Christopher Tsai
February 20, 2008
EE 362 – Applied Vision and Image Systems

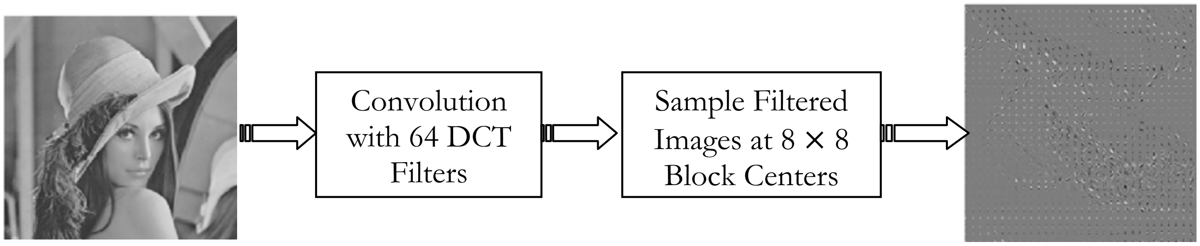
Tutorial IV – JPEG Black & White

Problem #1 – JPEG Compression of Grayscale Images

(a.) The initial steps in the JPEG-DCT computation essentially involve a convolution of the original image by 64 DCT filters, each accentuating different frequencies. By convoluting the original image with this bank of frequency-selective filters, the JPEG-DCT algorithm decomposes the image into content at 64 different spatial frequencies (or resolutions), much as the Fourier transform decomposes a function into its frequency components. As a result, we obtain a multiresolution representation of the original image: for each filter we apply to the original image, we obtain a coefficient image with the image content most resonant with the filter's passband frequency. For example, the first JPEG-DCT image will contain the mean of the image, likely most resonant with the dominant smooth areas or background in the original image. This convolution produces 64 replicas of the original image, each with content from the original image at the filter frequency.

JPEG-DCT then samples the filtered images at the centers of each 8×8 image block, building the coefficient image from the fully filtered replica images. Each block in the JPEG image comprises samples from each filtered replica, at the center of the respective region. Thus, for a single block, the 64 filtered images generate 64 coefficients indicating the variations in the block at each spatial frequency – or, equivalently, spatial resolution. The coalescence of all the coefficients in the JPEG block inform the content of the original image block, decomposed into discrete frequencies, with the coefficients indicating the amount of a certain frequency. The sampling is repeated for each 8×8 block in the image.

The following block diagram summarizes the discrete cosine transform (DCT) as convolution followed by sampling:

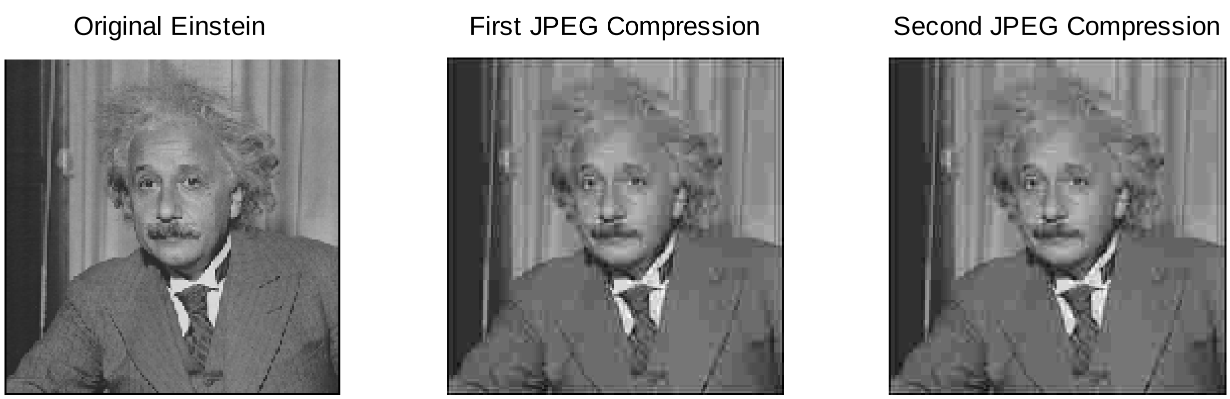


(b.) Information is lost only upon quantization. The larger the quantization step size, the more information we lose. Specifically, the *rounding* part of quantization kills the fineness of original data.

