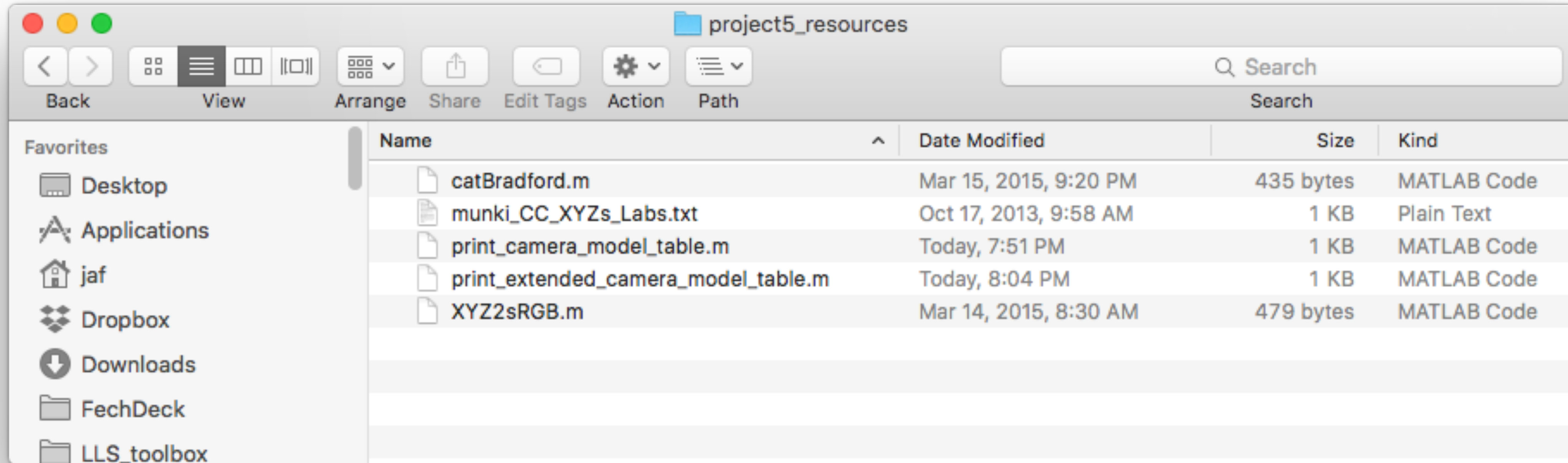
Project 5: Camera characterization

In this project you will characterize your digital camera so that it can be used to estimate the XYZ values of imaged surfaces. First, using the image of the ColorChecker chart you created in Project 1, you will extract the RGB values of the color patches, and use these values together with measured XYZ values for the patches to plot your 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 ΔE_{ab} 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.

1) Download the project5_resources.zip file from myCourses and unzip its contents to your working directory.



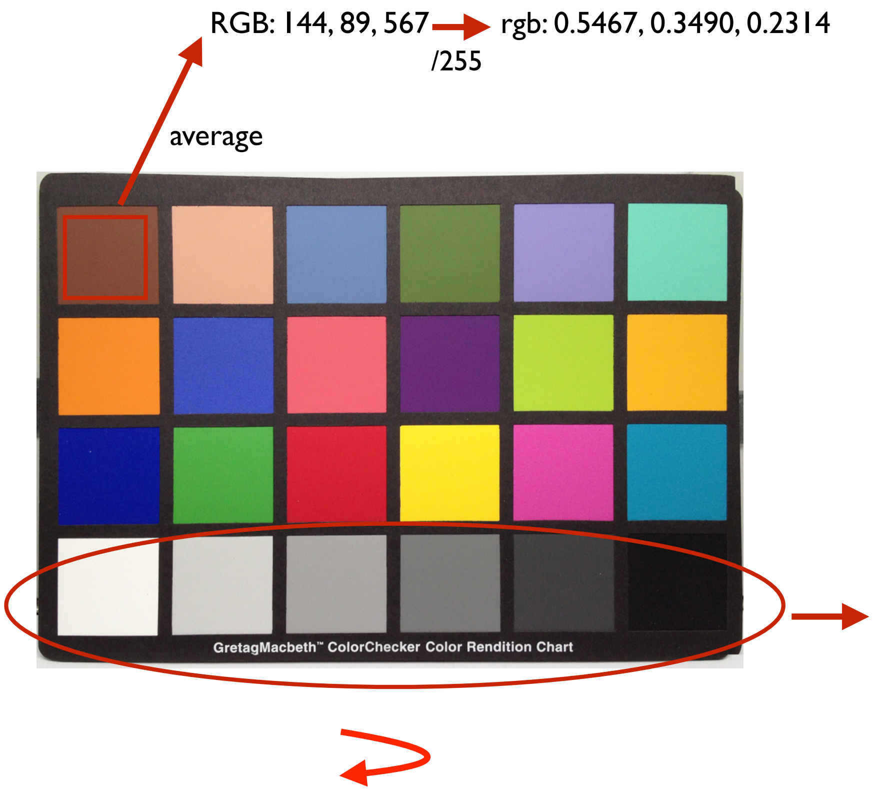
2) a) To characterize your camera you will need a high quality image of the ColorChecker chart in the D50 lightbooth. You took this image in Project 1 and processed it in Project 2, but if the quality of the image you have is poor you will need to take it again. ***It is important that this image is of high quality since the next three projects depend on it!*** Your image of the ColorChecker should be large, uniformly lit, well exposed, well cropped, and rectangular. If you haven't done so already, b) Use a photo-editor to crop and rectify the image of the chart area. c) Downsample the cropped/rectified chart image to 1125x800 and save it as a max quality .jpg or .png

