Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Reporte final de Proyecto Terminal:

"Sistema calificador de exámenes"

Alumnos:

Cárdenas Díaz González Hugo Ricardo 208368488

Hernández Nieto Jorge Alberto 208332479

Trimestre: 13-P.

Julio 2013.

Asesor: Dr. Risto Fermín Rangel Kuoppa.

Número económico: 27499.

Resumen

El presente documento habla de cómo desarrollar un sistema que sea capaz de revisar una hoja de respuestas de opción múltiple, cargarla y después proceder a revisar si las opciones son correctas y al último proporcionar el resultado de la evaluación de la hoja de respuestas que se obtuvo en un principio.

Objetivo general.

Construir un programa capaz de realizar la revisión y la evaluación de una hoja de respuestas de examen generado por el software “Sigexabloom” de un estudiante por medio de un escáner, y que muestre la calificación de la evaluación.

Objetivos específicos.

1. Desarrollar e implementar un módulo para filtrar la imagen digitalizada de la hoja de respuestas con un filtro Gaussiano.

2. Desarrollar e implementar un módulo para la aplicación del algoritmo que binarize la imagen digitalizada, una imagen sin ruido (filtrada).

3. Desarrollar e implementar un módulo para la aplicación del algoritmo que adelgace a la imagen.

4. Desarrollar e implementar un módulo para homologar las regiones de las imágenes generadas por el algoritmo “image skeletonization” con algoritmos de inundación o crecimiento.

5. Desarrollar e implementar un módulo para identificar las zonas de cruces verticales y horizontales

6. Desarrollar e implementar un módulo para validar cada opción de la hoja de respuestas como llenada o no.

7. Desarrollar e implementar un módulo para calcular la calificación del examen según el alumno haya llenado la hoja de respuestas.

Introducción

Actualmente hemos notado los problemas que tienen los profesores al estar calificando los exámenes uno por uno, lo cual provoca mucha pérdida de tiempo, ya que en cada grupo se registran aproximadamente entre 40 y 50 alumnos por trimestre la cual provoca una falta de manejo eficiente de la calificación de los exámenes, ya que el profesor está expuesto a cometer errores, y hace que los resultados se entreguen mucho después del día que se había aplicado el examen.

Justificación

La tecnología ha venido evolucionando de manera constante desde hace mucho tiempo, esto se debe a que el ser humano ha buscado muchas formas para hacer más fácil su vida, es por eso que aprovechando la tecnología actual, intentaremos hacer más fácil la forma en que los profesores califican los exámenes de opción múltiple.

Se ahorrara tiempo y se hará más fácil el proceso de evaluación, además contribuiremos de manera significativa a las expectativas de las nuevas exigencias del modelo educativo actual, que tanto aboga por la inserción de los actores del hecho educativo en la sociedad de la información apoyada básicamente en el empleo de las TIC´s.

Desarrollo del proyecto

Nuestro objetivo es obtener la calificación de un examen, específicamente de opción múltiple, con la mínima interacción del profesor.

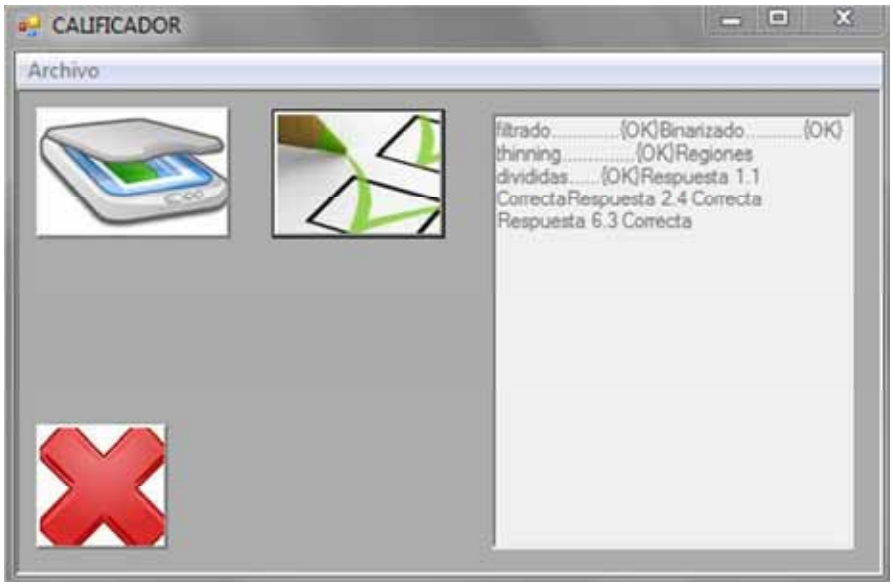
Para llevar a cabo esto, se planteó crear un software que digitalice una hoja de respuestas de un examen, que haga un chequeo a cada una de las preguntas junto con su respectiva respuesta, si la respuesta está mal seleccionada, la contará como incorrecta, en caso de que estén seleccionadas 2 opciones, también la contará como mala, y si la respuesta es correcta la contará como buena.

Esto conlleva a que se ahorre una gran cantidad de tiempo por parte del profesor a la hora de dar las calificaciones de los exámenes, y el alumno sabrá al momento cuál fué su calificación de su examen, además se tratará de que el software sea intuitivo tal que no se necesite mucho conocimiento de lenguajes de programación.

Involucrando cada uno de los objetivos particulares, se obtuvo un resultado satisfactorio del proyecto. A continuación se describe cada uno de ellos, haciendo hincapié en las fortalezas y debilidades que se encontraron en la paulatina realización de estos, tomando en cuenta que sin el conjunto y plena función de ellos este proyecto terminal no hubiese cumplido su fin.

Módulo 1:

En este módulo se desarrollo una interfaz y mediante la librería Twain, se desarrolló una aplicación que se encarga de digitalizar la imagen y guardarla.



