COMPUTER RESEARCH

Framing Professor: Pawel WZIETEK

RECOGNITION OF SHAPES BY IMAGE CORRELATION

Directed by: Nabil EL FODIL

Xavier

GOEMAN

Sebastien

GIRARD

Organization chart

First session: Discovery of the project, understanding of the required functions.

Second session: Game programming begins
'conversion'.

Third session: end of the programming of "conversion" by Nabil and beginning of programming of "Fourier" by Sébastien and Xavier.

Fourth session: We do not understand how the "fourn" function works and what it was supposed to return despite the book provided.

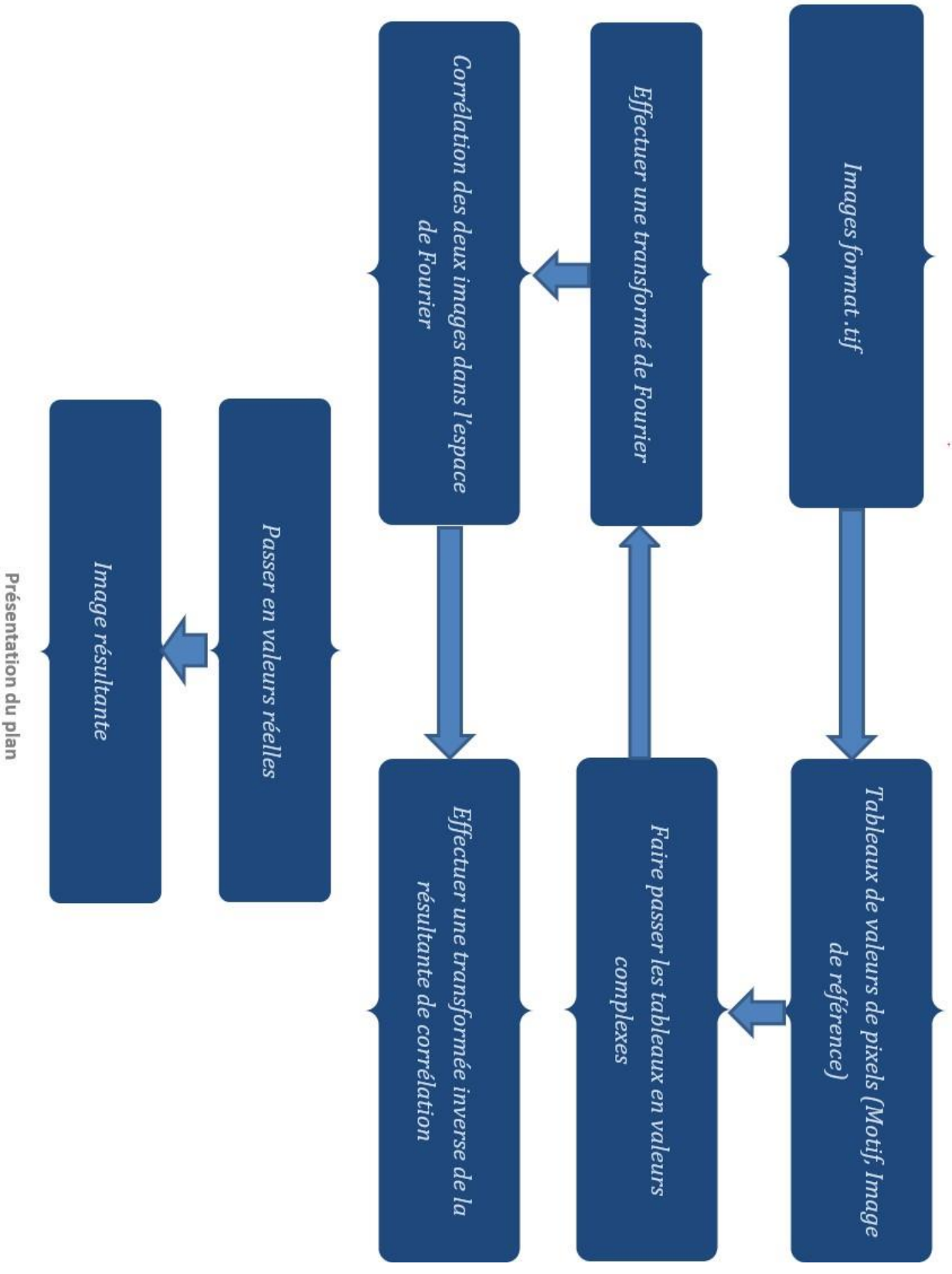
Fifth session: beginning of correlation Xavier and Sébastien, beginning of writing of the report byNabil.

