Universidad Autónoma de querétaro

Programación

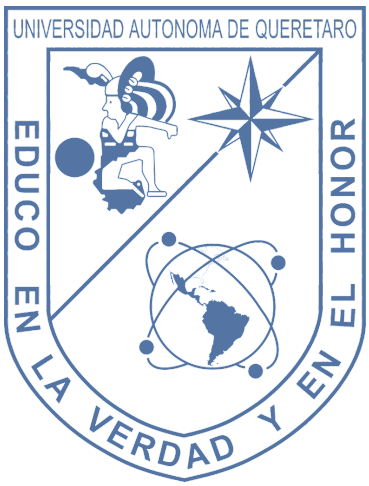
Facultad de Ingeniería

Análisis de la distribución de tamaños de partículas de recubrimiento de Renio y Boro

Integrantes:
Yessenia Guadalupe Hernández Rico
Citlali Araceli Rosales García
Irasema Toscano Ramírez

Ingeniería en Nanotecnología
Profesor:
MC. José de Jesús Santana Ramírez

23 de Mayo del 2018



I. Introducción.

Gracias a lo aprendido en el semestre, específicamente, del uso de programación en c++, se pueden realizar distintas herramientas que faciliten el trabajo, ya sea, en una investigación, proyecto, trabajo escolar, etc.
En el siguiente trabajo se utilizó este leguaje de programación para hacer el análisis de una imagen de un recubrimiento de Renio y Boro, a partir del tamaño de las partículas,y observar así su distribución en la imagen.

II. Problema.

Hacer el análisis de la distribución por tamaños de la imagen de un recubrimiento de Renio y Boro, a partir de la implementación de un algoritmo de reconocimiento de imagen con un programa hecho en c++; Dicho programa debe ser orientado a objetos y estar organizado por clases.

III. Marco teórico

A. Renio

1) Usos y Aplicaciones: El renio se usa como un aditivo para aleaciones a base de tungsteno y molibdeno para proporcionar propiedades útiles. Estas aleaciones se utilizan para filamentos de hornos y máquinas de rayos X. También se usa como material de contacto eléctrico ya que resiste el desgaste y resiste la corrosión del arco.
Los catalizadores de renio son extremadamente resistentes a la intoxicación (desactivación) y se utilizan para la hidrogenación de productos químicos finos. Algo de renio se usa en aleaciones de níquel para fabricar palas de turbina de un solo cristal.

B. Boro

1) Usos y Aplicaciones: El boro amorfo se usa como un encendedor de combustible para cohetes y en bengalas pirotécnicas. Le da a las bengalas un distintivo color verde.
Los compuestos más importantes de boro son el ácido bórico (o bórico), el bórax (borato de sodio) y el óxido bórico. Estos se pueden encontrar en gotas para los ojos, antisépticos suaves, polvos de lavar y esmaltes para azulejos. Borax solía usarse para hacer cloro y como conservante de alimentos.
El óxido bórico también se usa comúnmente en la fabricación de vidrio de borosilicato (Pyrex). Hace que el vidrio sea resistente y resistente al calor. Los textiles y el aislamiento de fibra de vidrio están hechos de vidrio de borosilicato.
El octaborato de sodio es un retardante de llama. El isótopo boro-10 es bueno para absorber neutrones. Esto significa que puede usarse para regular reactores nucleares. También tiene un papel en los instrumentos utilizados para detectar neutrones.

C. La visión artificial

La visión artificial es una disciplina que tiene como finalidad, reproducir artificialmente el sentido de la vista mediante el procesamiento e interpretación de imágenes, capturadas con distintos tipos de sensores, y utilizando para ello programas de computación [1].
La visión artificial, se define tradicionalmente en cuatro etapas principales:

- La primera fase, consiste en la **Captura de las imágenes** digitales mediante algún tipo de sensor como las cámaras.
- La segunda etapa consiste en el tratamiento digital de las imágenes. En esta etapa de **Preprocesamiento**, es donde, mediante filtros y transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma.
- La tercera fase se conoce como **Segmentación**, y consiste en la extracción de los elementos que nos interesan de la imagen.
- Por último, se llega a la etapa de **Reconocimiento**, en ella se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

En nuestro proyecto nos enfocaremos a las dos últimas etapas, la segmentación y el reconocimiento de imagen.

D. La programación orientada a objetos

Para la programación orientada a objetos es importante conocer los siguientes conceptos:

- Objeto: entidad que tiene unos atributos (variables) y unos métodos para operar con ellos.
- Mensaje: es el nombre de uno de los métodos de un objeto. Cuando se pasa un mensaje a un objeto, éste responde ejecutando el código de la función asociada.
- Método: determina como debe de actuar un objeto cuando se produce el mensaje asociado. En C++ un método es una función miembro del objeto.
- Clases: definición de un tipo de objetos.

E. Segmentación

La segmentación es uno de los pasos más importantes y más complicados en el procesado de imágenes, ya que permite la cuantificación y la

visualización de los objetos de interés [2].
En otras palabras, la segmentación consiste en la división de la imagen en las partes u objetos que la forman.
Esta división de la imagen se hace atendiendo a las características similares que existen entre los píxeles de la misma.
Existen varias técnicas de segmentación:

- Técnicas de segmentación basadas en los valores del píxel.
- Técnicas de segmentación basadas en el área.
- Técnicas de segmentación basadas en la extracción de bordes.
- Técnicas de segmentación basadas en la física.

F. Segmentación por umbralizado

La segmentación por umbralizado se basa en los valores del pixel.
El método de segmentación por umbralizado permite convertir una imagen en color o en escala de grises a binario de forma que los píxeles cuyos niveles de intensidad superen cierto umbral y tengan distinto valor al resto [3].
Algunas de las aplicaciones de la segmentación por umbralizado son:

- Medicina: detección de células en citología; localización de fracturas; análisis de ecografrías; imágenes de rayos X.
- Industria:supervisión automática de procesos industriales, como control de calidad, vigilancia o detección de fallos.
- Análisis de fotografías áreas o por satélite: clasificación de diferentes tipos de vegetación para el estudio de su degradación; clasificación y control de cultivos de agricultura; detección de objetos.

Existen muchas técnicas para definir el umbral, pero las más empleada es mediante el análisis del histograma de la imagen [4].
Si los valores de intensidad del objeto y del fondo difieren claramente, el histograma de la imagen presentará un aspecto bimodal, por lo tanto se logrará una separación excelente entre el objeto y el fondo, esto sería el umbral, al cual lo llamaremos umbral T [5] así cualquier píxel de la imagen que esté por encima del umbral T será considerado como objeto y cualquier píxel de la imagen que esté por debajo del umbral será considerado como fondo.
Por lo tanto se puede definir a la umbralización como:

$$g(x,y) = \begin{cases} sif(x,y) \geq umbralT \\ 0encualquierotrocaso \end{cases}$$

Pero para no hacerlo más complicado es mejor realizarlo de la manera contraria:

$$g(x,y) = \begin{cases} sif(x,y) \leq umbralT \\ 0encualquierotrocaso \end{cases}$$

IV. Filtros morfológicos

En matemáticas se utiliza la palabra morfología para designar una herramienta que se utiliza para extraer los componentes de una imagen empleados para representar y describir la forma de una región.
Estas operaciones se aplican a imágenes en blanco y negro ya que se utilizan funciones booleanas para realizarlas.

