ANTONIO SOUTO RODRIGUEZ

Programação Paralela

IMPACT LAB 2024

Antonio Souto Rodriguez
antonio.rodriguez@icomp.ufam.edu.br

Junhe
2024

PAVIC - LAB - UFAC

PAVIC LAB 2024

Images

Aula 01

Antonio Souto Rodriguez
antonio.rodriguez@icomp.ufam.edu.br

Outubro
2024

PAVIC - LAB - UFAC

Programação Paralela:

Processamento de Imagem : Images Filters

- **Exposure and Contrast**
  a. Contrast
  b. Brightness
  c. Dynamic Range (HDR)
  d. Motion Blur
  e. Rolling Shutter: Motion
  f. Rolling Shutter: Partial Exposure
- **Color**
  a. Intensity
  b. Color Cast
  c. White Balance
  d. Posterization
- **Focus**
  a. Front Camera Fixed Focus
  b. Focus

Prof: Antonio Souto Rodriguez

Outubro
2024

IMAGES

Images Filters

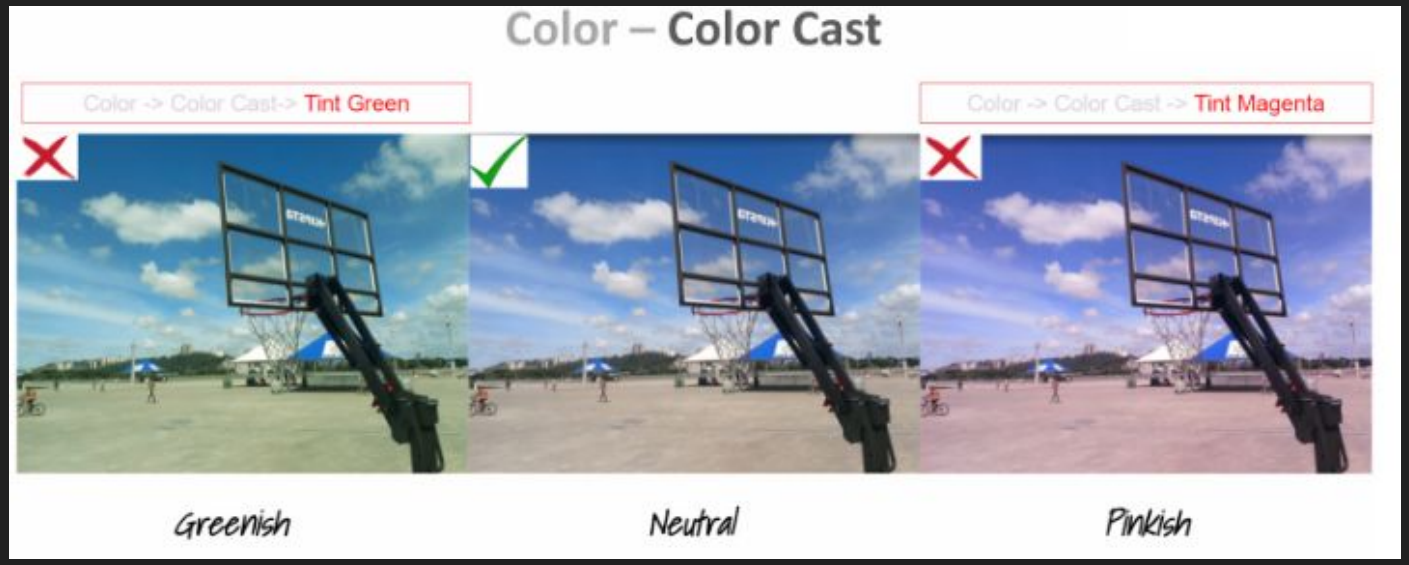
1. **Exposure and Contrast**
   a. Contrast
   b. Brightness
   c. Dynamic Range (HDR)
   d. Motion Blur
   e. Rolling Shutter: Motion
   f. Rolling Shutter: Partial Exposure
2. **Color**
   a. Intensity
   b. Color Cast
   c. White Balance
   d. Chromatic Aberration
   e. Posterization
3. **Focus**
   a. Front Camera Fixed Focus
   b. Focus

IMAGES

1. **Color**
   a. Intensity
   b. Color Cast
   c. White Balance
   d. Chromatic Aberration
   e. Posterization

