Display Characterization Lab

By Max Shooster and Malcolm Zale

Team 4

In this project we characterize our display so that it can faithfully reproduce specified color values. This involves developing forward (RGB->XYZ) and reverse (XYZ-> RGB) display models. To develop the forward model we use the ColorMunki and the Argyll software to measure the XYZ values of a set of RGB color patches presented on our display, and then process this data to derive look-up tables (LUTs) that compensate for the display?s non-linear response, and a matrix that estimates XYZs from linearized RGBs. To develop the reverse model we invert the forward model matrix and LUTs. To test our reverse model wes render and display an RGB image of the ColorChecker chart from its XYZ values and then measure the color differences between the real chart values and the displayed values.

Contents

Steps 1-3

```
close all;
load_ramps_data;
```

Step 4

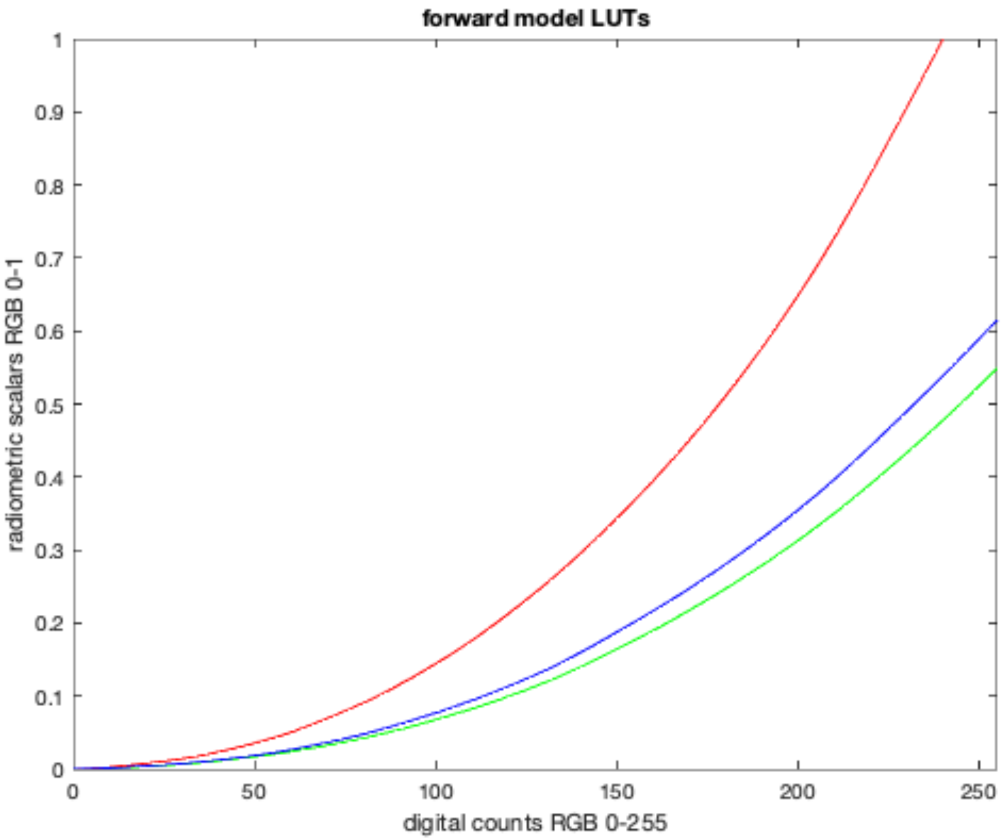
```
first_row = [ramp_R_XYZs(1, end) - XYZk(1,1), ramp_G_XYZs(1, end) - ...
    XYZk(1,1), ramp_B_XYZs(1, end) - XYZk(1,1), XYZk(1,1)];
second_row = [ramp_R_XYZs(2, end) - XYZk(2,1), ramp_G_XYZs(2, end) - ...
    XYZk(2,1), ramp_B_XYZs(2, end) - XYZk(2,1), XYZk(2,1)];
third_row = [ramp_R_XYZs(3, end) - XYZk(3,1), ramp_G_XYZs(3, end) - ...
    XYZk(3,1), ramp_B_XYZs(3, end) - XYZk(3,1), XYZk(3,1)];

M_fwd = [first_row;second_row;third_row]/XYZw(2,1)
```

```
M_fwd =

    0.3975    0.3682    0.1818    0.0012
    0.2080    0.7146    0.0700    0.0012
    0.0130    0.1097    0.9453    0.0021
```

