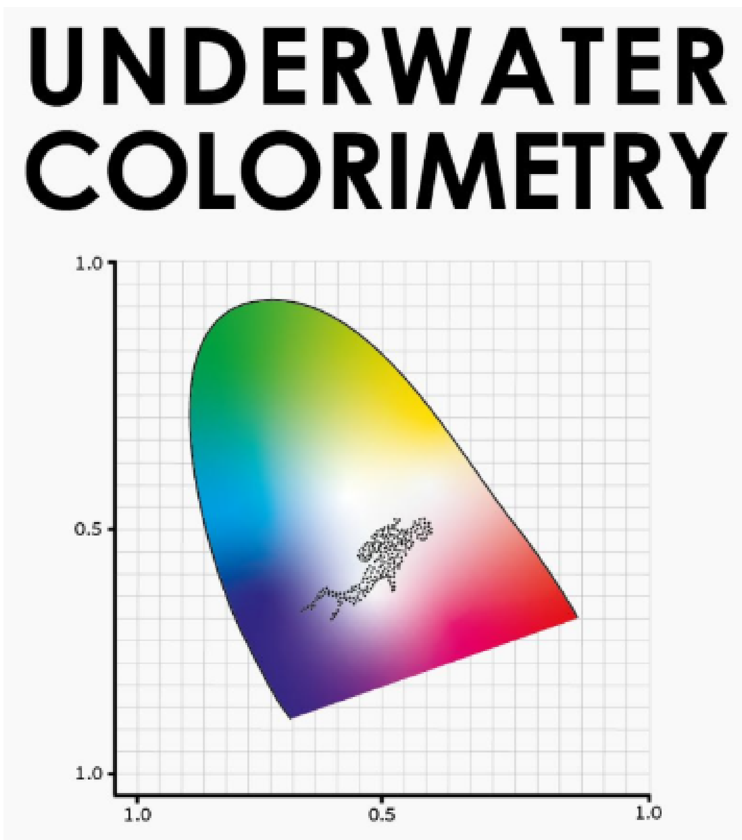
Underwater Colorimetry
Lab 1

IUI

January 2025



THE INTERUNIVERSITY
INSTITUTE
FOR MARINE SCIENCES
IN EILAT

RAW image manipulation & Basic image formation

Session objectives:

1. Hands-on experience with image formation.
2. Linearity check of camera sensors.
3. Working with different image formats based on the required application.

Required equipment:

1. Computer
2. MATLAB or Python
3. RAW images

Provided data for the lab:

Underwater Colorimetry GitHub Repository

Download the repository as a .zip file. It is very important to place the repository in a folder whose path **does not** contain any spaces or special characters!
The required data can be found in the folder *Lab1/Data*.

Provided for the lab	**Comment**
`MacbethColorCheckerReflectances.csv`	Reflectances of all patches of a Macbeth ColorChecker: The 1-24 corresponds to the patches in the numbering order given here.
`illuminant-D65.csv,illuminant-A.csv`	Spectral power distributions of two CIE standard illuminants: daylight: D65, incandescent: A.
`NikonD90.csv, Canon_1Ds_Mk_II.csv`	Spectral responses of two cameras, Nikon D90 and Canon 1Ds II.
`NikonImage.NEF`	An image containing a Macbeth ColorChecker acquired with a Nikon D90 in RAW format.
`CanonImage.CR2`	An image containing a Macbeth ColorChecker acquired with a Canon 600D in RAW format.

Table 1: Provided files and their descriptions.

Extract the zip of the repository to the desktop. When done, save your work in L drive. Not doing so will cause you to lose all your work overnight!!!

Lab Report Due

Sunday 26.1.25 at 9:00 am, by email

Submit to: **uwcolorimetry@gmail.com**

Your email title should include the lab number, your name and affiliation!!!

For example: Lab 1 - James Bond - University of Integers

Lab report: Maximum 3 Pages!

Please keep your reports clean and professional

Lab Overview

Exercise 1: *Converting RAW images*

Exercise 2: *Check the linearity of a sensor from a photo containing a color chart*

Exercise 3: *Simulate taking a "photo" of a Macbeth ColorChecker in air under a given illuminant with two different cameras*

Exercise 1

Manipulating RAW images

Use a camera that can capture an image in RAW format to take a picture of a Macbeth ColorChecker in daylight. Some smartphones might be able to do this. If your smartphone cannot produce RAW images, use one of the provided GoPros. While taking the images, make sure not to cast shadows of your hand or camera onto the color chart.

