Politecnico di Torino
III Facoltà di Ingegneria

Energy management of IoT application
Report laboratory session 2

Master degree in Computer Engineering

Authors: group 14

Kristina Rogacheva, Giuseppe Carrubba

February 21, 2022

Contents

1 Introduction **1**

2 Energy efficient image processing **2**

2.1 Image manipulation 2

2.2 Power consumption 2

2.2.1 Hungry-blue 2

2.2.2 Histogram Equalization 4

2.2.3 Custom function 5

2.3 Distortion 6

2.4 Evaluation of power consumption-distortion trade off 7

2.5 MATLAB scripts day1 10

3 Dynamic Voltage Scaling for OLEDs **12**

3.1 Dynamic Voltage Scaling 12

3.2 Brightness Compensation 15

3.3 Contrast Enhancement 17

3.4 Contrast Enhancement + Brightness Compensation 20

3.5 Optimal supply voltage 23

3.6 MATLAB scripts day2 28

CHAPTER 1

Introduction

The aim of the second laboratory session is the demonstration how manipulation of an image can be used to tradeoff image quality and power saving in emissive displays, for example, OLED. We examined different methods to manipulate images and evaluated the consequent power consumption. Then we applied the Dynamic Voltage Scaling (DVS) and observed the results.

The activity is divided in two main parts described in this report:

- Manipulation of the images and analysis of the power consumption
- Dynamic Voltage Scaling

CHAPTER 2

Energy efficient image processing

There are different types of displays: they are divided into two main categories - emissive and transmissive displays. During this laboratory experience, we worked on emissive displays like OLED. To test our scripts we used more than 200 photos from different databases that contain pictures with different colors and characteristics: USC SIPI hereinafter referred to as misc, BSDS500 training set hereinafter referred to as BSR, and some screenshots of our computers.

2.1 Image manipulation

Contrarily to the transmissive displays, in the emissive ones, all pixels are independent, and there is no backlight. This means that power consumption depends on the color components of all the pixels. Before performing different manipulations on the various photos, we computed the power consumption of the original pictures: we will use this value to confront it with the power consumption obtained after the different manipulations.

2.2 Power consumption

To estimate the power consumption of an image, firstly, we have to extract its RGB channels. RGB is an additive color space where all colors can be obtained from three colorants: red, green, and blue. After storing the values for the three channels for all the pixels of the image, we have to calculate the power consumption of the single pixel (2.1), then sum all the contributions to achieve the final power consumption (2.2).

$$P_{pixel} = w_R * R^y + w_G * G^y + w_B * B^y \tag{2.1}$$

$$P_{image} = w_0 + \sum_{i=1}^{N} P_{pixel_i}(R, G, B) \tag{2.2}$$

2.2.1 Hungry-blue

In emissive displays the most power-consuming color is blue. This is the reason why the first implemented algorithm to manipulate images is the `hungry-blue`. This algorithm reduces the value of the blue channel of all the pixels by some percent. The reduction percentage is passed to the function as an input. We called this function three times for each image, and we evaluated the results obtained by 1%, 5%, and 10% reduction.

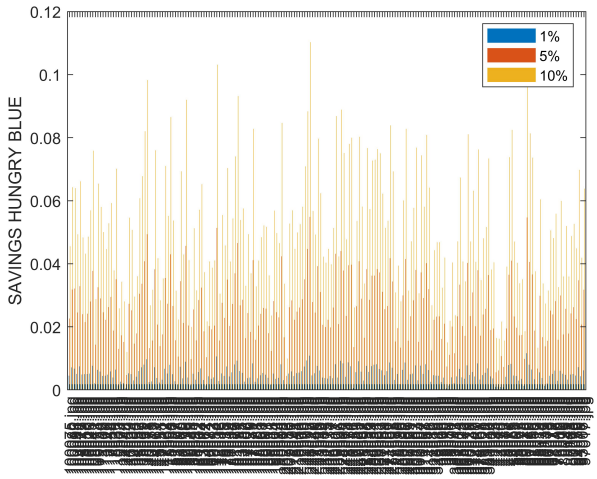

Figure 2.1: Power savings BSR dataset

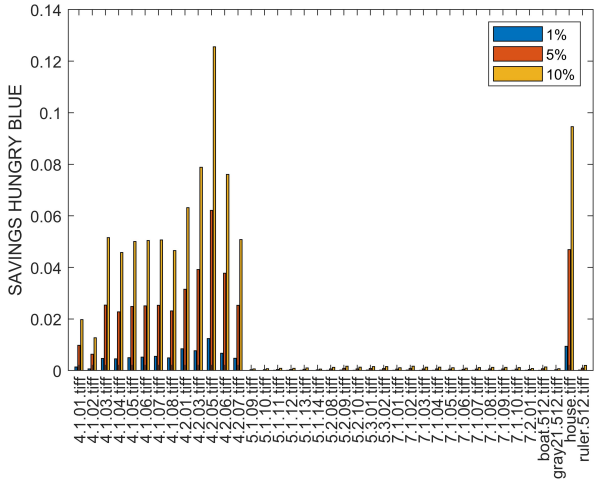

Figure 2.2: Power savings misc dataset

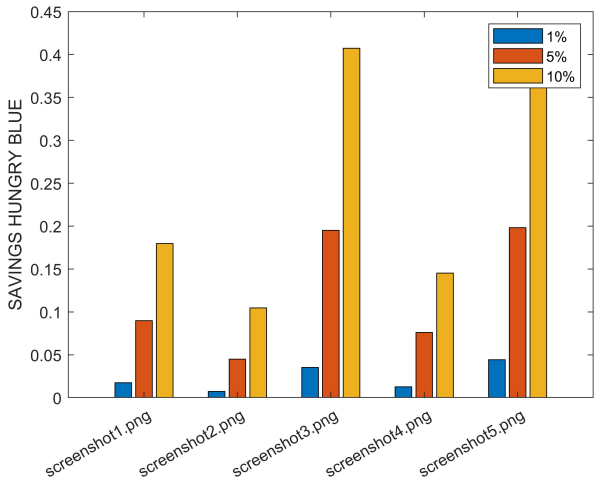

Figure 2.3: Power savings our screenshots

As we can see from the bar graphs 2.1, 2.2 and 2.3 the higher is the reduction of blue, the higher are the power savings. It is important to notice that we can obtain better results for higher levels of blue of the original image. In the bar graph 2.2 for the misc dataset, we can notice several pictures with low power reduction; this is because these pictures are originally black and white and used just one channel instead of 3 like the colored ones.

2.2.2 Histogram Equalization

The second function that we implemented to manipulate the images to improve the power consumption is the `Histogram equalization`. We used the Matlab function `histeq`: this command takes the histogram of the RGB components and flattens them.

