Midterm - Computer Vision

Conchari Cabrera Christian Ricardo
christian.conchari@ucb.edu.bo

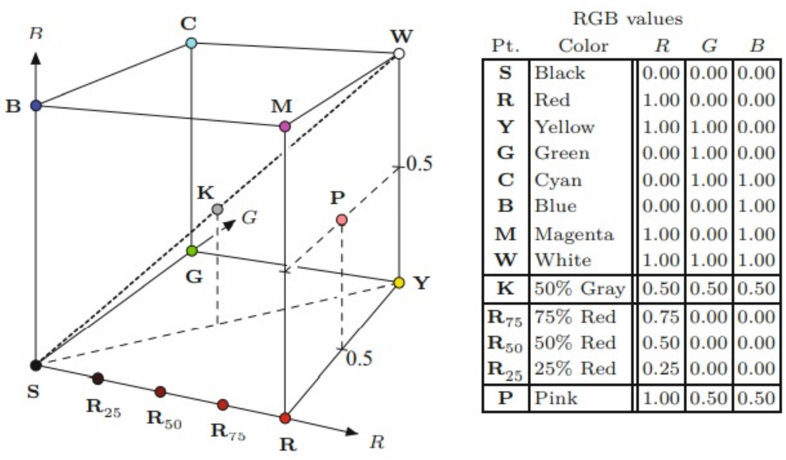
28/04/2021

1 Computer vision and image processing foundations

1.1 Why do we use an RGB based colour system for image processing?

The RGB is a very simple colour system, a basic knowledge of it is often sufficient for processing colour images or transforming them into other colour spaces. The RGB is an **additive colour system**, which means that all colours start with black (an empty array) and are created by adding the primary colours. To get all these different colours, you'd have to adjust the intensity of each of these beams separately. The hue and brightness of the resulting colour are determined by the intensity of each primary colour beam. OK, but let´s define why Red, Blue and Green are considered the primary colours, The explanation for this is that they correspond to the peak sensitivities of the colour receptor cells in our retina. It's worth noting that light has no colours, just wavelengths, and different wavelengths cause different reactions in our eyes' receptors, which our brain interprets it as what we know as colours.

The RGB colour schema encodes colours as combinations of the three primary colours: Red, Green, and Blue (R, G, B). This scheme is widely used for transmission, representation, and storage of colour images, we can visualise this RGB colour space as a 3D unit cube in which the three primary colours before mentioned for the coordinate axis. That component values are positive and commonly lie in the range of (0,255) in digital images.



RGB values

Pt.	Color	R	G	B
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

Figure 1: RGB Representation (W. Burger & M. Burge)[13]

Finally, we can respond to the question by stating that we use the RGB colour system because it is a simple way to display digital images with values ranging from 0 to 255 in three channels (Red, Green, and Blue), it is also simple to work with, and it is mostly composed of the three primary colours that our eyes respond to.

1.2 Why do we include a pre-processing stage for CV based systems?

We will have raw data in any process in which we will try to achieve some kind of result, whether it be signals, materials, or in this case images. Several times, this raw material will contain various characteristics or components that can be counterproductive to obtaining the desired result or even trigger bad results simply by being present. We can even think in the **GIGO** principle (Garbage in, garbage out), which tells us that if we enter wrong inputs, we will get wrong outputs.

If we go to a more practical way of thinking, whenever we want to capture real images of our environment, either in video or through photographs, we will be exposed to environmental noise, that no matter how high the quality of our instruments, it will always be present in smaller or larger quantities, This is when image pre-processing is needed, such as the use of a median filter to eliminate noise or histogram processing to enhance contrast. We may also mention image compression when working with large data sets and our computational capacity is restricted, and we could also discuss the need for dimensional reduction in certain cases, but in all cases, we are using some form of prepossessing to improve our output.

To answer the question posed, I will rely on the GIGO principle mentioned before, if we put wrong data into our system, we will not be able to obtain good or even correct results. That is why, in most cases, there is a stage known as prepossessing, which helps to improve the quality of our input data (in this case Images), so that not only the main processing stage of our method is made simpler, but also the results we want to achieve gets improved.

1.3 Explain how our visual system works.

Our visual system is composed of the **eye**, the **lateral geniculate nucleus** (LGN) and the **visual receiving area** in the **occipital lobe** of the brain. There are also some other processing area in the brain for visual signals which is called the **extra striate cortex**. Light reflects from the world and reaches our eye, forming an image on the retina. The cornea, which is responsible for 80% of the eye's focusing capacity, receives light first. It cannot, however, adjust its focus since it is kept in place. To do so, we use the lens, which provides the remaining 20% of our focusing ability, allowing us to respond to stimuli at various distances.

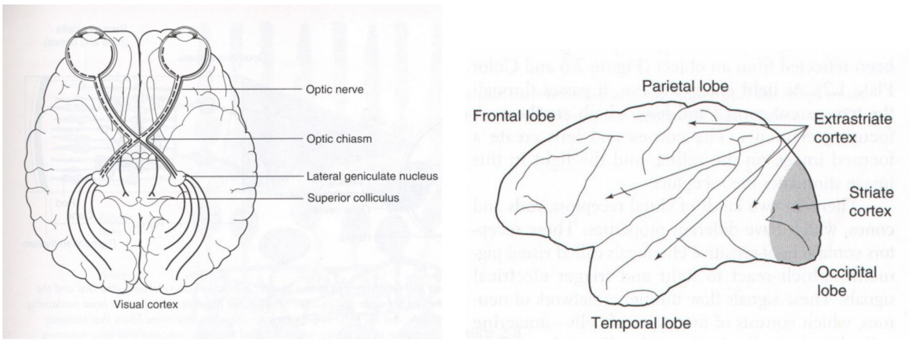


Figure 2: Visual System (Aditi Majumder)[11]

The retina, a thin layer of cells about the thickness of tissue paper situated at the back of the eye and rich in photo-receptors, forms the optical image of the eye. The pigmented epithelium is a dark pigmented layer that lies behind the retina. This absorbs the light that pass through the retina without being absorbed by the photo-receptors. The function of this layer is to prevent scattering of light behind the retina so that a sharp high-contrast image is enabled. Once the image is formed on the retina, the next step is to stimulate the receptors in the retina. The first layer is the receptors that contain light-sensitive chemicals to convert the light stimulus to electrical signals.

For a better understanding, is needed to be mentioned that there are two kinds of visual receptors, the **rods** and the **cones**. Thus the electrical signals generated in the receptors then flow through a network of neurons that consist of four types of cells - **amacrine**, **bipolar**, **horizontal** and **ganglion cells**. The axons of the ganglion cells forms the optic nerve which runs from the eye to the LGN.

Now, the inevitable question at this point is how light in the receptors is transformed to an electric signal. The inner segment of the receptors contain the nucleus and other cellular components. The outer segment contains billions of light sensitive pigment molecules. The pigments in the rods is called **rhodopsin**. When the pigment molecule absorbs the photon, it changes its shape through a biochemical process in such a way that the flow of electric current in and around the pigment molecule is changed. Therefore, the signal is transmitted through the different layers of the retina.

Part of the neural processing happens in the retina. This complex set of events in the outer membrane is called **pigment bleaching** since this is associated with a change in the colour of the pigment molecule. Once bleached, the pigments cannot absorb any more photons. They are restored to their prior unbleached state by actions of enzymes from the pigment epithelium after which they can again absorb photons.

As it was mentioned before, some neural processing occurs in the cells of the retina itself. Many neurons interconnected through convergence form what we call neural circuits. Neural circuits can be small comprising of a few neurons and can also be very large consisting of tens of thousands of neurons. To summarise in brief, the excitation of this neural circuits is what produces the interpretation of images in our brain.

Consider an analogy between an electronic image capture device (a camera) and our visual system to better understand the concepts mentioned above. Let's think of the retina as the camera lens, the photoreceptors will be the sensors that capture the information, the biochemical process by which the molecular pigments are converted into electrical stimuli will be the equivalent of an Analogue to Digital conversion (ADC), the neural circuits will be the electronic circuits where the processing is done and finally our brain will be the microprocessor where the final processing is done to obtain the desired result, in the case of our brain the visual perception is done.

