Lab 6 - Display Characterization: Team 14 (Jenee Janglois & Justin Peterson)

Table of Contents

Import Ramps Data 1
Derrive forward LUTs 1
Blue Channel Forward LUTs 2
Green Channel Forward LUTs 2
Plot all LUTs 3
Test quality of forward model 3
Generare reverse display matrix 4
Save and plot final display model 4
Display Reverse Matrix content 4
Print relevant functions 5
Test reverse display model 6

Import Ramps Data

```
%Invoke professor provided script
run load_ramps_data_1516;

% Fetch largest XYZs for R G and B channels
MAX_XYZS = [ramp_R_XYZs(11,:);ramp_G_XYZs(11,:);ramp_B_XYZs(11,:)];

M_fwd = derive_fwd_matrix(MAX_XYZS, black_XYZ, white_XYZ);
```

Derrive forward LUTs

```
% Subtract black from red ramp XYZ values
red_sub_black = ramp_R_XYZs - repmat(black_XYZ,11,1);

%Normalize values by display white
red_sub_black = red_sub_black ./ repmat(white_XYZ,11,1);

%Clip values outside the range of zero and one
red_sub_black(red_sub_black<0) = 0;
red_sub_black(red_sub_black>1) = 1;

%Multiply by inverse of first 3x3 of forward model
fwd_inv_three = inv(M_fwd(:,1:3));
est_RGB_radiometric_sclr = (red_sub_black * fwd_inv_three)';

ramp_R_RS = est_RGB_radiometric_sclr(1,:);
```

```
% define the 0-255 display values (digital counts) that correspond to
 the ramp values
ramp_DCs = round(linspace(0,255,11));
% interpolate the radiometric scalars across the full digital count
 range to form the forward LUTs
RLUT_fwd = interp1(ramp_DCs,ramp_R_RS(1,:),[0:1:255],'spline');
```

Blue Channel Forward LUTs

```
% Subtract black from red ramp XYZ values
blue_sub_black = ramp_B_XYZs - repmat(black_XYZ,11,1);

%Normalize values by display white
blue_sub_black = blue_sub_black ./ repmat(white_XYZ,11,1);

%Clip values outside the range of zero and one
blue_sub_black(blue_sub_black<0) = 0;
blue_sub_black(blue_sub_black>1) = 1;

%Multiply by inverse of first 3x3 of forward model
fwd_inv_three = inv(M_fwd(:,1:3));
est_RGB_radiometric_sclr_B = (blue_sub_black * fwd_inv_three)';

ramp_B_BS = est_RGB_radiometric_sclr_B(3,:);

% define the 0-255 display values (digital counts) that correspond to
 the ramp values
ramp_DCs_B = round(linspace(0,255,11));
% interpolate the radiometric scalars across the full digital count
 range to form the forward LUTs
BLUT_fwd = interp1(ramp_DCs,ramp_B_BS(1,:),[0:1:255],'spline');
```

