IV. Algoritmos de segmentación evaluados

1. Introducción

La plataforma *ITK* proporciona un conjunto básico de algoritmos pensados para usarse en el desarrollo y particularización de una aplicación completa para la segmentación. *ITK* incluye implementaciones de diversos algoritmos, pertenecientes a diversas familias de técnicas de segmentación. La Tabla 1 enumera las implementaciones de algoritmos de segmentación de *ITK*, dentro de la familia a la que pertenecen (8). Asimismo, en dicha tabla se resaltan los algoritmos evaluados en este trabajo, así como los parámetros de entrada de los mismos, junto con un comentario que resume las causas para su inclusión o exclusión de la evaluación realizada. Para la elección de los algoritmos a evaluar se tuvieron en cuenta los objetivos del proyecto (ver sección Alcance y objetivos). De esta manera, la ausencia de interfaz gráfica junto con la conveniencia, con vistas a la adecuada comparación, de tener un espacio paramétrico con cierta homogeneidad (tanto en la dimensionalidad como en la complejidad, desde el punto de vista del operador), ha configurado el conjunto de algoritmos a ser evaluados.

Técnica segment.	Algoritmo ITK	Eval.	N	Parámetros de entrada	Comentario
Region grow	Connected Threshold	No	4	$Seed_{x,y}$, Lower Threshold, Upper Threshold	Rechazado por mayor conveniencia de otros algoritmos región grow
	Otsu Multiple Threshold	**Si**	1	Número de umbrales a calcular	Escogido por la simplicidad de su espacio paramétrico
	Neighborhood Connected		4	$Seed_{x,y}$, Lower Threshold, Upper Threshold	Rechazado por mayor conveniencia de otros algoritmos región grow
	Confidence Connected	**Si**	2	$Seed_{x,y}$	Escogido como representante típico de los algoritmos tipo región grow
	Isolated Connected	**Si**	5	$SeedIn_{x,y}$, $SeedOut_{x,y}$, LowerThreshold	Seleccionado por la posibilidad de seleccionar seedOut
Water-sheds	Watershed segmentation	No	5	ConductanceTerm, difussionIterations, LowerThreshold, OutputScaleLevel, GradientMode	Espacio paramétrico de mayor complejidad
Level sets	Fast-Marching	No	7	$Seed_{x,y}$, sigma, sigmoidalpha, sigmoidbeta, timethreshold, stoppingvalue	Espacio paramétrico de mayor complejidad
	Shape detection	No	8	$Seed_{x,y}$, InitialDistance, Sigma, SigmoidAlpha, SigmoidBeta, curvatureScaling, propagationScaling	Definido a partir de fast-marching
	Geodesic Active Contours	No	7	$Seed_{x,y}$, InitialDistance, Sigma, SigmoidAlpha, SigmoidBeta, propagationscaling	Especialmente apropiado para GUI Definido a partir de fast-marching
	Threshold LevelSet	**Si**	7	$Seed_{x,y}$, InitialDistance, lowerthreshold, upperthreshold, [curvaturescaling=1.0]	Seleccionado para representar a los algoritmos level-set. Tiene un conjunto de parámetros similar a los otros algoritmos escogidos
	Canny-Edge Level-set	No	9	$Seed_{x,y}$, CannyThreshold, CannyVariance, AdvectionWeight, InitialModelIsovalue, MaximumIterations, [OutputSpeedImage]	Espacio paramétrico de mayor complejidad
	Geodesic Active Contours with shape guidance	No	15	$Seed1_{x,y}$, $Seed2_{x,y}$, $Seed3_{x,y}$, initialDistance, sigma, propagationScaling, shapePriorScaling, meanShapeImage, numberOfModes, shapeModeFilePattern, $Start_{x,y}$.	Espacio paramétrico de mayor complejidad
Híbridos	Fuzzy Connectedness and confidence Connectedness	No	4	outputAffinityMap, seedx, seedy, multiplier	Patentado
	FuzzyConnectedness and Voronoi classification	No	6	$Seed_{x,y}$, estimateMean, estimateVariance, meanTolerance, standardDeviationTolerance	Patentado
	Deformable model and Gibbs Prior	No	--	Requiere entrenamiento previo para su funcionamiento	No adecuado en el contexto del proyecto
Feature extraction	Hough Transform	No	--	--	No adecuado en el contexto del proyecto

Tabla 1: Algoritmos disponibles en ITK

2. Confidence connected

Se trata de uno de los algoritmos de crecimiento de regiones implementado en *ITK*.

Este algoritmo requiere cuatro parámetros de entrada por parte del operador. Dos de ellos son triviales (imagen a segmentar, y nombre del archivo que contendrá la imagen segmentada de salida), y los otros dos son las coordenadas espaciales (x, y) de la semilla. La semilla ha de ser un punto cualquiera del objeto a segmentar.

