**Programa de extracción de características de forma**

**Introducción**

El análisis de imágenes permite hallar diversos significados o descripciones de objetos en una imagen. Descriptores que pueden ser muy importantes para algún caso de estudio en particular. Por ejemplo, el análisis de imágenes de mamografías para la prevención del cáncer de mama, localización de objetos en imágenes satelitales, reconocimiento de rostros, reconocimiento de iris, sistema de control de tráfico, entre otros. Los descriptores que se pueden analizar en una imagen, pueden ser descriptores de color, textura, forma, movimiento y localización. En particular, este trabajo se enfoca a la obtención de características de forma de una región de interés. Las imágenes que se analizan son segmentadas con una sola región de interés, aplicando el método de detección de borde (conectividad-4) y la distancia radial normalizada.

**Forma**

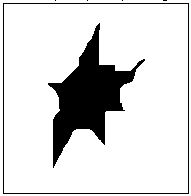
Las características de forma, también se le conoce como características morfológicas o geométricas. La forma es una de las características más importante para la caracterización de un objeto. Sin embargo, la mayoría de las formas se representan por rasgos primitivos uniformes que dificultan la observación de sus propiedades lógicas y estructurales [3].

Para hallar una región inscrita en una imagen, se deben contemplar dos propiedades importantes, la primera es mediante sus características externas (su contorno) y la segunda a través de sus características internas (los píxeles que comprenden la región). Generalmente, cuando se analizan las características de forma, se elige una representación externa. [2, 3].

Algunos métodos para determinar las características de una región son los códigos de cadena, aproximaciones poligonales, descriptores de Fourier, momentos de Hu, la longitud radial normalizada (*normalized radial length*) y la longitud cordal normalizada (*normalized chord length*) [2, 3]. En este trabajo se utiliza el método de longitud radial normalizada.

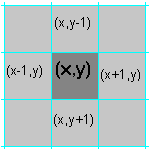
**Conectividad-4**

Se utilizó esta técnica para detectar el contorno de la región de interés. Dado que la imagen es binaria, donde la región de interés se identifica con valores de intensidad 0 (negro) y el fondo con valores de intensidad 255 (blanco), como se muestra en la figura 1.



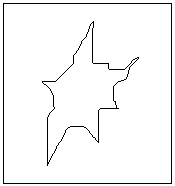
**Figura 1.** Imagen binarizada con la región de interés con intensidad = 0 y el fondo con intensidad = 255.

Con la técnica de conectividad-4 se recorre toda la matriz de la imagen , dónde y son las posiciones de los píxeles. En cada recorrido, se toma un píxel de interés y se verifica sí su pixel vecino (horizontal y vertical) están encendidos (tienen la misma intensidad) (Fig. 2), sí los 4 pixeles vecinos, tanto los horizontales y verticales del píxel de interés están encendidos, entonces se considera como conectividad-4 [4].



**Figura 2.** Dirección del recorrido de la técnica conectividad-4.

Sí el píxel de interés tiene un valor de intensidad = 0 y no es conectividad-4, entonces es un píxel del contorno. Tanto los píxeles del contorno y los píxeles internos del contorno, se almacenan sus posiciones en vectores diferentes. Finalmente se crea una nueva imagen que muestra únicamente el contorno de la región de interés (Fig. 3).

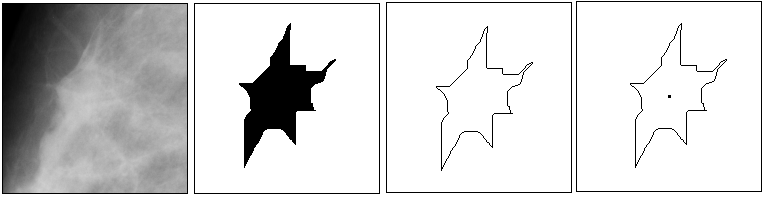


**Figura 3.** Contorno de la figura 1.

**Calculando el centro de la región de interés**

Después de haber obtenido el contorno de la región de interés, el siguiente paso es calcular el centro de dicha región (coordenada ), utilizando las siguientes fórmulas [3].

Particularmente, para este ejemplo se utilizó una imagen de escala de grises, en la cual se obtuvo una región de interés y se segmentó, para posteriormente calcular su contorno y su centro, como se muestra en la figura 4.

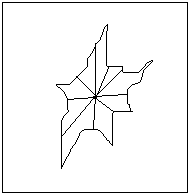


**Figura 4.** De derecha a izquierda, se muestra la imagen original, luego la imagen segmentada, después el contorno de la imagen segmentada y por último el centro del contorno.