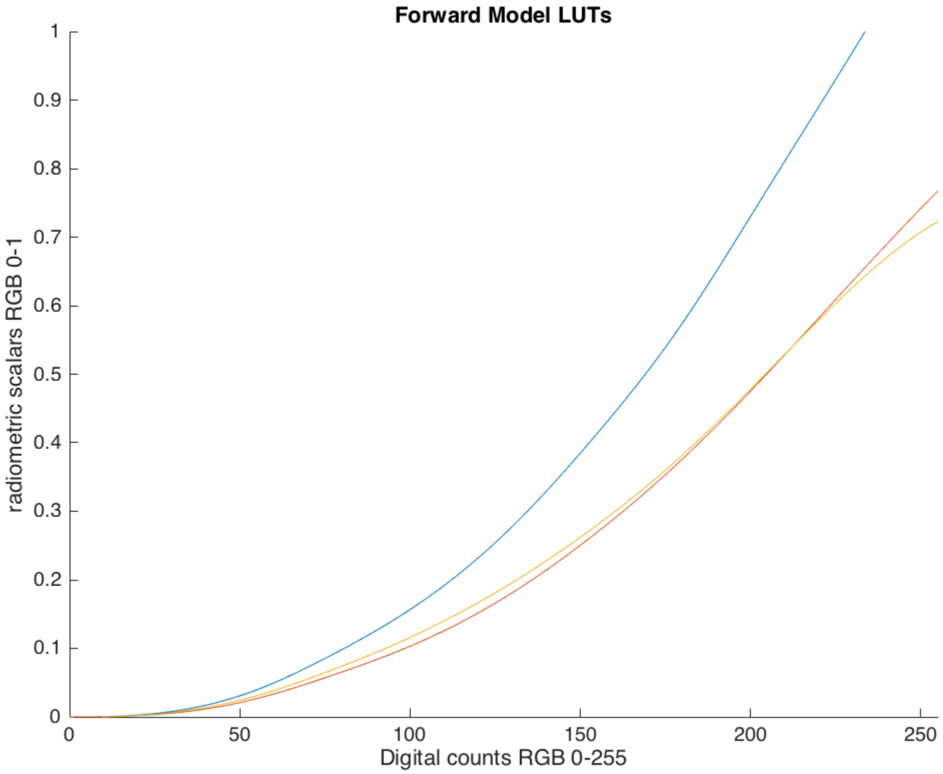
Green Channel Forward LUTs

```
% Subtract black from red ramp XYZ values
green_sub_black = ramp_G_XYZs - repmat(black_XYZ,11,1);

%Normalize values by display white
green_sub_black = green_sub_black ./ repmat(white_XYZ,11,1);

%Clip values outside the range of zero and one
green_sub_black(green_sub_black<0) = 0;
green_sub_black(green_sub_black>1) = 1;

%Multiply by inverse of first 3x3 of forward model
fwd_inv_three = inv(M_fwd(:,1:3));
est_RGB_radiometric_sclr_G = (green_sub_black * fwd_inv_three)';

ramp_G_GS = est_RGB_radiometric_sclr_G(2,:);

% define the 0-255 display values (digital counts) that correspond to
 the ramp values
```

```
ramp_DCs_G = round(linspace(0,255,11));
% interpolate the radiometric scalars across the full digital count
 range to form the forward LUTs
GLUT_fwd = interp1(ramp_DCs_G,ramp_G_GS(1,:),[0:1:255],'spline');
```

Plot all LUTs

```
hold on;
plot(0:255,RLUT_fwd, 0:255, GLUT_fwd, 0:255, BLUT_fwd);
axis([0,255,0,1]);
title('Forward Model LUTs');
xlabel('Digital counts RGB 0-255');
ylabel('radiometric scalars RGB 0-1');
```



Test quality of forward model

```
run test_forward_model_1516;
```

```
max_deltaE =

   24.6065


min_deltaE =
    0.1620


mean_deltaE =

   14.4464
```

Generare reverse display matrix

```
%inverse of the first three columns of the forward column matrix
M_rev = inv(M_fwd(:,1:3));
fprintf('M_rev = \n')
disp(M_rev);

% build the reverse LUT for the red channel
RLUT_rev = round(interp1(RLUT_fwd, 0:255,
 linspace(0,max(RLUT_fwd),1024), 'spline', 0));

% repeat for green and blue
GLUT_rev = round(interp1(GLUT_fwd, 0:255,
 linspace(0,max(GLUT_fwd),1024), 'spline', 0));
BLUT_rev = round(interp1(BLUT_fwd, 0:255,
 linspace(0,max(BLUT_fwd),1024), 'spline', 0));
```

```
M_rev =
    3.3373   -1.5656   -0.5220
   -0.9943    1.8796    0.0626
    0.0450   -0.1093    0.8891
```

Save and plot final display model

```
M_disp = M_rev;
XYZk = black_XYZ;
XYZk_disp = XYZk;
RLUT_disp = RLUT_rev;
GLUT_disp = GLUT_rev;
BLUT_disp = BLUT_rev;

% save the reverse model matrix, reverse LUTs as and black level as
 'display_model.mat'
 save('display_model.mat','M_disp','RLUT_disp','GLUT_disp','BLUT_disp','XYZk_disp'
```

