Test: Write C++ and Python code using openCV v.3.0+ to calculate dose from the film to be run on Ubuntu (15.10) linux.

Given:
2 TIFF images,featuring 4 rectangular film strips where one image is the pre-irradiation (film_bg.tif) and the other film is post-radiaiton (film.tif).

- There is no hard time limit but who finishes first will have a priority interview date. Recommended time is 10 days total.
- Note: the exposed film strips as seen on the TIFF image are not quite aligned and strips 3 and 4 are in the wrong order. You cannot use any external software to cut the strips out or crop the image.
- You must use openCV 3+ for all your image read/write, and arithmetic applications (log, divide, multiply, etc).
- For python you can use any modules in addition to openCV modules.
- All tiff images are 3x16 bits, where each color is 16 bit. You will use only Red 16-bit channel for color to dose conversion.
- Do not use green or blue channels.
- For python use Python 2.7 .
- Your result is a 16-bit TIFF image for each strip.
- Your dose values should be in the range from 0 to 2500!
- To test that your tiff dose images are correct – use ImageJ (free app) to open your tiffs and check dose values using X and Y profiles (select line, Ctrl-K). I am using ImageJ to check your images too.
- The code should compile and run on Linux (I use Ubuntu 15.10).

You will email to me the following:

- a 16-bit tiff dose.tif file with correct doses,
- your C++ and python codes,
- your instructions for me how to compile and run your code with g++ to get dose.tif. This include a complete console line to compile and link necessary libraries.
- To run the code, the command line should look like:

  $ ./convert film_bg.tif film.tiff
  and for python:
  $ python convert.py film_bg.tif film.tif
- the result of the above command line is dose.tif image with 16-bit dose values.

Dose conversion formula:

Convert film Red channel intensities to dose using the following equation:

$$D\ (mGy)\ [i,j] = k * 10000 * \log10\ (\ Ro[i,j]/R[i,j]\ )$$

where Ro[i,j] is the red background value of each pixel, and R[i,j] is the corresponding red post radiation pixel value.
k is a conversion factor that is calculated by spline interpolation from the X vs D calibration map given below.
log10 is a base 10 log.

X = {0, 187, 300, 540, 1030, 1810, 2775, 3790, 4800, 5470, 6000, 7200 };
D = {0, 60, 120, 221, 515, 1054, 2042, 3242, 5000 , 6323, 8000, 15000};

Recommended steps to code in C++ and python:

1. read the background and the exposed images;
2. for each strip find their corresponding pixels;
3. Calculate Dose by applying the calibration map and using spline interpolation for any X values between those given by the map.
4. Before dividing Ro/R convert Ro and R to CV_32F.
5. then take a log10 of Ro/R.
6. Then scale up by 10000 and convert this log10 back to CV_16UC1, so X = 10000 * log10 (Ro[i,j]/R[i,j])
7. This scaled value X will be used in interpolation.
8. You will get a range of X values for your pixels. This range should be in the range given in the interpolation curve. If you get an X outside of the range given, then you are doing something wrong.
9. perform interpolation for all pixels and strips in the right order and alignment to get a 16-bit dose D[i,j]
10. write a 16 bit dose.tif file which will have the dose values (16 bit) in place of the red values in the film.tif image.

For some this assignment may look too simple and boring, in this case:

If you want to make it more interesting, you can convert the final 16-bit dose images to an 8-bit color image (png), but you have to do a proper type and scale conversion, so you don't loose information. This can also be done using openCV functions and some C++ casting. And save the image using the jet color scheme, but scale it such that it does not create “rings” of the same color. Each color shows only once, as shown in a picture below.