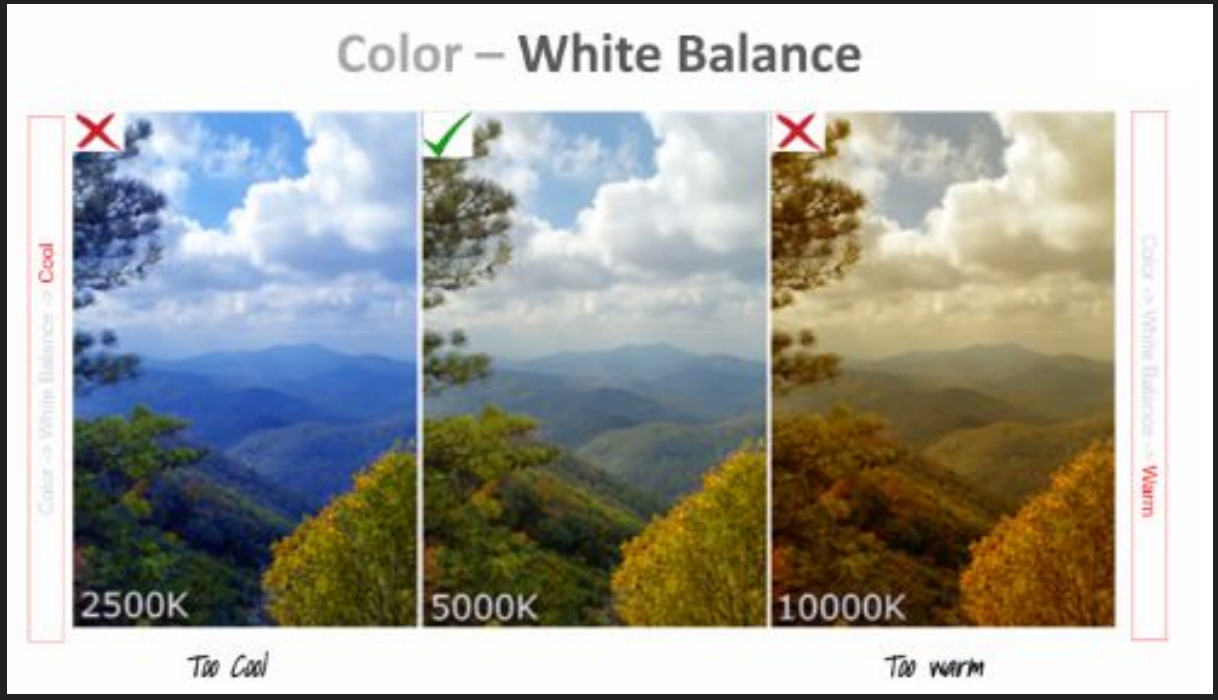
IMAGES

1. **Color**
   a. Intensity
   b. Color Cast
   c. White Balance
   d. Chromatic Aberration
   e. Posterization



IMAGES

1. **Color**
   a. Intensity
   b. Color Cast
   c. White Balance
   d. Chromatic Aberration
   e. Posterization



