

Análisis de patrones con EmguCV

Objetivo: Familiarizar al alumno con la utilización de EmguCV para el análisis de patrones de un set de imágenes y por webcam.

Creación de aplicación EmguCV

Para crear un proyecto de procesamiento digital de imágenes utilizando EmguCV, se seguirán los siguientes pasos.

Crear el Proyecto

Cree un nuevo proyecto tipo Aplicación de Windows Forms siguiendo los pasos de las figuras 1, 2 y 3.



Figura 1

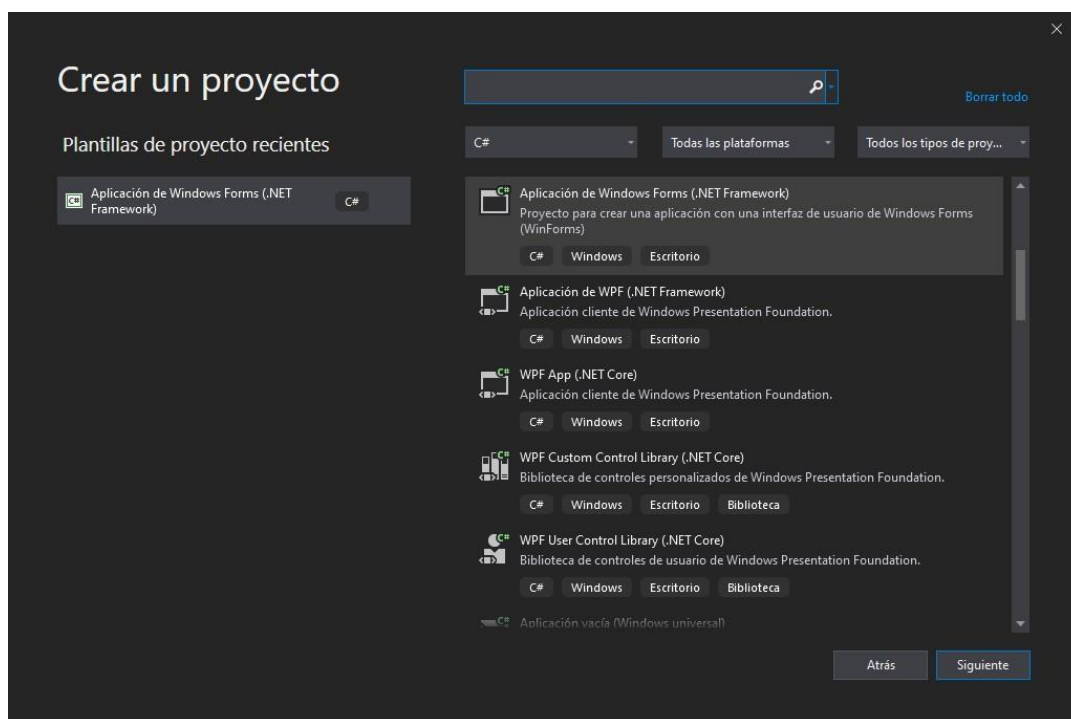


Figura 2

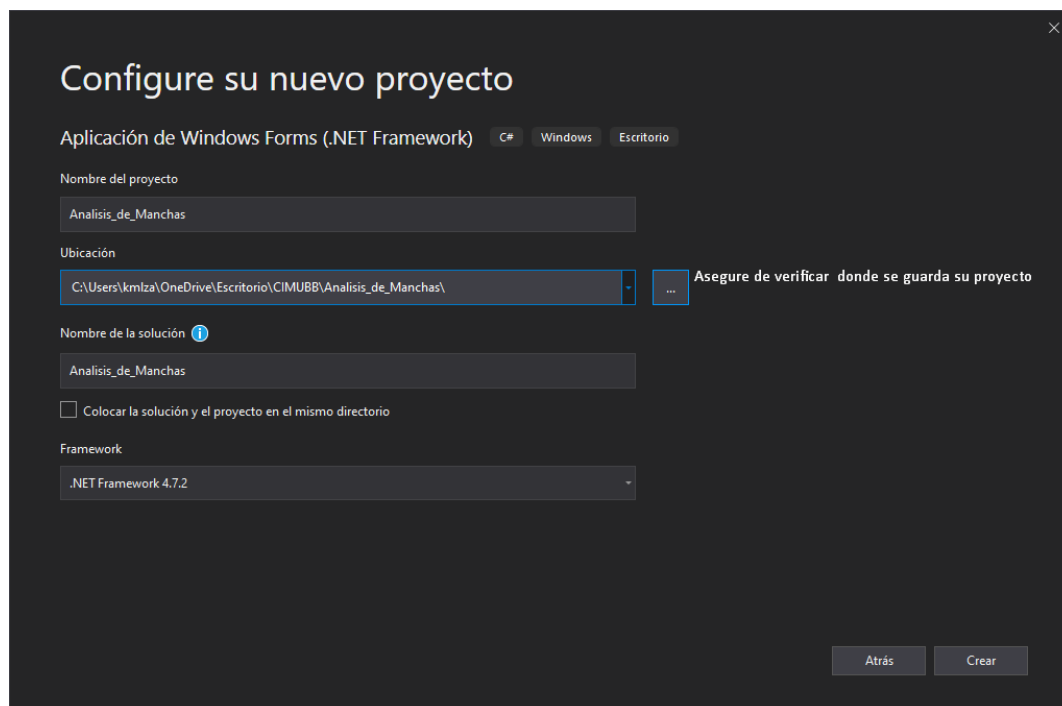


Figura 3

Agregar las referencias

Utilizaremos la versión EMGUCV 2.4.10, encontrada en la carpeta del practico o se puede descargar mediante el siguiente link:

<https://sourceforge.net/projects/emgucv/files/emgucv/2.4.10/>

→ libemgucv-windows-universal-cuda-2.4.10.1940-selfextract-zip.exe

Luego de instalar la versión, nos dirigimos a nuestro entorno de desarrollo y en el costado derecho hacemos click derecho en *Referencias* -> *Agregar referencias* (ver figura 4).

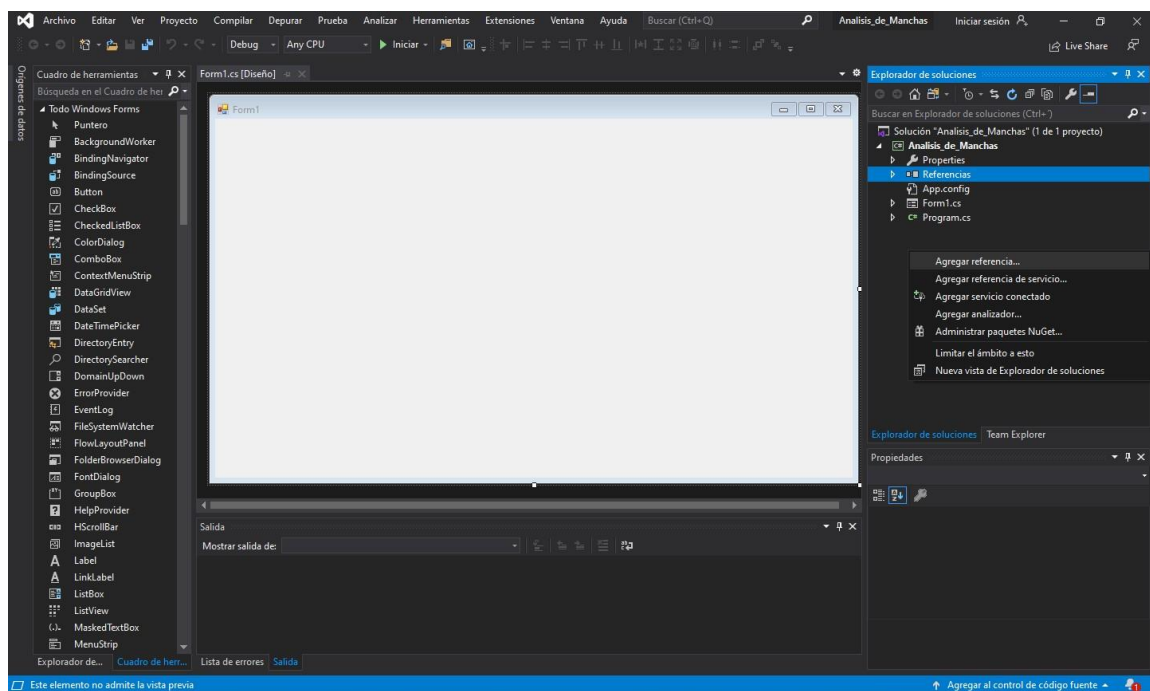


Figura 4

Presione el botón *Examinar*, se deben agregar las referencias a los siguientes archivos ubicados dentro de la carpeta *bin* en la carpeta de instalación de EMGUCV (ver figura 5 y 6).

Emgu.CV.DebuggerVisualizers.VS2010.dll
Emgu.CV.DebuggerVisualizers.VS2012.dll
Emgu.CV.dll
Emgu.CV.GPU.dll
Emgu.CV.ML.dll
Emgu.CV.OCR.dll
Emgu.CV.Stitching.dll
Emgu.CV.UI.dll
Emgu.CV.VideoStab.dll
Emgu.Util.dll



Desplegada la ventana, debe seleccionar la carpeta *bin* contenida en el directorio de EmguCV.

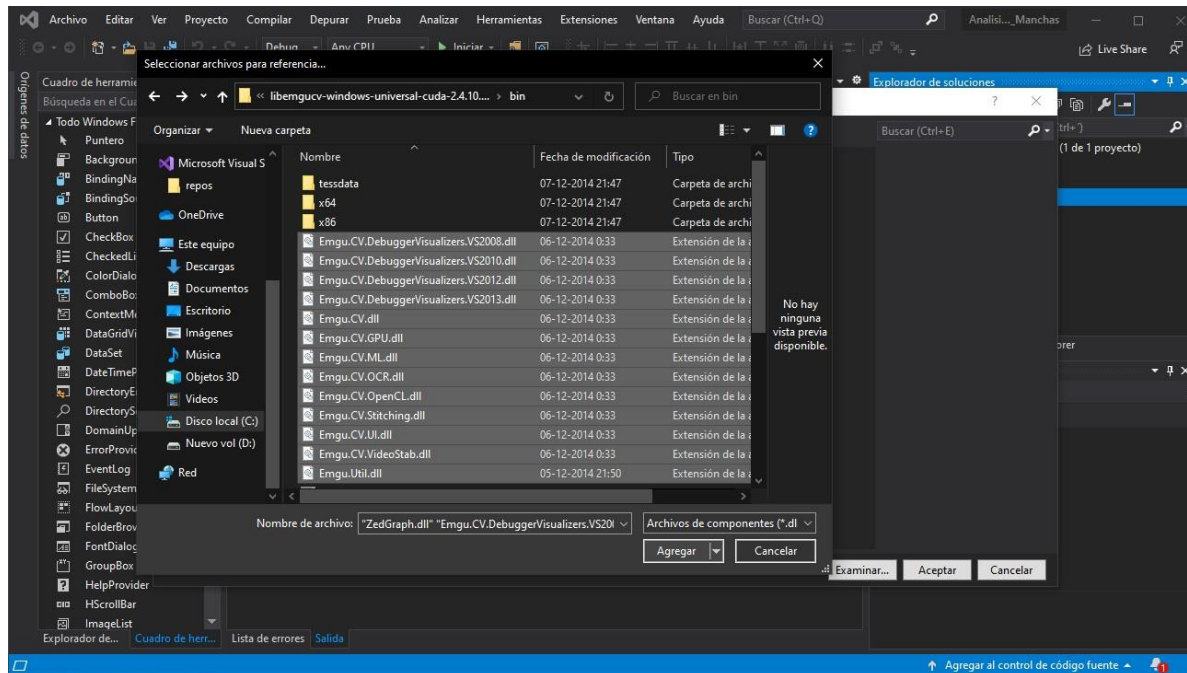


Figura 5

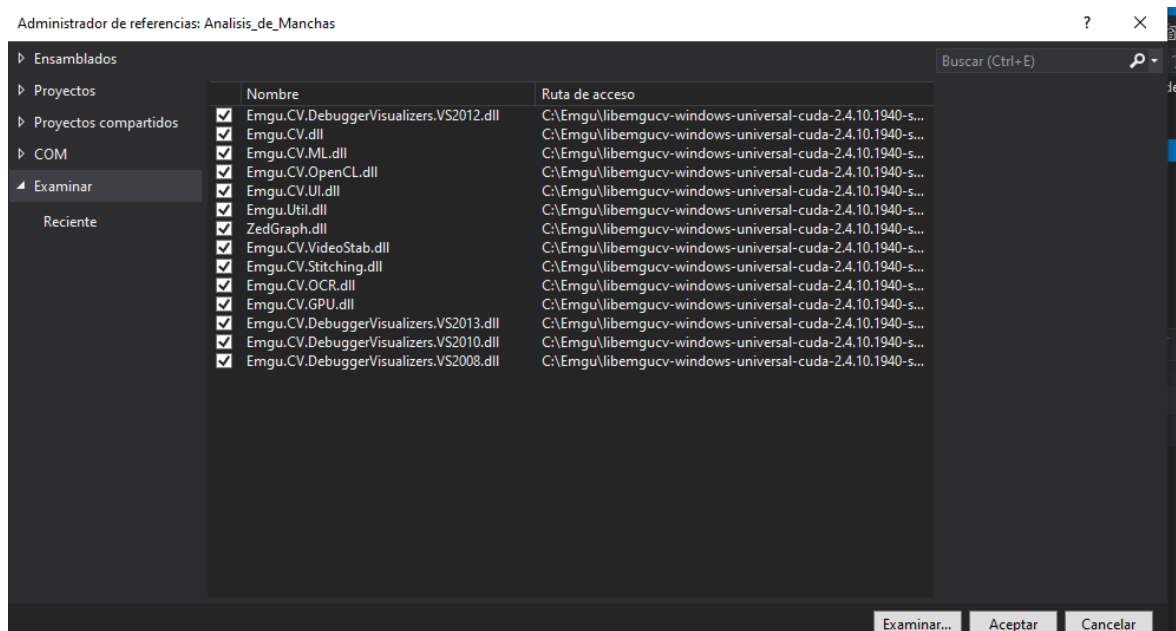


Figura 6

Debería visualizar los resultados en la sección de *Referencias* en el costado izquierdo (ver figura 7).

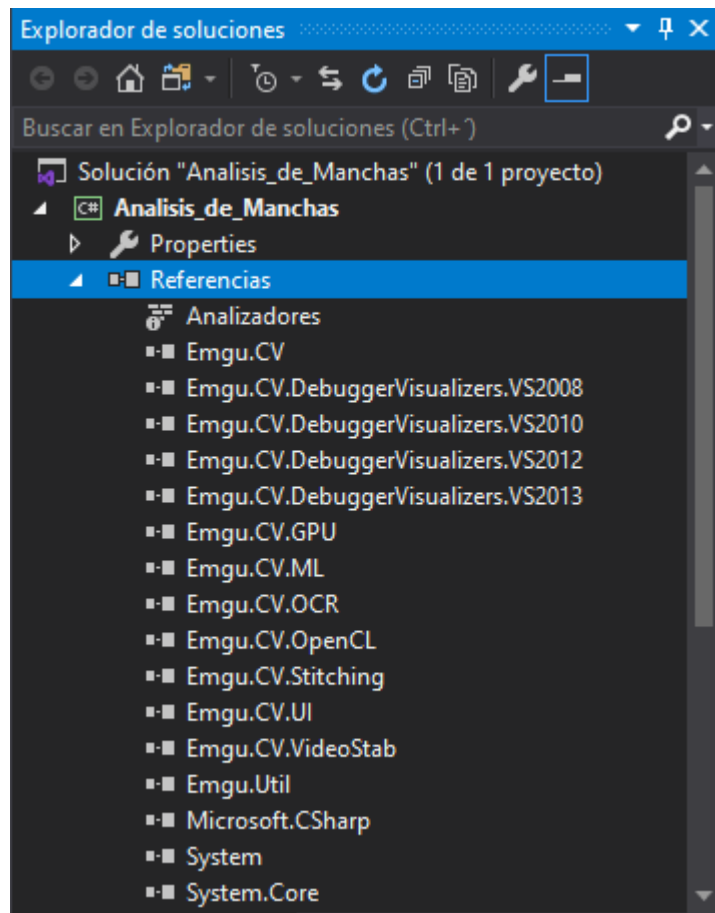


Figura 7

Luego es necesario agregar algunas referencias externas al proyecto. Seleccione las opciones *Agregar* -> *Elemento existente* haciendo click derecho sobre el nombre de la solución en el costado derecho del IDE (ver figura 8).

Dependiendo de si su sistema es de 64 o 32 bits, deberá seleccionar todo el contenido de una de estas carpetas. En el explorador debe seleccionar la opción "Todos los Archivos (*.*)", seleccione todo el contenido de la carpeta x64 si su sistema es de 64 bits, o de la carpeta x86 si es de 32 bits, y luego presione *Agregar*.