El criterio del que hace uso el algoritmo *ConfidenceConnected* para la inclusión o exclusión de cada píxel evaluado se basa en estadísticas simples de la región actual. En primer lugar, el algoritmo calcula la media y la desviación estándar de los valores de intensidad de todos los píxeles actualmente incluidos en la región. El punto donde el algoritmo se inicia es la semilla especificada por el usuario. A continuación se usa un factor multiplicador para multiplicar la desviación estándar y definir así un rango alrededor de la media. El valor de este factor multiplicador es proporcionado en tiempo de desarrollo, y el valor típico implementado en *ITK* por defecto en el algoritmo *ConfidenceConnected* es $f = 2.5$. Los píxeles vecinos de la región actual cuya intensidad cae dentro de dicho rango se aceptan e incluyen en la región. Cuando no hay más píxeles vecinos que cumplan el criterio, se considera que ha concluido la primera iteración del algoritmo. En ese punto, la media y la desviación estándar de los niveles de intensidad se vuelven a computar usando todos los píxeles incluidos en la región. Esta media y desviación estándar definen un nuevo rango de intensidad que se usa para visitar los píxeles vecinos de la nueva región, y evaluar así si sus intensidades caen dentro del rango definido. Este proceso iterativo se repite hasta que en alguna iteración no se puedan añadir nuevos píxeles, o hasta que se alcanza el número máximo de iteraciones. El número máximo de iteraciones es otro parámetro a especificar en tiempo de desarrollo. El valor por defecto del algoritmo implementado en ITK es $NumberOfIterations = 5$. La ecuación siguiente ilustra el criterio de inclusión usado por este algoritmo:

$$I(X) \in [m - f\sigma, m + f\sigma]$$

Las variables *m* y σ representan la media y la desviación estándar de las intensidades de la región, f es un factor definido por el usuario, *I()* es la imagen y *X* es la posición del píxel vecino en particular que está siendo evaluado para su inclusión o exclusión de la región.

Este algoritmo puede verse afectado por el ruido en la imagen, en especial al trabajar con regiones muy extensas. En caso de existir mucho ruido, suele ser conveniente pre-procesar la imagen usando un filtro suavizador que conserve los contornos de los objetos. Pueden usarse distintos filtros para este propósito, pero en el algoritmo evaluado en este proyecto se ha usado un filtro disponible en *ITK* llamado *CurvatureFlowImageFilter*. Este filtro realiza a la imagen un suavizado que trata de conservar los bordes de los objetos, de una manera muy similar a como lo hace un filtro clásico de difusión anisotrópica. El filtro emplea una formulación donde los contornos de intensidad homogénea se representan como *level sets*, donde los píxeles con una intensidad en particular conforman una curva de nivel. La función de las *level sets* se evoluciona bajo el control de una ecuación de difusión en la que la velocidad es proporcional a la curvatura del contorno:

$$I_t = \kappa|\nabla I|$$

Donde κ es la curvatura.

Las áreas con curvatura grande se difundirán más rápidamente que las áreas de baja curvatura. Por tanto, artefactos de ruido con áreas de extensión pequeñas tenderán a desaparecer rápido, mientras las interfaces de amplia extensión evolucionarán más lentamente, preservando así las transiciones abruptas entre objetos. Sin embargo, conviene resaltar que aunque la evolución en los bordes es lenta, aún así existe cierta difusión. En consecuencia, la aplicación continua de este tipo de esquema de filtrado resultará en la eliminación de información conforme se van atenuando los contornos abruptos de la imagen.

3. Isolated Connected

Al igual que *ConfidenceConnected*, *IsolatedConnected* es un algoritmo de crecimiento de regiones implementado en *ITK*, siendo su principio teórico de funcionamiento muy similar al anterior. En este algoritmo el usuario ha de especificar dos semillas y un umbral inferior de intensidad. El algoritmo construirá una región conectada a la primera semilla pero no conectada a la segunda. Para hacer esto, el filtro busca un valor de intensidad que puede usarse como umbral superior para la primera semilla. Se usa una búsqueda binaria para encontrar el valor que separa a ambas semillas.

Este filtro está diseñado para su uso en casos donde existen estructuras anatómicas adyacentes difíciles de separar de la estructura de interés. El hecho de seleccionar una semilla en una estructura y otra en la que no interesa posibilita al algoritmo buscar un umbral de intensidad que separa a ambas estructuras.

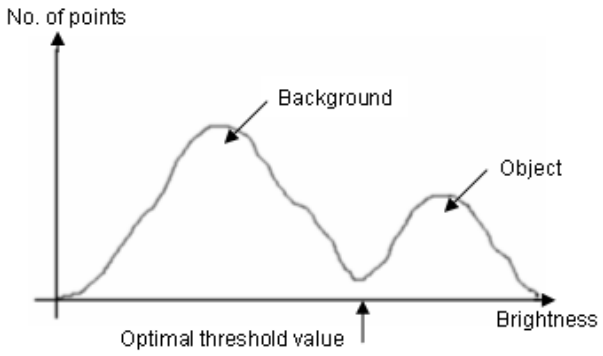
4. Otsu Multiple Segmentation

Se trata de un tipo de algoritmo clásico basado en el establecimiento del valor umbral. El algoritmo, ideado por Nobuyuki Otsu en 1979, está enclavado dentro de la familia de algoritmos por crecimiento de región implementados en *ITK*.

