P6 Project
Group 620
Computer Technology & Electronic and IT
Aalborg University
25 May 2022

AALBORG UNIVERSITY
STUDENT REPORT

Sixth Semester Students, Computer technology & Electronic and IT
ComTek & EIT
Frederik Bajers Vej 7
9220 Aalborg

Title:

XXXXXXXX

Project:

P6-project

Project Period:

February 2022 - May 2022

Project Group:

620

Participants:

Kazim Kerem Kocan
Khadar Guled
Laurits Winter

Group Supervisor:

Thomas B. Moeslund
Simon Buus Jensen

Page Count:
Appendix:
Finished 25th May 2022

Abstract:

Foreword

Kazim Kerem Kocan

Khadar Guled

Laurits Winter

Acronyms

ISO International Organisation of Standardisation

SDR Software Defined Radio

Contents

Foreword **iii**

Acronyms **iv**

1 Introduction **1**

2 Problem Analysis **2**
2.1 Camera 2
2.2 Image Processing 3
2.3 Environment 15
2.4 RFID 15
2.5 Cage 15
2.6 Noise 15
2.7 Scenario 15

3 Requirement specification **17**

4 Design **18**

5 Implementation **19**

6 Tests **20**

7 Discussion **21**

8 Conclusion **22**

1 Introduction

SDR (Software Defined Radio)

SDR

Problem Analysis 2

2.1 Camera

2.1.1 Basics of cameras

Light

Image acquisition

2.1.2 Camera properties

Zoom

Focus

Field of View

Aperture

The principle behind aperture is to limit the visibility of the lens. By doing so, the angles of which the light can enter the camera is vastly reduced. This increases the depth of field, as the image shifts further from a perspective view, and more towards an orthographic view. It is in practice not possible for the camera to transition entirely, but the closer it gets, the further the depth of field expands, along with the focal length narrowing. The major trade-off for this is the sharp reduction in the light entering the camera. This can be compensated for with a longer shutter speed.

Shutter speed

The shutter speed of a camera is determined by how long the shutters stay open for the camera. By adjusting the shutter speed, you can manipulate how much light enters the camera, and hits the sensors, thereby altering the brightness of the image. This can be useful when working in a dark environment, or working with very bright elements, such as the sun. The downside of working with extended exposure times is motion blur[**?**]. Whenever something is moving, The extended exposure time will make it leave a trail in it's path. This can be desired for artistic reasons, however for image processing, having this blurred trail can be a critical flaw for the application, and as a consequence ISO may be a preferred solution to adjust brightness3[**?**].

ISO

The most common application of ISO (International Organisation of Standardisation) is to brighten or darken pictures, when adjusting the exposure time isn't a viable option.

This could be in case of photographing moving elements, or recording video. Adjusting the ISO can alter the image, by Extrapolating imperfections in the captured image, by showing a grainy noise when it is configured high, and by washing out a certain level of fine material texture when configured low[**?**]. ISO manipulates the image by altering the data as the sensor inputs are mapped to the image. By working with the raw input, it can produce a much more precise and clear brightening of an image, as compared to what can often be achieved in post processing. ISO Got its name when it's predecessors "ASA" and "DIN" were combined into a single global standard in 1974, adopting the name of the organization that had managed the merging [**?**].

2.1.3 Data properties

Bayer pattern

Contrast

Color?

Where color be?

2.2 Image Processing

In this section, the basics of image and image manipulation are described.

2.2.1 Data interpretation and manipulation

Digital images are represented as two-dimensional arrays. An image is in these cases seen as a discrete function $f(x, y)$. A specific cell or dot in the image is called a pixel. Pixels present the values of the interpreted intensity from cells of the image sensor as described in section 2.1. Regarding the amount of bits for a single pixel, it is variable depending on the image type and color. Generally, a pixel contains 8 bits (equal to 1 byte) if it is in black-and-white. This means that there is $2^8 = 256$ different values. For a black-and-white image, the pixels are represented as a gray-scale value ranging from the interval from black (0) to white (255) [**?**].

When processing an image, it is usually beneficial to be able to modify the key features of an image. Generally, there exist two types of image processing: Point processing and neighborhood processing. For point processing, during an image processing each pixel of the input image is processed and then mapped to the output image with its exact position. This is especially useful for changing brightness and contrast.