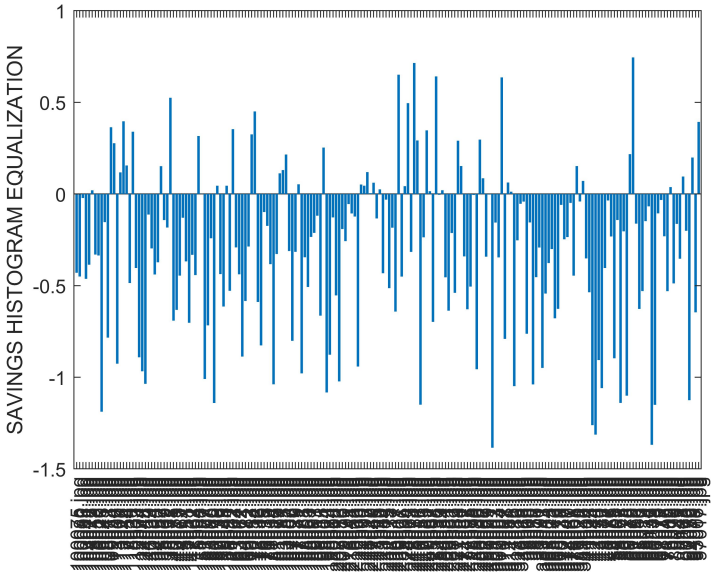

Figure 2.4: Power savings BSR dataset

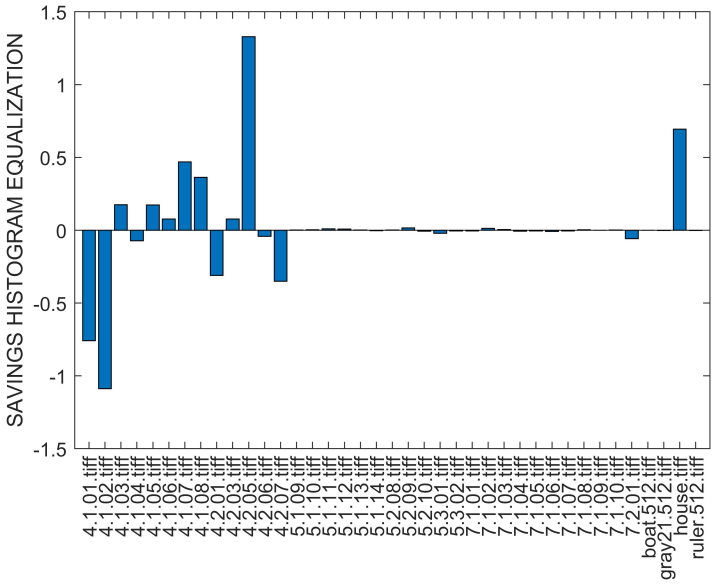

Figure 2.5: Power savings misc dataset

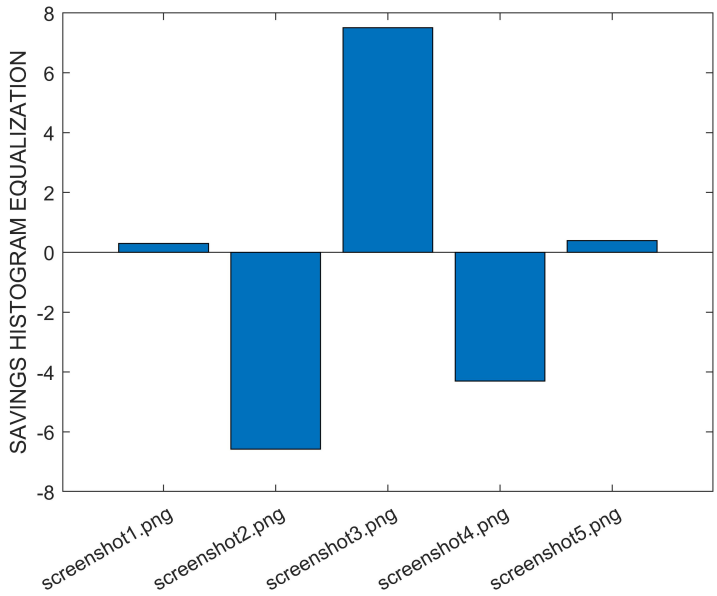

Figure 2.6: Power savings our screenshots

This function improves photos with high luminance, but in average it increases the power consumption of all databases; this is because in all the databases the majority of the images are characterized by low luminance and high contrast.

2.2.3 Custom function

The last function to manipulate the images to improve the power consumption is the `Custom` one. After noticing that the `Hungry-blue` function reduces a lot the power consumption, we decided to extend that function and reduce all the colors by some percent. Like for the Hungry-blue, we performed the manipulation three times for each picture: with 1%, 5%, and 10% of reduction.

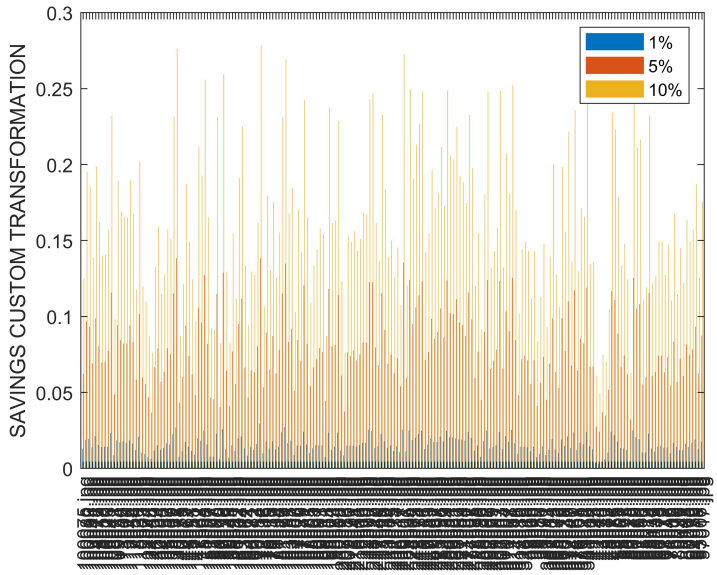

Figure 2.7: Power savings BSR dataset

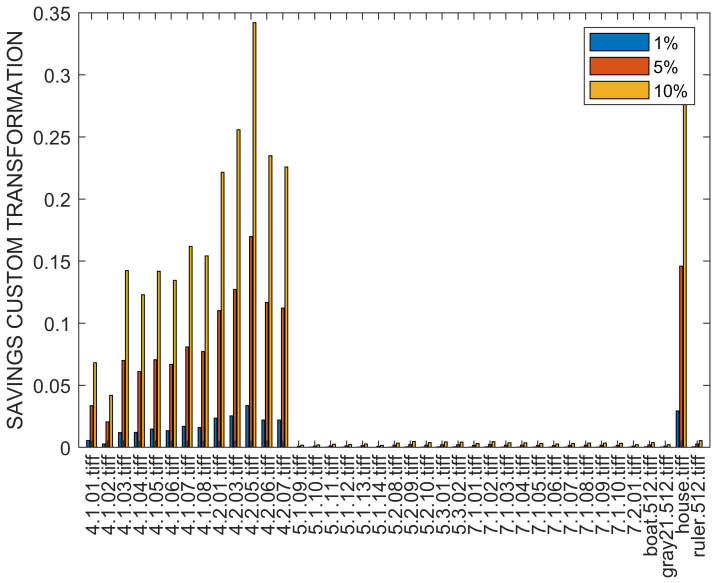

Figure 2.8: Power savings misc dataset

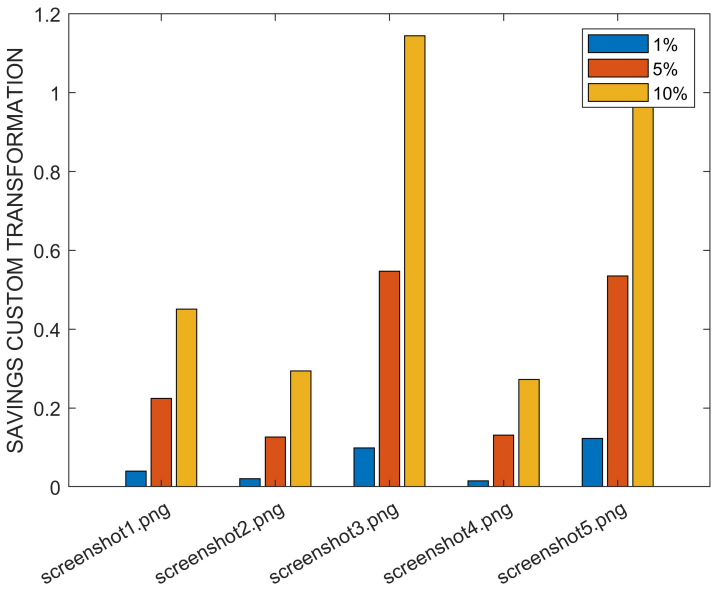

Figure 2.9: Power savings our screenshots

This is the function for which we obtained the best results in terms of power savings, as for the hungry-blue function, the bigger is the reduction, the bigger are the power savings.

