HARDWARE IMPLEMENTATION OF AN EDGE DETECTION ALGORITHM

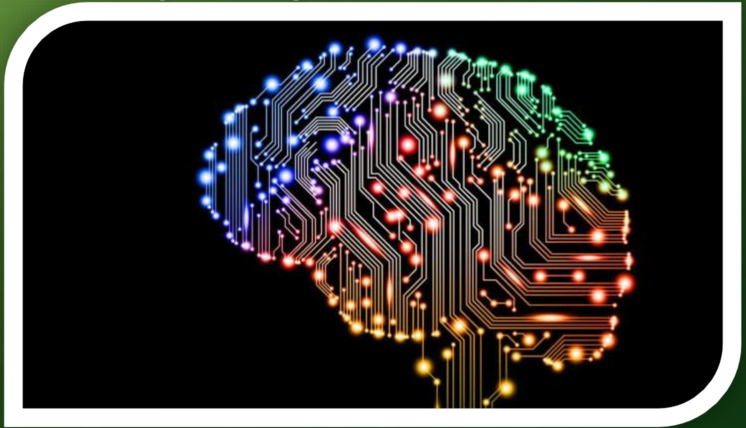
REALIZED BY:

- ABD ELSATAR BAHRI
- BECHIR BEN KAHLA
- MOHAMED HNEZLI

INTRODUCTION

ARTIFICIAL INTELLIGENCE

- Computer as their name said can compute difficult mathematical operation in a very fast way !
- But why not making them more intelligent ?
- The aim of Artificial Intelligence is to imitate the behaviour reputed intelligent of humans so that we can produce machines that perceives its environment and takes actions that maximize its chance of success at some goal
- Among the intelligent behaviors a human had we find the visual perception for which we associate the machine vision field !



MACHINE VISION

- It aims to making machines able to understand captured pictures or videos
- But it require some image processing tools !
- Edge detection is a fundamental tool in image processing, machine vision and computer vision
- It is an inevitable step for shapes recognition and features extraction
- Edge detection problem can be solved with a software solution !
- But for real time problems a hardware implementation is mandatory !

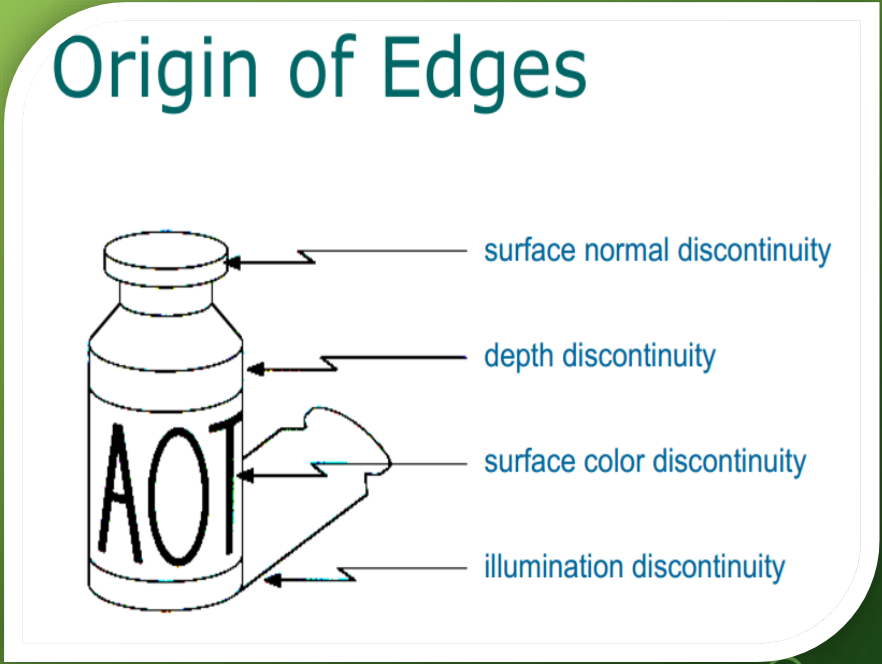
OBJECTIVE

- Implementation of a real time edge detection solution !

PRELIMINARY STUDY

EDGE DETECTION

- Edge detectors allow us to convert a 2d image into a set of curves more compact and easy to use than pixels so that we can extract the salient features of the scene.
- Edges are caused by a variety of factors (figure).
- But always an edge in an image is a significant local change in the image intensity.
- In 1D change is measured by the derivate but in 2D we use the gradient operator.
- As images are discrete function (matrix of discrete values) we must discretize the gradient operator.
- Therefore we find different operator approximating the gradient like (Sobel, Roberts, Prewitt, Canny …).