Important Message

DON'T delete your images and make sure to get both RAW and JPG images!

We will use them again in other Labs.

Fun fact:
An issue that can arise when working with color images is the loss of linearity due to the codec. Unfortunately, the most widely spread image codec .jpg is not linear. Together, we will utilize a custom image pipeline to obtain linear .tiff and .png files from proprietary RAW images. You will need to make use of this workflow when you are getting your files ready for exercise 2 of this lab and later sessions.

Steps

1. Go to *Lab1/Images* and read the **README** file for instructions about where to save your images.

2. Download your RAW image(s) to your computer and place them in the designated folder according to the **README** file.

3. Download Adobe DNG Converter for your operating system from HERE. (https://helpx.adobe.com/camera-raw/using/adobe-dng-converter.html).

   Note:
   Adobe DNG Converter is already installed on the computers in the computer lab!

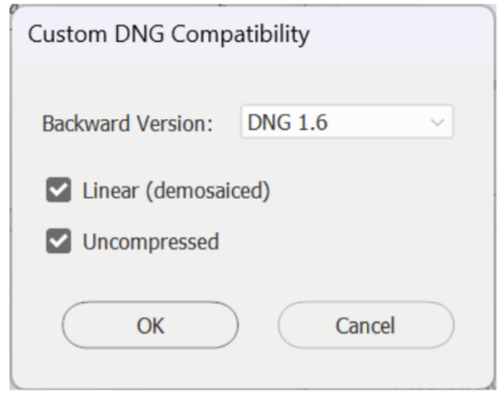
4. Double click and open the program. In section 1, click the "Select Folder" button and point it to the "raw" folder.

5. In Section 2, click "Select Folder", and choose the "dng" folder.

6. No need to do anything in Section 3.

7. In section 4, click the "Change Preferences" button. In the next screen that opens up, click the "Compatibility" drop down list. Select "Custom" and check both "linear" and "uncompressed".

   Set JPEG preview to Medium Size.



Leave "Lossy Compression" box UNchecked.

Leave Embed Original Raw File UNchecked.

Click OK to close the second window.

8. Click "Convert" to start raw to dng conversion! The dng file(s) should appear in the output folder you specified.

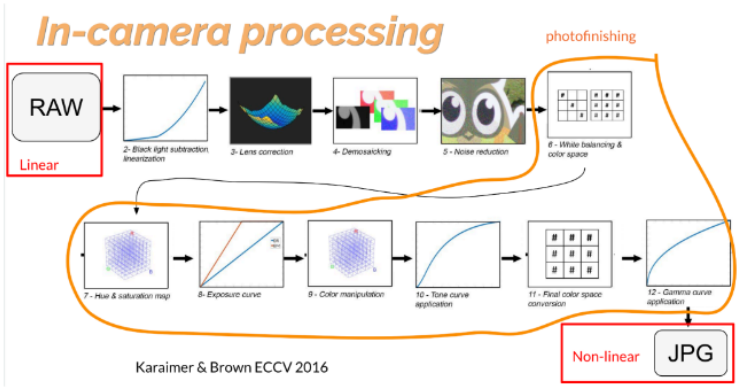
9. Now we will convert DNG images to .tiff files.

10. Open Matlab. We will run the:

    ```
    Exercise1.m
    ```

    script to create the tiff and png versions of the dng images you made with Adobe DNG converter in the previous step. This function relies on other functions in the folder named: "camera-pipeline-nonUI-master" in the repository. Make sure the repository, and all subfolders are on your path (use addpath, genpath functions if needed).

11. The script will also run the tiff2png function to create smaller and compressed files (png uses lossless compression, unlike jpg), because our lab computers cannot handle hundreds of full-size tiff files. In addition to converting a .tiff image to .png, this script will resize the images to half their original size. This may not be necessary, depends on which computer you are using.



