Camera Characterization Lab

By Max Shooster and Malcolm Zale

Team 4

In this project we characterize our digital camera so that it can be used to estimate the XYZ values of imaged surfaces. First, using the image of the ColorChecker chart created in Project 1, we extract the RGB values of the color patches, and use these values together with measured XYZ values for the patches to plot our camera?s tone transfer functions (TTFs). You will then fit functions to the TTFs and use them to linearize the camera?s response with respect to relative luminance. You will plot these linearized response functions and create images that visualize the camera?s original and linearized RGB responses to the patches in the ColorChecker chart. You will then use the linearized RGB data for the ColorChecker patches and measured XYZ values for the patches to derive a matrix that transforms linearized camera RGBs to estimated XYZs. You will then check the accuracy of this transformation by calculating ?Eab color differences between the ColorMunki-measured and camera-estimated patch values and visualizing images of the ColorChecker from both sets of values. Finally you will use your camera model to estimate the XYZ and Lab values of your colored patches from the image you created in Project 1 and will compare the ColorMunki-measured and camera-estimated values both numerically and visually.

Contents

Steps 1 - 3

```
% a) Used Photoshop to find the average RGB values for each of the patches in
% the chart. Manually define all of the average RGB values for each patch
cam_rgbs = [88, 214, 86, 77, 130, 126, 222, 51, 195, 54, 171, 238, 16, 61, ...
    148, 237, 169, 36, 235, 184, 134, 80, 33, 10; 59, 153, 111, 91, 124, ...
```

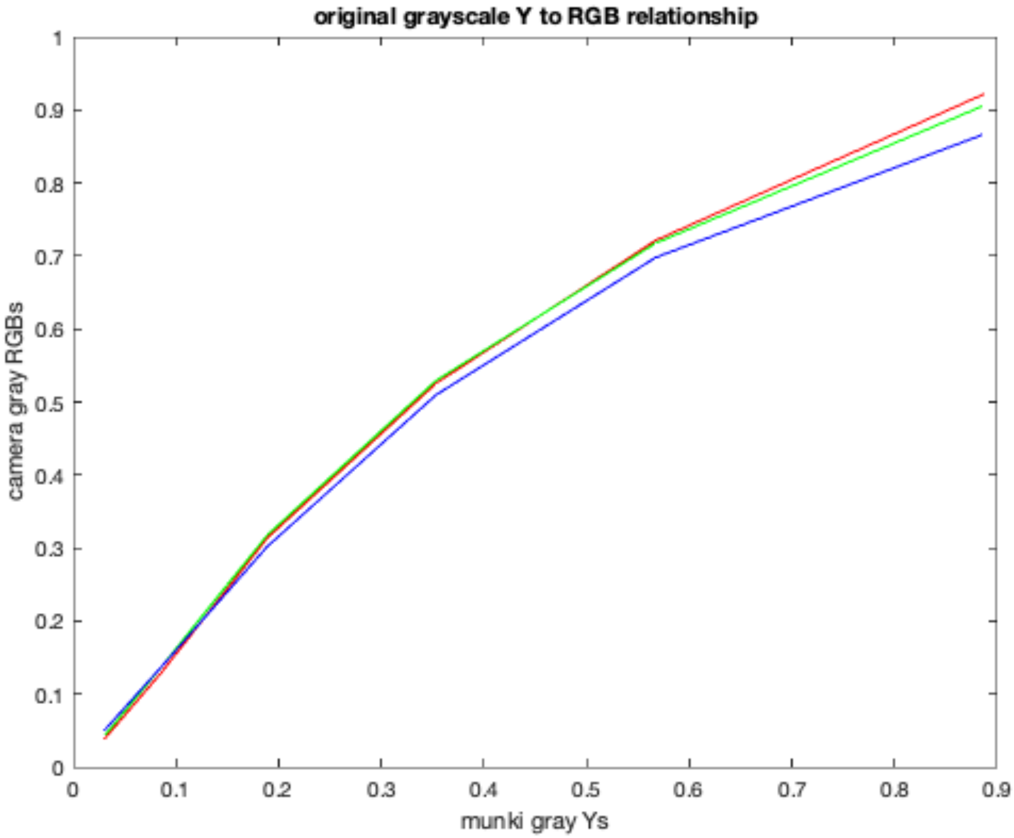
```
    204, 123, 63, 76, 34, 198, 175, 30, 120, 40, 208, 67, 107, 231, 183, ...
    135, 81, 35, 11; 47, 134, 142, 53, 174, 184, 53, 142, 83, 60, 80, 60, ...
    97, 57, 41, 59, 122, 142, 221, 178, 130, 77, 35, 13];

% b) Normalize these RGBs by dividing by them by 255.
norm_cam_rgbs = cam_rgbs./255;

% c) Extract the normalized RGB values for the gray patches (#19-24).
norm_cam_gray = norm_cam_rgbs(:,19:24);

% d) L/R flip the resulting array so the values run from low (black) to high (white)
low2high_gray= fliplr(norm_cam_gray);
```