Step 5

```
cancr = ramp_R_XYZs - XYZk;

norm_R_XYZs = cancr ./ XYZw(2,1);
%clip out of range values
norm_R_XYZs(norm_R_XYZs < 0) = 0;
norm_R_XYZs(norm_R_XYZs > 1) = 1;

ramp_R_RSs = norm_R_XYZs' * M_fwd(1:3,1:3)^(-1);
ramp_R_RSs = ramp_R_RSs';
% define the 0-255 display values (DCs) thatcorrespomd to ramp values
ramp_DCs = round(linspace(0,255,11));
% interpolate the radiometric scalars across the full digital count range
% to form the forward luts
x = ramp_DCs;
v = ramp_R_RSs(1,:);
xq = (0:1:255);
RLUT_fwd = interp1(x,v,xq,'pchip');

%now the green
cancg = ramp_G_XYZs - XYZk;
norm_G_XYZs = cancg ./ XYZw(2,1);
norm_G_XYZs(norm_G_XYZs < 0) = 0;
norm_G_XYZs(norm_G_XYZs > 1) = 1;
ramp_G_RSs = norm_G_XYZs' * M_fwd(1:3,1:3)^(-1);
ramp_G_RSs = ramp_G_RSs';
q = ramp_G_RSs(1,:);
GLUT_fwd = interp1(x,q,xq,'pchip');

%now the blue
cancb = ramp_B_XYZs - XYZk;
norm_B_XYZs = cancb ./ XYZw(2,1);
norm_B_XYZs(norm_B_XYZs < 0) = 0;
norm_B_XYZs(norm_B_XYZs > 1) = 1;
ramp_B_RSs = norm_B_XYZs' * M_fwd(1:3,1:3)^(-1);
ramp_B_RSs = ramp_B_RSs';
z = ramp_B_RSs(1,:);
BLUT_fwd = interp1(x,z,xq,'pchip');


% plots the LUTS
plot(xq, RLUT_fwd, 'r')
hold on
```

```
plot(xq, GLUT_fwd, 'g')
plot(xq, BLUT_fwd, 'b')
title 'forward model LUTs'
xlabel 'digital counts RGB 0-255'
ylabel 'radiometric scalars RGB 0-1'
xlim([0 255])
ylim([0 1])
```



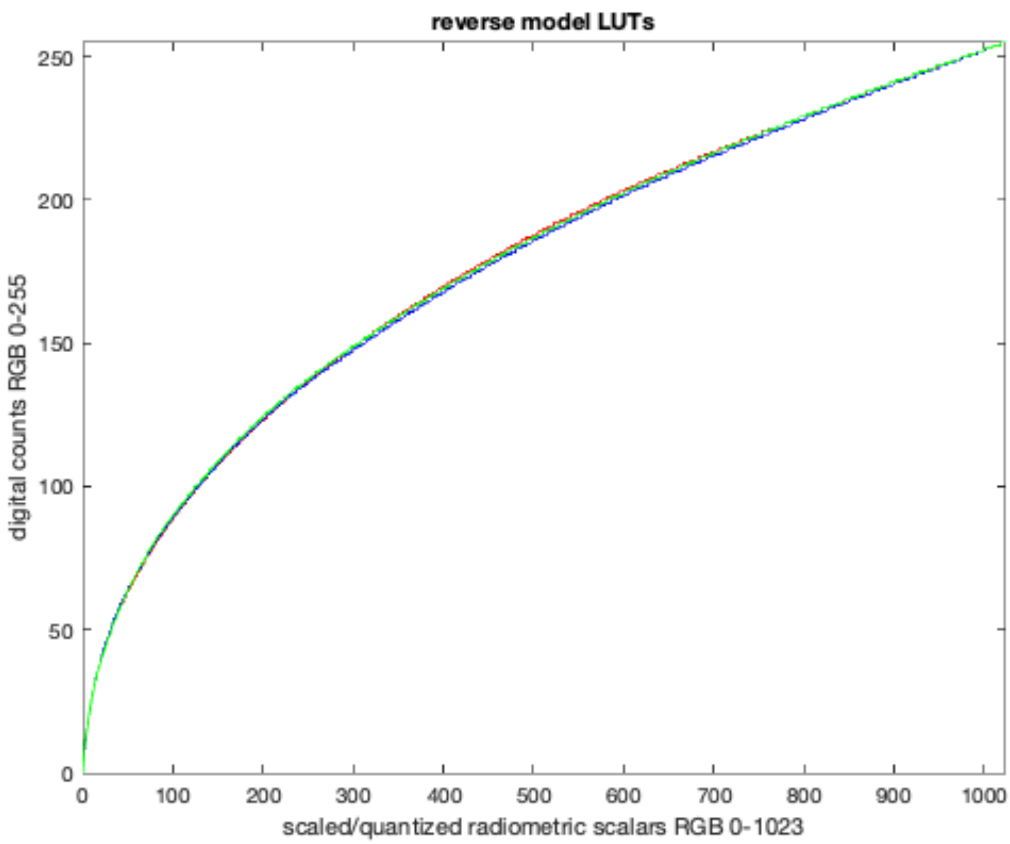
Step 6

```
M_rev = inv(M_fwd(1:3, 1:3));
```