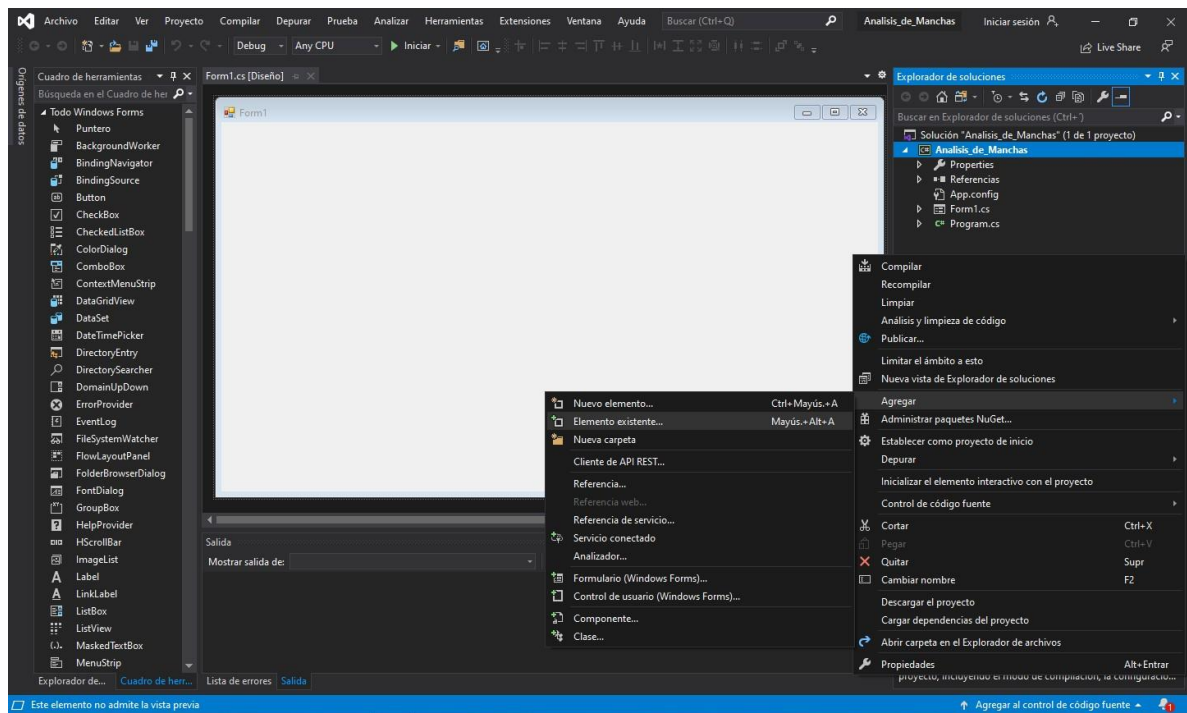


Figura 8

Nota: Recordar que las carpetas se encuentran alojadas en el directorio *bin* de EMGUCV. Finalmente, seleccionar las referencias agregadas. Selecciónelas una a una y configure las propiedades de (Copy To output directory) (Copiar siempre).

La ubicación de la carpeta debería ser la siguiente:

Sistema de 64 bits: **C:\~\Emgu\libemgucv-windows-universal-cuda-2.4.10.1940-selfextract-
zip\bin\x64**

Sistema de 32 bits: **C:\~\Emgu\libemgucv-windows-universal-cuda-2.4.10.1940-selfextract-
zip\bin\x86**

Método alternativo: Si obtiene un error relacionado con la falta de un archivo necesario para la ejecución del programa, copie directamente el contenido de la carpeta "*bin->X86*" o "*bin->X64*" y péguelo en la carpeta "*bin->Debug*" de nuestro proyecto.

Como resultado debería observar la lista de referencias en el menú *Referencias* del costado derecho del IDE, tal como se observa en la figura 9.

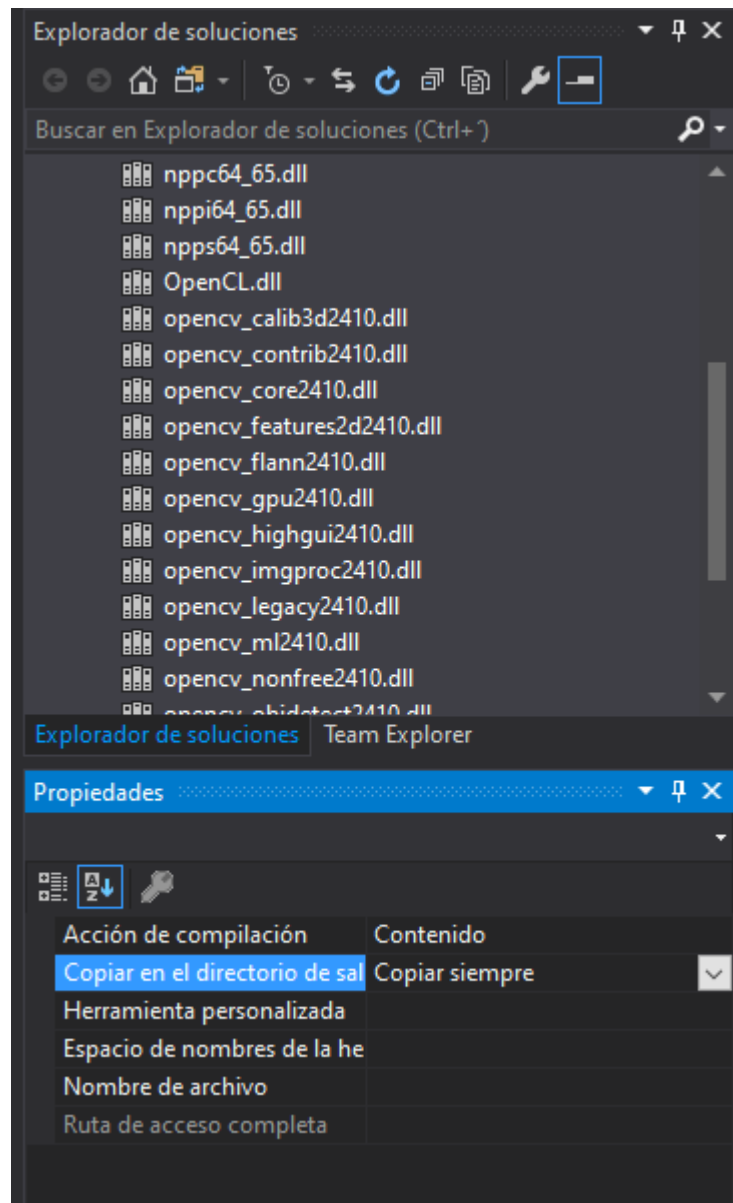


Figura 9

Agregando herramientas

Ahora nos falta agregar herramientas que vienen en EMGUCV para nuestro proyecto. Ingresamos al cuadro de herramientas en la parte izquierda del entorno de desarrollo o mediante comando CTRL + ALT + X. Como se observa en la figura 10 debe seleccionar *Agregar pestaña*.

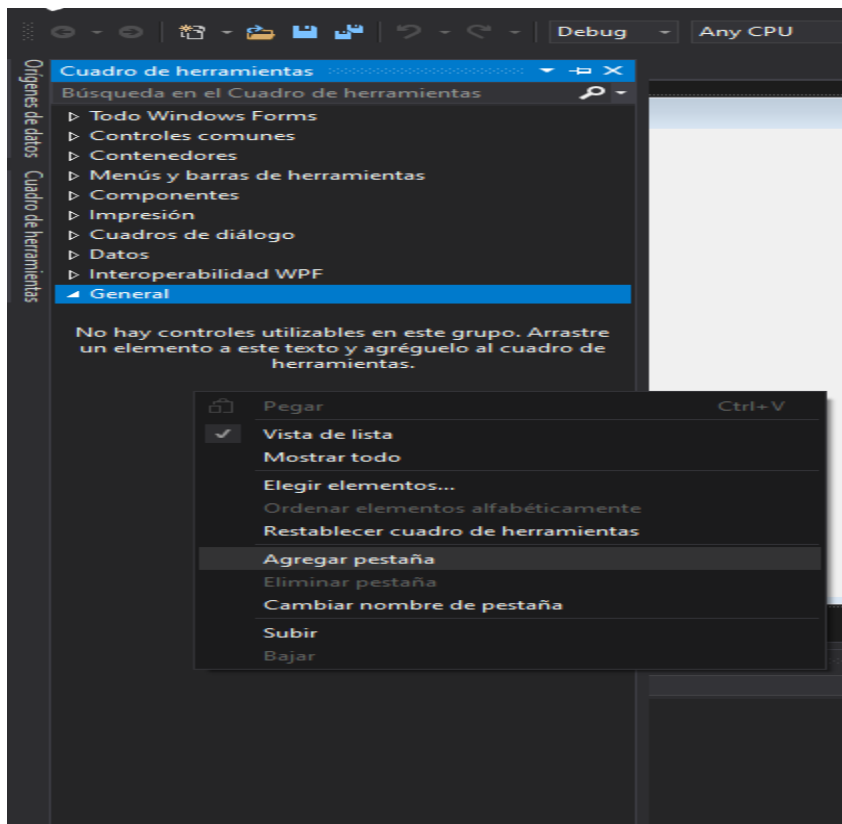


Figura 10

Le daremos el nombre de "Emgucv 2.4.10" para identificar nuestra pestaña. Luego con click derecho en la pestaña seleccionamos *Elegir elementos* (ver figura 11).