before

after

1125x800

3) a) Use your favorite method to find the <u>average</u> RGB values for each of the patches in the chart. b) Normalize these RGBs by dividing by them by 255. c) Extract the normalized RGB values for the gray patches (#19-24). d) L/R flip the resulting array so the values run from low (black) to high (white).

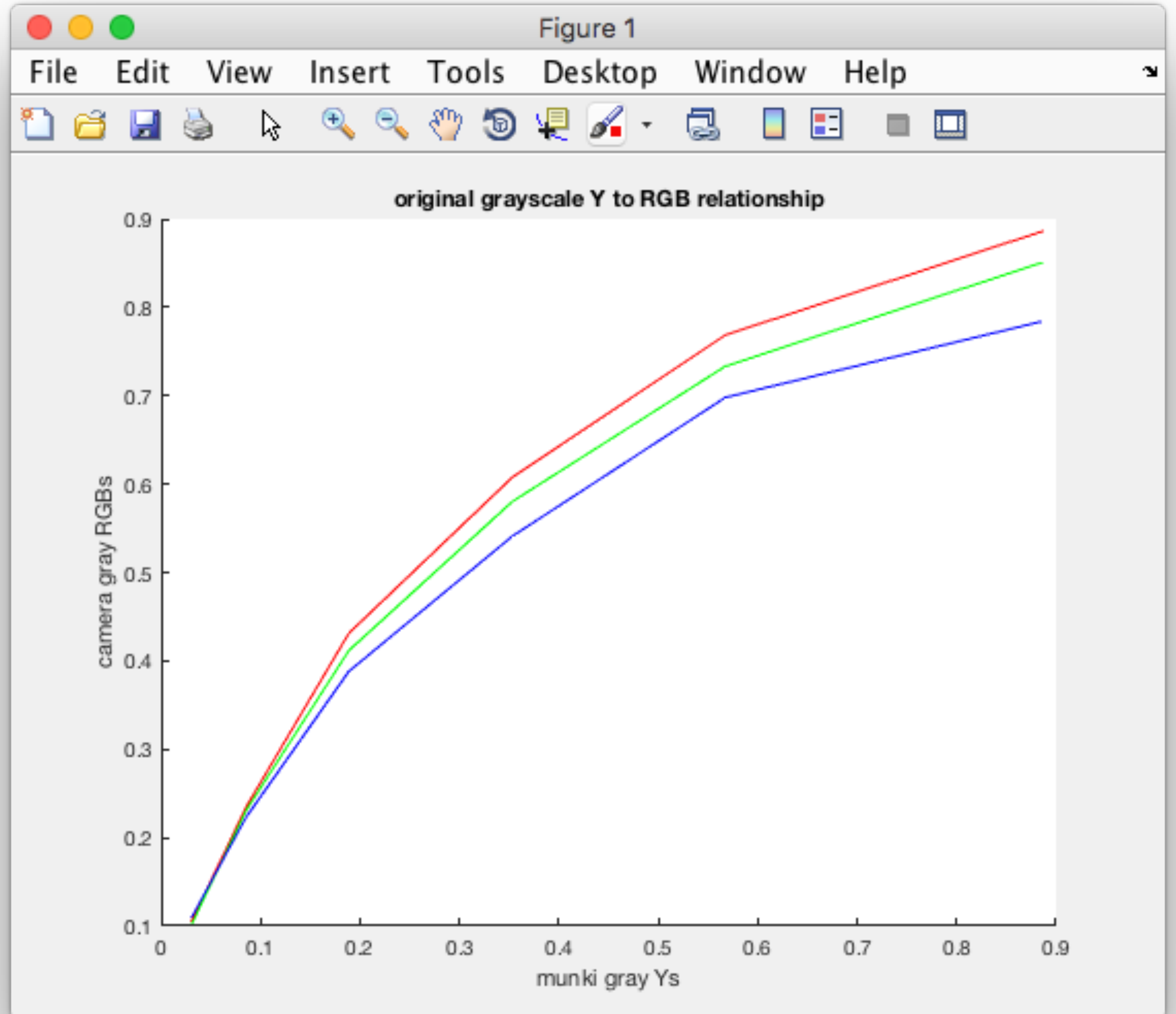



```
cam_RGBs =

   144    254    114    114    168    122   . . .
    89    179    143    130    148    225   . . .
    67    148    184     67    205    196   . . .
 . . .


cam_rgbs =

    0.5647    0.9961    0.4471    0.4471    0.6588    0.4784 . . .
    0.3490    0.7020    0.5608    0.5098    0.5804    0.8824 . . .
    0.2627    0.5804    0.7216    0.2627    0.8039    0.7686 . . .

  . . .


cam_gray_rgbs =

    0.1059    0.2353    0.4314    0.6078    0.7686    0.8863
    0.1020    0.2314    0.4118    0.5804    0.7333    0.8510
    0.1098    0.2235    0.3882    0.5412    0.6980    0.7843
```