A. Operaciones lógicas

El álgebra de Boole es un sistema matemático para formular proposiciones lógicas mediante símbolos así se pueden resolver estos problemas como si se tratara de álgebra normal.
Este tipo de álgebra se utiliza para el análisis y diseño de sistemas digitales. Las variables booleanas solo pueden tener el valor 0 o 1.[14]

V. La binarización de imágenes

Como vimos anteriormente, uno de los pasos más importantes del análisis de imágenes es la de segmentación en el cual las imágenes se pueden segmentar por medio de varios métodos, esto significa que a través de la segmentación de imágenes uno puede observar ciertos objetos o poner filtros a las imágenes que queramos, en este proyecto se utilizará la segmentación por umbralizado en la que se binarizará una imagen tomando valores de 0 hasta 255, en la que 0 es el color negro y 255 el blanco.
Para realizar el código de la umbralización en C++ hemos tenido también que estudiar el análisis de imágenes a través de los histogramas de imagen en las que se diferencía el objeto del fondo y se le da un color más obscuro (siendo el color negro) del fondo blanco.
Al finalizar nuestro estudio e investigación de los histogramas de imágenes y la de elegir un parámetro de umbralización pudimos realizar nuestro código en el lenguaje C++.

VI. Vecinos

Los vecinos hacen referencia a los espacios o pixeles que rodean a un pixel central que uno puede determinar.
Hay varios tipos de vecinos, como lo son los de 4 y 8 vecinos, esto quiere decir que ya sea que 4 pixeles son los que rodean a un pixel central o que sean 8 los que lo redeen, esto puede ejemplificarse de una mejor manera con las siguientes imágenes:

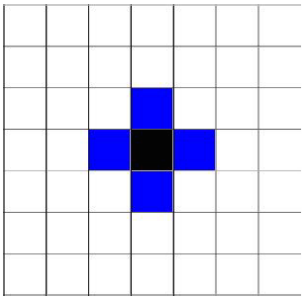


Fig. 1: 4 vecinos. Los vecinos son marcados con color azul y el pixel central de color negro.

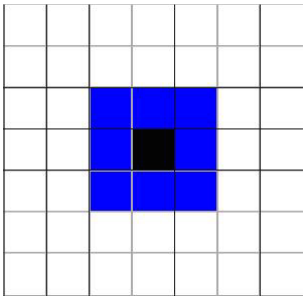


Fig. 2: 8 vecinos.

A. *Para qué utilizar a los vecinos?*

Bueno, como vimos en la parte de la segmentación de imágenes por umbralizado la imagen queda de una manera binaria (ya sea 0 y 1 o 0 y 255), esto transforma la imagen a blanco y negro y lo que nos interesa a nosotras es que los pixeles marcados de color negro sean las partículas pero para que el programa las pueda identificar debe leer los pixeles al rededor de los pixeles de color negro para marcar la ubicación de las partículas y luego poder realizar alguna otra acción, pero esta parte es sumamente importante, tal como la umbralización de las imágenes.

VII. Código de vecinos

El código de vecinos lo juntamos con el de binarizado para que a partir de la imagen en blanco y negro nos mapee la imagen y lo lea.

A. *pgmbasica1.h*

```
#ifndef PGMBASICA_H
#define PGMBASICA_H

#include <iostream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
using namespace std;


typedef struct imagen{
    char nombre[20];
    int cant_comen;
    int ancho;
    int alto;
    int grises;
    unsigned char im[800][800];
} IMAGEN;

int es_pgm(char *nombre){
    FILE *ent;
    unsigned char c1,c2;
    printf("\nChequeando: %s\n",nombre
        );
    if ((ent = fopen(nombre,"rb")) ==
        NULL)
        return 0;
    c1=fgetc(ent);
    c2=fgetc(ent);

    fclose(ent);
    if ((c1=='P') && (c2=='5'))
        return 1;
    else
        return 0;
}

IMAGEN lee_imagen(char *nombre)
{
    IMAGEN imagen;
    unsigned int c1,c2;
    unsigned char carac;
    FILE *ent;
    int i,j, ancho,alto,grises;

    ent = fopen(nombre,"rb");
    c1=fgetc(ent);
    c2=fgetc(ent);
    fgetc(ent); //Elimina el separador
    carac=fgetc(ent);
    imagen.cant_comen=0;
    while(carac=='#'){
        imagen.cant_comen=imagen.
            cant_comen+1;
        carac=fgetc(ent);
        while(carac!='\n')
            carac=fgetc(ent);
        carac=fgetc(ent);
    }
    ungetc(carac,ent);
    fscanf(ent,"%d%d",&ancho,&alto);
    fscanf(ent,"%d",&grises);
    imagen.ancho = ancho;
    imagen.alto= alto;
    imagen.grises = grises;
    strcpy(imagen.nombre,nombre);
    for(i=0; i<imagen.alto;i++)
        for(j=0; j<imagen.ancho; j++){
            carac=fgetc(ent);
            imagen.im[i][j]=carac;
        }
    fclose(ent);
    return(imagen);
}

void muestra_datos_imagen(IMAGEN *
    imagen){
    printf("\nArchivo: %s",imagen->
        nombre);
    printf("\nComentarios : %d",imagen
        ->cant_comen);
    printf("\nAncho : %d",imagen->
        ancho);
```

```
    printf("\nAlto  : %d",imagen->alto
        );
    printf("\nGrises: %d\n",imagen->
        grises);
}

void grabaimagen(IMAGEN *imagen){
    int i,j;
    unsigned char c;
    FILE *sal;

/*
    int nro;
    char nombre[14],nombre1[6];
    nro = (rand()%(9999 - 1000 + 1)) +
         1000;
    itoa(nro,nombre1,10);
    strcpy(nombre,"Hac_");
    strcat(nombre,nombre1);
    strcat(nombre,".pgm");
*/

    printf("\nCreando : %s",imagen->
        nombre);
    sal=fopen(imagen->nombre,"wb");
    fprintf(sal,"P5\n");
    fprintf(sal,"# Creado por C.\n");
    fprintf(sal,"%d %d\n",imagen->
        ancho,imagen->alto);
    fprintf(sal,"%d\n",imagen->grises)
        ;
    for(i=0; i<imagen->alto;i++)
        for(j=0; j<imagen->ancho; j++)
        {
            c=imagen->im[i][j];
            fputc(c,sal);
            //printf(" ",imagen->im[i
                ][j]);
        }
        //printf("");
//  fputc(final,sal);
    fclose(sal);
}

void aclara_imagen(IMAGEN *imagen, int
     valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]+
                valor) > 255)
                imagen->im[i][j]=255;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] +
                    valor;
}

void oscurece_imagen(IMAGEN *imagen,
    int valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]-
                valor) <0)
                imagen->im[i][j]=0;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] -
                    valor;
}

void negativo_imagen(IMAGEN *imagen){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            imagen->im[i][j]=255 -
                imagen->im[i][j];
}

void binariza_imagen(IMAGEN *imagen,
    int umbral){
    int i,j;
   // printf("-------lectura de imagen
        ---------------");
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if (imagen->im[i][j] <
                umbral){
                imagen->im[i][j]=0;
               // printf("0");
                }
               else{
                imagen->im[i][j]= 255;
                //    printf("1");
                }
}

void imprime_imagen(IMAGEN *imagen,
    int umbral){
    int i,j;
    printf("-------lectura de imagen
        ---------------");
    cout<<endl;
    for(i=0; i < imagen->alto; i++){
     printf(" ");
        for(j=0; j < imagen->ancho; j
            ++){
            if (imagen->im[i][j] <
                umbral)

                printf("1");
               else
                printf("0");}
            cout<<endl;
    }

}
void cuenta_ceros(IMAGEN *imagen, int
    umbral){
     int i,j;
     int p;
     int contador=0;
     float perimetro;
     float restante;
     float final;
     int x=imagen->alto;
     int y=imagen->ancho;
     int datos[x][y];
     int datos1[x][y];
    printf("-------lectura de imagen
        ---------------");
    //cout<<endl;
    for(i=0; i < x ; i++){
```

