IC Project 2

November 2023

Contents

1 Introduction **3**

2 Part 1 **3**
2.1 Channel extractor 3
2.2 Image tools 4
2.2.1 Negative 4
2.2.2 Mirror 5
2.2.3 Rotation 6
2.2.4 Intensity 6

3 Part 2 **7**
3.1 Golomb coding 7

1 Introduction

This project was around the design and implementation of tools for manipulating audio and image files. In the first part, it was implemented a simple image channel extractor and also some tools for image edition. In the second part, it was implemented a class for Golomb coding to encode/decode integers.

2 Part 1

2.1 Channel extractor

The first program implemented in file **ex_1.cpp** provided the functionality to extract a channel from an arbitrary RGB image. In order to build and test it in path **IC-42078/PART_1/**:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ ./ex_1 <path to image> <channel>
```

Passing as argument the path to the image and the channel (0, 1 or 2). The program will output the original image along with the single channel version. The example below shows the output for the image baboon.ppm after running the program for the channel 0 with the command *./ex_1 ../baboon.ppm 0*.

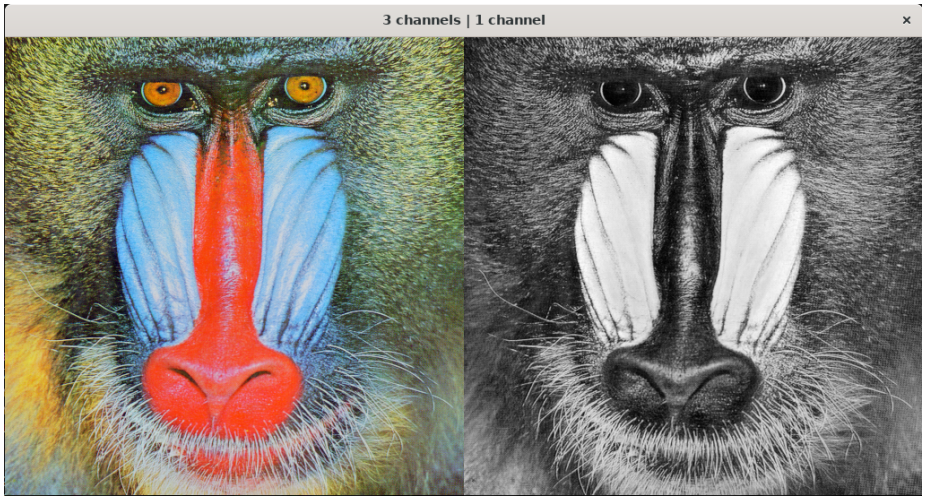

Figure 1: Original image at the right and single channel (Channel 0) image at the left

The program will also output a file for the channel requested with the standard name *ex_1_channel_n_image.jpg* with n as the requested channel.

2.2 Image tools

The second exercise of the guide consisted in the elaboration from scratch of four tools for image editing. They are the negative tool, the mirror tool, the rotation tool and the intensity modifier tool.
In order to build and test it in path **IC-42078/PART_1/**:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ ./ex_2 <path to image> <option> <arg>
```

In which *option* is in the set [0, 1, 2, 3] and the *arg* is the possible argument value passed to modify the image. The modified image is written in the file *ex_2_output_image.jpg*.

2.2.1 Negative

The negative tool was implemented based on the first exercise, in which the image was manipulated pixel by pixel. Now, instead of extracting a channel to make a new image, the values from each channel were set as the complement to 255, the maximum channel value.
The following image show the result of the command *./ex_2 ../baboon.ppm 0*

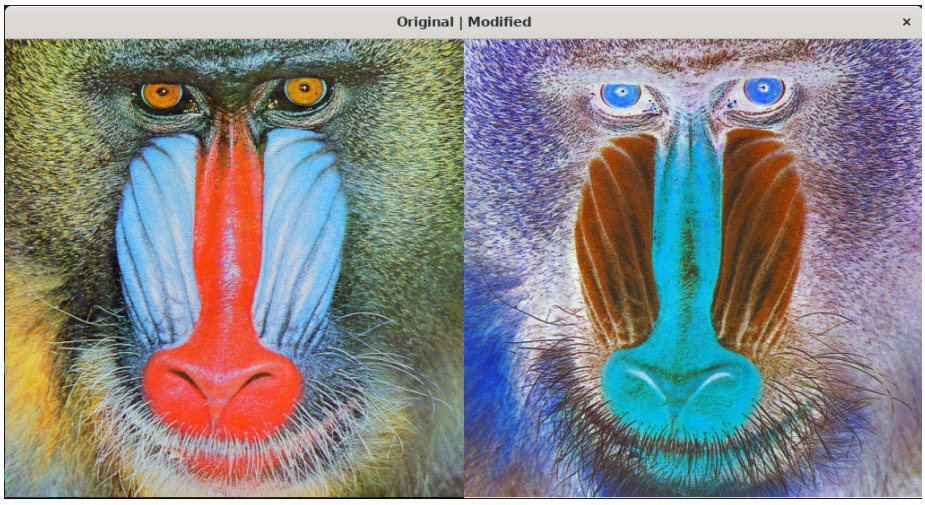

Figure 2: Original image at the right and negative image at the left

2.2.2 Mirror

The mirror tool was easily implemented using the template for pixel manipulation, only considering the direction to mirror around. The direction is passed as the *arg* value, that can be 0 (Horizontal mirror) or 1 (Vertical mirror).
The figure 3 shows the image vertically mirrored, while the figure 5 shows it horizontally mirrored:

