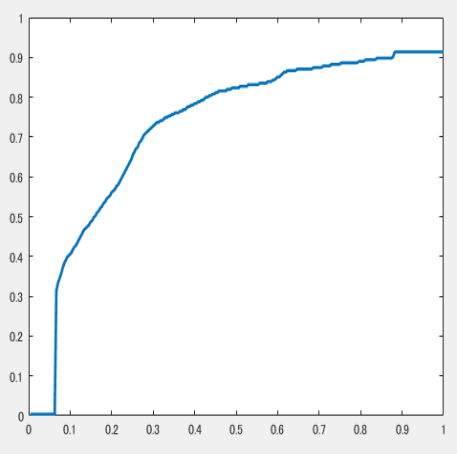
```
% トーンマッピング結果を出力画像に出力
Tone = round( (N-1)*G_dst ) + 1;

G_out = G_dst;
for i = 1:numel( G_dst )
  tone = Tone( i );
  G_out( i ) = Y( tone );
end

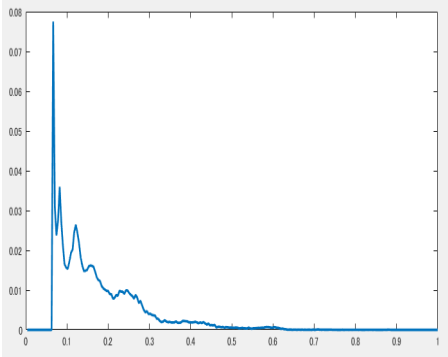
% トーンマッピング後の輝度画像のヒストグラム
H_out = hist( G_out(:), X );
figure(8), bar( X, H_out );

% 結果を出力
Jycc = Iycc_dst;
Jycc(:,:,1) = G_out;
I_out = ycbcr2rgb( Jycc );
figure(9), imshow( [I_dst,I_out] );
```

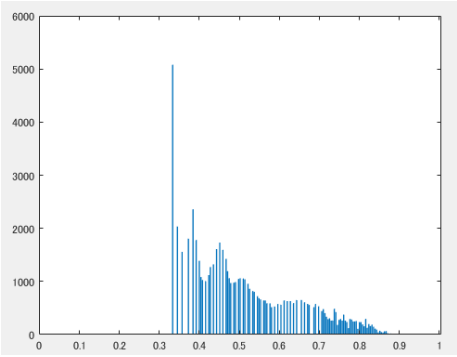
実験結果



得られた
トーンマッピングの
入出力関係



入力画像のヒストグラム

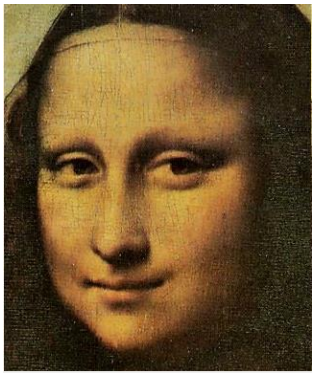


出力画像のヒストグラム

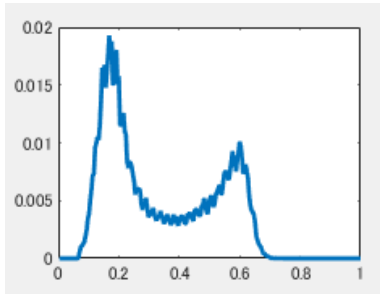
入力画像　　出力画像

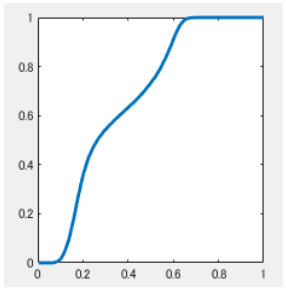
他の画像でも

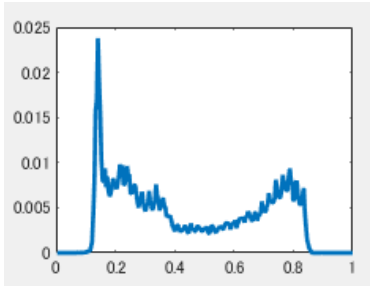
- face1.png と face2.png に置き換えてみる.

