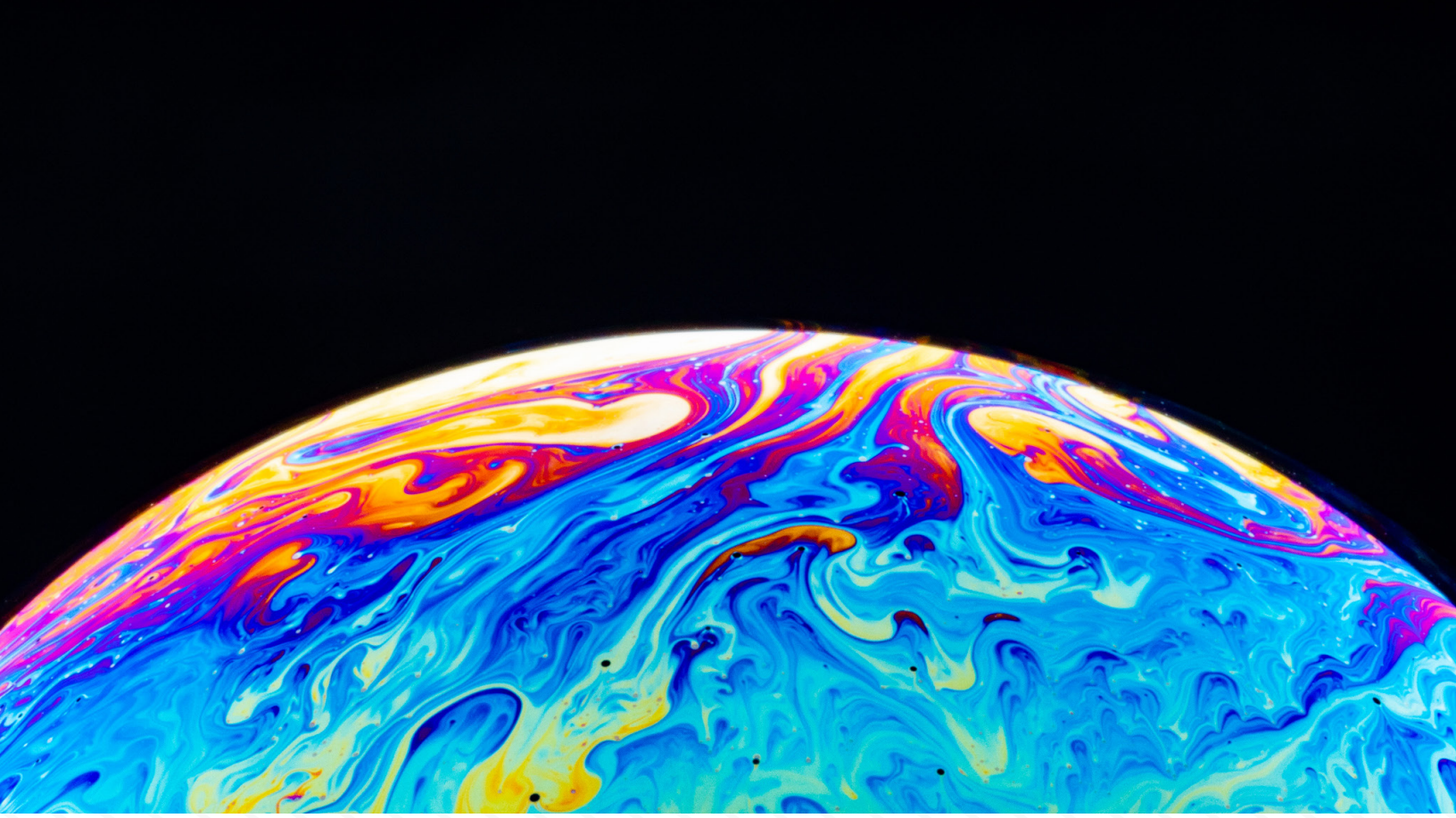
accusoft®

Imaging 'Quick Start' Guide

Colby Veal
Software Development Engineer in Test II



Imaging software is one of the core foundations of Accusoft. However, there can be some complex concepts and terms related to imaging that lead to bad practices or misleading expectations. This guide was created to get you up and running with some common knowledge, as well as give you some terms to research and things to keep in mind when you begin using our products.