Sixth session: Correction of "correlation" and "Fourier" that did not work.

Seventh session: Absent reference professor, we blocked on the creation of the resulting image, and we could not exploit the results returned by the "fourn" function.

Eighthsession: problem solved, the whole program works for full images but is not accurate enough, so we have to resume the function of "correlations" so that it can be done only on the contours of the shapes and not on thewholecircle.

Ninth session: finalizing the project, works for full and empty images, end of writing of the report and slideshow for the defense.


Images format .tif
Tableaux de valeurs de pixels (Motif, Image de référence)
Faire passer les tableaux en valeurs complexes
Effectuer une transformé de Fourier
Effectuer une transformée inverse de la résultante de corrélation
Corrélation des deux images dans l'espace de Fourier
Passer en valeurs réelles
Image résultante
Présentation du plan

I-I ntroduction:

The correlation-recognition project involves creating a C-language program, which allowspredefined shapes to be recognized and located in a "pattern"-rated image.

This project is spread over seven stages to be carried out before achieving the final result:

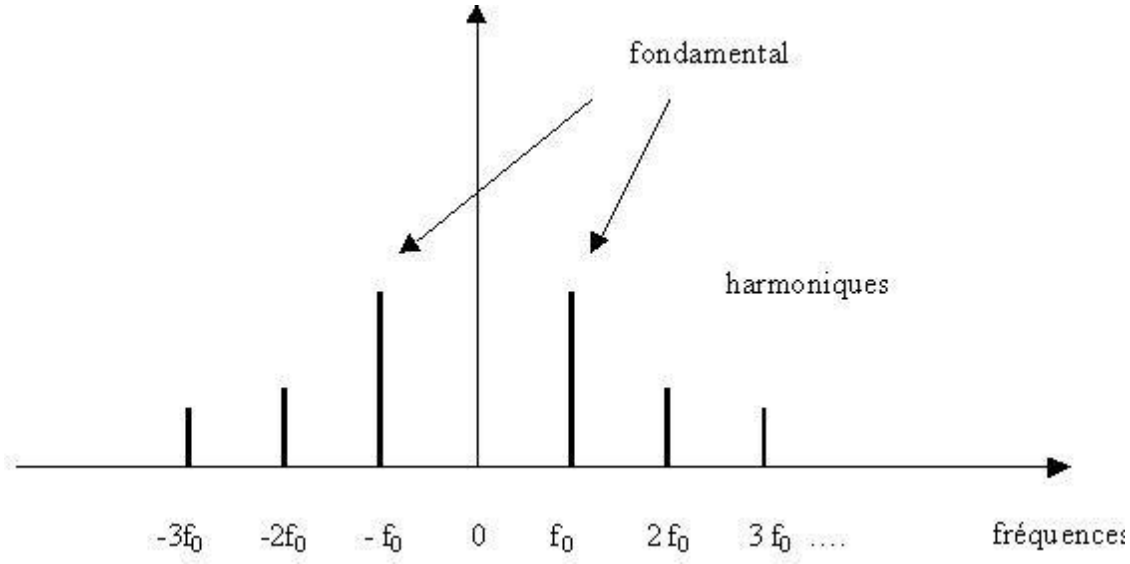
- Conversion of the image to format (.tif) and size (512x512) adapted, using Xnviewsoftware.
- Filling a table with pixel values.
- Switch to complex values.
- Make a Fourier transformation.
- Correlation of the two images in Fourier's space.
- Perform a reverse Fourier transformation of the result of correlation.
- Switch to real values