Step 4

```
% a) Load the ColorMunki-measured XYZ and Lab values of the ColorChecker
% chart provided in the file ?munki_CC_XYZs_Labs.txt? into two 3x24 arrays
% named ?munki_XYZs? and ?munki_Labs?.
load 'munki_CC_XYZs_Labs.txt'
munki_XYZs = (munki_CC_XYZs_Labs(:,2:4))';
munki_Labs = (munki_CC_XYZs_Labs(:,5:7))';

% b) Extract the Y values for the gray patches (#19-24), and
% c)Normalize by dividing the values by 100, and
% d) L/R flip the resulting vector so the entries run from low (black)
% to high (white)
gray_Ys = fliplr(munki_XYZs(2,19:24))./100;
```

Step 5

```
% Plot the normalized ColorMunki-measured gray-patch Ys calculated in step 4)
% vs. the normalized camera gray-patch RGBs calculated in step 3).

figure
plot(gray_Ys, low2high_gray(1,:),'r');
hold on;
plot(gray_Ys, low2high_gray(2,:), 'g');
plot(gray_Ys, low2high_gray(3,:), 'b');

title 'original grayscale Y to RGB relationship'
xlabel 'munki gray Ys'
ylabel 'camera gray RGBs'
```

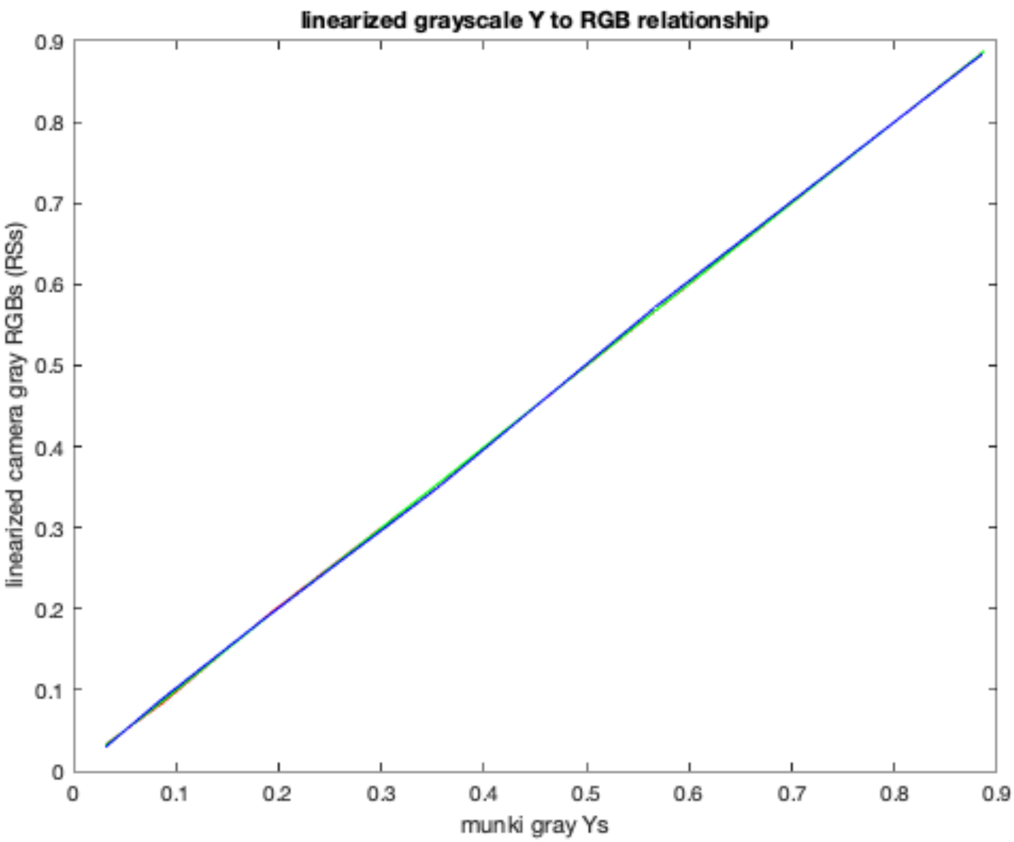


Step 6

```
% a) Fit low order polynomial functions between normalized