Histograms

An important requirement for image processing is the computer being able to 'see' the image. While a person can see if an image is black-and-white or if the image is too bright or too dark, the case for a computer detecting, for instance, the brightness of the image, is by using the available pixels and count the pixels of different values. This is possible using a histogram, which is described as a discrete function given in equation 2.1:

$$p(k) = \frac{n_k}{n} \tag{2.1}$$

In equation 2.1, it has been assumed that the image is black-and-white and therefore using a gray-scale intensity. n is the total amount of pixels. n_k refers to the amount of pixel with a gray-scale value of k. By dividing n_k with n, the probability of a specific gray value of k appearing on the image, $p(k)$, can be found. For color images, the principle is the same, however, three histograms are needed (due to having three channels, red, green and blue).

The histogram is generally shown as a graph, where the gray-scale intensity is shown in x-axis and the amount of pixel having that specific intensity is shown on the y-axis as shown in figure 2.1. Based on the shape of the histogram, the image features such as brightness or contrast can be detected.

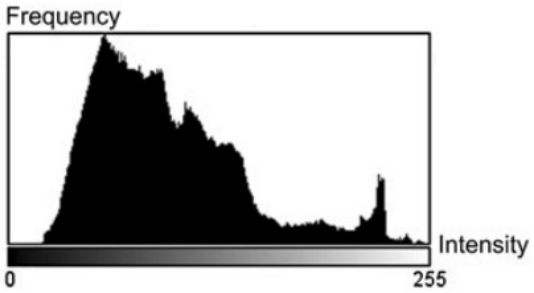


Figure 2.1. An example of a histogram. Image from [**?**, Chapter 4.3]

While it is possible using a histogram to detect several features of an image, it should be noticed that it is not possible to reconstruct an image based from a histogram. Thereby, it is also possible for several images to have the same histogram [**?**].

Histograms have so far been described as a tool to recognize distinct features of an image, but they can also be processed such that it can be used as an image manipulation tool - a term known as histogram processing. There exist different types of histogram processing regarding the type of operation needed. In the next section, some types of histogram processing have been described.

Histogram stretching

If the contrast is too low, the image might appear gray and unclear for humans. The characteristic of an image with low contrast is a narrow difference of gray-level intensity which is shown in the corresponding histogram of the image. An example of a gray-scale image with low contrast is shown in figure 2.2.

Figure 2.2. Low contrast image with corresponding histogram

A solution for this problem is to map the gray-level values into an other value such that the level of gray-level intensities are spread out, which improves the contrast. This method is known as histogram stretching. Based on the original image shown in figure 2.2, histogram stretching has been used to improve the contrast as shown in figure 2.3.

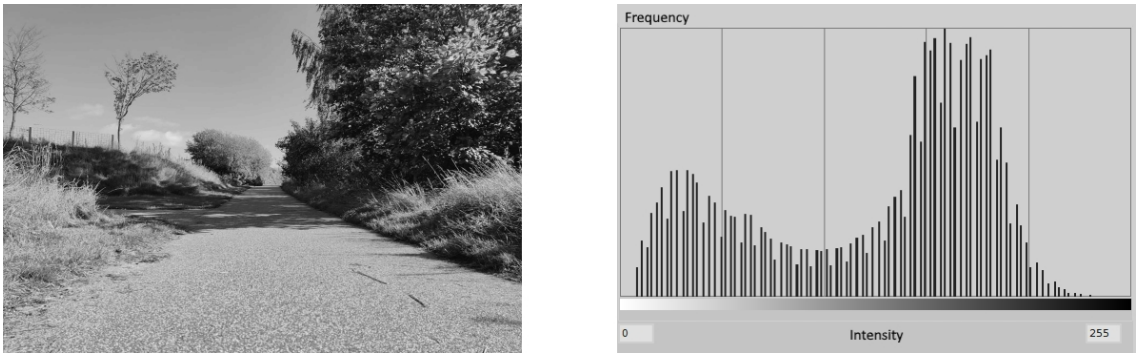


Figure 2.3. Improved contrast image from figure 2.2 with corresponding histogram

Thresholding

An other distinct feature for image processing is the ability to segment the image into several parts such as foreground with an object and background. While segmentation of an image is rather image analysis instead of image manipulation, it contains the necessary features for future image manipulation. Ideally, the segmentation between the background and object is visualized as a histogram with two significant peaks shown in figure 2.4. A histogram with those characteristics is bi-modal.