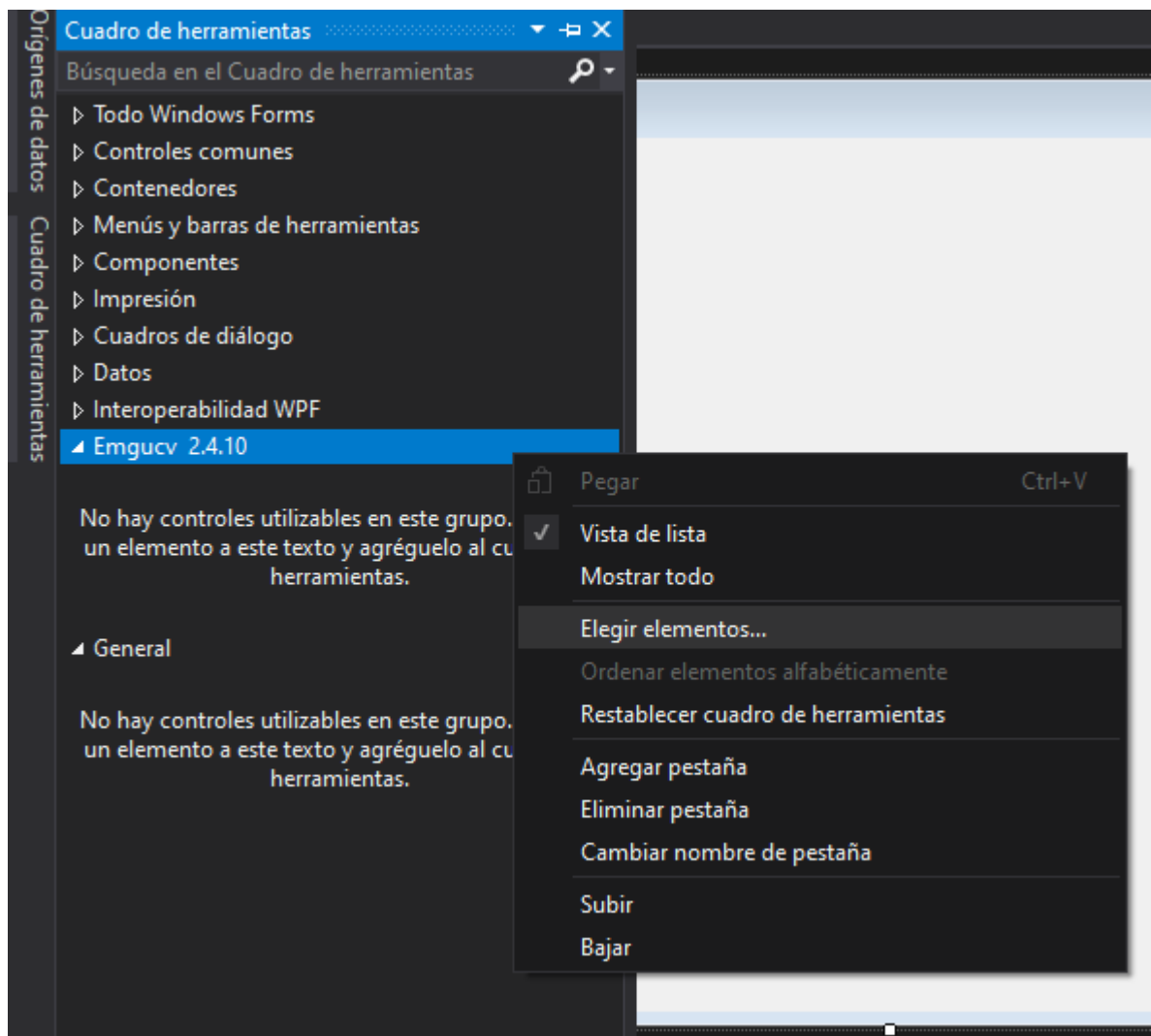


Figura 11

Elegimos la opción *Examinar* (figura 12) y nos dirigimos a la carpeta donde instalamos EMGUCV en la carpeta *bin* (figura 13).

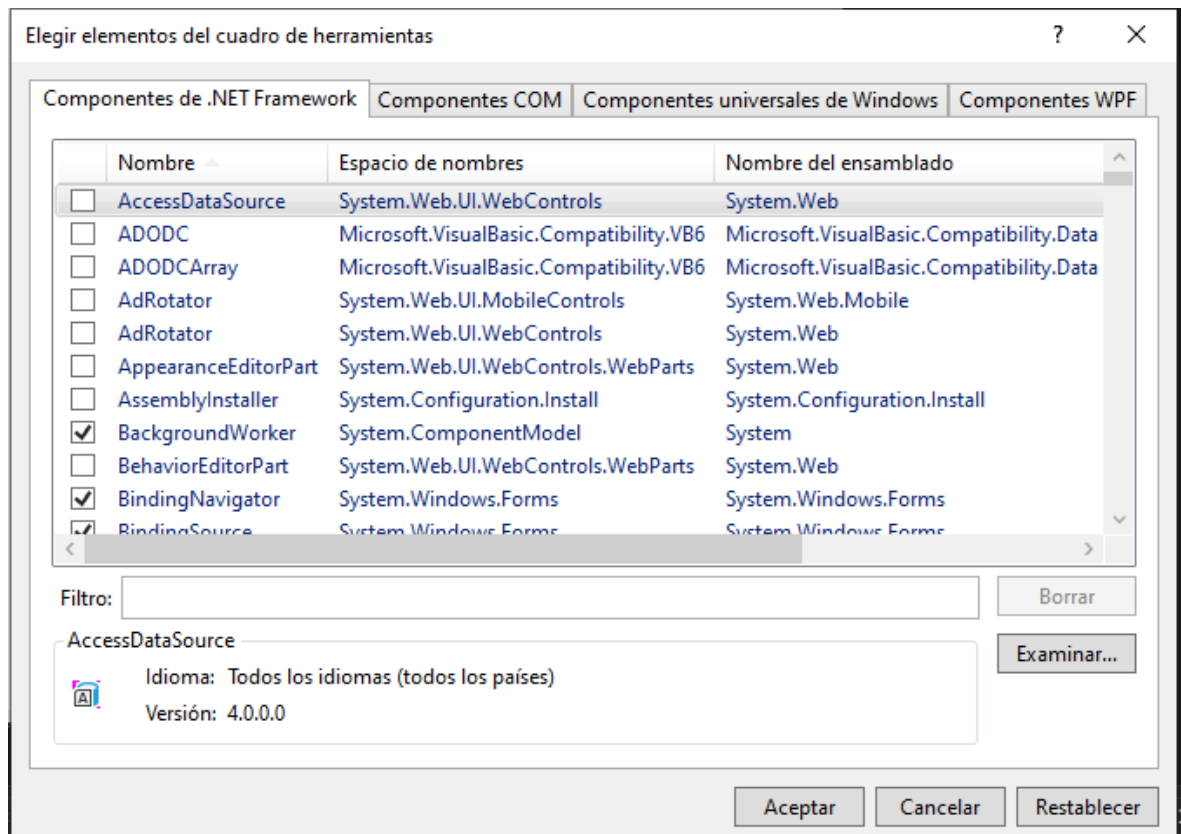


Figura 12

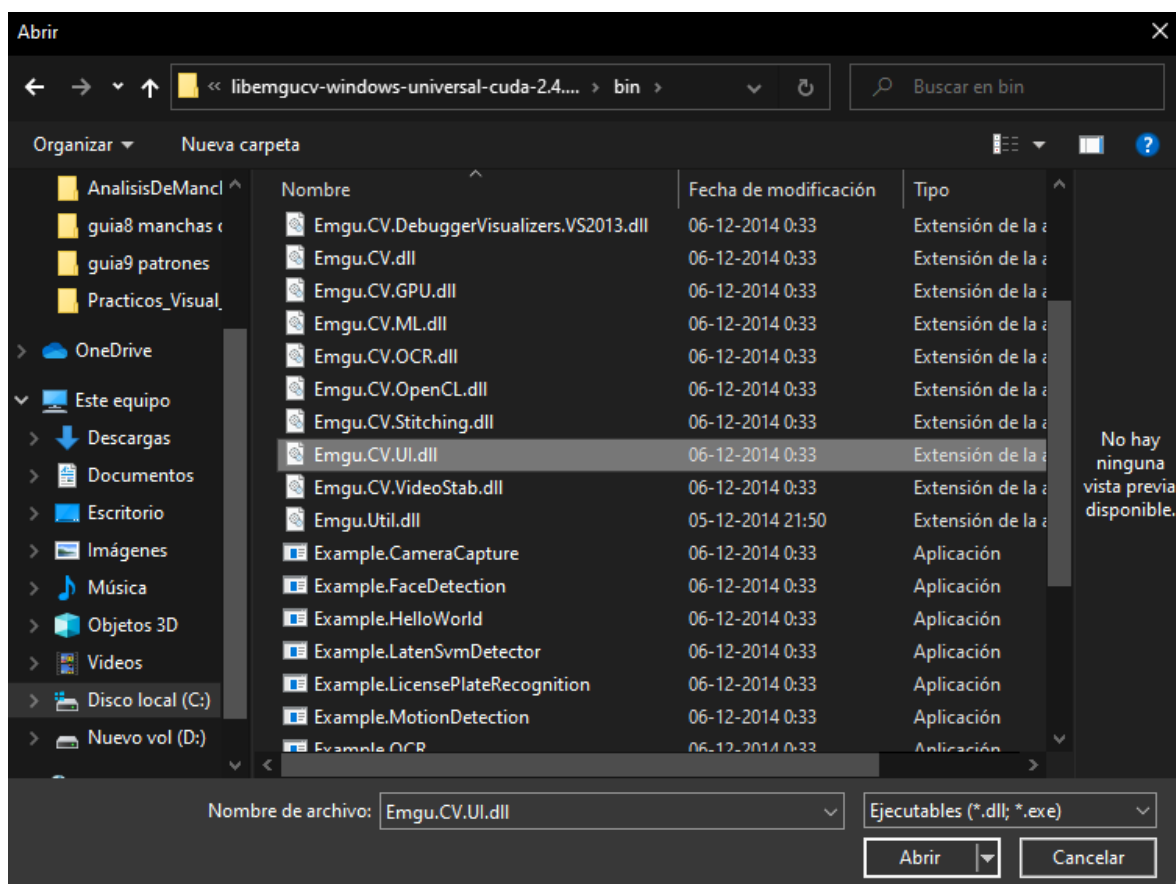


Figura 13

Elegimos el archivo *Emgu.CV.UI.dll* y presionamos *Abrir*. Como resultado debería observar lo mismo de la figura 14.

