Ergonomics

Here we shall explore the interesting field of ergonomics, or human factors. We shall be concerned with human activities that involve interaction with devices. The devices may be part of a manufacturing system, or a gadget, or even software on a computer. The purpose of our investigation is to *understand, evaluate, and thereby, to improve the interface between the human and the device*.

While I prefer the term ergonomics, I will use the terms HF and ergonomics as equivalent in the notes. The main topics will be divided into the following sections:

1. Examples of design problems from a point of view of HF.
2. Case Study. Improving design based on HF: digital images and JPEG
3. Comment on methodology and tools useful for HF

---

1. Examples of Product Design Problems and Ergonomics

Example 1. Office desk and chair design

Question: *How do we decide the height of the desk?*

Among other factors, the answer depends on:

(a) the height of the chair

(b) the size of the person who will be using the desk and chair

It turns out that design of ergonomic chairs is a favorite topic for several hundred researchers in HF. This might suggest that the problem is complex and interesting. It is certainly true that ergonomic office chairs sell for over HK$ 8000 each, so efforts into the design of a good chair can be economically fruitful.

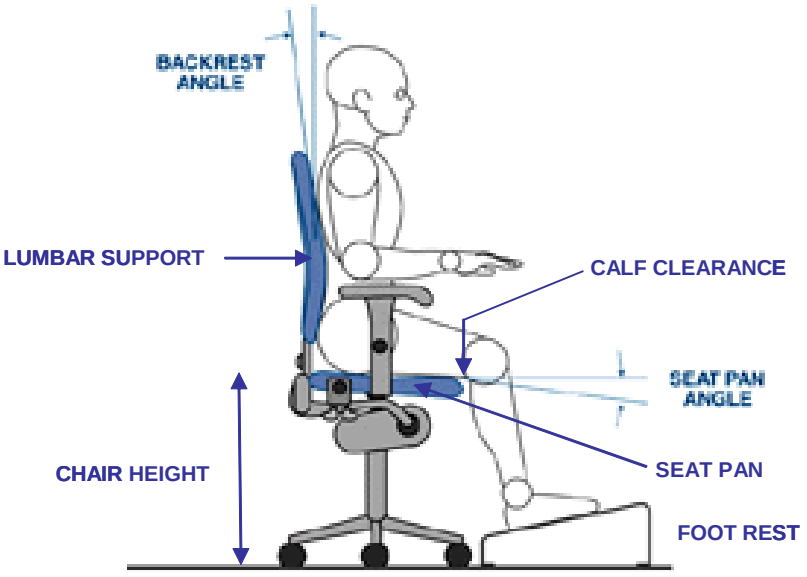

Figure 1. Chair design, some important factors

The following are some ***guidelines regarding the design of ergonomic chairs***:

(i) The length of the seat pan should provide a calf clearance to 95% women in the population. Ideally, the clearance should be at least 5cm.
(ii) The chair height should allow a light contact with the lower thigh when the user is sitting with both feet planted flat on the foot-rest (or floor).
(iii) The seat pan angle should be adjustable within ± 6°
(iv) The arm rests should be a height to match the elbow height when the arm and shoulder are relaxed and the upper arm is straight down from the shoulder.
(v) The backrest lumbar support should be between 15-25 cm above the compressed seat level (i.e. the height of the seat pan with the user in the chair).

Questions:

(a) What is involved in deriving such guidelines?
(b) Are the guidelines 'precise' enough ? [e.g. what is "light contact" ?]

As we consider these questions, we must also note the following:
(i) Some level of adjustability is required in various parts of the chair to allow different users to use a given chair design.
(ii) The design of a good chair (and therefore desk) will depend on the statistics of the population that is expected to use it. The table below shows some interesting statistics about average heights (in centimeters) of different national populations (the data is a few years old)

	USA	**Germany**	**Japan**	**Netherlands**
Males	175.5	174.5	165.5	182.5
Famales	162.5	163.5	153.0	169.6

Table 1. Average height of males and females in different populations

To make matters complicated, such statistics are time dependent: a web report from People's Daily newspaper claims that the average height of urban Chinese males has increased by 6 cm over the last 20 years.
Let's take the example of the chair height – it is clear that we need to allow adjustable height. How much adjustment should be allowed? It should be sufficient such that, say, 99% of the users can adjust it so as to achieve requirement (ii) above. Therefore, we need to first find out the statistical information of the length of lower leg for the population. How to get this data? Obviously, we cannot measure all possible users; luckily, we can use sampling (which will introduce a further error in our estimates). You will learn about sampling when we discuss Quality Control.

Let's consider what happens if we use a chair/desk that does not satisfy these requirements. What happens to such a user? Will he face discomfort? Risk of injury? If so, how do we evaluate such risks? How can we define/estimate/eliminate such risks?

Example 2. Keyboard design

In the introductory lecture, we saw examples of different types of keyboard designs for computers. Does it really matter what design we use? In fact, it matters in a way that is much more important than you may anticipate. Further, it is not merely the design of your keyboard, but also of the entire working environment (including the height of your chair, desk, and several other factors. The problem is caused by extended periods of use of a computer (e.g. at work, or playing video games!) in the wrong posture. There are several causes of the problem, e.g. the pressure felt by the wrist as it rests along the edge of the hard desk, the stress in the tendons due to the unnatural angle at which the hands are held during typing, etc. The stress, though it is generally small enough to be below our threshold of noticeable pain, adds up over time, leading to a problem called ***Repetitive Stress Injury*** (***RSI***). The most common one of these due to extended computer use is called the ***Carpal Tunnel Syndrome (CTS)***. This is a medical problem caused by the *compression of the median nerve as it enters the hand*. The symptoms include the numbness of thumb and fingers, pain along the median nerve including hand, wrist, elbow or shoulder, and weakness of thumb. In the worst cases, it must be treated by surgery; in mild cases, it can be reversed by long periods of rest (and so, lost work days). The main cause is: flexed (i.e. bent) or extended wrists when typing on a keyboard!