2.3 Distortion

All those transformations imply a distortion defined as a measure based on the Euclidean distance between images (2.3). The distortion is different from the perceived visual quality and to compute we have to convert the image to the L*a*b color space. This color space has three channels, one for the Luminance and the other two for colors. The distance between the two images can be defined in 2.4, where `N` is the number of pixels and `k` is the kth pixel.

$$\varepsilon(image_i, image_j) = \sum_{k=1}^{N} \sqrt{(L_{i,k} - L_{j,k})^2 + (a_{i,k} - a_{j,k})^2 + (b_{i,k} - b_{j,k})^2} \tag{2.3}$$

$$dist = \frac{\varepsilon(image_{new}, image_{original})}{W * H * \sqrt{100^2 + 255^2 + 255^2}} * 100 \tag{2.4}$$

2.4 Evaluation of power consumption-distortion trade off

Although from the previous graphs we can notice that with some manipulations we can obtain a good power saving, all the manipulations introduce some distortion. The following plots describe the trade-off between power saving and distortion.

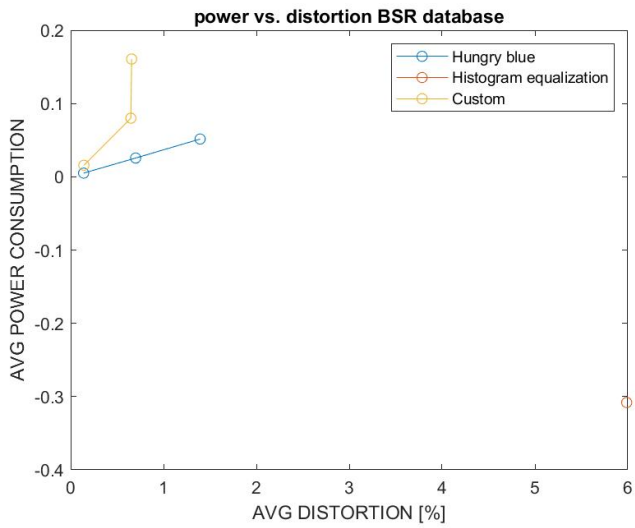

Figure 2.10: Power vs distortion BSR dataset

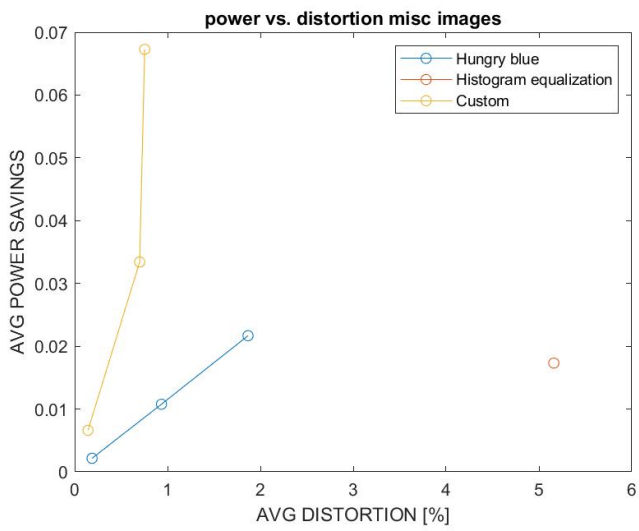

Figure 2.11: Power vs distortion misc dataset

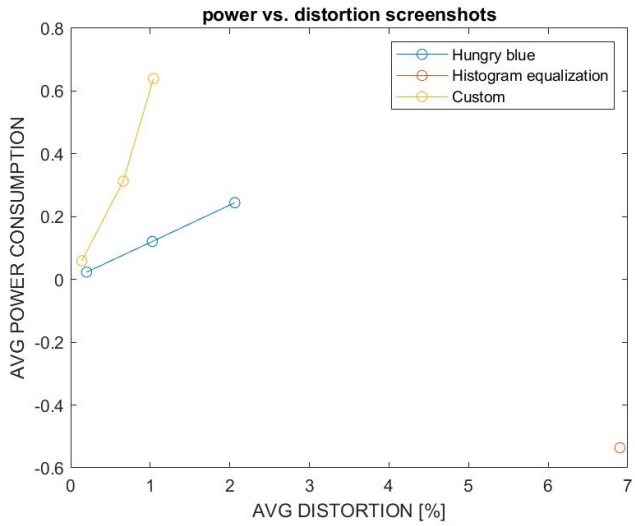

Figure 2.12: Power vs distortion our screenshots

The figures 2.10, 2.11, and 2.16 show the average results obtained by the different transformations. As we can see only Hungry-Blue and the Custom function give us the highest power savings average with respect to the average percentage of distortion. On the other hand, the Histogram Equalization has a completely different result presenting a huge distortion and sometimes negative power savings. This is due to the approach the function assumes to manipulate the images.

Particularly for the dataset MISC, the custom manipulation presents an exponential increase of power savings for nearly no cost in terms of distortion. So we can affirm that the custom manipulation gives the best results if we use the reduction percentage bigger than 5%.

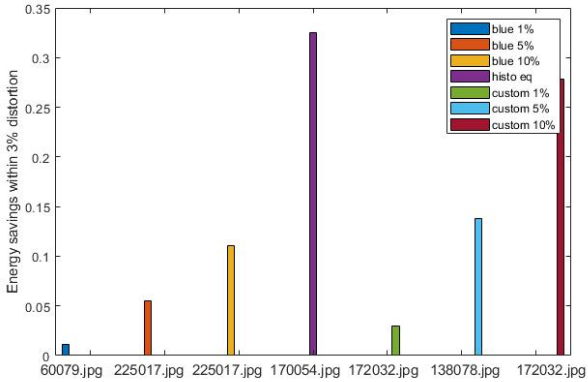

Figure 2.13: Best manipulation BSR dataset

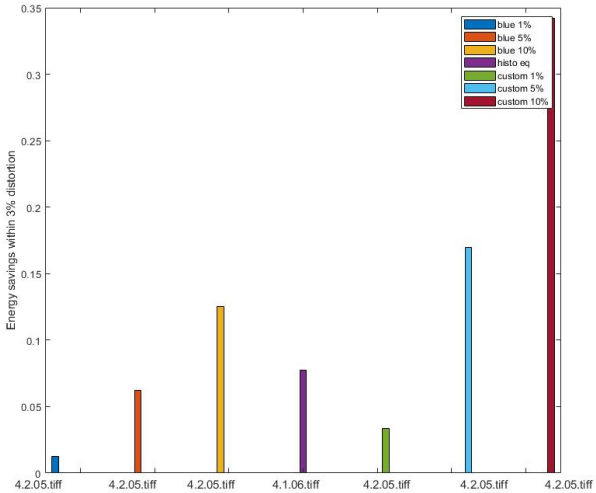

Figure 2.14: Best manipulation misc dataset

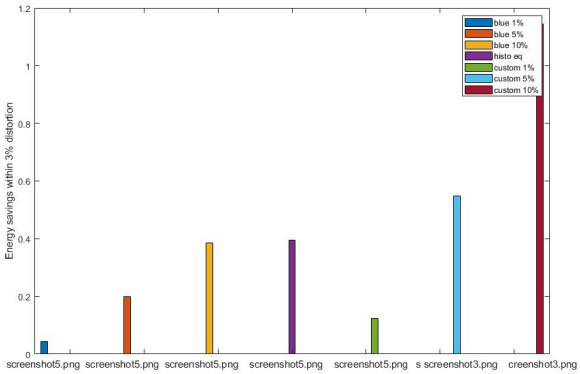

Figure 2.15: Best manipulation our screenshots

The plots 2.13, 2.14, and 2.15 show the photos that gave the best results depending on the chosen manipulation.

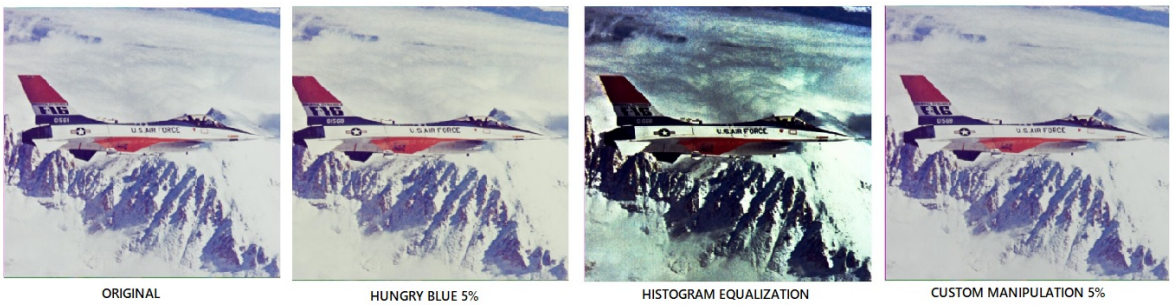

Figure 2.16: Images obtained after manipulations.