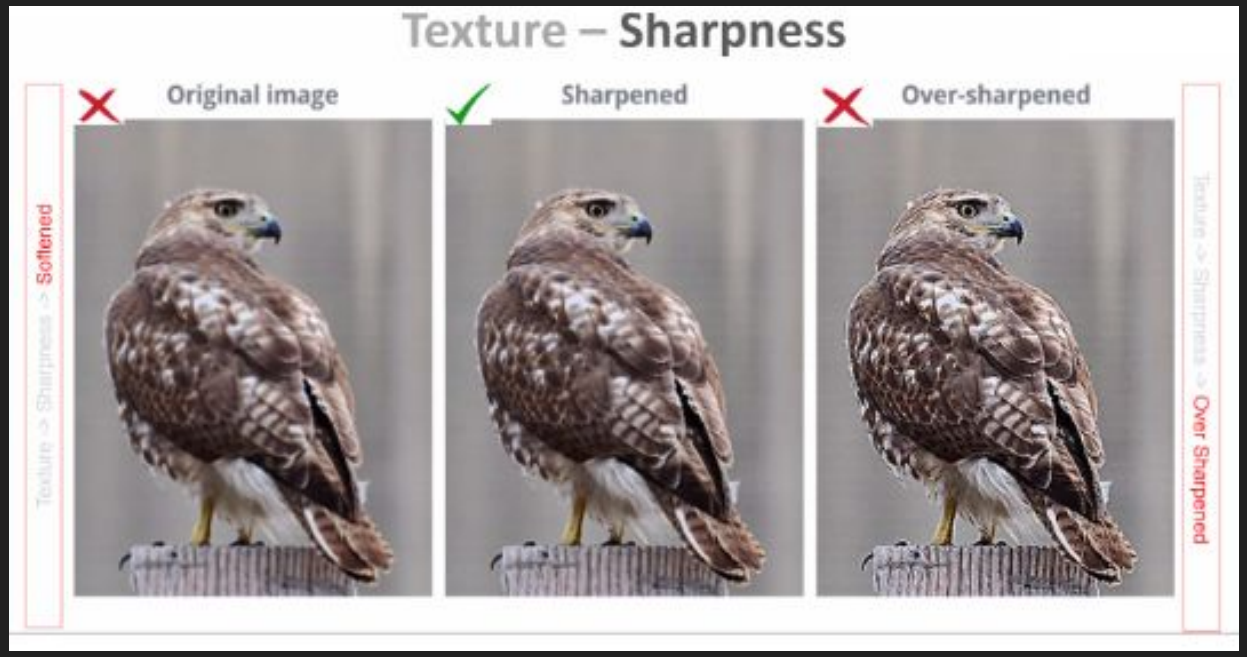
IMAGES

Images

1. **Texture**
   a. Sharpness
2. **Noise**
   a. Noise
3. **Articats**
   a. HDR Halo & Ghosting
   b. Lens Distortion
   c. Vignetting
   d. Flame
   e. Maze Artifact
   f. Compression Artifact
   g. Red/White Eyes
   h. Moire Artifact
   i. Spots. Dots, Circles Artifacts
   j. Blooming Smear

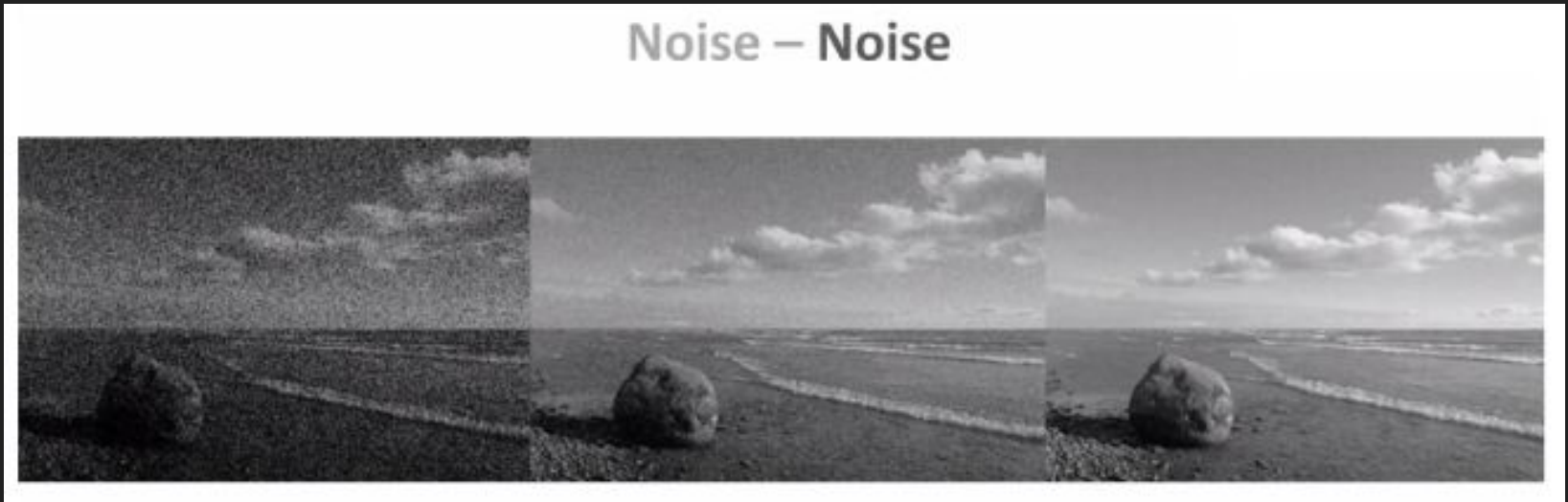
IMAGES

1. **Texture**
   a. Sharpness



IMAGES

1. **Noise**
   a. Noise



IMAGES

1. **Flash**
   a. LED
   b. Lens
2. **Zoom**
3. **Bokeh**
   a. Segmentation Errors
   b. Joggies
   c. Uneven Blur
   d. Incorrect Depth