Step 7

```
%build the reverse LUT for the red channel
RLUT_rev = uint8(round(interp1(RLUT_fwd, xq, linspace(0,max(RLUT_fwd),1024)...
, 'pchip', 0)));
%now the blue
BLUT_rev = uint8(round(interp1(BLUT_fwd, xq, linspace(0,max(BLUT_fwd),1024)...
, 'pchip', 0)));
%now the green
GLUT_rev = uint8(round(interp1(GLUT_fwd, xq, linspace(0,max(GLUT_fwd),1024)...
, 'pchip', 0)));
```

```
%defines x axis values
dave = 0:1023;
%plot the reverse LUTs
figure
plot(dave, RLUT_rev,'r')
hold on
plot(dave, BLUT_rev, 'b')
plot(dave, GLUT_rev, 'g')
title 'reverse model LUTs'
xlabel 'scaled/quantized radiometric scalars RGB 0-1023'
ylabel 'digital counts RGB 0-255'
ylim([0 255])
xlim([0 1024])
```



Step 8

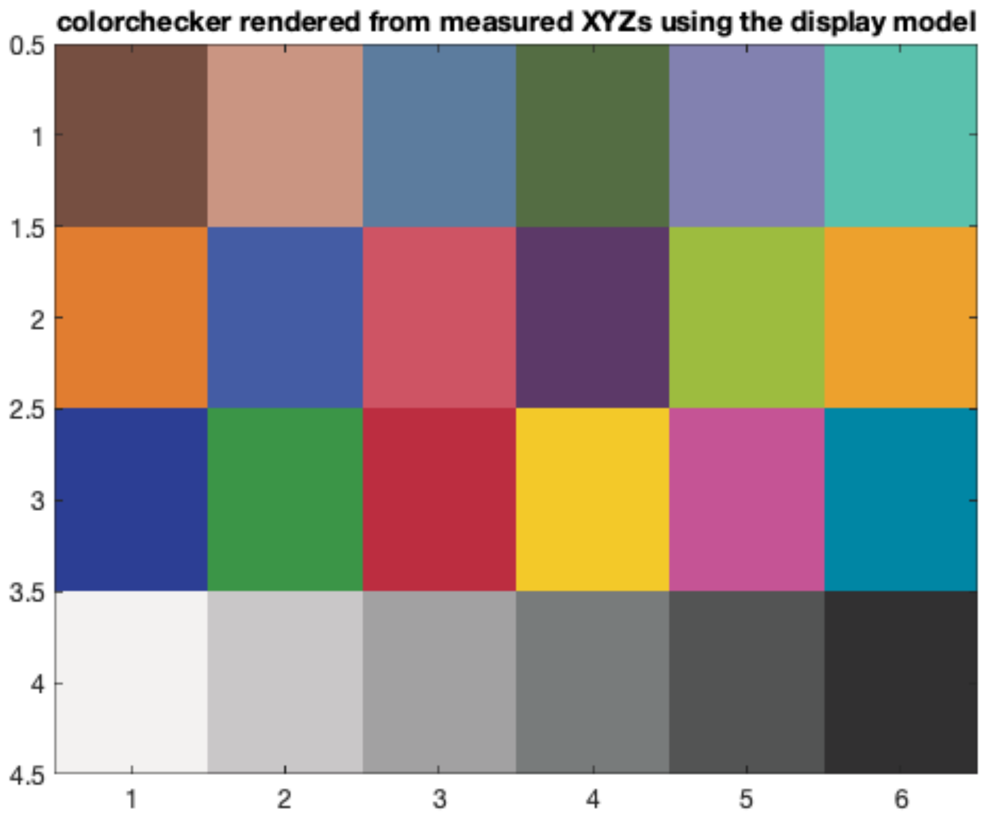
```
%a) Rename the model components
M_disp = M_rev;
RLUT_disp = RLUT_rev;
GLUT_disp = GLUT_rev;
BLUT_disp = BLUT_rev;
XYZk_disp = XYZk;
XYZw_disp = XYZw;
```

```
% b) Save the display white and black levels, reverse model matrix, and
% reverse LUTs as 'display_model.mat'
save('display_model.mat','XYZw_disp', 'XYZk_disp','M_disp','RLUT_disp',...
'GLUT_disp','BLUT_disp')
```