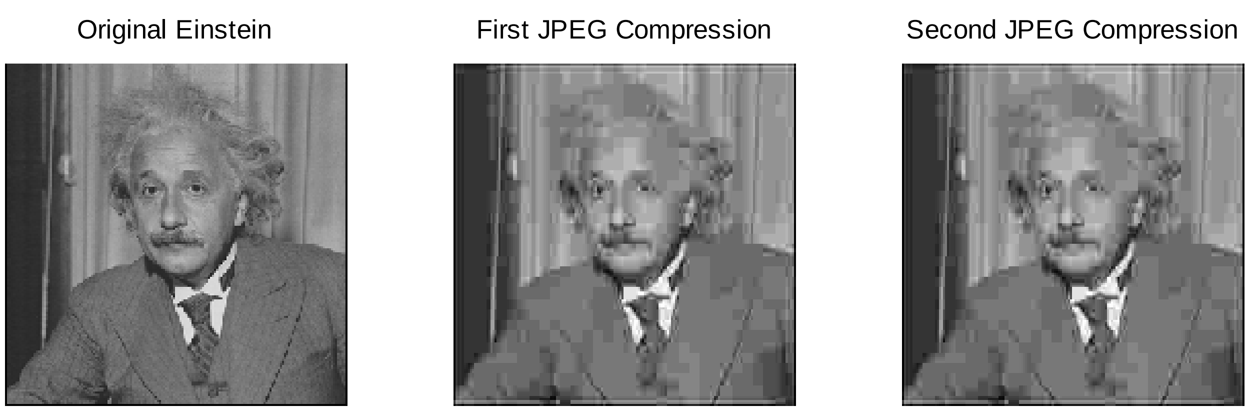
(c.) We quantize the JPEG-DCT coefficients and **not** the original image quantities because the JPEG-DCT coefficients indicate content at different spatial resolutions or frequencies, allowing us to selectively quantize less visible or less visually significant parts of the image more aggressively. Depending on how many edges or smooth regions the image contains, not all JPEG-DCT coefficients are equally important, whereas *all* image pixels generally hold equal value. Therefore, we cannot selectively quantize the original image since we cannot hierarchically rank the importance of one pixel against another, but the JPEG-DCT coefficients clearly sequester the smooth regions from the higher frequency content; since our eyes favor low-frequency content, we can afford to quantize the higher-frequency, less important coefficients at a larger step-size, thereby gaining compression without losing too much visual quality.

Pixels themselves do not perceptually decorrelate features, whereas the DCT coefficients decorrelate perceptually independent features into separate coefficients, allowing us to manipulate one set of features (such as sharp vertical edges) without tainting another. At the same time, the DCT exploits the human visual system's lowered sensitivity to high-frequency components, permitting us to quantize those values extremely coarsely since their artifacts are much less visible to the human eye.

(d.) Although successively zipping a file changes its file size, JPEG theoretically should not change an image in consecutive compressions. If an image is compressed using a JPEG-DCT quantization table of one quality, then uncompressing and recompressing the image using the same quality factor – and hence the same quantization scheme – will not alter the image further, since the same binning scheme will find the first compression's values already properly centered. Only changing the quantization table will change the JPEG compression; otherwise, the result of the first compression will fall exactly within the same bins during the second compression, and no further loss will occur since the quantization table will map all bin centers back onto themselves. We test this theory using Matlab's `imwrite` command:



The first and second JPEG compressions match identically. We conduct the same test using the tutorial's JPEG compression function, under a quality factor of 30:



Again, the results are comparable, although slight loss appears numerically, due not to JPEG itself but the truncation that follows it. In general, two consecutively compressed images will match; truncating the images in between will degrade the images slightly.

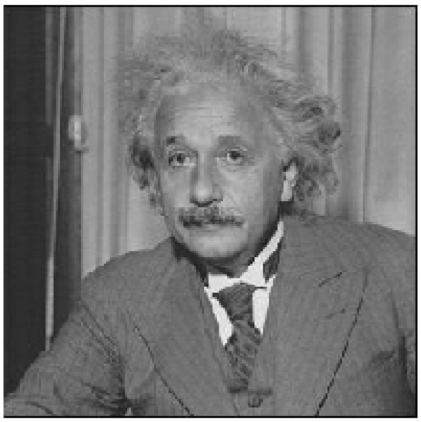
(e.) Because the pixels at the edges of 8×8 image blocks are compressed without any information about neighboring pixels in an adjacent block, JPEG-DCT block-compressed images will display blocking artifacts – sudden and irregular jumps from the edge of one block into another, without any correlation between originally similar pixels. This unnatural transition occurs because JPEG-DCT compresses blocks independently; the lack of consideration for surrounding blocks causes noticeable differences moving from one block to another.

All images are susceptible to blocking, but the ones in which blocking will be most obvious are images with features that extend across blocks; large, constant regions, for example, might appear in slightly different shades from one block to the next, therefore protruding quite noticeably as square-shaped regions dot the otherwise smooth landscape. Likewise, smooth contours or edges that cross a block boundary might also seem jagged or irregular following blockwise compression, since the effects of compression differ from one block containing the edge to another. Ultimately, the images that respond least favorably to block compression are the ones whose regions or features extend across several adjacent blocks.

