Chapter 1

Introduction to Digital Image Processing

1.1 Digital Image Processing

A digital image is defined as a two dimensional matrix. Each element has coordinates (x, y) and a set of values $f(x, y)$. The coordinates correspond to the position of a certain element in the image called pixel. Normally a pixel defines a square region on the digital image. There are also other definitions for pixels, but these are non-standard variations such as hexagonal pixels. The contents of the set of values depends directly on the format of the image. For example, in a grey scale image it might represent an intensity value between 0 and 255 (see figure 1.1). In a colour image one could have three values associated with one pixel, one value for each basic colour (e.g., red, blue and green).

Most image processing libraries will use a coordinate system that has the origin at the top left corner of the image. That may be a bit confusing at the beginning, but with some programming exercises one gets used to the idea. It also pays to learn the data structures used by a particular library, so the programmer has full control of the elements of the image. For instance, a very simple solution to store the image described above is to store each pixel value sequentially using an unsigned *char*. However, to recover the image one would have to know the original width of the image. That is usually the reason why libraries often use a header that describes how the image was stored. There are several standard formats for digital images such as TIFF, BMP, PNG, JPEG and many others.

There are many areas of study related to Computer Vision, such as image processing, image analysis, image understanding. Young ET AL.[1] suggest the following categories:

- Image Processing images in $\longrightarrow$ images out (improvements, filters)
- Image Analysis images in $\longrightarrow$ measurements out (size, texture, positions etc)
- Image Understanding images in $\longrightarrow$ high-level description out (what is there, what is the relationship with the environment etc)

Computer vision uses all the three areas described above to achieve automation and intelligent decisions based on images. According to Gonzalez and Woods [2], Computer Vision tries to emulates human vision. There is no agreement about the scope for each of these areas and they often mix with other areas such as Artificial Intelligence, Intelligent Systems and others. There is another area called Machine Vision, which overlaps with Computer Vision. Some authors consider Machine Vision to be a sub-set of Computer Vision, when the latter is used in industrial applications.

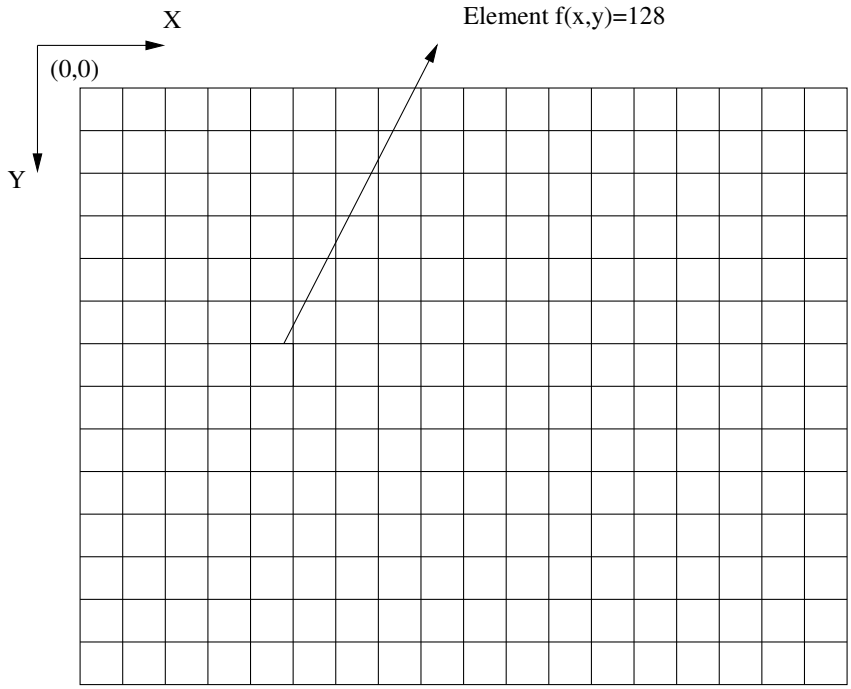

Figure 1.1: Matrix of pixels.

Another definition of Computer Vision is found in the OpenCV book [3]: "CV is the transformation of data from a still or video camera into either a decision or a new representation". Yet another one: "Computer Vision refers to the automated extraction of information regarding the objects or scene in one or more images. This differs from image processing, which is aimed at modifying an image for later human viewing or interpretation" (by David Lowe). Therefore, in computer vision the interpretations and decisions should be carried out automatically.

Some examples of applications:

- Image Enhancement $\longrightarrow$ Automatically improving some image characteristic to allow human beings to visualise better. An X ray image can be displayed with a better contrast so the operator can see certain features.
- Image Restoration $\longrightarrow$ Images that for some reason have faults that can be corrected. A noisy or a blurred image can often be corrected by removing the noise or by using deblurring filters.
- Image Segmentation $\longrightarrow$ It is useful to obtain measurements or to identify features that certain areas of the image contain. For example, an image from a microscope showing bacteria can be used to count them automatically. If the bacteria images contain different patterns (colours or texture), one can segment the images and them apply a blob counter filter.
- Feature Detection $\longrightarrow$ Sometimes it is useful to identify and locate specific features from the image. For example, edge detection is a common method used to find position, size or shape of objects in the image. Other common features that can be detected in images are point, corner and circle.

Figure 1.2: The histogram for two different images.

1.2 Image Properties

Some important properties of Digital Images are described next.