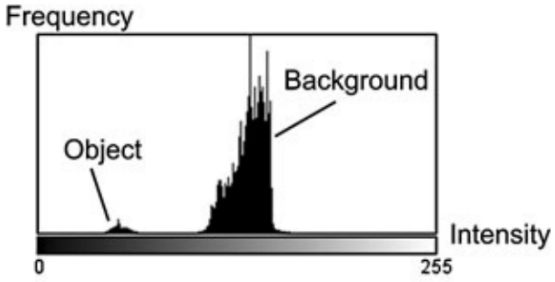


Figure 2.4. Ideal histogram [**?**, Chapter 4.4]

In figure 2.4, the peaks of object and background are clearly distinct from each other. However, it is generally difficult to distinguish objects from backgrounds due to the image containing a significant amount of information such as noise. One method of segmenting an image by reducing the amount of information is thresholding.

In thresholding, a threshold value T is given. If the pixel value of an input image is under or equal to the threshold value, it will be mapped to the intensity value 0 - which is shown as black. All pixel values above the threshold value will be mapped to 255 (white). The advantage of using the thresholding method is that the object and background will be distinguished as black and white respectively. For an 8-bit input image $f(x, y)$, output image $g(x, y)$ and the given threshold value T, the algorithm can be described as the following equation 2.2:

$$\begin{aligned} &\text{if } f(x,y) \quad \leq T \quad \text{then } g(x,y) = 0 \\ &\text{if } f(x,y) \quad > T \quad \text{then } g(x,y) = 255 \end{aligned} \tag{2.2}$$

During the process, some of the image's information will be lost or discarded, but the lost information is generally noise or otherwise redundant [**?**].

An example of using thresholding is shown in figure 2.5 and 2.6.

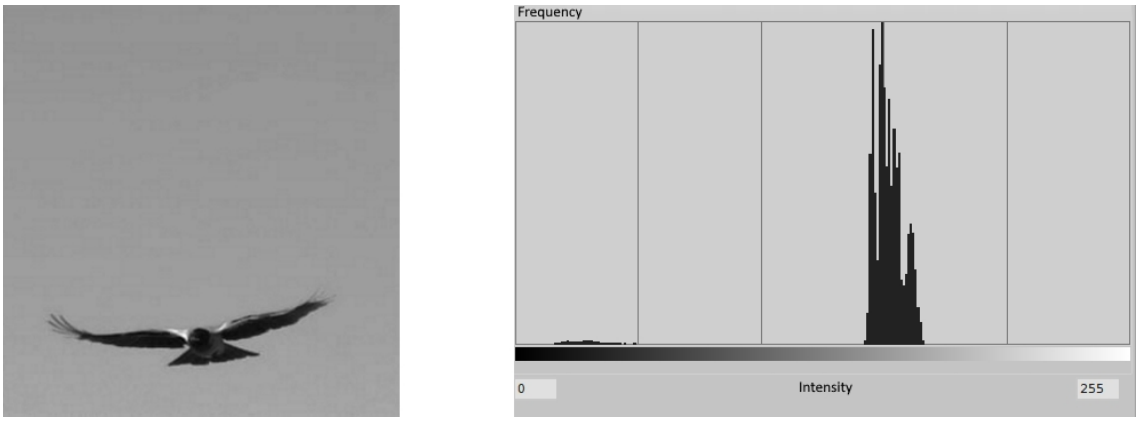
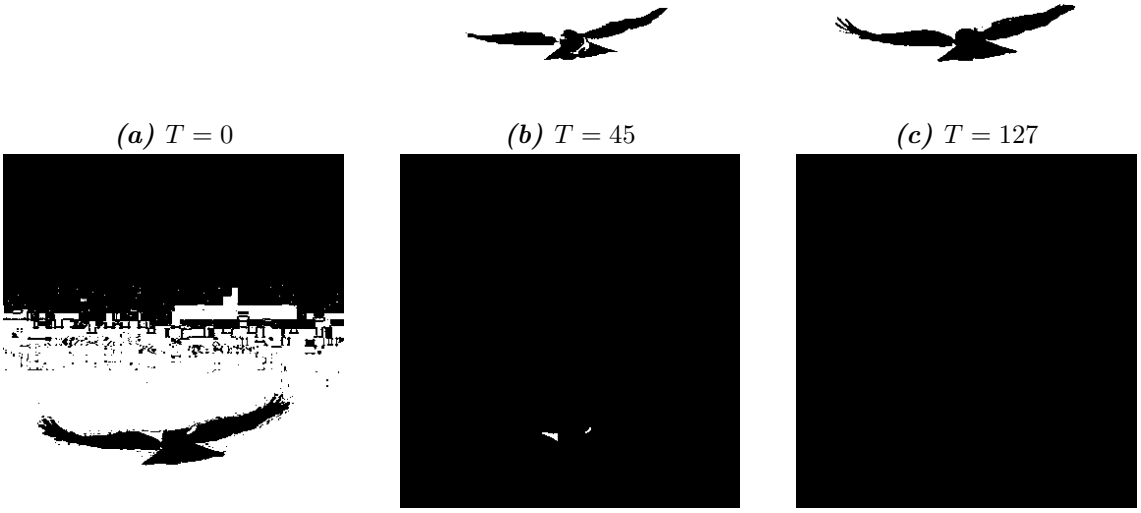