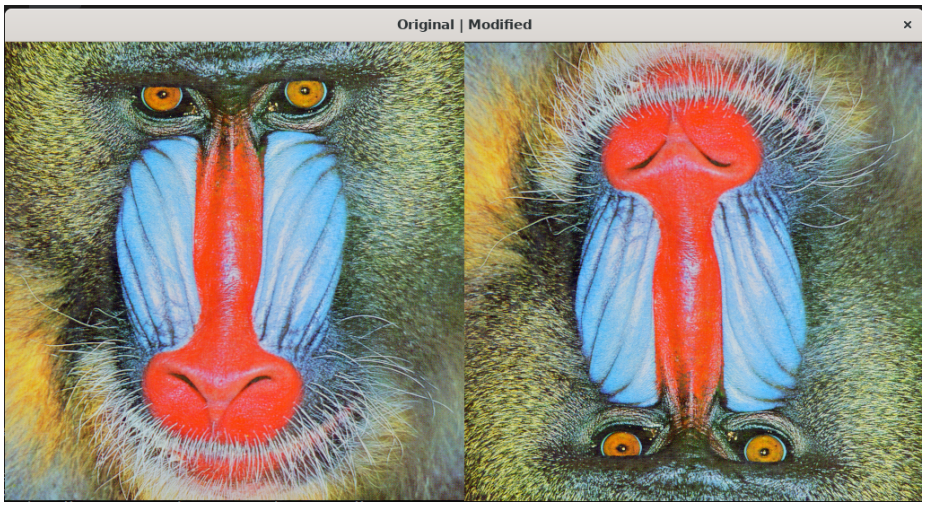

Figure 3: Original image at the right and vertically mirrored image at the left

Figure 4: Original image at the right and horizontally mirrored image at the left

The commands to output the two images above were, respectively, *./ex_2 ../baboon.ppm 1 1* and *./ex_2 ../baboon.ppm 1 0*.

2.2.3 Rotation

The rotation tool was implemented using a simple rotation matrix in which θ is the rotation angle:

$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

The rotation angle is always 90° times the number of turns. The number of turns is passed in *arg* and can be any integer.
The figure 5 shows the output for 3 turns with the command *./ex_2 ../baboon.ppm 2 3*.

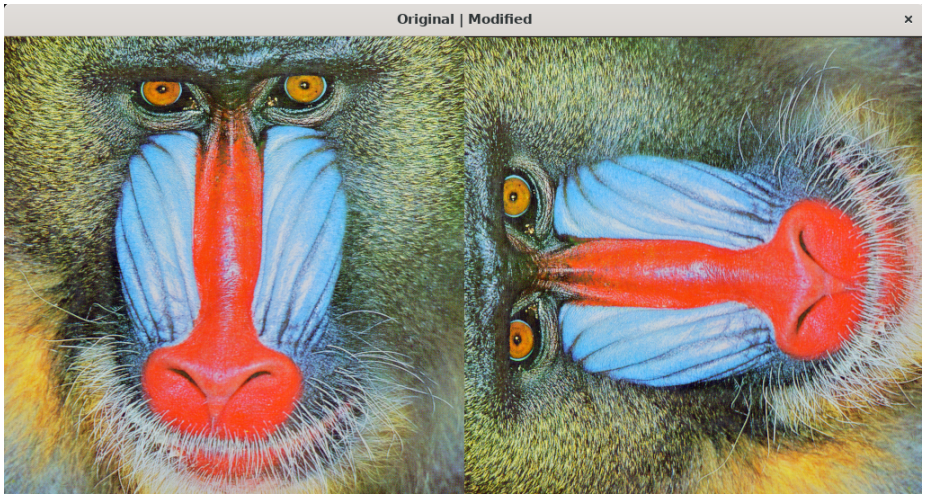

Figure 5: Original image at the right and three times rotated image at the left

2.2.4 Intensity

The intensity modifier was implemented by adding a value passed by the user in the *arg* to be added to each channel of each pixel in the image. Although the user could pass any value to the program, it will crop the final value to the interval [0, 255].
The two images below show the effect of increasing by 100 and decreasing 120, respectively, in the intensity. The commands to produce these effects were, also respectively, *./ex_2 ../baboon.ppm 3 100* and *./ex_2 ../baboon.ppm 3 -120*.