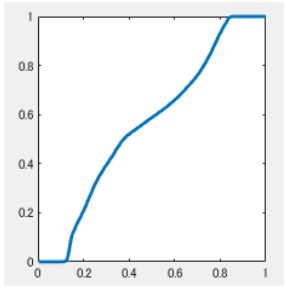


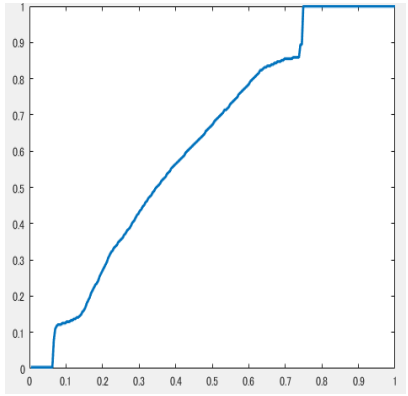


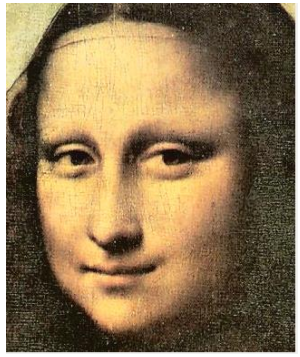
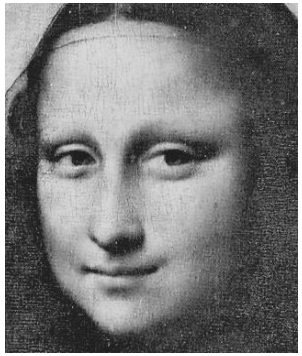
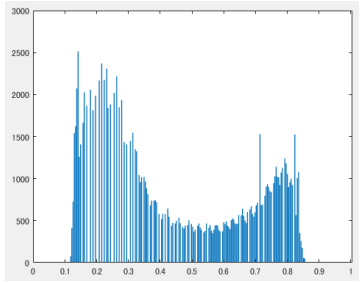






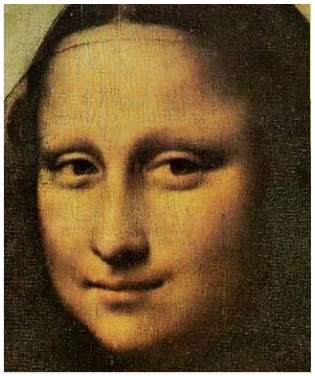


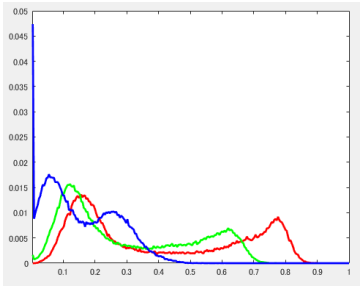


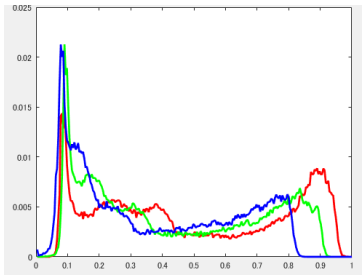




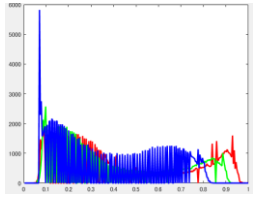
各色ごとに行った場合

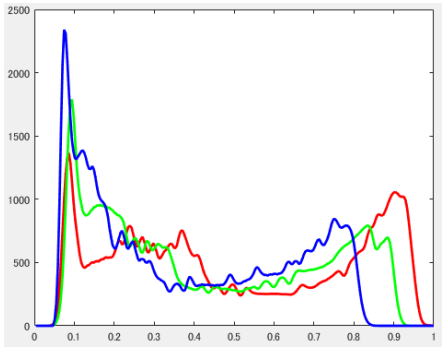
- YCbCr 変換をせず，R，G，B ごとにトーンマッピング











エイリアシングが生じ，
短冊（タンザク）状に見えるため，
可視化のため，ヒストグラムを平滑化