1.4 Why do we require to enhance an image?

There are a variety of reasons why it is important to enhance an image's characteristics. Image enhancement can be understood as the procedure of improving the quality and information content of the original data before processing, common practices include contrast enhancement, spatial filtering, section colouring or others. For example, when taking an image with a camera or video, the illumination in the environment, the camera's quality, or other factors can result in a low contrast image, in which case there may be sections of the image that cannot be clearly distinguished from other sections of the image.

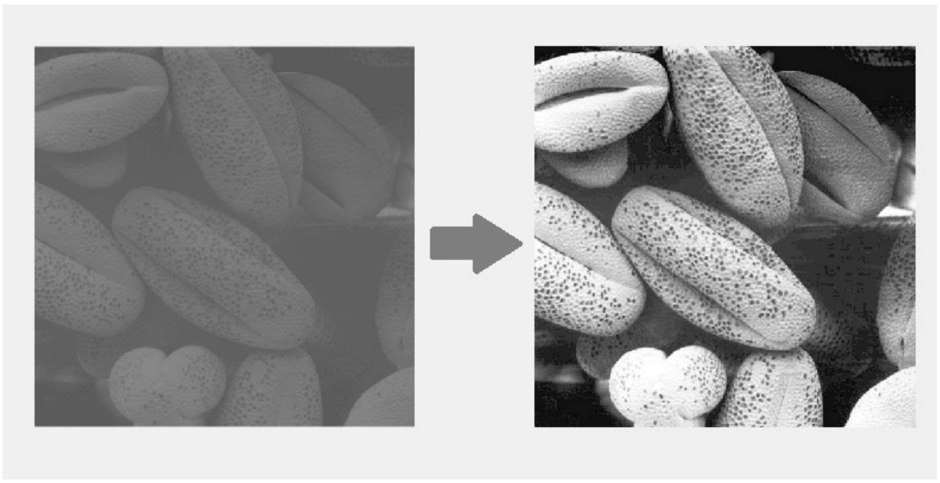


Figure 3: Contrast Enhancement

Once the image has been enhanced, better results will be obtained in further processing, such as finding edges or corners, even visually we can see a better quality image 3. Image enhancement may also be described as the use of processing to aid human visual analysis; for example, the colouring of different sections of grey-scale images based on characteristic points in the image histogram, allowing for the identification of parts of the image that aren't visible to the human eye.

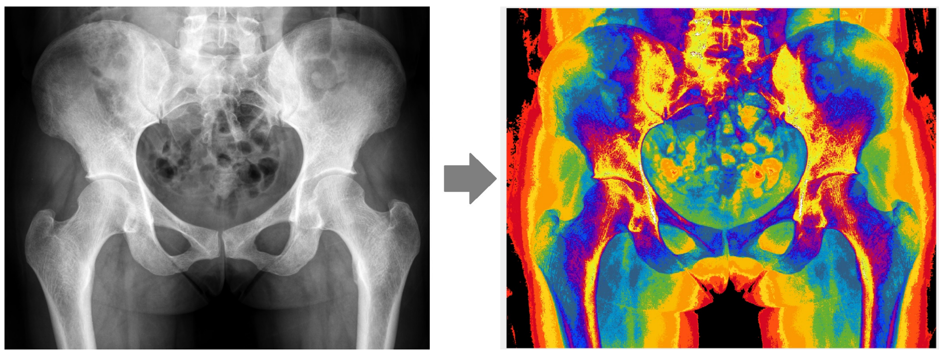


Figure 4: Histogram Colouring

In the image 4 we can see that some areas are more noticeable now that they are coloured, is a way of seeing that we have improved the information content of the original image.

OK, let's answer the main question, we need to enhance an image either as part of the pre-processing of our machine vision system or as the main processing in a histogram image colouring system. In both cases our main objective is to improve the quality and content of information of the raw image.

1.5 Why do we want to create binary images?

To begin with, let's define the binarization of an image as an operation by which we are reducing the present information, so that the only possible values are 0 or 255 (it is also possible to work with 0 and 1), so that our image will be composed only of intense white (255) and intense black (0). So what is the point of creating binary images? We can use this technique to separate objects or regions, which may be of interest to us, from the rest of the image.

Think for example of the **binary thresholding process**, which is commonly used to highlight contours of elements that we want to analyse in an image. The case could be that after applying a method to find the edges of an object in an image we want to binarize it in order to highlight the outlines found and the remnants or the rest of the pixels that we are no longer interested in, what would be happening is that the outlines would be converted to 255 and the rest of the pixels to 0.

Thus, the answer for the proposed question would be that we create binary images, through binarization processes, to be able to create masks, separate important elements from the rest of the image and highlight certain important image elements features, such as contours or corners.

1.6 Why would you use 32 bits of colour-depth?

Let's define what means by colour-depth. The Colour depth or bits per pixel is a concept in computer graphics that refers to the number of bits of information necessary to represent the colour of a pixel. The most common is that we find 16-bit or 24-bit colour depths, but with the evolution of computing and new graphics processors, we can now also find 32-bit colour depths. With 16-bit colour depth, computers and monitors can display as many as 65,536 colours, which is adequate for most uses. But if we want to see graphics with higher image quality, such as with better shadows, better transparencies and more convincing gradients, we could use the 32-bit depth that would support 16,777,215 colours.

It is clear that when it comes to increasing the colour depth, not everything will be good news, as you increase the support for more colours, more memory is required. Although this might not be a real inconvenience for modern GPUs, will it always be necessary to have the best image quality? If what we want is to impress the user with the best possible graphics, we will opt for 32-bit depth, but if what we want is to classify images using convolutional networks, clearly these last images that contain much more information will not be our best option, since in the case that we even end up compressing them, sacrificing a bit the quality of the image but also reducing the computational cost necessary to process them.

So why should you use 32-bit colour depth? As I tried to imply before, it depends a lot on the application, I would use this colour depth to show a final product where I want to impress the end-user, but I do not think I will use them for image processing, as it would entail a necessary computational cost, being that similar but equal or better results could be obtained by processing images of much less depth.

2 Image filtering and processing

2.1 Why would you require to filter an image in frequency domain?

We can see the representation of an image in the frequency domain by organizing the pixels according to changes in their intensity. We will define the high-frequency components as those pixels where there is a high rate of intensity change, such as corners or contours, on the other hand, we will define as low-frequency components, those pixels where there is a low rate of intensity change, such as it could be a single colour surface. To transform an image from the domain of space to the domain of frequency we will use the Fourier transform, where the low-frequency components will be in the centre of the image and the high-frequency components will be in the surroundings. So if we want to extract the high-frequency components of an image we may consider using a mask so that we only keep the central part of the image in frequency (such as a high pass filter) and in a similar way we could use a mask that only maintains the outside of the image in frequency to keep the low-frequency components (such as a low pass filter).

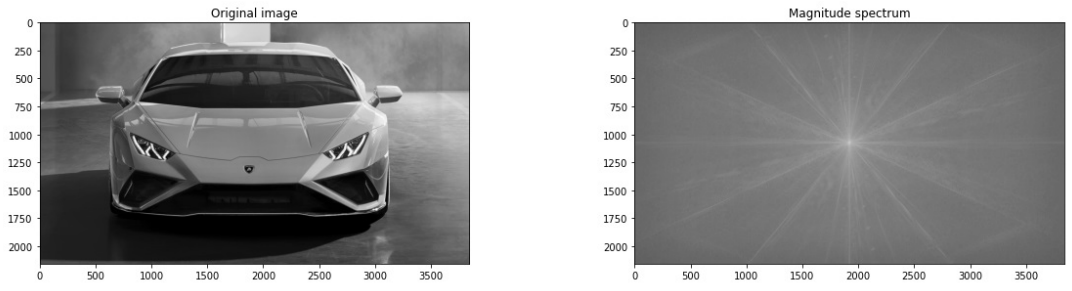


Figure 5: Frequency Domain

With all these concepts mentioned, being clear that we need a mask to filter images in frequency, we can vary its size and the regions it covers, in that way, of course, doing a thorough analysis, we could choose which frequency components to eliminate and which components to keep. . We could even play with masks in the shape of a hollow circle region (doughnut shape) to emulate what a band-pass filter would be in filtering electrical signals. So, we can achieve the following answer, we will need to filter an image in the frequency domain when we want to use high-pass filtering or low-pass filtering, controlling the components that we want to keep or eliminate, depending on the region that the mask covers. Now, it should also be mentioned that daily when we want to filter an image to eliminate noise in the image (high-frequency components) we will generally use a Gaussian filter kernel instead of transforming the image to the frequency domain and applying a mask to maintain only the central region (smoothed). Now let's not forget that a Gaussian filter also has a frequency representation that precisely coincides with a mask that only maintains a central region, therefore, it could be said that both concepts go hand in hand.