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’. b) Extract the Y values for the gray patches (#19-24). c) Normalize by dividing the values by 100. d) L/R flip the resulting vector so the entries run from low (black) to high (white).


```
% load the munki-measured XYZ and Lab data
% for the ColorChecker (D50 reference white)

% extract the Ys for the six gray patches
% normalize the data and transpose and reverse the
% vector for plotting

munki_gray_Ys =

    0.0307    0.0858    0.1889    0.3534    0.5674    0.8868
```

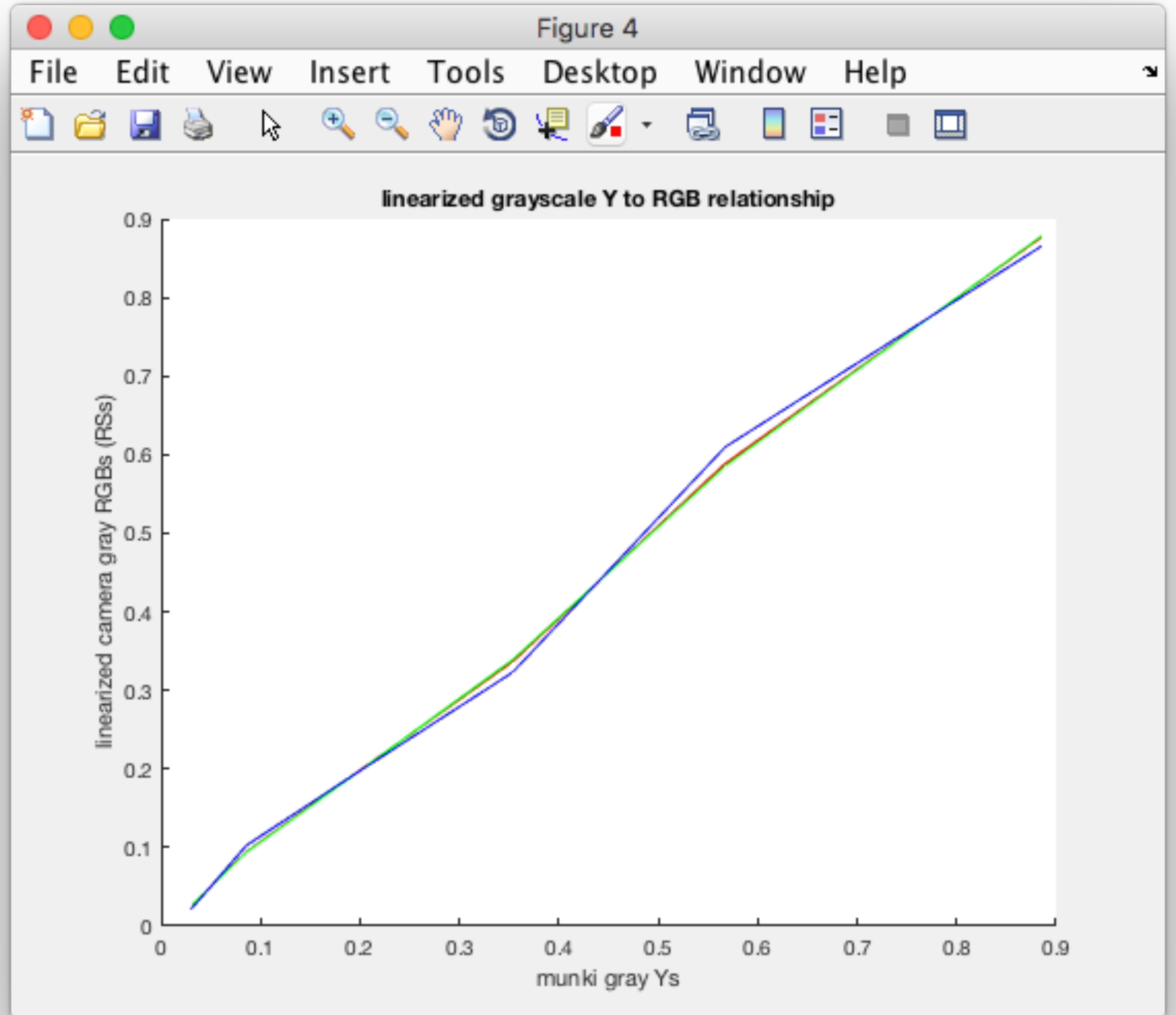
5) Plot the normalized ColorMunki-measured gray-patch Ys you calculated in step 4) vs. the normalized camera gray-patch RGBs you calculated in step 3). There will be separate curves for each of the R,G, and B channels. These curves are the tone transfer functions (TTFs) of your camera and show the (typically non-linear) response of the camera with respect to (relative) luminance.