IMAGES

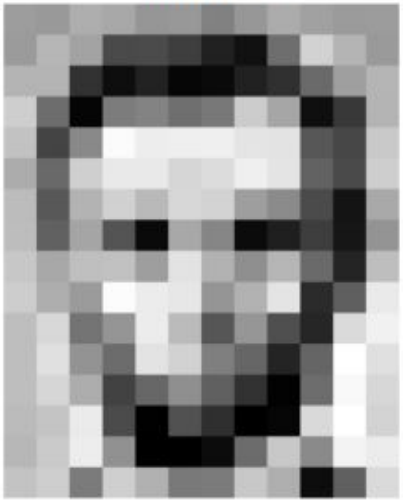
1. **Flash**
   a. LED
   b. Lens

IMAGES

Format Image

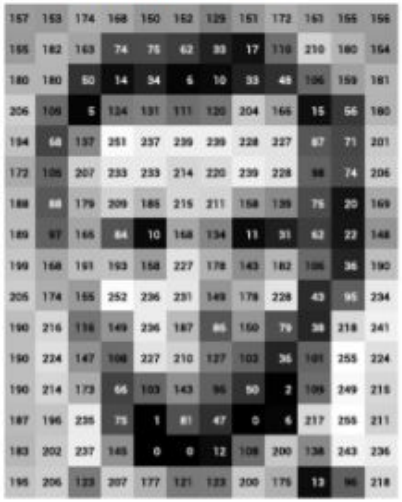
What computers 'see': Images as Numbers

What you see



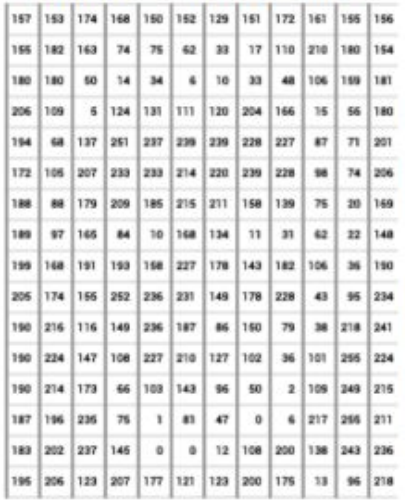
Input Image

What you both see



Input Image + values

What the computer "sees"



Pixel intensity values
("pix-el"=picture-element)

Levin *Image Processing & Computer Vision*

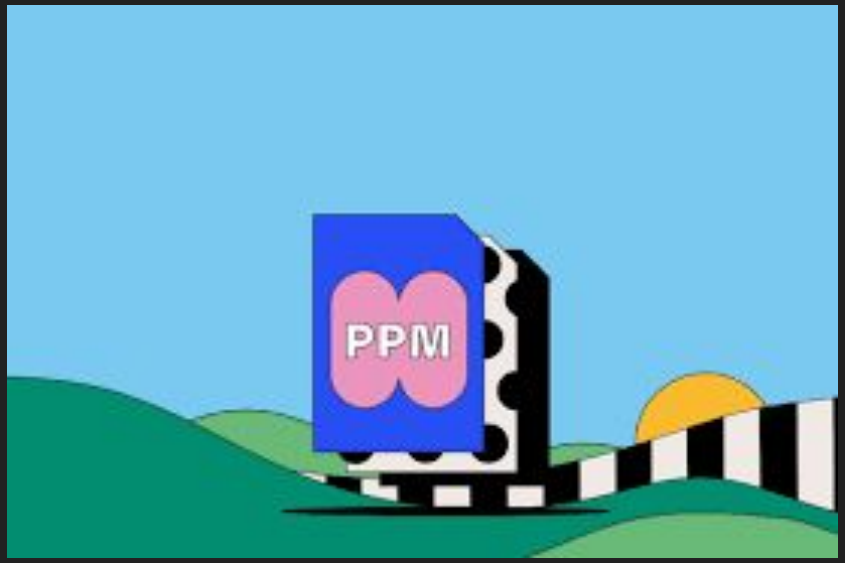
An image is just a matrix of numbers [0,255]. i.e., 1080x1080x3 for an RGB image.
Question: is this Lincoln? Washington? Jefferson? Obama?
How can the computer answer this question?

Can I just do classification on the 1,166400-long image vector directly?

No. Instead: exploit image spatial structure. Learn patches. Build them up

IMAGES

Format Image PPM??

