Class 408 & 428

Fundamentals of Digital Imaging

John Canosa
jcanosa@questra.com

Introduction

More and more embedded devices contain an imaging component. Beyond the obvious digital cameras and image capture enabled devices such as cell phones and PDAs, digital imaging knowledge is also important for devices with graphical displays and hard copy output capability. However, digital imaging covers more that just capturing an image and printing it. As shown in Figure 1, digital dmaging systems can cross many boundaries from image capture and processing to compression and storage, image transfer, image recognition and understanding as well as both hard and soft displays.

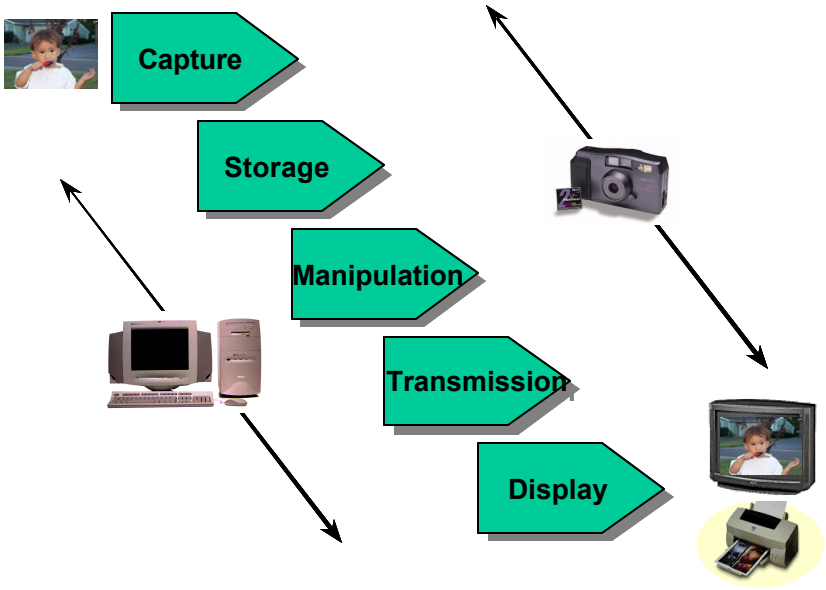

Figure 1 Imaging Chains

From a software developers point of view, dealing with the world of 2 dimensional images have some very interesting challenges that one does not encounter when dealing with the standard world of linear memory systems and signals. This paper gives a high level overview of digital imaging systems and gives some insight into the image chain associated with a basic digital camera and printer system.

Digital Camera Processing Pipeline

In order to understand the individual components of an imaging system, we should take a look at a typical implementation at the system level. The diagram in Figure 2 shows a data processing pipeline for a typical digital camera.