6) The first step in camera characterization is to linearize the camera's RGB response with respect to relative luminance (Y). To do this a) fit polynomial functions between the camera-captured gray-patch RGBs and the ColorMunki-measured gray-patch Ys, then b) use these functions to linearize the camera's responses to the ColorChecker patches (all 24 of them, not just the grays!!!). Finally c) clip out-of-range values, produced by quantization errors in the calculations. The resulting normalized, linearized RGB values are known as **radiometric scalars (RSs)**. MATLAB code that performs these tasks is shown below.

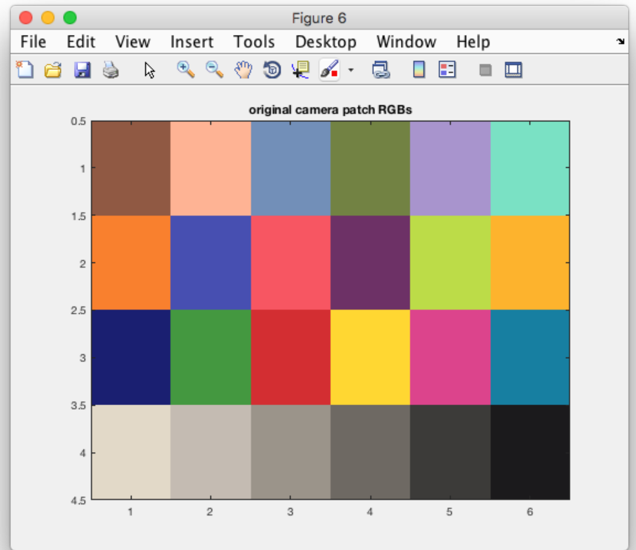
```
r = 1; g = 2; b = 3;

% a) fit low-order polynomial functions between normalized
% camera-captured gray RGBs and the munki-measured gray Ys
cam_polys(r,:)=polyfit(cam_gray_rgbs(r,:),munki_gray_Ys,3);
cam_polys(g,:)=polyfit(cam_gray_rgbs(g,:),munki_gray_Ys,3);
cam_polys(b,:)=polyfit(cam_gray_rgbs(b,:),munki_gray_Ys,3);


% b) use the functions to linearize the camera data
cam_RSs(r,:) = polyval(cam_polys(r,:),cam_rgbs(r,:));
cam_RSs(g,:) = polyval(cam_polys(g,:),cam_rgbs(g,:));
cam_RSs(b,:) = polyval(cam_polys(b,:),cam_rgbs(b,:));

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

7) Verify the quality of the linearization process by re-plotting the graph from step 5), using the <u>radiometric scalars</u> for the gray patches in place of the original values.

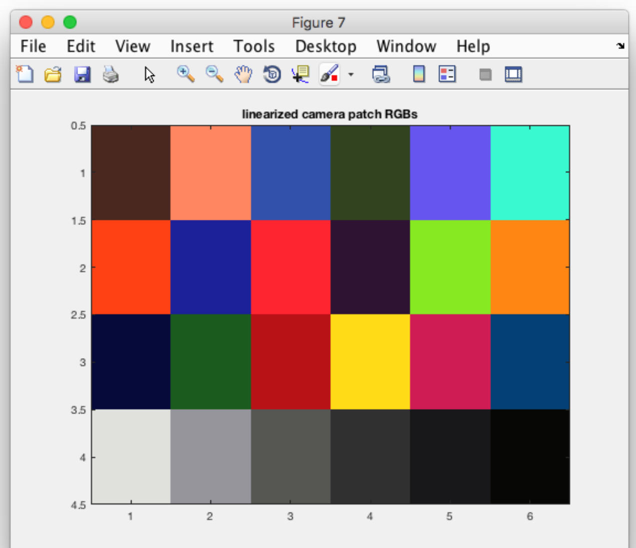


8) To see the impact of the linearization, visualize the pre- and post- linearization ColorChecker patch values by creating images like the ones shown. MATLAB code to do this is shown below. In general, the linearized image should appear darker.