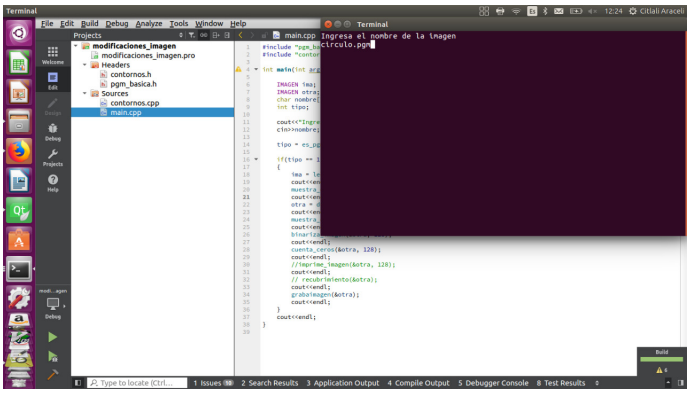
```
        for(j=0; j < y; j++)
             if (imagen->im[i][j] <
                 umbral)
                 datos[i][j]=1;
             else
                 datos[i][j]=0;
         }

    //cuenta unos horizontal
    p=0;
    for(i=0; i < x ; i++){
        for(j=0; j < y; j++){
           if(p != datos[i][j])
               datos1[i][j]=7;
           else
              datos1[i][j]=0;
          p=datos[i][j];
        }
     }
    //cuenta unos vertical
    for(j=0; j < x ; j++){
        for(i=0; i < y ; i++){
            if(p != datos[i][j])
            datos1[i][j]=7;
           else{
              if (datos1[i][j]!=7)
                  datos1[i][j]=0;}
            p=datos[i][j];
        }
        //cout<<endl;
       }
//Cuenta total
    contador=0;
      for(i=0; i < x ; i++)
          for(j=0; j < y ; j++)
              if( datos1[i][j]==7)
                contador++;



    //imprime ceros
    for(i=0; i < x; i++){
        printf(" \n");
       for(j=0; j < y; j++)
            printf("%i ",datos[i][j]);

}
   restante= (contador-1)/4;
   perimetro= contador-restante;
   final=perimetro/10;

    cout<<endl<<"El perimetro de cada
        particula en nanometros es: ";
     printf("%.2f",final); cout<<"nm."
         ;



}

void histograma(IMAGEN *imagen){
    int i,j;
    int mayor;
    float aux;
    int histo[256];
    for (i=0; i< imagen->alto; i++)
        for(j=0; j< imagen->ancho; j
            ++)
                histo[imagen->im[i][j]]=
                   histo[imagen->im[i][j
                   ]]+1;

    mayor = 0;
    for(i=0; i<256; i++)
       if (histo[i]>mayor)
            mayor = histo[i];

    for (i=0; i<256; i++){
        aux=(float)(histo[i])/mayor;
        aux=aux*100*2;
        histo[i] = (int)(aux);
    }

}

IMAGEN laplaciano(IMAGEN *imagen){
    IMAGEN ret;
    int numero;
    char nro[3];

    int m[3][3],i,j,k,l,s;
    m[0][0]= 0; m[0][1]=-1; m[0][2]=
        0;
    m[1][0]=-1; m[1][1]=-4; m
        [1][2]=-1;
    m[2][0]= 0; m[2][1]=-1; m[2][2]=
        0;

    for(i=1; i< imagen->alto-2; i++){
        for(j=1; j< imagen->ancho-2; j
            ++){
            s=0;
            for(k=0; k<3; k++)
                for(l=0; l<3; l++)
                    s=s+m[k][l]*(
                        imagen->im[i+k
                        ][j+l]);
        }
        ret.im[i][j]= s;
    }

    strcpy(ret.nombre,"Lp_");
    strcat(ret.nombre,imagen->nombre);
    ret.cant_comen = imagen->
        cant_comen;
    ret.ancho      = imagen->ancho;
    ret.alto       = imagen->alto;
    ret.grises     = imagen->grises;
    return ret;
}

IMAGEN duplica_imagen(IMAGEN *imagen){
    IMAGEN ret;
    int i,j;
    printf("\nDuplicando: %s\n",imagen
        ->nombre);
    strcpy(ret.nombre,"Bi_");
    strcat(ret.nombre,imagen->nombre);
    ret.cant_comen = imagen->
        cant_comen;
    ret.ancho      = imagen->ancho;
    ret.alto       = imagen->alto;
    ret.grises     = imagen->grises;
    for(i=0; i<imagen->alto; i++)
        for(j=0; j<imagen->ancho; j++)
            {
```

```
                ret.im[i][j]=imagen->im[i
                    ][j];
                //printf(" xxxx ",ret.im[i
                    ][j]);
            }
        return ret;
}

#endif // PGMBASICA_H
```

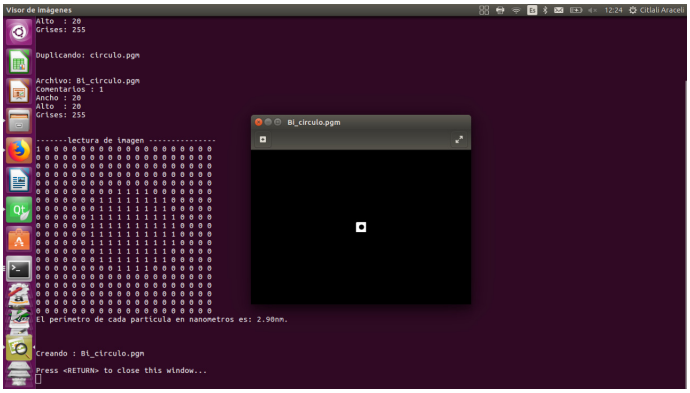
B. main.cpp

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
#include "pgmbasica.h"

using namespace std;

int main(int argc, char *argv[]){

    IMAGEN ima;
    IMAGEN otra;
    char nombre[20];
    int tipo;

    cout<<"Ingresa el nombre de la
        imagen"<<endl;
    cin>>nombre;

    tipo = es_pgm(nombre);

    if(tipo == 1)
    {
        ima = lee_imagen(nombre);
                cout<<endl;
                muestra_datos_imagen(&
                    ima);
                cout<<endl;
                otra = duplica_imagen
                    (&ima);
                cout<<endl;
                muestra_datos_imagen(&
                    otra);
                cout<<endl;
                binariza_imagen(&otra,
                     128);
                cout<<endl;
                cuenta_ceros(&otra,
                    128);
                cout<<endl;
                //imprime_imagen(&otra
                    , 128);
                cout<<endl;
                // recubrimiento(&otra
                    );
                cout<<endl;
                grabaimagen(&otra);
                cout<<endl;
    }
    cout<<endl;
}
```

VIII. Debug del contador de perímetro



(a) Eliges la imagen que vas a transformar con la binarización/umbralización que esté en el formato ".pgm" nuevamente



(b) Automáticamente se hace una copia binarizada de la imagen que se guarda en la carpeta principal, además de que se muestra con 1 y 0 la imagen, como último la imagen se parece a la original visualmente, el programa cuenta su perímetro y lo calcula con nanómetros.

IX. Las clases en C++

A. Por qué utilizar clases en C++?

Aunque se quisiera hacer, poner todo lo que se va a relaizar en el programa dentro de la función del main.cpp no es muy conveniente pues aunque uno lo estructure de manera ordenada el código no se verá estéticamente correcto ya que uno de los objetivos de la programación orientada a objetos en el cual se trata de encapsulas las funciones y sus métodos por separado y tan sólo en la parte principal del archivo, es decir, en el main.cpp llamaremos las funciones principales que se encuentran en las clases de los métodos junto con sus librerías correspondientes.
Otra ventaja de tener a los códigos acomodados en distintas partes además de un main.cpp es que es más fácil de visualizar los problemas y de actualizarlos de una manera más ordenada y práctica.

X. Interfaz e implementación

Con estos conocimientos que hemos mostrado anteriormente sobre las clases en la programación orientada a objetos podemos hablar ahora de la

interfaz y de cómo se implementa.
Imaginemos que la declaración de una clase le dice al compilador lo que hay que hacer y lo que no hay que hacer, es en este caso cuando se dice que es la interfaz de la clase.
Entonces la interfaz es cuando se declaran los métodos de una clase.
La implementación en la interfaz es cuando se muestra en código sobre el trabajo o lo que hacen los métodos de la clase.
En nuestro caso la interfaz es la "consola" en donde ocurre todo el programa y se muestran al usuario widgets o botones que le ayudan a realizar las acciones/métodos del programa.

XI. Códigos para la umbralización de nuestro análisis de imágenes

A. *pgmbasica.h*

```
#ifndef PGMBASICA_H
#define PGMBASICA_H

#include <iostream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
using namespace std;


typedef struct imagen{
    char nombre[20];
    int cant_comen;
    int ancho;
    int alto;
    int grises;
    unsigned char im[800][800];
} IMAGEN;

int es_pgm(char *nombre){
    FILE *ent;
    unsigned char c1,c2;
    printf("\nChequeando: %s\n",nombre
        );
    if ((ent = fopen(nombre,"rb")) ==
        NULL)
        return 0;
    c1=fgetc(ent);
    c2=fgetc(ent);

    fclose(ent);
    if ((c1=='P') && (c2=='5'))
        return 1;
    else
        return 0;
}

IMAGEN lee_imagen(char *nombre)
{
    IMAGEN imagen;
    unsigned int c1,c2;
    unsigned char carac;
    FILE *ent;
    int i,j, ancho,alto,grises;

    ent = fopen(nombre,"rb");
    c1=fgetc(ent);
    c2=fgetc(ent);
    fgetc(ent); //Elimina el separador
    carac=fgetc(ent);
    imagen.cant_comen=0;
    while(carac=='#'){
        imagen.cant_comen=imagen.
            cant_comen+1;
        carac=fgetc(ent);
        while(carac!='\n')
            carac=fgetc(ent);
        carac=fgetc(ent);
    }
    ungetc(carac,ent);
    fscanf(ent,"%d%d",&ancho,&alto);
    fscanf(ent,"%d",&grises);
    imagen.ancho = ancho;
    imagen.alto= alto;
    imagen.grises = grises;
    strcpy(imagen.nombre,nombre);
    for(i=0; i<imagen.alto;i++)
        for(j=0; j<imagen.ancho; j++){
            carac=fgetc(ent);
            imagen.im[i][j]=carac;
        }
    fclose(ent);
    return(imagen);
}

void muestra_datos_imagen(IMAGEN *
    imagen){
    printf("\nArchivo: %s",imagen->
        nombre);
    printf("\nComentarios : %d",imagen
        ->cant_comen);
    printf("\nAncho : %d",imagen->
        ancho);
    printf("\nAlto  : %d",imagen->alto
        );
    printf("\nGrises: %d\n",imagen->
        grises);
}

void grabaimagen(IMAGEN *imagen){
    int i,j;
    unsigned char c;
    FILE *sal;

/*
    int nro;
    char nombre[14],nombre1[6];
    nro = (rand()%(9999 - 1000 + 1)) +
         1000;
    itoa(nro,nombre1,10);
    strcpy(nombre,"Hac_");
    strcat(nombre,nombre1);
    strcat(nombre,".pgm");
*/

    printf("\nCreando : %s",imagen->
        nombre);
    sal=fopen(imagen->nombre,"wb");
    fprintf(sal,"P5\n");
    fprintf(sal,"# Creado por C.\n");
    fprintf(sal,"%d %d\n",imagen->
        ancho,imagen->alto);
    fprintf(sal,"%d\n",imagen->grises)
        ;
    for(i=0; i<imagen->alto;i++)
```