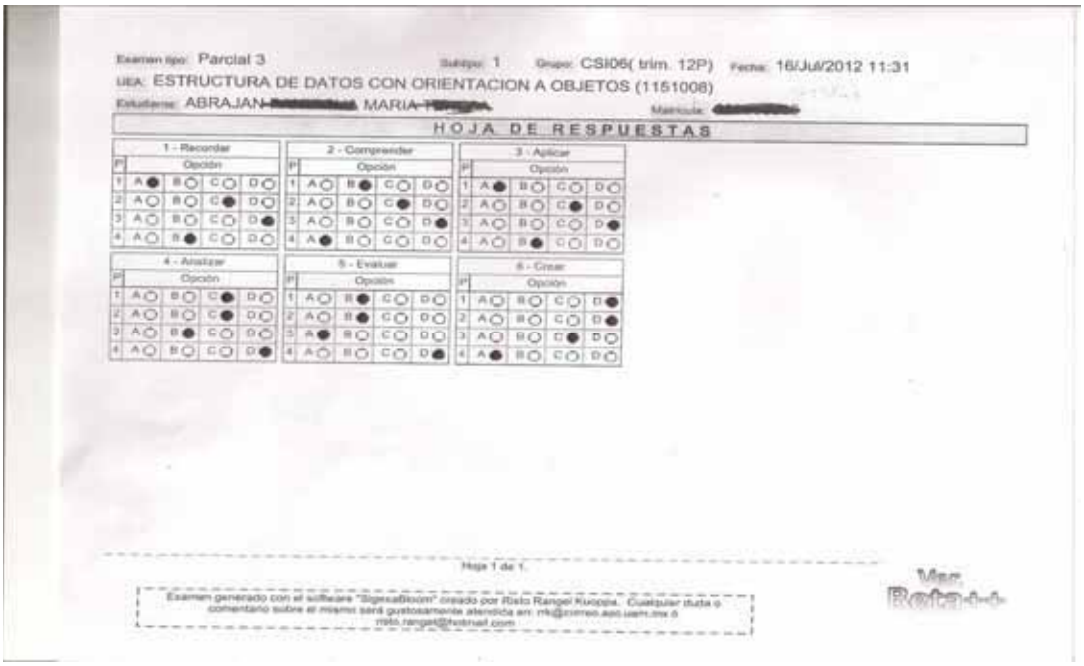
. Interfaz de Twain

Módulo 2

En este módulo se carga la imagen que anteriormente fue escaneada y guardada de la hoja de respuestas, después se le aplica el algoritmo de filtro gaussiano a la imagen para borrar los puntos negros que se introdujeron a la imagen al momento de escanearla o al momento de que los alumnos estaban resolviendo el examen, ya que causan ruido en la imagen.

El módulo cuenta con una sola clase, en donde se encuentra el algoritmo de filtro gaussiano y se pasan como parámetros el valor de la máscara y de la sigma para realzar el filtro gaussiano.

El proceso del filtro gaussiano es rápido, eficiente y arroja buenos resultados para proceder a realizar los siguientes módulos.



. Hoja con el filtro gaussiano

Módulo 3:

En este módulo se carga la imagen obtenida después de aplicar el módulo de filtro gaussiano y se le quitó el ruido a la imagen. Este módulo consiste en pasar la imagen a blanco y negro, eliminando todas las tonalidades grises en la imagen para poder realizar la calificación de la hoja de respuestas.

Primeramente se obtiene el color de cada pixel, y mediante una operación matemática obtenemos un valor, dependiendo de ese valor se compara con un umbral con valor de 150, si el color obtenido por el pixel es menor o igual al umbral, ese pixel se pone en blanco, en caso contrario se coloca el pixel en color negro.

Examen tipo: Parcial 3 Subtipo: 1 Grupo: CSI06(trim. 12P) Fecha: 16/Jul/2012 11:31

UEA: ESTRUCTURA DE DATOS CON ORIENTACION A OBJETOS (1151008)

Estudiante: ABRAJAN[redacted] MARIA[redacted] Matrícula: [redacted]

HOJA DE RESPUESTAS

1 - Recordar				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

2 - Comprender				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

3 - Aplicar				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

4 - Analizar				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

5 - Evaluar				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

6 - Crear				
P	Opción			
1	A○	B○	C○	D○
2	A○	B○	C○	D○
3	A○	B○	C○	D○
4	A○	B○	C○	D○

3. Hoja binarizada

El proceso de binarizar la imagen es rápido y no hace que se pierda información al momento de realizar este proceso, y los resultados obtenidos son satisfactorios para proceder a los siguientes módulos.

Módulo 4:

En este módulo se carga la imagen que anteriormente se obtuvo en el módulo de binarizar. El módulo consiste en hacer un esqueleto o “thinning” de la imagen, es decir, las líneas de la imagen se reducirán hasta tener un ancho de 1 pixel.

Al hacer esto eliminamos información que no nos es útil, el módulo funciona sin importar que la figura sea una forma uniforme o no, las líneas siempre están unidas y no se producen roturas en las líneas y en sus intersecciones.

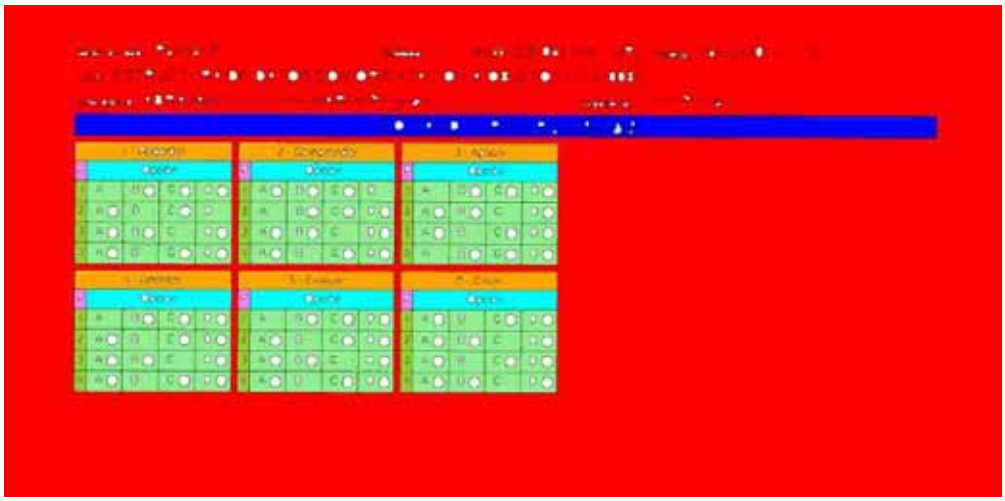
4. Hoja con Thinning

El proceso arroja resultados muy satisfactorios, no se rompen las líneas y las adelgaza, el único problema que se encontró en este módulo, es que se tarda demasiado en realizar el proceso de esqueleto de la imagen.

Módulo 5:

En este módulo se carga la imagen ya con la aplicación del algoritmo “thinning”. El módulo consiste en colorear las regiones de la hoja de respuestas, cada región de un color para hacer más fácil la evaluación de la hoja de respuestas.

Originalmente se planteó hacer este módulo mediante el algoritmo de Lloyd, pero después de una investigación descubrimos que si usamos el algoritmo llamado “Flood Fill”, reduciríamos el tiempo de ejecución del programa, y como el módulo anterior es demasiado tardado, decidimos utilizar este para mejorar el tiempo de procesamiento.



5. Imagen con inundación

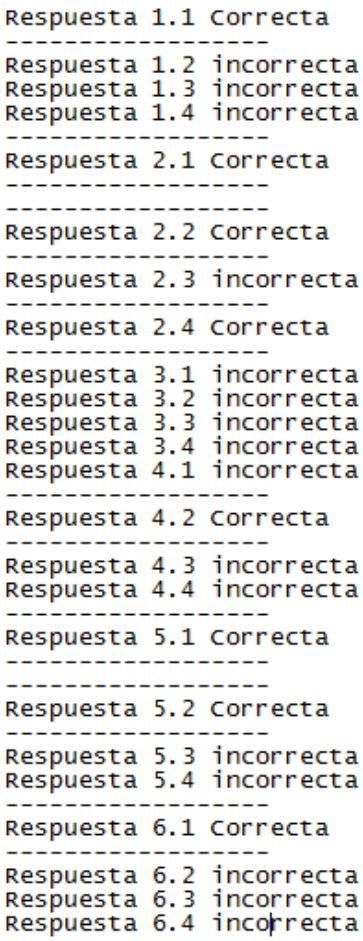
El resultado de este proceso es satisfactorio, ya que rellena cada región completamente de un color sin dejar pixeles en blanco, el proceso es rápido y arroja los resultados y permite pasar al siguiente proceso.

Módulo 6:

Este módulo se omitió, ya que al momento de ejecutar el módulo 4, se identificaron de forma muy clara los cruces verticales y horizontales de la hoja de respuestas.

Módulo 7 y Módulo 8:

Estos dos módulos se implementaron en uno solo, ya que automáticamente al momento de ir validando cada opción de cada respuesta, se puede decir si la respuesta es correcta y dar el resultado del examen.



6. Resultados

El proceso es rápido y se checan todas las opciones de todas las preguntas para verificar que no se llenaron dos opciones y si solo esta una opción llenada, que sea la correcta.

Conclusiones.

Podemos concluir que en base a los módulos creados, hemos logrado cumplir el objetivo general y los objetivos específicos del proyecto, ya que cada módulo se realizó satisfactoriamente con resultados aceptables.

Además al momento de trabajar con imágenes logramos expandir nuestros horizontes en base a programación, ya que no habíamos tenido oportunidad de utilizar el lenguaje de programación utilizado en este proyecto. Las imágenes y su

procesamiento mediante la computadora, representan un campo muy grande, el cual no tiene un campo de concentración definido, y puede ser utilizado en muchos estudios.

Los algoritmos utilizados en este proyecto también tienen aplicaciones en más campos, éstos algoritmos representan una gran ventaja para la elaboración de estudios para determinar regiones cancerosas, o también para la animación de películas en el caso del algoritmo de thinning.

Bibliografía

Digital image processing, Rafael C. Gonzalez, Richard E. Woods. .

Image processing in Java, Douglas A. Lyon.

Cómo programar C#, Harvey M. Deitel, Paul J. Deitel.

Procesamiento y análisis de imágenes, Rodríguez, Sossa.

msdn.microsoft.com

http://www.tecnohobby.net/ppal/index.php?option=com_content&view=article&id=21:zhangsuen&catid=41:pitopicosgenerales&Itemid=21

http://lodev.org/cgtutor/floodfill.html

http://docs.gimp.org/es/plug-in-gauss.html

http://www.twain.org/

Apéndice A: Códigos

Código del Módulo 1

Para este módulo, utilizamos una API, la cual fue elaborada para Twain, y se usa para conectar dispositivos de digitalización de imágenes, esto nos dio una gran ventaja al momento de comprender el funcionamiento de esta útil librería. Solo colocaremos el código que es útil para nosotros, ya que twain tiene aún más código para la digitalización de imágenes.

```
public calificador()
        {
            InitializeComponent();
```

```
            tw = new Twain();
            tw.Init(this.Handle);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                tw.Finish();
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose(disposing);
        }
private void menuItemScan_Click(object sender, System.EventArgs e)
        {
            if (!msgfilter)
            {
                this.Enabled = false;
                msgfilter = true;
                Application.AddMessageFilter(this);
            }
            tw.Acquire();
        }

        bool IMessageFilter.PreFilterMessage(ref Message m)
        {
            TwainCommand cmd = tw.PassMessage(ref m);
            if (cmd == TwainCommand.Not)
                return false;

            switch (cmd)
            {
                case TwainCommand.CloseRequest:
                    {
                        EndingScan();
                        tw.CloseSrc();
                        break;
                    }
                case TwainCommand.CloseOk:
                    {
                        EndingScan();
                        tw.CloseSrc();
                        break;
                    }
                case TwainCommand.DeviceEvent:
                    {
                        break;
                    }
                case TwainCommand.TransferReady:
                    {
                        ArrayList pics = tw.TransferPictures();
                        EndingScan();
                        tw.CloseSrc();
                        picnumber++;
```

```
                        for (int i = 0; i < pics.Count; i++)
                        {
                            IntPtr img = (IntPtr)pics[i];
                            PicForm newpic = new PicForm(img);
                            newpic.MdiParent = this;
                            int picnum = i + 1;
                            newpic.Text = "ScanPass" + picnumber.ToString() + "_Pic"
+ picnum.ToString();
                            newpic.Show();
                        }
                        break;
                    }
            }

            return true;
        }

        private void EndingScan()
        {
            if (msgfilter)
            {
                Application.RemoveMessageFilter(this);
                msgfilter = false;
                this.Enabled = true;
                this.Activate();
            }
        }
private void Escanear_Click(object sender, EventArgs e)
        {
            if (!msgfilter)
            {
                this.Enabled = false;
                msgfilter = true;
                Application.AddMessageFilter(this);
            }
            tw.Acquire();
        }
```

Funciónes de Twain.

```
public void Acquire()
        {
            TwRC rc;
            CloseSrc();
            if (appid.Id == IntPtr.Zero)
            {
                Init(hwnd);
                if (appid.Id == IntPtr.Zero)
                    return;
            }
            rc = DSMident(appid, IntPtr.Zero, TwDG.Control, TwDAT.Identity,
TwMSG.OpenDS, srcds);
            if (rc != TwRC.Success)
                return;

            TwCapability cap = new TwCapability(TwCap.XferCount, 1);
```

```
            rc = DScap(appid, srcds, TwDG.Control, TwDAT.Capability, TwMSG.Set,
cap);
            if (rc != TwRC.Success)
            {
                CloseSrc();
                return;
            }
public TwainCommand PassMessage(ref Message m)
        {
            if (srcds.Id == IntPtr.Zero)
                return TwainCommand.Not;

            int pos = GetMessagePos();

            winmsg.hwnd = m.HWnd;
            winmsg.message = m.Msg;
            winmsg.wParam = m.WParam;
            winmsg.lParam = m.LParam;
            winmsg.time = GetMessageTime();
            winmsg.x = (short)pos;
            winmsg.y = (short)(pos >> 16);

            Marshal.StructureToPtr(winmsg, evtmsg.EventPtr, false);
            evtmsg.Message = 0;
            TwRC rc = DSevent(appid, srcds, TwDG.Control, TwDAT.Event,
TwMSG.ProcessEvent, ref evtmsg);
            if (rc == TwRC.NotDSEvent)
                return TwainCommand.Not;
            if (evtmsg.Message == (short)TwMSG.XFerReady)
                return TwainCommand.TransferReady;
            if (evtmsg.Message == (short)TwMSG.CloseDSReq)
                return TwainCommand.CloseRequest;
            if (evtmsg.Message == (short)TwMSG.CloseDSOK)
                return TwainCommand.CloseOk;
            if (evtmsg.Message == (short)TwMSG.DeviceEvent)
                return TwainCommand.DeviceEvent;

            return TwainCommand.Null;
        }
```

Código del Módulo 2

En este código, el cual se encarga de aplicar un filtro gaussiano a la imagen, también se utilizó una API, la cual esta basada en el editor openSource GIMP, la cual se encargaba de aplicar algunos filtros y detectar bordes, se pensaba utilizar esta API al momento de detectar los bordes de la imagen, pero esto no fue necesario, ya que al momento de aplicar el algoritmo “Thinning”, los bordes fueron detectados automáticamente.

```
public partial class Mainform : Form
    {
        public Mainform()
        {
            InitializeComponent();
        }
```

```
        Canny CannyData;

        private void Directorio()
        {
            Directory.CreateDirectory("C:/Calificador");

            if (!Directory.Exists("C:/Calificador"))
            {
                Directory.CreateDirectory("C:/Calificador");
            }
            else
                MessageBox.Show("Carpeta lista");
        }

        private void Guardar()
        {
            string Nombre = "C:/Calificador/alumno_filtrado" + ".jpg";
            //verificamos si el final de la instancia en Jpeg
            if (Nombre.EndsWith("jpg"))
            {
                GaussianFilteredImage.Image.Save(Nombre, ImageFormat.Jpeg);
                //MessageBox.Show("Imagen guardada");
            }
        }

        private void toolStripLabel1_Click(object sender, EventArgs e)
        {
            //Stream myStream = null;
            OpenFileDialog of1 = new OpenFileDialog();
            of1.InitialDirectory = "C:\\";
            of1.Filter = "JPEG Files (*.jpg)|*.jpeg";
            if (of1.ShowDialog() == DialogResult.OK)
            {
                Imagen.Image = new Bitmap(of1.FileName);
            }
            /*try
            {
                Imagen.Image = Bitmap.FromFile("C:/Users/Jorge
Nieto/Pictures/PT/examen.jpeg");
            }
            catch (ApplicationException ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }*/
            Directorio();
        }

        private void selectFullImageToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            DateTime dt1 = new DateTime();
            DateTime dt2 = new DateTime();
            TimeSpan dt3 = new TimeSpan();

            dt1 = DateTime.Now;
            pg1.Value = 0;
```

```
            CannyData = new Canny ((Bitmap)Imagen.Image);
            pg1.Value = 10;

            dt2 = DateTime.Now;
            dt3 = dt2 - dt1;
            time.Text = dt3.ToString();
            pg1.Value = 100;
        }

        private void Mainform_Load(object sender, EventArgs e)
        {

        }

        private void BtnCannyEdgeDetect_Click(object sender, EventArgs e)
        {
            DateTime dt1 = new DateTime();
            DateTime dt2 = new DateTime();
            TimeSpan dt3 = new TimeSpan();
            float TH, TL, Sigma;
            int MaskSize;

            dt1 = DateTime.Now;
            pg1.Value = 0;
            TH = 40;
            TL = 5;

            MaskSize = 5;
            Sigma = (float)Convert.ToDouble(1);
            pg1.Value = 10;
            CannyData = new Canny((Bitmap)Imagen.Image,TH,TL,MaskSize,Sigma );

            GaussianFilteredImage.Image =
CannyData.DisplayImage(CannyData.FilteredImage);

            dt2 = DateTime.Now;
            dt3 = dt2 - dt1;
            time.Text = dt3.ToString();
            pg1.Value = 100;

        }


        private void Imagen_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Guardar();
            this.Dispose();
        }

        private void button2_Click(object sender, EventArgs e)
```

```
        {
            this.Dispose();
        }
    }


class Canny
    {
        public int Width, Height;
        public Bitmap Obj;
        public int[,] GreyImage;
        //Gaussian Kernel Data
        int [,] GaussianKernel;
        float MaxHysteresisThresh, MinHysteresisThresh;
        int KernelWeight ;
        int KernelSize =5;
        float Sigma = 1;
        public int[,] FilteredImage;
        public float[,] Gradient;

        public Canny(Bitmap Input)
        {
            MaxHysteresisThresh = 20F;
            MinHysteresisThresh = 10F;
            Obj = Input;
            Width = Obj.Width;
            Height = Obj.Height;
            EdgeMap = new int[Width, Height];
            VisitedMap = new int[Width, Height];

            ReadImage();
            return;
        }
public Canny(Bitmap Input, float Th, float Tl)
        {

            // Gaussian and Canny Parameters

            MaxHysteresisThresh = Th;
            MinHysteresisThresh = Tl;

            Obj = Input;
            Width = Obj.Width;
            Height = Obj.Height;

            EdgeMap = new int[Width, Height];
            VisitedMap = new int[Width, Height];

            ReadImage();
            return;
        }
public Canny(Bitmap Input, float Th, float Tl, int GaussianMaskSize, float
SigmaforGaussianKernel)
        {

            // Gaussian and Canny Parameters
```

```
            MaxHysteresisThresh = Th;
            MinHysteresisThresh = Tl;
            KernelSize = GaussianMaskSize;
            Sigma = SigmaforGaussianKernel;
            Obj = Input;
            Width = Obj.Width;
            Height = Obj.Height;

            EdgeMap = new int[Width, Height];
            VisitedMap = new int[Width, Height];

            ReadImage();
            DetectCannyEdges();
            return;
        }
private void GenerateGaussianKernel(int N, float S ,out int Weight)
        {

            float Sigma = S ;
            float pi;
            pi = (float)Math.PI;
            int i, j;
            int SizeofKernel=N;

            float [,] Kernel = new float [N,N];
            GaussianKernel = new int [N,N];
            float[,] OP = new float[N, N];
            float D1,D2;


            D1= 1/(2*pi*Sigma*Sigma);
            D2= 2*Sigma*Sigma;

            float min=1000;

            for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
            {
               for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
                {
                    Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] = ((1 / D1) *
(float)Math.Exp(-(i * i + j * j) / D2));
                    if (Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] < min)
                        min = Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];

                }
            }
            int mult = (int)(1 / min);
            int sum = 0;
            if ((min > 0) && (min < 1))
            {

                for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
                {
                    for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
                    {
```

```
                        Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(float)Math.Round(Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] * mult, 0);
                        GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(int)Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
                        sum = sum + GaussianKernel[SizeofKernel / 2 + i,
SizeofKernel / 2 + j];
                    }

                }

            }
            else
            {
                sum = 0;
                for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
                {
                    for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
                    {
                        Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(float)Math.Round(Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] , 0);
                        GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(int)Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
                        sum = sum + GaussianKernel[SizeofKernel / 2 + i,
SizeofKernel / 2 + j];
                    }

                }

            }
          //Normalizing kernel Weight
          Weight= sum;

         return;
        }
private int[,] GaussianFilter(int[,] Data)
        {
            GenerateGaussianKernel(KernelSize, Sigma,out KernelWeight);

            int[,] Output = new int[Width, Height];
            int i, j,k,l;
            int Limit = KernelSize /2;

            float Sum=0;


            Output = Data; // Removes Unwanted Data Omission due to kernel bias
while convolution


            for (i = Limit; i <= ((Width - 1) - Limit); i++)
            {
                for (j = Limit; j <= ((Height - 1) - Limit); j++)
                {
                    Sum = 0;
                    for (k = -Limit; k <= Limit; k++)
                    {
```

```
                        for (l = -Limit; l <= Limit; l++)
                        {
                            Sum = Sum + ((float)Data[i + k, j + l] * GaussianKernel
[Limit + k, Limit + l]);

                        }
                    }
                    Output[i, j] = (int)(Math.Round(Sum/ (float)KernelWeight));
                }

            }


            return Output;
        }
```

Código del Módulo 3

```
Bitmap image = new Bitmap("C:/Users/HUGO/Desktop/PT/pruebasexamen/filtro.jpg");

            for (int i = 0; i < image.Width; i++)
            {
                for (int j = 0; j < image.Height; j++)
                {
                    Color c = image.GetPixel(i, j);
                    //obtiene valor del pixelpixel

                    short gris=(short)((c.R+c.B+c.G)/3);

                    //compara el valor del pixel con el Umbral
                    if (gris<=210)
                    {
                        image.SetPixel(i, j, Color.Black);

                    }
                    else
                    {
                        image.SetPixel(i, j, Color.White);

                    }
                }
            }

            image.Save("C:/Users/HUGO/Desktop/PT/pruebasexamen/examenbin" + ".bmp",
ImageFormat.Bmp);
```