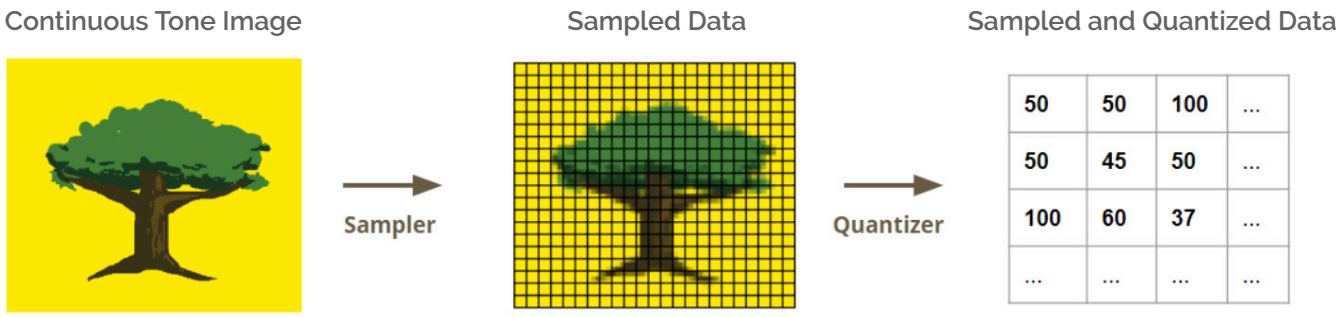
Two Formats for Representing Image Data

There are two different ways to represent image data on a screen. These are Raster and Vector format, and each type has its advantages and disadvantages that should be understood before working with them. It is important to note that they are each independent of each other and, although you can convert from one format to another, you cannot make any assumptions about a raster image based solely upon its vector counterpart, and vice versa. With that being said, we'll be focusing on Raster format for this post.

Raster Format

Raster, in its simplest form, is nothing more than an array of pixels organized into a grid, or matrix, where each cell contains a value representing information, with color data being the most common. These grids, along with the data with-in each 'cell' (called a pixel) come together to form the images we see on our screens. Some of the most common file types for Raster Data are JPEG, JPEG2K, Exif, TIFF, GIF, BMP, and PNG.

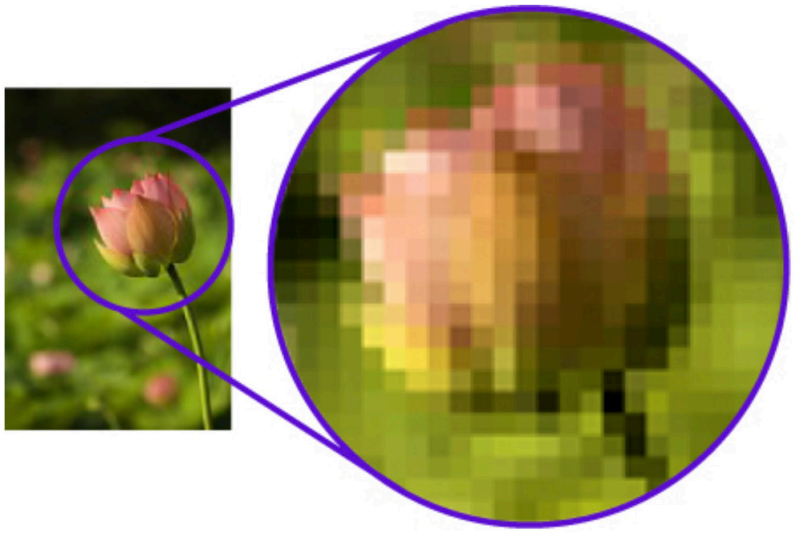
To get an image to display digitally, it must go through a process called sampling and quantization. Sampling takes the continuous image and breaks it into a matrix. Quantization then takes this grid and places a 'quantity', or numeric value, into each of the cells that represents the color to be displayed there.



