**Lab4\_1: Digital Image Processing with LabVIEW.**

The field of imaging or image processing has a very broad scope in today’s digital era. Image processing is a computational process that transforms one or more input images into an output image. Image processing is frequently used to enhance an image for human viewing or interpretation, for example to improve contrast or brightness. Alternatively, and of more interest to robotics, it is the foundation for the process of feature extraction such as object detection, pattern recognition, 3D modeling, etc.

Image processing mainly include the following steps:

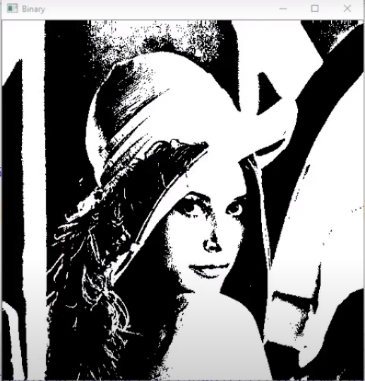
1. Importing the image via image acquisition tools
2. Analyzing and manipulating the image
3. Output image or a report which is based on analyzing the output image.

**IMAGES**

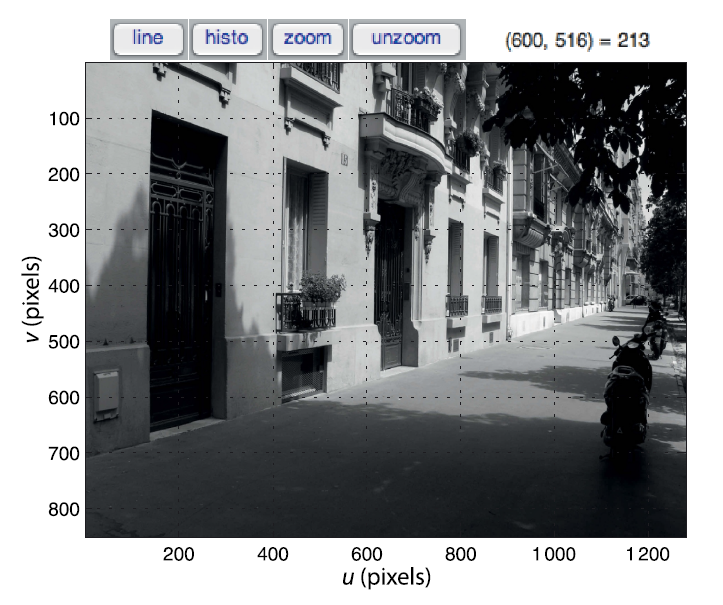
An image is defined as a two-dimensional function F (x, y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x, y) is called the intensity of that image at that point. When x, y, and amplitude values of F are finite, we call it a digital image.  In other words, an image can be seen as a rectangular array of picture elements (better known as pixels). Thus, along this lab we will internally consider it as a matrix of intensity values, where each element of that matrix (x, y pixel) tells us the intensity of the image in that specific point. This intensity is normally represented in a range from 0-255 or 0-1.  We can process or modify the values within the matrix by one or several algorithms. This is the principle of Digital Image Processing (DIP). Nevertheless, it is important to remember that an image can also represent other data than intensity, for example height, temperature, density, etc.