```
        for(j=0; j<imagen->ancho; j++)
        {
            c=imagen->im[i][j];
            fputc(c,sal);
            //printf(" ",imagen->im[i
                ][j]);
        }
        //printf("");
//  fputc(final,sal);
    fclose(sal);
}

void aclara_imagen(IMAGEN *imagen, int
     valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]+
                valor) > 255)
                imagen->im[i][j]=255;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] +
                    valor;
}

void oscurece_imagen(IMAGEN *imagen,
    int valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]-
                valor) <0)
                imagen->im[i][j]=0;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] -
                    valor;
}

void negativo_imagen(IMAGEN *imagen){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            imagen->im[i][j]=255 -
                imagen->im[i][j];
}

void binariza_imagen(IMAGEN *imagen,
    int umbral){
    int i,j;
   // printf("-------lectura de imagen
        ---------------");
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if (imagen->im[i][j] <
                umbral){
                imagen->im[i][j]=0;
               // printf("0");
                 }
               else{
                imagen->im[i][j]= 255;
                 //    printf("1");
                 }
}
void imprime_imagen(IMAGEN *imagen,
    int umbral){
    int i,j;
    printf("-------lectura de imagen
        ---------------");
    cout<<endl;
    for(i=0; i < imagen->alto; i++){
     printf(" ");
        for(j=0; j < imagen->ancho; j
            ++){
            if (imagen->im[i][j] <
                umbral)

                printf("1");
               else
                printf("0");}
            cout<<endl;
    }

}
void cuenta_ceros(IMAGEN *imagen, int
    umbral){
     int i,j;
     int p;
     int contador=0;
     float perimetro;
     float restante;
     float final;
     int x=imagen->alto;
     int y=imagen->ancho;
     int datos[x][y];
     int datos1[x][y];
    printf("-------lectura de imagen
        ---------------");
    //cout<<endl;
    for(i=0; i < x ; i++){
       for(j=0; j < y; j++)
            if (imagen->im[i][j] <
                umbral)
                datos[i][j]=1;
            else
                datos[i][j]=0;
        }

    //cuenta unos horizontal
    p=0;
    for(i=0; i < x ; i++){
        for(j=0; j < y; j++){
           if(p != datos[i][j])
               datos1[i][j]=7;
           else
              datos1[i][j]=0;
          p=datos[i][j];
        }
     }
    //cuenta unos vertical
    for(j=0; j < x ; j++){
        for(i=0; i < y ; i++){
            if(p != datos[i][j])
            datos1[i][j]=7;
           else{
              if (datos1[i][j]!=7)
                  datos1[i][j]=0;}
             p=datos[i][j];
        }
        //cout<<endl;
       }
//Cuenta total
```

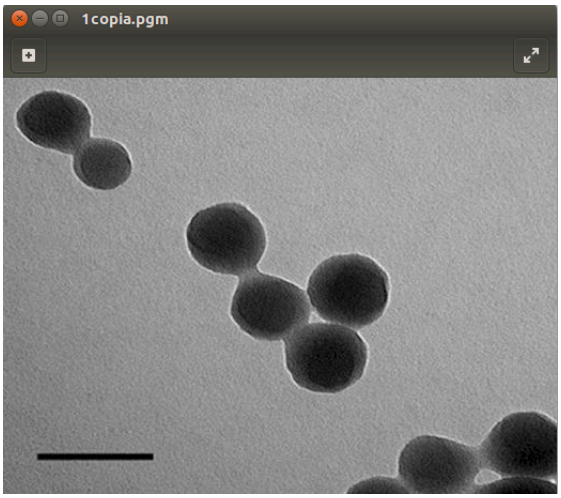
```
    contador=0;
      for(i=0; i < x ; i++)
          for(j=0; j < y ; j++)
              if( datos1[i][j]==7)
                contador++;



    //imprime ceros
    for(i=0; i < x; i++){
        printf(" \n");
       for(j=0; j < y; j++)
            printf("%i ",datos[i][j]);

}
   restante= (contador-1)/4;
   perimetro= contador-restante;
   final=perimetro/10;

    cout<<endl<<"El perimetro de cada
        particula en nanometros es: ";
     printf("%.2f",final); cout<<"nm."
         ;



}

void histograma(IMAGEN *imagen){
    int i,j;
    int mayor;
    float aux;
    int histo[256];
    for (i=0; i< imagen->alto; i++)
        for(j=0; j< imagen->ancho; j
            ++)
            histo[imagen->im[i][j]]=
                histo[imagen->im[i][j
                ]]+1;

    mayor = 0;
    for(i=0; i<256; i++)
       if (histo[i]>mayor)
            mayor = histo[i];

    for (i=0; i<256; i++){
        aux=(float)(histo[i])/mayor;
        aux=aux*100*2;
        histo[i] = (int)(aux);
    }

}

IMAGEN laplaciano(IMAGEN *imagen){
    IMAGEN ret;
    int numero;
    char nro[3];

    int m[3][3],i,j,k,l,s;
    m[0][0]= 0; m[0][1]=-1; m[0][2]=
        0;
    m[1][0]=-1; m[1][1]=-4; m
        [1][2]=-1;
    m[2][0]= 0; m[2][1]=-1; m[2][2]=
        0;

    for(i=1; i< imagen->alto-2; i++){
        for(j=1; j< imagen->ancho-2; j
            ++){
                s=0;
                for(k=0; k<3; k++)
                    for(l=0; l<3; l++)
                        s=s+m[k][l]*(
                            imagen->im[i+k
                            ][j+l]);
            }
            ret.im[i][j]= s;
        }

        strcpy(ret.nombre,"Lp_");
        strcat(ret.nombre,imagen->nombre);
        ret.cant_comen = imagen->
            cant_comen;
        ret.ancho      = imagen->ancho;
        ret.alto       = imagen->alto;
        ret.grises     = imagen->grises;
        return ret;
    }

    IMAGEN duplica_imagen(IMAGEN *imagen){
        IMAGEN ret;
        int i,j;
        printf("\nDuplicando: %s\n",imagen
            ->nombre);
        strcpy(ret.nombre,"Bi_");
        strcat(ret.nombre,imagen->nombre);
        ret.cant_comen = imagen->
            cant_comen;
        ret.ancho      = imagen->ancho;
        ret.alto       = imagen->alto;
        ret.grises     = imagen->grises;
        for(i=0; i<imagen->alto; i++)
            for(j=0; j<imagen->ancho; j++)
                {
                ret.im[i][j]=imagen->im[i
                    ][j];
                //printf(" xxxx ",ret.im[i
                    ][j]);
            }
        return ret;
    }

    #endif // PGMBASICA_H
```

B. *main.cpp*

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
#include "pgmbasica.h"

using namespace std;

int main(int argc, char *argv[]){

    IMAGEN ima;
    IMAGEN otra;
    char nombre[20];
    int tipo;

    cout<<"Ingresa el nombre de la
        imagen"<<endl;
    cin>>nombre;

    tipo = es_pgm(nombre);

    if(tipo == 1)
```

```
    {
        ima = lee_imagen(nombre);
                cout<<endl;
                muestra_datos_imagen(&
                    ima);
                cout<<endl;
                otra = duplica_imagen
                    (&ima);
                cout<<endl;
                muestra_datos_imagen(&
                    otra);
                cout<<endl;
                binariza_imagen(&otra,
                     128);
                cout<<endl;
                cuenta_ceros(&otra,
                    128);
                cout<<endl;
                //imprime_imagen(&otra
                    , 128);
                cout<<endl;
                // recubrimiento(&otra
                    );
                cout<<endl;
                grabaimagen(&otra);
                cout<<endl;
    }
    cout<<endl;
}
```