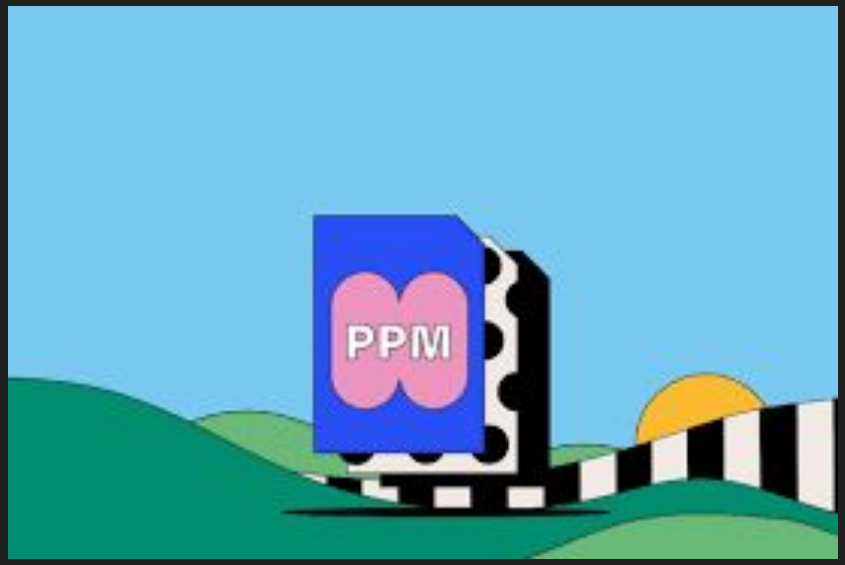


http://netpbm.sourceforge.net/doc/ppm.html

https://www.youtube.com/watch?v=HGHbcRscFsg

IMAGES

Image PPM:
Portable Pixel Map



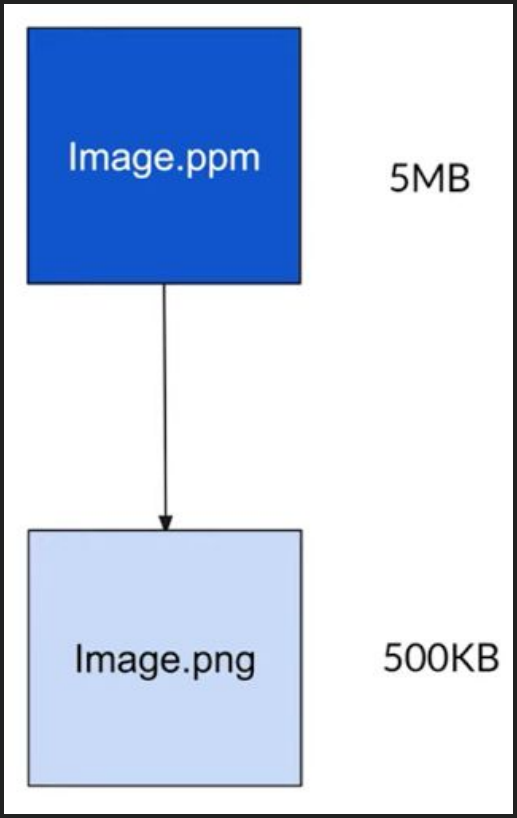

```
P3
# feep.ppm
8 8
255
255 255 0   255 255 0   255 255 0   255 255 0   255 255 0   255 255 0
255 255 0   255 255 0
255 255 0   255 255 0   255 255 0   255 255 0   255 255 0   255 255 0
255 255 0   255 255 0
0   0   0     0   0   0     0   0   0     255 0 255   0   0   0     0   0   0
0   0   0     255   0 255
0   0   0     0 255 127   0   0   0     0   0   0     0   0   0     0 255 127
0   0   0     0   0   0
0   0   0     0   0   0     0 255 127   0   0   0     0   0   0     0   0   0
0 255 127   0   0   0
255   0 255   0   0   0     0   0   0     0   0   0     255   0 255   0   0   0
0   0   0     0   0   0
255 255 0   255 255 0   255 255 0   255 255 0   255 255 0   255 255 0
255 255 0   255 255 0
255 255 0   255 255 0   255 255 0   255 255 0   255 255 0   255 255 0
255 255 0   255 255 0
```

https://www.youtube.com/watch?v=HGHbcRscFsg

IMAGES

Image Research

Image PPM:



Lossless Compression: Compression that can be reversed with no data loss.

Lossy Compression: Compression that loses data when uncompressed. Many times its data that didn't matter.

Raw Images: Pixel RGBa or RGB values.

External Libraries would be needed to handle PNG images directly.

