Report of Project of Computer Architecture

Emirhan Balcı Nº17048

May 24, 2023

1 Introduction

The aim is to develop a program in Assembly RISC-V to locate characters from the Star Wars saga in an image. Given a file with an image in RGB format, the program should generate a new image that identifies the character chosen by the user.

It is seen that each character has a different color tone in the image. Master Yoda has more shades of green, Darth Maul is more red, and Boba Fett is closer to cyan. These characters can be distinguished by their color tones.
Each of them has a dominant color tone which is mentioned above. The best way to locate a character is to find the center of mass of the character by the dominant pixels.

2 RGB

In this project, the image is an RGB formatted file with 8 bits of color depth. Which means every pixel contains three bytes corresponding to the RGB values. The image can be represented as a matrix where each value is a pixel with three components. However, image pixels are stored sequentially in a file in raw major1 order:

$$[\underbrace{R_{11}G_{11}B_{11}\cdots R_{1n}G_{1n}B_{1n}}_{1^{a}\text{ linha}}\underbrace{R_{21}G_{21}B_{21}\cdots R_{2n}G_{2n}B_{2n}}_{2^{a}\text{ linha}}\cdots\underbrace{R_{m1}G_{m1}B_{m1}\cdots R_{mn}G_{mn}B_{mn}}_{m\text{-ésima linha}}].$$

In order to convert the image into RGB formatted file, ImageMagick is used. With the use of the proper commands images can be converted into different formats. Before turning into PNG or JPG from RGB, these parameters should be used:
-size 320x180 and -depth 8

Some examples of converting between JPEG, PNG and RGB formats are as follows:
convert imagem.jpg imagem.rgb
convert -size 320x180 -depth 8 imagem.rgb imagem.png
convert -size 320x180 -depth 8 imagem.rgb imagem.jpg

3 HSV Color Space

The HSV color space encodes colors into Hue, Saturation, and Value components:
→ Hue represents hue on a color wheel.
→ Saturation represents the purity of the color. A pure color is said to be saturated. Mixing white, the color fades and is less saturated. Gray has zero saturation.
→ Value represents the lighting. The zero value represents darkness, that is, black. A high value means good lighting and colors are clearly visible.

Section	Name	Hue (°)
$R > G \geq B$	Orange	$60\frac{G-B}{R-B}$
$G \geq R > B$	Spring Green	$120 - 60\frac{R-B}{G-B}$
$G > B \geq R$	Yellowish Green	$120 + 60\frac{B-R}{G-R}$
$B \geq G > R$	Azure	$240 - 60\frac{G-R}{B-R}$
$B > R \geq G$	Violet	$240 + 60\frac{R-G}{B-G}$
$R \geq B > G$	Pink	$360 - 60\frac{B-G}{R-G}$

The calculation formulas for the other two components S and V are not used in the project because they will not be needed in this work. The main reason why the Hue value is used is to detect shades of color.

4 Image Segmentation

To find the characters, the first step is to identify the pixels for each. This process is called image segmentation. Since each character has a different tone from the others, we can only distinguish them with the Hue component. Table 1 shows the tone ranges that can be used for segmentation.

Character:	Hue:
Yoda	[40◦, 80◦]
Darth Maul	[1◦, 15◦]
Mandalorian	[160◦, 180◦]

5 Center of Mass

In order to calculate the center of mass, the program will traverse the image pixel by pixel. And it will call a function called 'indicator' that checks if the pixel belongs to the character or not. The coordinates (cx, cy) correspond to the "center of mass". If it is, the coordinates of the pixel will be added to the center of mass values. And every time it hit a valid pixel, a counter will be increased.

$$c_x = \frac{1}{N}\sum_x\sum_y \mathrm{I}_p(x,y)x, \qquad c_y = \frac{1}{N}\sum_x\sum_y \mathrm{I}_p(x,y)y, \qquad N = \sum_x\sum_y \mathrm{I}_p(x,y).$$

The calculation is simply the average of the coordinates that belong to the character. The calculation is done in integer arithmetic (do not use floating point instructions).

6 Functions

There are seven functions in this project. Here are the explanations for each one:

6.1 `main`

It's the main function of the program. First, prompts a question to select a character. If it's chosen then, call the rest of the functions respectively.

6.2 `read_rgb_image`

Reads a file with an image in RGB format into an array in memory called `buffer`. It's a void function and takes no arguments. First opens the file and checks if there is an exception occurring. Then, reads the file into `buffer` within the length of `172800` which is the number of all pixels in the image. Then, closes the file and returns.

6.3 `write_rgb_image`

Creates a new file with an image in RGB format. It's a void function and takes no arguments. First opens a file called `output.rgb` and checks if there is an error or not. Then, write the values of `buffer` into the file. Then, closes the file and returns.

6.4 `hue`

Calculates the Hue component from the R, G, and B components of a pixel. Takes three arguments and returns only one argument which is `a0`. These arguments are stored in `a0`, `a1` and `a2`. There are 6 `if` conditions to find the range of hue value of these RGB values. If those RGB values don't fit into any range, the function returns `1`, otherwise returns `0`.

6.5 `indicator`

Indicates whether or not RGB values belong to the character. Takes four arguments and returns only one argument which is `a0`. The first argument indicates which character will the function look for. And the rest represents the RGB values respectively. First calls `hue` function to get the hue value of these values. And it checks if the hue value fits into the character's hue interval. if it does returns `1`, otherwise returns `0`.

6.6 `location`

Calculates "center of mass" for the character. Takes two arguments and returns two values. Argument values are 'character' and '`buffer` address'. It iterates the whole image and calls `indicator` for each pixel. If all RGB values are the same, the pixel is not included in the calculation. If the pixel is one of the pixels of the character, `x` and `y` values are added up into registers. And the counter for the calculation is increased. After the iteration `cx` and `cy` values are divided with the counter. And the function returns these values.

Here is the center of mass values for each character:

Character	Center of Mass
Master Yoda	67 - 55
Darth Maul	66 - 170
Boba Fett	65 - 265

6.7 `draw_crosshair`

Draws a crosshair at the center of mass. Takes three arguments. These are '`buffer` address', 'x coordinate of the center of mass' and, 'y coordinate of the center of mass'. Iterates the image with the given `buffer` address and finds the pixels around the center of mass. Changes the RGB values for each pixel in order to draw a pure red crosshair at the center of mass. It does not return anything since it's a void function.

7 Output

Here is a sample output for Master Yoda: