Energy Management in Mobile Systems

Lab 3: Image processing

Andrea Floridia - 224906, Robert Margelli - 224854

Contents

1. Purpose of the lab session 2
2. Day 1: The algorithm 3
2.1. Histogram equalization 4
2.2. Hungry-blue modification.....5
3. Day 2: The algorithm..... 7
3.1 Day 2: Analysis of results..... 8
4. Conclusions..... 10
5. References..... 10

1. Purpose of the lab session

During this lab session we learned how various image filtering and compensation techniques can be applied and how they differ in their behaviour based on the original image's characteristics. Two are the main parameters that were taken into account when evaluating a technique's effectiveness: distortion and power consumption. The latter being the most important factor in our considerations.
All of the image processing was done in the Matlab environment and heavily automated with two scripts, one for the first day of the lab session and one for the second.

2. Day 1: The algorithm

In order to have an efficient way for performing different experiments in a row, we developed an algorithm and implemented it in the MATLAB script `lab3day1automator.m`.

The strategy behind the script is the following: given a target distortion (fixed for all images), the algorithm progressively increments the value of the parameter `b` (the amount subtracted from the B matrix, i.e. the blue channel). At each iteration, it computes the distortion of the new image with respect to the original version. It continues increasing the value of b until the distortion of the new image is greater than the target value. Once this value is reached, it returns to the last value of `b` and exits the loop. This value is used to compute the new RGB representation of the image and its power consumption. Other figures are computed as well, namely the power saving percentages and the distortion percentages. Before repeating the same process on the next image, histogram equalization on the original image is also performed. Since this optimization does not depend on a specific parameter, but on the image itself, it is simply done by calling the built-in MATLAB function `histeq`.

We repeated this process for different values of distortions: 1%, 10%, and 15%. Results have been collected and stored in the files named `resultDay1HB1`, `resultDay1HB10`, and `resultDay1HB15`. Images have also been stored in three folders: `imgcomparHB1`, `imgcomparHB10`, `imgcomparHB15`.

Table 1 reports results gathered for the experiments listed above. As a matter of example, in the following we analyse in more detail results obtained by setting a target distortion value equal to 10%.

Filename	PowOriginal	PowHungryBlue	PowHistEq	DistPercOrig2HungryBlue	DistPercOrig2HistEq	PowSaveOrig2HungryBlue	PowSaveOrig2HistEq	hungryAmount
4.1.03.tiff	7.75E+05	6.62E+05	7.96E+05	9.980807	19.29657	14.664623	-2.696191	22
4.1.04.tiff	6.54E+05	5.82E+05	6.07E+05	8.731512	11.344698	11.07989	7.118403	16
4.1.06.tiff	8.99E+05	7.87E+05	7.66E+05	9.892364	11.197861	12.474813	14.808398	19
4.1.08.tiff	9.33E+05	8.43E+05	6.35E+05	8.932518	20.726696	9.638863	31.928725	19
4.2.01.tiff	2.36E+06	2.20E+06	1.62E+06	9.458472	17.932294	6.69211	31.532877	13
4.2.02.tiff	5.98E+06	5.30E+06	3.77E+06	9.247343	21.870749	11.379257	36.935833	28
4.2.03.tiff	2.86E+06	2.57E+06	2.14E+06	9.826834	13.575062	10.056246	25.23613	16
4.2.04.tiff	2.71E+06	2.48E+06	1.79E+06	9.140767	18.629336	8.493495	34.133145	16
4.2.05.tiff	5.50E+06	4.61E+06	3.33E+06	9.489079	21.37457	16.227944	39.438513	28
4.2.07.tiff	2.10E+06	1.97E+06	1.70E+06	9.824561	17.022054	6.276069	18.998686	13
shot1.png	2.14E+06	1.71E+06	1.52E+08	9.942354	2353.858303	20.116015	-7013.106104	25
shot2.png	1.37E+08	1.12E+08	7.51E+07	9.813004	22.212451	18.147375	45.09838	37
shot3.png	1.57E+08	1.30E+08	1.21E+08	9.512205	16.129811	17.602555	22.910607	37
shot4.png	1.27E+08	1.07E+08	7.53E+07	9.686255	20.584637	15.409312	40.585218	31
shot5.png	1.47E+08	1.23E+08	1.16E+08	9.174818	15.443046	16.54074	21.300413	34

Table 1 – Results with target distortion of 10%.