Through this process, the computer is then able to interpret the data and display it accurately on a screen.

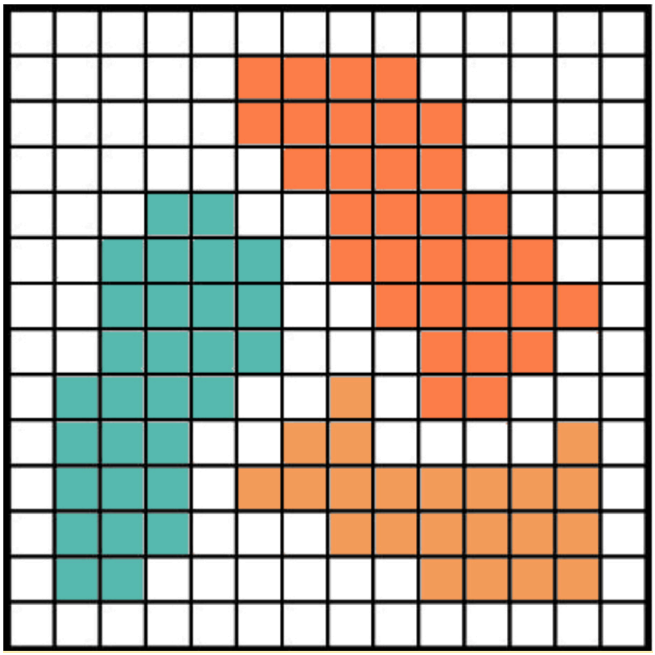
The Pixel

To understand raster data, it's fundamental to understand the pixel, similar to how biologists must understand the cell to understand whole organisms. A pixel (short for picture element) is an individual cell of a matrix that is mapped to a segment of the screen. This matrix represents our raster image, as shown below:



Source

You can see there are many individual squares that are one color only. Each square is a pixel, holding a numerical value representing a color value. Each pixel has an address, and you can access them in order to read or even modify the data stored there. Each Pixel has an 'address' that is its location in the Matrix.



As you can see, the grid starts with the top-left most pixel as the origin, and the bottom right most location as (width-1, height-1) location in a zero-indexed array. Different file formats will store this data slightly differently, and usually include some metadata and header information along with the pixel data for context.

Color Depth

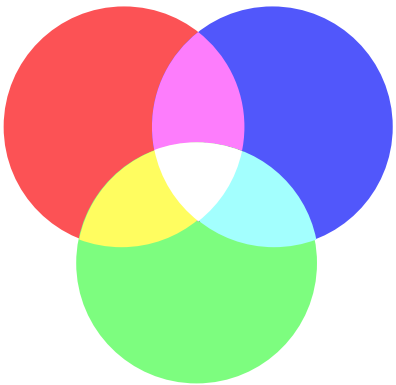
Now, you may be wondering, "How much data can be stored with-in a single pixel?". It depends upon its color-depth or bit-depth, which is measured in bpp, or bits per pixel. The common values for bit depth, and the number of colors each can represent, are listed below:

- 1bpp = 21 = 2 colors (Black and White)
- 2bpp = 22 = 4 colors
- 4bpp = 24 = 16 colors
- 8bpp = 28 = 256 colors (Grayscale OR Limited Color)
- 16bpp = 216 = 65536 colors (R/G/B Color)
- 24bpp = 224 = 16777216 colors (R/G/B Color)

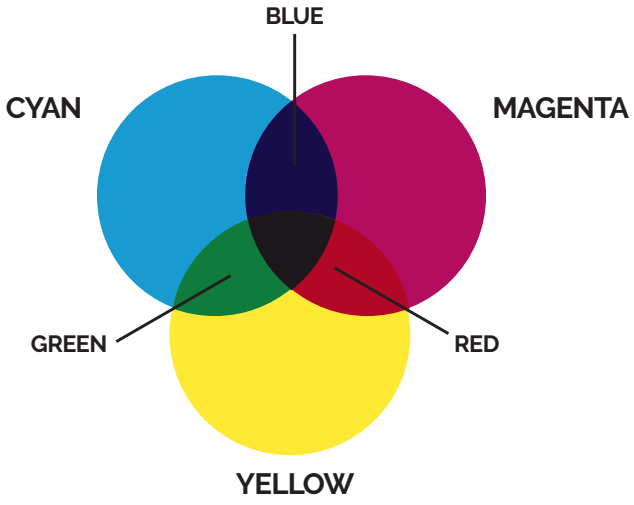
...and so on!