Step 9

```
%a) Run loadCIEdata. Use the loaded data with your ref2XYZ function to
% calculate XYZ values for the D50 illuminant.
cie = loadCIEdata;
XYZn_D50 = ref2XYZ(cie.PRD, cie.cmf2deg, cie.illD50);
% b) Load the ColorMunki measured XYZ and Lab data for the Colorchecker
% patches from the file ?munki_CC_XYZs_Labs.txt?, and extract the patch
% XYZ and Lab value
CM_XYZ_Lab = importdata("munki_CC_XYZs_Labs.txt");
CM_XYZ = CM_XYZ_Lab(:,2:4);
CM_XYZ = CM_XYZ';
CM_Lab = CM_XYZ_Lab(:,5:7);

% c)  Use the catBradford function to adapt the XYZ values from the D50
% illuminant used by the ColorMunki to the whitepoint of your display
% (XYZw_disp).
munki_XYZs_D65 = catBradford(CM_XYZ, XYZn_D50, XYZw_disp);

% d) Adjust the XYZ values by subtracting the display black level
% (XYZk_disp) from each one.
munki_XYZs_D65_adj = munki_XYZs_D65 - XYZk_disp;

% e) Multiply the XYZ values by the display matrix (M_disp) to produce
% linear RGB radiometric scalars (RSs).
%%%%% ISSUES?????
RSs = M_disp * munki_XYZs_D65_adj;

% f) Normalize the RSs by dividing by 100.
RSs = RSs/100;

% g) Clip any RSs that are <0.0 or >1.0.
RSs(RSs>1) = 1;
RSs(RSs<0) = 0;

% h)  Multiply the RSs by 1023, add 1, and round to the nearest integer
RSs = RSs * 1023;
RSs = RSs + 1;
RSs = round(RSs);

% i) Use the scaled RSs to index into the display LUTs to calculate the
% gamma-corrected RGB0-255 digital counts (DCs) for the display.
munki_CC_DCs(1,:) = RLUT_rev(RSs(1,:));
munki_CC_DCs(2,:) = GLUT_rev(RSs(2,:));
munki_CC_DCs(3,:) = BLUT_rev(RSs(3,:));

% j) Reshape the array into a 6x4x3 array
```

```
pix = uint8(reshape(munki_CC_DCs', [6 4 3]));

% k) Visualize the chart patches
pix = fliplr(imrotate(pix, -90));
figure;
image(pix);
set(gca, 'FontSize',12);
title('colorchecker rendered from measured XYZs using the display model');
```