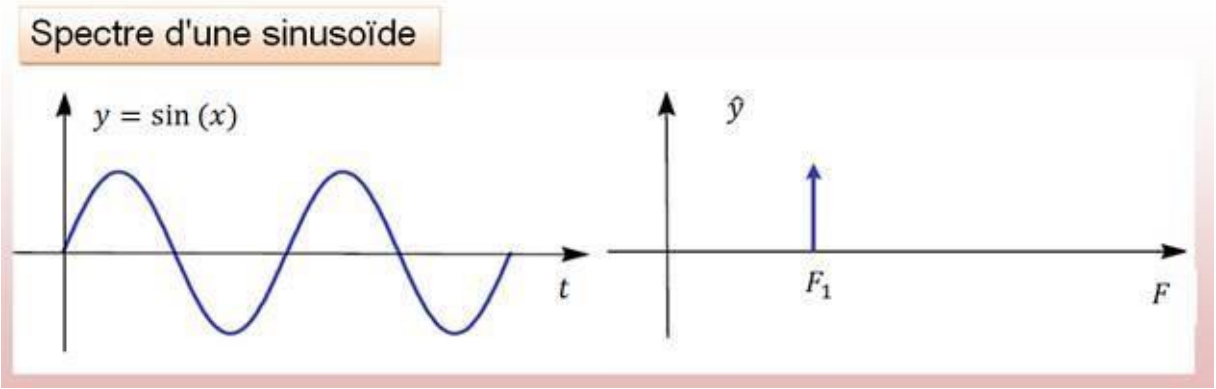
II- Mathematical definitions and tools:

a) Transformed by Fourier:

Joseph FOURIER, a French mathematician, stated in a memoir dated 1807 that it was possible, under certain conditions, to break down a periodic function f in the form of an infinite sum of sinusoidal signals.

$$f(V) \int_{-\infty}^{+\infty} F(T)E^{-2i\text{-}v}Dt$$





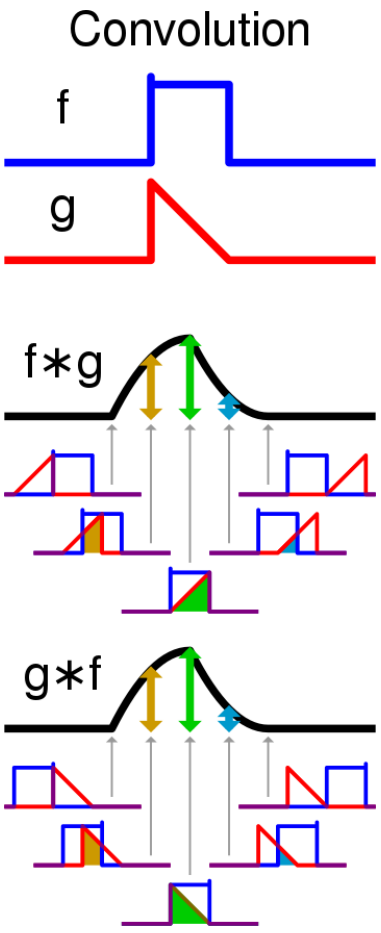
b) Convolution:

The convolution product generalizes the idea of slippery average, and is the mathematical representation of the notion of linear filter. It applies to both temporal data (for example in signal processing) and spatial data (in image processing). In statistics, a very similar formula is used to define cross-correlation.

The Fourier process of the product f g of two functions is not the product of Fourier de f and g. To recover this product, one must introduce the concept of convolée of two functions

Is f,g $C^{+\infty}_{0}$. We Defines The convolved de f * g de f And g By The formula:

$$(f(x) = \int_{-\infty}^{+\infty} F(X - T)G(T)Dt. \text{ (ii)}$$



The principle of correlation of a function is to convolution a function with its conjugate.

c) Reverse transformation:

As with Fourier's series decomposition, it is fundamental to try to "rebuild" the f function using Fourier'stransformed s, which will be noted the reverse transformation.

Either f a function f, we define its transformation of Fourier inverse by

$$F(f)(K)\int F(\underset{R}{T})E^{2i\text{-}Kt}Dt \; K \in R$$

III- From theory to CodeBlocks:

a) Introduction: Appendix 1

After introducing the above definitions and mathematical tools, we try to make them into C language, so that they can be used in image processing and form recognition.