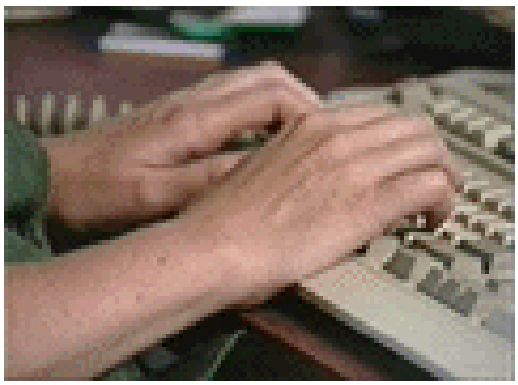

Figure 2. CTS: the cause

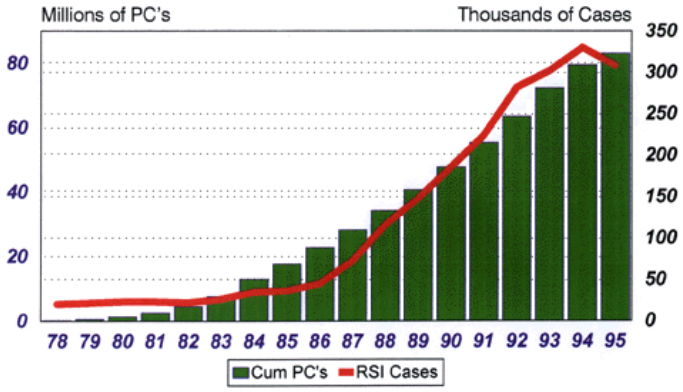

Figure 3. Correlation of PC usage with CTS
[source: Chicago ergonomics group]

Example 3. How to turn on the shower

In most bathrooms with a bath-tub, there are two water sources – the tub-faucet and the shower-head. However, only one can be turned on at a time, so a single control is usually provided to turn the water on/off, along with another control to switch the outlet of water from the shower-head to the tub-faucet, and vice-versa. On a recent trip to China, staying in a nice hotel, I encountered this example, where the switch-control is located in the most remarkable place: a ring below the tub-faucet! It took me a few minutes of searching, when I found a note stuck to the wall explaining how to turn the shower on! I later found an image of this, and several other amusing examples from a web-page, *www.baddesigns.com*.

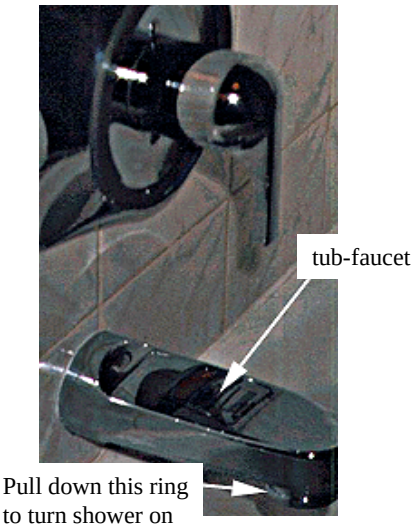

Figure 4. Poor design of shower controls
[Photograph courtesy of www.baddesigns.com]

Example 4. Flush in an airport.

This is also an example (from the same source) that I have encountered in an airport toilet (luckily, not Chek Lap Kok). The toilet flush is activated by pressing a button, which is located in a place that is not too obvious. Surely many people are too frustrated in searching for it and leave the toilet without flushing. The image below shows an example.

Figure 5. Press the button to flush
[Photograph courtesy of www.baddesigns.com]

Example 5. Is the water too hot? Too cold?

Many water faucets (and also showers in bathrooms) have two separate handles to turn on the hot-water and cold-water supplies to the faucet. In many cases, the H-handle turns clockwise, and the C-handle turns counter-clockwise to increase the respective flow rate. However, it often takes some trial and error to turn off the water, and sometimes at the cost of either getting hit by a blast of cold, or very hot water! By the way, the water taps in the UST swimming pool shower room are exactly this design!

Figure 6. Turn which way to turn off?
[Photograph courtesy of www.baddesigns.com]

There are hundreds of other examples of poor design. Among the ones presented above, the first one is related to the concept of comfort. The second is related to safety (which is an extreme case of discomfort). The last three are not really related to safety, but perhaps to inefficiency of use. In each case, an understanding of human physiology and psychology is important in improving the design. Later, we shall look at a case study: it will show how an understanding of the human body works helps to make more efficient designs.

2. Case Study: digital image files

An exciting new development in consumer electronics is that of digital cameras (digicams, for short). Unlike traditional film cameras, digicams store pictures in a digital "memory", which is a small electronics chip. Each image is stored as "data". What is the data composed of? A simplified model, which I will call the "RGB-pixel model" will suffice to demonstrate the fundamentals.

What is a digital image: pixels

Consider the image in Figure 7. It is a fairly colorful image, of a fish called a lion fish. The colors in the image are continuously (and sometimes sharply) changing from point to point. However, what if zoom in on a small area of the image? Figure 8 shows what happens if we repeatedly zoom in on smaller and smaller regions of the image – at some stage, it appears that the image is made of squares, each square being made of a single color. In reality, these "one-color-cells" don't have to be square, they can be rectangular, or some other shape, but we will assume that they are square.

Figure 7. An image of a lionfish

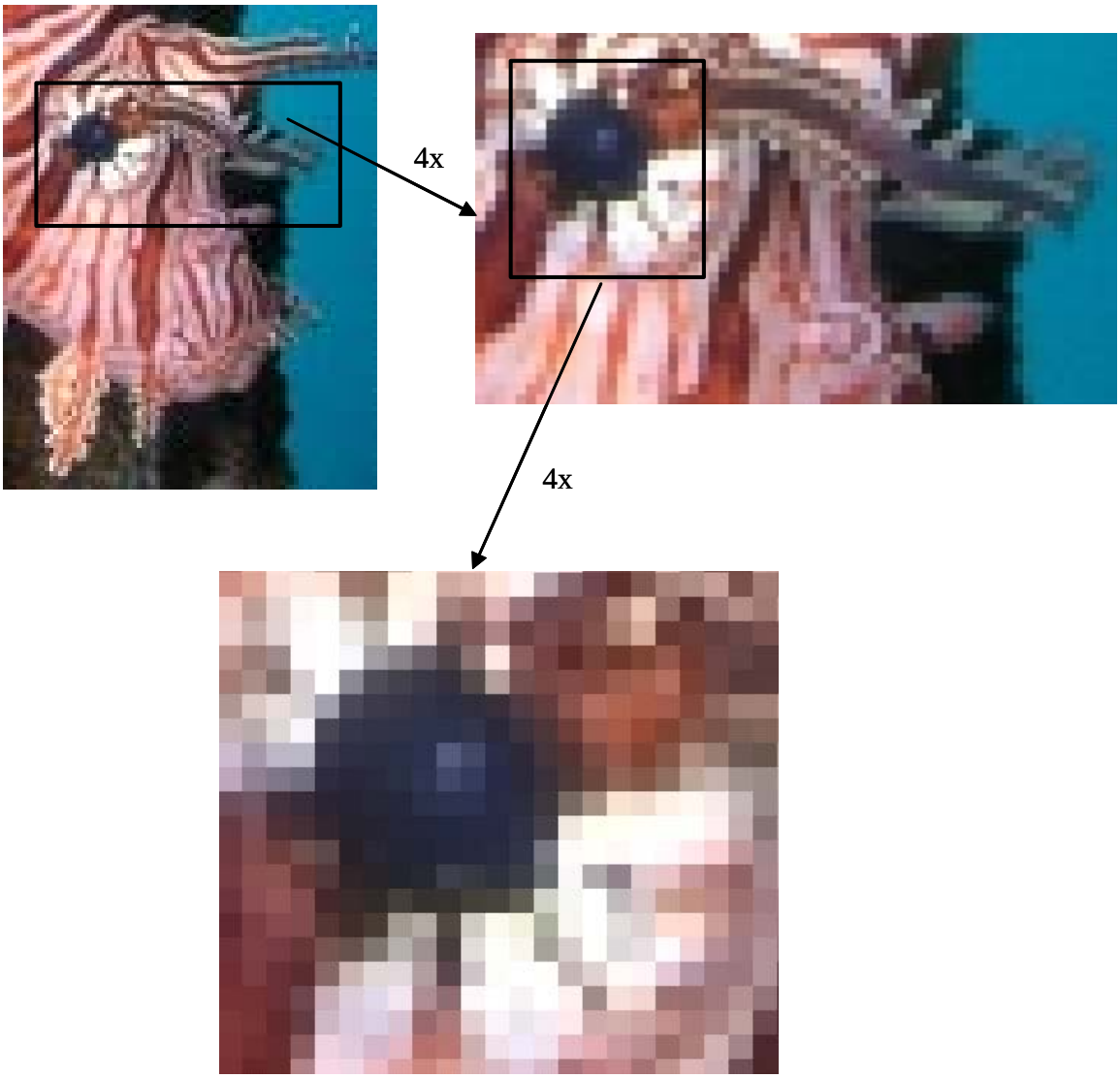

Figure 8. The image, upon repeatedly zooming in, shows pixels

Now we can imagine the entire picture being made up of a large grid, or array of one-color cells – let's call them "picture elements", or ***pixels*** for short. This method of storing the image data is also called a ***raster image***.

The RGB model

How do we represent the color of each pixel? As you all know, colors are our perception of visible light, which ranges roughly from wavelengths of ~380-750 nanometers (1 nanometer = 10^{-9} meters). So for each pixel, we can store the frequency of the corresponding color and its intensity. However, for technical reasons, this turns out not to be useful.

We are also commonly told that any color can be made up by combining the three primary colors, Red, Green and Blue in the correct proportions. We shall explore this idea further, below. But for now, let's assume this to be correct. Using this model, we can therefore convert the actual color of each pixel into its primary components, the Red level (R), Green

level (G) and Blue level (B). We can therefore store the color of each pixel by storing its RGB values.

Thus, the RGB-pixel model of images stores the entire image as a rectangular grid of pixels, with each pixel having its own R, G and B values.

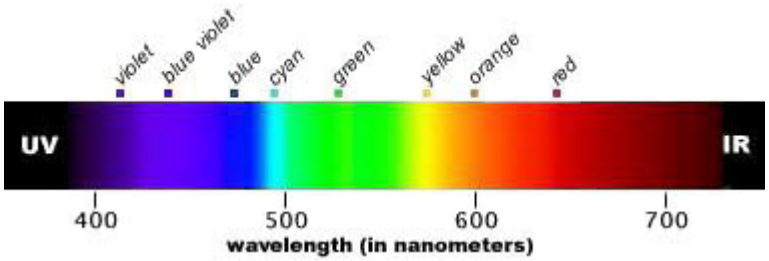

Figure 9. The visible spectrum

Further, we shall assume that each color can only take a value within a certain range (you can think of it as follows: outside these ranges, our eye cannot detect any change in the color intensity, since it is either too bright, or too dark). In practice, it is common to store each R, G, and B value as an 8-bit integer, which means it can take any integer value from 0-255.

[note: hence the total number of colors an image can have is 255x255x255 ≈ 16.58 million; some computer displays are also capable of displaying each of these colors, and claim to be able to display "millions of colors"]

The basics of compression
The image of the lion fish in Figure 7 is a resized version of my original. The original contains 1920x2560 = 4,915,200 pixels. If we use up one byte (= 8 bits) for each color in each pixel, we shall need to store a total of 4,915,200x3 = 14,745,600, or nearly 15 Mbytes.

This is a pretty big file – and it poses several problems:
(a) If we need to store several such images, we will need a very large storage medium on my digicam; typically, digicams store images on a card called a compact flash card, which costs approximately HK$ 1 per Mbyte (in 2006). Storing large number of large images will drive up the cost of storage.
(b) When I need to process the file, e.g. transfer it from my digicam to the computer, or include it in my lecture notes, I face slow transfer rates and large file sizes.
(c) If the file needs to be transferred, e.g. sent on internet to a friend, then the large size makes the transfer painfully slow.

Therefore, it is useful to explore if the image can be stored using smaller files, but containing the "same" information – here, the word "same" may mean "identical", or "visually the same". In other words, we have a need to ***compress*** the image file.

Lossless vs Lossy compression

We are all familiar with the idea that files can be "compressed" by the use of utilities, such as programs that can ZIP the file; this process can reduce the stored size of the file by as much as a factor of 3. Such ZIPping is "lossless" – in other words, when we UN-ZIP the zipped file, we get back an identical copy of the original. Thus ZIPping is a process that can reduce the size of stored data without losing any information. There are many ways to achieve this; since we are talking of images, let's take a simple example of a black-and-white image as shown in figure 10 below.