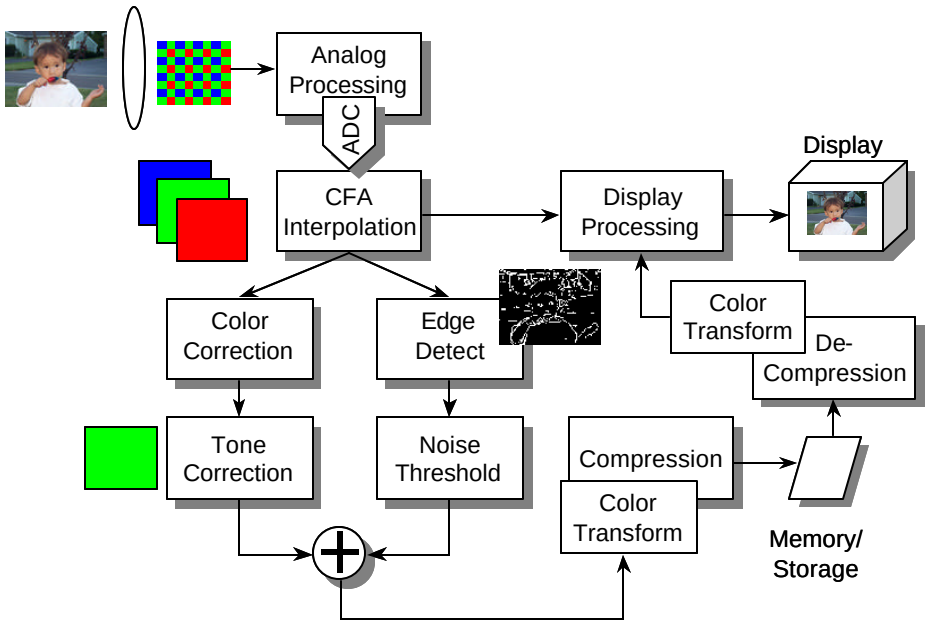

Figure 2 Camera Processing Pipeline

Sensors

As we can see from the figure it all starts with the image sensor. An image sensor is essentially a two dimensional array of light sensing picture elements (Pixels). Each pixel is exposed to the scene that we want to capture for a small period of time, typically in the several milliseconds range. The output of each pixel, which is an analog voltage is then digitized into an 8 to 12 bit value which is stored in memory. Today, images sensors in cameras are usually one of two types, a Charge Coupled Device (CCD) or a Complementary Metal Oxide Semiconductor (CMOS) device. Currently CCDs are recognized for their better image quality, but are difficult to interface with and can be quite power hungry . CMOS sensors have the distinct advantage of being built with the same semiconductor processes used to make logic and microprocessors – leading to some very highly integrated solutions with simple digital interfaces.

Creating Color

Silicon does not recognize color with the exception that it is more sensitive to energy at certain wavelengths within the electromagnetic spectrum. In order to discern different colors, sensor manufacturers place a series of color filters on individual pixels of a sensor. Commonly color filters of Red, Green and Blue (RGB) are applied to a sensor array. The manufacturers also tend to make use of the fact that the human eye is most sensitive to images in the green part of the spectrum – hence our ability to resolve differences in luminosity is concentrated in the green. A common filter array, called a Bayer pattern, takes advantage of this fact by providing twice as many green pixels as shown in the the leftmost part of the following figure.

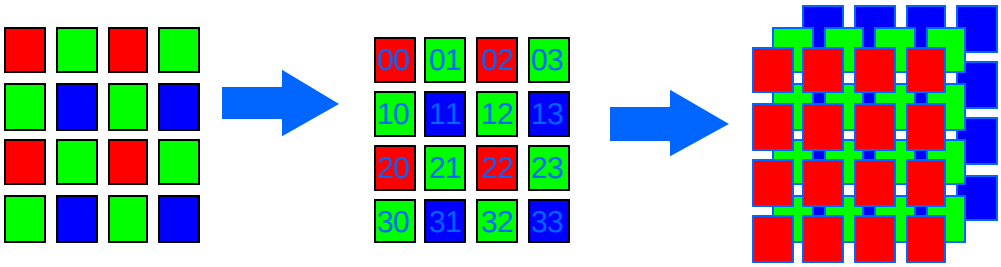

Figure 3 Bayer Pattern and Color Filter Array Interpolation

The dilemma for a system designer is that our images are displayed on a screen as triplets of RGB pixels, but we only have information for one color at each location in the array. For a 1024 x 1024 image sensor, we would only have 1,048,756 values of assorted colors. In order to create the RGB triplets need to display on a screen or send to a printer, we must create three complete planes of 1024x1024 pixels. One each for Red, Green and Blue. This is accomplished via a technique called Color Filter Array (CFA) interpolation. CFA interpolation is as much an art as a science. The math may be pretty straight forward, mostly summing and dividing, but the results depend highly on knowledge of the sensor and filter sensitivities to different wavelengths. Sample CFA calculations are as follows:

- **Mean Interpolation (Green)**
  - G11 = (G01 + G10 + G12 + G21)/4
- **Median Interpolation (Green)**
  - G11 = [(G01 + G10 + G12 + G21) - MAX(G01 + G10 + G12 + G21) - MIN(G01 + G10 + G12 + G21)]/2
- **Red and Blue may interpolated differently than green**
  - R11 = (R00 + R02 + R20 + R22) / 4
  - B11 = B11
  - R12 = (R02 + R22) / 2
  - B12 = (B11 + B13) / 2
  - B22 = (B11 + B13 + B31 + B33) / 4

Image Processing

Once the interpolation is complete our captured image has now ballooned up from 1MB (assuming an 8 bit A/D conversion step) to 3MB of complete image data. However we are not quite done from an image processing standpoint. The image from the sensor may have certain defects, noise artifacts and other quality issues that can be addressed using the processing power of the digital camera's microprocessor.