Figure 2.5. Gray-scale image with corresponding histogram

The desired object in figure 2.5 is the bird, while the background is the sky. In the histogram, a significant peak is shown at an intensity value of around 150. This indicates the background. However, a small peak is shown at the intensity value interval of around 25 to 60. This indicates the object as previously described and shown in figure 2.4. Thereby, it is not necessary to segment the image using the thresholding method, however, it has been performed and shown in figure 2.6 for further analyzing.



(a) $T = 0$ ***(b)*** $T = 45$ ***(c)*** $T = 127$

(d) $T = 150$ ***(e)*** $T = 200$ ***(f)*** $T = 255$

Figure 2.6. Image from figure 2.5 with different threshold values

Based on the histogram shown in figure 2.5 and equation 2.2, it is seen in figure 2.6b that most of the pixels for the object have an intensity value T under 45. Due to this, those pixels are mapped to 0 and are shown in the image as black. If the threshold value is increased to 127, it is seen in figure 2.6c that there is now an almost complete silhouette of the bird. However, looking at the histogram in figure 2.5, if T is increased more, then T would be greater or equal to the intensity values of the background. This is mostly shown in figure 2.6d and 2.6e, where in the first mentioned, parts of the background pixels are now being mapped to black and in the latter, almost the entire background has been mapped to black.

It is therefore important to find a good threshold value, where it is only the object and the background that is shown and segmented. Based on the example shown in figure 2.5 and 2.6, it is seen that in figure 2.6d, the silhouette of the bird is well-defined compared to figure 2.6c, however, a significant portion of the background is mapped to black. Therefore, figure 2.6c is more suitable for image processing in that case with a clear distinction between object and background. It can therefore be concluded that a good threshold value generally lies in the interval between the peak of the object and the peak of the background as shown in figure 2.5.

File format

Finally, an other important factor in image processing is the size of the image file. A raw digital image contains all data from the image sensor, which means that this type of image files have high quality in regards to details, but will naturally also take up significant memory and storage. This image file is advantageous for editing/manipulating photos, when a very high quality photo is desired. However, for a real-time system, e.g. a surveillance system like a security camera, the bigger file sizes and longer processing times is significantly disadvantageous. For systems, where a smaller file size or faster processing times are needed, it is preferable to compress the image. There exists two types of compression: Lossless and lossy compression.

In lossless compression, the data is compressed such that the original data can be be perfectly reconstructed from the compressed data. Thereby, despite that the file size has been reduced due to the compression, no data has been lost. The compression has been done using several algorithms such as Huffman coding [**?**]. In comparison to lossless compression, lossy compression focuses on minimizing the file size by removing redundant information. Therefore, some data might get lost in the process, which mean that the original image cannot be reconstructed perfectly based on the lossy compressed data. However, the compression algorithm has been optimized from the human vision's point of view, such that the compressed image looks very similar to the original. Compared to lossless compression, lossy compression is very efficient regarding the file size, where the compressed image's file size can be as small as 10% of the original size without visible degradation of the image quality [**?**]. In table 2.1, the different compression types are compared and their respective advantages and disadvantages have been highlighted. The disadvantages are taken from the point of view of image processing. Here, the preferable outcome is to have a small file size (thereby an efficient compressed image) that do not have any redundant data in the image.

Compression type	**Advantage**	**Disadvantage**	**Example of file type**
No compression (Raw)	Image of highest quality	Big file size Long processing time	.pgm, .ciff
Lossless	High quality image with smaller file size	Unnecessary data	.png, .gif
Lossy	Decent quality image with significantly smaller file size Minimal-to-none redundant data	Heavy compression might degrade the image quailty	.jpeg

Table 2.1. The advantage and disadvantage of different compression types

2.2.2 Edge detection

Edge detection is the process of finding the boundary of an object, meaning that in almost all cases, edges will represent the object in the image. The edge of an object can then be used to get a higher level of abstraction of the image, which means that the information in the image will be less thus easier to process. Before it is possible to find the edge of an image it is important to define what it is. Using a gray-scale image as a baseline it is possible to define an edge as a position where a significant change in gray-level takes place. Looking at figure 2.7 on the left there is shown an image. To find an edge on that image a slice of that image can be taken vertically that is one pixel wide, which is illustrated in the figure between the arrows. By using this slice, a graph can be made where we can interpret the intensity value in the slice as the height in the graph that is shown on the right of figure 2.7. When looking at the graph when there is a significant change in the gray-scale value in the slice, the graph will also have a significant change in the height. These significant changes in the height is illustrated as circles in the graph, and is where the edges is defined in an image [**?**, Chapter 5.2.2].