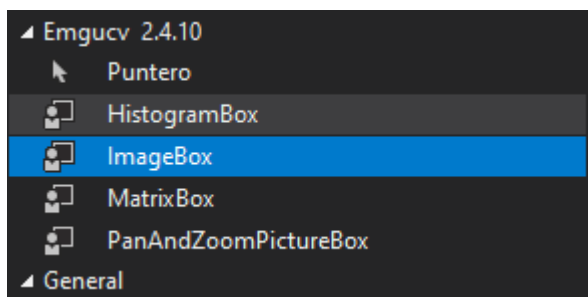


Figura 14

Agregar directivas

Finalmente, haciendo click sobre el *Form* principal, se abrirá una sección de código en la que deberá agregar las siguientes directivas (figura 15).

```
using Emgu.CV;  
using Emgu.CV.Structure;  
using Emgu.Util;  
using Emgu.CV.CvEnum;  
using Emgu.CV.DebuggerVisualizers;  
using Emgu.CV.VideoSurveillance;  
using Emgu.CV.ML;  
using Emgu.CV.ML.Structure;
```

Figura 15

Seleccionador de Imágenes

Añadir controles gráficos *ImageBox*, *Buttons*, *Label* y *OpenFileDialog* al *Form* principal. La figura 16 muestra los elementos agregados al *Form*.

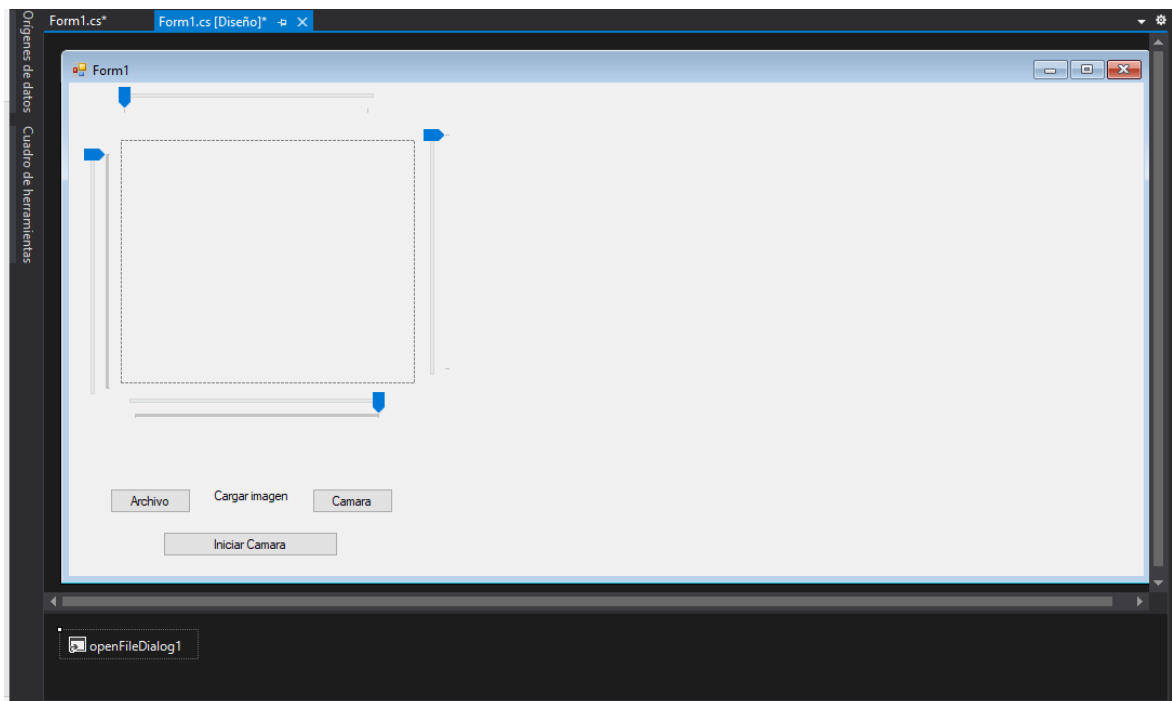


Figura 16

Propiedad	Label	Button	Button	Button	ImageBox	OpenFileDialog
Name	label1	button_Archivo	button_Camara	button_captura_webcam	imageBoxEntrada1	openFileDialog1
Text	Cargar imagen desde	Archivo	Cámara	Iniciar Cámara	-	-
SizeMode	-	-	-	-	StretchImage	-
Visible				False		

Luego hacer doble click en el botón “Archivo”, para poder programar su función con el siguiente código:

```
1 referencia
private void button_Archivo_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        choose_img = openFileDialog1.FileName;
        imagen_Capturada = new Image<Bgr, byte>(choose_img);

        imagen_Capturada = imagen_Capturada.Resize(640, 480, Emgu.CV.CvEnum.INTER.CV_INTER_NN);
        imageBoxEntrada1.Image = imagen_Capturada;
    }
    ROI();
}
```

Figura 18

Esto permitirá guardar la dirección del archivo imagen en la variable “choose_img”.

Nota: Las variables serán declaradas posteriormente para realizar la declaración en un solo paso.

Después hacer doble click en el botón “Cámara” para poder programar su función con el siguiente código:

```
1 referencia
private void button_Camara_Click(object sender, EventArgs e)
{
    button_captura_webcam.Visible = true;
}
```

Figura 19



Luego hacer doble click en el botón “Iniciar Cámara” para poder programar su función con el siguiente código:

```
1referencia
private void button_captura_webcam_Click(object sender, EventArgs e)
{
    if (_capture == null)
    {
        try
        {
            _capture = new Capture(0); // abrir camara

            _capture.ImageGrabbed += ProcesarFrame;
        }
        catch (NullReferenceException excpt)
        {
            MessageBox.Show(excpt.Message);
        }
        button_captura_webcam.Text = "Iniciar Captura";
    }
    else
    {
        if (_captureInProgress)
        {
            button_captura_webcam.Text = "Iniciar Captura";
            _capture.Pause();
        }
        else
        {
            button_captura_webcam.Text = "Detener Captura";
            _capture.Start();
        }

        _captureInProgress = !_captureInProgress;
    }
}
```

Figura 20

Aplicar ROI (región de interés) a la imagen

Si se desea procesar una región específica de la imagen, se debe utilizar esta herramienta antes del procesamiento. Para esto se utilizan controles *trackBar* (figura 21) los cuales delimitarán el área de interés. Además, se incluye un *ImageBox*, un *Button* y un *SaveFileDialog* (para guardar el área seleccionada).

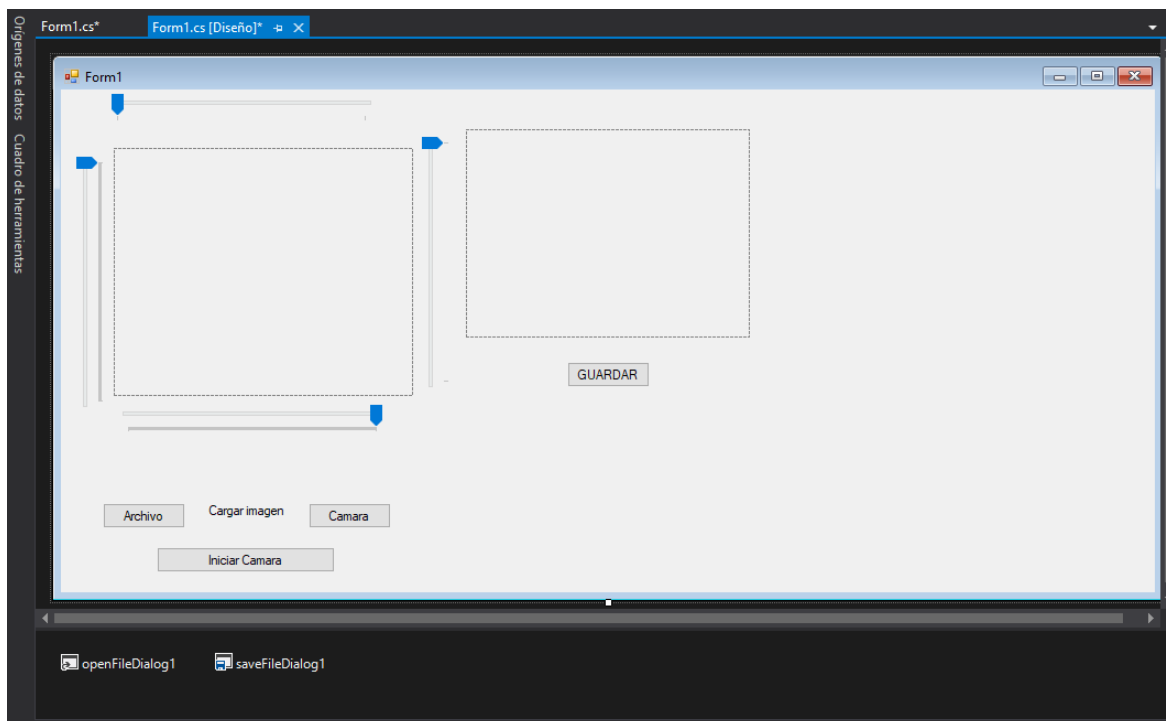


Figura 21

Agregados los controles, edite algunos atributos de los objetos gráficos:

Propiedad	trackBar (izquierdo)	trackBar (derecho)	trackBar (Superior)	trackBar (Inferior)	ImageBox	Button	SaveFileDialog
Name	trackBarROI_Y	trackBarROI_H	trackBarROI_X	trackBarROI_W	imageBoxROI	Guardar_I_ROI	-
AutoSize	False	False	False	False			-
Maximum	480	0	0	640			-
Value	480	0	0	640			-
Orientation	Vertical	Vertical	Horizontal	Horizontal			-
SizeMode	-	-	-	-	StretchImage		-
Text	-	-	-	-	-	Guardar	-