```
% visualize the original camera RGBs
pix = permute(cam_rgbs, [3 2 1]);
pix = reshape(pix, [6 4 3]);
pix = imrotate(pix, -90);
pix = flipdim(pix,2);
figure;
image(pix);
title('original camera patch RGBs');
```




```
% visualize the linearized camera RGBs
pix = permute(cam_RSs, [3 2 1]);
pix = reshape(pix, [6 4 3]);
pix = imrotate(pix, -90);
pix = flipdim(pix,2);
figure;
image(pix);
title('linearized camera patch RGBs');
```



9) a) The second step in camera characterization is to derive a matrix that estimates XYZ values from the RGB radiometric scalars calculated in step 6). Since the two spaces are three dimensional, The most basic matrix is a 3x3. To derive a 3x3 matrix that performs the operation, multiply the ColorChecker XYZ values provided in the file “munki_CC_XYZs_Labs.txt” by the pseudo-inverse of the vector of RGB radiometric scalars calculated in step 6). MATLAB code to do this is shown below. b) Show the resulting matrix in your report.

```
% 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 =

   31.4031   28.6209   11.1268
   17.6185   46.9253    9.1596
    3.0302    4.6695   50.4887
```

10) a) Use this matrix to estimate the XYZ values of the ColorChecker patches from the RGB RSs by multiplying them by the matrix (use the form XYZs = matrix * RSs). b) Show the resulting XYZs in your report.

```
% 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 10

   14.8163   50.5968   22.7289   15.1298   32.7290   43.8609   39.5959   14.0058   37.6238   10.1891
   13.3357   45.6655   24.4596   16.9833   31.5518   57.1161   30.3531   13.6574   26.1255    8.6233
    7.8089   24.4986   35.9943    8.0537   50.1013   46.3002    8.1060   31.2561   13.1119   10.8069

  Columns 11 through 20

   44.2464   47.2311    4.3590   14.9742   25.5988   56.8859   32.3684   12.8920   62.3902   42.0502
   53.4508   42.8947    4.2910   19.8054   16.7715   58.6110   22.5475   16.3905   64.6805   43.4579
   12.5885    9.2262   11.6567    7.9235    6.8883   11.4083   19.7796   24.6233   50.5185   35.3129