The code is made up of alibrary, or we declare all our functions.

The recognition of forms being carried out in the space of Fourier (domains of complexes), we introduce a structure rated Complex that will consist of two values, one real and one second imagunaire. Therefore, when "Complex X" is used, variable X will consist of two values, X.re and X.im.

The second structure will be used when converting the actual image into a complex, with "width" and "height" that defines the size of the image, with respect to "cdata", it is a structure that recovers and stores the values of the pixels of the image in a table.

We then have the statement of all the functions that we will use in the program.

b) Co-nversion: Appendix 2

This part of the program pleases the instructions 'conversion' allows us to convert the image into a complex,with a zero imaginary part.

We try to keep an equal image size (before and after conversion), we will use "imc.height" and "imc.width" for this.

Imc.cdata is an "imc.Height"-sized table × "imc.width," which contains all the values of pixels that are changed. The for loop is used to fill the complex table with real values, while handing over the imaginary part of the pixel to 0.

The second part of the function: complex_en_int is to iron in real once the correlation is made. When the pixel values are no longer between 0 and 255, which prevents the re-creation of an image, then a standardization of pixels defines as such:

whenx is the smallest value returned by Reverse Fourier.

ax'b''255' when x is the highest value returned by Reverse Fourier.

In order to determine the values of a and b, for all pixel vaors, we solved a system of two equations with two unknowns.

We get this way:

a- $\frac{255}{(max-min)}$ b-$\frac{(255\times min)}{(Max-min)}$

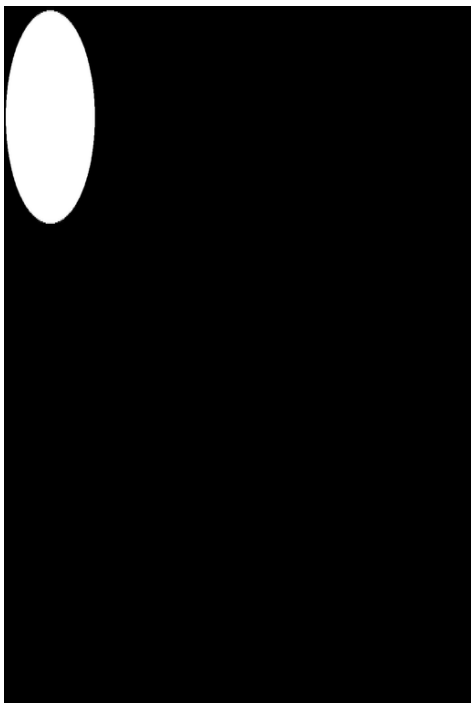
Max and min are determined by a loop that runs through the whole picture.

c) Fourier: Appendix 3

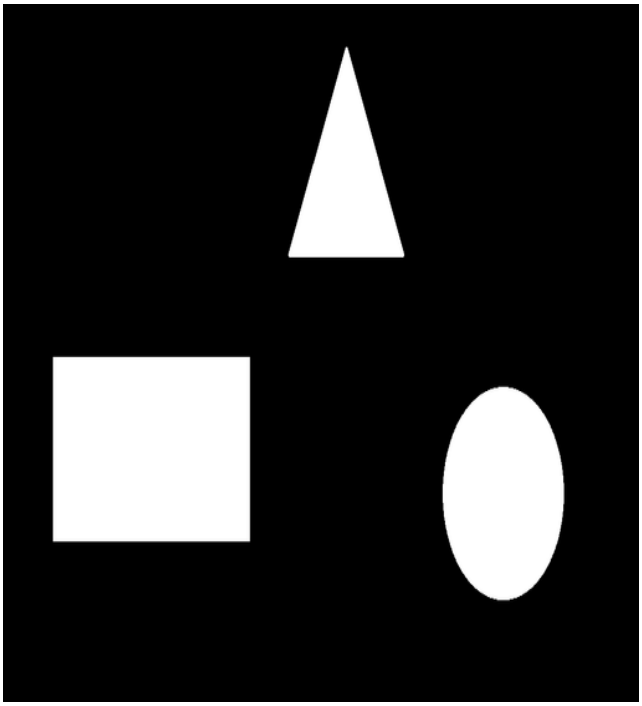
In order to facilitate the appeal of the fourn function (which was provided in the project) throughout the project, we create the furfunction, which allows us to avoid having to access the full program provided by default. This function acts directly onthe tables of reference and pattern images.