2.1 Histogram equalization

The purpose of the technique known as histogram equalization is flattening the histogram of the Value channel in the HSV space representation of an image such that the resulting histogram resembles a uniform distribution. This technique increments the difference between light and dark regions in images, modifying their luminance.

Referring to Table 1, most interesting cases are the following:

- 4.1.03 and shot1: normally the purpose of this technique is to reduce the power consumption. These two pictures are a clear example where it is not worth to use histogram equalization because the starting images are dark images. The rightmost pictures in figure 1 are those related to histogram equalization, here it's evident that the resulting images are brighter than the original ones (leftmost pictures) yielding a higher power consumption (column PowSaveOrig2HistEq in Table 1) than the nominal one. Furthermore, the distortion (column DistPercOrig2HistEq) introduced in shot1 is not acceptable, since it's not possible anymore to distinguish the content displayed.

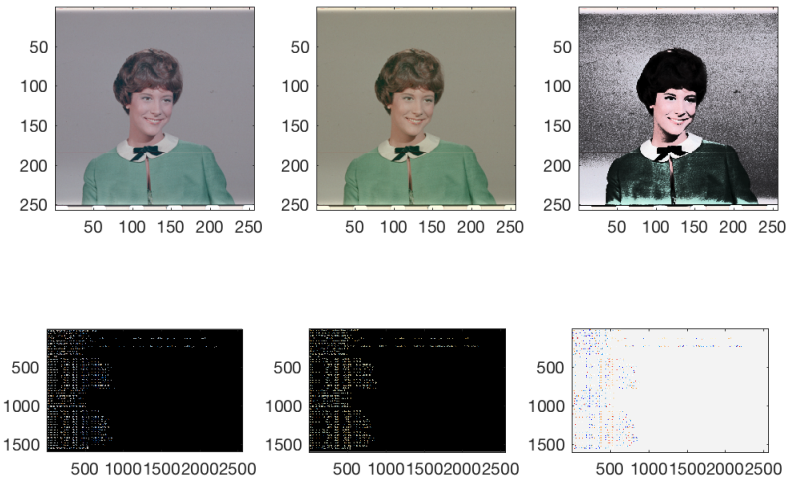

Figure 1 – The top picture is 4.1.03 whereas the bottom one is shot1.

- Shot3 and shot5: in contrast with what analysed in the previous bullet, these two pictures are an example of a situation in which it is worth to adopt this technique. For both, distortions are not negligible. However, there are also substantial power savings. Indeed, more than 20% of power saving is achieved in both situations. By looking at the rightmost pictures in Figure 2 we observe that, despite the high distortion percentages, it's still acceptable to display these images. Hence applications that do not directly deal with displaying visual media, such as web browsing or an IDE (the ones represented in figure 2), the appearance can be sacrificed for power savings. Indeed options for changing to high contrast representations of images are available on most operating systems (e.g. Windows 8). Of course, this reasoning does not apply to visual media-intensive applications, such as media players.

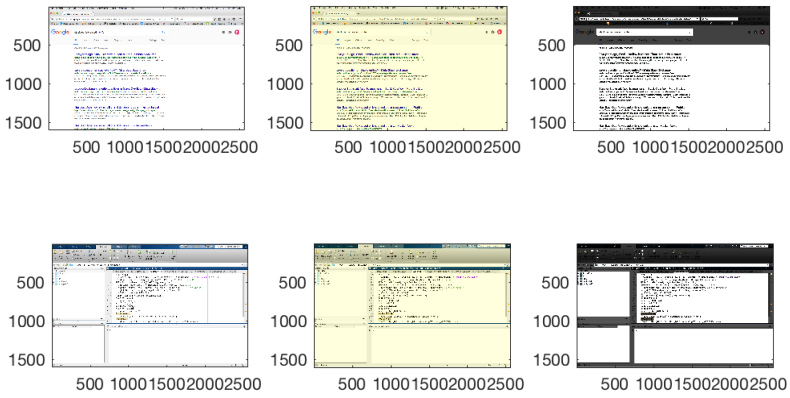

Figure 2 – First row: shot3. Second row: shot5.