  Columns 21 through 24

   23.8533   13.3853    6.8157    1.8394
   24.7809   13.8517    7.0339    1.9225
   18.8996   10.9704    5.9098    1.3346
```

11) Together, the polynomial functions you created in step 6) and the matrix you created in step 9) comprise a basic camera model. To evaluate the accuracy of this model, a) use your XYZ2Lab function to calculate Lab values from the XYZ values you estimated in step 10), and then b) calculate ΔE_{ab} color differences between these <u>estimated</u> Lab values and the <u>measured</u> Lab values in the "munki_Labs" variable created in step 4). c) Use (call don't paste) the **print_camera_model_table function** supplied in the resources to print a table like the one shown below. Consult the function header for usage.

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

measured vs. estimated ColorChecker Lab values

	measured			estimated			
patch #	L	a	b	L	a	b	dEab
1	37.1865	14.9985	15.2592	43.2647	12.3600	11.0441	7.8533
2	65.8188	16.8695	18.0267	73.3279	18.2618	20.5970	8.0581
3	49.9949	-3.1841	-23.5159	56.5450	-3.8241	-26.5951	7.2660
4	42.6411	-15.3251	20.0423	48.2390	-7.2048	18.6777	9.9567
5	54.6852	9.6978	-26.7126	62.9707	8.3958	-33.1911	10.5980
6	71.2441	-33.1391	-0.5010	80.2449	-30.3083	0.9877	9.5521
7	62.2558	34.1094	57.7774	61.9577	35.6242	42.1320	15.7215
8	39.5890	9.9980	-43.6388	43.7375	5.3474	-41.7062	6.5247
9	51.8424	48.1403	16.0636	58.1560	45.7355	19.5328	7.5947
10	29.4495	22.4255	-21.7661	35.2486	15.4900	-13.2020	12.4529
11	71.6264	-24.3441	57.6850	78.1404	-20.1141	55.4496	8.0823
12	72.2288	20.6039	69.0149	71.4835	17.0635	54.4867	14.9719
13	28.6402	18.5907	-51.4092	24.6111	3.0710	-34.1377	23.5669
14	54.6309	-39.5493	32.8341	51.6165	-22.6906	25.0000	18.8329
15	42.5988	54.6049	25.7315	47.9708	45.6217	22.8902	10.8457
16	82.4265	3.8689	78.8570	81.0773	0.9207	63.9630	15.2428
17	51.5476	49.5154	-14.3758	54.6032	43.1770	-2.5027	13.8015
18	49.3892	-26.5473	-28.6645	47.4827	-17.9566	-24.1897	9.8721
19	95.4458	-0.4414	0.0244	84.3188	0.0601	3.1472	11.5677
20	80.0339	0.1309	-0.9345	71.8647	0.4472	0.7813	8.3534
21	66.0107	-0.0004	-1.1463	56.8613	-0.1759	3.2608	10.1570
22	50.5546	-0.6207	-0.9616	44.0195	0.1914	1.4106	6.9996
23	35.1532	-0.0632	-0.9708	31.8839	0.3414	-0.4944	3.3286
24	20.3224	-0.2858	-0.5603	15.0752	-0.3445	3.0013	6.3421
						min	3.3286
						max	23.5669
						mean	10.7309

12) While the basic camera model with a 3x3 matrix is a good first start, it can almost certainly be improved by adding terms to the matrix that compensate for interactions and non-linearities in the relationship between the RGBs and XYZs. To do this a) create the vector RSrgbs_extd that represents the original set of radiometric scalars used to derive the 3x3 matrix plus additional terms that represent products of the individual RGB channels (interactions) and squares of the individual RGB channels (non-linearities). Then b) derive the matrix for this extended model, by multiplying the ColorMunki measured XYZ values by the pseudo-inverse of this vector. The resulting matrix will be 3x11 to account for the added terms. MATLAB code that illustrates the process is shown below. c) Show the resulting matrix in your report.

```
% 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 10

   57.1924   15.6866   27.5017   41.2519  -54.3317   20.1728   84.8493  -31.2641  -29.4963  -24.8269
   36.4723   50.1716   22.5163   28.3379  -50.4702    9.0754   99.5690  -24.1544  -36.4450  -22.8196
   11.2293  -10.5528   91.3596    8.5611  -22.9866    2.1365   92.5498  -13.8286   -2.2205  -63.2025

  Column 11

   -1.5142
   -1.7897
   -0.6266
```

13) a) Use this extended matrix to estimate the XYZ values of the ColorChecker patches from the RGB RSs. To do this you will need to use the extended representation of the radiometric scalars that includes the interaction and square terms (use the form XYZs = matrix * RSs_extd). MATLAB code that illustrates the process is shown below. b) Show the resulting XYZs in your report.

```
% estimate XYZs from the RSs using the extended matrix and RS
% representation
cam_XYZs = cam_matrix3x11 * RSrgbs_extd;

cam_XYZs =

  Columns 1 through 10

   17.8685   53.2620   21.0601   15.7058   24.6994   32.5390   36.9343   12.9566   29.0224   12.6322
   16.0867   49.3503   22.6861   18.7846   23.2761   44.0611   28.1189   11.8791   18.0748   10.2518
   10.2173   30.6058   32.1697    8.9698   32.9292   36.7513    2.8619   30.6696    9.5384   15.2131

  Columns 11 through 20

   36.8726   48.1131    5.2696   11.1980   25.7906   56.8326   24.3970   11.1669   83.4463   47.7134
   46.3709   43.6933    4.7135   18.8110   15.8205   56.5128   14.1945   15.6650   86.9588   49.2909
    9.6988    3.5328   16.5581    6.7608    6.3932    5.9255   18.9884   25.7880   71.9688   41.3940

  Columns 21 through 24

   26.5380   15.1674    7.5146    0.9789
   27.8556   15.9836    7.8676    0.9571
   22.7802   14.1024    7.9686    1.3838
```

14) Now evaluate the accuracy of your extended camera model. a) use your XYZ2Lab function to calculate Lab values from the XYZ values you estimated in step 13), and then b) calculate ΔE color differences between these <u>estimated</u> Lab values and the <u>measured</u> Lab values provided in the file “munki_CC_XYZs_Labs.txt”. c) Use the **print_extended_camera_model_table** function provided in the resources to print a table like the one shown below. The min, max, and mean differences should all be smaller than the ones you calculated in step 11). If not, see me for guidance.