**IMAGES TYPES**

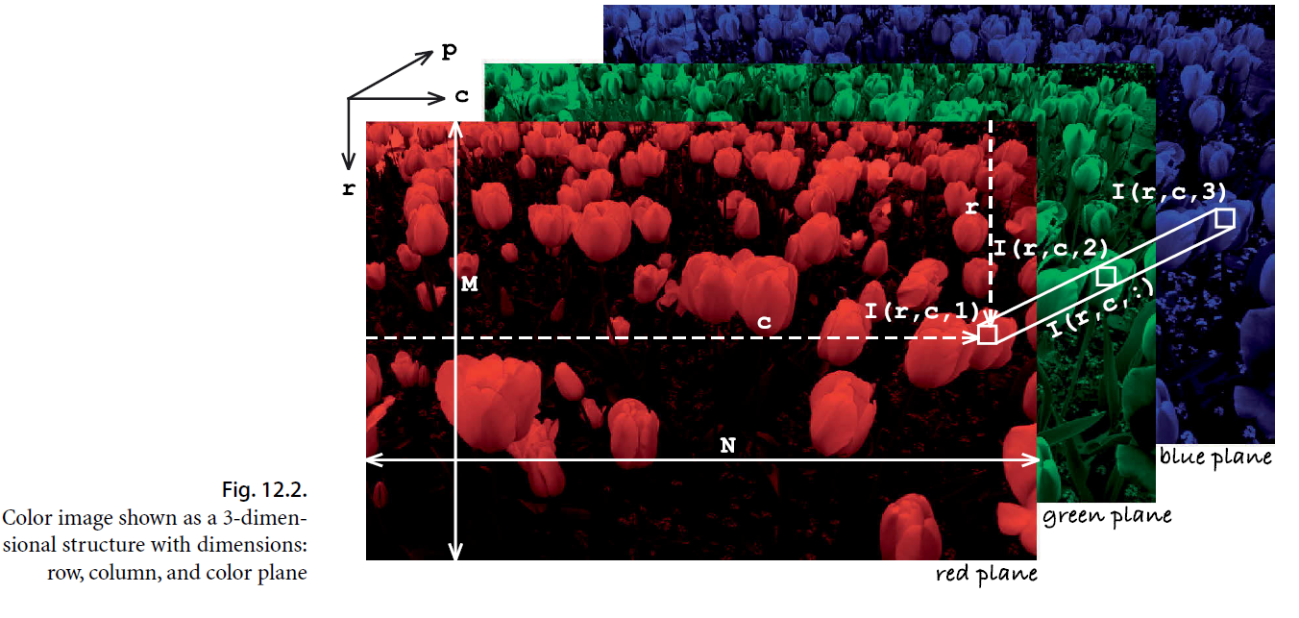
**BINARY IMAGE**– The binary image as its name suggests, contain only two-pixel elements (i.e., 0 & 1) where 0 refers to black and 1 refers to white. This image is also known as Monochrome.



**8-bit COLOR FORMAT–** It is the most famous image format. It has 256 different shades of colors in it and commonly known as Grayscale Image. In this format, 0 stands for Black, 255 stands for white, and 127 stands for gray.



**RGB IMAGE-** A RGB image is what we normally call a colored image. The red, green, and blue planes represent the intensity of certain color within the image. When we overlay these 3 planes, any color can be formed. This gives us the impression of thousands of colors when we see a digital image. Everything is about the individual intensities of the R, G, B components.

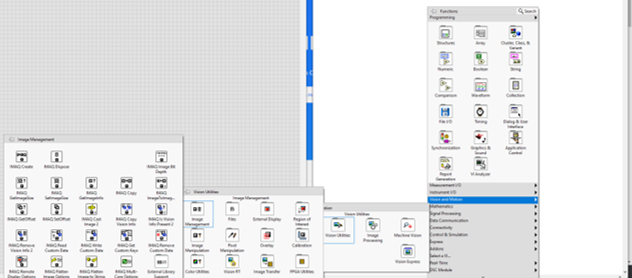


*For more information, please refer to: "LabVIEW Machine Vision and Image Processing Course Manual", National Instruments or "Robotics, Vision and Control", Peter Corke*

**IMAGE FORMATS**

A very large number of image file formats have been developed and are comprehensively catalogued in [Wikipedia](https://en.wikipedia.org/wiki/Image_file_formats). The most popular is JPEG which is used for digital cameras and webcams. TIFF is common in many computer systems and often used for scanners. PNG and GIF are widely used on the web. The internal format of these files is complex but a large amount of good quality open-source software exists in a variety of languages to read and write such files. In our case, we will use LabVIEW to work with images. Nevertheless, it is important for you to know that there are several other options out there. Another software, which is widely used for DIP, is Python or MATLAB, as both offer a lot of toolkits for this purpose.

As for now, you should have LabVIEW installed on your computer together with the Vision and Motion Toolkits. You will require them to properly run the provided VIs.