Bits per Pixel refers to the amount of data stored within each pixel. This is generally used when referencing color data. The overwhelmingly most common are 1 (for black and white) and 24 (for color images). With 1- bit images, you're allowed to represent two (2) unique colors, with those two colors represented by 'flipping the bit' to switch between black and white. Most forms used in the business world use this bit depth. This will result in a smaller file size, but you will not be able to represent any complex color data with such a limited bpp.

To represent color, the most common format for digital images is 24 bpp, usually associated with the RGB (Red/Green/Blue) color space, also known as the Additive Color Model. This allocates 1 byte to each color (or channel) and thus allows for ~16 million unique colors. It creates all of these colors by using Red/Green/Blue and combining those colors using various ratios to create unique ones. For printed images it is common to see 32 bpp used to represent color in the form of CMYK (Cyan/Magenta/Yellow/Key (Black)). This model is subtractive, as instead of adding colors together to create it's ~4 million unique colors, it subtracts colors from other colors. K, or black, is the subtraction of all of these colors, and is why it is used for printing. These two color models are opposites of each other, as you can see here:



RGB Color Cube
(Additive)



CMY Color
(Subtractive)

32 bpp can also be used to represent an RGB image that has an 8-bit alpha channel appended to it, referred to as RGBA. This alpha channel stores the 'transparency' or 'opacity' of the image.

Resolution

Now that we know what type of data is stored in each pixel and how much data there can be in each, let's discuss - how many pixels can be used to represent an image? The answer resides in an image's resolution. This is a value that is usually given in a width x height format with the value corresponding to *Pixels per Inch* or *Dots per Inch* (for the purpose of this post, these two phrases may be treated as equivalent). This value represents how many pixels you want to use to display the image data on the screen, or how many dots you want to represent the image data when printed. Generally, the higher the DPI, assuming the physical size of the image stays constant, the higher quality the image, especially when printed.

Source

As you can see in the above picture, the image size stays the same but the amount of 'dots' used to represent it increases, creating a higher quality image. But, to further prove this, let's look at what happens when one of the variables changes.

300dpi example

72dpi example

Source

In this example, we see the image size has remained constant, but less dots are being used to represent the image. This, as expected, has resulted in less definition between the changing colors and edges, and thus a less sharp image. Let's look at another example, this time where the image size changes but the number of dots stays constant.

Image printed at 4x6

Image printed at 20x30

This resulted in a similar loss of quality and is usually referred to as an image appearing 'pixelated'.

In the first case, the loss of quality was a result of the amount of data used to represent the image decreasing. In the latter case, the area with which the same amount of data had to represent was increased, thus stretching the pixels across a larger area. It's important to note that it is possible to decrease in size and maintain quality, but **increasing** in size causes a loss in quality.

DPI can range from 0 to as high as you'd like, but it's important to balance the quality of the image with the ensuing file size that quality will create, as we'll get into next. At Accusoft, the recommended DPI for scanned images when performing operations such as OCR or Barcode Recognition is 300.

The Big Picture

Resolution and Bit Depth have a large impact on the quality of an image, and the quality of the image generally determines the file size of the image. The higher the image DPI and bpp, the larger the file size will be (in bytes). However, these numbers are only part of the equation. You need to know the physical dimensions of the file (in inches) in order to be able to calculate the total file size. Here's the equation you'd use:

$$((DPI*width\_in)*(DPI*height\_in))*bpp/8$$

Breaking this equation down into simpler terms, you take the total number of pixels and multiply by the number of bytes stored in each pixel to get the file size of an image. Let's run through an example.

