Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: Camilo Esteban Zapata O.
Corregido por (alumno ayudante): José I. Veloso Inzunza

# Umbralización

Umbralizar una imagen consiste en convertirla a binaria, es decir, con sólo dos posibilidades de valores (blanco o negro) y recortar cierto rango de valores. Es ampliamente utilizada para segmentar una imagen, logrando separarse esta de un fondo.

La umbralización es una técnica de segmentación ampliamente utilizada en las aplicaciones industriales. Se emplea cuando hay una clara diferencia entre los objetos a extraer respecto del fondo de la escena. Los principios que rigen son la similitud entre los píxeles pertenecientes a un objeto y sus diferencias respecto al resto. Por tanto, la escena debe caracterizarse por un fondo uniforme y por objetos parecidos.

## Desarrollo

Lea la guía con atención antes de escribir el código.

1. Cree un nuevo proyecto de Windows Forms en lenguaje C#, guiándose por la figura 1.
    a. En la ventana de diseño agregue dos botones:
        i. “Cargar Imagen”
        ii. “Umbralizar”.
    b. Incorpore además 2 *PictureBox*, de tamaño adecuado para visualizar la imagen, y modifique la propiedad SizeMode a “Zoom”.
    c. Añada por último un *NumericUpDown* y modifique en propiedades, sección Datos, la opción *Minimun* y *Maximun* con 0 y 255 respectivamente.

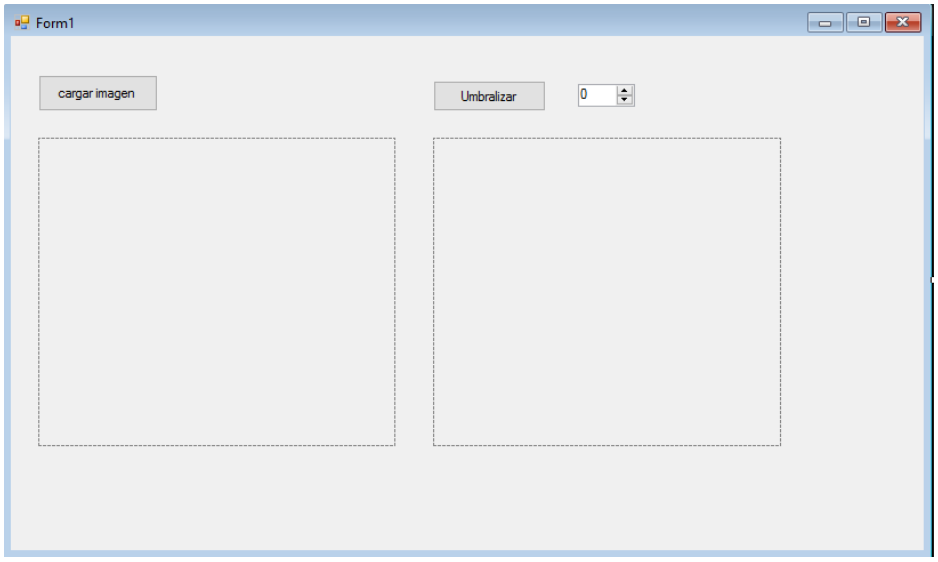


Figura 1

2. Defina una variable global de tipo *string* llamada Ruta para almacenar la dirección de la imagen con la cual se trabajará. Para que la dirección pueda ser almacenada completa se le dará un largo elevado al *string*.

```
public partial class Form1 : Form
{
    String[] Ruta = new string[10000];

    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }
}
```

3. En el botón “Cargar Imagen” cree una variable de tipo *OpenFileDialog* llamada abrir.
    a. Acceda al método *ShowDialog().*
    b. Guarde en el *string* declarado en el paso 2. La propiedad *FileName* convertida a *string*.
    c. Genere un *Bitmap*, a partir de la imagen señalada en dicha dirección.
    d. Imprima en un *PictureBox* la imagen.

```
private void button1_Click(object sender, EventArgs e)
{
        OpenFileDialog abrir = new OpenFileDialog();
        abrir.ShowDialog();
        Ruta[0] = abrir.FileName.ToString();
        Bitmap Imagen = new Bitmap(Ruta[0]);          //Creo Bitmap

        //--------------------Imprimo Imagen real----------------------

        pictureBox1.Image = Imagen;
}
```