**STEGANOGRAPHY WITH LABVIEW**

Sending hidden messages, which cannot be discovered even if communications are intercepted or eavesdropped, is a matter that has occupied mankind since ancient times. As early as the 5th century BC, the Greek historian Herodotus related how the Athenian general Histieus, while planning a revolution, shaved the head of his most faithful slave, tattooed him a message and waited for his hair to grow back before sending it to Aristagoras. Upon the arrival of the slave, Aristagoras, aware of the existence of the message, shaved his head and read the hidden tattoo that urged him to start the revolt against the Persians. This is one of the first known cases of steganography. Stenography deals with the study and application of various techniques to hide information through messages and objects, within others.

In computer terms, steganography is the discipline that studies the set of techniques whose purpose is to hide sensitive information, messages or objects, within other so-called container files, usually multimedia: digital images, videos or audio files, with the aim that the information may go unnoticed by third parties and can only be retrieved by a legitimate user.

Unlike cryptography, which consists of sending encrypted information so that it is not intelligible to an unauthorized third party, steganography goes a step further and tries to hide even that the communication itself is taking place. These techniques were initially developed for military or espionage applications, as it seems logical, but we must not ignore the civil uses that have appeared over the years to protect secrets that we want to keep safe, without raising suspicions about their existence.

In Steganography:

* Information is hidden within another object.
* It is about composing hidden messages so that only the sender and receiver know that the message exists.
* The information goes unnoticed so that we do not know its existence.

In Cryptography:

* Encryption techniques are used to make a message unreadable through the use of a key or pair of keys.
* It is used when communicating through an unreliable medium such as the Internet, where the information must be protected from third parties.
* We know that you are passing sensitive information, but it is encrypted.

It is important to note that both techniques can complement each other, by combining steganography and cryptography, greater security can be achieved.

In this lab we will try to encode a piece of text or audio within an image in LabVIEW. Remember than an image can be seen as a matrix of pixels, which are formed by bits. Text, we see in computers follow the [ascii](https://elcodigoascii.com.ar/) notation, which at the end of the day is represented by bits as well.

Based on this information, answer:

**How do you think we could hide text within an image?**

\*\*\*---YOUR ANSWERS HERE in blue---\*\*\*

When we introduce texts within the values of the pixels of an image, we slightly modify the intensity of each pixel. The amount of information we can hide within the image will clearly depend of its type and resolution. We can use gray scale or RGB images in stenography. As you might have imagined already, more information can be hidden in a colored image, as we have 3 planes.

\*Why do you think we don´t use binary images in stenography? State your answer\*\*

\*\*\*---YOUR ANSWERS HERE in blue---\*\*\*

The number of bits we modify of the image with the hidden information is up to us. However, as it is logic, the more bits of a pixel we alter, the more the color or intensity of that pixel will vary.

Below you will find some images that have been modified 1bit, 2bits, 4bits, 5bits, 6bits, 7bits using LabVIEW.  As you can see, in order to keep a message hidden, we cannot modify so many bits of the pixels of our image.

\*\*What would happen if we modify 8bits of each pixel in the image? \*\*

\*\*\*---YOUR ANSWERS HERE in blue---\*\*\*





**YOUR TASK**

Before starting with your task, make sure you have answered the previous questions as they will be graded.

As mentioned already, we aim to implement an application in LabVIEW that help us to hide a piece of text within an image. To do so (similar to the one shown by the lecturer), follow these instructions:

* Design a model of your application (in a paper sheet). Explain what is the procedure you will follow to implement this system
* Start looking at the provided ImagenPixel.vi example
* Design and implement your own solution to reach our target. Remember, we aim to be able to change 1,2,4,8 bits of any image.
  + You can use any image of your preference (resolution > 640×480)
  + The SW design is up to you, but good programming practices will be graded (scalability and flexibility of the code, friendly GUI, processing speed, etc.)
* Submit the following documents in order to complete this laboratory:
  + This document with your answers
  + The created application
  + The following images: original, 1bit\_modified, 2bits\_modified, 4bits\_modified, 8bits\_modified