Luego haga doble click en el *trackBar* izquierdo, *trackBar* derecho, *trackBar* superior y *trackBar* inferior para poder programar su función con el siguiente código:

```
1 referencia
private void ttrackBarROI_Y_Scroll(object sender, EventArgs e)
{
    ROI();
}

1 referencia
private void ttrackBarROI_H_Scroll(object sender, EventArgs e)
{
    ROI();
}

1 referencia
private void ttrackBarROI_X_Scroll(object sender, EventArgs e)
{
    ROI();
}

1 referencia
private void ttrackBarROI_W_Scroll(object sender, EventArgs e)
{
    ROI();
}
```

Figura 22

Luego hacer doble click en el botón *Guardar*, para poder programar su función con el siguiente código:

```
1 referencia
private void Guardar_I_ROI_Click(object sender, EventArgs e)
{
    saveFileDialog1.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
    saveFileDialog1.FileName = "guardar imagen";
    saveFileDialog1.Filter = "JPG (*.jpg) |*.jpg |All files (*.*)|*.*";
    saveFileDialog1.FilterIndex = 2;
    saveFileDialog1.RestoreDirectory = true;
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        save_img = saveFileDialog1.FileName;
        image_ROI.Save(save_img);
    }
}
```

Figura 23

Nota: ROI() es la llamada a la función para elegir el área de interés. Esta será definida más adelante.

Búsqueda de formas

Antes de hacer un análisis de patrones, se debe realizar una búsqueda de formas (figuras) las cuales posteriormente serán comparadas con el patrón en cuestión. Para esto se utilizan controles *trackBar* los cuales determinan el umbral superior e inferior para cada capa de color (Rojo, Verde y Azul). Además, se incluye un *ImageBox*, un *Button*, tres *label* y un *RichTextBox* (Figura 24).

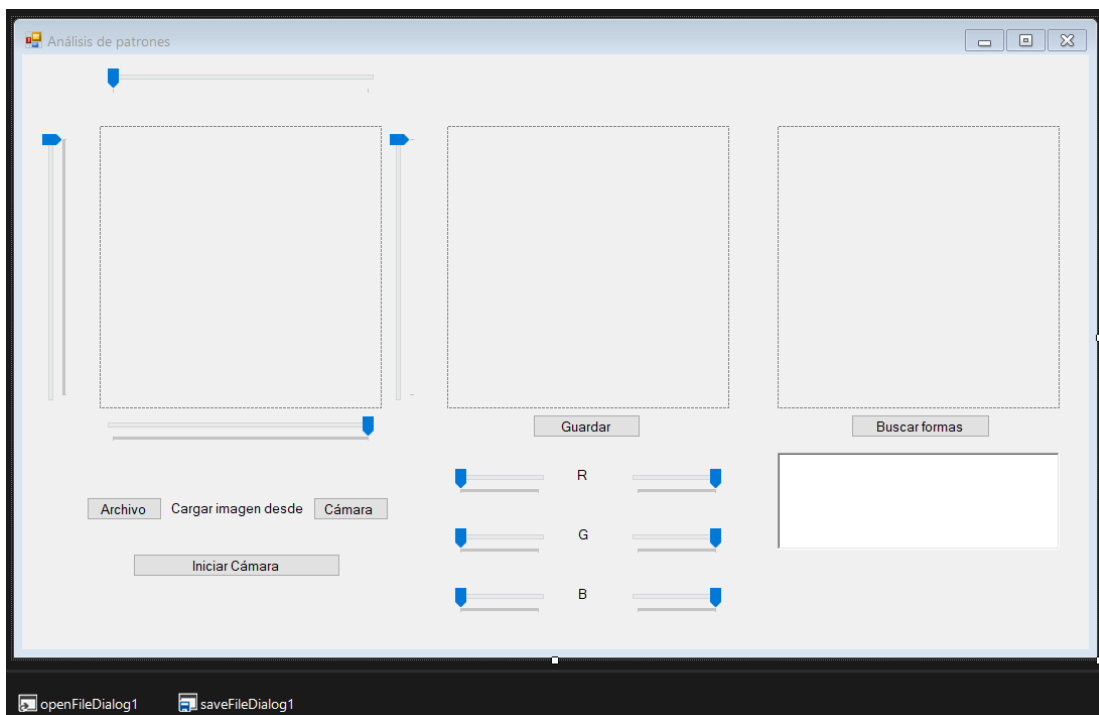


Figura 24

Agregados los controles, edite algunos atributos de los objetos gráficos:

Propiedad	trackBar	trackBar	trackBar	trackBar	trackBar	trackBar	ImageBox	Button	Label	Label	Label	RichTextBox
Name	r_min	r_max	g_min	g_max	b_min	b_max	imageBox_Filtro	button_Busqueda_F_Click	label_r	label_g	label_b	richTextBox_Analisis_M
AutoSize	True	True	True	True	True	True			-			
Maximum	255	255	255	255	255	255			-			
Value	0	255	0	255	0	255			-			
Orientación	Horizontal	Horizontal	Horizontal	Horizontal	Horizontal	Horizontal			-			
SizeMode	-	-					StretchImage		-			
Text	-	-					-	Buscar formas	R	G	B	



Luego hacer doble click en los *trackBar* y agregue el siguiente código de las figuras 25, 26 y 27:

```
1 referencia
private void r_min_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}

1 referencia
private void r_max_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}
```

Figura 25

```
1 referencia
private void g_min_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}

1 referencia
private void g_max_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}
```

Figura 26



```
1 referencia
private void b_min_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}

1 referencia
private void b_max_Scroll(object sender, EventArgs e)
{
    imgFiltrada = image_ROI.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
    imgFiltrada._Dilate(4);
    imgFiltrada._Erode(4);
    imageBox_Filtro.Image = imgFiltrada.Not();
}
```

Figura 27

Después haga doble click en el botón “Buscar Formas”, agregue el siguiente código de la figura 28:

```
1 referencia
private void button_Busqueda_F_Click(object sender, EventArgs e)
{
    BL = GetBlobs(imgFiltrada, 500, imgFiltrada.Rows * imgFiltrada.Cols); // BL guarda la la cantidad de manchas

    Image<Bgr, Byte> image_ROI_Formas = image_ROI.Clone(); ;

    if (BL != null)
    {
        for (int i = 0; i < BL.Length; i++)
        {
            image_ROI_Formas.Draw(BL[i].Rectangle, new Bgr(Color.Red), 1);
        }
    }
    else
    {
        MessageBox.Show("No se encontraron formas");
    }

    richTextBox_Busqueda_F.Text = "Formas Encontradas: " + Convert.ToString(BL.Length);
    imageBoxROI.Image = image_ROI_Formas;
}
```

Figura 28

Análisis de patrones

Para esto se utilizan controles ImageBox, un Button, dos TextBox, dos label y un RichTextBox (ver figura 29).

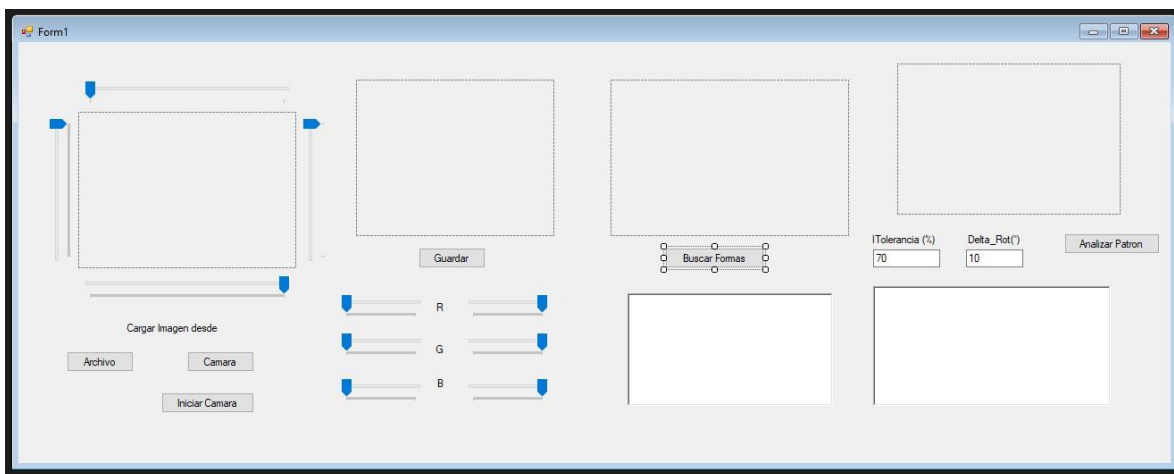


Figura 29

Agregados los controles, edite algunos atributos de los objetos gráficos:

Propiedad	ImageBox	Button	Label	Label	TextBox	TextBox	RichTextBox
Name	imageBox_patron	button_Analisis_P	label_t	label_d_r	porcentaje_coin	Delta_Rot	richTextBox_Analisis_P
AutoSize			-				
Maximum			-				
Value			-				
Orientation			-				
SizeMode	StretchImage		-				
Text	-	Analizar patrón	Tolerancia (%)	Delta_Rot(°)	70	10	