% camera-captured gray RGBs and the munki-measured gray Ys

r=1;
g=2;
b=3;

cam_polys(r,:) =polyfit(low2high_gray(r,:), gray_Ys,3);
cam_polys(g,:) =polyfit(low2high_gray(g,:), gray_Ys,3);
cam_polys(b,:) =polyfit(low2high_gray(b,:), gray_Ys,3);


%b) Use the function to linearize the camera data
cam_RSs(r,:) = polyval(cam_polys(r,:), norm_cam_rgbs(r,:));
cam_RSs(g,:) = polyval(cam_polys(g,:), norm_cam_rgbs(g,:));
cam_RSs(b,:) = polyval(cam_polys(b,:), norm_cam_rgbs(b,:));


% c) Clip out-of-range values
cam_RSs(cam_RSs<0) = 0;
cam_RSs(cam_RSs>1) = 1;
```

Step 7

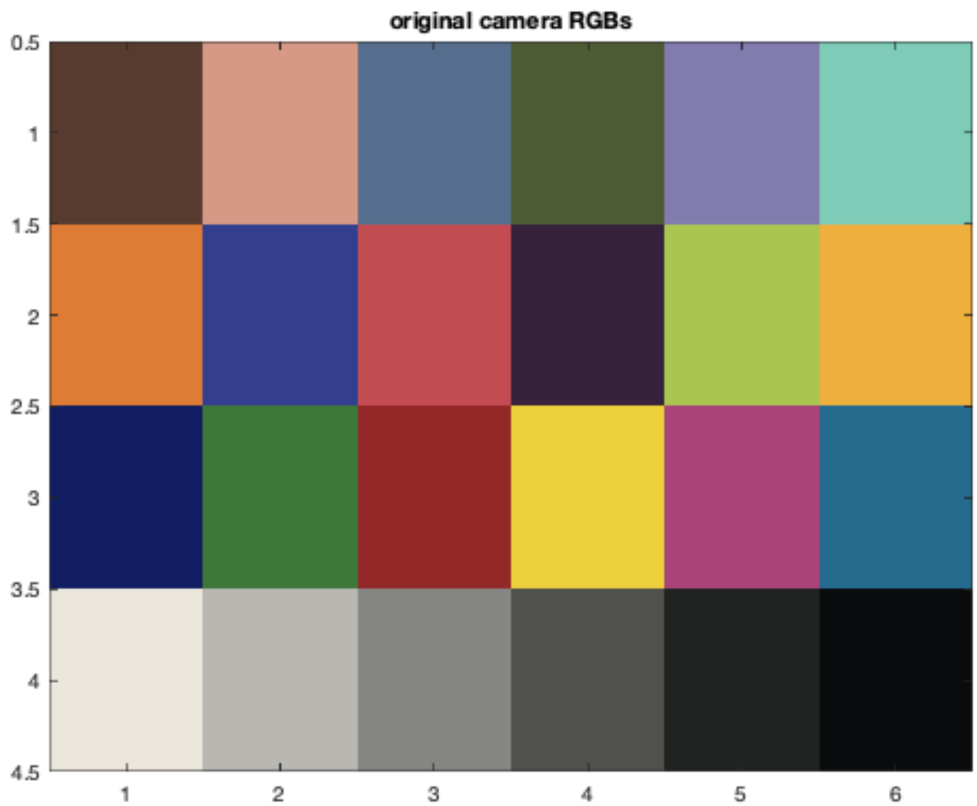
```
% Verify the quality of the linearization process by re-plotting the graph
% from step 5), using the radiometric scalars for the gray patches in place
% of the original values.

figure
plot(gray_Ys, fliplr(cam_RSs(1,19:24)),'r');
hold on
plot(gray_Ys, fliplr(cam_RSs(2,19:24)), 'g');
plot(gray_Ys, fliplr(cam_RSs(3,19:24)), 'b');

title 'linearized grayscale Y to RGB relationship'
xlabel 'munki gray Ys'
ylabel 'linearized camera gray RGBs (RSs)'
```