d) Correlation: Appendix 4

This part of the program consists of multiplying the two tables constituting the reference image and the pattern, which are in complex values, which amounts to a simplemultiplication of two complex values. The result is then obtained below.

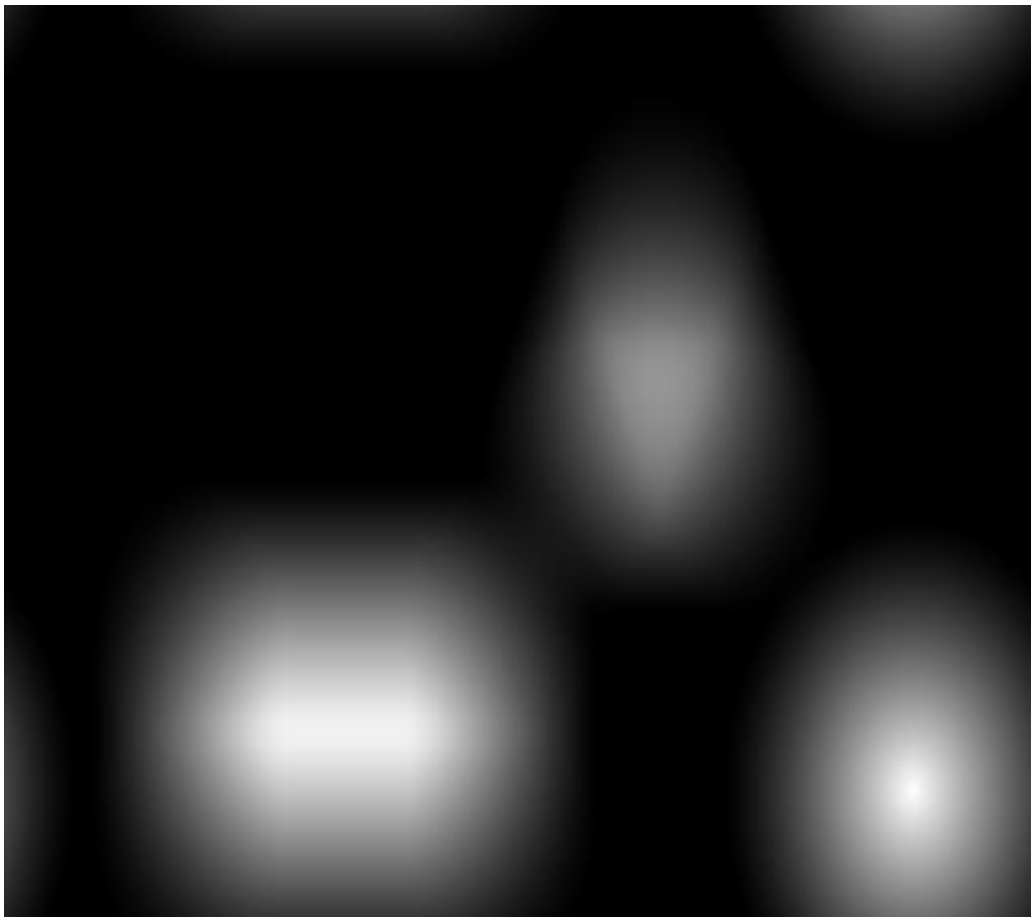


Pattern image



Reference image

=

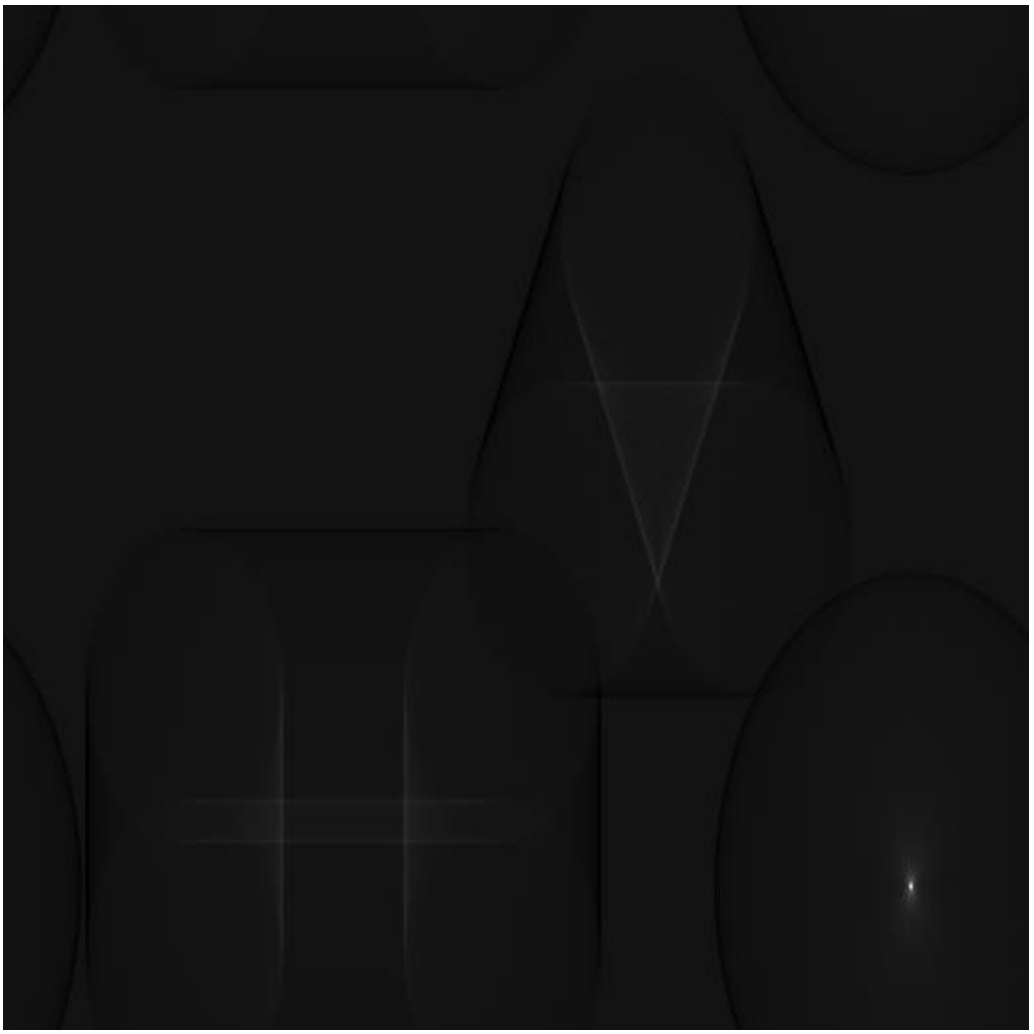


Resulting image

e) Correlation2: Appendix 5

To achieve a correction factor, a second correlation is introduced, which consists of applying a correlation only on the outline of the pattern image and that of reference.

The correlation2, acts as a filterin the frequency domain,so we will have a better accuracy on the resulting image, so we get by correlating the same images as above (pattern image and reference image) the following result:



Mathematical demonstration:

$f(x) \rightarrow F(k)$

$f'(x) \rightarrow ikf(k)$

$f(x,y) \rightarrow F(k_x,k_y)$

$\frac{\partial f}{\partial x} \rightarrow ik_xF(k_x,k_y)$

C(x,y)-∫∫ f (x - x- y - y) m(x-, y)dx'dy'

$\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\ \vec{\nabla F}$

The same goes for the m(x,y)function, which is the function that represents The pattern image.