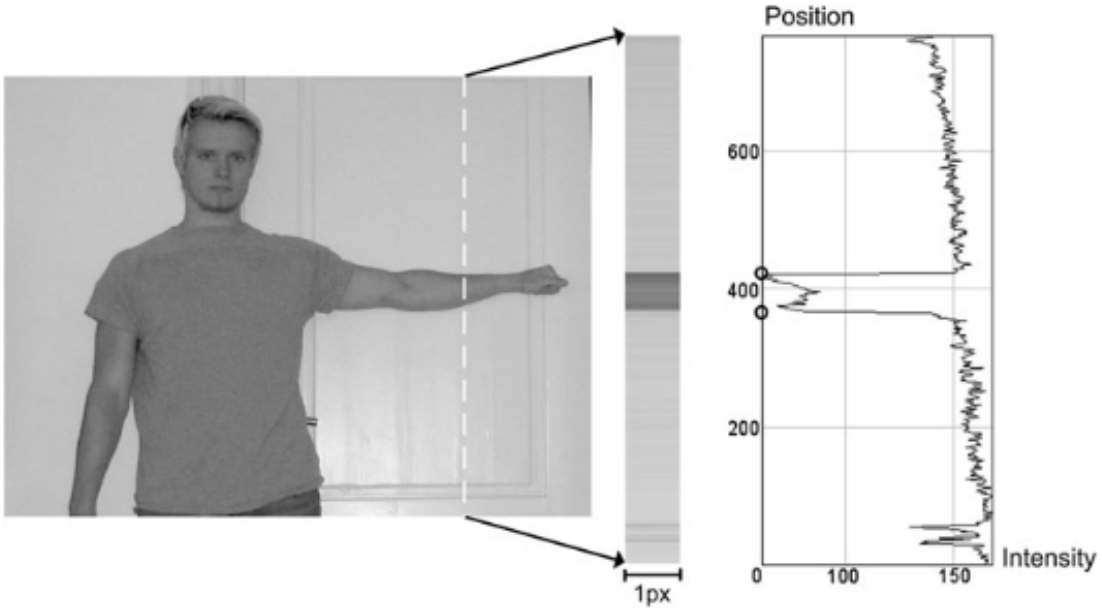


Figure 2.7. A single column of the image is enlarged and presented in a graph. This graph contains two very significant changes in height, the position of which is marked with circles on the graph. This is how edges are defined in an image. Image from [**?**, Chapter 5.2.2]

To detect edges in an image the directional change in the intensity can be used, this concept is gradient in image processing. This can be done by representing the previous image in a 3D graph as seen in figure 2.8. In the graph it is shown that the width and height of the image is represented by the x- and y-direction respectively, whereas the z-direction is represented by the intensity of the image interpreting it as height.