2.2 Explain the reason for filtering images.

We can find many reasons for which it is a good idea to filter the images that enter our system before processing them, one of these reasons can be found through the GIGO principle mentioned in previous sections. Many times when capturing the images we will introduce frequency components that can be detrimental to our processing to a greater or lesser extent. For example, if we wanted to perform a histogram processing on an image, if there was noise present we would have problems making the calculations and very possibly we would get wrong results. Something similar would happen if we had noise in our image and we wanted to find contours, the algorithm that we want to apply may find non-existent contours because of the noise present.

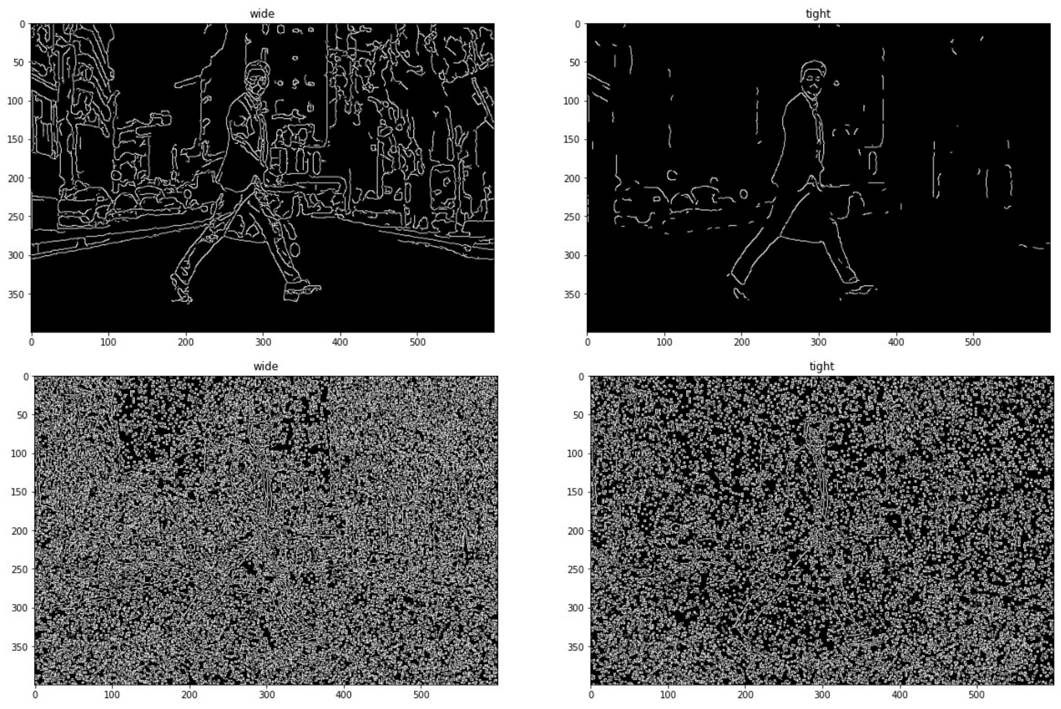


Figure 6: Filtered Image Find Contours vs. Non-filtered Image Find Contours

In the figure 6 we can see why it is important to filter an image with noise before processing it. Going the other way, we can also use filtering to highlight edges or in some cases even to isolate them from the rest of the image as it would be done with any other method to find contours. As would be the case of applying a gradient by adding the application of Sobel filters in the horizontal direction and the vertical direction. Well, summing up everything that was said, we need to filter the images to obtain better results, during the pre-processing stage (to eliminate harmful frequency components), as well as part of the processing, in the case of using high-pass filters. to highlight edges or find corners more easily.

2.3 Explain the difference between linear and non-linear filters.

Let's start by defining what a linear filter is about. Linear filters get their name from the fact that they combine the neighbouring input pixels in a linear weighted manner. This combination can be done by convolving the filter kernel with the image pixels in the spatial domain, or multiplying the filter kernel frequency spectrum with the image frequency domain representation. In addition, if we want to define more formally what a linear filter is about, we could say that they are those that meet the characteristics of a linear system, so that convolution is possible. On the other hand, non-linear filters can be defined as those in which their output is not given by a linear combination of their inputs, the most common example of non-linear filters are median filters. Where the median filter replaces every image pixel by the median of the pixels in the current pixel neighbourhood.

So in simple words we could say that the difference between a linear filter and a non-linear filter is that, Linear filters are those where the pixels of the filtered image can be obtained by convolving the filter kernel with the input image, or by multiplying their corresponding representations in Fourier frequency. On the other hand, non-linear filters are those where obtaining the output image cannot be done through the two aforementioned operations, instead, it must be done through a non-linear function.

2.4 Explain the differences between the Median, bilateral, and Gaussian filters.

The median filter is defined as a non-linear filter, instead both the bilateral filter and the Gaussian filters are linear filters. At this point we can find the first difference between these filters.

A median filter converts the input image to the output image using the statistical median method, all the elements of the array must be organized by minor to major, then choose the centre one, and change it in the original array centre.

$$median(\underbrace{a_0, a_1, \ldots,}_{\text{n values}} a_n, \underbrace{a_{n+1}, \ldots, a_{2n}}_{\text{n values}}) = a_n \tag{1}$$

In the other hand, a Gaussian filter converts the input image to the output image using the next set of equations.

$$g[i]_p = \sum_{q \in s} G_{\sigma_s}(\|p - q\|) G_{\sigma_r} * I_q \tag{2}$$

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2}{2\sigma^2})} \tag{3}$$

The bilateral is based on the Gaussian filter, but it´s added an extra function for the pixel intensity, they converts the input image to the output image using the next set of equations.

$$BF(I)_p = \frac{1}{W_p} \sum_{q \in s} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(I_p - I_q) * I_q \tag{4}$$

Where:

$$W_p = \sum_{q \in s} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(I_p - I_q)$$

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2}{2\sigma^2})}$$

As we can see from the aforementioned, we can find another difference between the aforementioned filters in the equations that are used to formulate them. We can find another difference between these filters in the different results obtained with each one, first, let's compare a Gaussian filter with a Median filter, the Gaussian filter will only blur the edges and reduce the contrast, on the other hand, the median filter will reduce the noises. keeping the edges relatively sharp. Now if we compare the Gaussian filter with the bilateral filter, as was said recently with the Gaussian filter some edges will be lost, while with the bilateral filter some edges will be preserved. Finally, we can find that a Gaussian filter and a Bilateral filter will process the images faster than a Median filter because processing additions and multiplications are much easier than processing sorting. So, from all the mentioned before, we can summarize the difference between the Median, Gaussian and Bilateral filters in the following points: the median filter is non-linear, while the other two are linear, let's also mention that the three are formulated mathematically differently, although the Bilateral filter could be thought of as a derivative of the Gaussian filter, finally, we can say that they differ through the results obtained with them, being that with the median filter it is possible to eliminate noise while maintaining the edges, while with the Gaussian filter it is only possible to blur the edges and reduce the noise, let's mention that the Bilateral filter stands out against the Gaussian, which reduces the noise and maintains some edges.

2.5 Explain the intuition behind the Sobel Operator.

The Sobel operator is one of the classic gradient-based edge detection methods, which are a simple case of trying to find the regions in an image where there is a rapid change in pixel intensity, we can also see the Sobel operator as an approximation to a gradient of an image, where we get the X and Y components separately. The next equations define two 3x3 kernels used to formulate the Sobel operator.

$$C_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad C_y = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5}$$

Let's analyze the kernel G_x to find the gradient in the x direction, let's see that we have negative values on the left side and positive values on the right side, and we are preserving the middle pixels a bit because we have the 2 values in between, which have a slightly higher weight than the others Basically, what we are trying to do is find the difference between two regions of our image, simply differentiating one from the other. To do this we will pass the Sobel kernel on each pixel of our image and this will give us a specific answer taking into account the pixels that surround it, and we will process the data in a similar way as shown in the following image.

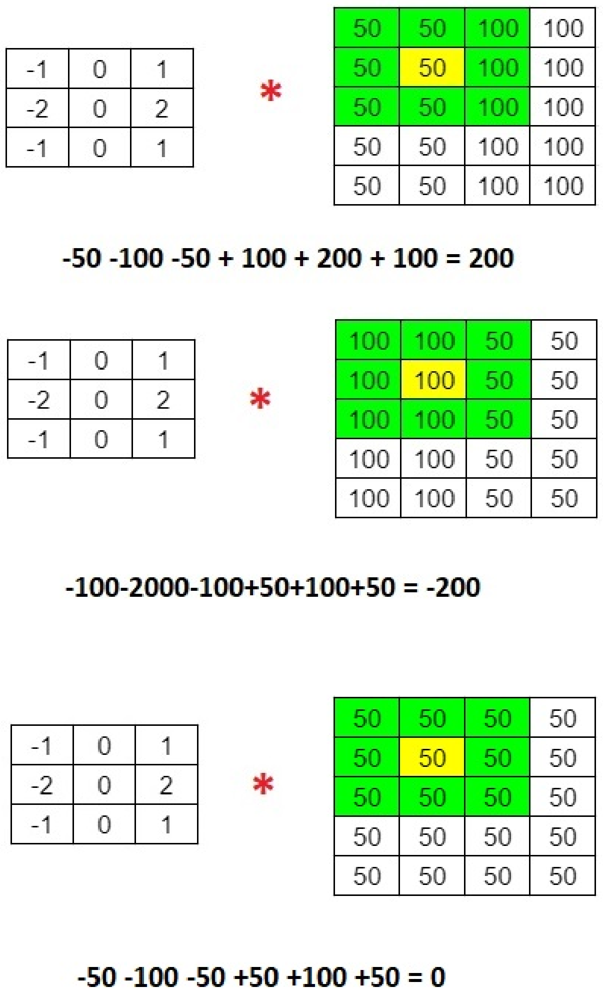


Figure 7: Sobel kernel example

We will scale the positive and negative values between 0 and 255, for which the negative values will scale to 0 and the positive values will scale to 255. Something similar will happen when we want to find the gradient for y direction, using the kernel G_y. In both cases we will obtain grayscale images where we will be able to see the large ones in the x and y directions. But what really interests us is to unite these two in a single image to be able to find the gradient of the image, for doing this we will need to find the gradient magnitude.

$$G = \sqrt{G_x^2 + G_y^2} \tag{6}$$

Now this magnitude will always be positive, if we obtain low values in G_x and low values in G_y, the magnitude will be a low value that will be scaled to 0 (black). If we get a high value in G_x and a low value in G_y it will be scaled to a medium value (gray), and if we get a high value of G_x and a high value of G_y it will scale to 255 (white), that way We can even see how big the border is in this section.

2.6 Explain how does the Canny Edge detector work.

Canny Edge detection method is one of the easiest and most useful ways to find the edges of an object in an image. It was developed to answer the question How can we consistently represent thick and thin edges? This, as this method allows us to detect certain edges based on their characteristics and the characteristics of the other edges that surround it or are connected to it.
We can divide the workflow of this algorithm into the following steps:

- Noise reduction;
- Gradient calculation;
- Non-maximum suppression;
- Double threshold;
- Edge Tracking by Hysteresis.

Let's start by explaining the first stage, **noise reduction**, Canny edge detection algorithm is especially sensitive to noise, so it is necessary to apply a blur to remove noise from the image before moving on to the later stages, the most common is to use Gaussian filter. In the next stage, the **gradient** is calculated, for this the Sobel operator mentioned in the previous section is usually used. By convolving the nuclei G_x and G_y with the image we will obtain the gradients in the x and y directions. Then the magnitude G and the slope θ of the gradient are calculated as follows:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta_{(x,y)} = \arctan \frac{G_y}{G_x} \tag{7}$$

As mentioned above, the total gradient values will be scaled between 0 and 255 depending on the intensity of the edge. Only the thin edges should be present in the final picture. To thin out the edges, we must use **non-maximum suppression**. The concept is straightforward: the algorithm iterates over all of the points on the gradient intensity matrix, looking for the pixels with the highest value in the edge directions. The purpose of the algorithm is to check if the pixels on the same direction are more or less intense than the ones being processed. For example, if we have a pixel (i, j) being processed, and the pixels on the same direction (i, j-1) and (i, j+1). If one those two pixels are more intense than the one being processed, then only the more intense one is kept. If there are no pixels in the edge direction having more intense values, then the value of the current pixel is kept. The result will be the same image, but with thinner edges.

In a next stage, the **double threshold** step identifies pixels with such a high intensity that we are certain they contribute to the final edge, pixels with an intensity value that is not high enough to be considered solid, but not low enough to be considered non-relevant for edge detection. pixels that aren't important to the edge, and pixels that aren't relevant to the edge detection. and pixels that are considered as non-relevant for the edge. The double threshold is used to identify strong pixels (those with an intensity greater than the high threshold) and non-relevant pixels (intensity lower than the low threshold). All pixels with an intensity that falls between the two thresholds are labeled as weak. Finally, hysteresis transforms weak pixels into strong ones depending on the threshold results, if and only if at least one of the pixels around the one being processed is a strong one.

2.7 Why would you rather use HSV instead of RGB?

Let's start by defining that the HSV colour system defines a colour model in terms of its components Hue, Saturation, Brightness. Working with the RGB result is very useful for images/videos with all-colour sections, but it can cause problems when selecting sections with varying brightness, shadows, or colours. For example, if we wanted to detect a traffic sign of a certain colour using a camera, many factors could influence it, let's think about shadows, lighting, etc. The first solution we would think of is trying to define Threshold values to detect colours. within a range of values, which would correspond to the colour of the image, but to do this we would need a lot of time and effort. To simplify our work we need a method that does not require so many parameters, and that is where we can choose to use HSV. With this new method, we describe colour in a much simpler way, since now we only need to transform the colour tone to now be able to capture the desired value.

Let's think about another example, now we want to detect if in a landscape image it is day or night, using the RGB colour system we would have to analyze the colours of the sky, the different colours of shadows and others in some way and with a lot of effort to determine if it is a landscape by day or by night. On the other hand, if we use the HSV colour system, the most differentiating parameters between day and night will be the Value (Brightness) channel, which tends to tend to 255 for daytime landscapes and tend to 0 for night landscapes.

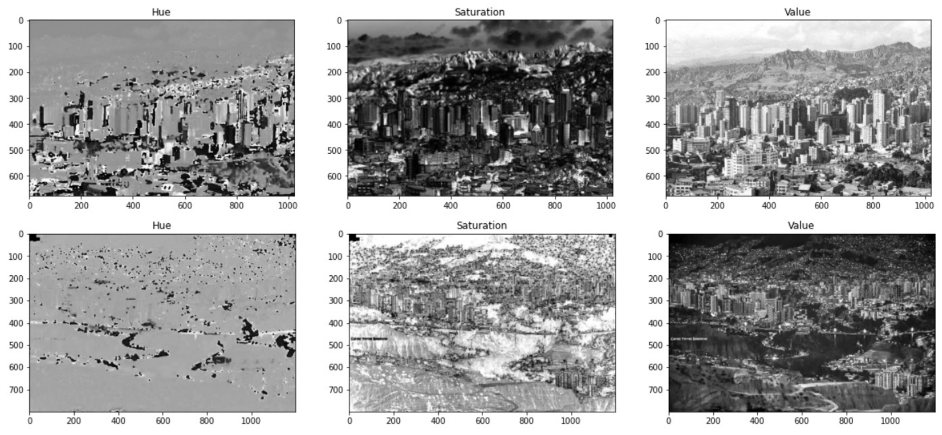


Figure 8: HSV System Image Representation

Once again, to answer the question posed, I will resort to saying that using HSV instead of RGB depends on the application we want to implement and the conditions in which we are going to work. If what we want is to detect an object of a certain colour in an environment where it can vary, either due to lighting, shadows, camera quality or background noise, it is better to use the HSV system that it offers us. an easier way to solve this problem, where we would only have to vary fewer parameters to achieve our objective, or if it were the case that in reality what we would like to detect is the amount of brightness of an object in an image, it will be much better for us to work with HSV.