Código del Módulo 4

```
public static Boolean thinning(Boolean firstpass)
        {
            int[,] di = new int[image.Width, image.Height];
            int m = 0;
            inicial(di);
            Boolean[] paux = new Boolean[8];
            for (int i = 1; i < image.Width - 1; i++)
            {
                for (int j = 1; j < image.Height - 1; j++)
                {
                    int pa = (int)(image.GetPixel(i, j).GetBrightness());

                    if (pa == 0)
                    {
                        int p0 = (int)(image.GetPixel(i - 1, j).GetBrightness());
                        int p1 = (int)(image.GetPixel(i - 1, j +
1).GetBrightness());
                        int p2 = (int)(image.GetPixel(i, j + 1).GetBrightness());
                        int p3 = (int)(image.GetPixel(i + 1, j +
1).GetBrightness());
                        int p4 = (int)(image.GetPixel(i + 1, j).GetBrightness());
                        int p5 = (int)(image.GetPixel(i + 1, j -
1).GetBrightness());
                        int p6 = (int)(image.GetPixel(i, j - 1).GetBrightness());
                        int p7 = (int)(image.GetPixel(i - 1, j -
1).GetBrightness());

                        if (p0 == 1) paux[0] = true;
                        else paux[0] = false;
                        if (p1 == 1) paux[1] = true;
                        else paux[1] = false;
                        if (p2 == 1) paux[2] = true;
                        else paux[2] = false;
                        if (p3 == 1) paux[3] = true;
                        else paux[3] = false;
                        if (p4 == 1) paux[4] = true;
                        else paux[4] = false;
                        if (p5 == 1) paux[5] = true;
                        else paux[5] = false;
                        if (p6 == 1) paux[6] = true;
                        else paux[6] = false;
                        if (p7 == 1) paux[7] = true;
                        else paux[7] = false;

                        int A = vecinos(paux);//suma de vecino del pixel
                        int B = trancision(paux);
                        if (2 <= A && A <= 6)//si hay menos de 2 o más de 6 vecinos
al rededor del pixel
                        {
                            if (B == 1)//si hay solo una trancision de 0 a 1
                            {
                                if (firstpass)//esta en la primera sub-iteracion
                                {
                                    if ((p0 * p2 * p4 == 0) && (p2 * p4 * p6 == 0)
&& (p1!=0))
                                    {
                                        di[i, j] = 1;
```

```
                                    m++;
                                }
                            }
                            else
                            {
                                if ((p0 * p2 * p6 == 0) && (p0 * p4 * p6 == 0)
&& (p7!=0))
                                {
                                    di[i, j] = 1;
                                    m++;
                                }
                            }
                        }
                    }
                }
            }
            if (m == 0)
                return false;
            borra(di);
            return true;
        }
```

Código del Módulo 5

```
static void inundacion(Bitmap image, Point px, Color original, Color reemplazo)
        {
            Queue<Point> pixel = new Queue<Point>();
            pixel.Enqueue(px);
            while (pixel.Count > 0)
            {
                Point n = pixel.Dequeue();
                if (!relacion_color(image.GetPixel(n.X, n.Y), original))
                    continue;
                Point w = n, e = new Point(n.X + 1, n.Y);
                while ((w.X > 0) && relacion_color(image.GetPixel(w.X, w.Y),
original))
                {
                    image.SetPixel(w.X, w.Y, reemplazo);
                    if ((w.Y > 0) && relacion_color(image.GetPixel(w.X, w.Y - 1),
original))
                        pixel.Enqueue(new Point(w.X, w.Y - 1));
                    if ((w.Y < image.Height - 1) &&
relacion_color(image.GetPixel(w.X, w.Y + 1), original))
                        pixel.Enqueue(new Point(w.X, w.Y + 1));
                    w.X--;
                }
                while ((e.X < image.Width - 1) && relacion_color(image.GetPixel(e.X,
e.Y), original))
                {
                    image.SetPixel(e.X, e.Y, reemplazo);
                    if ((e.Y > 0) && relacion_color(image.GetPixel(e.X, e.Y - 1),
original))
                        pixel.Enqueue(new Point(e.X, e.Y - 1));
                    if ((e.Y < image.Height - 1) &&
relacion_color(image.GetPixel(e.X, e.Y + 1), original))
```

```
                    pixel.Enqueue(new Point(e.X, e.Y + 1));
                e.X++;
            }
        }
    }
```

Código del Módulo 7 y Módulo 8

Esta función se repite para cada sección de preguntas de la hoja de respuestas, ya que se utilizó la misma idea para evaluar todas las respuestas, por eso creímos conveniente solamente poner la función de una sección, ya que a lectura se haría muy lenta.

```
public static int calificarRecordar()
        {
            Bitmap image = new Bitmap("C:/Tempo/examen_inundacion.bmp");
            int contrecordar = 0;

            float a11 = (float)(image.GetPixel(278,443).GetBrightness());
            float c11 = (float)(image.GetPixel(445,493).GetBrightness());
            float b11 = (float)(image.GetPixel(362,542).GetBrightness());
            float d11 = (float)(image.GetPixel(531,590).GetBrightness());

            int[] p1 = new int[3] {362,445,531};
            int[] p2 = new int[3] {278,362,531};
            int[] p3 = new int[3] {278,445,531};
            int[] p4 = new int[3] {278,362,445};


            //pregunta 1.1****************************************
            int i=0;
            float respinc1=0;

            if (a11 != 1.0)
            {
                while(i<3)
                {
                    respinc1= (float)(image.GetPixel(p1[i],443).GetBrightness());

                    if (respinc1 == 1)
                    {
                        i++;
                        //Console.WriteLine("a"+i);
                    }
                    else
                        i = 4;
                }
            }
            else
            Console.WriteLine("Respuesta 1.1 incorrecta");

            if (i == 3)
            {
```

```
    Console.WriteLine("------------------");
    Console.WriteLine("Respuesta 1.1 Coreecta");
    Console.WriteLine("------------------");
    contrecordar++;
}


//pregunta 1.2*****************************************************
int j = 0;
float respinc2 = 0;

if (c11 != 1.0)
{
    while (j < 3)
    {
        respinc2 = (float)(image.GetPixel(p2[j], 493).GetBrightness());

        if (respinc2 == 1)
        {
            j++;
            //Console.WriteLine("a"+i);
        }
        else
            j = 4;
    }
}
else
    Console.WriteLine("Respuesta 1.2 incorrecta");

if (j == 3)
{
    Console.WriteLine("------------------");
    Console.WriteLine("Respuesta 1.2 Coreecta");
    Console.WriteLine("------------------");
    contrecordar++;
}

//pregunta1.3**************************

int k = 0;
float respinc3 = 0;

if (b11 != 1.0)
{
    while (k < 3)
    {
        respinc3 = (float)(image.GetPixel(p3[j], 542).GetBrightness());

        if (respinc3 == 1)
        {
            k++;
            //Console.WriteLine("a"+i);
        }
        else
            k = 4;
    }
```

```
    }
    else
        Console.WriteLine("Respuesta 1.3 incorrecta");

    if (k == 3)
    {
        Console.WriteLine("------------------");
        Console.WriteLine("Respuesta 1.3 Coreecta");
        Console.WriteLine("------------------");
        contrecordar++;
    }

    //pregunta1.4****************************************
    int l = 0;
    float respinc4 = 0;

    if (d11 != 1.0)
    {
        while (l < 3)
        {
            respinc4 = (float)(image.GetPixel(p4[l], 590).GetBrightness());

            if (respinc4 == 1)
            {
                l++;
                //Console.WriteLine("a"+i);
            }
            else
                l = 4;
        }
    }
    else
        Console.WriteLine("Respuesta 1.4 incorrecta");

    if (l == 3)
    {
        Console.WriteLine("------------------");
        Console.WriteLine("Respuesta 1.4 Coreecta");
        Console.WriteLine("------------------");
        contrecordar++;
    }

    return contrecordar;
}


public static int resultados()
{
    int a = 0;
    int b = 0;
    int c = 0;
    int d = 0;
    int e = 0;
    int f = 0;

    int resultado = 0;
```

```
        a = calificarRecordar();
        b = calificarComprender();
        c = calificarAplicar();
        d = calificarAnalizar();
        e = calificarEvaluar();
        f=calificarCrear();

        resultado = (a + b + c + d + e + f) / 6;
        return resultado;
    }
```