```
Extended camera model color error
camera->camera_RGBs->extended_camera_model->estimated_XYZs

              measured vs. estimated ColorChecker Lab values
                    measured                         estimated
patch #        L         a         b         L         a         b        dEab
      1     37.1865  14.9985  15.2592  47.0881  13.1344   9.0926  11.8129
      2     65.8188  16.8695  18.0267  75.6688  15.1323  14.3582  10.6536
      3     49.9949  -3.1841 -23.5159  54.7475  -3.8287 -24.1197   4.8340
      4     42.6411 -15.3251  20.0423  50.4342 -13.2875  19.0957   8.1105
      5     54.6852   9.6978 -26.7126  55.3556   9.9823 -24.2123   2.6042
      6     71.2441 -33.1391  -0.5010  72.2693 -32.3605  -0.5401   1.2880
      7     62.2558  34.1094  57.7774  59.9960  35.5585  65.8075   8.4669
      8     39.5890   9.9980 -43.6388  41.0233  10.3118 -45.4750   2.3510
      9     51.8424  48.1403  16.0636  49.5867  52.3878  15.6592   4.8263
     10     29.4495  22.4255 -21.7661  38.2905  19.9358 -20.2200   9.3141
     11     71.6264 -24.3441  57.6850  73.7856 -24.0814  56.8383   2.3341
     12     72.2288  20.6039  69.0149  72.0230  17.1763  81.8009  13.2390
     13     28.6402  18.5907 -51.4092  25.9024   9.1335 -44.8391  11.8364
     14     54.6309 -39.5493  32.8341  50.4654 -42.5414  27.7338   7.2331
     15     42.5988  54.6049  25.7315  46.7381  51.7359  22.9109   5.7725
     16     82.4265   3.8689  78.8570  79.9048   5.8435  82.2271   4.6492
     17     51.5476  49.5154 -14.3758  44.5106  55.4266 -18.2247   9.9637
     18     49.3892 -26.5473 -28.6645  46.5319 -25.8130 -27.9036   3.0467
     19     95.4458  -0.4414   0.0244  94.7208  -0.7569  -0.1802   0.8167
     20     80.0339   0.1309  -0.9345  75.6319   0.5194  -0.9204   4.4190
     21     66.0107  -0.0004  -1.1463  59.7581  -1.3025   0.3961   6.5703
     22     50.5546  -0.6207  -0.9616  46.9529  -1.4384  -2.4444   3.9799
     23     35.1532  -0.0632  -0.9708  33.7056  -0.6729  -6.0543   5.3207
     24     20.3224  -0.2858  -0.5603   8.6285   2.1113  -8.7275  14.4637