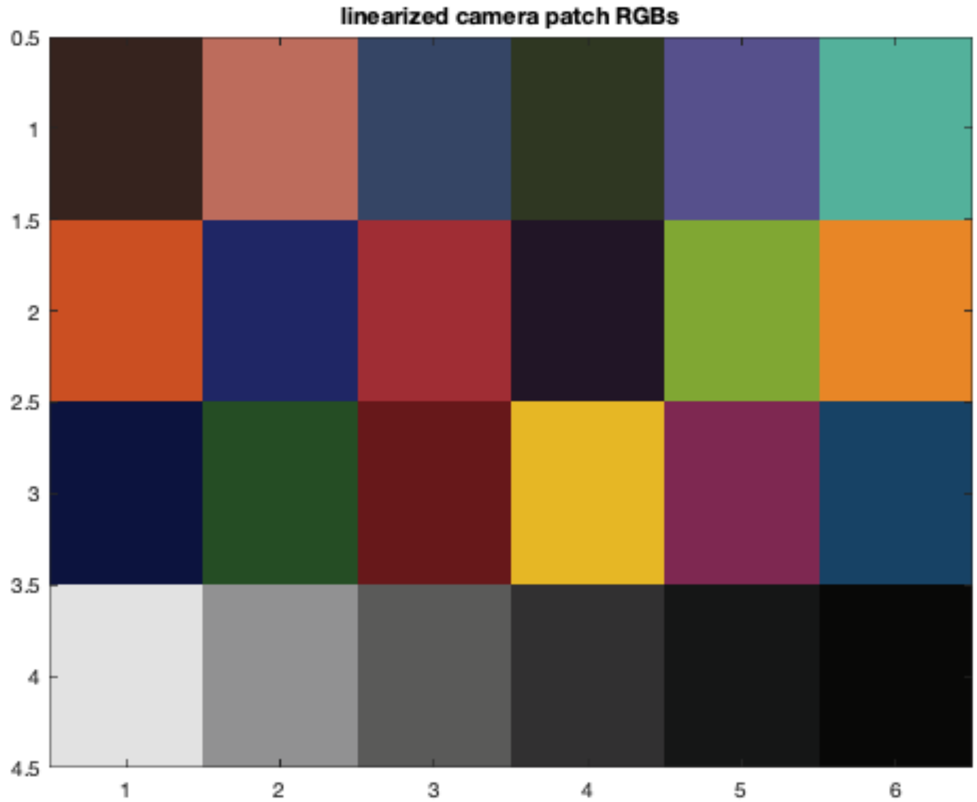


Step 8

```
% a) Visualize the original camera RGBs
pix = reshape(norm_cam_rgbs', [6 4 3]);
pix = uint8(pix*255);
pix = imrotate(pix, -90);
pix = flipdim(pix, 2);
figure;
```

```
image(pix);
title('original camera RGBs');


% b) Visualize the linearized camera RGBs
pix = reshape(cam_RSs', [6 4 3]);
pix = uint8(pix*255);
pix = imrotate(pix, -90);
pix = flipdim(pix, 2);
figure;
image(pix);
title('linearized camera patch RGBs');
```





Step 9

```
% Use the munki-measured ColorChecker XYZs and camera-captured RGB RSs to
% derive a 3x3 matrix that can be used to estimate XYZs from camera RGBs

cam_matrix3x3 = munki_XYZs * pinv(cam_RSs)
```

```
cam_matrix3x3 =

   41.6143   22.9530   16.9022
   16.1207   59.1840    9.5317
   -3.5902   -4.0371   79.0530
```

Step 10

```
% Estimate the ColorChecker XYZs from the linearized camera rgbs using
% the 3x3 camera matrix

cam_XYZs = cam_matrix3x3 * cam_RSs
```

```
cam_XYZs =

  Columns 1 through 7

   13.9774   46.6882   21.5166   14.8407   30.5531   39.6922   42.4064
   12.7078   40.4549   23.2050   16.9950   29.2513   52.0348   32.4306
    7.9882   24.2996   29.3304    8.9225   40.9461   44.1635    6.3497

  Columns 8 through 14

   15.1517   33.6474    9.8060   39.3395   52.4334    7.6903   15.3547
   14.4476   22.5922    8.3956   48.7708   47.2028    7.4130   21.4844
   30.1368   13.2914   10.9867   11.2051    6.3972   18.7431    9.4788

  Columns 15 through 21

   20.7742   56.5110   29.5616   16.3622   72.2176   46.3366   28.6134
   13.1704   58.4902   20.2724   20.6075   75.2092   48.2263   29.8358
    6.3053    5.4545   22.8513   29.7996   63.2177   40.9193   24.8029

  Columns 22 through 24

   15.5493    6.8812    2.5383
   16.1544    7.1910    2.6334
   13.6076    6.3199    2.0474
```