2.8 Why do we require to improve Generalized Hough transform method?

The generalized Hough transform (GHT) is the modification of the traditional Hough transform using the principle of template matching, that in a nutshell, it's a digital image processing technique for locating tiny sections of an image that fit a reference image. The GHT main objective is to find the reference point that can be seen as the reference centre of a complex shape. Another important point to mention is that the GHT uses the R-table, whose main purpose is to relate the edge directions with all the vectors Rk = (rk, a), which are two orthogonal scale factors.

Now that we are clear on how GHT works, we can mention some of the problems that it has. The principal problem of working with the GHT is that it's very **computationally expensive** if we want to work with not only detecting the same figure of the reference, but we also want to detect similar figures scaled in size or with different degrees of rotation, then we will have to add two more dimensions to our analysis and this will greatly increase the necessary processing resources. For example, if we wanted to work with images in real time, we would not be able to use GHT because of its high cost.

Over time many improvements were made to GHT, for example in a publication by Du-Ming Tsai in 1997, where he presented an improved two-stage GHT procedure for the recognition of overlapping objects. Or the publication presented by Clark F. Olson in 1998, where he analyses the improvements that can be gained in the generalized Hough Transform method for recognizing objects through the use of imperfect perceptual grouping techniques. I mentioned the previous researches in order to show that over the years different improvements on the GHT algorithm have been implemented for specific applications.

So, why do we need to improve the GHT algorithm? As mentioned above, this algorithm needs a reference figure to detect similar figures in an image, this method starts to give problems when the figure that we want to detect is rotated or scaled with respect to the reference image, to solve this problem, the required computational cost is increased. So, in order to be able to use this algorithm in current applications we need to improve it in many aspects. At the same time, let's not forget that this algorithm is competing with more modern methods such as FAST, SIFT or ORB with which good results can be obtained even working with real-time images, therefore, in order to be used, the GHT algorithm needs to be improved.

2.9 Explain how is morphological transformations related to the creation of binary masks.

Binary masks are usually used in applications of image segmentation by colour or edge detection by thresholding edges within a certain range of thickness, in either case the result we obtain is a mask containing the information we require to analyse, the sections that correspond to a certain colour, or the contours of a figure.

Now, once the masks have been obtained, it is possible that for different reasons, we obtain masks that do not cover the sectors we require, leaving empty spaces or leaving remaining sections that will not contribute anything to our analysis. In this case, we will need to apply morphological operations to improve our mask and in this way achieve the objective of our application. For example, to remove remnants we will use the erosion operation, and to fill incomplete masks we will use the dilation operation.

So, we can say that, morphological operations are related to the creation of binary masks, in that they are necessary steps to improve them in order to cover the areas that need to be analysed, and thus to obtain better results in subsequent processing steps.

2.10 Explain the expected issues of colour-based segmentation.

When we talk about image segmentation based on colour, we refer to techniques that we will use to separate different figures within an image based on the colour composition of its pixels. To carry out the segmentation by colour we can use different techniques such as selection by colour ranges, or by comparing distances to a standard colour.

The first problems that we could encounter when working with colour segmentation are those that are generated at the moment of capturing the images, such as background noise, bad lighting, shadows, or other problems that can cause confusion when segmenting the images by colour, for example, if we have an object of a dark colour, we could confuse it with shadows of other objects in the image. While these problems can be solved by changing the colour system from RGB to HSV in order to handle the hue and brightness parameters, it is much more common to work with RGB values. Another problem that could arise is when defining the parameters for the colour ranges to be segmented, being that finding the range of values that can cover all the colour values present in the image can be an arduous task, another point that must be taken into account is that the figure must be of a solid colour or at least with similar colour tones, because if a figure is composed of more than two colours, we will not be able to detect it completely with a single mask. So, with all the mentioned before, we can summarise the main problems of working with colour segmentation problems generated by the noise and illumination of the images, in choosing the right parameters to define the colour ranges to segment a certain object, being that if we want to work with objects that are composed of several colours we will have to use multiple masks to cover the whole object.

3 Visual Information Compression and Analysis

3.1 Why lower dimension images are related to fast processing times?

We can view an image as a combination of three arrays: an array of pixels in the red channel, an array of pixels in the green channel and an array of pixels in the blue channel. Each of these arrays will have a specific number of columns and rows. So we can agree that an image will be a high dimensionality element, this dimensionality depends on the information it stores, i.e. if it is a colour image with 3 RGB channels, or if it is a greyscale image with only a one-pixel channel. We must also consider the number of columns and the number of rows it has, in short, the relationship between so many values is what produces the high dimensionality mentioned above. We can see this idea more clearly in a graphical way in the figure 1
Let us now mention the concept of computational cost. We can define it simply as the amount of time and memory space required to arrive at a solution by executing computational code. There are different ways to measure computational cost, one of them is through Big-O notation. Where we can find different functions to describe the computational cost, linear, logarithmic, quadratic polynomial or exponential.

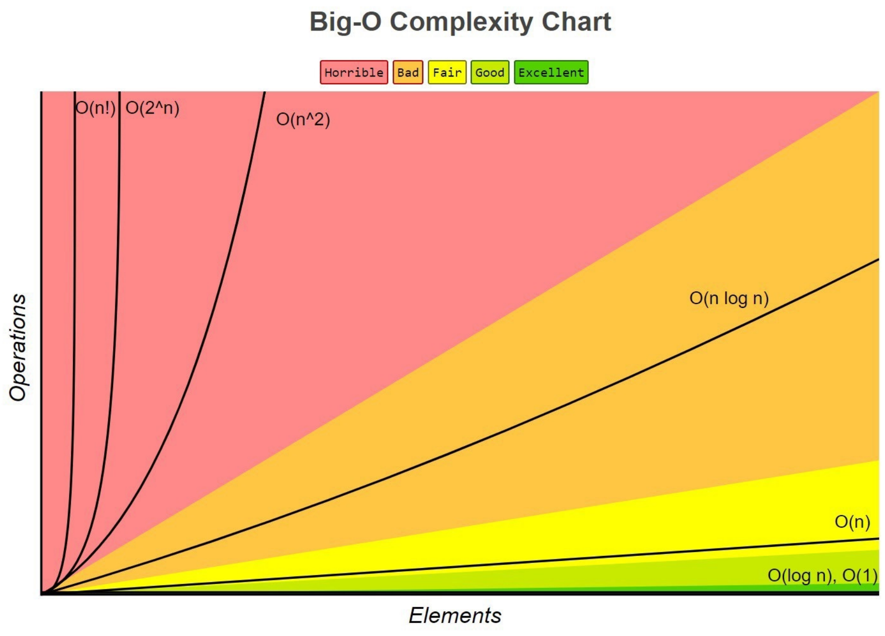


Figure 9: BIG-O Notation Computational Cost Functions (Leihua Ye) [7]

Now let me relate the computational cost to the high spatial dimensionality of an image. There is a quadratic relationship between the computational cost and the number of dimensions of the data we are processing. This implies that the higher the dimension of data we want to process, the higher the computational cost, and therefore the more hardware resources we need. In the case of images, the larger the dimensions of our image, the higher our computational cost will be. Thus, to answer the question posed, low-dimensional images have a lower computational cost and therefore, less processing time is needed to generate a result.

3.2 Explain the encoder - decoder couple mechanism for compression.

Let's start with the encoder mechanism, first, let's note that it is composed of three stages: Mapper, Quantizer and Symbol coder. Let's look at each one in more detail, in the **Mapper** phase what we do is to

reduce spatial redundancies by grouping pixels that have similar characteristics, it is related to compression algorithms such as the discrete cosine transform. Further in the **Quantizer** stage, we reduce the precision of the mapper's output, i.e. if we have decimal point data we transform it to integer data. Finally, in the **Symbol coder** stage, we will take the output of the preceding stage and generate fixed variable codes, to represent the information in a binary notation for processing by the computer.
Now let's talk about the Decoder mechanism, which consists of the Symbol Decoder and Inverse mapper stages. Let's start with the Symbol Decoder, which in a few words is the inverse process of the previously mentioned Symbol Coder, in this case, we decode the information to prepare it for the next stage, the Inverse Mapper, where we do the inverse process related to the one performed in the Mapper stage.

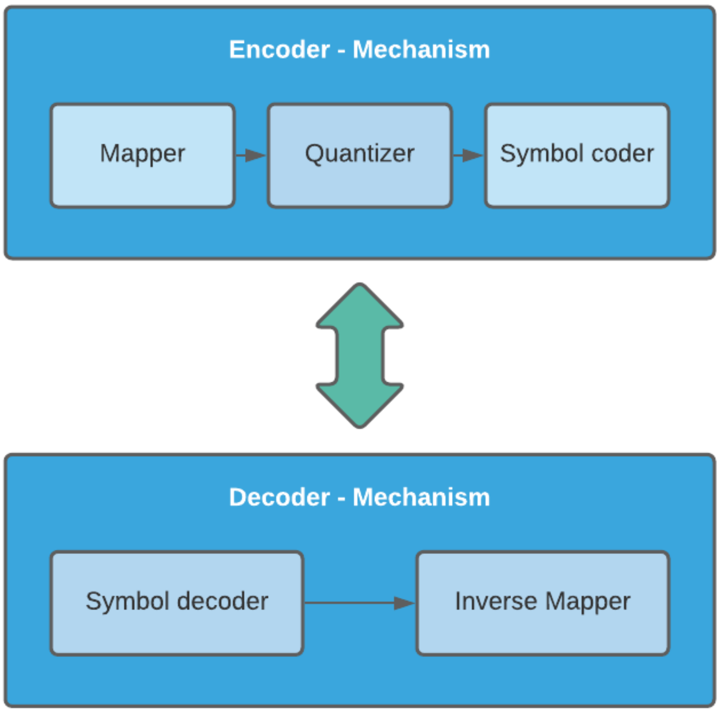


Figure 10: Encoder - Decoder Mechanism

To answer the question posed above, we can use the figure 10, as we can see they are two interrelated processes, starting because basically, the decoder mechanism uses the reverse steps to the encoder mechanism. For example, we will have an input image, we will apply the Mapper stage to compress the image and reduce the redundancies of the image, followed by the quantization stage and finally, we will encode the information to be able to store it. If later we want to view the image we will use the decoder mechanism, where we will first decode the information we have stored, to apply the inverse process to the one applied during the Mapper stage, in this way we will be able to view the image.

3.3 Why is JPEG commonly used?

Although JPEG is known as an image file format, it is actually a compression algorithm that works with the discrete cosine transform. It is a lossy compression method, which subdivides the image into 8x8 blocks to reduce information redundancies. The use of this compression algorithm works best on photographs and paintings of realistic scenes with smooth variations in tone and colour, which may be one of the reasons why it is so widely used. We could answer the question by saying that, JPEG is such a widely used format, because of its very efficient compression algorithm, thanks to this it is very light in memory and perfect for web use, it could also be because as mentioned above it is usually the most used compression mechanism and format in digital cameras and so many of the images captured by cameras are in this format.

3.4 Explain how JPEG can be used to compress without losses.

When we say **lossy** compression, we're referring to the fact that the compression will also reduce some of the image content. We'll go through some of the stages of JPEG compression implementation to see if we can avoid losing details. Colour conversion and sub-sampling will be performed in the first process, beginning with RGB data and dividing it into luminance and colour components. The amount of data is not decreased in this step since only the representation of the same information changes, but since the human observer is much more sensitive to intensity information than colour information, colour information can be sub-sampled without significant loss of information from the visible image. That is why it is said that it uses a perceptual model based on the human psycho-visual system, discarding information imperceptible to the human eye.

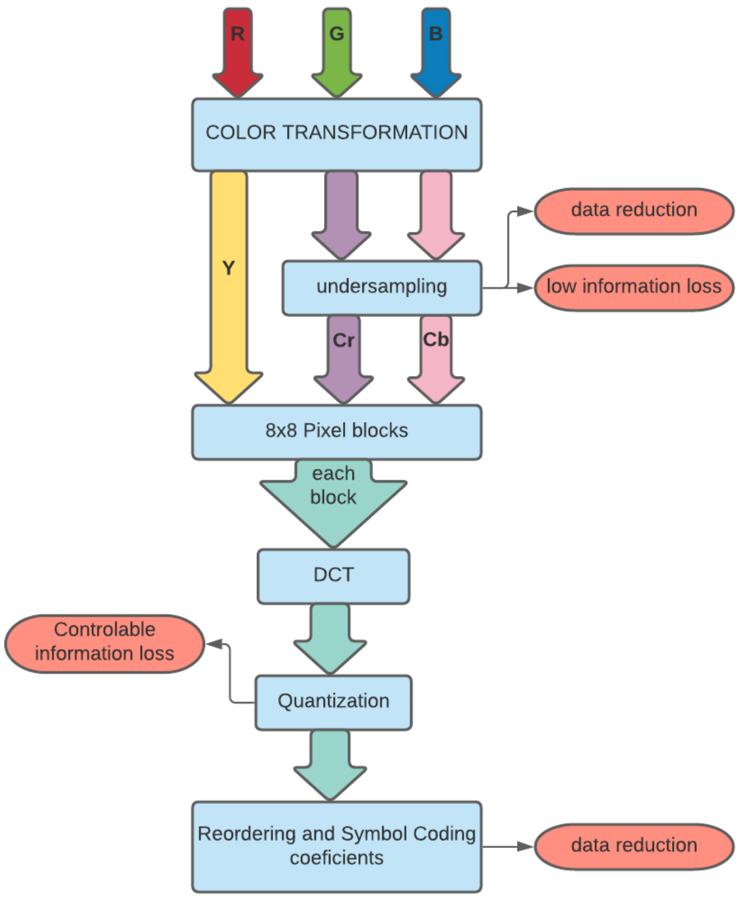


Figure 11: JPEG compression steps

The DCT transform is applied to each of the sub-sampled luminance and color elements, which are divided into 8x8 blocks. The coefficient of the spatial frequencies for the vertical and horizontal orientations describes the image quality after transformation. In the frequency domain, we have to store 64 frequency coefficients, so far we have not lost data yet. To reduce the amount of data required to store the 64 coefficients, these are quantized, depending on this step more or less information can be lost. Because of quantization, more coefficients are reduced to zero, and as mentioned above, these coefficients with value 0 will be found in the higher frequencies rather than in the lower frequencies. In this way, the amount of data is also significantly reduced, without loss of information perceptible to the human eye. Summarizing all of the above in brief words, lossless JPEG compression can be used to reduce information that is barely perceptible to the human eye, instead of reducing all information equally.

3.5 Why does the lossless compression techniques cannot achieve higher compression rates than the lossy compression techniques?

The challenge of encoding information with the fewest resources possible is one of the key concerns in information theory. Data compression is the term for this operation, and it can be done in two ways: lossy or lossless, depending on whether the original data can be retrieved with or without errors. Although lossless compression is required in many applications, compression ratios obtained with lossless techniques are significantly lower than those possible with lossy compression. Typically, depending on the image, lossless compression ratios range from about 1:2 to 1:3, relating this with a compressión percentage of 20

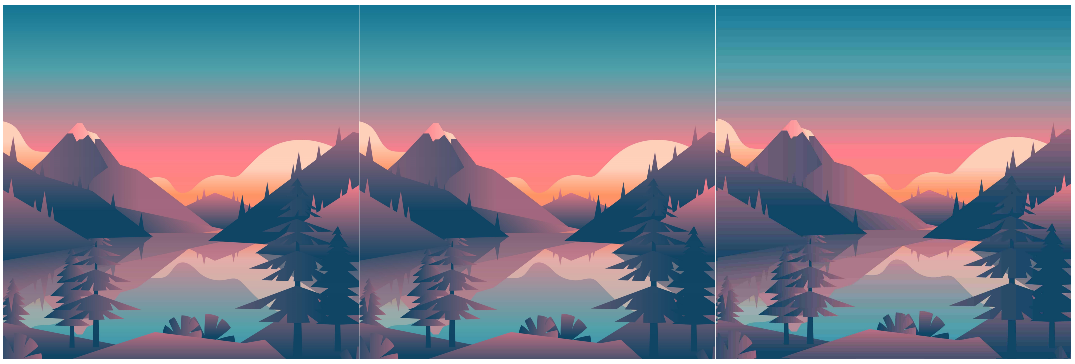


Figure 12: Different JPEG compression ratios

In the figure 11, the first image on the right is the uncompressed image, the next is the compressed image with a 20% reduction and the third is the same compressed image with an 80% reduction. As we can see, as we increase the compression ratio, the loss of information will become more evident, we can even notice that in the third image the above-mentioned blocking pattern appears. It is evident that JPEG compression has a limit to avoid the lossy compression. Most sources define the limit of compression in information theory as entropy. While this is a complex and difficult concept to explain, I will briefly refer to it as the uncertainty, noise or disorder that a system releases. We can find that a considerable part of the digitised information is noise, which by definition cannot be compressed, as it is random and has no relationship between its values. Let's think of a grey card, to the human eye we will see a solid colour when digitising it if we enlarge a sector of the digitised image, we will notice that it is not a solid colour, and in reality, there are darker or lighter pixels randomly distributed, in other words, noise has been added, we can find a measure of this noise through the amount of entropy per pixel. This amount of entropy per pixel will remain constant in the images we digitise, and since noise is incompressible, no matter how much we compress the image the noise will still be present. Finally, through lossless compression, we will reach a point where we can no longer compress the image, because of the aforementioned entropy, so to further compress the image we will have to start reducing relevant information, as well as losing information, and that is why lossless compression methods cannot reach considerably high compression levels.