Above there is an example of a photo before and after all three manipulations performed separately. It's worth noticing that not only the Custom function gives us better results in terms of the tradeoff between power savings and distortion, but it is also visually the closest to the original picture.

2.5 MATLAB scripts day1

Here, the most important MATLAB scripts and functions developed for our analysis for the day1 are listed and explained shortly.

- **Day1.m**: This is the main script that calls all the functions developed and returns the filled data structures.
- **auto_image_listN.m**: These functions have been developed for the three folders of images. This function creates and fills the data structure associated with the image lists computing several parameters. Inside the function, every manipulation is applied for each image in the list and we produce the figure power savings vs. Distortion.
- **hungry_blue.m**: This function contains the script that applies the hungry blue transformation to an image.
- **histogram_eq.m**: This function contains the script that applies the histogram equalization transformation to an image.
- **custom.m**: This function contains the script that applies the custom transformation to an image.
- **RGB_distribution.m**: This script is capable to produce a histogram that shows the color spectrum of an image.
- **distortion.m**: This script contains a function that is used to compute the distortion of an image after manipulation.
- **image_difference.m**: This script contains a function that calculates the difference between two images.
- **power_consumption.m**: This script contains a function in charge of evaluating the power consumption of an image.

- **gray2rgb.m**: This script has been imported from `"Mathworks Community site"` and it has been used to manipulate black and white images with only one color channel to transform them into RGB images creating 3 color channels. The transformation introduces some margin of error because it approximates the color channels but the approximation is negligible for our analysis.
- **bar_manipulation.m**: Three scripts have been developed with prefix `bar` and as a suffix the name of the manipulation function used. This script produces a histogram graph to compare the power savings obtained by applying transformations to a set of images.
- **best_manipulation_under_constraint.m**: This script searches, for each manipulation, the best images in terms of power savings that satisfy also the distortion constraints. For each list of images, we plot a bar graph to better understand the results obtained.

CHAPTER 3

Dynamic Voltage Scaling for OLEDs

During the first part of the laboratory experience, we studied how changes the power consumption of OLED screens with respect to the displayed colors. Now we will analyze the power consumption that depends on the input current because until now supply voltage was set to maximum to support the full luminance of pixels.

3.1 Dynamic Voltage Scaling

The function `displayed_image()` scales the supply voltage to reduce the maximum current that can flow and save power. By doing so, we changed slightly the RGB color of some pixels, which means that we sacrificed the image quality for power saving. This function takes as input the current associated with the image and the supply voltage. The power consumption depends linearly on the latter, so this is the variable we changed to compute different results. The function operates in two modes: `saturated` and `distorted`. Below we can see the distortion caused by the DVS for the three datasets.

