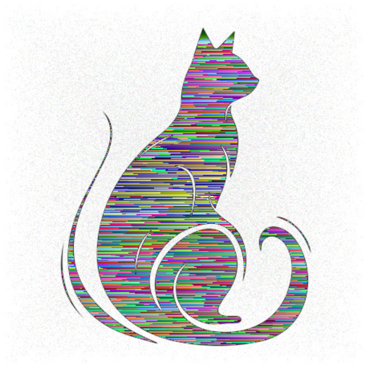

December 31, 2025

SORTING A CAT

Abusing ImageMagick, FFmpeg
and other shenanigans

"A wizard is never late, nor is he early, he arrives precisely when he means to."

J. R. R. Tolkien

Contents

1 What

This write-up will be introducing you to a couple of topics, to show how multiple parts where put together to fulfill a simple idea.

That Idea, "what would happen if I sorted and images pixels?"

This PoC (Proof of Concept) is the minimum viable product to get to that desired result. The main PoC centers around the PPM format[1] with a couple of helping programs to transform the images used and produced.

1.1 What is ImageMagick?

ImageMagick is a magical program that allows for more programmable ways of changing an image, from the terminal or an API.

This program will mainly be used for converting between image formats, because of the complexity of the PNG and JPG formats. This can be used for much more, however.

1.2 What is the PPM format?

The PPM (Portable Pixel Map) format is used in this project to store, the image data when not in program memory (or being used by the program). The format, being very easy and stored in ascii plan text (unless you go with one of the binary formats), allows for easy reading and creation.

An example of the format is as follows:[1][2]

```
P3
3 2
255
255    0    0
  0  255    0
  0    0  255
255  255    0
255  255  255
  0    0    0
```

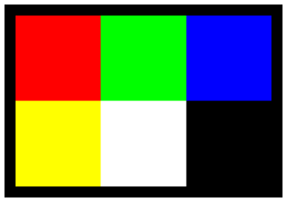

Figure 1: Output Example

The first line, "P3", specifies which format the file is, or the *"Magic Bytes"* (file signatures).[2]

The next line, "3 2", specifies the *width* and *height* of the image.

The third line, "255", specifies the maximum value for each color.[1]

[1]The PPM file format is defined by the Netpbm project, along with its cousins PBM (Portable Bitmap Format) and PGM (Portable Graymap Format).[1]

[2]You can use the column(1) command on linux to format the data into rows like the one shown. The specific command used was "column -tR0", which makes a table (-t) that has all rows aligned right (-R0).

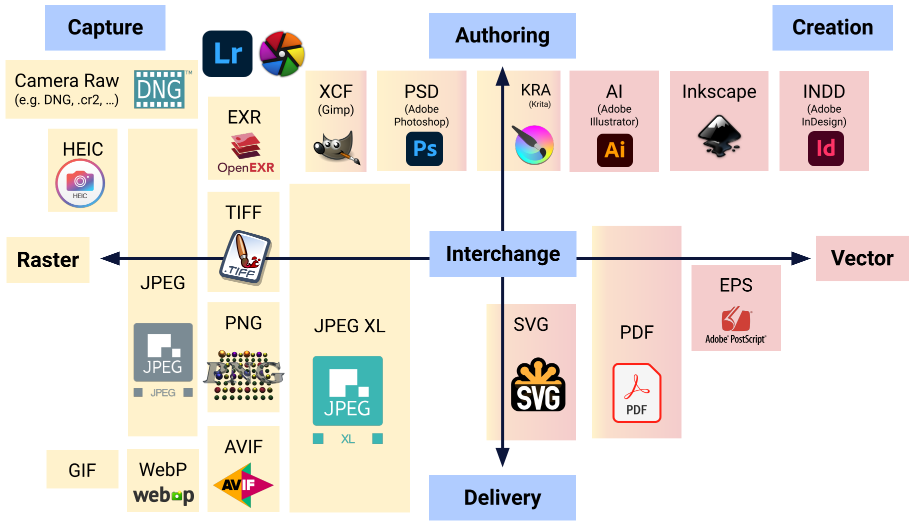

Figure 2: Image formats by scope

The next line, "255 0 0", species the Red, Green, Blue, values for that pixel.

This relative simplicity being a great plus, the storage size of the images is dreadfully inefficient (although this is to some part addressed with P6). This format requires at most nine bytes for 3 bytes of pixel color data.

To fix this there is another format, a brother to P3 PPM, P6 PPM, which stores the color data as direct binary. This allows for a 1:1 conversion ratio, but still pretty bad in comparision with JPEG(Joint Photographic Experts Group), PNG(Portable Network Graphics), or even WebP (each with a different use).

1.3 What is FFmpeg?

FFmpeg is a great tool for anything video or audio (but mainly video) related. Many applications use it under the hood because of its ability to transcode many video types.

For this simple project, we will use FFmpeg to compile a bunch of images together into a video, and even a Gif.

1.4 The Theory

The process of doing a sufficently big task should be seperated into multiple discreate operations. In this vain, this can be broken down into multiple steps, and two paths, creating the program, and manipulating the input/output files.