Figure 10. A black-and-white image

This is an image made of 203x191 pixels. The color of each pixel can be stored as one number: 0 (if it is black), and 1 (if it is white).
In the uncompressed form, we could use just one bit per pixel, and therefore store the entire file in 203x191 = 38,773 bits = 38,773/8 Bytes = 4847 Bytes.
Alternatively, note that within the picture, there are long strings of 0's and 1's. For example, in the image, just above the boy's head, there is a large region of white space. In each row, there are approximately 200 contiguous 1's (we call this a string of 100 1's). We could just store this as 1x200 : this uses one bit to store the 1, plus perhaps 2 bytes to store the length of the string, for a total of 17 bits. Thus, instead of storing an array of bit-values, we can alternatively store the data in as an array of *pairs of values*: (color, length of string with that color). Of course, if the color changes very frequently in a picture, that is, the picture has a lot of changing colors, then this method may even take more space than the original file – however, in most images, even color images, there are many regions with long strings of equal valued pixels. Therefore, for real world images, this method compression is fairly efficient – on the average, reducing file sizes by as much as 2 or three times. This basic strategy is called ***run-length-encoding*** of the image.
Note that this scheme is lossless – in other words, we can convert the *raster format* data to the *run-length-encoded* format, and back again, without any loss of information.

Different formats
Based on the method of compression, and the way in which the image data is stored, there are several different formats in which images can be stored. The original (1920x2560 pixel) image of the lion fish of Figure 7, when stored in with RGB values in a format called bitmap format, or BMP, uses approximately 14 MBytes. BMP is a lossless format.

I then try to store it using a lossless compression format called PNG that uses a very sophisticated method of run-length-encoding. The resulting file size is 7.9MBytes – surely a significant saving, but maybe still too big for putting on the web.
Now we try a human factors approach to compression: if the objective is to view the file, then perhaps we can compress the file as follows: first get rid of the data that will not be visible to the human eye – this will reduce the total information in the file; then compress the resulting information using run-length-encoding. This strategy is ***lossy*** – in the sense that some information in the original file is lost, and cannot be recovered from the new file. However, since that part of the information was not easily visible to humans (or not important for viewing the image), then the reward can be tremendous.

One such compression method is used in storing image files using a format called JPEG (or JPG). Most images on the web now use JPG format. Let's first get an idea of the savings we can get, and the loss in quality, to get some motivation to understand JPEG better. The original lion-fish file, stored as a high quality JPEG file (very little information is thrown away) used 3.67 MBytes! If you view these two images (the 14MByte BMP format file and the 3.67MB JPG format file) on a computer screen, it is unlikely that you will detect any difference between the two files with your eyes. I also tried some lower quality settings, where more and more data is thrown away. If the highest quality setting is rated at 1.0, then the JPEG file size at quality=0.8 was 0.83MByte; at 0.6, it was 0.5MByte; at 0.4, it was 0.35MByte, and at quality =0.2, it was 0.2MByte). When viewed on a fairly nice computer monitor, it is only at a quality of 0.2 that the image starts to look poor. I have provided links to these images from the lecture notes web-page. But of course, most of you now have access to digicams, and you should experiment with these data formats yourself.

How does JPEG work?
Now that we have seen how great JPEG works in storing images, let us try and understand the ergonomics principles that go into the design of JPEG.

How to compress depends on how we see things – we want to reduce information, then we should first discard the information that is most difficult to see. How we see things really depends on two organs in the body: the eyes and the brain. The eyes sense, and the brain interprets. We would like to believe that our vision system works fairly well – while that's true, but the *brain also adds a lot of other things into the picture* before deciding what the sensory input from the eyes really means. This is the reason that our color perception can be fairly unreliable. To prove this point, let's look at some classic examples of optical illusions.

Example 1. The checker shadow illusion. In the figure 11 below, the squares A and B are perceived to have different colors, but in reality, they are both identical in color.

Example 2. The Koffka ring illusion. Again, a uniform colored ring is seen in front of different colored backgrounds. When the ring is split and moved, the two split halves appear to have totally different shades (though they do not).

Example 3. We are not even good at distinguishing black from white – in Figure 13 you will find yourself seeing black spots inside the white circles, where there are none!

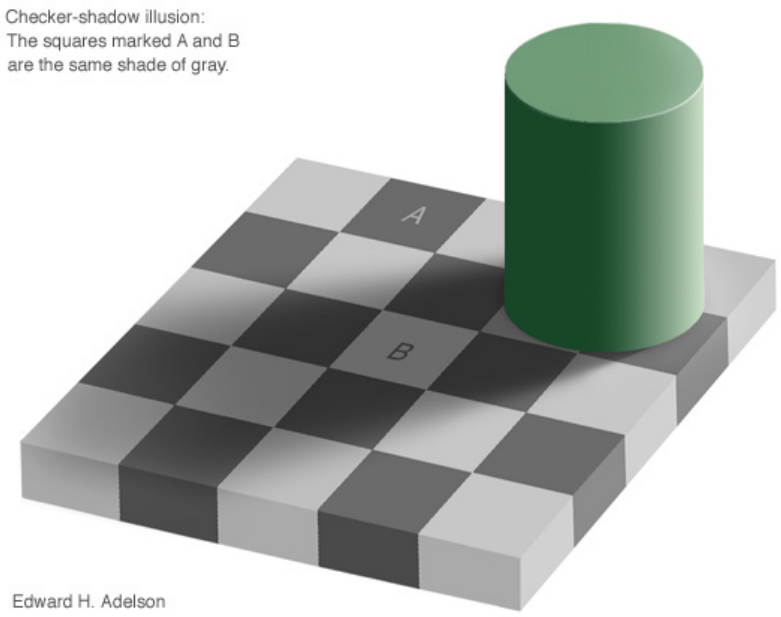

Figure 11. The Checker shadow illusion.

