Task 1: Skin Detection Part 1 (practical)

Face detection is also a search for skin patches in an image to identify areas (and sclaes) to look for faces. In this exercise, we extract skin color and try to find a suitable color sub-space that allows us to easily classify skin pixels. We are not yet learning the classfier itself but simply perform data preparation steps.

Use a standard framework for image manipulation:

- **OpenCV** (https://opencv.org) is an advanced computer vision library original written for C/C++. But there are also bindings for Python, Java, and other languages.
- **scikit-image** (http://scikit-image.org) is an advanced computer vision library written in Python. It provides all basic image manipulation operations as well as advanced feature extraction algorithms (however, not SIFT but alternative approaches to SIFT)
- **Java AWT and JAI** (see http://www.oracle.com/technetwork/java/javase/tech/index.html) provide basic and advanced image processing methods.
- …many other libraries, so pick your preferred option.

a) **Know your library:** Understand how color space transformations work. Identify the RGB space that is used (sRGB, linear RGB, RGB CIE, …) and verify some of the color mapping functions (it is sufficient to check the mapping to X,Y,Z). Any difference to the course material? Also check other sources to validate the correct implementations. Summarize shortly your findings and provide a short example code for color transformations. What is the default color space if you load a JPEG/PNG/TIFF image?

b) **Create a data set:** Download images with faces and cut out (with an image tool) areas which only show skin (no eyes, no beard, no hair, no glasses, no hats or veils). Ensure that you have samples with different lightning (indoor, outdoor, warm/cold light) and complexion. Also produce a few negative samples with no skin at all (but may come close to skin like sand). It is sufficient to have around 1M pixels for the positive and negative samples.

c) **Data preparation:** Before we can learn a classifier, we need to find the best input data that allows us to separate positive from negative samples. To this end, write some code that maps all pixels into a two dimensional color subspace and output the frequencies for the positive and the negative samples. Visualize this output with a 2-dimensional scatter plot (Excel is just fine for that; use one color for positive samples and another color for negative samples). Find a color subspace that (visually) best separates positive from negative samples.