Apéndice B: Guía rápida del producto

Es recomendable utilizar un escáner tipo portátil, así se obtiene una imagen del tamaño requerido, evitando el tener que editar la imagen para quitar espacios extras generados por el escáner.

Al momento de ejecutar la instalación de este programa, es necesario agregar el archivo ejecutable de la API de Canny Edge en la línea 218, para que el programa pueda hacer uso de este.

Primeramente se abre el programa, se da clic en el botón de escanear, se hará el escaneo de la hoja y se abrirá una ventana como la siguiente:

Imagen 1: escanear

Después se da clic en el botón iniciar, para empezar a aplicar el filtro gaussiano a la imagen

Imagen 2: Filtro gaussiano

Después aparecerá automáticamente la siguiente ventana, se empieza a aplicar el módulo de binarizar

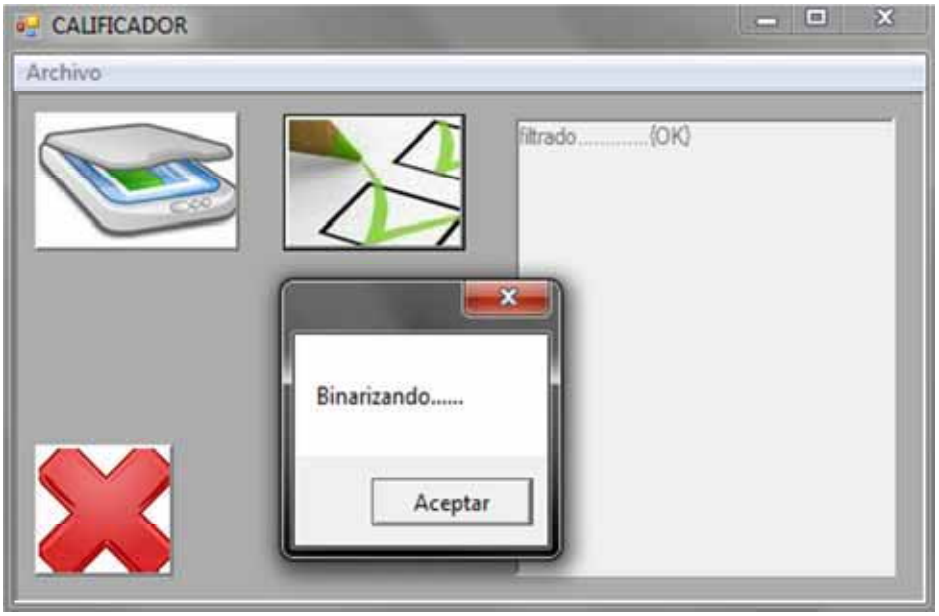


Imagen 3: Binarizar

Luego aparecerá la siguiente imagen, indicando que se está aplicando el módulo de “thinning”

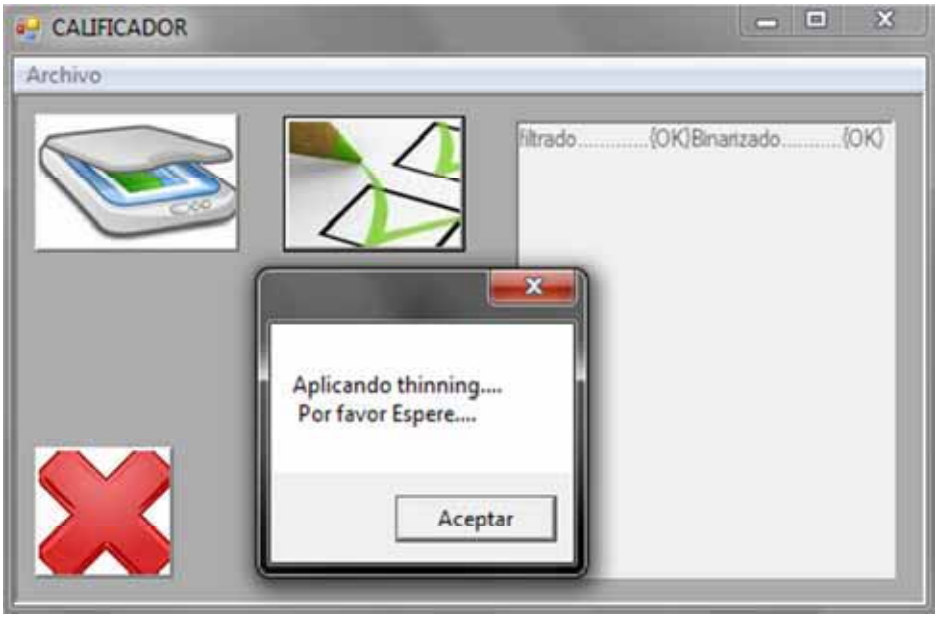


Imagen 4: esqueletonizar

Aparecerá la siguiente imagen indicando que se está aplicando el módulo de crecimiento de regiones