Step 10

```
%a) Convert the RGB0-255 DCs calculated in step 9i) into doubles, multiply
% them by 100/255, to rescale them to a 0-100 range, and then convert them
% to back to unsigned integers (uint8s). The resulting values will be
% RGB0-100 DCs that if sent to your display by the Argyll software should
% reproduce the colors in the ColorChecker chart
double(munki_CC_DCs);
munki_CC_DCs_rescaled = uint8(munki_CC_DCs .* (100/255));

% b-c) Create a matrix ?table4ti1? like the one shown below. The first column is
% the integers 1-30. Columns 2-4 rows 1-24 are the values from step a).
% Rows 25-27 are 0?s and rows 28-30 are 100?s
table4ti = zeros(30, 4);
table4ti(:, 1) = (1:30)';
```

```
table4ti(1:24, 2:4) = munki_CC_DCs_rescaled';
table4ti(28:30, 2:4) = 100;

% d-e)  Use the MATLAB statement ?disp_XYZs = importdata('disp_model_test.ti3',' ',20);?
% to load the measured XYZ values of the displayed patches into your workspace .
% This will create a structure named disp_XYZs. The data will be in disp_XYZs.data.
disp_XYZs = importdata('disp_model_test.ti3',' ',20);

% f) Extract the XYZ data for the displayed CC patches
% (rows 1-24, cols 5-7), and display black and display white (rows 25-27
% and 28-30 cols 5-7 respectively). Average the three measurements of
% display black and display white to estimate XYZk and XYZw.
XYZ_ti3 = disp_XYZs.data(1:24, 5:7);
XYZk_ti3 = disp_XYZs.data(25:27, 5:7);
XYZw_ti3 = disp_XYZs.data(28:30, 5:7);
XYZk_t13_avg = mean(XYZk_ti3);
XYZw_t13_avg = mean(XYZw_ti3);

% g) Use your XYZ2Lab function to calculate Lab values for the displayed CC
% patches from the XYZs. Use XYZw as the reference white.
disp_labs = XYZ2Lab(XYZ_ti3', XYZw_t13_avg);
% h) ) Use deltaEab function to calculate color differences between the real
% Lab values of the CC patches extracted in step 9b) and the displayed
% Lab values calculated in step g
labs_diff = deltaEab(CM_Lab', disp_labs);
% i) Calculate the min, max, and mean delta Es for the comparison
min_lab = min(labs_diff);
max_lab = max(labs_diff);
mean_lab = mean(labs_diff);
% j) Summarize the differences between the real and displayed patches
print_display_model_error(CM_Lab', disp_labs, labs_diff);
```

```
Display model color error
XYZ_real->display_model->RGB_disp->display

              Real vs. displayed ColorChecker Lab values
                    real                        displayed
patch #      L        a        b        L        a        b        dEab
      1   37.1865  14.9985  15.2592  37.8185  13.5801  15.3640   1.5564
      2   65.8188  16.8695  18.0267  67.2161  16.7859  18.9165   1.6588
      3   49.9949  -3.1841 -23.5159  50.9431  -4.7105 -22.6126   2.0112
      4   42.6411 -15.3251  20.0423  42.7965 -16.8444  18.1699   2.4162
      5   54.6852   9.6978 -26.7126  55.7283   7.6562 -25.3616   2.6611
      6   71.2441 -33.1391  -0.5010  71.7205 -37.9279  -1.2914   4.8769
      7   62.2558  34.1094  57.7774  63.8030  36.5834  57.0799   3.0001
      8   39.5890   9.9980 -43.6388  41.7331   7.7538 -40.6801   4.2880
      9   51.8424  48.1403  16.0636  53.8793  48.9684  18.2009   3.0664
     10   29.4495  22.4255 -21.7661  32.1915  18.8016 -17.6742   6.1151
     11   71.6264 -24.3441  57.6850  71.4654 -28.1063  53.6931   5.4878
     12   72.2288  20.6039  69.0149  73.8846  20.4913  66.4719   3.0367
     13   28.6402  18.5907 -51.4092  31.4067  15.1380 -46.6034   6.5322
```