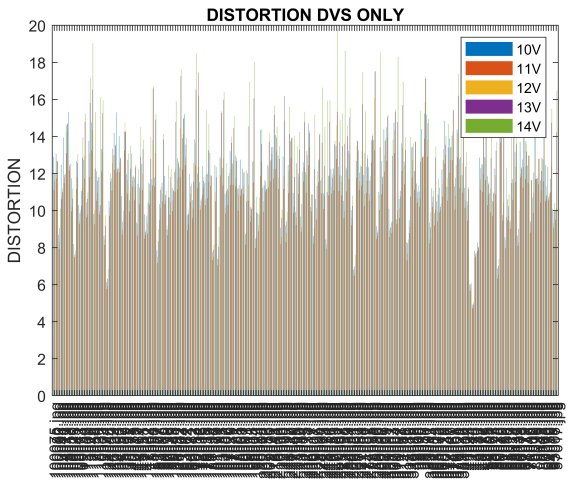

Figure 3.1: DVS only distortion for BSR dataset

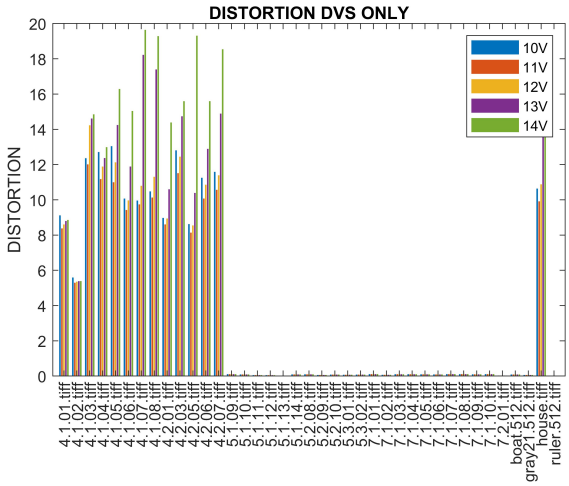

Figure 3.2: DVS only distortion for misc dataset

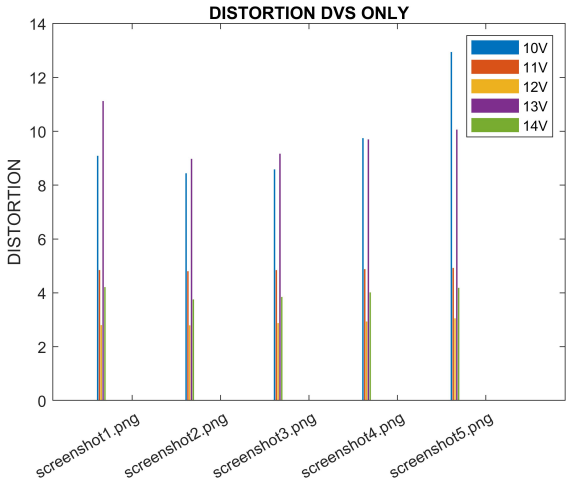

Figure 3.3: DVS only distortion for our screenshots

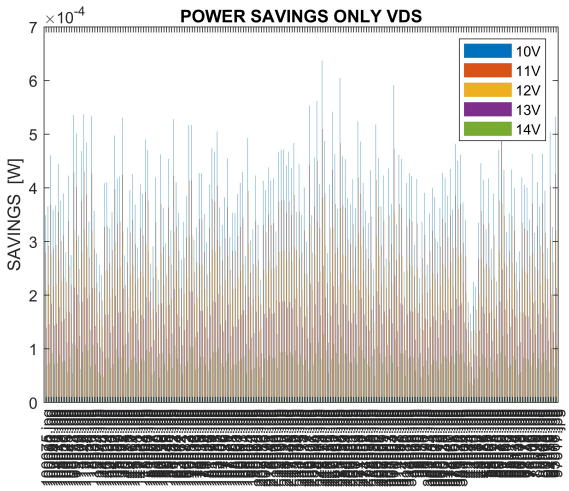

Figure 3.4: DVS only power savings for BSR dataset

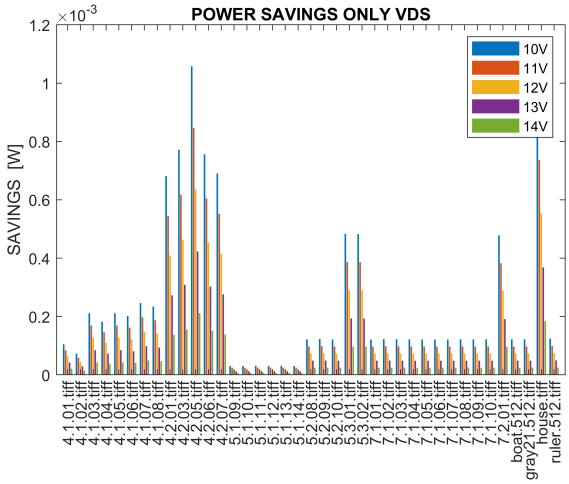

Figure 3.5: DVS only power savings for misc dataset

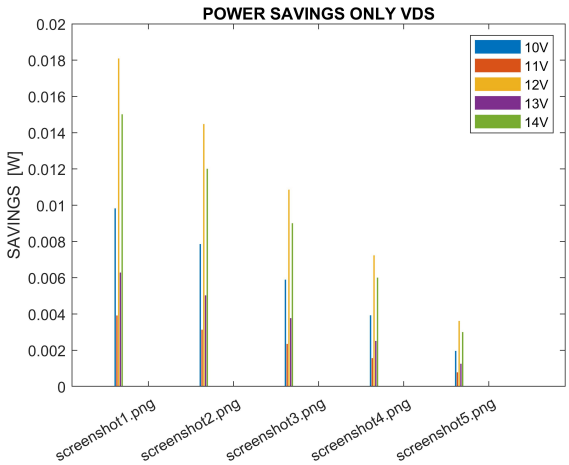

Figure 3.6: DVS only power savings for our screenshots

As it can be seen from the images from 3.1 to 3.3, there is a huge distortion between the images, but also a great power saving as it can be noticed in figures from 3.4 to 3.6. To compensate for the distortion caused by the DVS, we computed some manipulations on the images. The aim was to find the best image manipulation technique with good power saving but maintaining acceptable image quality. We used three different functions to manipulate the images: `Brightness Compensation`, `Contrast Enhancement` and `Brightness Contrast` which is the combination of the previous two functions.

3.2 Brightness Compensation

The first function to manipulate the images is `Brightness Compensation`. Its main idea is to increase (3.2) the pixel value in the HSV color space by b (3.1) to compensate for the decrease in the supply voltage as in the function 3.7.

$$b = \frac{(100 - ((100 * Vdd_mod)/Vdd))}{100} \tag{3.1}$$

$$V' = v + b \tag{3.2}$$

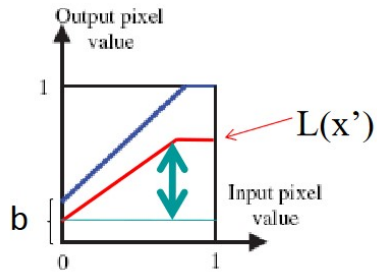

Figure 3.7: Brightness Compensation function

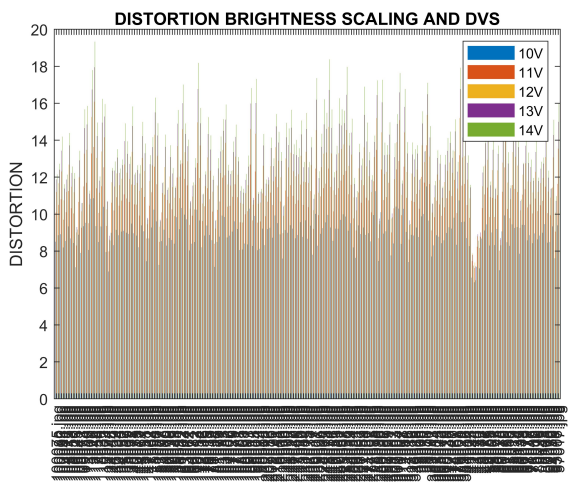

Figure 3.8: Brightness Compensation distortion for BSR dataset

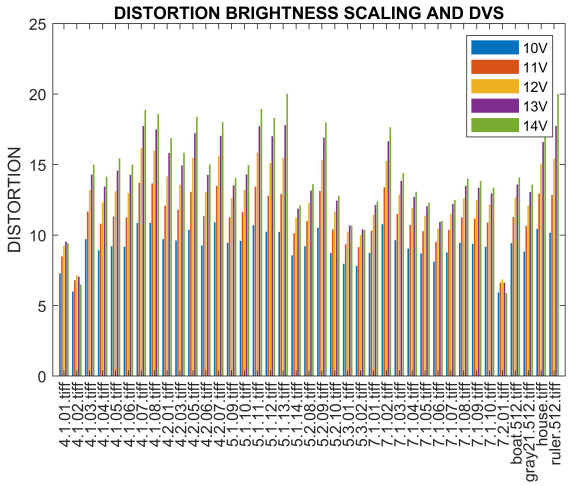

Figure 3.9: Brightness Compensation distortion for misc dataset

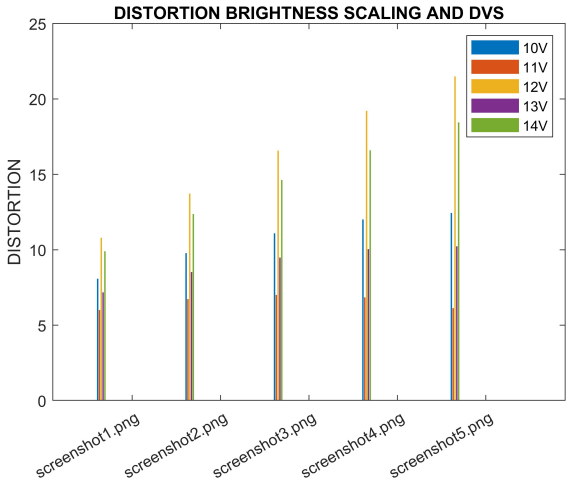

Figure 3.10: Brightness Compensation distortion for our screenshots

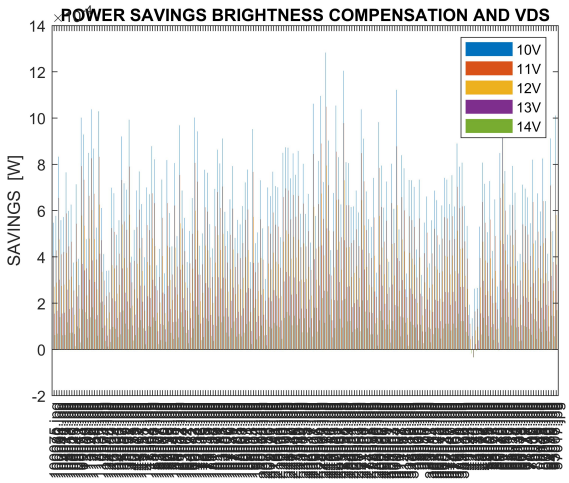

Figure 3.11: Brightness Compensation power savings for BSR dataset

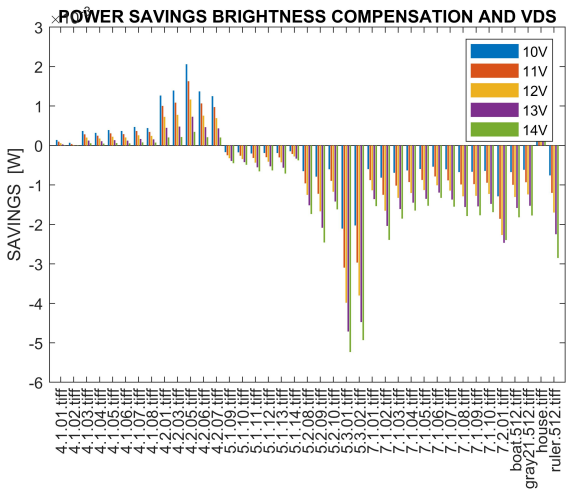

Figure 3.12: Brightness Compensation power savings for misc dataset

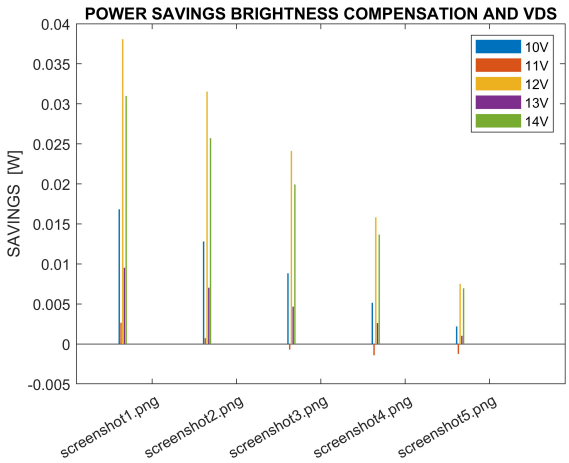

Figure 3.13: Brightness Compensation power savings for our screenshots

As it can be seen from the images from 3.8 to 3.10, the distortion of the pictures is slightly better. The most interesting results, though, are the ones for the power savings (figures from 3.11 to 3.13): for the BSR dataset the power savings doubled, and they grew slightly even for our screenshots; the power reduction is linear with the voltage scaling. But for the misc dataset there were some negative values, which means that for these photos the power consumption for the manipulated images is bigger than the original ones. We can only assume that this is because these photos are black and white, so they cannot be compensated through these manipulations.