Figure 12. The koffka ring illusion

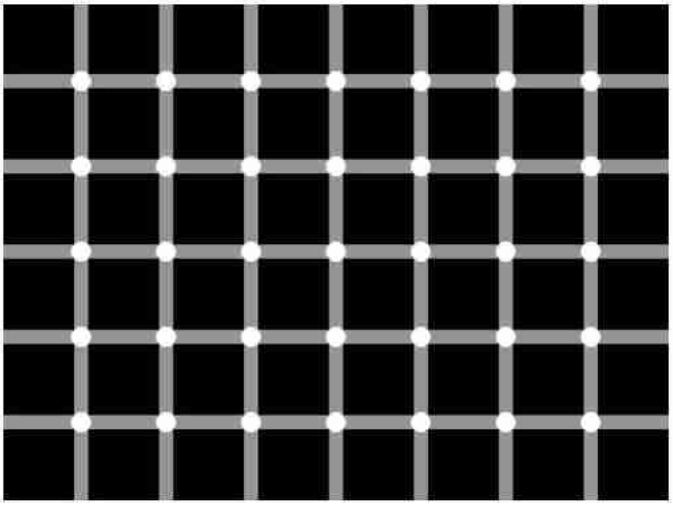

Figure 13. How many black spots are there?

A look at the eye

So really, what we see is not equal to the pure sensory input we receive from the eyes. So how is the eye really working? We know that visible light is composed of all wavelengths between 350 nm and 750 nm. Thus, what we call red is ~630 nm, Green is ~530 nm, and Blue is around 470 nm. But then, why is it that other colors are formed by combinations of R-G-B? Is it that combination of these colors give rise to new wavelengths (e.g. by interference)? But if that is the case, why do different proportions R-G-B give different colors? Certainly this seems to be conflicting with our experience that R-G-B in different proportions yield different colors. In order to understand this a little better, we need to look deeply into our own eyes. Figure 14 shows the basic structure of a human eye, with some details expanded.

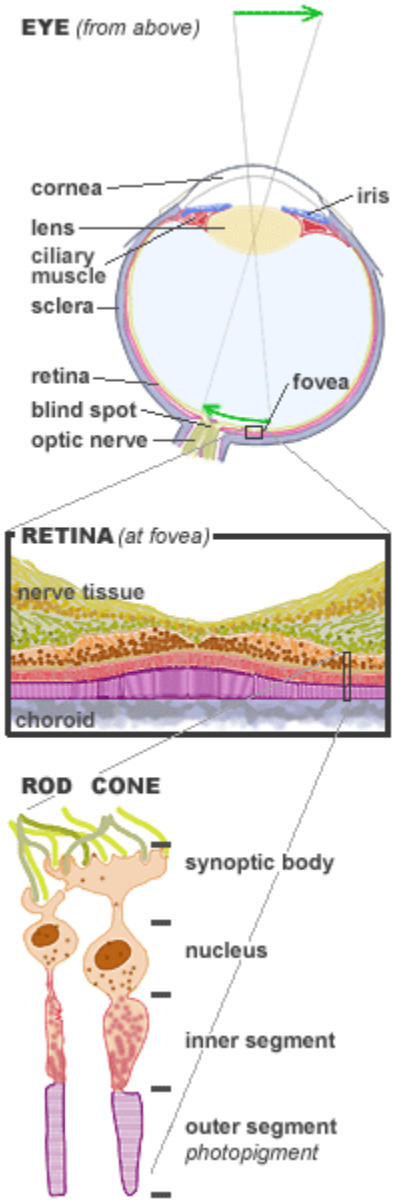

Figure 14. The basic structure of the eye [source: www.handprint.com]

What we “see” is an interpretation by our brain of an image that is formed on the retina, which is the screen of our eye. The image formed on the eye must be first sensed. This is done by special cells called photoreceptor cells. We have two types of photoreceptors, around 100 million rods, and 6 million cones are distributed across the retina (in a non-uniform way – a small region called the fovea has a large density of two types of cone cells, R- and G-, and this is where we focus to look at details such as contrast changes for edge detection). The rods are very sensitive, and are turned off in bright conditions – they are only used for vision in dark conditions (e.g. very low light rooms, night vision) – called ***scotopic*** vision. In scotopic conditions, we cannot detect any colors.

[TRY THIS: asking a friend to prepare sheets of different colors. Sit in a relatively dark room with your eyes shut for a few minutes. Open your eyes and let them adjust to the dark, and then ask your friend to show you the sheets one by one. You will find that you cannot identify their colors reliably!]

On the other hand, the cones work only in ***photopic*** conditions – conditions of brighter light. In scotopic conditions (e.g. after sitting in dark for a few minutes), they are turned off. As we shall see, these are the cells that help us to “see” colors. There are three different types of cones – each has a different pigment, which reacts to light differently (Figure 15).

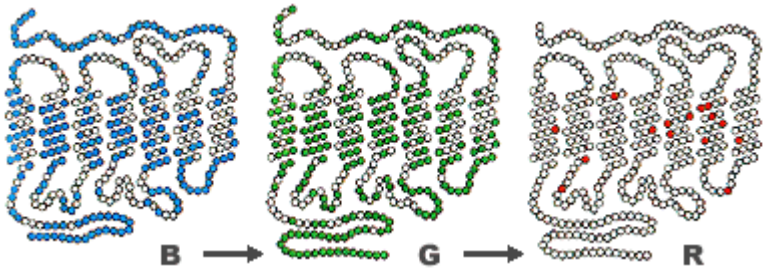

Figure 15. Photoreceptors in cones. The colored circles in B-cones indicate positions where the molecular structure is different from that of rhodopsin, the photopigment in rod cells; similarly, the colored circles in the G-cones indicate where they differ from B-cones, etc. R-cones are very similar to G-cones (hence only a few red circles).
[source: www.handprint.com]

The three types of cones react differently to different frequencies of light. This is shown in the figure below.