IMAGES

Image Research

Raw Image PPM:

Type	Magic number		Extension	Colors
	ASCII (plain)	Binary (raw)		
Portable BitMap	P1	P4	.pbm	0–1 (white & black)
Portable GrayMap	P2	P5	.pgm	0–255 (gray scale), 0–65535 (gray scale), variable, black-to-white range
Portable PixMap	P3	P6	.ppm	16 777 216 (0–255 for each RGB channel), some support for 0-65535 per channel

PPM example [edit]

This is an example of a color RGB image stored in PPM format. There is a newline character at the end of each line.

```
P3
3 2
255
# The part above is the header
# "P3" means this is a RGB color image in ASCII
# "3 2" is the width and height of the image in pixels
# "255" is the maximum value for each color
# The part below is image data: RGB triplets
255   0   0  # red
  0 255   0  # green
  0   0 255  # blue
255 255   0  # yellow
255 255 255  # white
  0   0   0  # black
```

Image (magnified)

IMAGES

Image Research

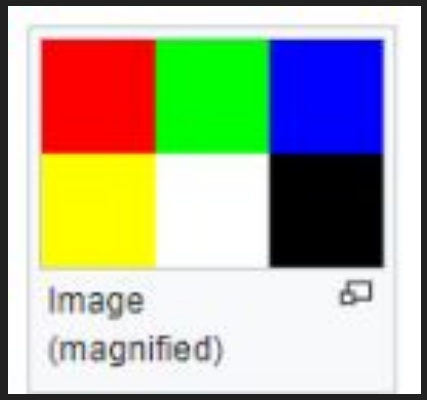
Raw Image PPM:

Type	Magic number		Extension	Colors
	ASCII (plain)	Binary (raw)		
Portable BitMap	P1	P4	.pbm	0–1 (white & black)
Portable GrayMap	P2	P5	.pgm	0–255 (gray scale), 0–65535 (gray scale), variable, black-to-white range
Portable PixMap	P3	P6	.ppm	16 777 216 (0–255 for each RGB channel), some support for 0-65535 per channel

PPM example [edit]

This is an example of a color RGB image stored in PPM format. There is a newline character at the end of each line.

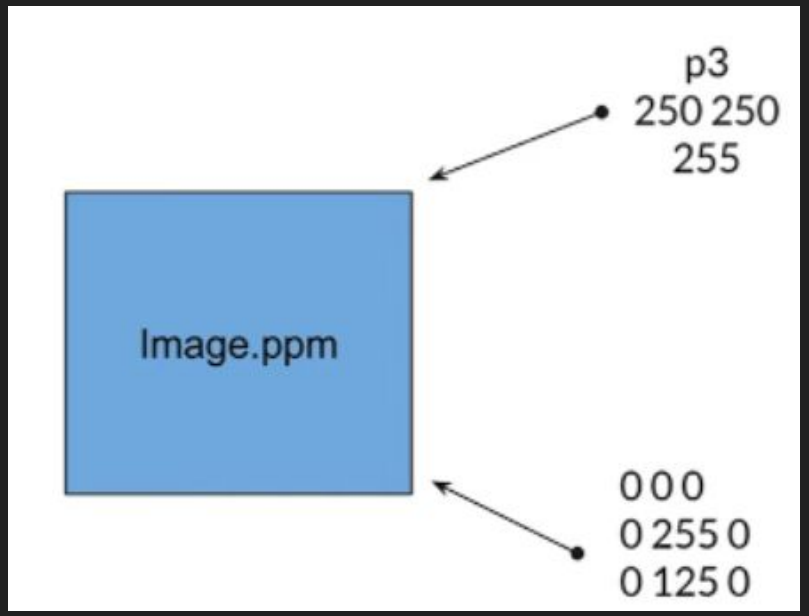
```
P3           # "P3" means this is a RGB color image in ASCII
3 2          # "3 2" is the width and height of the image in pixels
255          # "255" is the maximum value for each color
# The part above is the header
# The part below is the image data: RGB triplets
255   0   0  # red
  0 255   0  # green
  0   0 255  # blue
255 255   0  # yellow
255 255 255  # white
  0   0   0  # black
```



IMAGES

Image Research

Raw Image PPM:



Header: Each image data starts with header information that describes basic info about the image and its type.