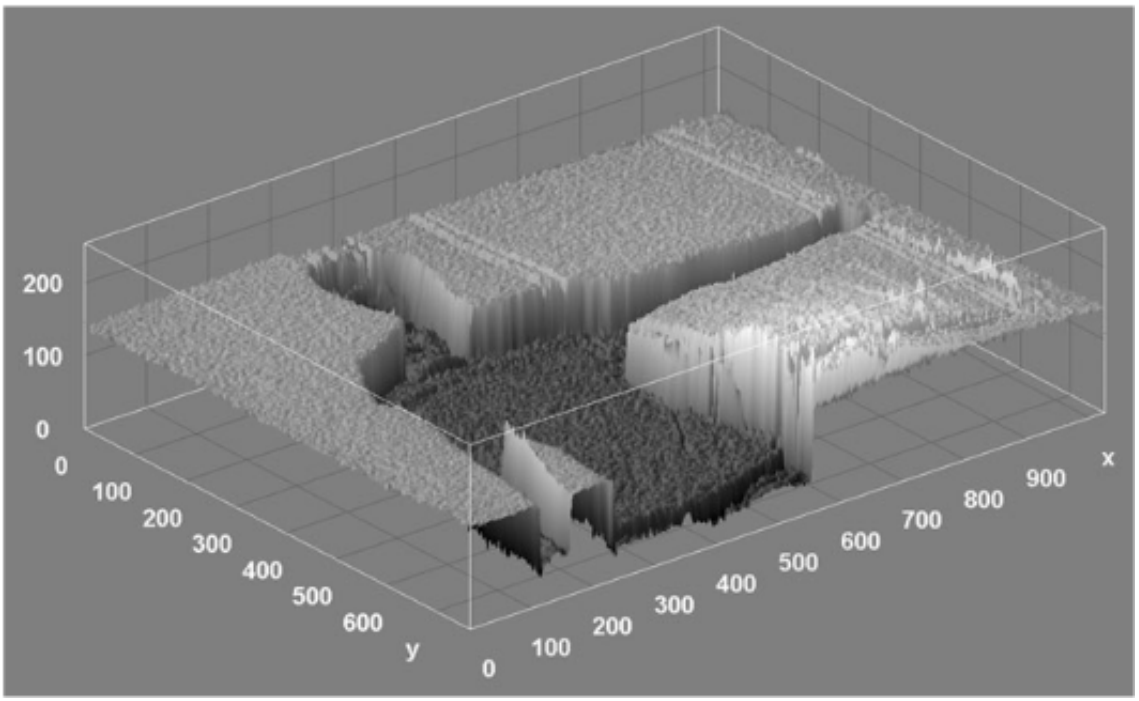


Figure 2.8. A 3D representation of the image from figure 2.7, where the intensity of each pixel is interpreted as a height. Image from [**?**, Chapter 5.2.2]

For each point in the image there would be two partial gradients that would span a plane in the x- and y-directions intersecting a chosen point. These partial gradients can be used to determine the direction of the edge, as well as using their derivatives to calculate the resulting gradient by using eq. 2.3

$$\overrightarrow{G}(g_x, g_y) = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \tag{2.3}$$

A gradient also has a magnitude, which is how steep the change of height is in the direction of the gradient and is also the length of the gradient vector. The length of the gradient vector can be calculated using the Pythagorean theorem solved for magnitude as seen in eq. 2.4

$$Magnitude^2 = g_x^2 + g_y^2 \Leftrightarrow Magnitude = \sqrt{g_x^2 + g_y^2} \tag{2.4}$$

For faster implementation the magnitude can be approximated by using eq. 2.5

$$Magnitude = |g_x| + |g_y| \tag{2.5}$$

As the partial gradients are first order derivatives it can not be calculated because an image is not a continuous curve and therefore needs an approximation. To calculate the approximation of the gradient it is possible to use the difference between the previous and next value seen in eq. 2.6

$$g_x(x,y) \approx f(x+1,y) - f(x-1,y) \tag{2.6a}$$
$$g_y(x,y) \approx f(x,y+1) - f(x,y-1) \tag{2.6b}$$

This approximation will result in a gradient value that is positive when the pixel change from dark to bright and negative when it is reverse. It is possible to get the opposite value by switching the signs seen in eq. 2.7

$$g_x(x,y) \approx f(x-1,y) - f(x+1,y) \tag{2.7a}$$
$$g_y(x,y) \approx f(x,y-1) - f(x,y+1) \tag{2.7b}$$

Equation 2.6 can be applied with a 2D kernel using correlation, a 2D kernel is used because a 1 dimensional kernel is too sensitive to noise. A popular 2D kernel that is used is the Sobel kernel where the center pixel is weighed more which can be seen in figure 2.9

Sobel

Vertical

-1	0	1
-2	0	2
-1	0	1

Horizontal

-1	-2	-1
0	0	0
1	2	1

Figure 2.9. A Sobel kernel. Image from [**?**, Chapter 5.2.2]

What happens when using the sobel kernel is that the coefficients multiply the corresponding values around the source pixel then adding them together to get a sum of the values. This value will be a negative number if it goes from a bright to dark or in other words a bigger number to smaller number, in the other spectrum if it goes from a smaller number to a bigger number it will be a positive number. In the cases where the pixels are uniform the number given will be 0. This process where you calculate the sum can be seen in figure 2.10