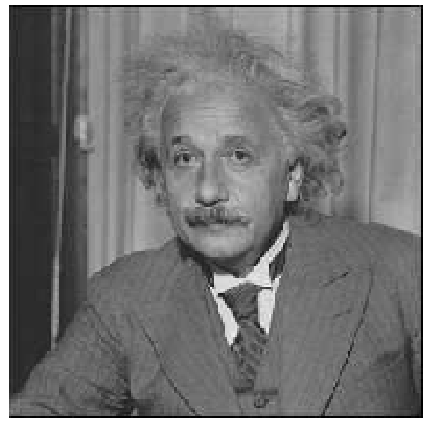
Problem #2 – Grayscale Error Metrics

We compress the Einstein image according to JPEG quality levels of 75, 50, 25, and 0:

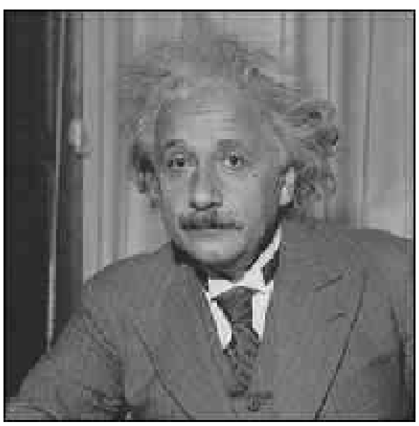
Compressed Einstein (Q = 75)



Compressed Einstein (Q = 50)



Compressed Einstein (Q = 25)

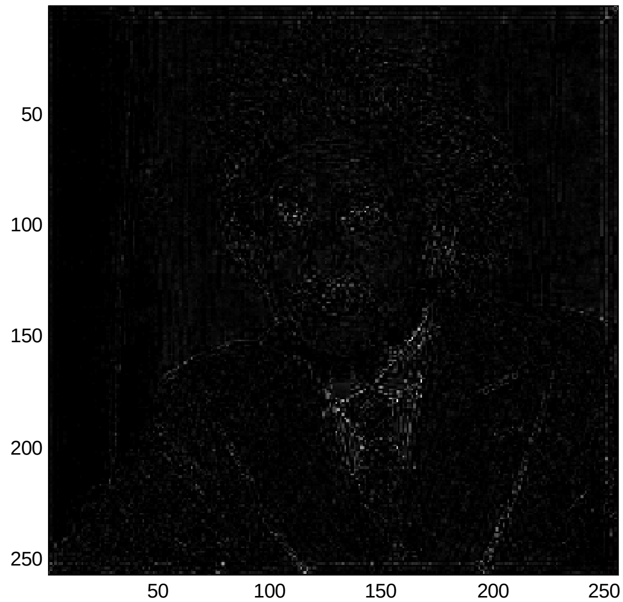


Compressed Einstein (Q = 0)

As expected, the blocking artifacts grow worse and worse as we decrease the quality factor (or, equivalently, as we increase the quantization step size). For the Q = 0 image, Einstein is almost unrecognizable because we have so few levels to represent the image.

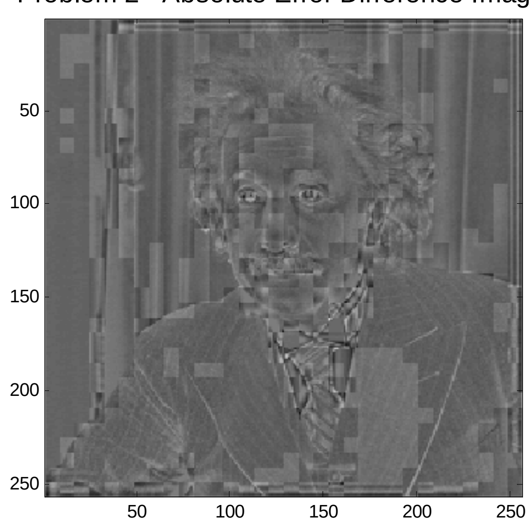
(a.) The square error image for Q = 25 manifests that the greatest error occurs along the edges, such as those between Einstein's shirt and suit jacket, as well as edges between Einstein's face, hair, and the background wall:

Problem 2A - Square Error Image for Q = 25

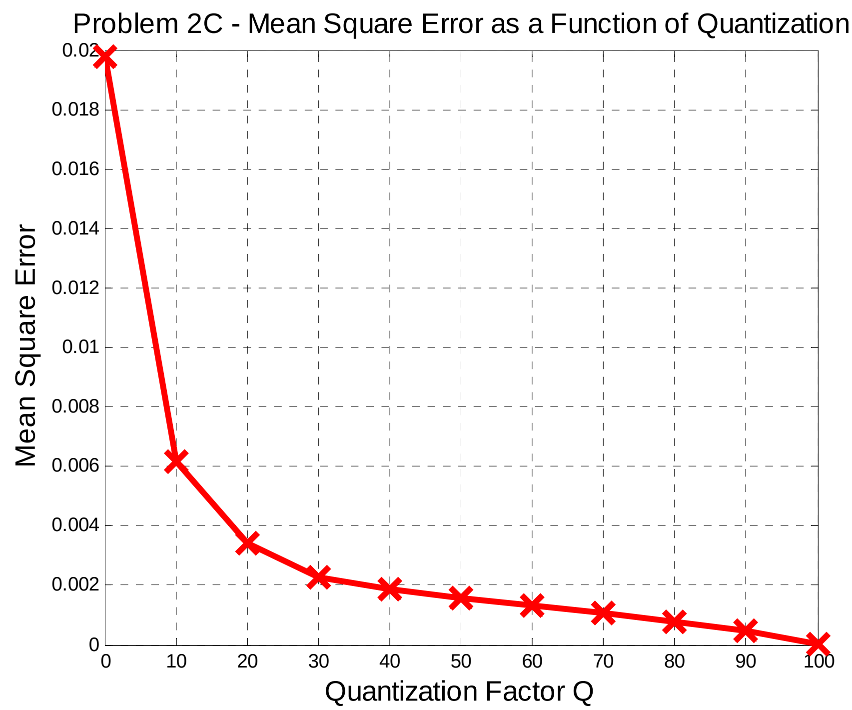


(b.) Indeed, we expect greatest error along the edges because the JPEG-DCT algorithm quantizes the high frequency coefficients most aggressively; by using fewer quantization levels on the high-frequency coefficients, JPEG-DCT essentially discards more edge information than smooth region data, since the average loss is higher for fewer quantization levels. As a result, we see the most squared error along these edges, especially along sharp or sudden transitions such as those between Einstein's white shirt and dark jacket. Furthermore, we can see blocking artifacts visibly in the absolute error image:

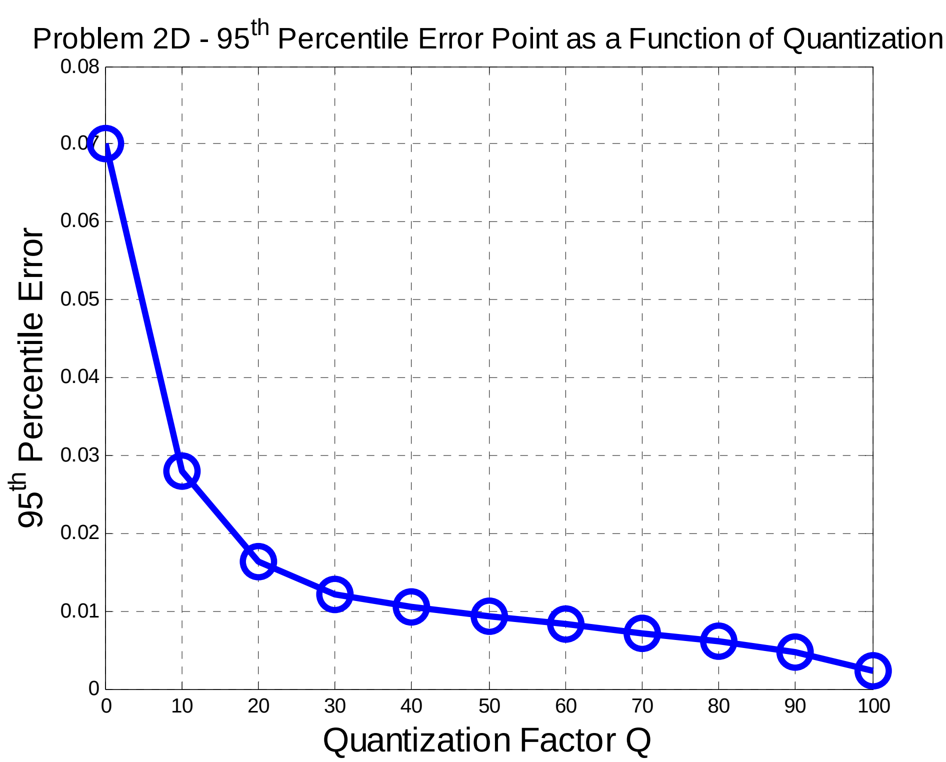
Problem 2 - Absolute Error Difference Image



(c.) To compute the mean square error, we first normalize the original Einstein image and our compressed version of it. Then, we define $MSE = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(einstein - compressed)^2$:



(d.) To compute the 95th percentile error, we sort the error values and remove the highest 5%:



(e.) If we were evaluating image quality for this image, then we would favor the 95th percentile error metric since it displays much more evenly spaced error values across the entire range of quantization factors. Juxtaposing the two plots, we notice that, while both metrics capture the decrease in distortion with higher quality factors, the mean square error values cluster around 0.02 with little disparity to differentiate the Q = 25 and Q =50 image, for example. As a result, the more finely spaced 95th percentile metric better captures the steadily increasing image quality (with increasing quality factor) and better reflects the overall quality of the image. Apparently, discarding the extreme outliers in the top 5% allows the metric to be less sensitive to edge loss (where the largest errors reside) and therefore more representative of quality throughout the image, as our eyes might extract.

On the other hand, one could also argue the merits of the mean square error metric by citing the lack of improvement past Q = 50. In other words, some human eyes – like my own – struggle to discern much quality improvement beyond a certain detectability threshold; the errors are much harder to catch as we increase Q. Thus, in this sense, by representing smaller improvements with increasingly smaller decreases in error, the mean square error metric more accurately reflects our perception that the image improves less and less as we detect fewer and fewer errors. The steep decline around Q = 0 also justifies our vision of the sudden improvement achieved when moving from the unrecognizable block image to the first fully recognizable Einstein. In conclusion, both metrics have their merits, although the 95th percentile error metric is more successful at summarizing the general global quality since it disregards the most extreme edge effects.

Tutorial V – JPEG Color

Problem #1 – YCbCr Information

In order to familiarize ourselves with the information content in each channel of the YCbCr color space, we load a color image of Typhlosion and decompose it into YCbCr:

Full-Color Typhlosion

Luminance (Y) Channel

Blue Chrominance (Cb) Channel

Red Chrominance (Cr) Channel

Typhlosion illustrates the information content well because its body comprises both red and blue parts.

The blue chrominance channel (Cb) isolates all shades and tints of blue in the original color image, such as the blue stripe along Typhlosion's back and head, as well as the greenish blue present in the image background. Cb represents a color difference between blue and a reference value.

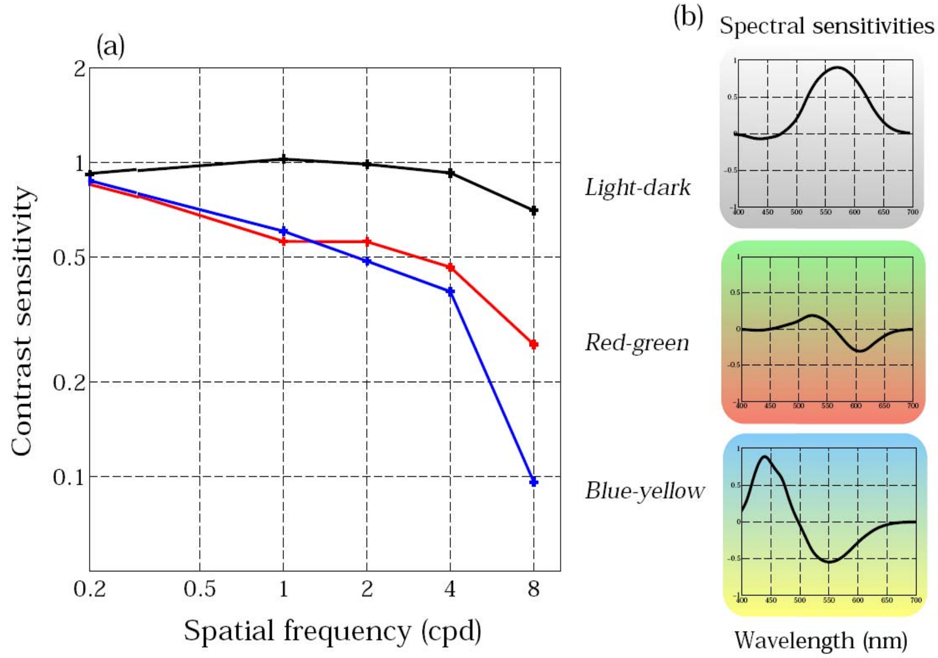
The red chrominance channel (Cr) isolates all shades and tints of red, including those in the tan-colored torso of Typhlosion. Cr represents a color difference between red and a reference value. Just as red does not appear in the Cb channel, blue does not appear in the Cr channel; we can see that the Cb and Cr images of Typhlosion are mutually exclusive, with missing (black) content in one channel appearing strong (white) in the other.

Despite its grayscale appearance, the luminance channel (Y) contains most of the edge detail to which our eye is most responsive. Whereas the edges in the chrominance channels are soft and often imperceptible, the luminance channel displays them in all its high-frequency detail, essentially replicating the original image in grayscale. Thus, we can begin to understand why image compression algorithms decimate and aggressively quantize the chrominance channels; the human eye is most sensitive and receptive to the information presented in the luminance channel, as the principal variations and features appear much more clearly and completely in luminance than they do in either color channel. We can recognize Typhlosion from the luminance channel alone; in the two chrominance images, the ferret's identity is much less obvious.

Mathematically, we can transform an RGB signal to a YCbCr signal with a linear transformation represented as matrix multiplication:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.996 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Problem #2 – Exploitation of YCbCr for Lossy Compression



Our general visual insensitivity to high-spatial-frequency color information allows us to discard high frequency content in the Cb and Cr channels, thereby compressing the bandwidth of two of our three channels and achieving a high compression ratio without sacrificing noticeable image quality. As the Typhlosion example reveals, our visual system perceives the most high-spatial-frequency detail in light-dark modulations such as those represented in the luminance channel; on the other hand, the high-spatial-frequency variations in color-difference signals are generally imperceptible to the human eye, so we can completely omit them without degrading the viewed image. As a result, compression algorithms like JPEG typically exploit the imperceptibility of high frequency chrominance by leaving only low-frequency content in the color difference channels (CbCr). This reduction in bandwidth achieves high compression at very little cost, since our eyes cannot readily discern the high frequency components; color edges are effectively invisible, so removing them from the signal preserves most of the image quality, which we can still perceive from the detail in the luminance (Y) channel (see Typhlosion).

Problem #3 – Compression of Color Images in Different Spaces

To accentuate the properties inherent in color channels, we load a fundamentally colored image into Matlab – the cover art for the Capcom classic, Phoenix Wright:

Phoenix Wright RGB Image

Phoenix Wright Red (R)

Phoenix Wright Green (G)

Phoenix Wright Blue (B)

We notice that the three channels appear roughly equivalent in information content; all of the color component images contain enough high frequency data (edges) to distinguish the characters from their surroundings. The image is recognizable in any of the three channels.

The Red-Green-Blue (RGB) image transforms readily into a Luminance-Chrominance (YCbCr) image, which we display on the following page:

Phoenix Wright YCbCr Image

Phoenix Wright Luminance (Y)

Phoenix Blue Chrominance (Cb)

Phoenix Red Chrominance (Cr)

Clearly, it makes no sense to display an RGB image from the YCbCr components, as we attempt to do in the upper-left trichromatic image; we simply obtain a pseudocolor image bearing shamefully little resemblance to the true color image. However, the individual channels hold meaning. The luminance (Y) channel displays all of the high-spatial-frequency data that we can perceive, so it is the most visually recognizable of the three channels; the chrominance channels contain pure color data, whose high-spatial-frequency content is not nearly as perceptible to our visual system. We can, however, clearly distinguish Phoenix's blue suit from the blue chrominance (Cb) channel, while the red tie protrudes prominently from the red chrominance (Cr) channel. Thus, we can crudely correlate the blue and red chrominance channels with the red and blue channels, with the primary difference being the opponent-color referencing in the blue and red chrominance channels; a low Cb value indicates more yellow content, whereas a low Cr value simultaneously admits more green content.

(a.) Our investigation of color channel compression begins with an aggressive quantization of the blue (B) channel in the RGB image:

Coarse Blue Reconstruction

Compressed Red (R) at Q = 75

Compressed Green (G) at Q = 75

Compressed Blue (B) at Q = 5

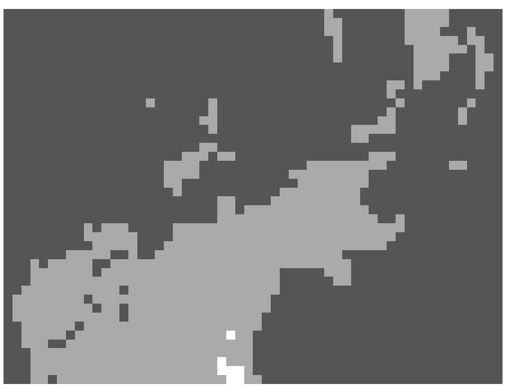
As the lower-right image reveals, the blue channel essentially loses all fine structure under our coarse (Q = 5) quantization. However, the reconstructed image retains most of its content! We can still distinguish the human characters from their surroundings. The only artifact of our aggressive quantization is the immense blockiness throughout the image, none more visible than the dark blue on Phoenix Wright's suit; because the quantized blue data varies coarsely and transitions sharply, the corresponding blue regions in our reconstruction – namely, Phoenix's suit – also appear blocky to reflect the extremely limited range of values we have to represent shades of blue.

(b.) Similarly, we can aggressively quantize the blue chrominance (Cb) channel while preserving the luminance and red chrominance channels. We reconstruct the following image:

Coarse Cb Reconstruction

Compressed Luminance (Y) at Q = 75

Blue Chrominance (Cb) at Q = 5



Red Chrominance (Cr) at Q = 75

From the lower-left blue chrominance image, we can tell that quantization has severely crippled the blue chrominance channel to the point that no detail is visible, and no features identifiable. However, the reconstruction suffers only a discoloration. Unlike the blue quantization detailed in the previous part, the chrominance compression simply alters color information by restricting the range of blue we can represent; however, the suit displays few blocking artifacts because all of that detail resides in the luminance channel, which we quantized quite finely (Q = 75). As a result, we conclude that the YCbCr color space decorrelates fine, feature-related detail and pure color, allowing us to aggressively quantize a single color without compromising the quality of the features or edges.

Of course, the color information is also a visible artifact, but the strangely altered hair color in Phoenix Wright and Miles Edgeworth are arguably less severe than the blocky regions observed after blue quantization. All in all, Cb quantization reduces the range of blue values without drastically reducing the range of spatial detail our image can display.

(c.) Setting various chrominance components to zero, we witness the dominance of the opponent colors:

Phoenix Wright Complete Reconstruction

Lost Luminance (Y = 128)

Lost Blue Chrominance (Cb = 0)

Lost Red Chrominance (Cr = 0)

When we eradicate blue chrominance, blue's opponent color, yellow, floods our image. Likewise, as we remove red chrominance, red's opponent color, green, overwhelms the viewing plane. Setting luminance to its midpoint essentially removes all perceptible high-spatial-frequency detail, as the edges outlining Phoenix's body and characters' faces fade into a blur of red and blue. Colors still appear because the chrominance channels contribute pure color, but the edges vanish with the luminance.

Finally, we can set the chrominance channels to their midpoint values, now eradicating luminance:

Phoenix Wright Complete Reconstruction

Lost Luminance (Y = 0)

Lost Blue Chrominance (Cb = 128)

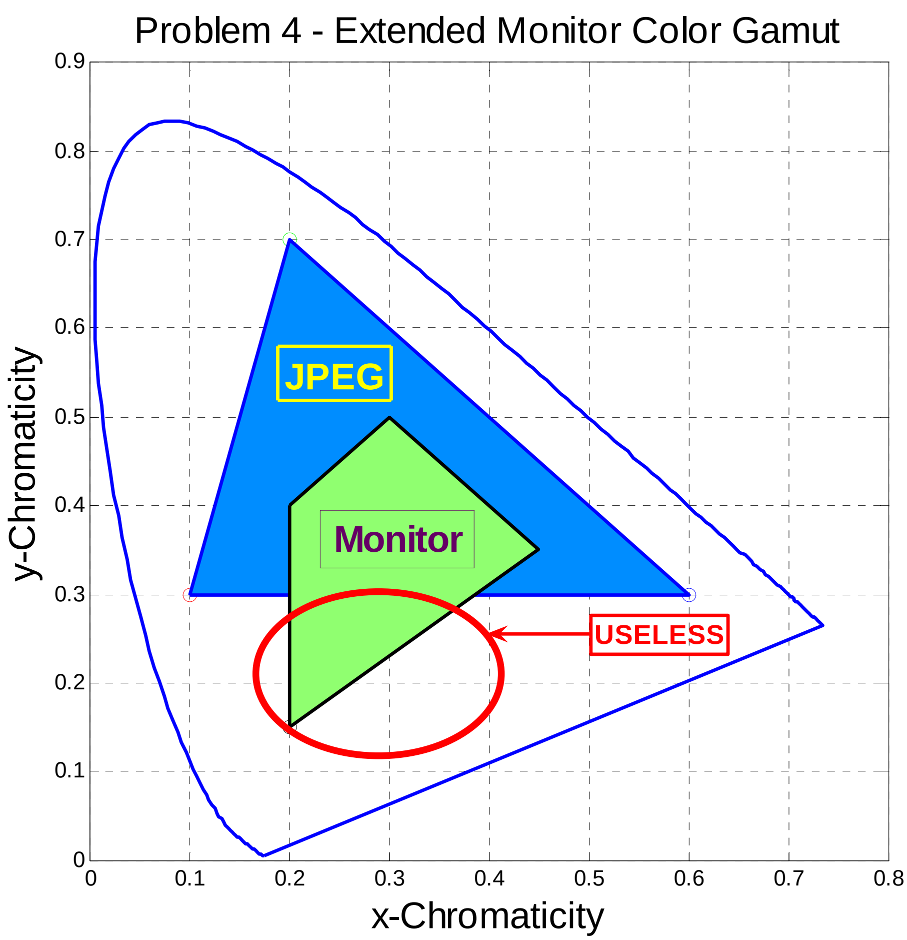
Lost Red Chrominance (Cr = 128)

The luminance-deficient image is darker than its Y = 128 counterpart because a low luminance also invites low brightness; however, the red and blue chrominance components persist, as the background crimson and Phoenix's dark blue suit attest! Meanwhile, when we set chrominance components to 128, the focus color completely disappears from the image, accentuating the opposite channel; setting Cb = 128 removes the blue from Phoenix's suit while highlighting the pink throughout the background, whereas setting Cr = 128 removes all the red while preserving the now-conspicuous blue in Phoenix's suit. Interestingly, Phoenix's tie also turns blue!

Bonus Problem #4 – The Fourth Primary in JPEG

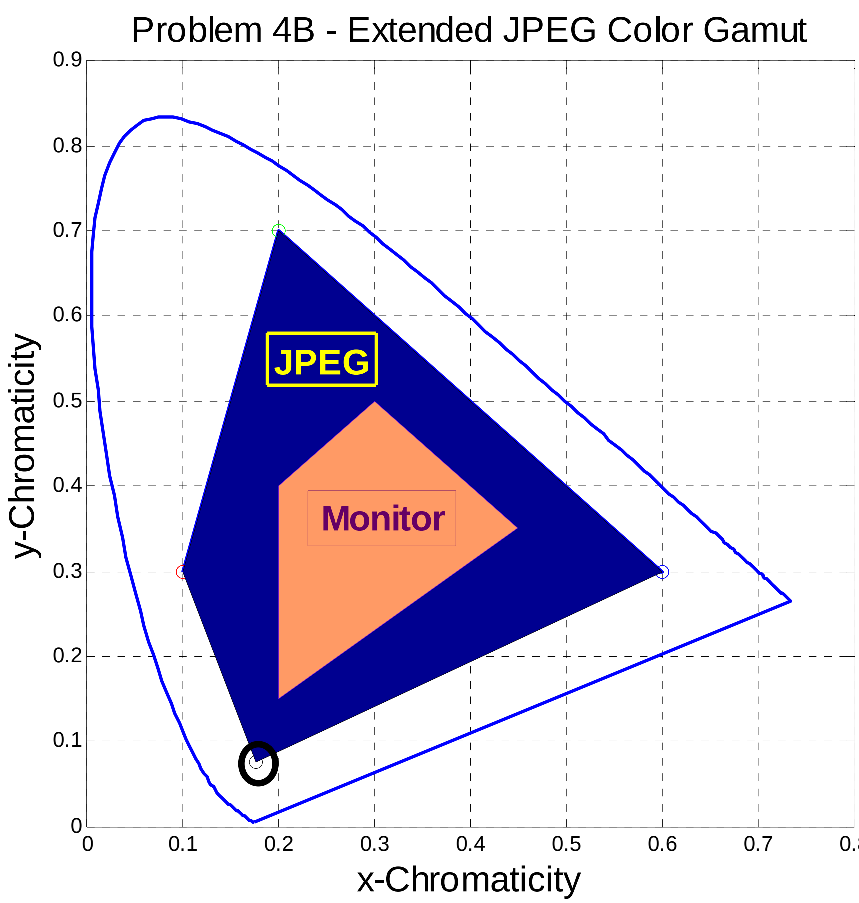
We assume that the JPEG color gamut is a small subset of the visible gamut, and that the gamut of the three-primary monitor is itself a smaller subset of the JPEG gamut.