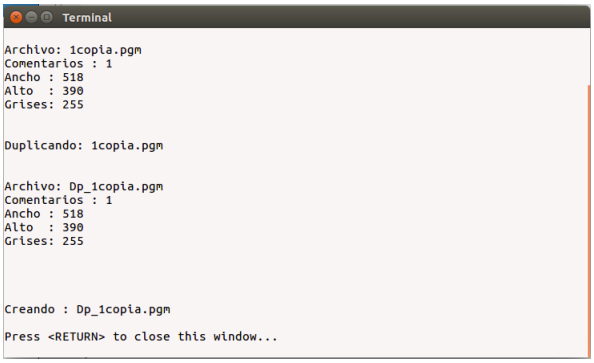
C. *Debug*



(c) Eliges la imagen que vas a transformar con la binarización/umbralización que esté en el formato ".pgm"

(d) Se le pide al usuario ingresar el nombre de la imagen a transformar, en este caso se llama "1copia.pgm"

(e) Se ingresa el nombre completo de la imagen junto con su formato pgm



(f) El programa corre y se leen los datos o características del archivo junto con el nombre del duplicado transformado a través del código.

(g) Se va a la carpeta del código en Qt y se observa el archivo duplicado, se abre y se observa la binarización de blanco y negro en las imágenes.

D. *Observaciones del código de binarización*

En nuestro código pusimos como parámetro en la umbral el número 128 (como se puede apreciar en la parte del main.cpp) decidimos tomar este parámetro tomado a partir de la literatura que leímos.

XII. Códigos de la interfaz

A. *mainwindow.cpp*

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include"algoritmos.h"
#include"contornocv.h"

// CONSTRUCTOR.
MainWindow::MainWindow(QWidget *parent
    ) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
//DESTRUCTOR.
MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked
    ()
{
      QString filename = QFileDialog::
          getOpenFileName(this,tr("
          Choose"),"",tr("Imagenes (*.
          pgm)"));
      if(QString::compare(filename,
          QString()) !=0)
      {
          QImage image;
          bool valid = image.load(
              filename);

          if(valid)
          {
              ui->label_3->setPixmap(
                  QPixmap::fromImage(
                  image));
          }
          else
          {
              cout<<"Error de imagen,
                  favor de subir en
                  formato .pgm"<<endl;
          }
      }
   }

void MainWindow::
    on_pushButton_2_clicked()
{
    IMAGEN ima;
    IMAGEN otra;
    char nombre[20]="recubrimiento.pgm
        ";
    int tipo;
    tipo = es_pgm(nombre);

    if(tipo == 1)
    {
        ima = lee_imagen(nombre);
        cout<<endl;
        otra = duplica_imagen(&ima);
        cout<<endl;
        binariza_imagen(&otra, 128);
        cout<<endl;
        // recubrimiento(&otra);
        cout<<endl;
        grabaimagen(&otra);
        cout<<endl;
    }
    cout<<endl;
}
void MainWindow::
    on_pushButton_3_clicked()
{
 QPixmap pix("/home/citlali/Imagenes/
     Bi_recubrimiento.pgm");
 ui->label_4->setPixmap(pix);
}
```

B. *mainwindow.h*

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QtCore/QBuffer>
```

```
#include<QFileDialog>
#include<QLabel>
#include<QFileDialog>
#include<QPushButton>
#include <QMainWindow>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *
        parent = 0);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();
void on_pushButton_2_clicked();
    void on_pushButton_3_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

C. *algoritmos.h*

```
#ifndef ALGORITMOS_H
#define ALGORITMOS_H
#include <iostream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
using namespace std;


typedef struct imagen{
    char nombre[20];
    int cant_comen;
    int ancho;
    int alto;
    int grises;
    unsigned char im[800][800];
} IMAGEN;

int es_pgm(char *nombre){
    FILE *ent;
    unsigned char c1,c2;
    printf("\nChequeando: %s\n",nombre
        );
    if ((ent = fopen(nombre,"rb")) ==
        NULL)
        return 0;
    c1=fgetc(ent);
    c2=fgetc(ent);

    fclose(ent);
    if ((c1=='P') && (c2=='5'))
        return 1;
    else
        return 0;
}
```

```
IMAGEN lee_imagen(char *nombre)
{
    IMAGEN imagen;
    unsigned int c1,c2;
    unsigned char carac;
    FILE *ent;
    int i,j, ancho,alto,grises;

    ent = fopen(nombre,"rb");
    c1=fgetc(ent);
    c2=fgetc(ent);
    fgetc(ent); //Elimina el separador
    carac=fgetc(ent);
    imagen.cant_comen=0;
    while(carac=='#'){
        imagen.cant_comen=imagen.
            cant_comen+1;
        carac=fgetc(ent);
        while(carac!='\n')
            carac=fgetc(ent);
        carac=fgetc(ent);
    }
    ungetc(carac,ent);
    fscanf(ent,"%d%d",&ancho,&alto);
    fscanf(ent,"%d",&grises);
    imagen.ancho = ancho;
    imagen.alto= alto;
    imagen.grises = grises;
    strcpy(imagen.nombre,nombre);
    for(i=0; i<imagen.alto;i++)
        for(j=0; j<imagen.ancho; j++){
            carac=fgetc(ent);
            imagen.im[i][j]=carac;
        }
    fclose(ent);
    return(imagen);
}

void muestra_datos_imagen(IMAGEN *
    imagen){
    printf("\nArchivo: %s",imagen->
        nombre);
    printf("\nComentarios : %d",imagen
        ->cant_comen);
    printf("\nAncho : %d",imagen->
        ancho);
    printf("\nAlto  : %d",imagen->alto
        );
    printf("\nGrises: %d\n",imagen->
        grises);
}

void grabaimagen(IMAGEN *imagen){
    int i,j;
    unsigned char c;
    FILE *sal;