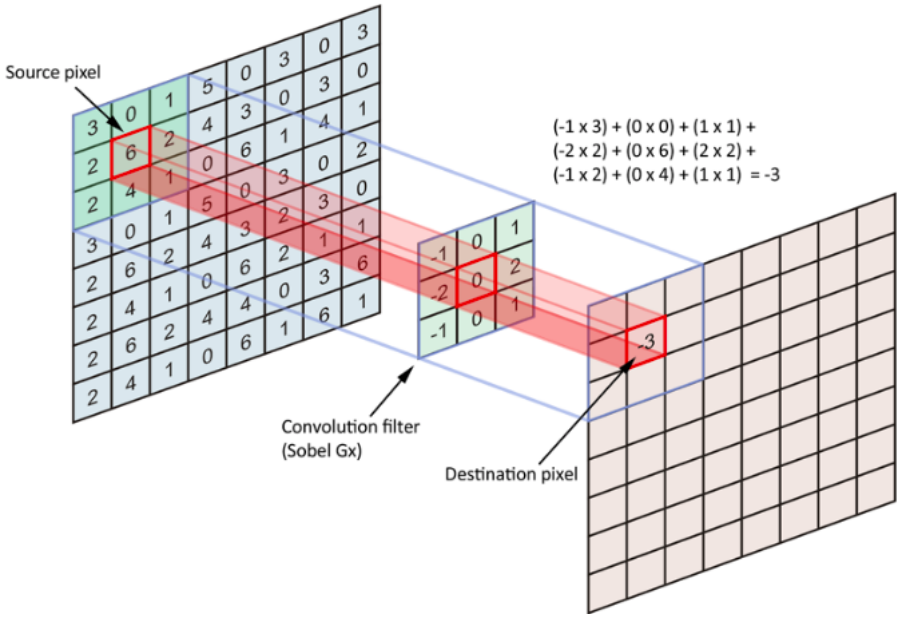


Figure 2.10. Sobel kernel calculation of convolution. Image from [**?**]

It is possible to use the sobel kernel by combining the vertical and horizontal correlated images and then binarize the resulting image by thresholding it, but a better solution is to use the canny edge detector. The canny edge detector is an algorithm that uses the sobel kernel and also gives pixel-thin edges, this process is done in 5 steps:

1. Apply a Gaussian filter to smooth the image and remove noise
2. Find the gradients using the sobel kernel
3. Apply non-maximal suppression, by comparing a pixel with its neighbors in the gradient direction
4. Apply double threshold comprising of T_{high} and T_{low}
5. Categorize an edge that is localized higher than T_{high} as a strong edge and as weak edge if between T_{high} and T_{low}, then only keep edges that are strong or connected to a strong edge