For PPM:
type
width height
max RGB value

Body: All the pixel data that describes the image.

https://www.cs.rhodes.edu/welshc/COMP141_F16/ppmReader.html

IMAGES

Image Processing with C++

PPM Images Download

https://github.com/ferrabacus/p3images

```
1   #include <iostream>
2   #include <fstream>
3
4   using namespace std;
5
6   int main() {
7
8
9
10      return 0;
11  }
```

C:\Users\ansor\source\repos\Impact-Lab-2024\Computação_Heterogênea_2024\Impact-Lab_Computação_Heterogênea_2024

https://www.youtube.com/watch?v=028GNYC32Rg&list=PLG5M8QIx5IkzdGkdYQeeCK__As6sI2tOY

IMAGES

Image Processing with C++

```cpp
#include <iostream>
#include<fstream> // Read and Write Images

using namespace std;


// Images Processing C++

//Image PPM


int main() {
    // Create a images
    ofstream image;

    image.open("PPM_Images.ppm");

    if (image.is_open()) {
        // Place header info
        image << "P3" << endl;
        image << "250 250" << endl; // Image Sizes
        image << "255" << endl; // Set RGB max
    }
    // close image
    image.close();

    return 0;


}
```

```cpp
#include <iostream>
#include<fstream> // Read and Write Images

using namespace std;

// Images Processing C++

//Image PPM

int main() {
    // Create a images
    ofstream image;

    image.open("Images\\PPM_Images_01.ppm");

    if (image.is_open()) {
        // Place header info
        image << "P3" << endl;
        image << "250 250" << endl; // Image Sizes
        image << "255" << endl; // Set RGB max

        // Across the images
        for (int y = 0; y < 250; y++) {
            for (int x = 0; x < 250; x++) {
                image << x << " " << x << " " << x << endl;
            }
        }
    }
    // close image
    image.close();

    return 0;
```

C:\Users\ansor\source\repos\Impact-Lab-2024\Computação_Heterogênea_2024\Impact-Lab_Computação_Heterogênea_2024

IMAGES

Image Research

PPM Images Download

https://github.com/ferrabacus/p3images

```
srand(time(0));
if (image.is_open()) {
    // place Header info
    image << "P3" << endl;
    image << "250 250" << endl; // Image Size
    image << "255" << endl;
    for (int y =0; y<250;y++)
        for (int x = 0; x < 250; x++) {
            //        Red          Green        Blue
            //image << x << " " << x << " " << x << endl; // black to White
            image << (x*y)% 255 << " " << x << " " << x << endl;
        }
}
image.close();
```

PAVIC - LAB - UFAC

PAVIC LAB 2024

Images

Aula 02

Antonio Souto Rodriguez
antonio.rodriguez@icomp.ufam.edu.br

Outubro
2024

IMAGES

Image Processing with C++

```cpp
#include <iostream>
#include<fstream> // Read and Write Images

using namespace std;


// Images Processing C++

//Image PPM


int main() {
    // Create a images
    ofstream image;

    image.open("PPM_Images.ppm");

    if (image.is_open()) {
        // Place header info
        image << "P3" << endl;
        image << "250 250" << endl; // Image Sizes
        image << "255" << endl; // Set RGB max
    }
    // close image
    image.close();

    return 0;


}
```

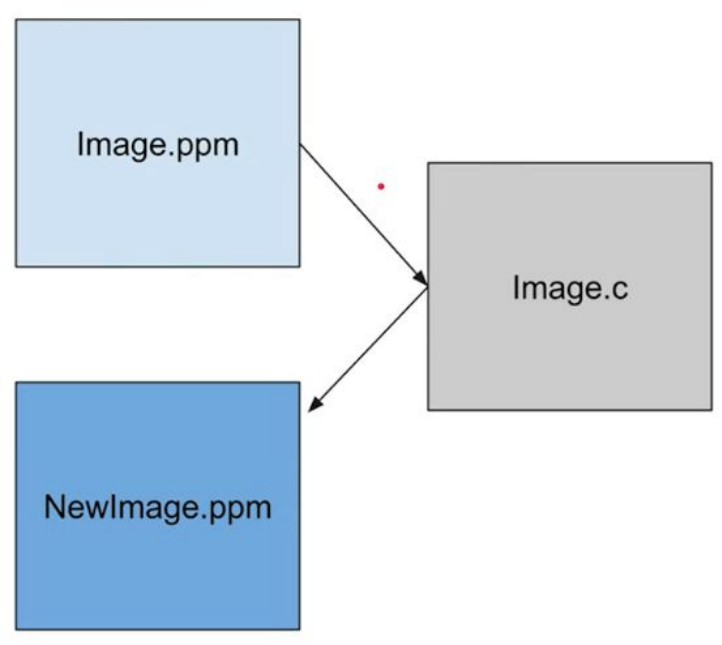
```cpp
#include <iostream>
#include<fstream> // Read and Write Images

using namespace std;

// Images Processing C++

//Image PPM

int main() {
    // Create a images
    ofstream image;

    image.open("Images\\PPM_Images_01.ppm");

    if (image.is_open()) {
        // Place header info
        image << "P3" << endl;
        image << "250 250" << endl; // Image Sizes
        image << "255" << endl; // Set RGB max

        // Across the images
        for (int y = 0; y < 250; y++) {
            for (int x = 0; x < 250; x++) {
                image << x << " " << x << " " << x << endl;
            }
        }
    }
    // close image
    image.close();

    return 0;
```