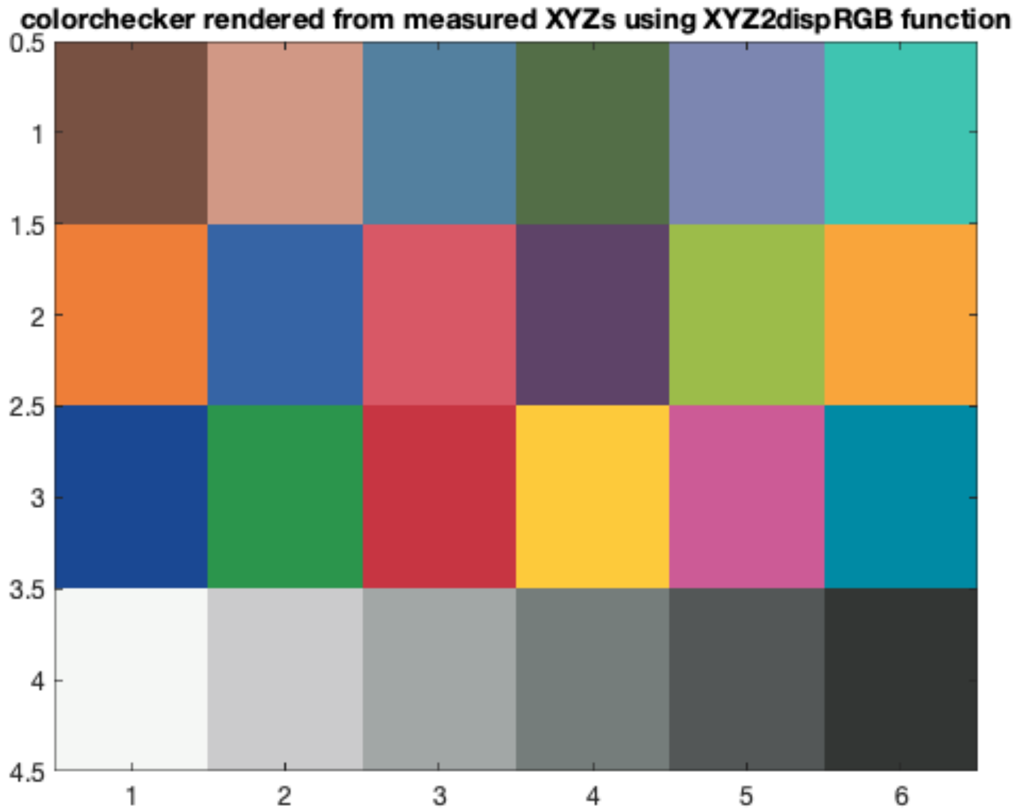
```
      14   54.6309 -39.5493  32.8341  54.2615 -44.5599  30.5310   5.5269
      15   42.5988  54.6049  25.7315  44.9968  54.4685  27.4401   2.9476
      16   82.4265   3.8689  78.8570  83.2793   3.0114  75.5195   3.5499
      17   51.5476  49.5154 -14.3758  53.8885  49.2581 -11.5149   3.7054
      18   49.3892 -26.5473 -28.6645  52.1336 -18.6884 -24.0063   9.5390
      19   95.4458  -0.4414   0.0244  96.7889  -1.6979   0.8169   2.0027
      20   80.0339   0.1309  -0.9345  81.1299  -0.2348  -1.0393   1.1602
      21   66.0107  -0.0004  -1.1463  67.3709  -2.1923  -0.9313   2.5886
      22   50.5546  -0.6207  -0.9616  51.0754  -3.2087  -0.0179   2.8035
      23   35.1532  -0.0632  -0.9708  36.0412  -1.0851  -0.7419   1.3731
      24   20.3224  -0.2858  -0.5603  22.3840  -2.0983   1.1051   3.2107

                                                     min       1.1602
                                                     max       9.5390
                                                     mean      3.5464
```

Step 11

```
function disp_RGBs = XYZ2dispRGB(display_model,XYZs, XYZn)
%XYZ2DISPRGB Takes as input the ?display_model.mat? created in step 8), XYZs, a
% 3xn array of XYZ values, and XYZn, a 3x1 vector that represents the XYZs
% of reference white, and returns a 3xn array of RGBs0-255 ready for display.
munki_XYZs_D65 = catBradford(XYZs, XYZn, display_model.XYZw_disp);
munki_XYZs_D65_adj = munki_XYZs_D65 - display_model.XYZk_disp;
RSs = display_model.M_disp * munki_XYZs_D65_adj;
RSs = RSs/100;
RSs(RSs>1) = 1;
RSs(RSs<0) = 0;
RSs = RSs * 1023;
RSs = RSs + 1;
RSs = round(RSs);
munki_CC_DCs(1,:) = display_model.RLUT_disp(RSs(1,:));
munki_CC_DCs(2,:) = display_model.GLUT_disp(RSs(2,:));
munki_CC_DCs(3,:) = display_model.BLUT_disp(RSs(3,:));

disp_RGBs = uint8(reshape(munki_CC_DCs', [6 4 3]));

end
```

```
% a) Create the XYZ2dispRGB function
% b) Confirm that your function works by using it to render an image of the
% ColorChecker chart like the one shown below from the D50 referenced XYZ
% data in ?munki_CC_XYZs_Labs.txt? (also loaded in step 9b).
% c) Include a listing of your function in your report.

display_model = load('display_model.mat');
disp_RGBs = XYZ2dispRGB(display_model, XYZ_ti3', XYZw);
new_pix = fliplr(imrotate(disp_RGBs, -90));
figure;
image(new_pix);
```

```
set(gca, 'FontSize',12);
title('colorchecker rendered from measured XYZs using XYZ2dispRGB function');
```



Feedback

Malcolm did steps 1-7. Max did steps 8-12. There were some technical issues with this project. Malcolm's display broke in the middle of gathering data, so Max had to retake the data on his display. Other than this technical issue there were no serious problems with this project. Max found it useful to continue to get experience using the ColorMunki.

Published with MATLAB® R2019b