Lab 2 - Energy-Efficient Image Processing

1. Objective

The lab demonstrates how image manipulation can be used to tradeoff image quality to save power in emissive displays.

Day 1	Estimate power consumption in OLED displays
	Estimate distortion between two images
	Experiment the impact of simple transformations on power and distortion
Day 2	Learn how to interface with an embedded OLED display
	Test the transformations from day 1 on the provided display

2. Day 1

Goal: learn to manipulate images in MATLAB and study the effect of image manipulations on power consumption and image distortion.

a) Identification of images

Test images for the evaluation will be:

- 10 (color) images taken from the USC SIPI Database.
  http://sipi.usc.edu/database/database.php?volume=misc
- 5 images representing different screenshots of your computer (use images that differ significantly!)

b) Manipulation of images

For this step select one test image to experiment the flow. Basic steps include:

- Import the image (check `imread()` function for supported formats)
- Extract R, G, B channels
- Convert between different color spaces

NOTE: Please refer http://www.mathworks.it/help/toolbox/images/

c) Evaluation of power consumption

Define a MATLAB function that estimates the power of an image. Power of a single pixel for a specific OLED display is given by the following equation:

$$P(R,G,B) = w_R * R^\gamma + w_G * G^\gamma + w_B * B^\gamma$$

Where:

- $w_R = 2.13636845*10^{-7}$
- $w_G = 1.77746705*10^{-7}$
- $w_B = 2.14348309*10^{-7}$

- $\gamma = 0.7755$

and R,G,B are the intensities of the color channels (between 0 and 255). The power of the entire image is obtained by summing the power value of each pixel plus the baseline power $w_0 = 1.48169521*10^{-6}$:

$$\mathbf{P_{image} = w_0 + \Sigma_{i=1...N} Pi(R,G,B) \quad [W]}$$

Where N is the number of pixels of the image.

d) Evaluation of image distortion

Define a MATLAB function that evaluates the distortion w.r.t. the original image. Use two different metrics of distortion:

1) The sum (over all pixels) of the Euclidean distance between corresponding pixels in the in L*a*b* space, that is:

$$\mathbf{\varepsilon(image_i, image_j) = \Sigma_{k=1...N} (sqrt((L_{i,k}-L_{j,k})^2+(a_{i,k}-a_{j,k})^2+(b_{i,k}-b_{j,k})^2)}$$

Where subscript i and j refer to the two images and k denotes the k-th pixel of the image. To convert RGB into Lab space (and vice versa) use the built-in MATLAB functions `rgb2lab()` and `rgb2lab()`).

2) The Mean Structural Similarity Index (MSSIM), a complex metric considering brightness, contrast and structure differences. Use the built-in MATLAB function `ssim()` to compute it.

Compare the two metrics and analyze which one correlates better with visual similarities.

e) Assignment: Evaluation of various strategies on image modification

Experiment with the following image manipulation strategies and analyze their impact on power consumption and distortion using the original image as reference. Basic set of techniques to demonstrate:

- Simple pixel-wise transformations (e.g., reduce the hungry blue component by subtracting some fixed value to the blue channel)
- Histogram equalization.
- Other brightness/contrast modifications (use your imagination)

Find the best possible image manipulation method to minimize the power consumption while limiting the average image distortion within the given constraints (1%, 5%, 10%), as follows:

1) When using the distance in L*a*b space as a distortion metric, the relative distortion in percentage between two images is computed as:

$$\mathbf{\varepsilon (image_{original}, image_{result}) / \varepsilon (white, black) * 100 (\%)}$$

2) When using the MSSIM as a distortion metric, the relative distortion in percentage between two images is simply:

MSSIM($image_{original}$, $image_{result}$) * 100 (%)

Notice that histogram equalization is not a pixel-based transformation. Moreover, histogram equalization and other brightness/contrast modifications require conversion of RGB into another color space such as HSV (see material on the web).
Please apply the implemented methods on the entire image set by organizing everything into a script so that you can automatically test and evaluate all images at once.