$\vec{\nabla}_{F\nabla m} - \iint \frac{\partial f}{\partial x}(x - x,\ y - y)\ \frac{\partial m}{\partial x}(x^-,\ y)dx'dy'\ \frac{\partial f}{\partial}\ \frac{\partial m}{\partial}$

$(ik_xF)\ (ik_xM)\ (ik_yF)\ (ik_yM)$

$(k_x2-k_y2)FM$

IV- Conclusion:

This computer project is part of the theme of image processing, which is itself widely used in technologies and intonative versions of several industries, such as road safety with license plate recognition, and that of speed limits, and has post-production verification in order to adapt products to sales and consumption criteria. The model on which we have worked to its limits, it is only two-dimensional, and is adapted only to basic shapes such as squares and circles.

V-Appendixes:

a) Appendix 1: definitions.h

```
#define DEFINITIONS_H_INCLUDED

#include "../tifwrap/tifwrap.h"

typedef struct
{
  float re, im;
} Complex;

typedef struct
{
  uint32 width;
  uint32 height;
  Complex *cdata;
} Image_complexe;

/*conversion image réelle en image complexe */
Image_complexe  int_en_complexe(bwimage_t  im);

/*coversion image complexe en image réelle */
bwimage_t  complexe_en_int(Image_complexe cim);

/*Transformé de fourier de l'image complexe*/
 void fourier (Image_complexe imc, int signe);

Image_complexe  produit(Image_complexe image, Image_complexe motif);

Image_complexe  produit2(Image_complexe ciml, Image_complexe cim2);

void fourn(float data[], unsigned long nn[], int ndim, int isign);

#endif // DEFINITIONS_H_INCLUDED
```

b) Appendix 2: conversion.c

```
#include "../tifwrap/tifwrap.h"
#include "definitions.h"

Image_complexe  int_en_complexe(bwimage_t  im)
{
    bwimage_t image;
    Image_complexe imc;

    int i,j;

    imc.height=im.height;
    imc.width=im.width;
    imc.cdata= (Complex*) malloc((imc.height) * (imc.width)*sizeof(Complex));

    for(i=0;i<imc.height*imc.width;i++)
        {
            imc.cdata[i].re=im.rawdata[i];
            imc.cdata[i].im=0;
        }
return imc;
}

bwimage_t  complexe_en_int(Image_complexe cim)
{
    bwimage_t im;
    Image_complexe imc;

    float min = 1000,max =0;
    int i;
    im.height=cim.height;
    im.width=cim.width;
```

```
33      im.data=(uint8**)malloc(im.height*sizeof(uint8*));
34      im.rawdata=(uint8*)malloc((im.width) * (im.height) * sizeof(uint8));
35
36      for(i=0; i<im.height; i++)
37          {
38              (im.data)[i]=im.rawdata + (i*im.width);
39          }
40
41      for(i=0;i<im.height*im.width;i++)
42          {
43
44              if (cim.cdata[i].re<min)
45                  min = cim.cdata[i].re;
46              if (cim.cdata[i].re>max)
47                  max= cim.cdata[i].re;
48          }
49
50      float b=(-255*min)/(max-min);
51      float a= 255/(max-min);
52
53      for(i=0;i<im.height*im.width;i++)
54          {
55              im.rawdata[i]=(uint8)(a*cim.cdata[i].re +b);
56          }
57
58      printf("min2 = %f \nmax2= %f\n",min,max);
59      printf("pour la normalisation :\na=%lf\nb=%lf\n",a,b);
60
61  return im;
62  }
```

c) Appendix 3: furnace.c

```
#include <math.h>
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr
#include "../tifwrap/tifwrap.h"
#include "definitions.h"

 void fourier (Image_complexe imc, int signe)
{
    unsigned long nn[2];
    nn[0]=imc.width;
    nn[1]=imc.height;
    int isign=signe;

    fourn((float*)imc.cdata,nn,2,isign);

}
```