Para determinar el centro de la región de interés se usaron los valores de intensidad de la imagen original.

**Calculando la distancia radial normalizada**

Para comenzar a calcular los descriptores de forma, se debe de construir el vector de la distancia radial normalizada. Para construir este vector, se parte del centro de la región de interés calculando la distancia (distancia euclidiana) hacia cada uno de los pixeles del contorno de dicha región (Figura 5) [4, 5].



**Figura 5.** Distancia radial normalizada.

Posteriormente se normaliza el vector de distancia radial, donde cada uno de sus componentes se dividen entre la distancia máxima.

**Extracción de características de forma**

En la tabla I se muestra las características que se calcularon en este trabajo [4, 5].

**Tabla I.** Características de forma (distancia radial normalizada).

|  |  |
| --- | --- |
| Característica | Fórmula |
| Perímetro |  |
| Área | A = Suma de todos los píxeles del interior y el borde de la región de interés. |
| Circulación |  |
| Media de la distancia radial normalizada |  |
| Desviación estándar de la distancia radial normalizada |  |
| Compacidad |  |
| Contraste |  |
| Skewness |  |
| Kurtosis |  |
| Promedio de intensidad dentro de la región de interés (N = número total de píxeles dentro de la región de interés) |  |

**Mini tutorial de Extracción de características de forma en Java**

Se realizó la implementación del cálculo de forma a partir del vector de distancia radial normalizada en Java. Esta implementación produjo algunas interfaces de usuarios que se muestran a continuación.

En la Figura 6 se muestra la ventana principal del programa, la cual contiene cinco botones que se explicarán más a delante. Para comenzar a realizar el cálculo primeramente se debe de seleccionar la imagen segmentada y la imagen original.

Descripción de los botones:

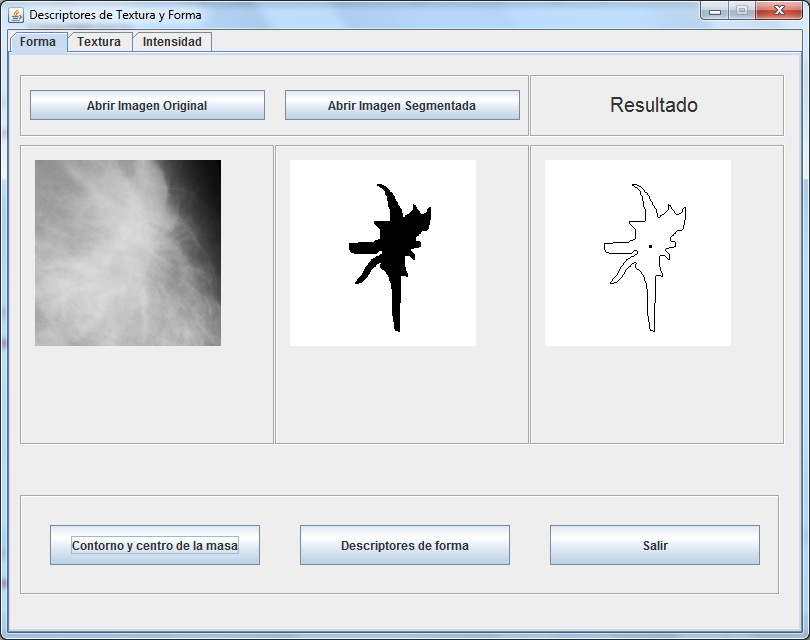
***Abrir imagen Original***: al presionar este botón, se despliega la imagen de la Figura 7.

***Abrir Imagen segmentada***: al presionar este botón, se despliega la imagen de la Figura 7.

***Contorno y centro de la masa***: se calcula el contorno de la región de interés utilizando la técnica de conectividad-4 y se obtiene el centro de la región de interés utilizando la técnica de momentos. El resultado se despliega en la tercera sección de la figura 6.

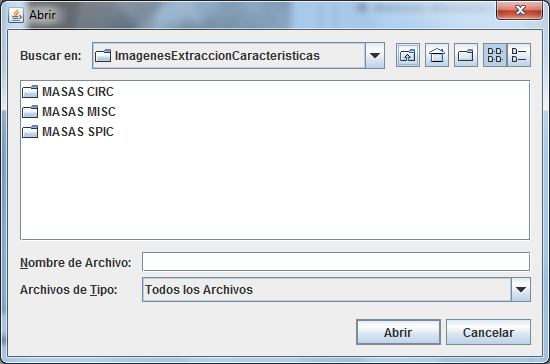
***Descriptores de forma***: se calcula los descriptores de forma y se despliega la pantalla x.