3.3 Contrast Enhancement

The second manipulation is the `Contrast Enhancement`. In the HSV color space, we scaled (3.4) the pixel values by b (3.3) as we can see from the figure 3.14.

$$b = \frac{(100 - ((100 * Vdd_mod)/Vdd))}{100} \tag{3.3}$$

$$V' = \frac{V}{b} \tag{3.4}$$

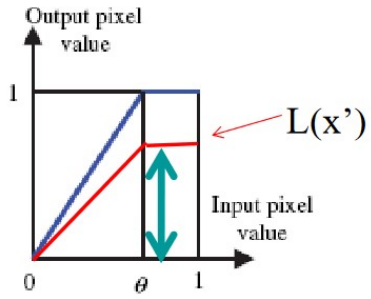

Figure 3.14: Contrast Enhancement function

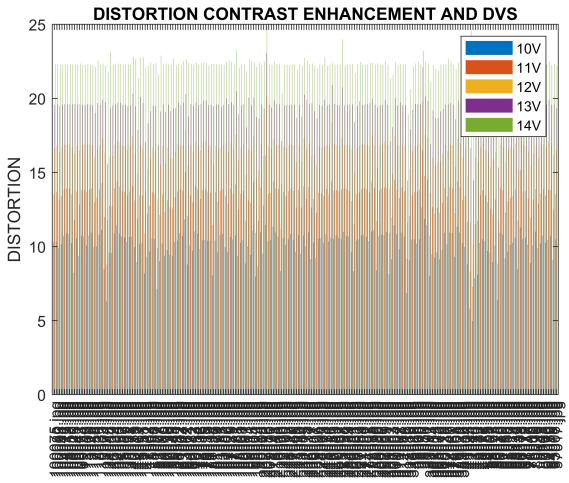

Figure 3.15: Contrast Enhancement distortion for BSR dataset

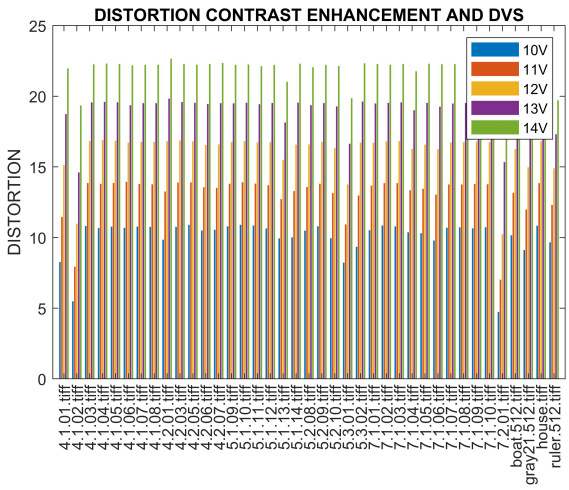

Figure 3.16: Contrast Enhancement distortion for misc dataset

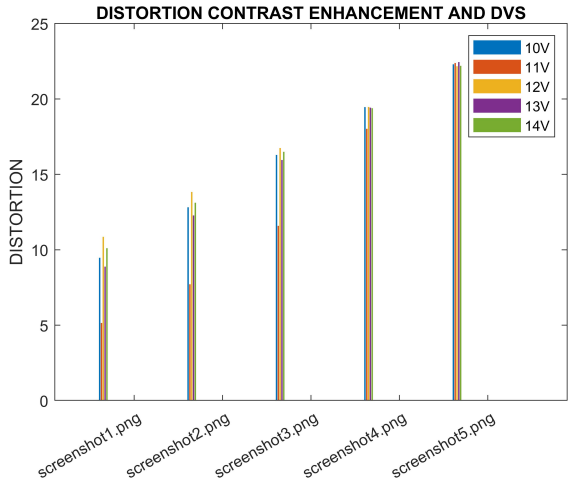

Figure 3.17: Contrast Enhancement distortion for our screenshots

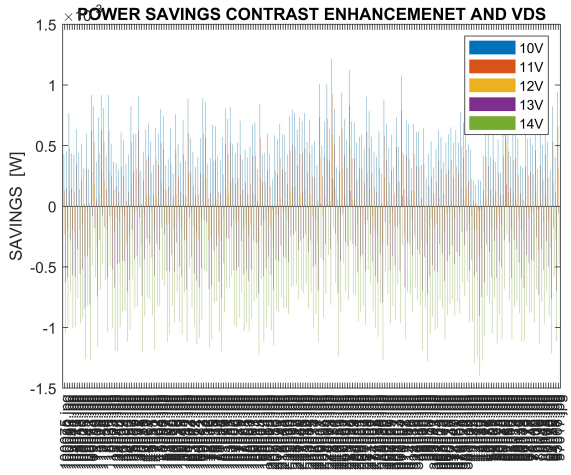

Figure 3.18: Contrast Enhancement power savings for BSR dataset

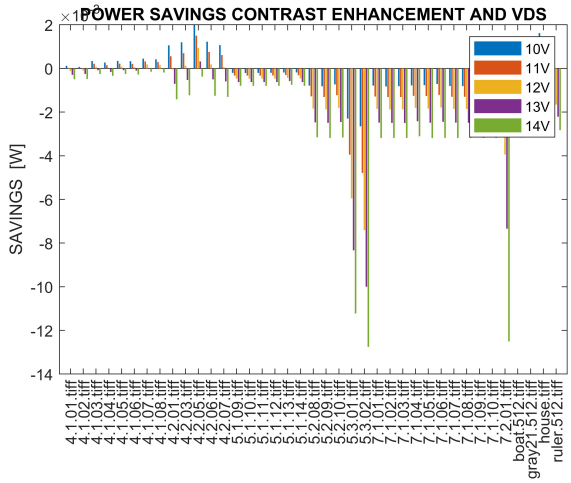

Figure 3.19: Contrast Enhancement power savings for misc dataset

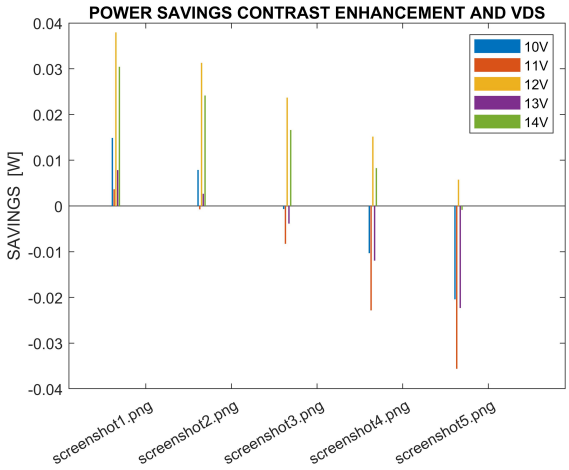

Figure 3.20: Contrast Enhancement power savings for our screenshots

As it can be seen from the images from 3.15 to 3.17, the distortion of the pictures is slightly higher than the previous ones and than the original photos. Regarding the power savings (figures from 3.18 to 3.20) the power reduction is still linear with the voltage scaling, but the results are a lot different. For the misc dataset there were negative values not only for the black and white images, but also for the colored ones, specially the values with higher Vdd, which means that for these photos the power consumption for the manipulated images is bigger than the original ones. The same pattern can be noticed in the other two datasets: In the pictures where is already a huge contrast this manipulation is nearly useless, and provide us with bigger power consumption.