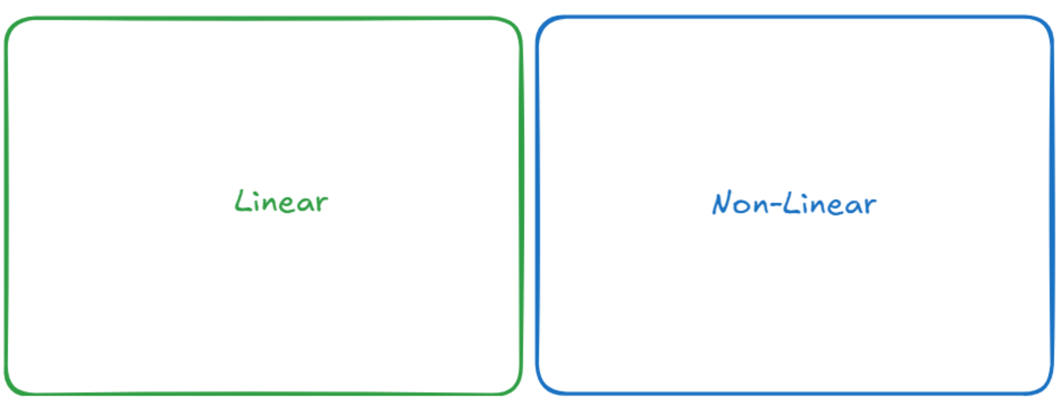
RAW images are naturally very dark (they do not contain any non-linearities that enhance their brightness), so to visualize what is in the scene, you may multiply an image with 2 or 3 to brighten it when you display it (don't save it with brightened values)!

To display the images use:

```
I = imread('Image_Location.tif')
imshow(2*I)
```

Include in your report - Exercise 1

1. A "brightened", side by side comparison of the non linear and linear images. The GoPro will automatically create .jpg images which are not linear, use them in this plot.
2. Discuss whether there are any visible differences.



Exercise 2

Check the linearity of a sensor from a photo containing a color chart

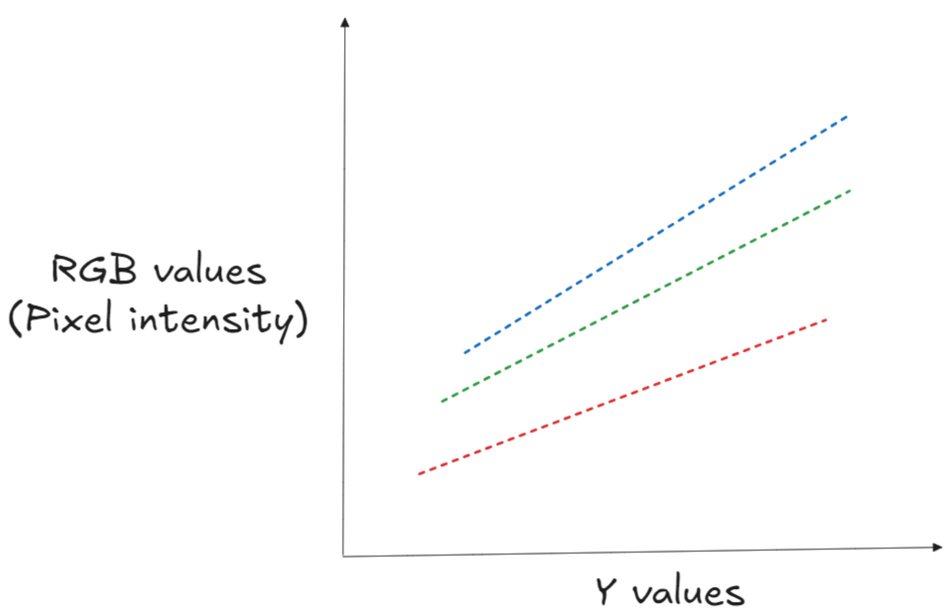
As we learned in the lecture today, it is important to know certain characteristics of your camera before using it to collect scientific data. One of the most important properties is a sensor that is linear with radiance.

But how do you check for linearity of your camera's sensor?

1. Convert a RAW image containing a color chart to a linear .png.
2. Extract average RGB values of the achromatic patches (grays).
3. Plot these values against the Y value of each achromatic patch (you can calculate that yourself or use published values).
4. Check if the extract RGB's of the gray patches are linear with Y.