3. Day 2

Goal: apply the transformations studied during Day 1 to a real OLED display to assess the real impact on visual quality. Learn how to interface with an embedded OLED display.

Provided Hardware:

- Arduino Uno Board
- 128x128 New Haven Display NHD-1.5-128128ASC3

f) Connect the Arduino and the OLED using the provided cables

Function	Arduino Pin	OLED Pin
SCK	D13	SCK
MOSI	D11	MOSI
NRST	D6	/RES
Chip Select	D5	OLEDCS
Data/Control	D4	D/C
VDD	3.3V	VDD
GND	GND	GND

g) Write MATLAB code to interface with the Arduino Uno

Define a MATLAB script to send an image to the provided Arduino board using RS-232 over USB. First read the image and convert it to the appropriate format:

a. 18-bit RGB (6-bit per channel) instead of the standard 24-bit (8-bit per channel)

b. B-G-R color order
c. Width = 128 pixels, Height = 128 pixels

To reduce the number of bits per channel use truncation or (better) rounding. To resize the image use the `imresize()` function provided by MATLAB.
Having done this, send the pixels to the Arduino using the Serial I/O functions provided by MATLAB (`serial()`, `fopen()`, `frwrite()`, `flushoutput()`, `fclose()`).

Hint: Use a special code to signal to the Arduino the "start" of a new image, to improve stability. For example, you may use the 2 most significant bits of each byte, which are not used in the transmission of color information (since each channel only uses 6-bit).

Use a baud rate not larger than 115200 to avoid data losses. Remember that opening a serial connection resets the Arduino board, so add some wait statements to let the board finish its setup before starting to send pixels.

h) Modify the provided Arduino code to receive and forward image pixels

Modify the provided Arduino code to support receiving pixel data on the RS-232 connection and forwarding them to the OLED display using SPI (over GPIO).

- **Provided Arduino code:** all basic OLED control functionalities (initialize display, setup, send pixel, etc.)
- **Missing Arduino Code:** high-level code using the basic functionalities to receive an entire image from MATLAB and forward it to the OLED.

1. Modify the Arduino `setup()` function to perform the following operations.
   - Setup the pins used by the Arduino/OLED interface, configure the RS-232 and SPI interfaces (provided function: `OLED_setup()`)
   - Initialize the OLED display with default settings (provided function: `OLED_init()`)

2. Write code to handle pixels coming from MATLAB over RS-232. **Hint:** You may need to use/implement the following standard Arduino functions:
   - `Serial.read()`: read one char from the RS-232 channel (cast output to uint8_t).
   - `Serial.available()`: check if there are bytes available for reading in the RS-232 channel
   - `serialEvent()`: the default Arduino Interrupt Service Routine to handle RS-232 events (received bytes)

   And the following functions from the provided OLED library:
   - `OLED_Data_128128RGB()`: write the next color component to the screen and possibly update the pixel address for the next pixel.
   - `OLED_SetRowAddress_128128RGB()` and `OLED_SetColumnAddress_128128RGB()`: set the starting address before starting to send a new image

- `OLED_WriteMemoryStart_128128RGB()`: enable writing to the OLED's internal memory

Use the Arduino documentation and the OLED datasheet as references for writing the code.

i) Assess the visual impact of image transformations on the provided OLED

Repeat the same image transformations used during Day 1 and check their impact on the provided OLED. Analyze possible differences in terms of power saving and image distortion, due to the different color representation and image size.

First, implement the transformations in MATLAB and then send the modified image to the OLED display. Then, try to implement the same transformations directly in the Arduino code. What are the limitations? (**Hint:** the Arduino memory cannot store the entire 128x128 image).

4. Organization, Deadlines, and Evaluation

This lab accounts for 2 points overall. Due to the limited availability of boards, we suggest that students get organized into **one- or two-person groups** (no 3-person group). Each group will have to deliver a short (4-5 pages plus plots/tables etc) document including:

- The outcome of the experimentation of points a) to d)
- The description of the results relative to point e) and i)

Each group should also attach all the **modified code** useful to reproduce their results.