Luego haga doble click en el botón “Analizar patrón” y agregue el código de las figuras 30, 31 y 32 (todas las imágenes pertenecen al mismo código).



```
private void button_Analisis_P_Click(object sender, EventArgs e) //analizar patron
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK) ;
    {
        choose_patron = openFileDialog1.FileName;

        Image<Bgr, byte> image_patron = new Image<Bgr, byte>(choose_patron);
        imageBox_patron.Image = image_patron;
        Image<Bgr, byte> image_ROI_patron = image_ROI.Clone();
        Image<Gray, byte> image_BP_filtrada = imgFiltrada.Clone();
        Image<Gray, byte> image_patron_filtrada = image_patron.InRange(new Bgr(b_min.Value, g_min.Value, r_min.Value), new Bgr(b_max.Value, g_max.Value, r_max.Value));
        image_patron_filtrada.Dilate(4);
        image_patron_filtrada.Erode(4);
        BL_P = GetBlobs(image_patron_filtrada, 500, image_patron_filtrada.Rows * image_patron_filtrada.Cols); //BL guarda la cantidad de manchas
        image_patron_filtrada.ROI = BL_P[0].Rectangle;
        Image<Gray, byte> image_patron_filtrada_Rot = image_patron_filtrada.Clone();
        List<double> correlaciones = new List<double>();
        int ang_rot = Convert.ToInt32(Delta_Rot.Text);
        double coin_max = 0;

        Image<Gray, float> result;
```

Figura 30

```
for (int i = 0; i < BL.Length; i++)
{
    image_BP_filtrada.ROI = BL[i].Rectangle;
    image_patron_filtrada_Rot = image_patron_filtrada_Rot.Resize(image_BP_filtrada.Width, image_BP_filtrada.Height, INTER.CV_INTER_NN);

    for (int delt_ang = 0; delt_ang <= 360; delt_ang += delt_ang + ang_rot)
    {
        image_patron_filtrada_Rot = image_patron_filtrada_Rot.Rotate(Convert.ToDouble(delt_ang), new Gray(0), false);
        BL_P = GetBlobs(image_patron_filtrada_Rot, 500, image_patron_filtrada.Rows * image_patron_filtrada.Cols);
        image_patron_filtrada_Rot.ROI = BL_P[0].Rectangle;
        image_patron_filtrada_Rot = image_patron_filtrada_Rot.Resize(image_BP_filtrada.Width, image_BP_filtrada.Height, INTER.CV_INTER_NN);
        result = image_BP_filtrada.MatchTemplate(image_patron_filtrada_Rot, Emgu.CV.CvEnum.TM_TYPE.CV_TM_CCORR_NORMED);

        for (int x = 0; x < result.Data.GetLength(0); x++)
        {
            for (int y = 0; y < result.Data.GetLength(0); y++)
            {
                correlaciones.Add(result.Data[x, y, 0]);

                if (result.Data[x, y, 0] > Convert.ToDouble(porcentaje_coin.Text) / 100)
                {
                    image_ROI_patron.Draw(BL[i].Rectangle, new Bgr(Color.Blue), 1);
                }
            }
        }

        if (correlaciones.Max() > coin_max)
        {
            coin_max = correlaciones.Max();
        }
        correlaciones.Clear();
        image_BP_filtrada.ROI = Rectangle.Empty;
    }
    image_patron_filtrada.ROI = Rectangle.Empty;
```

Figura 31

```
image_patron_filtrada.ROI = Rectangle.Empty;
if (coin_max > Convert.ToDouble(porcentaje_coin.Text) / 100)
{
    richTextBox_Analisis_P.Text = "Se encontraron coincidencias con un: " + Convert.ToString(Math.Round(coin_max * 100, 2)) + "%";
    imageBoxROI.Image = image_ROI_patron;
}
else
{
    richTextBox_Analisis_P.Text = "No se encontraron coincidencias";
    imageBoxROI.Image = image_ROI_patron;
}
}
```

Figura 32

Crear funciones necesarias

Cree la función “ROI”, de acuerdo con la figura 33.

```
6 referencias
public void ROI()
{
    v_ROI_X = this.trackBarROI_X.Value;
    v_ROI_W = this.trackBarROI_W.Value - this.trackBarROI_X.Value;
    v_ROI_Y = 480 - this.trackBarROI_Y.Value;
    v_ROI_H = this.trackBarROI_Y.Value - this.trackBarROI_H.Value;
    imagen_Capturada.ROI = new Rectangle(v_ROI_X, v_ROI_Y, v_ROI_W, v_ROI_H);
    image_ROI = imagen_Capturada.Copy();
    imageBoxROI.Image = image_ROI;
    imagen_Capturada.ROI = Rectangle.Empty;
}
```

Figura 33

Crear función “ProcesarFrame”, de acuerdo con la figura 34.

```
3 referencias
private void ProcesarFrame(object sender, EventArgs arg)
{
    _capture.ImageGrabbed -= ProcesarFrame;
    frame = _capture.RetrieveBgrFrame(); // foto capturada en RGB
    imagen_Capturada = frame.Resize(640, 480, Emgu.CV.CvEnum.INTER.CV_INTER_NN);
    imageBoxEntrada1.Image = imagen_Capturada;
    _capture.ImageGrabbed += ProcesarFrame;
    ROI(); // para seleccionar el área de interés
}
```

Figura 34



Crear función “GetBlobs”, de acuerdo con las figuras

```
1 referencia
public BlobPM[] GetBlobs(Image<Gray, Byte> Src, int MinArea, int MaxArea)
{
    Image<Gray, byte> mask = Src.CopyBlank();

    int count = 0;
    for (var contours = Src.FindContours(CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_SIMPLE,
    RETR_TYPE.CV_RETR_EXTERNAL); contours != null; contours = contours.HNext)
    {
        if ((contours.Area > MinArea) && (contours.Area < MaxArea))
        {
            count++;
        }
    }
    ///////////////////////////////////
}
```

Figura 35

```
if (count > 0)
{
    BlobPM[] Blob_Res = new BlobPM[count];
    Byte Label = 0;
    count = 0;
    for (var contours2 = Src.FindContours(CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_SIMPLE,
    RETR_TYPE.CV_RETR_EXTERNAL); contours2 != null; contours2 = contours2.HNext)
    {
        if ((contours2.Area > MinArea) && (contours2.Area < MaxArea))
        {
            mask = Src.CopyBlank();
            mask.Draw(contours2, new Gray(255), -1);
            CvInvoke.cvSetImageROI(mask, contours2.BoundingRectangle);
            Blob_Res[count].Mask = mask;
            Blob_Res[count].Contour = contours2;
            Blob_Res[count].Area = contours2.Area;
            Blob_Res[count].Perimetro = contours2.Perimeter;
            Blob_Res[count].Rectangle = contours2.BoundingRectangle;
            Blob_Res[count].ID = Label;
            Label++;
            count++;
        }
    }
}
```

Figura 36



```
if (Blob_Res != null)
{
    for (int i = 0; i < Blob_Res.Length; i++)
    {
        Blob_Res[i].Moments = Blob_Res[i].Mask.GetMoments(true);
        Blob_Res[i].HuMoments = Blob_Res[i].Moments.GetHuMoment();
        Blob_Res[i].angle =
            (float)(0.5 * Math.Atan2(2 * Blob_Res[i].Moments.mu11,
            (Blob_Res[i].Moments.mu20 - Blob_Res[i].Moments.mu02)));
        Matrix<float> HU = new Matrix<float>(1, 7);
        HU.Data[0, 0] = (float)Blob_Res[i].HuMoments.hu1;
        HU.Data[0, 1] = (float)Blob_Res[i].HuMoments.hu2;
        HU.Data[0, 2] = (float)Blob_Res[i].HuMoments.hu3;
        HU.Data[0, 3] = (float)Blob_Res[i].HuMoments.hu4;
        HU.Data[0, 4] = (float)Blob_Res[i].HuMoments.hu5;
        HU.Data[0, 5] = (float)Blob_Res[i].HuMoments.hu6;
        HU.Data[0, 6] = (float)Blob_Res[i].HuMoments.hu7;
    }

    return Blob_Res;
}
else
    return null;
}
```

Figura 37



Crear variables y estructuras globales

La creación de variables y estructuras globales deben ser declaradas dentro de la clase Form1 (*public partial class Form1: Form*), tal como se indica en las figuras 38 y 39.

```
3 referencias
public partial class Form1 : Form
{
    string choose_img;
    string choose_patron;
    string save_img;
    Image<Bgr, Byte> frame;
    Image<Bgr, Byte> imagen_Capturada = new Image<Bgr, Byte>(640, 480); // Imagen base
    private Capture _capture = null;
    private bool _captureInProgress;
    Image<Bgr, Byte> image_ROI;

    Image<Gray, Byte> imgFiltrada;
    int v_ROI_X = 0; //posición del inicial en x del ROI en
    int v_ROI_Y = 0; //posición del inicial en y del ROI
    int v_ROI_W = 640; // ancho del roid
    int v_ROI_H = 480;

    BlobPM[] BL;
    BlobPM[] BL_P;
}
```

Figura 38

```
5 referencias
public struct BlobPM
{
    2 referencias
    public Image<Gray, byte> Mask { get; set; }
    1 referencia
    public Contour<Point> Contour { get; set; }
    6 referencias
    public Rectangle Rectangle { get; set; }
    0 referencias
    public Point Centroid { get; set; }
    1 referencia
    public int ID { get; set; }
    1 referencia
    public double Area { get; set; }
    1 referencia
    public double Perimetro { get; set; }
    5 referencias
    public MCvMoments Moments { get; set; }
    8 referencias
    public MCvHuMoments HuMoments { get; set; }
    0 referencias
    public float response { get; set; }
    1 referencia
    public float angle { get; set; }
}
```

Figura 39



Comunicación entre subprocessos

Debido a que la función “ProcesarFrame” trabaja en un proceso distinto, se debe quitar la alerta de comunicación entre procesos. Esto se hace escribiendo la siguiente línea de código (figura 40) en la función principal “Form1()”.

```
1 referencia  
public Form1()  
{  
    InitializeComponent();  
  
    CheckForIllegalCrossThreadCalls = false;  
}
```

Figura 40

Antes de Ejecutar

Debemos configurar la opción de depuración para nuestra versión de Emgucv. Haga click derecho en nuestro proyecto y elija la opción *Propiedades* (ver figura 41).