- 4.2.05: this picture (Figure 3) is a possible example of a resulting image displayed on a media player and here appearance is important more than anything else. Notwithstanding the huge power savings, the image is clearly not acceptable. Shapes are still recognizable, but colours are almost lost.

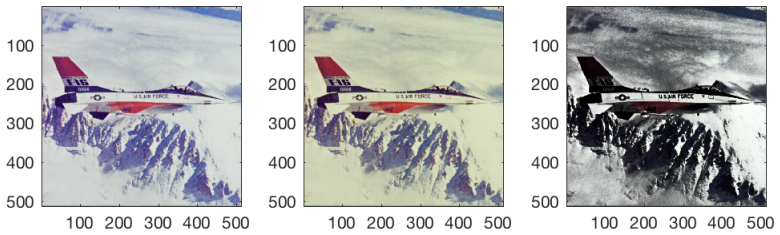

Figure 3 - 4.2.05.

2.2 Hungry-blue modification

The so-called hungry-blue image modification simply aims at reducing power consumption by decreasing the blue component from each pixel composing an image. The following cases highlight the effectiveness of this approach:

- 4.2.05 and 4.1.03: among the media oriented image set (i.e. excluding the screenshots) these images are the ones for which this technique provides the best saving in terms of power consumption (~16.2% and ~14.7% respectively). This is due to the fact that they both contain a large set of pixels with a high value for the blue channel. If we take for instance 4.2.01 which is mostly represented by the red colour we see the lowest saving (~6.7%).
  We notice that with this technique images tend to become yellowish. This is, in some cases, acceptable even in visual media oriented applications. For instance, some video software applications provide a “movie” mode when watching a film which in fact displays content in this manner; others instead apply this effect to lower the stress affecting the human eye when staring at a computer screen [1]. The target of these applications is clearly not reducing power consumption, but we can see how the latter can be integrated in the design of such applications without additional programming effort.

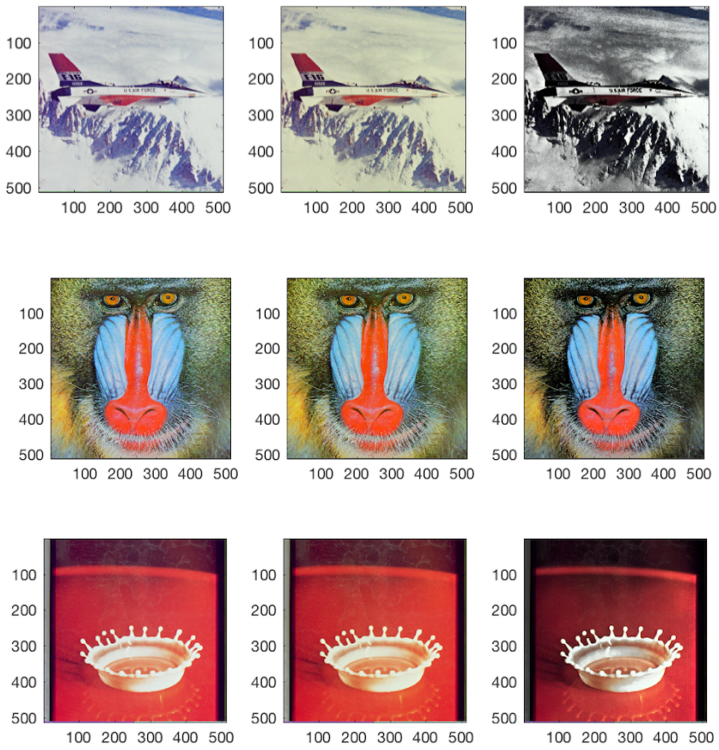

Figure 4 – In order: 4.2.05, 4.1.03 and 4.2.01. The second column shows the results from hungry-blue filtering.