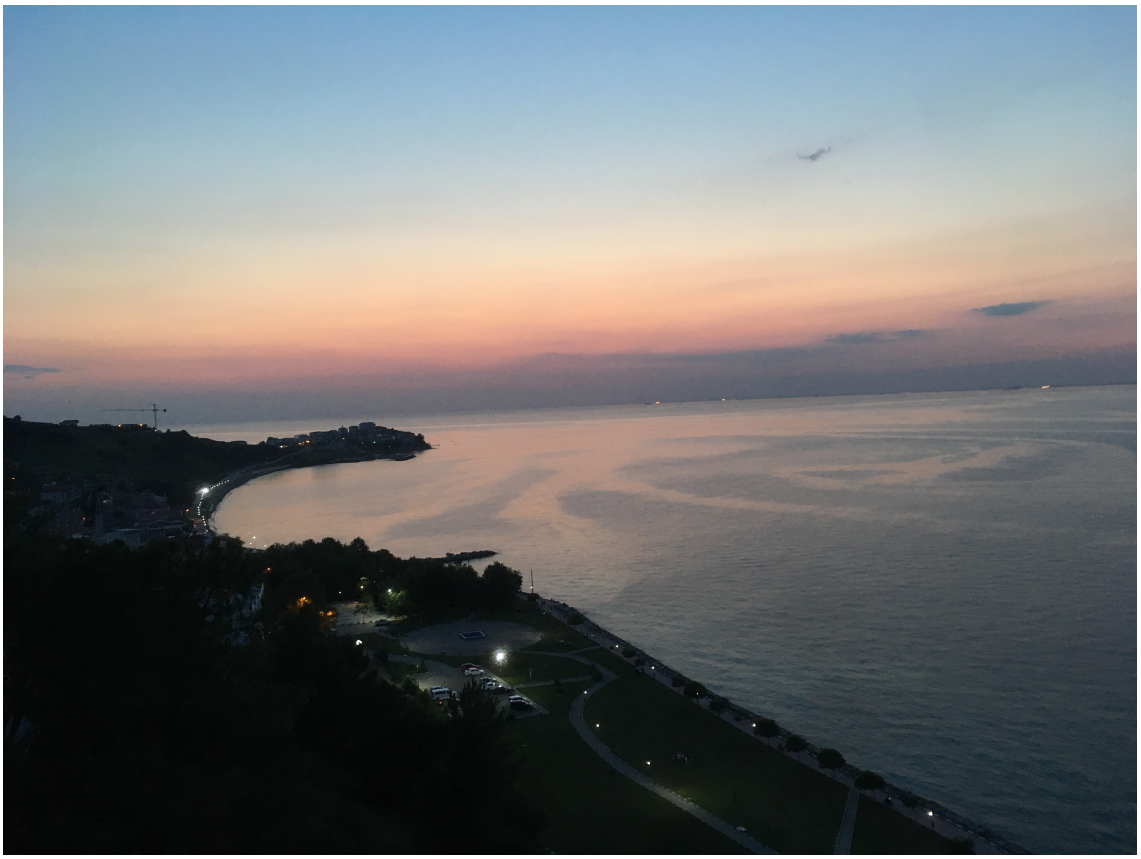


Figure 2.11. original

Figure 2.12. Sobel

Figure 2.13. Canny

2.3 Environment

2.4 RFID

2.5 Cage

2.6 Noise

2.7 Scenario

To solve the problem statement, the system needs a means to detect a cage, to identify the cage, to observe the movement of said cage, to analyze these observations and to store the resulting information. To detect, and identify the cage an RFID sensor is a functional solution. It will register the cage and it's associated ID when the cage is within range. To observe it, a camera will be utilized. This will stream the movements of the cage to the processing device. To process this data, a PC or a specialized device is required to do the heavy lifting. A database will be used to store the results of the analysis, along with the RFID tag that is associated.

1. The system can start only when the RFID sensor has detected the cage is in range and then get the cages ID, which can the be stored

2. After the cage has been detected the camera can start streaming the video and the movement of the cage.
3. With the streaming data the footage can be processed by a computer in real or near real time to identify the cage and detect the direction of the cage.
4. When the direction of the cage has been found it can then be sent, along with the cages ID, to a database which then can be stored for easy access and readability.

From the presented scenario, an overview of the systems functionalities can be outlined. This is done by a use case diagram where the actors are the PC (Computer), the RFID sensor, the camera, and the database where the primary actor is the PC. The use case diagram is shown in figure 2.14

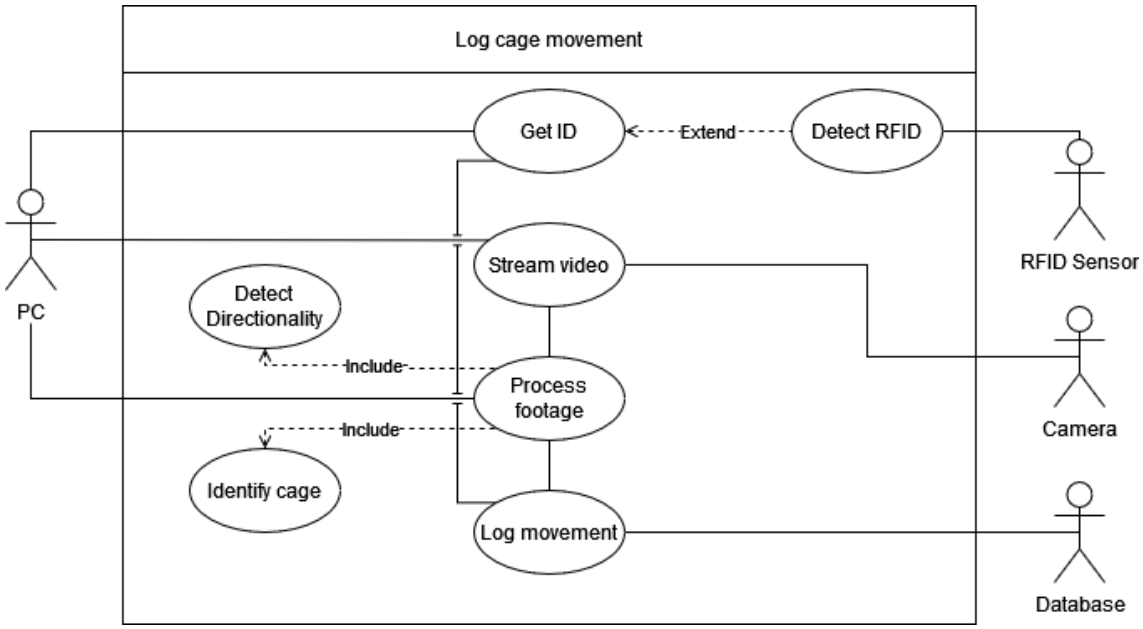


Figure 2.14. Use Case

Requirement specification 3

Design 4

5 Implementation

Tests 6

7 Discussion

Conclusion 8