Tarea 20

Angel Manrique Pozos Flores; N.C M07211505
Tecnológico Nacional de México,
Blvd. Industrial, Mesa de Otay, 22430 Tijuana, B.C., México.

19 de mayo de 2016

Implementar el algoritmo de crecimiento de regiones visto en clase.

1. Segmentación por crecimiento

Los métodos de segmentación por crecimiento de regiones son técnicas locales que se basan en tomar un pixel o un conjunto de pixeles, como una región inicial denominada "*semilla*ÿ a partir de estas "*crecer* Ïa región agregando pixeles con propiedades similares hasta llegar a ciertos limites.

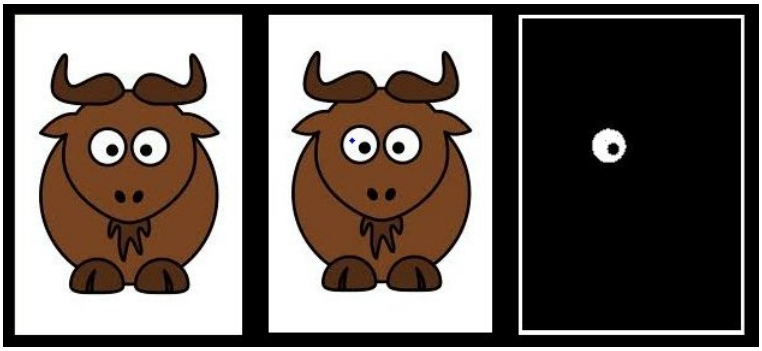

Figura 1: Imagen representativa de la segmentación por crecimiento.

Esta aproximación basada en el crecimiento de regiones, explora el importante hecho de que los pixeles que están cercanos unos a otros presentan ciertas similaridades en sus valor en la escala de grises.

Este método presenta algunas desventajas que lo hacen poco practico en un nivel general, uno de los inconvenientes es seleccionar el criterio adecuado ya que

dependerá de la imagen que se desee segmentar, otro de los inconvenientes es la selección inicial de las semillas ya que sin información a priori de la ubicación aproximada de los objetos en la escena, resulta casi imposible llegar a formular un conjunto inicial de semillas que nos aseguren buenos resultados.

2. Selección de semillas

La selección de las semillas dependerá exclusivamente de la naturaleza del problema, por ejemplo si queremos detectar algún objecto utilizando imágenes infrarrojas, podemos enfocarnos en tomar los pixeles mas brillantes.

Sin un conocimiento previo, podríamos graficar el histograma y escoger los valores de niveles de grises que corresponden a los picos mas grandes.

Otra cosa que debemos tener en consideración es escoger el criterio de similaridad, donde esta homogeneidad, debe ser basada en las características de las regiones en la imagen por ejemplo:

- Intensidad.
- Variancia.
- Color.
- Textura.
- Forma.
- Tamaño.

3. Crecimiento de la región

En el articulo "Segmentation throught variable-order surface fitting"de los autores Besl & Jain. Sugieren una forma de crecimiento basado en un nivel de threshold y una función que checa el error donde:

- Una imagen digital es cuantizada en una cantidad de muestras.
- La idea es ajustar una superficie de menor orden sobre la región de interés.

$$f(x,y,a,m) = \sum_{i+j \leq m} a_{ij} x^i y^j \tag{1}$$

- Si el error es pequeño, entonces podemos concluir que los valores del pixel pertenecen a la misma región.

$$E(R,a,m) = \sum_{(x,y)\in R} [g(x,y) - f(x,y,a,m)]^2 \tag{2}$$

En general podemos decir que este método combina aspectos de umbralamiento y conectividad, donde realiza el agrupamiento de los pixeles utilizando dos criterios, la proximidad y la homogeneidad.

La mayoría de los algoritmos que utilizan el criterio de proximidad efectúan el agrupamiento de pixeles según criterios de división (split) y unión (merge) de pequeñas regiones según las características estadísticas de los niveles de gris.

El otro criterio, la homogeneidad, se implementa mediante una función que cuantifica similitud entre las regiones. Esta función se basa generalmente en una comparación de características estadísticas de intensidades, por ejemplo, dos regiones pueden considerarse homogéneas si sus intensidades mínimas y máximas se encuentran en un rango determinado.

En la mayoría de los casos, la homogeneidad de las regiones se evalúa calculando la probabilidad de que las regiones sean realmente sub-regiones compuestas por pixeles muestreados de una región mas grande, este valor se compara con una probabilidad mínima umbral que considera a las regiones como homegeneas (Haralick, 1992).