                                                            min       0.8167
                                                            max      14.4637
                                                            mean      6.5794
```

15) Save your extended camera model (polynomial function coefficients and the 3x11 matrix) in a file called ‘cam_model.mat’ so it can be used in later projects. The MATLAB code to do this shown below.

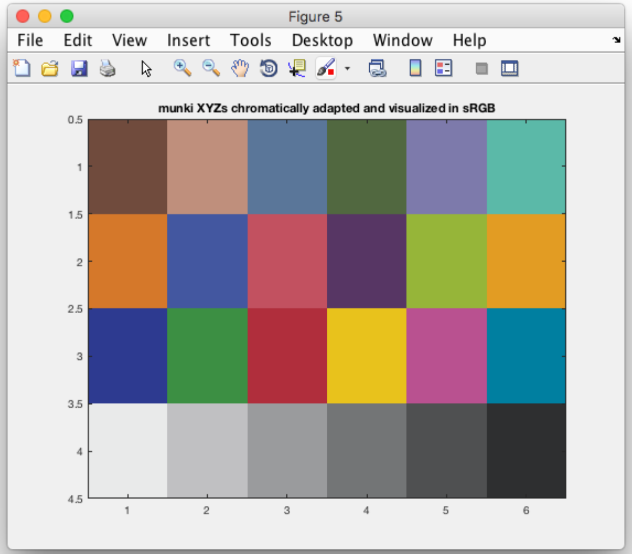
```
% save the (extended) camera model for use in later projects
save('cam_model.mat', 'cam_polys', 'cam_matrix3x11');
```

16) a) Create a function **cam_XYZs = camRGB2XYZ(cam_model, cam_RGBs)** that takes as input your ‘cam_model.mat’ and ‘cam_RGBs’ a 3xn vector of camera-captured $RGBs_{0\text{-}255}$ and returns ‘cam_XYZs’ a 3xn vector of model-estimated XYZs. This function can be developed from parts of the code shown in steps 6),12), and 13). In particular, note that since you are using the extended 3x11 matrix, you will need to do something akin to step 12a) to create an RSrgbs_extd dataset when using the matrix to estimate XYZs. b) Test the function by using it to estimate XYZs from the cam_RGBs you extracted from the chart image in step 3). Confirm that the XYZ values are the same as those you calculated in step 13). c) Show the code for the function and the resulting XYZ values in your report.

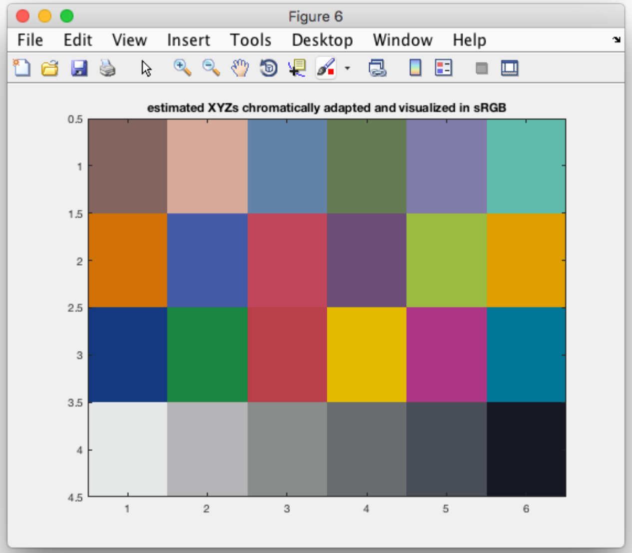
```
% b) test that the camRGB2XYZ function works correctly
cam_XYZs = camRGB2XYZ('cam_model.mat', cam_RGBs)

cam_XYZs =

  Columns 1 through 10

   17.8685   53.2620   21.0601   15.7058   24.6994   32.5390   36.9343   12.9566   29.0224   12.6322
   16.0867   49.3503   22.6861   18.7846   23.2761   44.0611   28.1189   11.8791   18.0748   10.2518
   10.2173   30.6058   32.1697    8.9698   32.9292   36.7513    2.8619   30.6696    9.5384   15.2131

  Columns 11 through 20

   36.8726   48.1131    5.2696   11.1980   25.7906   56.8326   24.3970   11.1669   83.4463   47.7134
   46.3709   43.6933    4.7135   18.8110   15.8205   56.5128   14.1945   15.6650   86.9588   49.2909
    9.6988    3.5328   16.5581    6.7608    6.3932    5.9255   18.9884   25.7880   71.9688   41.3940

  Columns 21 through 24

   26.5380   15.1674    7.5146    0.9789
   27.8556   15.9836    7.8676    0.9571
   22.7802   14.1024    7.9686    1.3838
```

17) To visualize if your camera model and and camRGB2XYZ function are working correctly, create images like those shown below, that visualize the ColorChecker patches from the ColorMunki-measured and camRGB2XYZ-estimated XYZ values. Note that you will need to use the chromatic-adaptation function **catBradford** to convert the XYZs (that are calculated relative to a D50 illuminant) to the nominally-D65 whitepoint of your display, and then use the function **XYZ2sRGB** to produce the RGB values. Sample MATLAB code and outputs are shown below.

```
% visualize the munki-measured XYZs as an sRGB image
munki_XYZs_D65 = catBradford(munki_XYZs, XYZ_D50, XYZ_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');
```




```
% visualize the camera-estimated XYZs as an sRGB image
cam_XYZs_D65 = catBradford(cam_XYZs, XYZ_D50, XYZ_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');
```



18) a) Use the “publish” menu/function in Matlab to document all the code, results, and figures you generated in steps 3-13. b) Include your names and team number at the beginning of the report. c) Include a feedback section at the end of the report that briefly discusses i) who did what parts of the project, ii) any problems you had with the project, iii) any parts of the project you thought were valuable, and iv) any improvements you’d like to see. d) Submit this document as a single PDF named “teamX_project5_report.pdf” to the dropbox on myCourses.