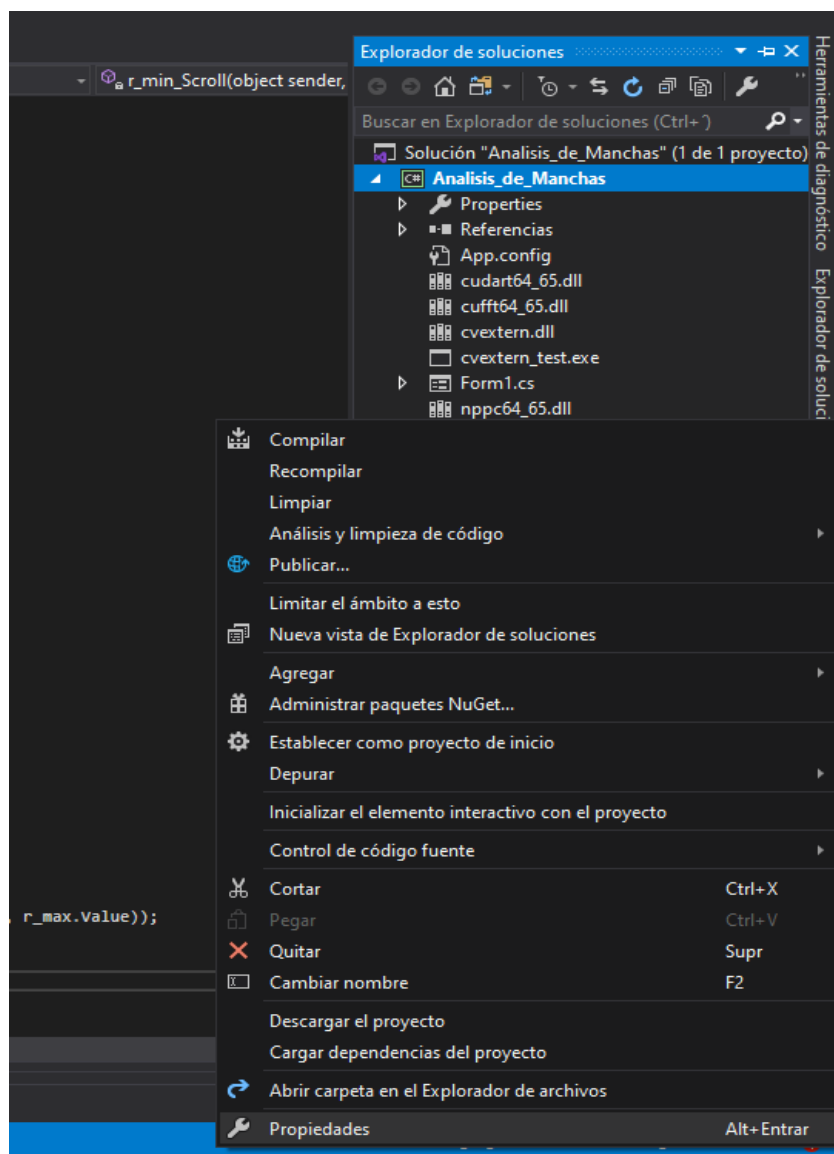


Figura 41

Luego en *Compilación* -> *Plataforma de destino* (figura 42) cambiamos “Any CPU” por “x64”, guardamos y ejecutamos.

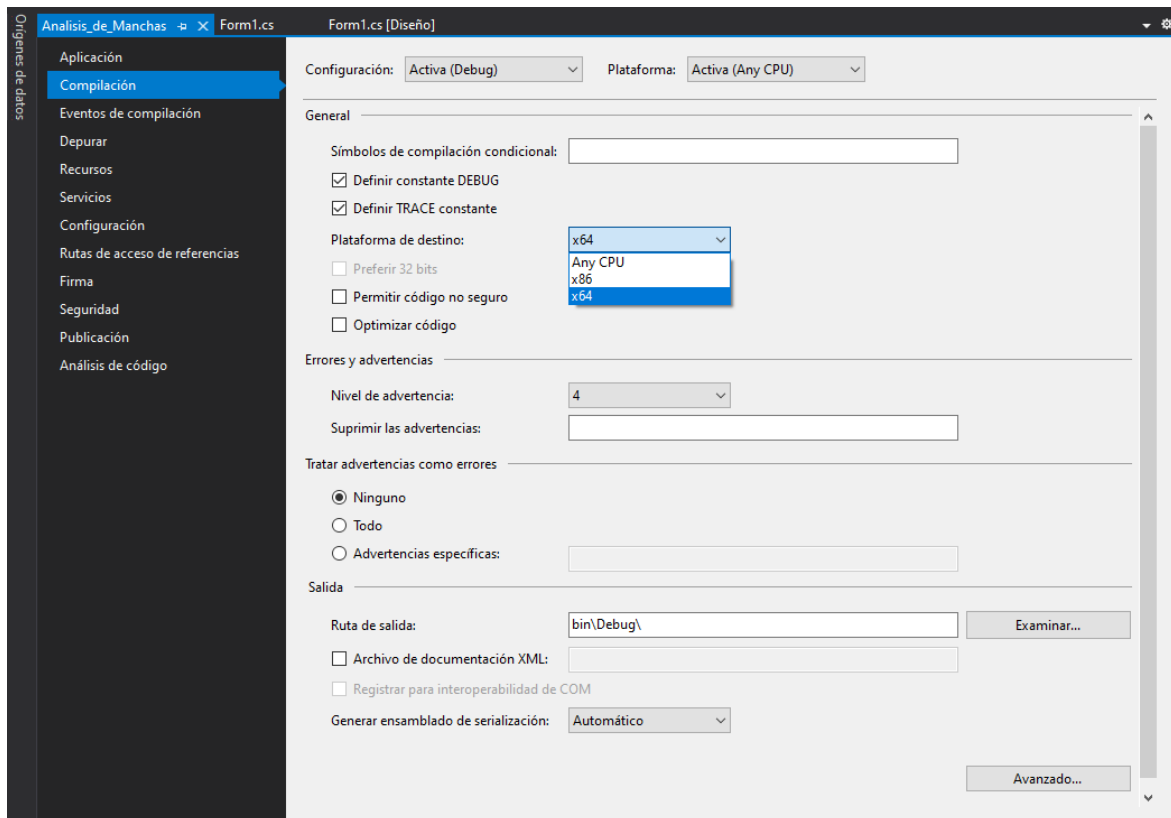


Figura 42

Ejecutar la aplicación

Ahora es posible ejecutar la aplicación. Seleccione una imagen en formato JPG. Un ejemplo del funcionamiento se muestra en la figura 43.

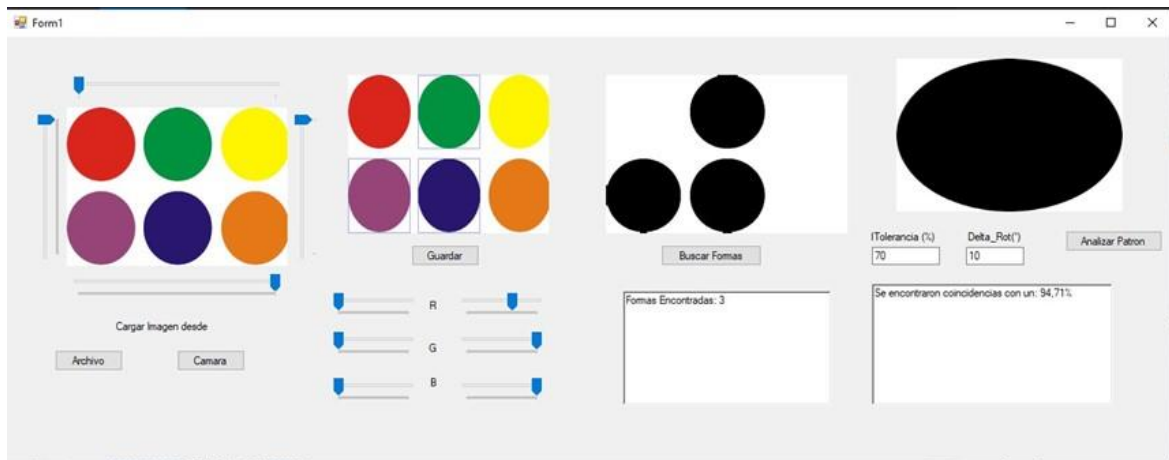


Figura 43