d) Appendix 4: correlation.c

```c
#include "../tifwrap/tifwrap.h"
#include "definitions.h"


Image_complexe  produit(Image_complexe ciml, Image_complexe cim2)
{
    Image_complexe imp;
    int i;

    imp.height=ciml.height;
    imp.width=ciml.width;
    imp.cdata=(Complex*)malloc((ciml.width)*(ciml.height)*sizeof(Complex));

/******* MULTIPLICATION DES PIXRELS DE L'IMAGE ET DU MOTIF*******/

    for(i=0;i<ciml.height*ciml.width;i++)
        {
            imp.cdata[i].re = ciml.cdata[i].re * cim2.cdata[i].re - (ciml.cdata[i].im * cim2.cdata[i].im);
            imp.cdata[i].im = ciml.cdata[i].re * cim2.cdata[i].im + (ciml.cdata[i].im * cim2.cdata[i].re);
        }

return imp;
}
```

e) Appendix 5: correlation2.c

```c
#include "../tifwrap/tifwrap.h"
#include "definitions.h"

Image_complexe  produit2(Image_complexe ciml, Image_complexe cim2)
{
    Image_complexe imp;
    int c,l,i;
    float kx, ky;

    imp.height=ciml.height;
    imp.width=ciml.width;



    imp.cdata=(Complex*)malloc((ciml.width)*(ciml.height)*sizeof(Complex));
/******* MULTIPLICATION DES PIXRELS DE L'IMAGE ET DU MOTIF*******/
    for(c=0;c<ciml.height;c++)
        {
            for(l=0;l<ciml.width;l++)
                {
                    if(c<=imp.width/2)
                        {
                            if(l<=imp.height/2)
                                {
                                    ky=c/(float)imp.height;
                                    kx=l/(float)imp.width;
                                    i=l*imp.width+c;
                                    imp.cdata[i].re = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].re - (ciml.cdata[i].im * cim2.cdata[i].im));
                                    imp.cdata[i].im = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].im + (ciml.cdata[i].im * cim2.cdata[i].re));
                                }
                            else
                                {
                                    ky=c/(float)imp.height;
                                    kx=(imp.width-l)/(float)imp.width;
                                    i=l*imp.width+c;
                                    imp.cdata[i].re = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].re - (ciml.cdata[i].im * cim2.cdata[i].im));
                                    imp.cdata[i].im = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].im + (ciml.cdata[i].im * cim2.cdata[i].re));
                                }
                        }
                    else
                        {
                            if(l<=imp.height/2)
                                {
                                    ky=(imp.height - c)/(float)imp.height;
                                    kx=l/(float)imp.width;
                                    i=l*imp.width+c;
                                    imp.cdata[i].re = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].re - (ciml.cdata[i].im * cim2.cdata[i].im));
                                    imp.cdata[i].im = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].im + (ciml.cdata[i].im * cim2.cdata[i].re));
                                }
                            else
                                {
                                    ky=(imp.height - c)/(float)imp.height;
                                    kx=(imp.height-l)/(float)imp.width;
                                    i=l*imp.width+c;
                                    imp.cdata[i].re = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].re - (ciml.cdata[i].im * cim2.cdata[i].im));
                                    imp.cdata[i].im = (kx*kx + ky*ky)*(ciml.cdata[i].re * cim2.cdata[i].im + (ciml.cdata[i].im * cim2.cdata[i].re));
                                }
                        }
                }
        }

return imp;
}
```

f) Appendix 6: tiftest.c (hand)

```c
#include "../tifwrap/tifwrap.h"
#include "definitions.h"

int main()
{

    bwimage_t motif,image,impfin;
    Image_complexe imc1,imc2,imp;
    int retval=0;
    int retval2=0;
    int i;

    retval=EEALoadImage("test2.tif", &image); // fonction pour charger l'image utilisateur
        printf("Load: %d\n", retval);

    retval2 =EEALoadImage("carre_motif.tif", &motif); //fonction pour charger l'image de référence
      printf("Load: %d\n", retval2);


    imc1 = int_en_complexe(image);
    imc2 = int_en_complexe(motif);
    fourier(imc1,1);
    fourier(imc2,1);

    imp= produit2(imc1,imc2);
    fourier(imp,-1);
    impfin=complexe_en_int(imp);

    retval2=EEADumpImage("corr.tif", &impfin);


  return 0;
}
```

VI- Sitography:

https://www.math.u-ps ud.fr/~harari/enseignement/math256ter/

https://fr.wikipedia.org

Table of Materials