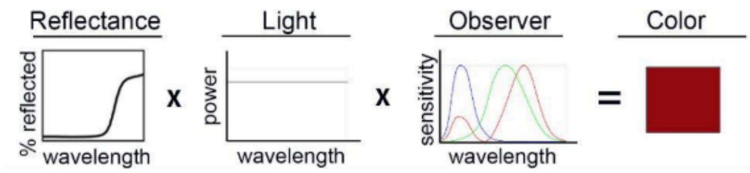
Include in your report

1. Plot of Y-RGB, checking for linearity for your RAW image.
2. Repeat the plot of Y-RGB but with your JPG image.
3. Discuss what the data show.
4. Are the responses of all images linear? Why? Why not?
5. How are the images different? What do you see?
6. Explain each step of the code and how you got the linearity plot, it is recommended to run the code cell by cell.



Exercise 3

Simulate taking a "photo" of a Macbeth ColorChecker in air under a given illuminant with two different cameras



The goal is to see how the components of the image formation model influence the appearance of the image and how white balancing works and what it does.

You will learn how to visualize the difference in color data in chromaticity diagrams to answer the following questions:

- Do two images of the same scene taken by different cameras look the same?
- How can we compare color images taken with two different cameras?

	Nikon	Cannon	GoPro
Illuminant A	R L S'	R L S'	R L S'
Illuminant D65	R L S'	R L S'	R L S'

Include in your report - Exercise 3

Steps 1 - 3

1. 4 white-balanced figures:
   - 2 Nikon, 2 Cannon

   Each camera under 2 different lighting conditions (A & D65).
2. 4 **not** white-balanced figures:
   - 2 Nikon, 2 Cannon

   Each camera under 2 different lighting conditions (A & D65).

Show the color charts side-by-side, as "taken" by the 2 cameras for each illuminant.

Discuss **(1 paragraph maximum!)**

Are there differences between the resulting charts? Why?
Explain how the code works, cell by cell.

Guidance for writing the code

The full code for this exercise is available on the course GitHub. If you feel comfortable enough, try to follow this guidance and write your own code.

Step 1: Simulate a Macbeth ColorChecker

- Get the necessary data from the GitHub repository:
  - Reflectances of Macbeth ColorChecker.
  - Cameras spectral sensitivities (Nikon, Cannon and GoPro).
  - Light data (illuminant D65 and illuminant A).
- Make sure that all the imported data is on the same wavelength range, if needed use the MATLAB function interp1 and Interpolate to a common range (400:10:700) for calculations.

```
%   Vq = INTERP1(X,V,Xq)
interpolates to find Vq,
the values of the underlying
function V=F(X) at the query
points Xq.
```

- **Calculate radiance** for the ColorChecker for a given illuminant and a given camera. It is recommended to write your own dedicated function to calculate radiance.
  Further details:

```
radiance = getradiance(reflectance,light,observer)

% This function calculates radiance
from given reflectance, light, and
observer spectra.
```

As showed in the lecture, calculating radiance can be expressed as a simple matrices multiplication.

$$Reflectance = \rho \in R^{(m\times n)}$$

$$Light = L \in R^{(1\times n)}$$

$$Observer = S_c \in R^{(3\times n)}$$

Where:

- m is the number of different reflectances.
- $m = 24$ for a MacbethColorChecker.
- N is the number of wavelength steps.

To take advantage of the power in matrices multiplication we will need to check our dimensions.

$$Radiance = \rho \cdot diag(L) \cdot S_c'$$

And dimension wise:

$$(24 \times 3) = (24 \times 31) \cdot (31 \times 31) \cdot (31 \times 3)$$

- **Visualize Colors**
  Write the function:

```
function mcc = visualizeColorChecker(RGB)

% RGB is expected to be a matrix of 24 x 3.
% The input colors are expected to be in the
right (typical) order for a Macbeth ColorChecker.
```

The visualizeColorChecker function takes a 24×3 matrix of RGB values representing the Macbeth ColorChecker.

- **visualizeColorChecker(RGB) Logic:**
  1. A for Loop (for i = 1:24) iterates over each row of the RGB matrix, where RGB(i,:) represents the RGB values of the i-th color patch.
  2. For each color patch, it calls the function visualizeColor with the RGB values and a size of 100 pixels (a default size for visualization).
  3. The result is stored in cell array called *imgs*.
  4. After generating the image for each color patch, use:

     ```
     imtile(imgs, 'GridSize', [4 6])
     ```

     The *imtile()* function will arrange the images in a grid of size 4×6 (which corresponds to the typical layout of a Macbeth ColorChecker with 24 color patches).
  5. **Output**:
     mcc: The combined image representing the entire Macbeth ColorChecker.