- Resolution $\longrightarrow$ Number of pixels. The more pixels the more information an image has, but it also demands more computer effort for any processing and more space for storage. A resolution of about a milion pixels (an image 1000x1000) is similar to what the human eye can see. Some authors consider that the number of pixels only represents the dimension of the image, while for them resolution is defined as the number of pixels per length of image (often printers are described in this manner).

- Depth $\longrightarrow$ Depending on how each pixel is represented it can contain different amounts of information about the image. The simplest representation is a binary image where each pixel can be either dark (0) or bright (1). A slightly better representation is a greyscale image, with different levels between dark and bright. Because the human eye is capable of differentiating about 200 levels of grey it is convenient to represent grey images using one byte, in such a way that a pixel can be between 0 to 255. Finally, colour images are represented using a mix of basic colours. The human eye can distinguish about 50000 different colours, using sensors that are capable of perceiving different frequencies of light. It is possible to represent colour images using a combination of red, green and blue. A common form of colour images uses one byte per colour, so each pixel needs 24 bits. A pixel with values (red=0,green=0,blue=0) represents black, while a pixel (red=255,green=255,blue=255) represents white.

- Histogram $\longrightarrow$ The distribution of the values for the pixels in an image is useful because it shows certain characteristics of the image and the elements it contains. There are grey scale histogram and colour histograms. Histograms do not characterise objects uniquely due to the fact that they do not store any information about the positions of the pixels. This ambiguity problem is often called the *scrambling problem*. One can find identical histograms for completely different images (figure 1.2).

There are also some basic relationships between pixels that are useful for a number of applications. Here we present the concept of adjacency for binary images. The concept extends to grey scale and colour images. Suppose that a pixel p at (x, y) is adjacent to a pixel q. If their values are the same and they are located at a pre-defined neighbourhood, then they are considered *adjacent*. One can define vertical and horizontal directions or diagonals where each pixel can have up to m neighbours.

- 4-adjacency $\longrightarrow$ Two pixels p and q with the same values are 4-adjacent (or 4-connect) if q belongs to the set of 4 neighbour positions.

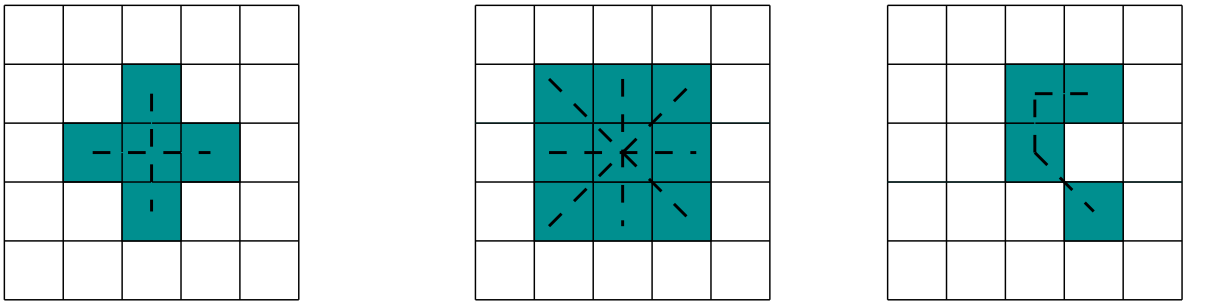

Figure 1.3: Pixel neighbourhood.

- 8-adjacency $\longrightarrow$ Two pixels p and q with the same values are 8-adjacent (or 8-connect) if q belongs to the set of 8 neighbour positions.
- m-adjacency $\longrightarrow$ Two pixels p and q with the same values are m-adjacent (or m-connect) if q belongs to the set of m neighbour positions.

The concept of adjacency can be applied in segmentation and in object recognition. For example, binary images of a scene can be interpreted to have certain objects. Such objects are identified if the segmentation is somehow able to separate them from the background.

1.3 References

Gonzalez and Woods [2], chapters 1 and 2.

1.4 Exercises

The exercises for this chapter use the open-source software GIMP (GNU Image Manipulation Program). This tool is freely available from the web for other OS, such as Windows, Linux and Mac.

1.4.1 Exercise 1

Open the figure *Akiyo1.jpg*. This is the first frame that belongs to a well known sequence that was used in many articles to show some aspect of Image Processing.

1. Save the image using a different type (choose from tif, bmp, png and other formats supported by GIMP). What is the difference in the file sizes? Is the information about each pixel still the same?
2. Change the colour mode from RBG to grey-scale. Save the new image as bmp.
3. Show the histogram for the image. Change the histogram (image $\rightarrow$ colours $\rightarrow$ curves).
4. Use GIMP to change the image. Use rotation, scaling and other geometrical transformations.

1.4.2 Exercise 2

In this exercise, you will use a tool called ImageMagick (`http://www.imagemagick.org/`). There are many commands available, but in this exercise you will only use *import* and *convert*.

1. Choose an image file (e.g., peppers.png).
2. Read the manual contents for these commands, e.g.,

   ```
   man convert
   ```

3. Use the following command from a terminal:

   ```
   convert -rotate 45 peppers.png peppers45.png
   ```

4. `convert -resize 100%x80% peppers.png peppers_def.png`
5. The command import works like print screen, it saves the X-window instance to a file:

   ```
   import test.jpg
   ```

   then point and click on the window you want. After two beeps, you can see the saved image.

Bibliography

[1] I. T. Young, J. J. Gerbrands, and L. J. V. Vliet, *Fundamentals of Image Processing.* Delft University of Technology, 1998.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Prentice Hall, 2008.

[3] G. Bradski and A. Kaehler, *Learning OpenCV.* o'Reilly, 2008.