- Shot3: as far as the screenshots are concerned we notice that overall hungry-blue behaves well. However, in the experiment with a target distortion of 1% we noticed that shot3 sets itself apart from the others (all with saving below 1%) with a saving of ~2.2%. Shot3 is in fact a typical Google search result page, mostly composed of white pixels and so most contain a high value in the blue channel. In general, trying to achieve a distortion of only 1% is a hard task as it requires an extremely fine granularity of the `b` parameter.

3. Day 2: The algorithm

The developed algorithm is very similar to the one described in chapter x. In this case, the MATLAB script we are referring to is called `lab3day2automator.m`.

However, despite similarities, the two algorithms have quite different goals. Here, the first step is finding a value for V_{DD} such that the power consumption is lower than the initial one and the distortion below a given threshold (fixed for all images). Once this lower V_{DD} is found, the flow described during the lab session is applied. The image with DVS is computed and then the three techniques presented during the lab session are applied, namely: brightness compensation, contrast enhancement and a combination of the two. All of these depends on the value chosen for the `b` parameter (which has nothing to do with the one computed for the hungry-blue filter).

Since this parameter needs to be set, we decided to use an iterative approach. Given the new voltage value, the algorithm starts increasing values of the parameter trying to reach a distortion equal to half the initial threshold. If, for some reasons, it is not able to reach a value of distortion below the target, then the value of the parameter that yields the lower value of distortion is kept. This flow applies to brightness compensation, contrast enhancement and the combination of both (called Combo in our script). It is worth to be mentioned that since in contrast enhancement we are dealing with a parameter which is multiplied instead of being added, we included the constraint that the value should be greater than 1 in order to be considered as valid (whereas for brightness compensation it's between 0 and 1).

Generally, at the end of these steps we end up with a distortion at least lower than the initial one (when only DVS is applied) and in some cases also halved. Hence, we have a slack of distortion that can be exploited to further reduce V_{DD} in order to reach a lower power state. Another iteration starts, and for the three images previously compensated the algorithm diminishes the value of V_{DD} until the distortion is again more or less equal to the one specified at the beginning.

The final goal is to reduce the supply voltage as much as possible by leveraging these techniques, obtaining results not achievable by resorting solely to DVS.

Finally, results are stored in files along with the resulting images. As in the previous assignment, we tried several values of distortion: 1%, 10% and 15%. Results are stored in files `resultDayperc1`, `resultDayperc10`, `resultDayperc15`. Images are stored in the `imagecomparXdist` folder (here X is the value of the target distortion). For the sake of completeness, we also saved the images after further lowering the supply voltage for all three techniques analysed (`imagecomparOpt` folder).

3.1 Day 2: Analysis of results

Although the above method is applied in equally to all 15 images, results are strongly dependant from the input picture. From the theory we know that brightness compensation works well on images rich in colours (i.e. with a wide colour spectrum) while contrast enhancement is more suited for images with small differences. We here propose two tables containing results from experiments with a target distortion of 10%. The notation "Opt" (optimized for power) is relative to the final step in our algorithm where we try to furtherly apply DVS to distorted images until the original target distortion is reached.

Table 2 reports few of the results we obtained from our experiments. The upper portion is related to power savings in both non-optimized and optimized cases, while the bottom reports absolute values found by running our algorithm. In detail, columns 2 to 5 contain V_{DD} values of the optimized cases and 6 to 8 the value of the `b` parameter for the three techniques that were used.