Step 11

```
% a) Use XYZ2Lab function to calculate Lab values from the XYZ values
% estimated in step 10) using D50 for XYZn
cie = loadCIEdata;
XYZn_D50 = ref2XYZ(cie.PRD, cie.cmf2deg, cie.illD50);
Camera_Labs = XYZ2Lab(cam_XYZs, XYZn_D50)

% b) Calculate ?Eab color differences between these estimated Lab values
% and the measured Lab values in the ?munki_Labs? variable created in step 4
CD_CLabs_munkiLabs = deltaEab(Camera_Labs, munki_Labs);

% c) Use the print_camera_model_error function to print a table
print_camera_model_error(munki_Labs, Camera_Labs, CD_CLabs_munkiLabs)
```

```
Camera_Labs =

  Columns 1 through 7

   42.3196   69.7923   55.2829   48.2537   61.0027   77.3016   63.6972
   11.2810   22.8371   -3.9764   -8.9976    8.9711  -30.2123   36.7194
    8.7223   14.8635  -18.7654   15.5046  -25.5680   -1.5095   52.3446
```

```
  Columns 8 through 14

   44.8681   54.6498   34.7935   75.3085   74.3193   32.7293   53.4756
    7.4545   47.4943   14.4511  -22.7261   18.8103    5.1868  -28.4467
  -38.0086   12.9956  -14.5467   54.6344   70.4468  -38.0075   22.5672

  Columns 15 through 21

   43.0188   81.0105   52.1439   52.5173   89.4909   74.9675   61.5123
   45.3574    0.2852   43.4281  -18.5139   -0.6242   -0.4576   -0.5970
   16.8912   86.3971  -12.8672  -24.2853   -1.1155   -1.4564   -0.3248

  Columns 22 through 24

   47.1764   32.2378   18.5116
   -0.1552   -0.5242   -0.0155
   -0.7457   -1.7623    1.1724
```

```
Camera model color error
camera->camera_RGBs->camera_model->estimated_XYZs

        colormunki measured vs. camera estimated ColorChecker Lab values
                   measured                    estimated
patch #      L        a        b        L        a        b       dEab
      1   37.1865  14.9985  15.2592  42.3196  11.2810   8.7223   9.1050
      2   65.8188  16.8695  18.0267  69.7923  22.8371  14.8635   7.8363
      3   49.9949  -3.1841 -23.5159  55.2829  -3.9764 -18.7654   7.1525
      4   42.6411 -15.3251  20.0423  48.2537  -8.9976  15.5046   9.5983
      5   54.6852   9.6978 -26.7126  61.0027   8.9711 -25.5680   6.4614
      6   71.2441 -33.1391  -0.5010  77.3016 -30.2123  -1.5095   6.8027
      7   62.2558  34.1094  57.7774  63.6972  36.7194  52.3446   6.1971
      8   39.5890   9.9980 -43.6388  44.8681   7.4545 -38.0086   8.1264
      9   51.8424  48.1403  16.0636  54.6498  47.4943  12.9956   4.2085
     10   29.4495  22.4255 -21.7661  34.7935  14.4511 -14.5467  12.0112
     11   71.6264 -24.3441  57.6850  75.3085 -22.7261  54.6344   5.0480
     12   72.2288  20.6039  69.0149  74.3193  18.8103  70.4468   3.1044
     13   28.6402  18.5907 -51.4092  32.7293   5.1868 -38.0075  19.3905
     14   54.6309 -39.5493  32.8341  53.4756 -28.4467  22.5672  15.1662
     15   42.5988  54.6049  25.7315  43.0188  45.3574  16.8912  12.8002
     16   82.4265   3.8689  78.8570  81.0105   0.2852  86.3971   8.4676
     17   51.5476  49.5154 -14.3758  52.1439  43.4281 -12.8672   6.2997
     18   49.3892 -26.5473 -28.6645  52.5173 -18.5139 -24.2853   9.6695
     19   95.4458  -0.4414   0.0244  89.4909  -0.6242  -1.1155   6.0658
     20   80.0339   0.1309  -0.9345  74.9675  -0.4576  -1.4564   5.1271
     21   66.0107  -0.0004  -1.1463  61.5123  -0.5970  -0.3248   4.6115
     22   50.5546  -0.6207  -0.9616  47.1764  -0.1552  -0.7457   3.4170
     23   35.1532  -0.0632  -0.9708  32.2378  -0.5242  -1.7623   3.0559
     24   20.3224  -0.2858  -0.5603  18.5116  -0.0155   1.1724   2.5208

                                                          min      2.5208
                                                          max     19.3905
```

```
                                                   mean     7.5935
```