(a.) In the color-matching tutorial, we noted that the fourth monitor primary allowed us to cover a larger area of the visible gamut. However, this fourth primary helps in the display of the JPEG image only to the extent that the additional area afforded by the fourth monitor gamut vertex overlaps with the JPEG gamut:



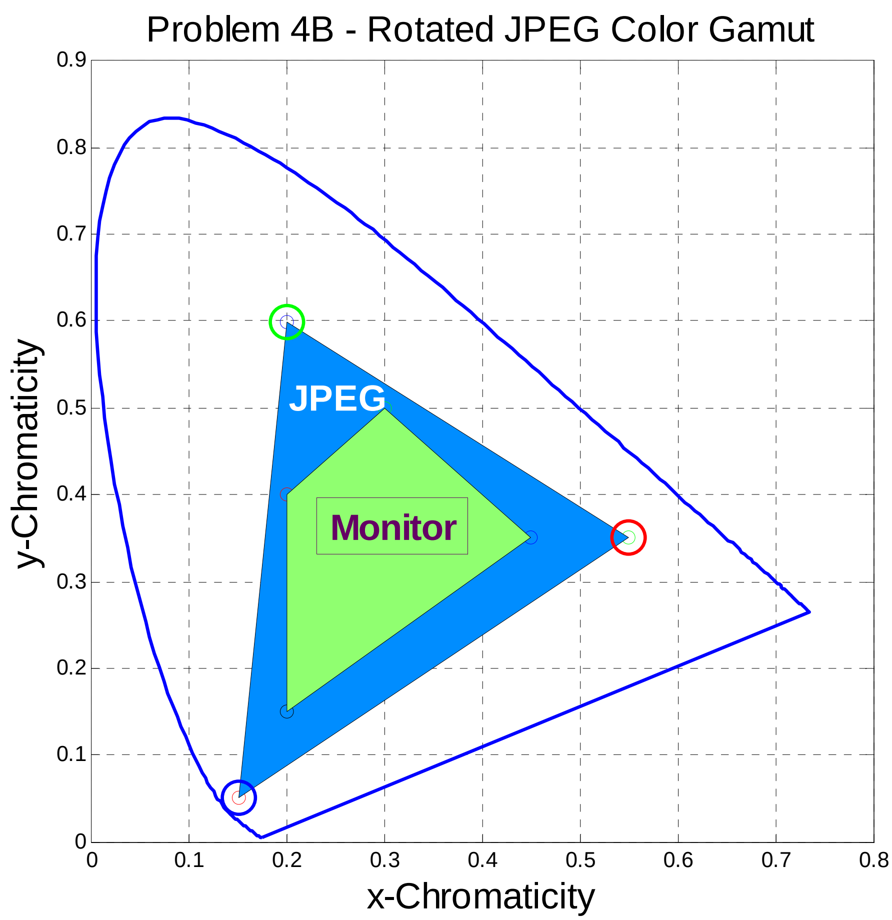
Notice that the fourth primary – augmented to the monitor gamut as a fourth vertex – could add a significant number of colors (area) to the displayable gamut, but all the additional colors that fall outside the JPEG gamut do not contribute to displaying the colors in the JPEG image; if the original JPEG gamut does not include a newly displayable color, then the addition of that color to the color monitor's gamut will not matter when displaying the JPEG image. Even though the augmented gamut clearly boasts a wider range of displayable colors, JPEG images will contain and therefore require only colors in its own subset, rendering all exterior displayable colors irrelevant to its quality. In conclusion, the additional colors that fall within the JPEG gamut will help reproduce JPEG colors on the screen, but colors external to the JPEG gamut will not contribute at all.

(b.) A cost-*inefficient* method involves the brute force addition of a fourth color primary to the JPEG basis so that its gamut fully envelopes the entire gamut of the augmented color monitor:



In order to implement this additional primary, the JPEG standard would have to support a fourth color channel, therefore necessitating a fourth image (33% more data overhead) as well as performing an additional set of calculations upon compression. In other words, instead of declaring images as stacks of three component images, we would have to support image stacks of *four* component images every time we declare a color image, either for computation or storage. The storage overhead itself would complicate matters, but the computational costs of an additional channel would prove cumbersome for compression algorithms optimized for speed. This simply does not seem efficient given our well-established comfort with three primary colors.

However, a more efficient option exists. We can modify the standard *without any extra data costs* by simply rotating the existing JPEG gamut so that it maximally covers the color monitor gamut. We can implement this JPEG gamut rotation and translation by assigning slightly different color primaries for each of the three vertices, thereby completely covering the monitor's gamut:



Physically, we have reassigned the primary chromaticity coordinate pairs of each JPEG basis, which entails a simple linear coordinate transformation. With a simple matrix multiplication, we can now fully exercise the full range of our new color monitor's display spectrum!