Para ello se desarrollo el siguiente código en C++ utilizando las librerías de visión de OpenCV.

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>


int xDim,yDim;
unsigned long total[3];
int coont;
int Diff, mean[3], temp[3];
static int count = 0;
int Threshold = 50;
```

```
IplImage *Image1;
IplImage *Image2;
CvScalar s = {0,0,0,0};
CvScalar s11 = {0,0,0,0};

void GrowColor(int,int);

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

//variables iniciales
int x, y;
int Xseed = 40, Yseed = 20;

total[0] = total[1] = total[2] = coont = 0;

//carga la imagen
Image1 = cvLoadImage("cua2.jpg",
   CV_LOAD_IMAGE_UNCHANGED );
yDim = Image1 -> height;
xDim = Image1 -> width;

//crea el arreglo y la imagen donde iran contenidas
   las semillas
IplImage* img = cvCreateImage(cvGetSize(Image1),
   Image1->depth , Image1->nChannels );
Image2 = cvCreateImage(cvGetSize(Image1), Image1->
   depth , Image1->nChannels );
cvZero( Image2 );


for (y = Yseed - 5; y <= Yseed + 5; y++)
     for (x = Xseed - 5; x <= Xseed + 5; x++)
        if ((x > 0) && (y > 0) && (x < xDim) && (y <
          yDim))
         {
          coont++;
          s = cvGet2D(Image1,y,x);
          total[0] += (s.val[0]);
          total[1] += (s.val[1]);
          total[2] += (s.val[2]);
```

```
            }


//Crece a lo largo de X y Y
for (y = 0; y < yDim; y++)
      for (x = 0; x < xDim; x++)
          GrowColor(y,x);


//imprime la imagen original y la imagen de las
   semillas sobre la mascara
cvNamedWindow( "Original", 1 );
cvShowImage( "Original", Image1 );
cvNamedWindow( "Grow Color", 1 );
cvShowImage( "Grow Color", Image2 );

cvWaitKey(0);
return 0;
}

// funcion de crecimiento
void GrowColor(int y, int x)
{

s.val[0]=0;
s.val[1]=0;
s.val[2]=0;
s.val[3]=0;

if((x < 1)&&(y < 1))

    s = cvGet2D(Image1,y,x);

if(s.val[0]==0)
  {
        s11 = cvGet2D(Image1,y,x);
        mean[0] = total[0] / coont;
        mean[1] = total[1] / coont;
        mean[2] = total[2] / coont;

        temp[0] = abs(s11.val[0] - mean[0]);
```

```
        temp[1] = abs(s11.val[1] - mean[1]);
        temp[2] = abs(s11.val[2] - mean[2]);

        Diff = (int)(sqrt((temp[0]*temp[0]+temp[1]*temp
            [1]+temp[2]*temp[2])/3));

                if (Diff < Threshold)
                {
                total[0] += abs(s11.val[0]);
                total[1] += abs(s11.val[1]);
                total[2] += abs(s11.val[2]);
                coont++;
                if ((x > 0) && (y > 0))
                cvSet2D(Image2,y,x,s11);
                }
    }

}
```

Se utilizo la imagen de la figura 2 para la realización de las pruebas.

Figura 2: Imagen utilizada para las pruebas de segmentación por crecimiento de regiones.

Las salidas que se obtienen se van definiendo en base a un criterio de thresholding que se va definiendo de acuerdo a la región que se quiere encontrar.

(a) Segmentación por semillas para un tono de gris alto.

(b) Segmentación por semillas para un tono de gris alto.

(c) Segmentación para tonos de grises altos o muy brillantes.

Figura 3: Salidas correspondientes a las secciones de niveles de gris utilizando el crecimiento por semillas.

4. Bibliografía

1 A simple región growing implementation in Python.
http://tinyurl.com/zhrzy4e

2 Jain et al., "Region Growing".
http://tinyurl.com/j53r2te

3 J. Anali & A. Ivan., "Diseño de un algoritmo de segmentación de imágenes aplicando el funcional de Mumford-Shah para mejorar el desempeño de algoritmos clásicos de segmentación", Universidad Nacional de Trujillo.

4 Meschino G. & Moler E., "Algoritmo de crecimiento de regiones con característicos de texturas: una aplicación en biopsias de médula osea", Universidad Nacional de Mar del Plata.