INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

“Practica 5 – Técnicas de Umbralado”

-Tinoco Videgaray Sergio Ernesto

Grupo: 5BV1

Materia: Visión Artificial

14/11/22

- Introducción

Cuando se necesita procesar una imagen es necesario realizar un preprocesamiento para resaltar los elementos que sean de nuestro interés, por ejemplo, si necesitamos procesar una imagen que tiene ciertos elementos como figuras, nuestro principal objetivo es resaltar esas figuras para que posteriormente podamos analizarlas a fondo en una etapa posterior.

Una técnica para resaltar estos elementos es realizando una binzarizacion, la cual consiste en dividir nuestra imagen en 2 elementos: el fondo y el objeto de interés. Esto se consigue truncando los valores a blanco donde los niveles de intensidad sean superiores a nuestro umbral y a 0 los que sean inferiores. Sin embargo, no es fácil determinar qué nivel de umbral es el adecuado para la imagen ya que podemos tener valores que sean muy variados en nuestro objeto de interés.

Para ello se han desarrollado múltiples técnicas que nos permiten calcular el umbral adecuado para nuestra imagen y así poder resaltar nuestro objeto de interés.

Estas técnicas se les conoce como “Técnicas de Umbralado” y se basan en algoritmos iterativos que van a ajustar un nivel de umbral que divida mejor nuestra imagen.

Algunas tecnicas conocidas son:

- Metodo de ISODATA.
- Metodo del valor entre picos.
- Metodo de Otsu.

En esta practica se van a implementar los 3 métodos de Umbralizado.

- Desarrollo

Se importan las bibliotecas correspondientes.

```
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <iostream>
```

Se define una función para convertir la imagen en escala de grises.

```
Mat convertirGrises(Mat imagen,int filas,int columnas)
{
    Mat imagenG(filas, columnas, CV_8UC1);
 //Obtiene la matriz de niveles de gris de la imagen
    cvtColor(imagen, imagenG, COLOR_BGR2GRAY);

    return imagenG;
}
```

Se define la función para obtener el histograma de la imagen.

```
int* obtenerHistograma(Mat imagen,int filas, int columnas)
{
    int k;
    int* histograma = new int[256]();
    for (int i = 0;i < filas;i++)
    {
        for (int j = 0;j < columnas;j++)
        {
           //Obtengo el nivel de intensidad
            k = imagen.at<uchar>(Point(j, i));
              histograma[k]++;
         }
     }

    return histograma;
}
```

Posteriormente se programa la función para aplicar la técnica de IsoData:

```
int IsoData(Mat imagen, int T, int filas, int columnas)
{
    float g1 = 0,g2=0,m1,m2;
    float T2=(float)T;
    int* hist;
    //Particionar imagen
    hist = obtenerHistograma(imagen, filas, columnas);

    for (int i = 0;i < 1;i++)
    {

        g1 = 0;
        g2 = 0;
        for (int k = 0;k <= T;k++)
        {
            g1 += hist[k];
        }

        for (int k = T + 1;k <= 255;k++)
        {
            g2 += hist[k];
        }


        m1 = (g1) / (T + 1);
        m2 = (g2) / (255 - T);


        T2 = (m1 + m2) / 2;

    }
     return T2;
}
```

Se programa la función para aplicar la técnica de valor entre picos:

```
float ValorEntrePicos(Mat imagen, int filas, int columnas)
{
    float g1 = 0, g2 = 0, m1, m2;
    float T = 0;
    int* hist;
    int max = -1,d=0,k=0;
    //Particionar imagen
    hist = obtenerHistograma(imagen, filas, columnas);


    for (int i = 0;i < 254;i++)
    {
        d = pow((i - 255), 2) * hist[i];
        if (d > max)
        {
            max = d;        //Actualizo el valor máximo
            k = i;   //Actualizo el nivel del valor más alto
        }
    }

    T = ((k - 255) / 2) + 255;
    return T;
}
```