For each of the 24 colors, you will need to cal, and write, a second function called **visualizeColor** to generate a color patch.

```
function testPatch = visualizeColor(RGB,M)

% The size of the square image patch to create.
% Set M = 100.
```

These patches are arranged into a 4×6 grid to visually mimic the Macbeth ColorChecker.

– **visualizeColor(RGB,M) Logic**

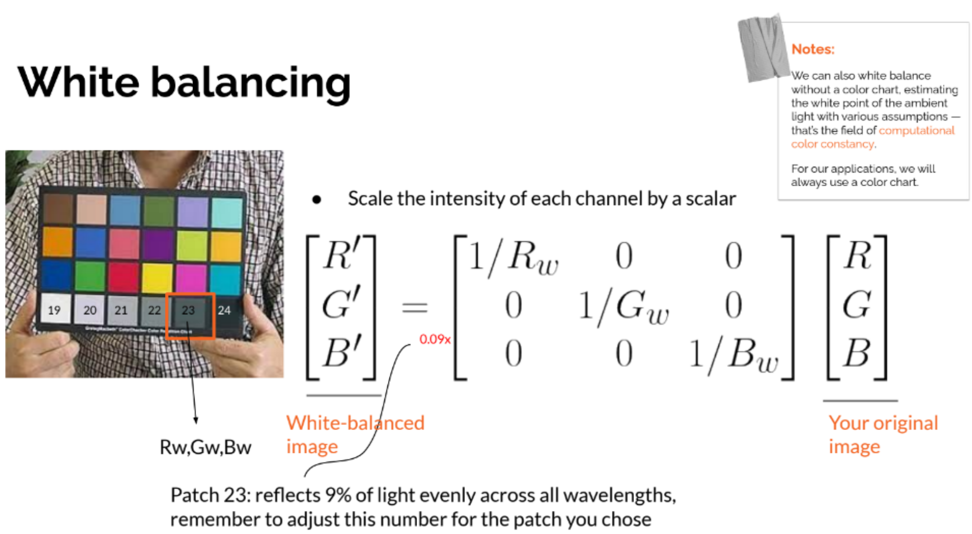
1. **Input:**
   (a) **RGB**: A 1×3 vector representing the RGB values for a single color.
   (b) **M = 100**
2. **Initialization:**
   A M×M×3 matrix (representing an RGB image) is initialized to zero, where each color channel (red, green, blue) is represented by a separate slice along the third dimension.
   ```
   testPatch = zeros(M,M,3)
   ```
3. **Looping over color channels:**
   ```
   for j = 1:3
   ```
   (a) The loop fills the color channels (red, green, blue) of the matrix *testPatch* using the values from the RGB input vector.
   (b) The RGB values from the input are assigned to the respective color channels across the entire M×M patch.

4. **Output:**
   testPatch: A square patch of size M×M, filled with the specified RGB color.

```
function testPatch = visualizeColor(RGB,M)

if nargin<2
    M = 500;
end

testPatch = zeros(M,M,3);
rows = 1;
cols = 1;

for j = 1:3
    testPatch(1 + (rows-1)*M:M*rows,1+(cols-1)*M: ...
    M*cols,j) = RGB(j);
end
```

Step 2: White-balance

- Choose a gray patch (23rd) for white balancing (9% reflectance).
- Divide radiance values by the selected patch value.



Step 3: Visualize White-Balanced Colors

- Use the same visualization function.
- Save the image as:

```
Macbeth_wb.png
```

Lab Report Due

Sunday 26.1.25 at 9:00 am, by email

Submit to: **uwcolorimetry@gmail.com**

Your email title should include the lab number, your name and affiliation!!!

For example: Lab 1 - James Bond - University of Integers

Lab report: Maximum 3 Pages!

Please keep your reports clean and professional

Lab Overview

Exercise 1: *Converting a RAW image to a linear image*

Exercise 2: *Check the linearity of a sensor from a photo containing a color chart*

Exercise 3: *Simulate taking a "photo" of a Macbeth ColorChecker in air under a given illuminant with two different cameras*