***Salir***: al presionar este botón se cierra el programa.



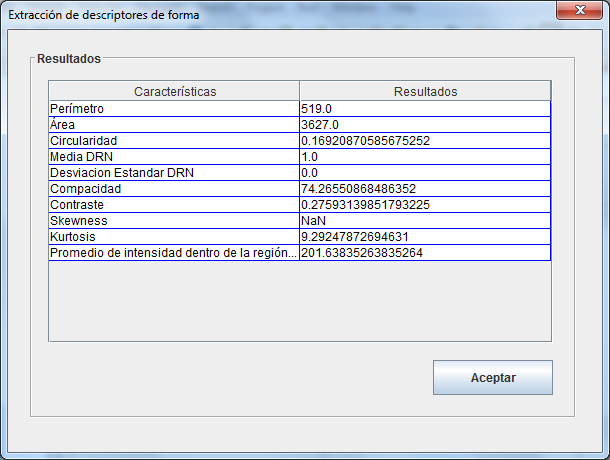
**Figura 6.** Ventana principal para seleccionar la imagen e inicial el cálculo.

En la Figura 7 se muestra la ventana para seleccionar la imagen sobre la cual se realizará el cálculo de extracción de características (imagen segmentada y binarizada). Así mismo, se debe seleccionar la imagen original.



**Figura 7.** Ventana para seleccionar la imagen.

En la Figura 8 se muestra la ventana con una tabla de los valores de las características de forma, y por último el botón ***Aceptar*** conduce nuevamente a la Figura 6.



**Figura 8.** Ventana para mostrar los resultados.

**Descripción del código fuente**

Código para detectar el borde de la región de interés usando el algoritmo de conectividad-4.

**boolean** cuatroVecinos( **int** i, **int** j ){

**int** vecino = 0;

**if**( ( j - 1 ) > -1 )

**if**( MATRIZ\_IMAGEN[ i ][ j - 1] == 0 )

vecino++;

**if**( ( i - 1 ) > -1 )

**if**( MATRIZ\_IMAGEN[ i - 1 ][ j ] == 0 )

vecino++;

**if**( ( j + 1 ) < WIDTH )

**if**( MATRIZ\_IMAGEN[ i ][ j + 1] == 0 )

vecino++;

**if**( ( i + 1 ) < HEIGHT )

**if**( MATRIZ\_IMAGEN[ i + 1 ][ j ] == 0 )

vecino++;

**if**( vecino == 4 )

**return** **true**;

**else**

**return** **false**;

}

Código para recorrer toda la matriz de la imagen y hace la llamada a conectividad-4.

**public** **void** calcularBordeImagen(){

NodoPixel nodoPixel;

**for**( **int** i = 0; i < HEIGHT; i++ ){

**for**( **int** j = 0; j < WIDTH; j++ ){

**if**( MATRIZ\_IMAGEN[ i ][ j ] == 0 ){

nodoPixel = **new** NodoPixel();

**if**( cuatroVecinos( i, j ) ){

nodoPixel.setX( i );

nodoPixel.setY( j );

pixelImagenArea.add( nodoPixel );

}

**else**{

nodoPixel.setPixelPerimetroX( i );

nodoPixel.setPixelPerimetroY( j );

pixelContorno.add( nodoPixel );

}

}

}

}

}

Código para resaltar el borde de la región de interés.

**public** **void** rellenarImagenFondo(){

Iterator< NodoPixel > iterador = pixelImagenArea.iterator();

**while** ( iterador.hasNext() ) {

NodoPixel pos = iterador.next();

MATRIZ\_IMAGEN[ pos.getX() ][ pos.getY() ] = (**byte**)255;

}

contornoPixelcentral();

MATRIZ\_IMAGEN[ ( **int** )**this**.centroX ][ ( **int** )**this**.centroY ] = (**byte**)0;

}

Código para calcular el centro de la masa (se muestran dos funciones).