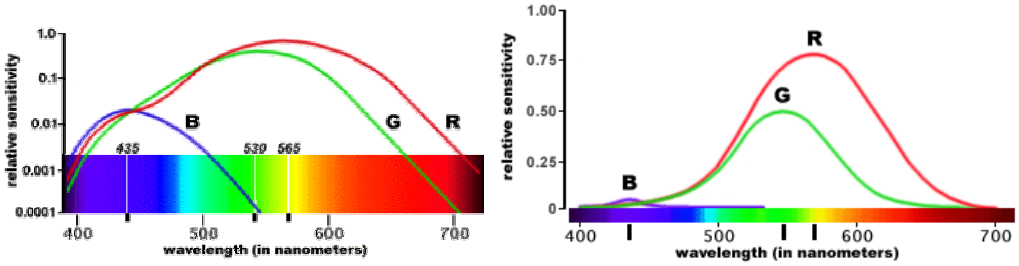

Figure 16. Sensitivity of different cones to light, (a) log scale, (b) linear scale
[source: www.handprint.com]

But what do we mean by sensitivity? The sensory response is fairly complex, and not well understood. We have 100 million rods, and 6 million cones, but only 1 million nerves going form the eye to the brain – in other words, on average, the signal from 100 rods and 6 cones is averaged and sent on one nerve fiber. Secondly, signals sent in the nerve are constant amplitude electric impulses – a higher intensity is signaled by sending a larger number of impulses in the same time duration. Thus there are two complicating factors. Firstly, the cones vary in sensitivity to any given frequency, but when they respond, the response is just a single impulse – a single impulse generated by a cone cell gives no information about the frequency of the light. Secondly, the signals from several cones are combined and sent on a single nerve fiber. So how can we perceive colors? Somehow, the brain must combine the information coming from the different cones and use it to figure out the dominant wavelengths in the light falling in each tiny region of the retina.

There are different theories of how our brain handles all this information. The ***trichromatic theory***, proposed by Thomas Young and later developed by Herman von Helmholtz, is based on the premise that the brain estimates the color of a given frequency of light by comparing the absolute value and the proportional responses of the three types of cone cells to the signal. In other words, response of the sensors of only three types can help us to distinguish millions of colors. Thus, each type of cone cell should have some response (however small) to each frequency of visible light (otherwise we would not be able to distinguish colors in the range where one type cell had no response). The theory explains some of the aspects of human vision (but fails to explain some phenomena). Since only three types of receptors in the eyes are used to interpret all different colors, therefore we can simulate (i.e. construct the same sensation) any color by mixing the correct intensity of light composed of three different frequencies. It is logical to select those frequencies to which our receptors are most sensitive – hence our choice of the RGB model. Without going into more details of these theories, let's explore some aspects of vision more pertinent to JPEG: hue discrimination and lightness discrimination.

Hue discrimination is our ability to distinguish between two spectral (i.e. purely of one frequency) light sources. Surely, if the two sources have almost the same wavelength, our cone cells will emit the same response, and we cannot distinguish between them. Since the response curves of the cone cells are quite different, it follows that in different parts of the visible spectrum, our ability to distinguish between different hues varies. For instance, in the red range, we cannot distinguish between two sources with wavelength that are 6 nm apart; however, in the yellow range, we can distinguish between two sources that are only 2 nm apart.

The other aspect is ***Lightness discrimination***. Lightness refers roughly to the "grey level" of a source. We process lightness either through response of rods, or through the combined response of the different types of cones (one theory, called the ***opponent processing theory***, claims that lightness is sensed by the addition of the response of the R-cells and the G-cells, which are much more sensitive than the B-cells anyway).

Various experiments designed to verify the trichromatic and the opponent processing models have succeeded in showing several important results:

(a) Our eyes are much more sensitive to lightness changes than to hue changes.

(b) Most of our sensory interpretation of lightness in bright conditions comes from the total stimulation of the R and G cells.
(b) Our retinal response to inputs of different lightness is proportional to the *ratio of their lightness* – this is a specific instance of a psychological phenomenon called ***Weber's law***. In other words, if we have an equal intensity light shining on three objects of lightness 1, 2, and 4 units, then the response of our eyes to the objects will be in amounts x, 2x, and 3x respectively (not 4x for the third object). In other words, for objects with low lightness, we can perceive relatively small variation is lightness; for objects with high lightness, we need much larger changes in order to be able to perceive the difference (this is the reason that that we cannot see stars in daytime).

How can this understanding help to compress digital images? The first step is to map the RGB values of a color to an equivalent form, as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This conversion is invertible – in other words, given any RGB values, we can find the corresponding (unique) set of YCbCr values; similarly, given the YCbCr values, we can compute the corresponding RGB values uniquely. Mathematically, this is equivalent to saying that the 3x3 matrix above has an inverse. Physically, the Y, Cb and Cr values have some meaning – the R and G components contribute most of the magnitude of the Y value, and, from observation (b) above, Y corresponds to the lightness of the color of a pixel; it is commonly called the ***Luminance*** value for the pixel. The other two values, Cb and Cr are measures of two different color components, and are called the ***Chrominance*** values. [Note: sometimes the YCbCr values are equivalently referred to the YUV values).

The following figure shows how the sensitivity of the average human eye to different channels (Y = Luminance, Cb :=: Blue-Yellow chrominance, Cr = Red-Green chrominance) – by sensitivity, we mean the ability to detect variations.

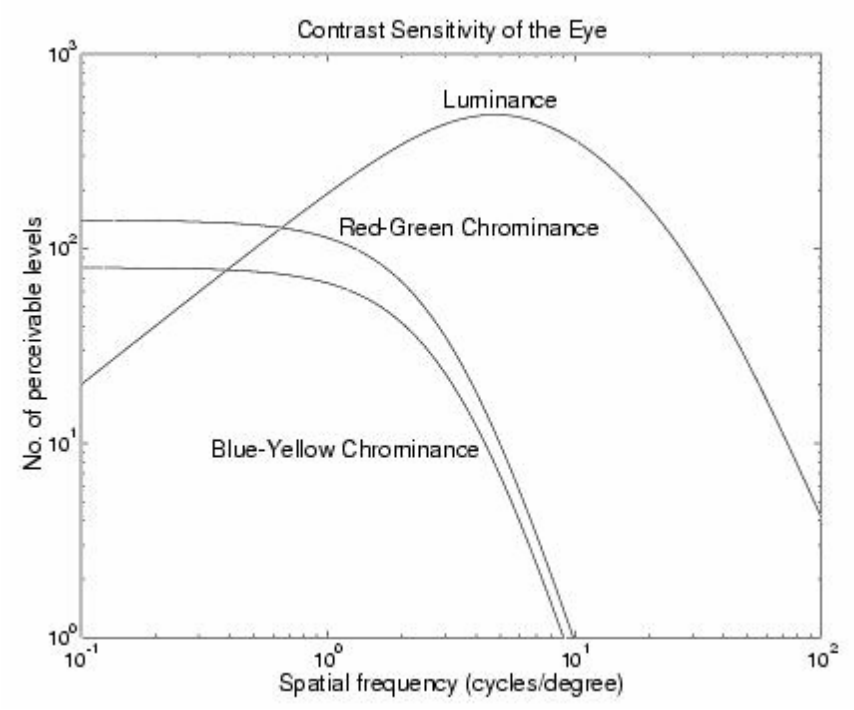

Figure 17. [source: Nick Kingsbury, Rice University]