3.4 Contrast Enhancement + Brightness Compensation

The last function truncates the high-end and low-end pixel values of the image but is adapted to maintain the contrast, as it can be seen in the figure 3.21. The variables gu, gl depend on the original supply voltage and the reduced one:

$$gl = -\frac{(1 - \frac{Vdd}{Vdd_{mod}}}{2} * \frac{Vdd_{mod}}{Vdd} \tag{3.5}$$

$$gu = b + gl \tag{3.6}$$

While c and d derive from :

$$c = \frac{1}{gu - gl} \tag{3.7}$$

$$d = \frac{gl}{gu - gl} \tag{3.8}$$

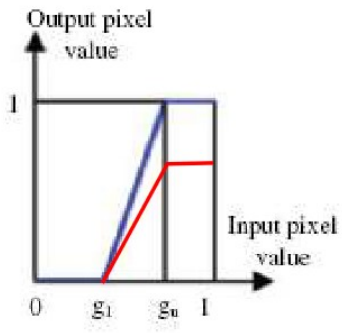

Figure 3.21: Combined function

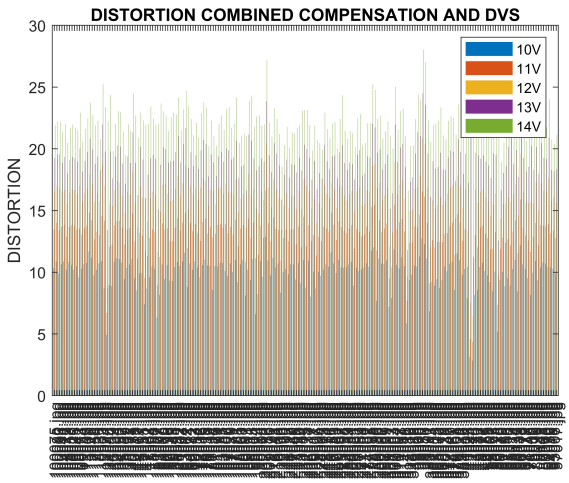

Figure 3.22: Combined distortion for BSR dataset

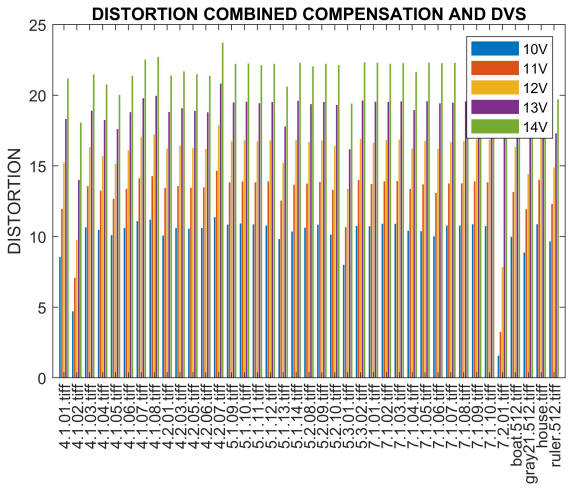

Figure 3.23: Combined distortion for misc dataset

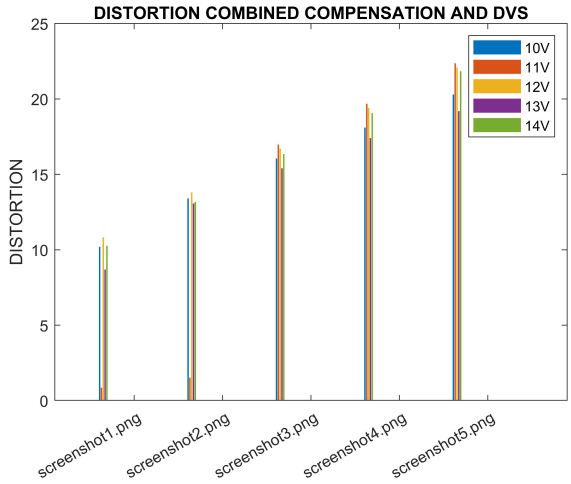

Figure 3.24: Combined distortion for our screenshots

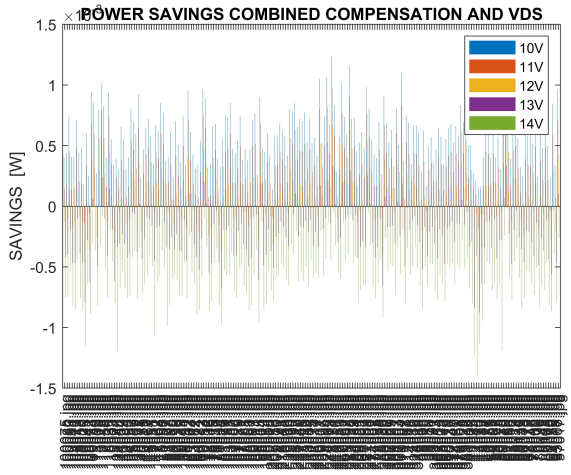

Figure 3.25: Combined power savings for BSR dataset

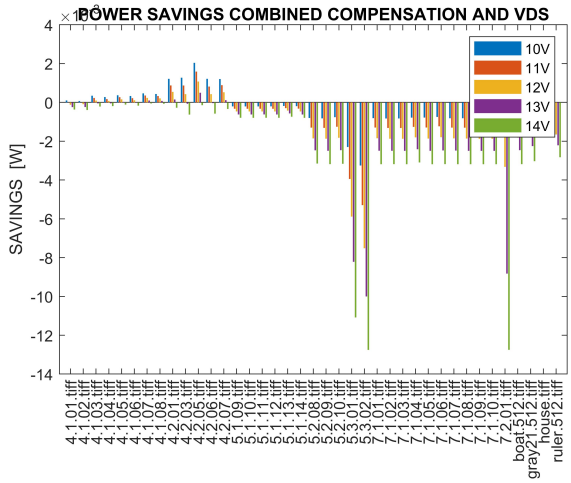

Figure 3.26: Combined power savings for misc dataset

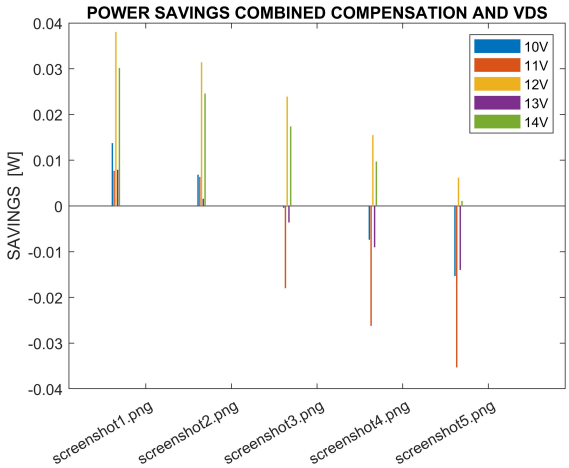

Figure 3.27: Combined power savings for our screenshots

As it can be seen from the images from 3.22 to 3.24, the distortion of the pictures is the worst. Regarding the power savings (figures from 3.25 to 3.27) the power reduction is very similar to the Contrast enhancement ones.

3.5 Optimal supply voltage

The goal now is to find the optimal supply voltage and compensated image with respect to the power consumption minimization and to the perceived visual quality. The image **??** is the original picture that we are going to manipulate.

Figure 3.28: Starting image

As we can see from the first set of images on which we performed just the DVS in saturation mode, the quality of the image is maintained until 12 V, then the picture becomes darker and darker. The photo remains recognizable even with the lowest voltages, so after comparing the images with the power savings obtained in the previous part, the best picture is the one created using 12 V.

On the second set of images, we performed just the DVS in distortion mode, the quality of the image is maintained just for the first image, then the color of the pixel becomes more and more distorted and different from the original image. The photo becomes not recognizable with the lowest

Figure 3.29: DVS only saturated images with decreasing V_{DD} [from 14 V to 10 V]

Figure 3.30: DVS only distorted images with decreasing V_{DD} [from 14 V to 10 V]

voltages, so after comparing the images with the power savings obtained in the previous part, the best picture is the one created using 13 V.