Se declara la función para aplicar la técnica de Otsu:

```
float Otsu(Mat imagen, int filas, int columnas)
{
    float gb = 0, gf = 0,wb=0,wf=0, mb=0, mf=0, ob=0, of=0,
Ow=0,Omin=INFINITY;
    float T = 0;
    int* hist;
    int tam=filas*columnas;
    //Particionar imagen
    hist = obtenerHistograma(imagen, filas, columnas);
```

```
    for (int i = 0;i < 254;i++)
    {
        //Calculo gb y gf
        T = i;
        gb = 0;gf = 0;
        for (int j = 0;j<=T;j++)
        {
            gb += hist[j];
        }

        for (int j = T+1;j <=255;j++)
        {
            gf += hist[j];
        }

        //Calculo wb y wf
        wb = 0; wf = 0;
        for (int j = 0;j <= T;j++)
        {
            wb += hist[j];
        }
        wb /= tam;

        for (int j = T + 1;j <= 255;j++)
        {
            wf += hist[j];
        }
        wf /= tam;

        //Calculo mb y mf
        mb = 0; mf = 0;
        for (int j = 0;j <= T;j++)
        {
            mb += j*hist[j];
        }
        mb /= gb;

        for (int j = T + 1;j <= 255;j++)
        {
```

```
            mf += j * hist[j];
        }
        mf /= gf;


        //Calculo Ob y Of
        ob = 0; of = 0;
        for (int j = 0;j <= T;j++)
        {
            ob +=pow(j-mb,2)*hist[j];
        }
        ob /= gb;

        for (int j = T+1;j <= 255;j++)
        {
            of += pow(j - mf, 2) * hist[j];
        }
        of /= gf;

        //Varianza Ow
        Ow = (wb * ob) + (wf * of);
        if (Ow < Omin)
        {
            Omin = Ow;
        }
    }

    return Omin;
}
```

Se declara la función para binarizar con los umbrales calculados.

```
Mat Binarizar(Mat imagen, int filas, int columnas,int umbral)
{

    Mat imagenB(filas, columnas, CV_8UC1);  //Objeto Mat para la imagen
en niveles de gris

    for (int i=0;i < filas;i++)
    {
```

```
        for (int j=0;j < columnas;j++)
        {
            if (imagen.at<uchar>(Point(j, i)) >= umbral)
            {
                imagenB.at<uchar>(Point(j, i)) = uchar(255);
            }
            else
            {
                imagenB.at<uchar>(Point(j, i)) = uchar(0);
            }

        }
    }

    return imagenB;
}
```

Se definen las variables en la función principal.

```
    char NombreImagen[] = "gato.png";
    int T,borde;
    float isodata, vpicos, otsu;
    double sigma;
    Mat imagen, imagenGrises,imagenResultante;
```

Se captura el umbral inicial para el IsoData

```
    cout << "Ingrese un umbral inicial\n";
    cin >> T;

    while (T>254 || T<=0)
    {
        cout << "Por favor ingresa un entre 1 y 254\n";
        cin >> T;
    }
```

```
    int filas = imagen.rows;    //Numero de filas
    int columnas = imagen.cols; //Numero de columnas
```

Invoca las funciones correspondientes para los 3 metodos.

```
    imagenGrises = convertirGrises(imagen, filas,columnas);
    isodata= IsoData(imagenGrises, T, filas, columnas);
    vpicos = ValorEntrePicos(imagen,filas,columnas);
    otsu = Otsu(imagen, filas, columnas);
```

Aplico los 3 umbrales a la imagen con la función binarizar.

```
imagenId = Binarizar(imagenGrises, filas, columnas, isodata);
imagenVp = Binarizar(imagenGrises, filas, columnas, vpicos);
imagenOtsu = Binarizar(imagenGrises, filas, columnas, otsu);
```

Muestra la imagen original y los valores calculados en consola.