**public** **double** momentoDeOrden( **int** p, **int** q ){

**double** suma = 0;

**for**( **int** i = 0; i < **this**.HEIGHT; i++ ){

**for**( **int** j = 0; j < **this**.WIDTH; j++ ){

**if**( (**this**.MATRIZ\_IMAGEN[ i ][ j ]& 255) != 255 ){

suma = suma + ( Math.*pow*( i, p ) \* Math.*pow*( j, q ) \* **this**.MATRIZ\_IMAGEN\_ORIGINAL[ i ][ j ] );

}

}

}

**return** suma;

}

**public** **void** momentoCentral(){

xBarra = momentoDeOrden( 1, 0 ) / momentoDeOrden( 0, 0 );

yBarra = momentoDeOrden( 0, 1 ) / momentoDeOrden( 0, 0 );

}

Código para obtener la distancia radial normalizada.

**public** **void** obtenerDistanciaRadialNormalizada(){

Iterator< NodoPixel > iterador = **this**.pixelContorno.iterator();

**int** x = 0;

**while** ( iterador.hasNext() ) {

NodoPixel pos = iterador.next();

vectorDistnaciaRN[ x ] = Math.*sqrt*( Math.*pow*( pos.getX() - centroX, 2) + Math.*pow*( pos.getY() - centroY, 2) );

x++;

}

}

Código para normalizar el vector de distancia radial normalizada.

**public** **void** normalizarVectorDRN(){

**double** mayor = **this**.vectorDistnaciaRN[ 0 ];

**for**( **int** i = 1; i < **this**.pixelContorno.size() ; i++ )

**if**( vectorDistnaciaRN[ i ] > mayor )

mayor = vectorDistnaciaRN[ i ];

**for**( **int** i = 0; i < **this**.pixelContorno.size() ; i++ )

vectorDistnaciaRN[ i ] = vectorDistnaciaRN[ i ] / mayor;

}

Código para obtener los datos estadísticos de la forma.

**public** **void** obtenerCaracteristicasForma(){

perimetro = **this**.pixelContorno.size();

area = **this**.pixelImagenArea.size() + perimetro;

circularidad = ( 4 \* Math.*PI* \* area ) / (Math.*pow*( perimetro, 2 ) );

mediaDRN = calcularMediaDRN();

desviacionEstandarDRN = calcularDesviacionEstandarDRN();

compacidad = Math.*pow*( per86imetro, 2) / area;

contraste = ( PixF - PixB ) / PixF;

}

**public** **double** calcularMediaDRN(){

**double** suma = 0;

**for**(**int** i = 0; i < perimetro; i++ ){

suma = suma + vectorDistnaciaRN[ i ];

}

**return** suma / perimetro;

}

**public** **double** calcularDesviacionEstandarDRN(){

**double** suma = 0;

**for**(**int** i = 0; i < perimetro; i++ ){

suma = suma + ( Math.*pow*( vectorDistnaciaRN[ i ] - mediaDRN, 2 ) );

}

**return** Math.*sqrt*( suma / perimetro );

}

**public** **void** calcularSkewness\_Kurtosis(){

Iterator< NodoPixel > iterador = pixelImagenArea.iterator();

**double** suma1 = 0;

**double** suma2 = 0;

**while** ( iterador.hasNext() ) {

NodoPixel pos = iterador.next();

**int** i = pos.getX();

**int** j = pos.getY();

**double** x = (**this**.MATRIZ\_IMAGEN\_ORIGINAL[ i ][ j ]&255) - promedioNivelGrisRegion;

suma1 += Math.*pow*( x, 3 );

suma2 += Math.*pow*( x, 4 );

}

skewness = suma1 / ( Math.*sqrt*( suma1 ) \* **this**.pixelImagenArea.size() );

kurtosis = suma2 / ( Math.*sqrt*( suma2 ) \* **this**.pixelImagenArea.size() );

}

**Referencias**

[1] N. Bang and K. Um. Structural Analysis and Matching of Shape by Logical Property. In Springer-Verlag, editor, Structural, Syntactic, and Statistical Pattern Recognition, Korea, 2004.

[2] R. C. Gonzalez and R. E. Woods. Tratamiento digital de imágenes. Addison-Wesley Díaz de Santos, Estados Unidos de América, 1992.

[3] N. Mark and A. Aguado. Feature Extraction & Image Processing. Academic, Great Britain, 2002.

[4] Delogu, P., Evelina-Fantacci, M., Kasae, P. and Retico, A. Characterization of mammographic masses using a gradient-based segmentation algorithm and a neural classifier.

[5] Kinnard, L., Shih-Chung, B., Wang, P., Freedman, M. and Chouikha, M. Separation of Malignant and Benign Masses using Maximum-Likelihood Modeling and Neural Networks.