4. En el botón “Umbralizar.
    a. Cree un *Bitmap* nuevamente de la imagen con la ruta especificada en el paso 3.
    b. Defina una matriz de ancho y alto del tamaño del *Bitmap* creado mediante las propiedades *Width* y *Height* para guardar los valores de intensidades de los pixeles de la imagen.
    c. Defina un array de tamaño 256 para almacenar histograma.
    d. Comience el recorrido de la matriz.
    e. En una variable de tipo *Color* guarde el dato del pixel leído con el método *GetPixel*.
    f. En una variable de tipo *int*, guarde la conversión de los valores de intensidades en RGB a escala de grises (como se vio en la experiencia anterior).
    g. Almacene el dato de intensidad en el valor de histograma correspondiente.

```
private void button2_Click(object sender, EventArgs e)
{
    Bitmap Imagen = new Bitmap(Ruta[0]);
    //--------------Definiciones-----------------------------------------------------

    int[,] matriz = new int[Imagen.Width, Imagen.Height]; // Defino matriz para guardar valores de pixeles obtenidos
    int[] histograma = new int[256];                      //Defino histograma

    // -----------------Comienza recorrido de la matriz de la imagen-------------------------------------------------

    for (int f = 0; f < Imagen.Width; f++)                //Empiezo a recorrer matriz de imagen
    {
        for (int c = 0; c < Imagen.Height; c++)
        {
            Color pixel = Imagen.GetPixel(f, c);          //Extraigo información por pixel
            int Grisescala = (pixel.R * 59 + pixel.G * 30 + pixel.B * 11) / 100;//Convierto a escala de grises
            matriz[f, c] = Grisescala;                   //Guardo el valor de pixel en escala de grises en matriz
            histograma[Grisescala] = histograma[Grisescala] + 1;
        }
    }
}
```

5. Para generar la imagen ecualizada es necesario tomar el valor ingresado en el *NumericUpDown* como límite. Es decir, los valores inferiores a este serán 0 y los superiores 255, dejando la imagen en sólo dos niveles.
    a. Cree un *bitmap* con las mismas dimensiones de la imagen original.
    b. Defina una variable *color* para almacenar el valor que guardaremos como umbralizado.
    c. Defina una variable de tipo *int* para almacenar el valor ingresado. Ingrese a dicho valor mediante la propiedad *Value* del objeto *NumericUpDown* y conviértalo a entero.
    d. A través de un condicional *if*, consulte si el valor de pixel leído es menor al valor umbral. En dicho caso guarde un nuevo valor en dicho pixel como 0. En caso contrario guárdelo como 255.
    e. Cree el color con el valor de intensidad guardado, a través del método *FromArgb* de la clase Color.
    f. Guarde el color en el pixel presente con el método *setpixel* del *Bitmap* creado para almacenar la imagen umbralizada.

```
Bitmap bin = new Bitmap(Imagen.Width, Imagen.Height);
Color bn = new Color();
```

```
    int numero = Convert.ToInt32(numericUpDown1.Value);
    if (matriz[f, c] < numero)
    {
        matriz[f, c] = 0;
    }
    else
    {
        matriz[f, c] = 255;
    }
    bn = Color.FromArgb(matriz[f, c], matriz[f, c], matriz[f, c]);
    bin.SetPixel(f, c, bn);
}
```

6. Finalmente imprima en el segundo *PictureBox* el *Bitmap* umbralizado.

```
        bn = Color.FromArgb(matriz[f, c], matriz[f, c], matriz[f, c]);
        bin.SetPixel(f, c, bn);
    }
}
pictureBox2.Image = bin;
```

El resultado debería poder separar objetos del fondo, con un correcto valor de umbralización, como se muestra a continuación.