Some explanation is necessary for this figure. The horizontal axis denotes spatial frequency. Imagine that we have a series of stripes of equal width, each with a different lightness. The lightness of a stripe is varying according to a sinusoidal function. A neighboring pair of stripes makes one "cycle". Thus, a spatial frequency of 1 cycle per degree is roughly equivalent to a pattern with 8.73 mm wide stripes viewed from a distance of 1m. For such stripes, we are able to distinguish between approximately 100 different levels of luminance, or chrominance (since the three graphs are nearly the same value at this level). However, as the frequency increases, i.e. the stripe width decreases, our ability to distinguish between different levels of chrominance falls rapidly; meanwhile, our ability to distinguish different luminance levels actually increases, until a frequency of around 5 cycles per degree. Beyond spatial frequencies of 10 cycles per degree (viewing from 1m, stripe widths of 0.8 mm or below), our sensitivity to different levels of intensity falls off; at these frequencies, we cannot distinguish any variations in chrominance values.

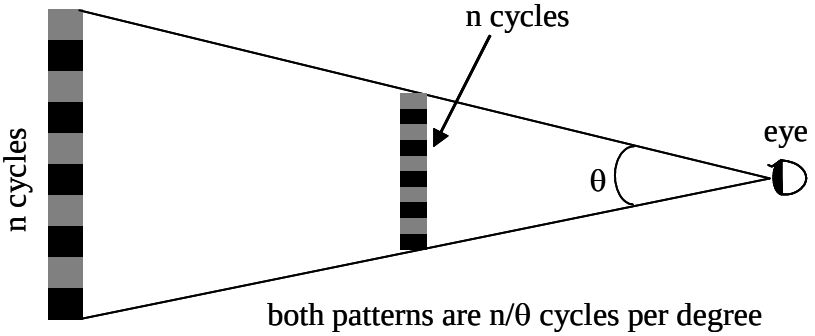

Figure 18. Relationship between viewing distance and spatial frequency

Just converting the RGB values to YUV does not, of course, reduce the amount of data. But it helps us to convert the data into components, some of which are "more important" than others – the goal now is to throw away some "irrelevant" data (data that is not easily visible to our eyes), and we shall throw away less of the Y channel, but more of the Cb and Cr channels.

In order to understand a typical method used by JPEG, we need to go a little bit into its mathematics, in particular, we need to learn a little bit about a very remarkable function called the discrete cosine transform, or DCT. The DCT is a sort of a 2-dimensional version of Fourier transforms. A simple, non-technical description of the idea is as follows: suppose we have a function as shown in the picture below. It is possible (under some assumptions that we shall ignore) to approximate the function as a sum of sinusoidal functions of increasing frequencies. The more approximating frequencies we use, the better is the approximation. Thus, we can choose to imagine that any function is really made up of a series of sinusoidal functions – the higher frequency components change very rapidly, and therefore give high-resolution details; the lower frequency components follow the approximate shape of the function. In the figure below, you can see how the complex function, T, can be approximated as a sum of simpler sinusoidal functions y1, …, y5. Note also that when the low amplitude, high frequency component (function y5) is subtracted from T to give T2, the resulting curve is rather similar to the original function T.

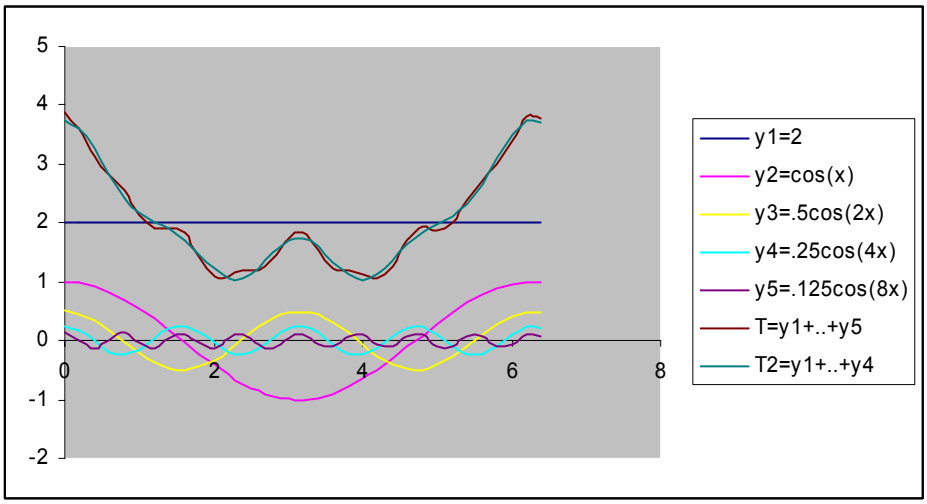

Figure 19. Approximating a function by a series of sinusoidal functions

The DCT function does a similar thing, except, in 2-dimensions. The DCT function is given by:

$$f(p,q) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} 4A(i,j)\cos\left(\frac{\pi p}{n}(i+1/2)\right)\cos\left(\frac{\pi q}{n}(j+1/2)\right)$$, where, *A(i, j)* is the Y (or Cr, or Cb) value of the pixel, depending on which channel we are working with.

The idea is as follows: First convert all the RGB values to YCrCb values. Now we can separate the Y, Cb and Cr components into three arrays, to be handled separately (hence

these are called the three channels). The entire image is partitioned into “blocks”, each block being a smaller array of 8x8 pixels. Each block is handled separately by the DCT program.