Origin of Edges
AOT
surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

SOLUTION IMPLEMENTATION

- We need to read the image pixel by pixel and at each clock cycle insert the new pixel in the processing chain.
- With new high-resolution standards, edge detection operators must receive input data at a rate up to 3 Giga samples per second
- For example, the HD video standard (1080x1920 p) requires 60 fps, with approximately 2.1 megapixel per frame, and 126 megapixel per second resulting a data rate of 3 Gigabit per second using uncompressed RGB encoding
- So edge detection operators require high computation power !
- Therefore, to solve this problem, hardware implementation is essential which offers much greater speed than software implementation

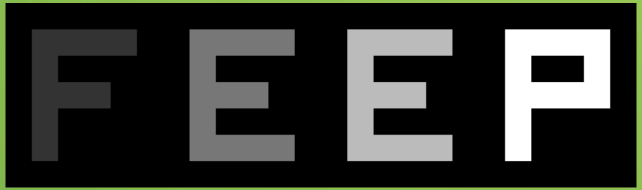
WORK STEPS SPECIFICATION

WORK STEPS

- As the project duration is a bit limited and some resources are unavailable, we took the decision to implement a test bench model for edge detection.
- It means that we will not work on real time video but we will work with PGM images.
- Our VHDL model will
  - Read a PGM image
  - Process the image
  - Store the processed image

PGM SOURCE IMAGE READING

- PGM files are a simple grayscale image description
- P2

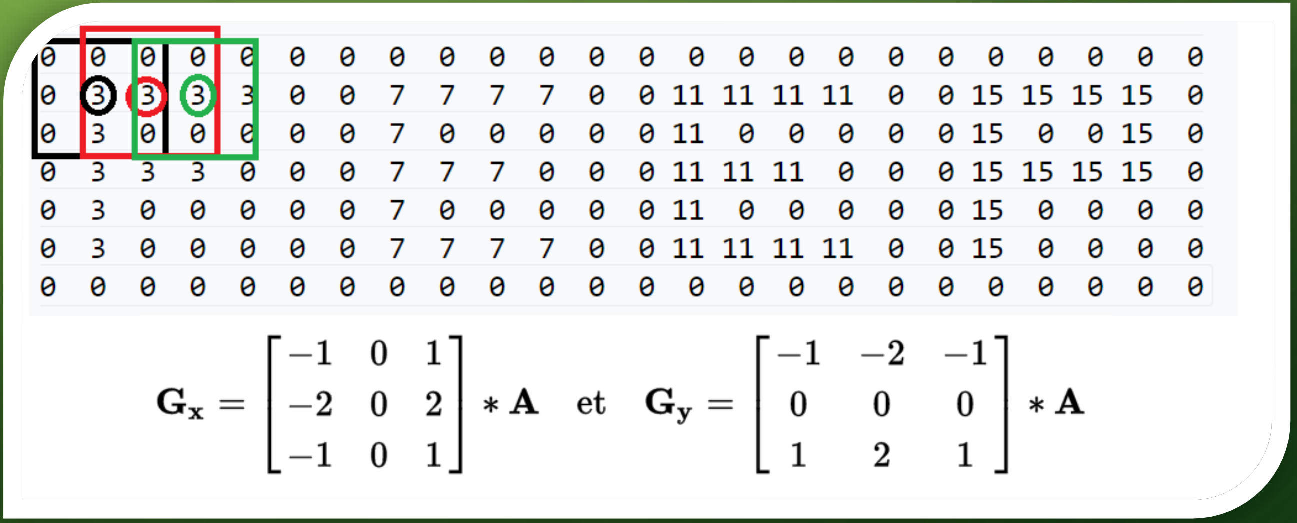


```
P2
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 0 0 0 0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0 0  0 15  0  0  15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0     0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

- We will use a VHDL library which will help us in reading the image pixels
- This part of code wont be synthesizable, It is a testbench !

PIXEL BY PIXEL PROCESSING

- In this part, we will process the image we previously read
- The processing step consists of applying the convolution matrix for each pixel
- So we will need to traverse the image pixel by pixel and in each clock cycle calculate the gradient value for the corresponding sub image
- We could use Sobel here as edge detection operator (Sobel uses 2 Convolution Matrix



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	3	3	3	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	15	15	15	0
0	3	0	0	0	0	0	7	0	0	0	0	0	11	0	0	0	0	0	15	0	0	15	0
0	3	3	3	0	0	0	7	7	7	0	0	0	11	11	11	0	0	0	15	15	15	15	0
0	3	0	0	0	0	0	7	0	0	0	0	0	11	0	0	0	0	0	15	0	0	0	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad \text{et} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{A}$$

RESULTING IMAGE STORE

- So we have got for each pixel the corresponding gradient value
- We store the resulting values in a new image which will show the detected edges of the original one !

NEEDED RESOURCES

- As we will design a test bench model for edge detection the test will be possible with the Modelism simulator so only a computer with the Modelsim software is needed to achieve the project.

DEADLINE

- The project’s deadline is set for the 16th of January 2017

THANK YOU
for your
ATTENTION!