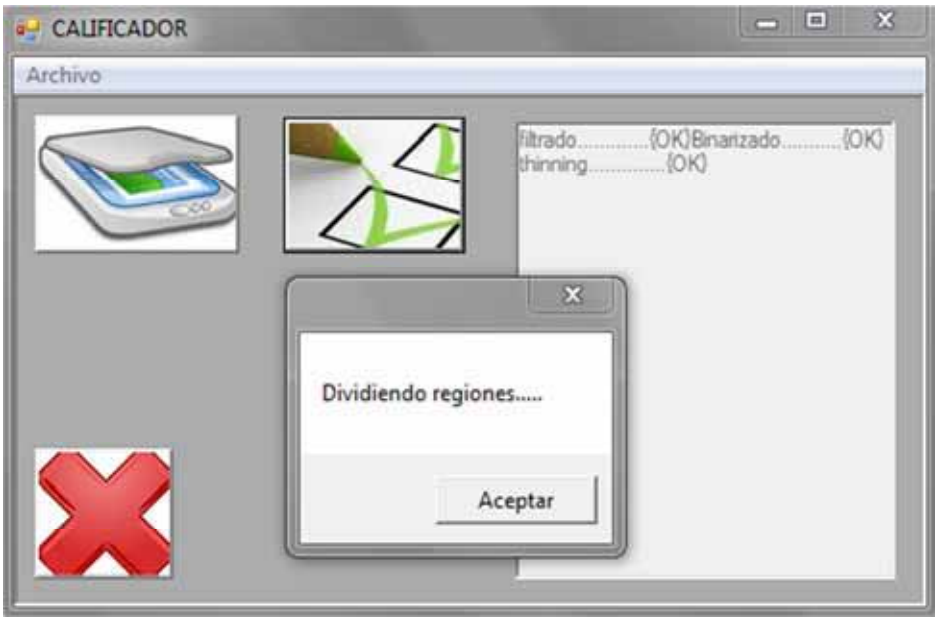


Imagen 5: Crecimiento de regiones

Aparecerá la siguiente imagen indicando que se está aplicando el módulo calificación

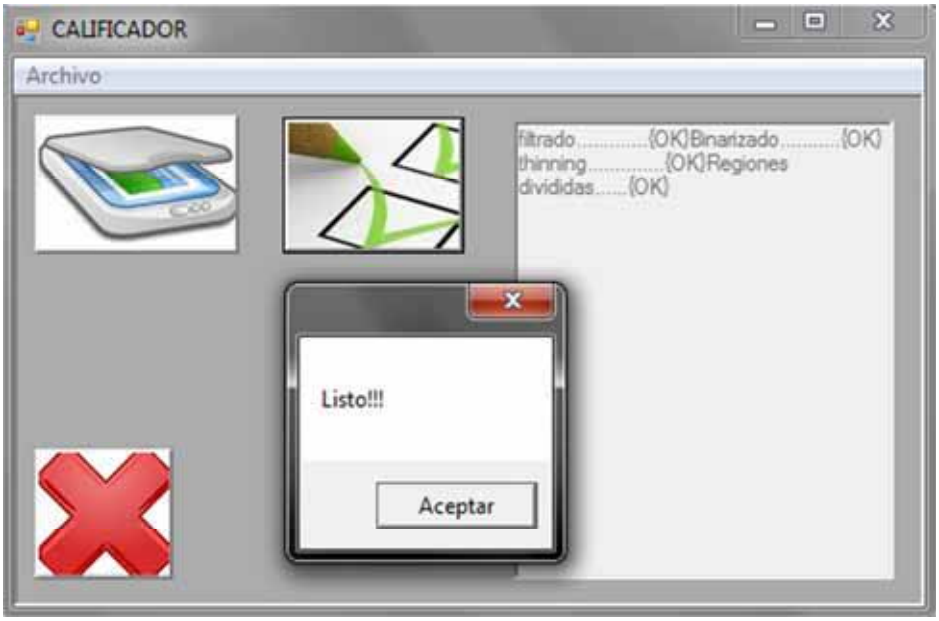


Imagen 6: calificar

Al último mostrara la siguiente imagen, indicando que ya termino de calificar

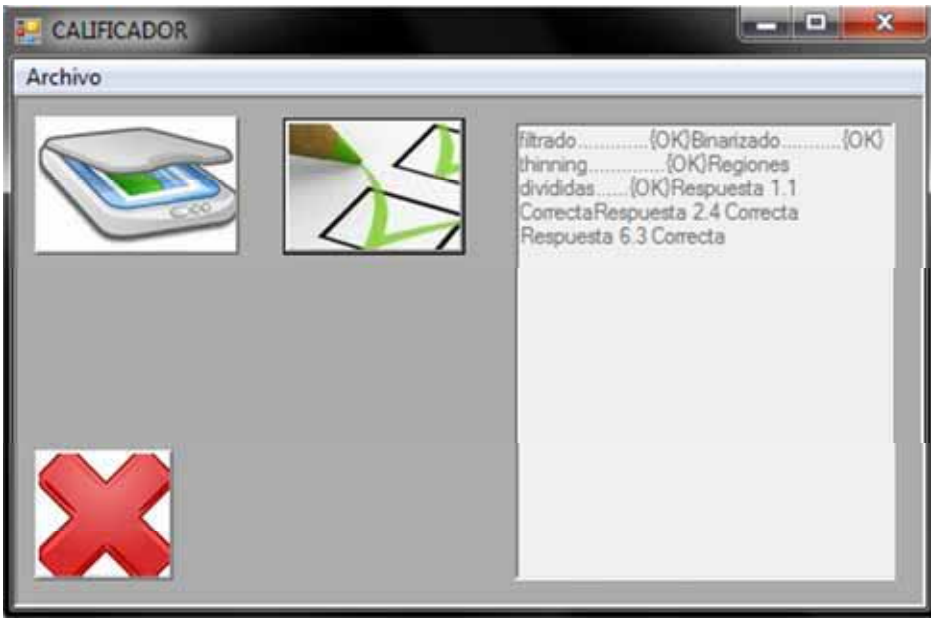


Imagen 7: resultado