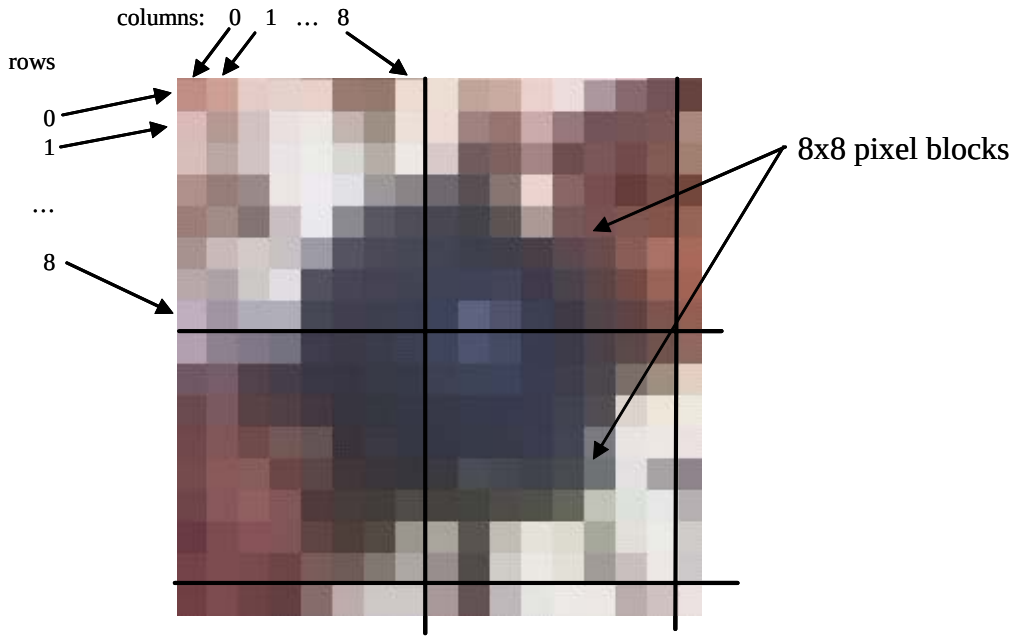

Figure 20. The JPEG “block” structure

Thus the DCT input is an 8x8 array of numbers, and its output is also an 8x8 arrays of numbers. However, the nice property of the DCT is that in the output array, the top-left hand values contain the low frequency data, while the bottom-right hand side contain the high frequency data (in other words, the coefficient of the low frequency components are in the top-left side, while the coefficients of the high-frequency components are in the bottom right hand side). This separation is useful, since we can now choose to just ignore some portion of the values in the bottom left side, and just replace them by zeros.

There are different exact mechanisms to do so in different implementations of JPEG. So we shall only look at the broad principles that are important: ***quantization*** and ***encoding***.
Quantization is the method by which the set of all numbers in the DCT matrix will be approximated by a smaller set of numbers; those coefficients that are “relatively small” are just set to zero; likewise, those numbers that represent higher frequencies that are irrelevant for vision are also set to zero. This process causes a loss of some information, but allows us to compress the data to a much smaller size. How to decide which coefficients are “relatively small”? This is done based on two things: (a) We look at all the numbers in the DCT output matrix. There are 64 coefficients ranging the minimum, say *a*, to the maximum, *b*. (b) Next, we look at the “level of compression quality” desired by the user. Based on these two factors, we set up a small set of coefficients (e.g. 16 different values, which are also in the range approximately from a, ,b). This new set of numbers is called a quantization *lookup table*. Finally, each coefficient in the DCT output matrix is replaced by the number in the lookup table that is closest to it. This approximation causes some loss of information, which is the only lossy step in the JPEG compression technique.

Finally, notice that we have divided the data of each pixel into three independent values, Y, Cb and Cr. Based on our understanding of human vision, we make a final adjustment. The Y value lookup table is larger than the Cb and Cr lookup tables. In other words, much more data in the Cb and Cr channels is thrown away, relative to the Y channel. This technique of using much lower resolution for the Cb and Cr values is sometimes referred to as sub-sampling of the chroma channels. Some of you who are familiar with video processing may have seen references to 411-compression (often stored as 411 video files) – in such files, the 8x8 DCT matrix is replaced by a series of 2x2 matrices, and for each 2x2 matrix, the each element retains its own (quantized) Y value, but the Cr and Cb values are replaced by the (quantized) level for the average Cr (or Cb) values of the four cells. Thus, for each four Y values, there is only one Cb value, and one Cr value.
Typically, depending on the desired quality of the JPEG file, quantization can allow us to reduce the total data by approximately 2-times to 5-times, without appreciable loss of quality.

After every entry in the DCT output matrix has been replaced by its quantized approximation as described above, the entire file is compressed using some form of run-length-encoding (a common method used in JPEG files is called Huffman coding). The actual process of this coding is not relevant from the ergonomics point of view, so we shall skip the details here. However, as we saw in our earlier discussion of run-length-encoding schemes, this step yields a further compression by a factor of approximately 2-times. Thus, on the whole, the JPEG compression method gives us a compression of somewhere between 4-times and 10-times over uncompressed RGB data. This is verified by our lionfish example.

3. Methodology and Tools in Ergonomics

From the examples and case study above, we can conclude that:

(a) Products designed for use by humans must consider factors such as ‘ease of use’, ‘comfort’ and ‘safety in use’.

(b) In order to define these terms, we must have a clear understanding of how the human body functions. Even for mechanical products such as chairs, shoes, etc., we must be able to measure the critical human dimensions (e.g. distance from bottom of foot to the thigh), and correlate these to some aspect of the product (e.g. chair seat height). The study of measurement of human body is called ***anthropometry***.

(c) There is a variation among the humans that will use the product -- therefore we need to understand the ***statistical variations*** of the anthropometric data.

Ergonomics is the study of ***optimization of product design***; typically objectives are maximizing the comfort, safety, etc. The tools used are anthropometry and psychology.

Typical ergonomic design approach may be as follows:

- the optimal product is designed based on anthropometric measurements;
- the statistical variations among the expected population that will use it are considered;
- the design is modified to allow critical parameters to be adjusted so as to ‘fit’ the need of a certain percentage of the population.