Step 12

```
% a) Split the radiometric scalars (cam_RSs) into r,g,b vectors
RSrgbs = cam_RSs;
RSrs = RSrgbs(1,:);
RSgs = RSrgbs(2,:);
RSbs = RSrgbs(3,:);

% Create vectors of these RSs with multiplicative terms to% represent
% interactions and square terms to represent non-linearities in% the
% RGB-to-XYZ relationship
RSrgbs_extd = [RSrgbs; RSrs.*RSgs; RSrs.*RSbs; RSgs.*RSbs; RSrs.*RSgs.*RSbs;
    RSrs.^2;  RSgs.^2; RSbs.^2;  ones(1,size(RSrgbs,2))];

% Find the extended (3x11) matrix that relates the RS and XYZ datasets
cam_matrix = munki_XYZs * pinv(RSrgbs_extd)
```

```
cam_matrix =

  Columns 1 through 7

   74.8642  -13.0139   31.1955  127.3028  -52.9710   78.5354   44.2934
   34.4786   46.3482   16.5750  103.0286  -28.4375   65.2629   45.8865
    6.5713  -30.6177  118.2509   89.6041  -40.3982   63.6034   60.3353

  Columns 8 through 11

  -69.9971  -64.8468  -52.3993   -0.9157
  -52.1577  -73.9694  -49.9377   -0.9952
  -37.1186  -49.0918  -90.9693   -1.6496
```

Step 13

```
% Estimate XYZs from the RSs using the extended matrix and RS representation
cam_XYZs = cam_matrix * RSrgbs_extd;
cam_XYZs
```

```
cam_XYZs =

  Columns 1 through 7

   15.5520   46.2076   19.6337   14.1032   27.0306   32.7708   37.7963
   13.7395   41.6311   21.3162   16.9564   25.5598   44.2353   28.6506
    8.4291   25.8746   28.5486    8.1627   35.8771   37.3173    2.3940
```

```
  Columns 8 through 14

   12.6629   29.5928   10.9537   36.6052   50.9990    6.8008   10.2048
   11.0408   18.5797    8.7087   45.7014   46.3052    5.7975   18.5017
   29.2963    9.8487   12.3805    8.7041    6.2207   20.4481    6.0035

  Columns 15 through 21

   22.4068   58.2716   26.4801   11.1557   85.4466   48.9467   29.4791
   12.9866   58.4116   16.8129   16.0120   88.8948   51.0272   30.8189
    5.4246    7.2853   20.2511   27.5719   73.2977   42.3525   25.8655

  Columns 22 through 24

   15.9246    6.6986    1.9561
   16.5713    7.0038    2.0015
   14.3907    6.2440    0.9809
```

Step 14

```
% a) use XYZ2Lab function to calculate Lab values from the XYZ values estimated in step 13)
Camera_Labs_Update = XYZ2Lab(cam_XYZs, XYZn_D50);

% b) calculate ?E color differences between these estimated Lab values and
% the measured Lab values provided in the file ?munki_CC_XYZs_Labs.txt?.
CD_CLabsUp_munkiLabs = deltaEab(Camera_Labs_Update, munki_Labs);

% c) Use the print_camera_model_error function to print a table
print_camera_model_error(munki_Labs, Camera_Labs_Update, CD_CLabsUp_munkiLabs)
```

```
Camera model color error
camera->camera_RGBs->camera_model->estimated_XYZs

        colormunki measured vs. camera estimated ColorChecker Lab values
                   measured                    estimated
patch #      L        a        b        L        a        b       dEab
      1   37.1865  14.9985  15.2592  43.8570  14.1684   9.7138   8.7142
      2   65.8188  16.8695  18.0267  70.6158  17.9354  13.4686   6.7025
      3   49.9949  -3.1841 -23.5159  53.2938  -4.5206 -20.9249   4.4026
      4   42.6411 -15.3251  20.0423  48.2050 -13.3015  18.2057   6.1988
      5   54.6852   9.6978 -26.7126  57.6168   9.9272 -24.5823   3.6312
      6   71.2441 -33.1391  -0.5010  72.3855 -32.0364  -1.1198   1.7034
      7   62.2558  34.1094  57.7774  60.4721  36.3101  70.3965  12.9331
      8   39.5890   9.9980 -43.6388  39.6490  14.2854 -45.6655   4.7427
      9   51.8424  48.1403  16.0636  50.1918  51.9606  15.6571   4.1815
     10   29.4495  22.4255 -21.7661  35.4173  20.5339 -17.6189   7.5095
     11   71.6264 -24.3441  57.6850  73.3513 -23.0893  59.5600   2.8400
```

```
    12   72.2288   20.6039   69.0149   73.7431   17.5376   70.2451    3.6343
    13   28.6402   18.5907  -51.4092   28.8958   13.0708  -48.2101    6.3851
    14   54.6309  -39.5493   32.8341   50.0990  -48.3989   30.4744   10.2187
    15   42.5988   54.6049   25.7315   42.7430   54.2014   20.5667    5.1826
    16   82.4265    3.8689   78.8570   80.9671    4.7736   78.1324    1.8637
    17   51.5476   49.5154  -14.3758   48.0235   49.0412  -14.8265    3.5844
    18   49.3892  -26.5473  -28.6645   46.9902  -27.8699  -30.1734    3.1275
    19   95.4458   -0.4414    0.0244   95.5365   -0.4956    0.0573    0.1106
    20   80.0339    0.1309   -0.9345   76.6955   -0.6866   -0.3039    3.4944
    21   66.0107   -0.0004   -1.1463   62.3544   -0.8967   -0.7594    3.7844
    22   50.5546   -0.6207   -0.9616   47.7153   -0.3055   -1.8811    3.0011
    23   35.1532   -0.0632   -0.9708   31.8157   -0.5556   -2.1490    3.5735
    24   20.3224   -0.2858   -0.5603   15.4951    0.6127    8.6593   10.4456

                                                              min       0.1106
                                                              max      12.9331
                                                              mean      5.0819
```