Display Reverse Matrix content

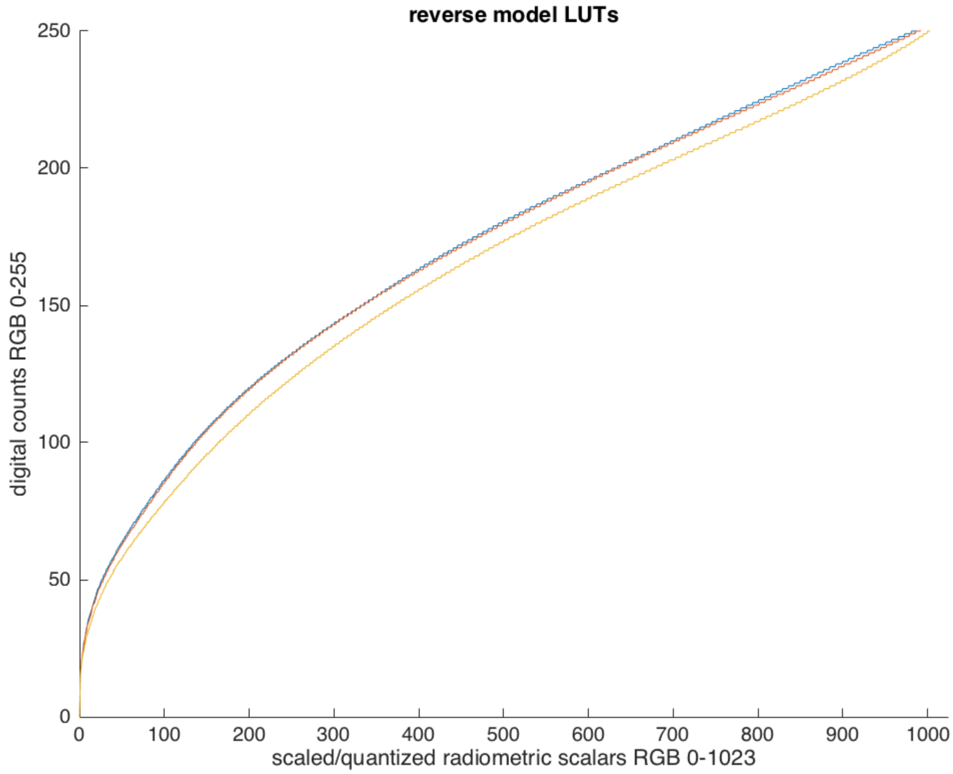
```
fprintf('M_disp = M_rev = \n');
disp(M_rev);

fprintf('XYZk_disp = XYZk = \n');
```

```
disp(XYZk);

M_disp = M_rev =
    3.3373   -1.5656   -0.5220
   -0.9943    1.8796    0.0626
    0.0450   -0.1093    0.8891

XYZk_disp = XYZk =
    0.1419    0.1358    0.2691
```

Print relevant functions

```
dbtype('derive_fwd_matrix.m');

clf;
hold on;
plot(0:1023,RLUT_rev,0:1023,GLUT_rev,0:1023,BLUT_rev);
axis([0,1024,0,250]);
title('reverse model LUTs');
xlabel('scaled/quantized radiometric scalars RGB 0-1023');
ylabel('digital counts RGB 0-255');
hold off;
```

```
function [ M_fwd ] = derive_fwd_matrix(MAX_XYZS, BLACK_XYZS, WHITE_XYZS)

  red = (MAX_XYZS(1,:) - BLACK_XYZS)';
  green = (MAX_XYZS(2,:) - BLACK_XYZS)';
  blue = (MAX_XYZS(3,:) - BLACK_XYZS)';

  M_fwd = [red green blue BLACK_XYZS'] ./ WHITE_XYZS(:,2);
end
```



Test reverse display model

```
cie = loadCIEData();
D50_XYZ = ref2XYZ(cie.illE, cie.cmf2deg, cie.illD50);
D65_XYZ = ref2XYZ(cie.illE, cie.cmf2deg, cie.illD50);

Munki_Patch_XYZs = importdata('munki_CC_XYZs_Labs.txt');
Munki_Patch_XYZs = Munki_Patch_XYZs(:,2:4);

adapt_XYZs = catBradford(Munki_Patch_XYZs',D50_XYZ, D65_XYZ);

% Subtract XYZ black from each adapted value
adapt_XYZs = adapt_XYZs' - repmat(black_XYZ,24,1);

%Multiply by matrix to obtain radiometric scalars
scalars = adapt_XYZs * M_disp;

% Normalize scalars by 100
scalars = scalars./100;

% Clip any out of range values
scalars(scalars<0) = 0;
scalars(scalars>1) = 1;

%Multiply scalars by 1023 and round to nearest integer
```

```
scalars = round(scalars * 1023) + 1;

%Index into appropriate lookup tables
R_LUT_RESULT = RLUT_rev(scalars);
G_LUT_RESULT = GLUT_rev(scalars);
B_LUT_RESULT = BLUT_rev(scalars);

% Convert to 8 bit unsigned integers
R_LUT_RESULT = uint8(R_LUT_RESULT);
G_LUT_RESULT = uint8(G_LUT_RESULT);
B_LUT_RESULT = uint8(B_LUT_RESULT);
```

Published with MATLAB® R2015b