Let's say we have "Letter" size image, which is 8.5" x 11", 300 DPI and 24 bpp. Here's how we'd begin:

$$(300 * 8.5) * (300 * 11) = 2550 * 3300 = 8{,}415{,}000 \text{ bytes}$$

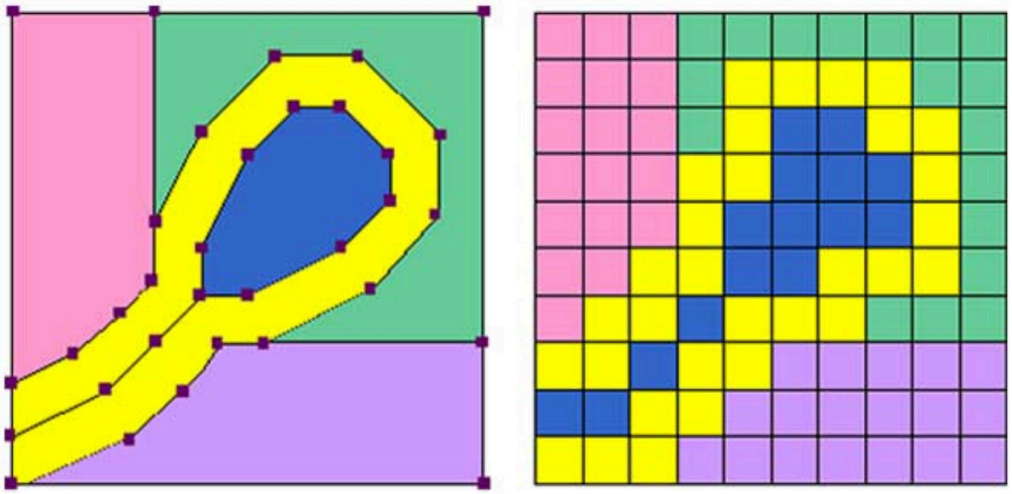
However, this number has an assumption of 1-byte depth. This is a bit of a confusing term (no pun intended), so I'll elaborate a bit more (see previous comment). This calculation implicitly assumes 8 bpp (bits per pixel). Another way of saying that is 1 BPP (Byte Per Pixel). However, for our particular image has 24 bpp, or 3 BPP (Bytes Per Pixel). So, from here we get:

8,415,000(24/8)=25,245,000 bytes OR approx. 25 MB*

You may see this image compressed in some way, such as JPEG or CCIT, greatly reducing that size on disk. The 25 MB is the size the image takes up in memory, uncompressed. So, be sure you're using the correct combination needed for your images to stay away from 'hard drive space creep'.

A Note on Vector Format

Most of the information presented here concerns the raster format, but I'd be remiss to not touch on Vector Format here as well. While Raster images use a matrix to represent its data, Vector images use lines to draw its data. Because of this, vector images are scalable in size without losing quality and generally take less storage - but not always.



Vector **Raster** Source

As you can see, the two are fundamentally different. It is possible to convert between the two via processes called rasterization, converting from vector to raster, and vectorization, converting from raster to vector. When doing so, no assumptions should be made in regards to the output image based upon the input image, such as resultant file size after conversion. Fun fact:, Vector format images can also store entire raster images inside of them, and can even have both vector and raster data on the same page simultaneously.

Imaging at Accusoft

Accusoft has many tools that make performing imaging operations relatively simple. ImageGear is the most robust option, as it runs the gambit of imaging operations and accompanying features, such as OCR. ImageGear also provides options for working with Vector format files, like PDF and SVG, and even Microsoft Office formats. ImageGear comes in a variety of platforms, like C#, Java, or C++, and runs on Windows and Linux.

For someone in need of something more streamlined and lightweight, ImagXpress may be the better solution. ImagXpress is a .NET Framework solution and concerns itself only with raster data. If you have additional needs, like OCR, barcode scanning, or PDF support, we have several products that have individual modules built for those purposes. They all work together seamlessly to give your application the support it needs.

PICTools is a more developer-centric approach to imaging, allowing you to integrate even the most minute details of compression into your application It is available in many versions and runs on many different OS and CPU architectures.

Hopefully this helped kick start your journey into imaging, and maybe Accusoft can accompany you along the way!