Step 15

```
% Save the (extended) camera model for use in later projects
save('cam_model.mat', 'cam_polys', 'cam_matrix');
```

Step 16

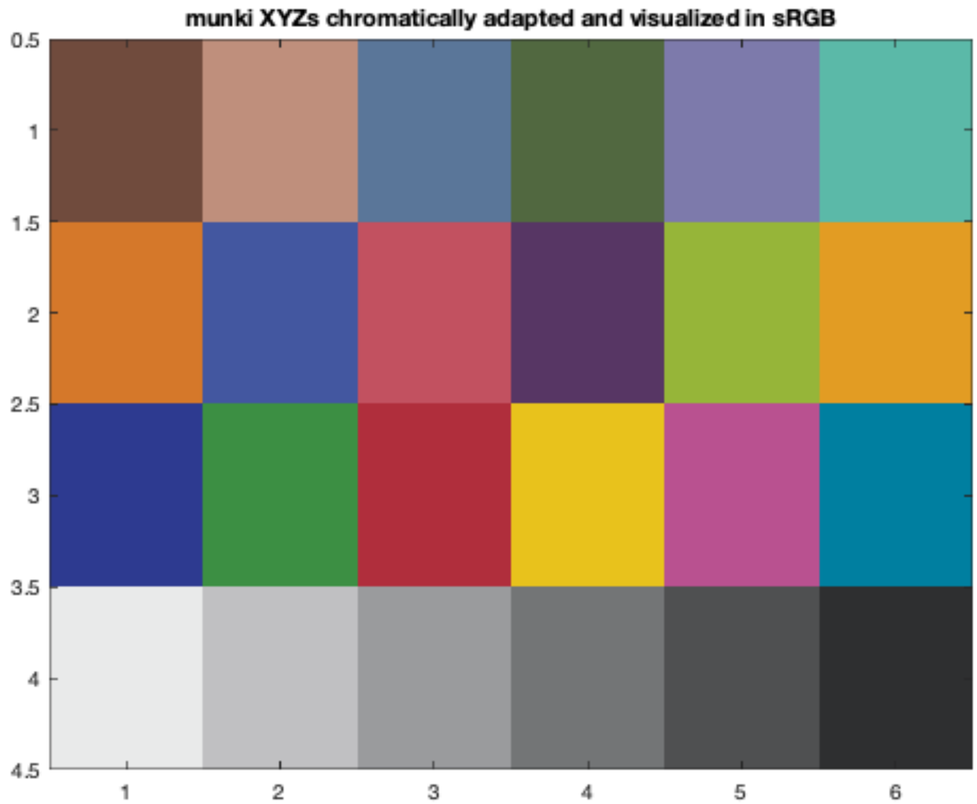
```
% Test that the camRGB2XYZ function works correctly
cam_XYZs = camRGB2XYZ('cam_model.mat', cam_rgbs)
```