Image adjustments can best be thought of in terms of the adjustments for an old TV set – that is brightness, contrast, sharpness and tint. Brightness adjustment consists of adding or subtracting a base value from each of the pixels, typically a pixel value of 0 is darkest and 255 is the lightest. This has the effect of lightening (adding a constant) or darkening (subtracting a constant) an image. Contrast adjustment is a little more complicated. Adjusting the contrast of image consists of creating a histogram of all the values in an image and then applying a multiplication factor that spreads the histogram over the entire 8 bit range of the image. This is shown in the following figure.

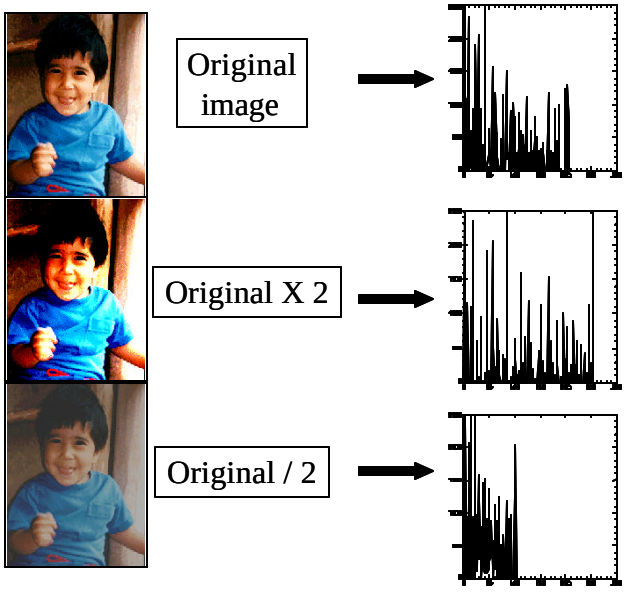

Figure 4 Contrast Adjustment

Sharpness adjustment brings us in to the realm of matrix math. An image is simply a 2 dimensional array of values that can broken down an manipulated over small sections, called kernels. Typical image processing in a digital camera is done using 3x3 or 5x5 kernels. During a sharpening (high pass filter or edge enhancement) or blurring (low pass filter) exercise, each pixel is evaluated and altered based on it and its nearest neighbor's (the kernel) values.

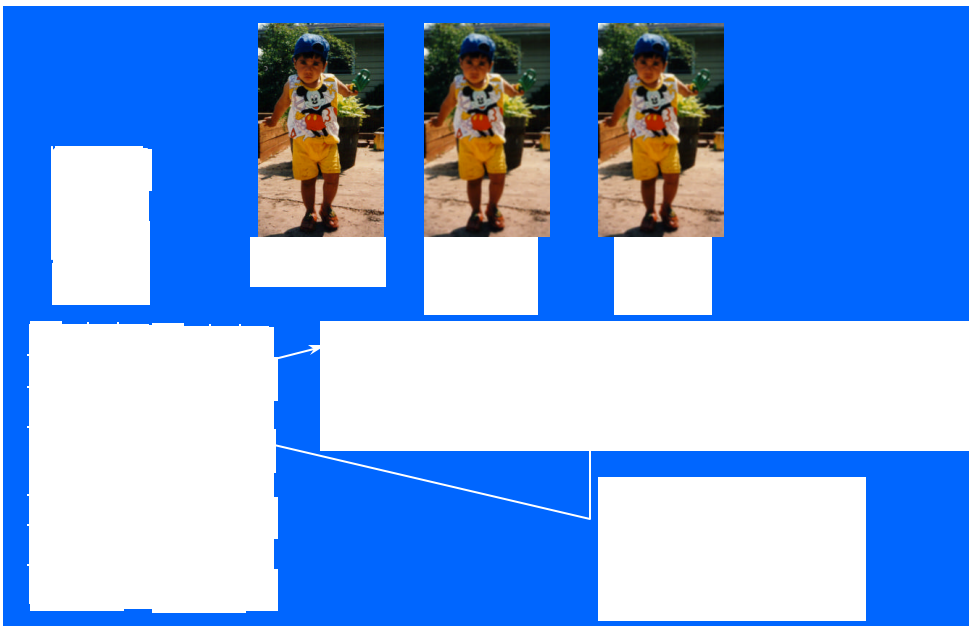

Figure 5 Blurring

Color correction is the process of adjusting the color of an image based on the lighting conditions it was taken under. The human eye and mind perceive colors very differently depending of the light source under which an object is observed. As well, skin tones recreation is highly dependent on lighting and sensor filters. Color correction is done by individually manipulating each of the pixel in color planes, adjusting the red, green and blue to create the most pleasing combination and faithful color reproduction.

Image Compression

Digital images are quite large. Even a low end, one MegaPixel camera creates finished images that are over 3MB in size. To conserve storage space and decrease transfer times, images are often compressed. Compression is the process of removing redundant or unimportant information to make reduce the amount of storage space required by an image. Compression can be lossless, which allows for complete recreation of the image with any loss of data, or lossly which means that unimportant data is thrown away, but visually the image still looks complete. The most common for of compression was created by the Joint Photographic Experts Group (JPEG).

JPEG compression is typically done in a different color space than the normal RGB. Transforming the image to a Luminance/Color based color space (YCrCb), via a 3x3 martix multiplication allows us to take advantage of the eye's relative insensitivity to color information. In fact, on of the first parts of reducing the size of the image is to throw away 75% of the color information as shown below:

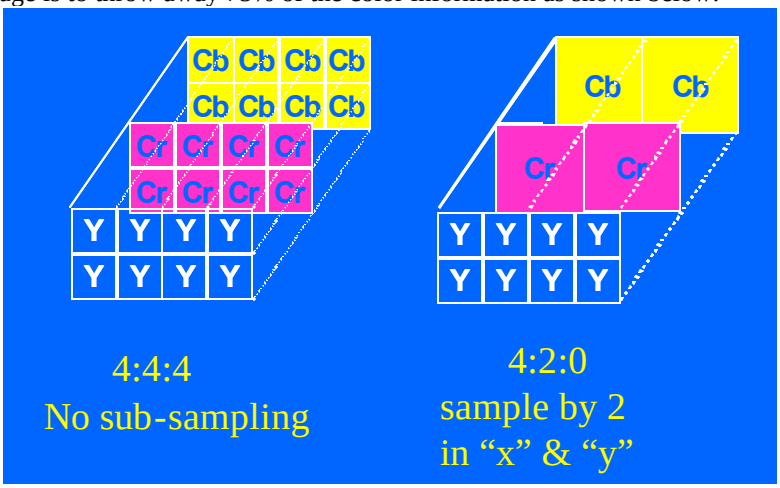

Figure 6 Color Subsampling

With 4:2:0 color subsampling, the luminance channel is left alone while each of the chrominance channels undergoes a 4:1 reduction. This is typically done by taking the average of a 2x2 array of chrominance bytes to generate a single chrominance byte.

Once we have undergone the color space conversion and performed the subsampling, we are ready to do the JPEG compression of the remaining image information. JPEG, like most compression algorithms, has three important steps – transformation, quantization and encoding. These steps in the JPEG algorithm are as follows:

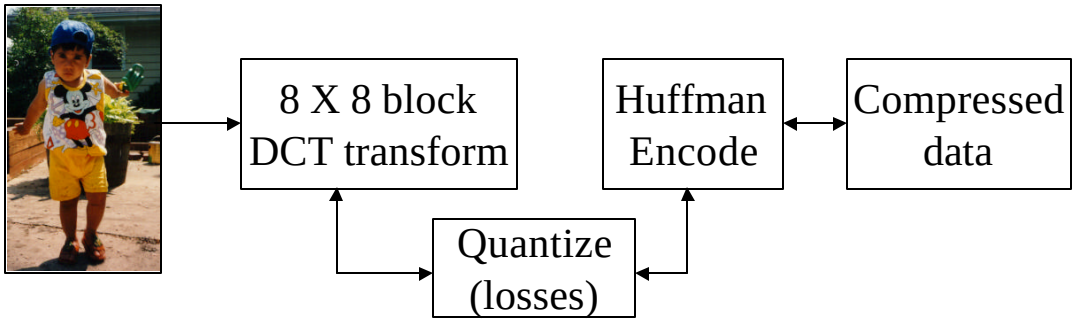

Figure 7 JPEG Compression

JPEG breaks an image into 8x8 blocks and uses the Discrete Cosine Transform (DCT) to gather the lower frequency information (which the eye is more sensitive to) into the upper left hand side of the resulting 8x8 matrix. In fact the upper left most value is a scaled DC average of the entire 8x8 block.

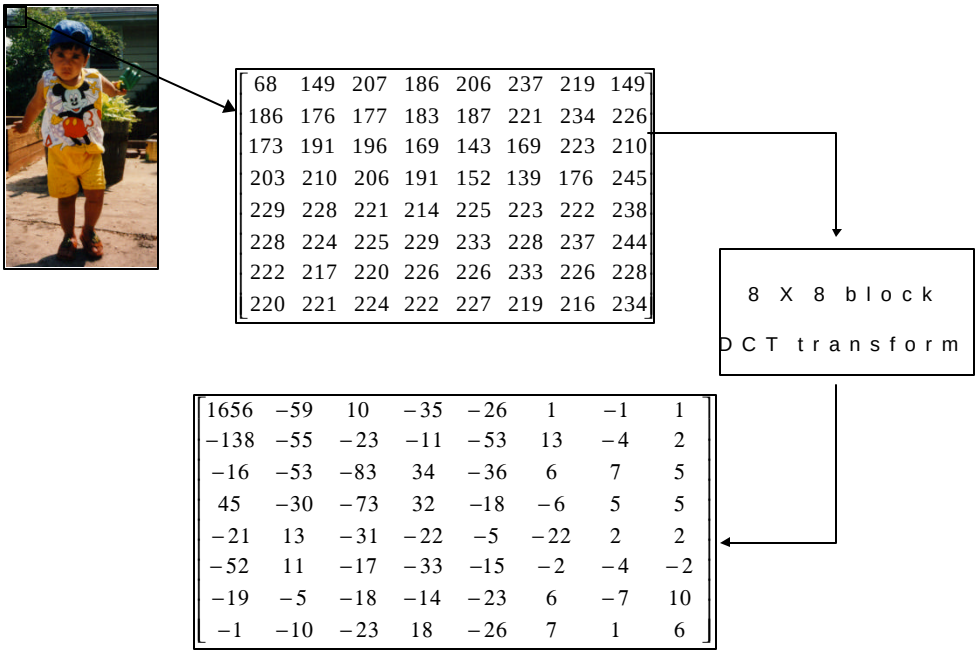

Figure 8 Discrete Cosine Transform

The resulting 8x8 transformed block is then divided by an 8x8 array of quantization values. This is where the only losses in the lossy JPEG compression algorithm occur. The quantization table is also what defines your compression ratio. Note that a fixed quantization table will result in different compression ratios depending on the content of the image. Images with a lot of low frequency content will compress more readily than busy images with lots of high frequencies.