3.6 Explain the low-pass effect that JPEG produces when aiming at very high compression rates

In the last sections was mentioned that the data reduction is done using the sub-sampling of the colour information, the quantization of the DCT coefficients and the Huffman coding. For a high compression rate configuration, the sub-sampling is done and the quantization array is selected to force most coefficients to 0. In that case, the image gets visible frequency components after decompression.

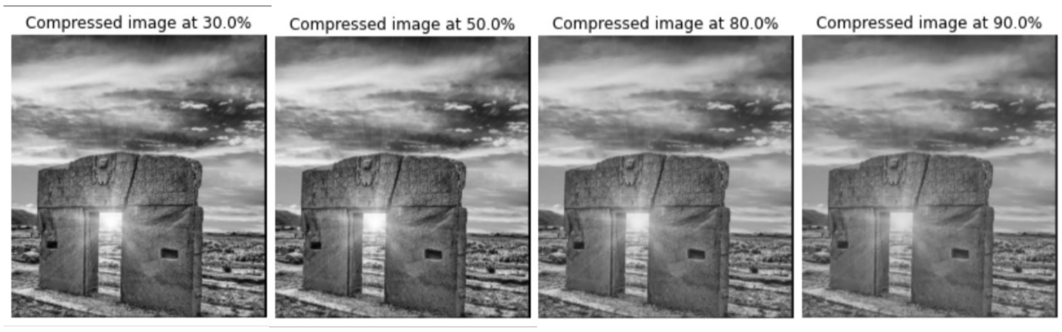

Compressed image at 30.0%
Compressed image at 50.0%
Compressed image at 80.0%
Compressed image at 90.0%

Figure 13: JPEG compression rates

The greater the compression rate applied to the image, the greater the notoriety of artefacts leftover from the compression process. Looking at this from a practical perspective, as we increase the compression percentage we will begin to notice components in the form of blocks, which are remanents of the 8x8 block segmentation to apply the DCT conversion, the appearance of these remaining components is known as blocking. So, understanding a low-pass effect as the attenuation of high-frequency components in the image, which are edges or sharp changes in tone, generating a kind of smoothing in some sectors of the image. It is also necessary to mention the appearance of remnants of the subdivision of the image into 8x8 blocks during compression, which become much more visible when the compression rate is increased.

3.7 Explain the main dierences between JPEG and JPEG2000.

JPEG2000 compression is an image compression and encoding standard, based on the JPEG compression algorithm, and was created in 2000, hence its name. This new method is based on the use of the wavelet transform instead of the discrete cosine transform used in JPEG compression.

Reviewing a little bit the steps of the JPEG2000 compression, using as reference the work of M. W. Marcellin, M. J. Gormish, A. Bilgin & M. P. Boliek [6]. The first step in the JPEG2000 compression algorithm is to **divide the image into rectangular, non-overlapping tiles** on a regular grid. Components with different sub-sampling factors are tiled concerning a high-resolution grid, which ensures spatial consistency of the resulting tile components. For these initial parts, two transforms are defined, **the YCrCb transform** commonly used with original JPEG images and the **Reversible Component Transform** (RCT) which provides similar decorrelation, but allows lossless reconstruction of all components. In the next stage, for each tile, a dyadic **wavelet transform** is performed using the floating-point wavelet transform or the integer wavelet transform. The **wavelet coefficients are then subjected to uniform scalar quantization** using a fixed dead-zone around the origin after transformation. This is done by dividing each coefficient magnitude by a quantization phase size and rounding it down. Each sub-band (of tile) is then subjected to a **packet partition** after quantization. Each sub-band is divided into standard non-overlapping rectangles by the mentioned before packet partition. Finally, code blocks are generated by dividing each packet partition position into non-overlapping rectangles of the same size. For entropy coding, each code block enters into an entropy coding stage separately. Context-dependent binary arithmetic coding of bit-planes is used for this coding.

With all the above mentioned in the previous paragraph, we can notice that a clear difference between JPEG and JPEG2000 is the way they are implemented. JPEG 2000 proves to be a much better compression tool. Due to the way the compression engine works, JPEG 2000 promises a higher final image quality, even when using lossy compression. Finally, in a substantial way we could mention that the main differences mention between JPEG and JPEG2000 are in the way their compression algorithms are implemented, in the ratio of information loss vs. compression ratio, and even mention that JPEG2000 can implement lossless compression at high compression ratios, something that we saw in previous questions JPEG is not capable of.

3.8 Why is compression so important for transmission?

As has been mentioned in the last sections, the objective of image compression is to reduce the amount of information present in images, without losing the visual quality of the image itself. If we reduce the amount of information present in the image, we will also reduce the computational cost necessary to process it, the cost to transmit it and the bandwidth necessary to transmit it through a communication channel. Is this the reason why the JPEG format is one of the most used on the web? as we have already seen, it has a good compression ratio taking into account the compression ratio and the amount of information lost. So, to answer the question we can mention that a compressed image will need less bandwidth to be transmitted and at the same time it will take less time to transmit, this results in lower cost-saving and productivity increasing.

3.9 Explain the role of features when recognizing objects.

An image consists mainly of three features that are important from an object recognition point of view: Edges, Corners and Blobs. Edges are lines, circles or even complex shapes that delimit objects in an image. Corners refer to the intersection of 2 or more edges, often corners are enough to obtain more than relevant information about an object. And finally, we have Blobs, these are very bright points or sections, which usually do not present very relevant information. To see the features of objects within an image more clearly, we can often resort to the gradient, as seen in the following figure.

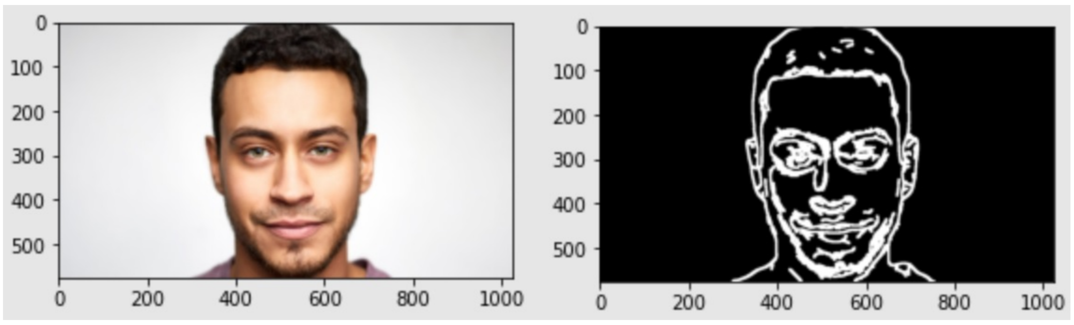


Figure 14: Image Gradient for features detection

So, why are features so important to detect objects, for example, let's pretend we are using ORB to detect objects. We will train our program to detect the most important features in a training image, then if we want to identify objects similar to our training image in video capture, what we will do is to compare its features with the ones we have saved from our training image, if there is a high coincidence then we can say that we have successfully detected the object. Even if we are working with neural networks to classify objects, the most relevant parameters when detecting whether an image belongs to a set or not will be the features, such as the contours of the eyes, the mouth, or the corners that define changes of direction, among others. So we can say that features are important in object recognition because they are the base parameters with which the algorithm we are implementing will work to carry out the detection.

3.10 Why do we require to use PCA for feature extraction?

In some cases we will be working with images that contain so much information that we will need to visualise the data in high dimensions, reducing their number. One way to do this is through **Feature Extraction**, which is the process of deriving new features from existing ones, which give more information and are less redundant. It is mainly used in pattern recognition and image processing applications where the dimension of the data is large. To apply this method we will use Principal Component Analysis or PCA, which in a nutshell is used to decompose a multivariate data set into a set of successive orthogonal components that explain a maximum amount of variance. To better understand what PCA is all about, we will start by explaining the main components it works with. The first of these is the covariance matrix, which contains an estimate of how the variables are related to each other. We will also have eigenvectors and eigenvalues, in a simple way eigenvectors represent directions and eigenvalues represent magnitude. Finally, we will have variability, which is related to the explanation of the behaviour of a variable. High variability usually indicates a signal, while low variability usually indicates noise. Therefore, the more variability in a particular direction is, in theory, indicative of something important that we want to detect. So, to answer the question of why we need to use PCA for feature extraction, we can say that PCA is necessary to detect the most important features, for that it relates the largest eigenvalues with the most relevant information, in that way together with the variance and the eigenvectors we manage to choose the most important features, at the same time the covariance matrix will tell us which features we should disregard because of their redundancy.

3.11 Why PCA can be used as an image denoising algorithm?

As mentioned in the previous question, PCA is a method to identify patterns in the data and express the data in such a way that we can highlight the most important components or features. One of the advantages of using PCA is that by identifying the most important components in our data, if we compress the information, reducing the dimensionality of the data, we will not lose much information. Since it keeps the most important characteristics and avoids reductions, we can call PCA a correlation technique. We can then use PCA to eliminate noise from the image by calculating local statistics and estimating the corresponding PCA transformation matrix by traversing a window through the image sections. You can use Murali Mohan, Subramanyam, and Giri Prasad's technique to remove noise from an image. PCA will be used in two stages: first, it will be used to eliminate the majority of the noise, and then the first stage's effects will be optimized.

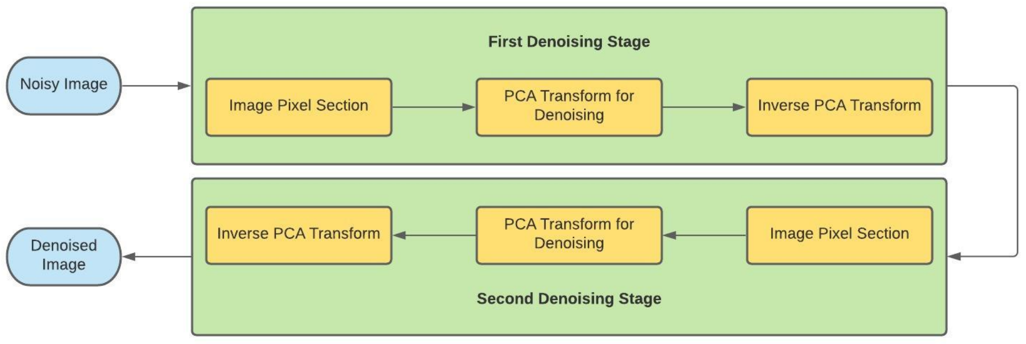


Figure 15: Image Denoising with PCA

So we can say that the reason why PCA can be used as an image denoising algorithm is that, we can use PCA to identify the most important components in our data, and eliminate redundancies without losing too much information, as it maintains the most important characteristics. important, so we can remove the noise from the image by calculating the local statistics and estimating the corresponding PCA transformation matrix.

3.12 Explain how PCA is related to the dimensionality reduction task

Dimensionality reduction refers to reducing the number of input variables for a data set. For example, if our data model is represented by rows and columns, as in an image, then the input variables are the columns that are fed as input to a model to predict the target variable. The input variables are also called features. A popular approach to dimensionality reduction is to use a technique known as feature projection, where the algorithms used are named projection methods. These methods seek to reduce the number of dimensions in the feature space whilst also preserving the most important structure or relationships between the variables observed in the data. The most common approach to dimensionality reduction is called Principal Component Analysis or PCA. In addition to everything explained above, it can be complemented that by reducing the number of characteristics, a projection method is used where the data of m columns (it could be an image) are projected to a subspace with m or fewer columns, maintaining the characteristics most relevant. . Now we can understand the steps used to obtain the eigenvectors and eigenvalues through matrix methods such as auto-decomposition. A final definition of PCA using everything mentioned so far in the last questions would be what is an orthogonal projection of the data in a space of less dimension, known as the principal subspace so that the variance of the projected data is maximized.

So, to answer how the PCA is related to the dimensionality reduction task, we can say that, PCA is one of the most used methods to carry out this task, being a method of projection of data to a lower-dimensional subspace, so that the variance is maximized and therefore, as explained in previous questions, we can say that we maintain the most important features and eliminate redundancies, thus reducing the dimensionality of our data without losing relevant information.

References

[1] Brownlee, J. (2020, 18 agosto). Principal Component Analysis for Dimensionality Reduction in Python. Machine Learning Mastery. `https://machinelearningmastery.com/principal-components-analysis-for-dimensionality-reduction-in-python/`

[2] Babu, Y. M. M., Subramanyam, M. V., & Prasad, M. G. (2012). PCA based image denoising. Signal & Image Processing, 3(2), 236.

[3] How does the JPEG compression work? (2011, 19 septiembre). Image Engineering. `https://www.image-engineering.de/library/technotes/745-how-does-the-jpeg-compression-work#:%7E:text=The%20JPEG%20compression%20is%20a,setting%20(or%20chose%20presets)`

[4] Suomela, J. (2011, 17 junio). Which is the limit of lossless compression data? (if there exists such a limit) [Comment]. Theoretical Computer Science Stack Exchange. `https://cstheory.stackexchange.com/questions/7027/which-is-the-limit-of-lossless-compression-data-if-there-exists-such-a-limit`

[5] Arora, M. (2019, 21 septiembre). Feature Extraction-Principal Component Analysis - Mansi Arora. Medium. `https://medium.com/mansiarora20448/feature-extraction-principal-component-analysis-a10705b330ce`

[6] M. W. Marcellin, M. J. Gormish, A. Bilgin and M. P. Boliek, "An overview of JPEG-2000," Proceedings DCC 2000. Data Compression Conference, 2000, pp. 523-541, doi: 10.1109/DCC.2000.838192.

[7] Ye, L. P. R. (2021, 25 enero). Python String Manipulation for Data Scientists in 2021. Medium. `https://towardsdatascience.com/python-string-manipulation-for-data-scientists-in-2021-c5b9526347f4`

[8] What is Data Compression? — Barracuda Networks. (s. f.). Barracuda. `https://www.barracuda.com/glossary/data-compression#:%7E:text=compression%20is%20MPEG.-,Why%20Data%20Compression%20is%20Important,transmission%20time%2C%20and%20communication%20bandwidth.&text=A%20compressed%20file%20also%20requires,costs%2C%20and%20also%20increases%20productivity`

[9] Sahir, S. (2019, 27 enero). Canny Edge Detection Step by Step in Python — Computer Vision. Medium. `https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123`

[10] Computer Hope. (2018, 13 noviembre). What's the difference between 16-bit, 24-bit, and 32-bit color? `https://www.computerhope.com/issues/ch001557.htm#:%7E:text=Like%2024%2Dbit%20color%2C%2032,color%20supports%204%2C294%2C967%2C296%20color%20combinations.`

[11] Majumder, A. (s. f.). Chapter 3 Notes. Donald Bren School of information & Computer Sciences. `https://www.ics.uci.edu/~majumder/vispercep/chap3notes.pdf`

[12] Finding the Edges (Sobel Operator) - Computerphile. (2015, 4 noviembre). [Vídeo]. YouTube. https://www.youtube.com/watch?v=uihBwtPIBxM

[13] Burger, W., & Burge, M. J. (2016). Digital image processing: an algorithmic introduction using Java. Springer.

[14] Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.

[15] Bellomo, G., Bosyk, G.M., Holik, F. et al. Lossless quantum data compression with exponential penalization: an operational interpretation of the quantum Rényi entropy. Sci Rep 7, 14765 (2017). https://doi.org/10.1038/s41598-017-13350-y