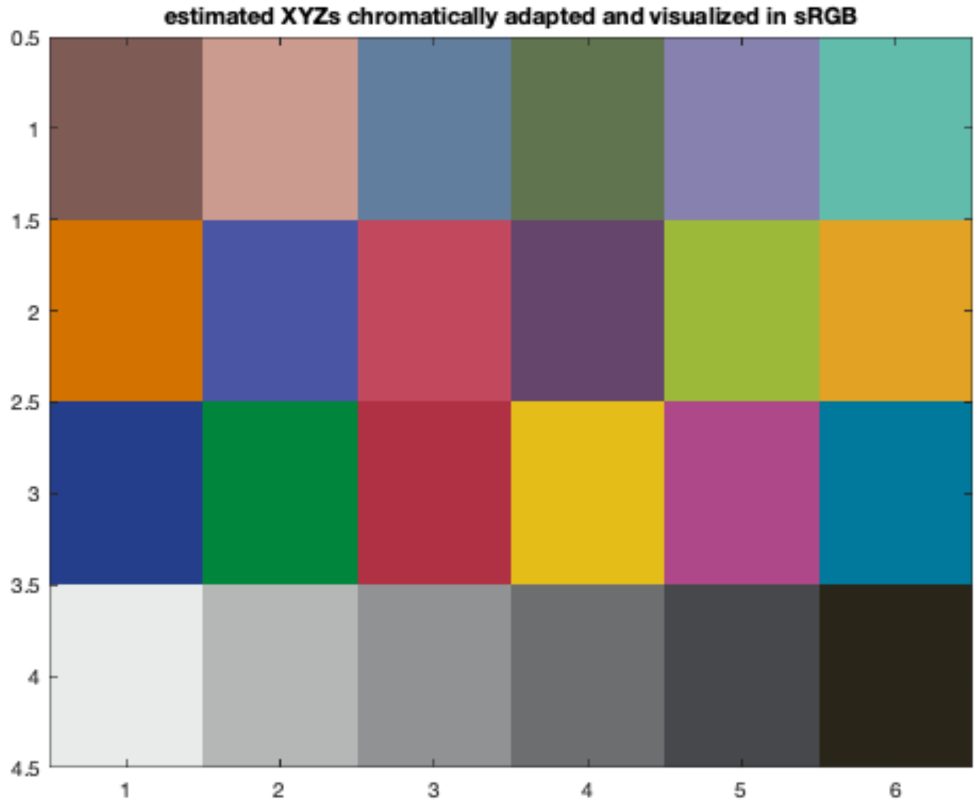
```
cam_XYZs =

  Columns 1 through 7

   15.5520   46.2076   19.6337   14.1032   27.0306   32.7708   37.7963
   13.7395   41.6311   21.3162   16.9564   25.5598   44.2353   28.6506
    8.4291   25.8746   28.5486    8.1627   35.8771   37.3173    2.3940

  Columns 8 through 14

   12.6629   29.5928   10.9537   36.6052   50.9990    6.8008   10.2048
   11.0408   18.5797    8.7087   45.7014   46.3052    5.7975   18.5017
   29.2963    9.8487   12.3805    8.7041    6.2207   20.4481    6.0035

  Columns 15 through 21

   22.4068   58.2716   26.4801   11.1557   85.4466   48.9467   29.4791
   12.9866   58.4116   16.8129   16.0120   88.8948   51.0272   30.8189
    5.4246    7.2853   20.2511   27.5719   73.2977   42.3525   25.8655

  Columns 22 through 24
```

```
15.9246    6.6986    1.9561
16.5713    7.0038    2.0015
14.3907    6.2440    0.9809
```

Step 17

```
% a) Visualize the munki-measured XYZs as an sRGB image
XYZn_D65 = ref2XYZ(cie.PRD, cie.cmf2deg, cie.illD65);
munki_XYZs_D65 = catBradford(munki_XYZs, XYZn_D50, XYZn_D65);
munki_XYZs_sRGBs = XYZ2sRGB(munki_XYZs_D65);
pix = reshape(munki_XYZs_sRGBs', [6 4 3]);
pix = uint8(pix*255);
pix = imrotate(pix, -90);
pix = flipdim(pix,2);
figure;
image(pix);
title('munki XYZs chromatically adapted and visualized in sRGB');

% b) Visualize the camera-estimated XYZs as an sRGB image
cam_XYZs_D65 = catBradford(cam_XYZs, XYZn_D50, XYZn_D65);
cam_XYZs_sRGBs = XYZ2sRGB(cam_XYZs_D65);
pix = reshape(cam_XYZs_sRGBs', [6 4 3]);
pix = uint8(pix*255);
pix = imrotate(pix, -90);
pix = flipdim(pix,2);
figure;
image(pix);
title('estimated XYZs chromatically adapted and visualized in sRGB')
```


munki XYZs chromatically adapted and visualized in sRGB
0.5
1
1.5
2
2.5
3
3.5
4
4.5
1
2
3
4
5
6



Feedback

Malcolm did steps 1-10. Max did steps 11-18. Neither Malcolm nor Max had any problems with this project. We both agreed however that while we appreciated being given so much assistance with the provided code -- especially during this very busy moment in the semester -- this partially took away from the experience of writing it ourselves, and we feel that it may have been more helpful if we were given more room to develop this ourselves so that we could further our understanding of the material. Overall, however, we still agreed that it was a fun project and it was interesting to actually apply our knowledge of color science to this specific application.

Published with MATLAB® R2019b