1.4.1 Creation

To create such a program, you must of corse have some knowledge on the creation of programs, that being out of scope for this, (you could view my code).

The program will need to take the PPM file in, turn it into usable data, be able to process the data, then export the finished file.

The import and export just follow the image format described above. For processing, we need to be able to sort a very large amount of data, which although may seem simple, images have a lot of pixels.

So we need sort the file a little faster. For this I used an algorithm called quick sort, which, as the name suggests, is a quick sorting algoritm. There are other algoritm's that are faster, but they may require more restrictions, and/or may be parallel operations (this means that multiple operations are happening simultaiusly).

You can role your own quick sort (meaning create your own), which allows you to modify it how you like, or you can use a premade function/library.[3]

There are also multiple ways you could sort it. For example, one may store the image in a single dimentional array (meaning the image is converted to a line). This means when sorting all the largest values will be pushed to the end of the array, which means that the bottom right – depending on how the file is imported and exported – will house the brightest values.[4]

Once you got it to sort the image, you can have it, if you made rolled your own sorting, export the images during the sorting to be able to make a movie of sorts out of the sorting process.

2 How

To be able to acomplish this inital idea, "sorting the pixels of an image", we need to put together these technologies. I will be using the programming language C,[5] but any programing language can be used, as long as it can process files, and minipulate arrays.

[3]An example of a function like this is the one that is included in "stdlib.h" or the standard C library, as "qsort()". `https://www.man7.org/linux/man-pages/man3/qsort.3.html`

[4]You can sort the image to other directions, but this requires some clever math, or cleaver importing and exporting of the images.

[5]This is of course an unusual choice, but because of my familiarity, but note that the original project was not memory safe, as it was a PoC. `https://www.open-std.org/jtc1/sc22/wg14/` `https://gcc.gnu.org/`

2.1 Running it

First we need to convert a image to the PPM format, and potentially resize[6] the image, using ImageMagic. You can do that with the following commands:

```
# Just replace yourimage with the desired file name
magick <yourimage> <yourimage>.ppm

# If you want/need to resize the image use the ‘resize‘ flag
magic <yourimage> -resize 50% <yourimage>.ppm
```

From here, just point your program to yourimage, and run. It may take a couple of seconds, and expecially if your image has a lot of same colors, or is pretty large (pretty much anything above 1000x1000 is large in this case).

Lastly, we take the images and combine them together, using FFmpeg. FFmpeg can recognize the file format, even though it is relativly unused, so no need to convert them into a more familiar format. You can make a video and a gif using FFmpeg, and the following commands:[7]

```
# To an mp4
# The image files should be named <image>0001.ppm, <image>0002.ppm, etc
#   For this to work.
ffmpeg -r <fps> -pattern_type glob -i '<image>*.ppm' -c:v libx265
    <outfile>.mp4
# ‘-r‘:        Specifies the Frames Per Second of the video
# ‘-pattern_type‘: Specifies the pattern matching of the file name.
# ‘glob‘:      Specifies the type of pattern matching[3]
# ‘-i‘:        Specifies the input type
# ‘-c:v‘:      Specifies the video codex as libx265


# To convert to gif
#   It is also possible to convert directly to gif, by changing the .mp4
    of
#   the previous command.
ffmpeg -i <imagefile>.mp4 <outputfile>.gif
```

[6]To resize an image using ImageMagic use the ‘-resize‘ flag, between the input file, and the output file. There are more options with this flag, for more information you can view the following: https://imagemagick.org/script/command-line-options.php#resize

[7]For the pattern matching there is technically more options to combine them together. Using the ‘-pattern_type glob‘ was just the one that worked the most consistently for me. https://shotstack.io/learn/use-ffmpeg-to-convert-images-to-video/

“Talk is cheap, show me the code.”

Linus Torvalds

2.2 The Code

The code is a little long, and morphed some into another project, but the basic program can be found here:
https://gist.github.com/SeptembersEND/cb529f75dea5a970f557b269538d8434

That simple PoC turned into a child project:
https://github.com/SeptembersEND/sortcat.

3 Conclusion

You are able to, with this base project, to explore many other image manipulation techniques, like edge detection, the Conway’s Game of Life, Bluring...

You could also venture into creating images, or perhaps creating a 3D abstraction layer. You could even start trying to write to the screen, or use a graphics library. There are many avenues one can take through this.

Here are some sources if you want to learn more:

3b1b	—	site	*What is a convolution?*
Wikipedia	—	site	*Conway’s Game of Life*
Wikipedia	—	site	*Edge Detection*
Wikipedia	—	site	*Gaussian Blur*
Antoine Lelievre	—	site	*Introduction to modern rendering*
Eric Arnebäck	—	site	*Beginner Computer Graphics*
Stefan Pijnacker	—	site	*Graphics Programing*

4 Sources

References

[1] Netpbm. (2022, June 26). Wikipedia. https://en.wikipedia.org/wiki/Netpbm

[2] Wikipedia Contributors. (2019, October 19). List of file signatures. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/List_of_file_signatures

[3] glob(7) - Linux manual page. (2025). Man7.org. https://www.man7.org/linux/man-pages/man7/glob.7.html

There are 2829 words in this document.