C:\Users\ansor\source\repos\Impact-Lab-2024\Computação_Heterogênea_2024\Impact-Lab_Computação_Heterogênea_2024

IMAGES

Image Research

PPM Image - Applying filters to Images



I recommend reading from one file and then writing it to another file. It simplifies the process a bit.

First we are simply going to read over the header information.

Then we will apply our blue filter by reading an entire RGB value at a time, converting those strings to numbers, adding more blue, saving it to the new file.

IMAGES

Image Processing with C++

```
/*
APPLYING A FILTERS TO IMAGES - BLUE FILTER
*/
#include <iostream>
#include<fstream> // Read and Write Images
#include<ctime>
#include<cstdlib>
#include<sstream>
#include "../Images/codeTimer.h"
#include"../Images/codeTimer.h"

using namespace std;
// PPM Images Processing C++

int main() { ... }
```

```
int main() {
    //Read images
    ifstream image;
    // Write a images
    ofstream newimage;

    image.open("./images/Monument.ppm");
    newimage.open("./images/newimage01.ppm");

    //copy over Header Information
    string type = "", width = "", height = "", RGB = "";
    image >> type;
    image >> width;
    image >> height;
    image >> RGB;
    //Copy Header to new Images
    newimage << type << endl;
    newimage << width << " " << height << endl;
    newimage << RGB << endl;

    // reading strings
    string red = "", green = "", blue = "";
    //int value
    int intRed = 0, intGreen = 0, intBlue = 0;

    // Read every Pixel
    { // Start timer
        Timer timer;
        while (!image.eof()) { ... }
    }
    newimage.close();

    cout << type << width << height << RGB << endl;

    return 0;
```

C:\Users\ansor\source\repos\Impact-Lab-2024\Computação_Heterogênea_2024\Impact-Lab_Computação_Heterogênea_2024

IMAGES

JPEG ?? -

- stb_image.h
- stb_image_write.h
- stb_image_resize.h

http://github.com/nothings/stb

```
//Load Images
#define STB_IMAGE_IMPLEMENTATION
// Write Images
#define STB_IMAGE_WRITE_IMPLEMENTATION

#include "include/stb_image.h"
//#include "stb/stb_image_resize.h"
#include "include/stb_image_write.h"
```

IMAGES

https://github.com/Code-Breako/Image-Processing

- stb_image.h
- stb_image_write.h
- stb_image_resize.h

http://github.com/nothings/stb

```
//Load Images
#define STB_IMAGE_IMPLEMENTATION
// Write Images
#define STB_IMAGE_WRITE_IMPLEMENTATION

#include "include/stb_image.h"
//#include "stb/stb_image_resize.h"
#include "include/stb_image_write.h"
```

IMAGES

JPEG ?? -

```
//Load Images
#define STB_IMAGE_IMPLEMENTATION
// Write Images
#define STB_IMAGE_WRITE_IMPLEMENTATION

#include "include/stb_image.h"
//#include "stb/stb_image_resize.h"
#include "include/stb_image_write.h"

//Open Image
        unsigned char* img = stbi_load(“input.jpg”, &width, &height, &channels, 4);

// Write Output Image
        stbi_write_png("output.jpg", width, height, 4, img, 4 * width);

// Free Image Memory
        stbi_image_free(img);
```

IMAGES

```
//Open Image
        int width, height, channels;
        unsigned char* img = stbi_load(c, &width, &height, &channels, 4);

// Check if IMage Open
        if (img == nullptr) {
                MessageBox::Show("Failed to open the image!", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);
                return;
        }

// Process the image to make it green
        for (int i = 0; i < width * height * 4; i += 4) {
                img[i + 1] = 255; // Set the green channel to 255
        }

// Save the output image
        stbi_write_png("output.png", width, height, 4, img, width * 4);
        Output_Image_Box_01->ImageLocation = "output.png";

// Free image memory
        stbi_image_free(img);


      // Converting color to grayscale

      Color to Grayscale Equation
      Ylinear = 0.2126RLinear + 0.7152GLiners +0.0722BLiners
```