On the third set of images, we performed the DVS in distortion mode with the Brightness Compensation manipulation, the quality of the image is maintained just for all voltages, then the photo becomes darker and darker. The photo remains recognizable with the lowest voltages, so after comparing the images with the power savings obtained in the previous part, the best picture is the one created using 12 V.

On the fourth set of images, we performed the DVS in distortion mode with the Brightness Compensation manipulation, the quality of the image is maintained just for all voltages, then the photo becomes darker and darker. The photo remains recognizable with the lowest voltages, so after com-

Figure 3.31: Brightness scaling saturated images with decreasing V_{DD} [from 14 V to 10 V]

Figure 3.32: Brightness scaling distorted images with decreasing V_{DD} [from 14 V to 10 V]

paring the images with the power savings obtained in the previous part, the best picture is the one created using 12 V.

On the fifth set of images, we performed the DVS in saturation mode with the Contrast Enhancement manipulation, the quality of the image changes a lot with different voltages: For higher voltages, it is too bright, and for the lower ones it is too dark. The photo remains barely recognizable with the voltages at the extremes of the range, so after comparing the images with the power savings obtained in the previous part, the best picture is the one created using 12 V.

On the sixth set of images, we performed the DVS in distortion mode with the Contrast Enhancement manipulation, the quality of the image changes a lot with different voltages: For higher voltages, it is too bright, and for the lower ones it is too dark. The photo remains barely recognizable with the

Figure 3.33: Contrast Enhancement saturated images with decreasing V_{DD} [from 14 V to 10 V]

Figure 3.34: Contrast Enhancement distorted images with decreasing V_{DD} [from 14 V to 10 V]

voltages at the extremes of the range, so after comparing the images with the power savings obtained in the previous part, the best picture is the one created using 12 V.

On the seventh set of images, we performed the DVS in saturation mode with the Combined manipulation, the quality of the image changes a lot with different voltages: For higher voltages, it is too bright, and for the lower ones it is too dark. The photo remains barely recognizable with all the voltages as it is way too distorted. All the colors are way too different from the original image, so after comparing the pictures with the power savings obtained in the previous part, we cannot determine the best manipulation as they are not comparable with the original image.

On the eighth set of images, we performed the DVS in distortion mode with the Combined manipulation, the quality of the image changes a lot with different voltages: For higher voltages, it is

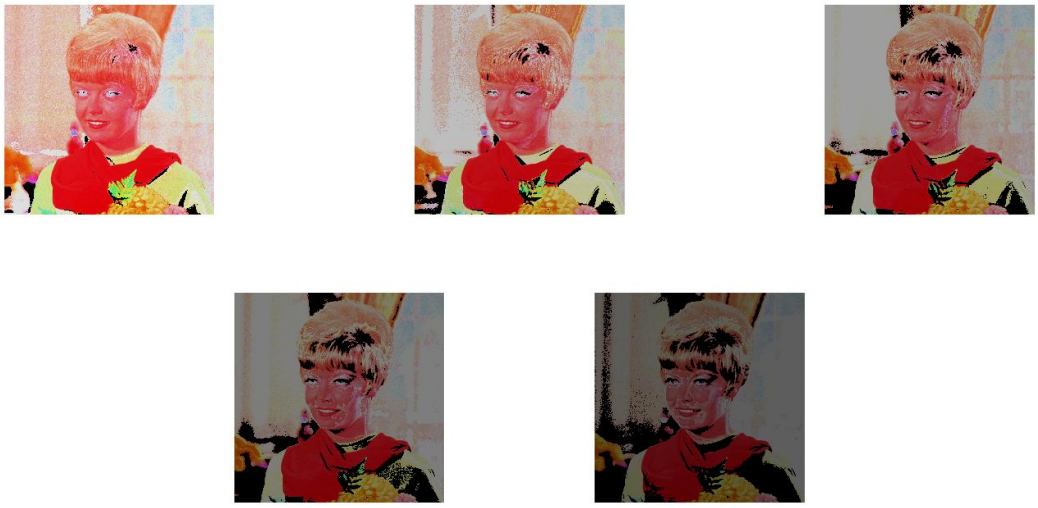

Figure 3.35: Combined function saturated images with decreasing V_{DD} [from 14 V to 10 V]

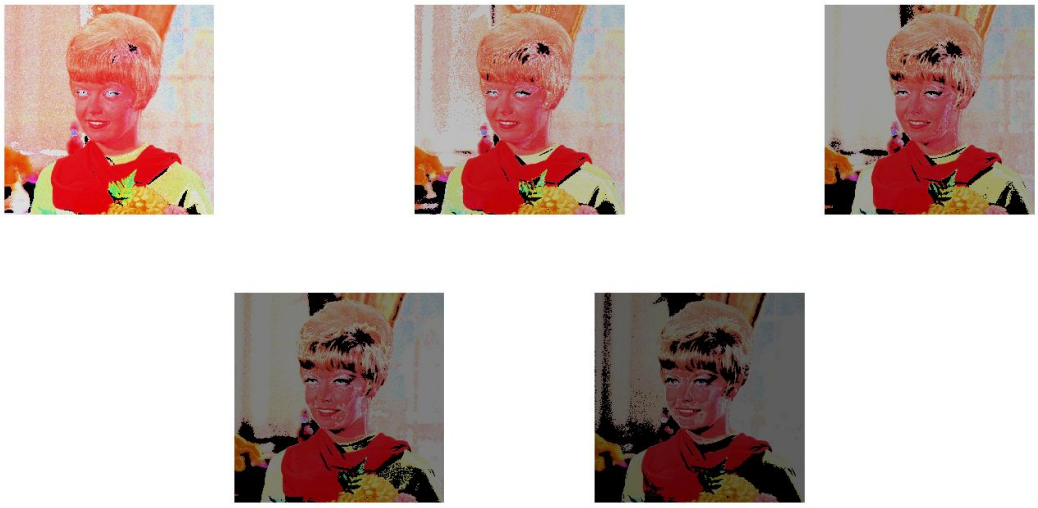

Figure 3.36: Combined function distorted images with decreasing V_{DD} [from 14 V to 10 V]

too bright, and for the lower ones it is too dark. The photo remains barely recognizable with all the voltages as it is way too distorted. All the colors are way too different from the original image, so after comparing the pictures with the power savings obtained in the previous part, we cannot determine the best manipulation as they are not comparable with the original image.

3.6 MATLAB scripts day2

Here, the most important MATLAB scripts and functions developed for our analysis for the day1 are listed and explained shortly.

- **Day2.m**: This is the main script that calls all the functions developed and returns the filled data structures with all the results used to plot figures.
- **auto_transf_listN.m**: These functions have been developed for the three folders of images. This function creates and fills the data structure associated with the image lists computing several parameters like current pixel consumption or the power consumed by the panel. Inside the function, we apply DVS or DVS with different compensations at each image with different values of VDD.
- **brightness_scaling.m**: This function contains the script that applies the brightness scaling to an image considering a value of b calculated depending on the scaling of Vdd.
- **contrast_enhancement.m**: This function contains the script that applies the contrast enhancement compensation to an image.
- **brightnessContrast.m**: This function contains the script that applies the combined compensation to an image.
- **distortion.m**: This script contains a function that is used to compute the distortion of an image after manipulation.
- **image_difference.m**: This script contains a function that calculates the difference between two images.
- **Ppanel.m**: This script contains a function in charge of evaluating the power consumption of the panel for a certain image. It returns a double value.
- **Ipanel.m**: This script contains a function in charge of computing the Current consumption for a certain image. It returns a matrix that contains value of current for each pixel for each channel RGB.
- **gray2rgb.m**: This script has been imported from `"Mathworks Community site"` and it has been used to manipulate black and white images with only one color channel to transform them into RGB images creating 3 color channels. The transformation introduces some margin of error because it approximates the color channels but the approximation is negligible for our analysis.
- **bar_graph_savings.m**: This function contains the script that take in input the image data set, the structure with the results of computation and the saving path. It plots a bar graph that shows the power savings compared to the power consumption of the display considering the original image.
- **bar_graph_distortion.m**: This function contains the script that take in input the image data set, the structure with the results of computation and the saving path. It plots a bar graph that shows the distortion obtained after DVS and compensation compared to the original image.