/*
    int nro;
    char nombre[14],nombre1[6];
    nro = (rand()%(9999 - 1000 + 1)) +
         1000;
    itoa(nro,nombre1,10);
    strcpy(nombre,"Hac_");
    strcat(nombre,nombre1);
    strcat(nombre,".pgm");
*/

    printf("\nCreando : %s",imagen->
```

```
        nombre);
    sal=fopen(imagen->nombre,"wb");
    fprintf(sal,"P5\n");
    fprintf(sal,"# Creado por C.\n");
    fprintf(sal,"%d %d\n",imagen->
        ancho,imagen->alto);
    fprintf(sal,"%d\n",imagen->grises)
        ;
    for(i=0; i<imagen->alto;i++)
        for(j=0; j<imagen->ancho; j++)
        {
            c=imagen->im[i][j];
            fputc(c,sal);
        }
//  fputc(final,sal);
    fclose(sal);
}

void aclara_imagen(IMAGEN *imagen, int
     valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]+
                valor) > 255)
                imagen->im[i][j]=255;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] +
                    valor;
}

void oscurece_imagen(IMAGEN *imagen,
    int valor){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if ((imagen->im[i][j]-
                valor) <0)
                imagen->im[i][j]=0;
            else
                imagen->im[i][j]=
                    imagen->im[i][j] -
                    valor;
}

void negativo_imagen(IMAGEN *imagen){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            imagen->im[i][j]=255 -
                imagen->im[i][j];
}

void binariza_imagen(IMAGEN *imagen,
    int umbral){
    int i,j;
    for(i=0; i < imagen->alto; i++)
        for(j=0; j < imagen->ancho; j
            ++)
            if (imagen->im[i][j] <
                umbral)
                imagen->im[i][j]=0;
            else
                imagen->im[i][j]= 255;
}

void histograma(IMAGEN *imagen){
    int i,j;
    int mayor;
    float aux;
    int histo[256];
    for (i=0; i< imagen->alto; i++)
        for(j=0; j< imagen->ancho; j
            ++)
            histo[imagen->im[i][j]]=
                histo[imagen->im[i][j
                ]]+1;

    mayor = 0;
    for(i=0; i<256; i++)
       if (histo[i]>mayor)
            mayor = histo[i];

    for (i=0; i<256; i++){
        aux=(float)(histo[i])/mayor;
        aux=aux*100*2;
        histo[i] = (int)(aux);
    }

}

IMAGEN laplaciano(IMAGEN *imagen){
    IMAGEN ret;
    int numero;
    char nro[3];

    int m[3][3],i,j,k,l,s;
    m[0][0]= 0; m[0][1]=-1; m[0][2]=
        0;
    m[1][0]=-1; m[1][1]=-4; m
        [1][2]=-1;
    m[2][0]= 0; m[2][1]=-1; m[2][2]=
        0;

    for(i=1; i< imagen->alto-2; i++){
        for(j=1; j< imagen->ancho-2; j
            ++){
            s=0;
            for(k=0; k<3; k++)
                for(l=0; l<3; l++)
                    s=s+m[k][l]*(
                        imagen->im[i+k
                        ][j+l]);
        }
        ret.im[i][j]= s;
    }

    strcpy(ret.nombre,"Lp_");
    strcat(ret.nombre,imagen->nombre);
    ret.cant_comen = imagen->
        cant_comen;
    ret.ancho      = imagen->ancho;
    ret.alto       = imagen->alto;
    ret.grises     = imagen->grises;
    return ret;
}

IMAGEN duplica_imagen(IMAGEN *imagen){
    IMAGEN ret;
    int i,j;
    printf("\nDuplicando: %s\n",imagen
        ->nombre);
    strcpy(ret.nombre,"Bi_");
    strcat(ret.nombre,imagen->nombre);
```

```
    ret.cant_comen = imagen->
        cant_comen;
    ret.ancho      = imagen->ancho;
    ret.alto       = imagen->alto;
    ret.grises     = imagen->grises;
    for(i=0; i<imagen->alto; i++)
        for(j=0; j<imagen->ancho; j++)
            ret.im[i][j]=imagen->im[i
                ][j];
    return ret;
}


#endif // ALGORITMOS_H
```

D. contornocv.h

```
#ifndef CONTORNOCV_H
#define CONTORNOCV_H

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include<iostream>
#include <cstdio>
#include <cstdlib>
#include <stdio.h>
#include <string>
#include <string.h>
#include <time.h>
#include<QDebug>
using namespace cv;
using namespace std;

Mat Contorno();
int muestra();
#endif // CONTORNOCV_H
```

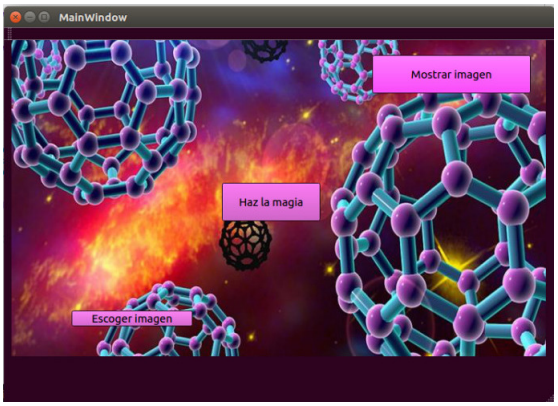
E. contornocv.cpp

```
#include "contornocv.h"
// MOSTRAR IMAGEN EN CONTORNO

cv::Mat Contorno(){
    cv::Mat image= cv::imread("/home/
        citlali/Proyecto_progra/
        Bi_recubrimiento.pgm");
    cv::Mat contours;
    cv::Mat gray_image;

    cvtColor( image, gray_image,
        CV_RGB2GRAY );

    cv::Canny(image,contours,10,350);
    cv::namedWindow("Contorno");
    cv::imshow("Contorno",contours);
    cv::waitKey(0);

    Mat imagen;     // Matriz que
        guardara la imagen
    imagen = contours;
return contours;
}

int muestra(){

    FILE *fichero;
    int *matriz;
    int c,i; // note: int, not char,
        required to handle EOF

    char nombre[20]="Dp_oro.pgm";
    fichero = fopen(nombre, "r" );

    //FILE* fp = std::fopen("prueba.
        txt", "r");
    if(!fichero) {
        std::perror("fallo al abrir");
        return EXIT_FAILURE;
    }

    matriz=&c;
    cout<<&matriz<<endl;
    i=0;
    while ((c = std::fgetc(fichero))
        != EOF) { // standard C I/O
        file reading loop
       //std::putchar(c);
        cout<<*(matriz)<<endl;