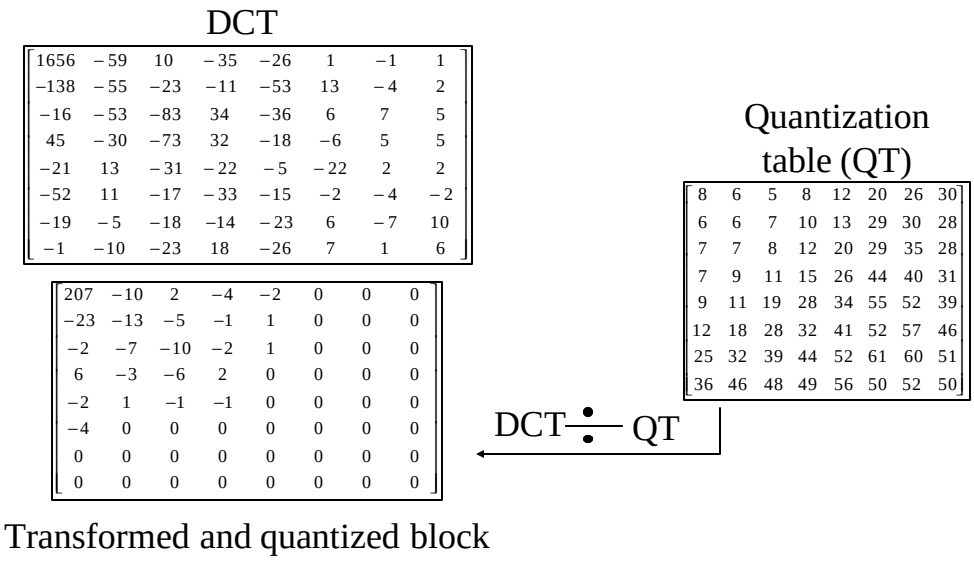

Figure 9 Quantization

Notice that once the block is quantized, most of the elements in the 8x8 array are now zero and not add any size to the compressed image. The resulting 8x8 matrix is “zig-zag” ordered which is a convenient way to generate long runs of zeros in the resulting data stream.

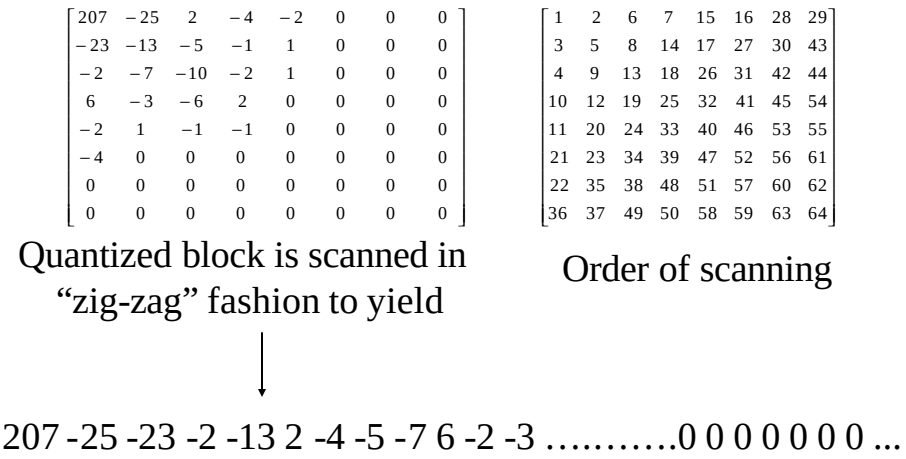

Figure 10 Zig Zag Ordering

The final step to the JPEG algorithm is run length encoding of resulting data stream. Huffman coding is typical although arithmetic encoding is also allowed in the standard. Huffman coding also contributes to the compression ratio of the final image by replacing "runs" of digits by a unique code based on their probability of occurrence. Since they appear most frequently, the long runs of zeros will be replaced by the shortest Huffman code.

Once the compression is complete, we simply have a Huffman encoded data set. This must be stored in a data file. JPEG is NOT a file format and does not specify any particular file format. The JPEG File Interchange Format (JFIF) is defined by the JPEG organization, but has fallen out of the favor of digital camera manufacturers who mostly now use a file format called EXIF 2.0.

Conclusion

This paper has just brushed the surface of the image capture, processing, storage, display and transmission that is becoming increasingly ubiquitous in consumer electronics. Software and hardware lines are becoming blurred as component manufacturers are trying to satisfy the unique needs of imaging equipment OEMs. It tales a team with expertise in the entire imaging chain to make correct decisions for the next generation of imaging products.