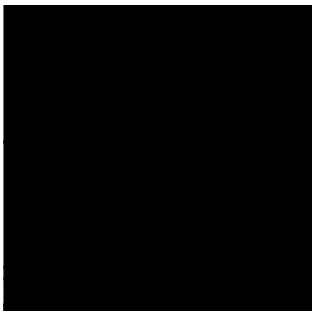
```
    imshow("Imagen Original", imagen);
    imshow("Imagen IsoData", imagenId);
    imshow("Imagen Valor entre Picos", imagenVp);
    imshow("Imagen Otsu", imagenOtsu);
    cout <<"\n\nIsoData: "<< isodata;
    cout << "\n\nValor entre picos: " << vpicos;
    cout << "\n\nOtsu: " << otsu;
```

- Resultados

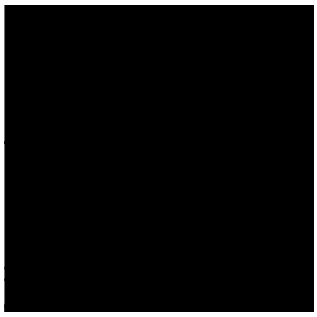
```
IsoData: 364

Valor entre picos: 128

Otsu: 618.074
```

Umbrales calculados.

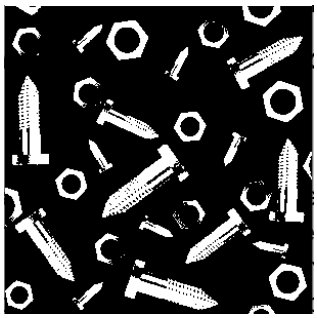
Imagen Original.



Umbralizado con ISODATA



Umbralizado con Valor entre picos

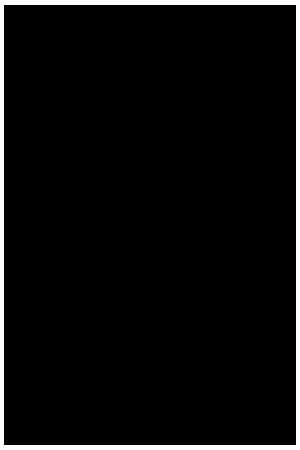


Umbralizado con OTSU

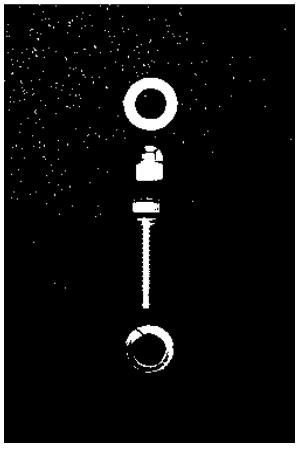
```
IsoData: 537

Valor entre picos: 158

Otsu: 111.984
```

Umbrales calculados.

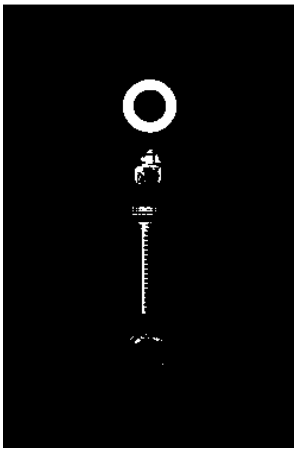
Imagen Original.



Umbralizado con ISODATA



Umbralizado con Valor entre picos



Umbralizado con OTSU

Conclusiones.

Realmente es difícil determinar que algoritmo es mejor para el Umbralizado ya que va a depender de nuestra imagen y el objeto de interés, por lo que considero no siempre van a funcionar para todo tipo de imágenes y van a tener un margen de error. En el caso de esta practica el algoritmo mas eficiente se le atribuye al método de Otsu por sus resultados tan precisos y el menos eficiente considero que fue el ISODATA al tener un margen de error muy elevado.

Referencias.

Serquen, J. A. P. (s. f.). *Segmentación por umbralización método de otsu*.

https://es.slideshare.net/JorgeAntonioParraSerquen/segmentacin-por-umbralizacin-mtodo-de-otsu