    }

    if (std::ferror(fichero))
        std::puts("I/O error al leer")
            ;

    else if (std::feof(fichero))
        std::puts("fin satisfactorio")
            ;

    std::fclose(fichero);
     return *(matriz);
}
```

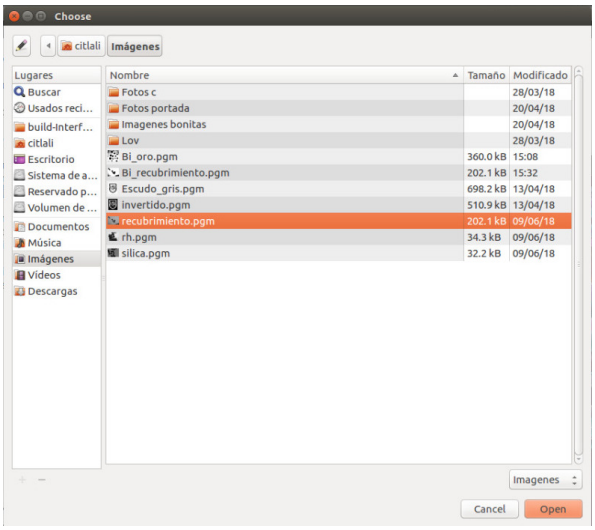
F. main1.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

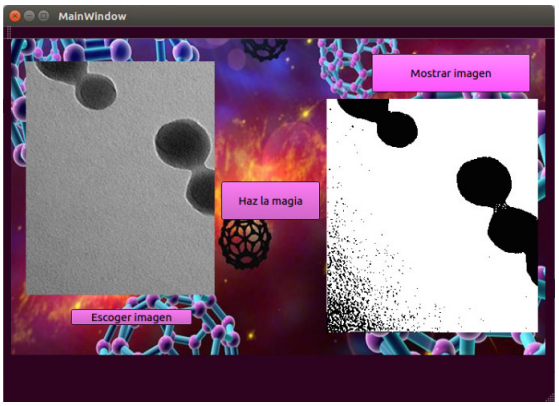
G. Debug



(h) Se abre la interfaz con sus respectivos botones que le piden al usuario elegir la opción que desea, en este caso se da clic en el botón de "cargar imagen".



(i) Se abre una ventana donde te pide elegir la imagen de alguna de las carpetas de tu escritorio, como vimos anteriormente la imagen debe estar en formato pgm para que pueda funcionar.



(j) Se muestran las 2 imágenes, la que se le hizo el tratamiento del binarizado y la original.

XIII. Observaciones

Al realizar el proyecto nos encontramos con diversas dificultades, tanto identificar las funciones y colocarlas en clases (.h y su respectivo .cpp).

Otro problema que tuvimos fue al intentar cargar la imagen y transformarla en la interfaz, no podiamos relacionar el botón con la imagen ya modificada.

XIV. Análisis de resultados

Para resolver algunos problemas nos apoyamos con algunos videos como el de insertar la imagen en la interfaz.
El proyecto identifica las partículas resaltando su color a negro, basándose en un programa orientado a objetos, donde incluimos varios conceptos de lo aprendido en clase (puteros, ciclos for, condicionales como if y while, entre otros) y aplicamos otras como los contructores y destructores.

XV. Conclusión

En este proyecto utilizamos algunas cosas de lo visto durante todo el semestre en la clase de Programación, algunas cosas eran básicas como por ejemplo para hacer la binarización de las imágenes en C++ se utilizaron ciclos for con arreglos que leen la imagen o la analizan a través de matrices, también el de declarar funciones y saberlas usar, como lo son las funciones *int* que retornan valores a diferencia de las funciones *void* que no retornan la mayor parte del proyecto fueron cosas más avanzadas, ya que eran cosas sobre la programación orientada a objetos en las cuales tuvimos que dividir los códigos en clases las cuales se dividían en sus respectivos .cpp y el .h, también el de saber utilizar los objetos (o widgets) a través de la plataforma de QtCreator ya que debemos saber qué objetos conectar con alguna acción, también utilizamos lo que son los constructores y destructores de datos que se utilizan básicamente para la memoria de la computadora, éstos operadores son muy importantes porque gracias a ellos la información del programa se almacena en una parte específica de la memoria (que por cierto se trata de una memoria dinámica pues se hace el uso de apuntadores) y para que la computadora no vaya almacenando tantas cosas al estar usándola se debe de tener un destructor que borre todos los datos que se utilizaron al correr el programa.
Lamentablemente no pudimos terminar el proyecto al 100% como se tenía planeado pues no tuvimos las bases necesarias para saber poner nuestras ideas en códigos, sin embargo pudimos realizar una parte importante del proyecto que fue la de la segmentación de imágenes que como explicamos en la parte del marco teórico es la parte más primordial del análisis de imágenes y esa parte por lo menos sí pudimos realizar.

Sin embargo al haber realizado este proyecto nos hemos dado cuenta del impacto que tiene la programación sobre lo que vivimos y utilizamos día a día, al haber aprendido bastantes cosas en la clase de programación nos hemos dado cuenta que éstos saberes los podemos utilizar más adelante en nuestra carrera y que han servido de gran apoyo para proyectos futuros dentro o fuera de la universidad.

References

[1] *Ahuactzin, J., E. Talbi, P. Bessiere, y E. Mazer (1992). Using genetic algorithms for robot motion planning. En European Conference on Artificial Intelligence (ECAI92).*

[2] *N. Bergman. Recursive Bayesian Estimation: Navigation and Tracking Applications. Dissertation nr. 579, Linkoping University,Sweden, 1999.*

[3] *Javier González Jiménez. Visión por Computador. Ed. Paraninfo. 2000.*

[4] *Rafael C. González, Richard E. Woods. Tratamiento digital de imágenes, Ed. Pratt W.K. 1991.*

[5] *Digital Image Processing, Ed.Pratt W.K 1991.*

[6] *Chaparro Laso, Eva. Tracking automático de objetos en secuencias de imágenes usando Filtro de Partículas. s/f*

[7] *Fernando Otero, Exequiel Soulé, Gloria Frontini, Guillermo E. Eliçabe.Estimación de la distribuición de tamaños de partículas poliméricas embebidas en una matriz de polímeros.Instituto Nacional de Ciencia y Tecnología de los Materiales (INTEMA), Universidad Nacional de Mar del Plata (UNMdP) y Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) Mar del Plata, Argentina.*

[8] *Renio. Royal Society of Chemistry.ohn Emsley, Natures Building Blocks: An A-Z Guide to the Elements, Oxford University Press, New York, 2nd Edition, 2011.*
Royal Society of Chemistry

[9] *Boro. Royal Society of Chemistry. John Emsley, Natures Building Blocks: An A-Z Guide to the Elements, Oxford University Press, New York, 2nd Edition, 2011.*
Royal Society of Chemistry

[10] *Qin HU, Lin LIU, Xin-bao ZHAO, Si-feng GAO, Jun ZHANG1, Heng-zhi FU.Effect of carbon and boron additions on segregation behavior of directionally solidified nickel-base superalloys with rhenium. 24 June 2013*

[11] *Effect of rhenium content on microstructures and mechanical properties of electron beam welded TZM alloy joints. March 2018.*

[12] *Angarita Gutiérrez, Luis Guillermo. Síntesis de películas delgadas por la técnica de magnetrón sputtering a partir de blancos de Renio y Boro.2017*

[13] *Carlos Gustavo Rodriguez Medina, Oscar Daniel Chuk, Regina Bertero, Pablo Trigo. Medición del Tamaño de Partículas de Minerales Mediante Procesamiento Digital de Imágenes. s/f*

[14] *David Grau López. Desarrollo de un software para la edición de imágenes digitales. 2013*

Link a la plantilla Overleaf