Filename	Orig2DVSPowSav	Orig2BrightDVSPowSav	Orig2ContrDVSPowSav	Orig2ComboDVSPowSav	Orig2BrightDVSPowSavOpt	Orig2ContrDVSPowSavOpt	Orig2ComboDVSPowSavOpt
4.1.06.tiff	42.389.653	34.999.274	35.810.596	36.257.757	53.169.594	51.089.818	51.410.201
4.2.05.tiff	44.116.946	36.631.657	37.223.015	35.883.685	54.244.551	52.871.354	54.631.746
4.2.07.tiff	41.958.791	35.539.936	35.531.737	33.632.646	51.753.266	50.890.959	53.446.981
shot1	21.892.003	21.892.003	19.071.925	21.892.003	21.892.003	28.794.118	21.892.003

Filename	newVddBrightness	newVddContrast	newVddCombo	foundVdd	foundBrightB	foundContrB	foundComboB
4.1.06.tiff	12.275.000	12.425.000	12.425.000	13.200.000	0.100000	1.150.000	0.050000
4.2.05.tiff	12.300.000	12.400.000	12.250.000	13.175.000	0.125000	1.150.000	0.075000
4.2.07.tiff	12.325.000	12.375.000	12.150.000	13.175.000	0.100000	1.150.000	0.075000
shot1	13.375.000	12.575.000	13.375.000	13.375.000	0.000000	1.150.000	0.000000

Table 2 – Results for the notable cases with target distortion of 10%.

Some notable cases:

- Shot1: An example where the effectiveness of contrast enhancement is confirmed by this screenshot of a terminal application, which is mostly black and in average the image is uniform. Here, this technique is the best providing a power saving of ~28.8% w.r.t. the original one, in opposition to ~21.9% of the other two techniques.
- 4.2.07 and 4.2.05: Here "Combo" yields the best results. In fact, a final V_{DD} of 12.15V and 12.25V respectively have been reached by the algorithm. These entail a power saving of ~53.5% and ~54.6%, the best among other values of savings for the same images.
  These pictures are rich in colours, shades and details so one technique might not be enough, hence the algorithm managed to apply the combination of the two compensations effectively. In practice the differences among technique applications is slightly noticeable (Figure 5).

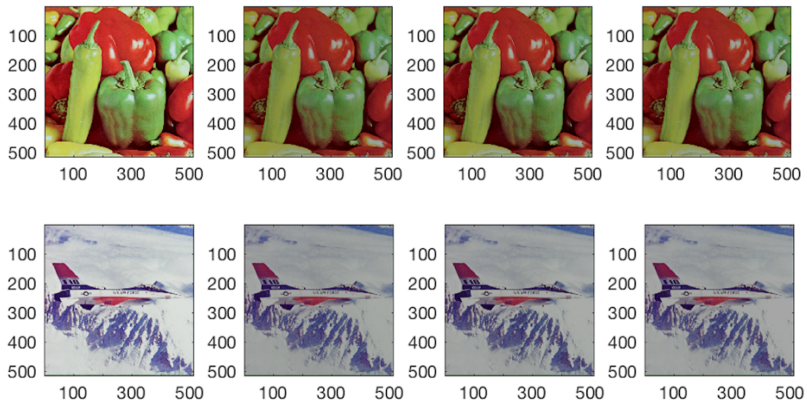

Figure 5 –4.2.07 and 4.2.05. The columns are in order: original, optimized brightness, opt. contrast, opt. combo.

- 4.1.06: This is an occurrence where brightness compensation has worked best yielding ~53% power saving versus ~51% with the other techniques. Altogether the saving is not extremely better and this case confirms that with pictures with a high level of detail and colours it is hard to predict which technique works best.

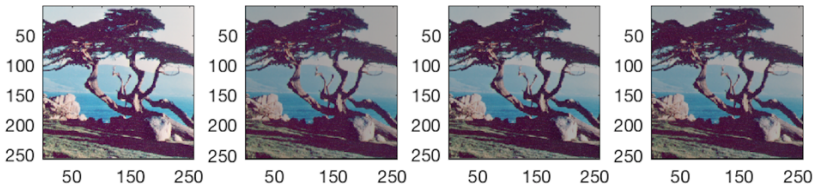

Figure 6 – 4.1.06. Original and optimized versions.

4. Conclusions

In this lab session we learned about power-aware image processing techniques and analysed results from several cases. In general, we were able to predict results based on what we know from the theory and our framework provides a scalable way to test image processing techniques. Their efficiency depends directly on which image we are considering so both screenshots and normal pictures were used in the experiments.

5. References

[1] f.lux: software to make your life better: *https://justgetflux.com/*