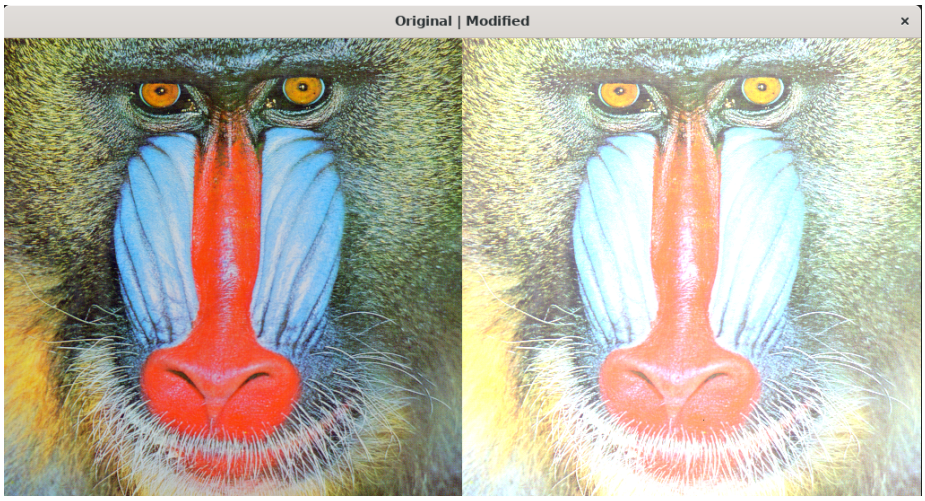

Figure 6: Original image at the right and image with intensity increased by 100 at the left

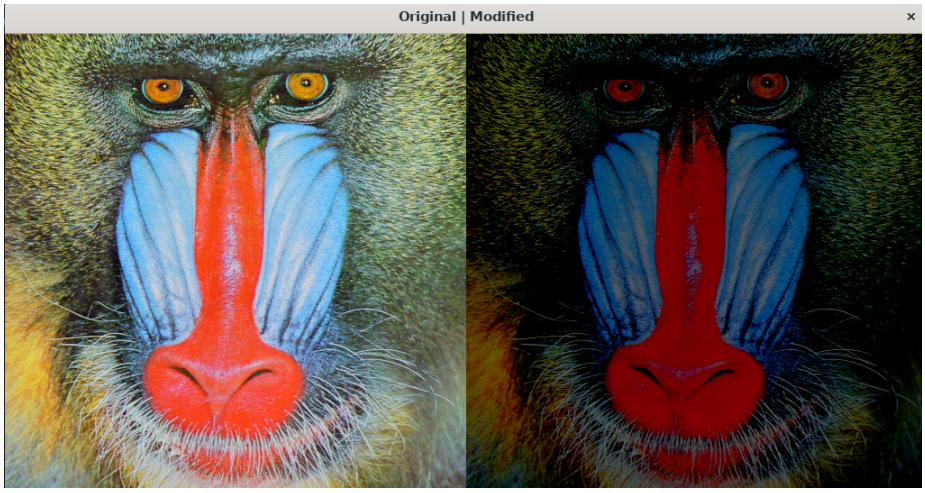

Figure 7: Original image at the right and image with intensity decreased by 120 at the left

3 Part 2

3.1 Golomb coding

The second part of the work consisted of the implementation of a class for Golomb coding. It implemented functions for encoding an integer and decode

the respective codeword.
In order to build and test it in path **IC-42078/PART_2/**:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ ./ex_3 <m> <value>*
```

In which m is the encoding parameter and *value* is the value to be encoded, which is optional. Without providing *value*, the program will output a list of the encoded values, followed by the decoded value generated from the encoding for the integers from 0 to 20.
The following output is the result of the command *./ex_3 5*:

- Original Value: 0 Encoded Bits: 0000 Decoded Value: 0
- Original Value: 1 Encoded Bits: 0001 Decoded Value: 1
- Original Value: 2 Encoded Bits: 0010 Decoded Value: 2
- Original Value: 3 Encoded Bits: 00110 Decoded Value: 3
- Original Value: 4 Encoded Bits: 00111 Decoded Value: 4
- Original Value: 5 Encoded Bits: 01000 Decoded Value: 5
- Original Value: 6 Encoded Bits: 01001 Decoded Value: 6
- Original Value: 7 Encoded Bits: 01010 Decoded Value: 7
- Original Value: 8 Encoded Bits: 010110 Decoded Value: 8
- Original Value: 9 Encoded Bits: 010111 Decoded Value: 9
- Original Value: 10 Encoded Bits: 011000 Decoded Value: 10
- Original Value: 11 Encoded Bits: 011001 Decoded Value: 11
- Original Value: 12 Encoded Bits: 011010 Decoded Value: 12
- Original Value: 13 Encoded Bits: 0110110 Decoded Value: 13
- Original Value: 14 Encoded Bits: 0110111 Decoded Value: 14
- Original Value: 15 Encoded Bits: 0111000 Decoded Value: 15
- Original Value: 16 Encoded Bits: 0111001 Decoded Value: 16
- Original Value: 17 Encoded Bits: 0111010 Decoded Value: 17
- Original Value: 18 Encoded Bits: 01110110 Decoded Value: 18
- Original Value: 19 Encoded Bits: 01110111 Decoded Value: 19
- Original Value: 20 Encoded Bits: 01111000 Decoded Value: 20