Excluyendo los parámetros triviales del algoritmo (imagen a segmentar, *filebase* para nombrar las segmentaciones que se generan, *fileextension*), el único parámetro que recibe este algoritmo es el número de umbrales en los que dividir el histograma de la imagen.

El algoritmo de Otsu usa la minimización del error por mala clasificación como criterio para clasificar los píxeles. El objetivo es encontrar un umbral (o varios, según el parámetro con que se llame al algoritmo) que clasifica la imagen en dos (o más) clústeres de manera que se minimice el área bajo el histograma en el clúster que está al otro lado del umbral. Esto equivale a minimizar la varianza dentro de cada clase o, lo que es lo mismo, maximizar la varianza entre las clases. Se generan tantos umbrales como se especifique como entrada al algoritmo.

Para ilustrar mejor el funcionamiento del algoritmo, supóngase que se invoca al mismo indicando *n* como el número de umbrales a calcular. El algoritmo generará *n+1* imágenes, conteniendo cada una las regiones correspondientes a las zonas que posean los niveles de grises correspondientes a dicho umbral en el histograma de la imagen original.



La figura de la derecha muestra el histograma de una imagen sujeta al análisis del algoritmo de Otsu, que se basa en la hipótesis de que existe un valor umbral bajo/sobre el cual puede diferenciarse al objeto en cuestión del fondo. El algoritmo OtsuMultipleThreshold evaluado en este proyecto tan sólo difiere de este en que la hipótesis de la que parte es que en lugar de 2 lóbulos, el histograma de la imagen contiene n lóbulos (siendo n un parámetro de entrada del algoritmo), generando como salida n+1 objetos diferenciados.

El formulismo matemático detrás del algoritmo de Otsu se expone a continuación:

$$\mu(k) = \sum_{l=1}^{k} l \cdot p(l)$$

$$\omega(k) = \sum_{l=1}^{k} p(l)$$

$$\mu T = \sum_{l=1}^{N_{max}} l \cdot p(l)$$

$$\sigma_B^2(k) = \frac{(\mu T \cdot \omega(k) \cdot \mu(k))^2}{\omega(k)(1 - \omega(k))}$$

Siendo k la posición índice del histograma en el rango de niveles de gris (de 0 a 255 niveles, si se emplean 8 bits en la codificación).

Siendo p el valor normalizado del histograma para la posición actualmente indexada.

Siendo ω y μ los valores acumulativos de primer y segundo orden del histograma normalizado.

Siendo μT el nivel medio total de la imagen.

5. Threshold level set segmentation

Este algoritmo es una extensión de la segmentación *connected-component*, dentro del marco de algoritmos de *level sets*. A continuación se detallan los principios teóricos generales de funcionamiento de los algoritmos de segmentación por *level sets,* (8), necesarios para la explicación detallada del esquema de funcionamiento del algoritmo *ThresholdLevelSet*.

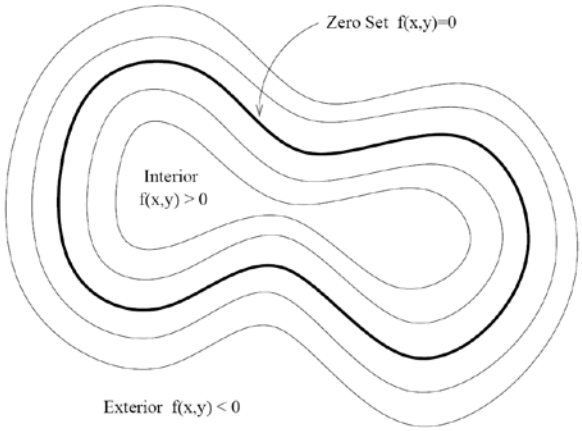


Ilustración 13: concepto de curva cero en un conjunto de level sets

El paradigma de la segmentación por *level sets* consiste en un método numérico para controlar la evolución de contornos y superficies. En lugar de manipular el contorno directamente, el contorno se introduce como la curva de nivel cero dentro de una función de orden superior llamada *level-set function*, $\psi(X,t)$. Esta función se hace evolucionar bajo el control de una ecuación diferencial. En cualquier momento puede obtenerse el contorno envolvente extrayendo la curva de nivel cero $\Gamma\big((X),t\big) = \{\psi(X,t) = 0\}$ a partir de la salida, como muestra la Ilustración 13. Entre las principales ventajas de esta técnica de segmentación están que permite modelar formas arbitrariamente complejas, y que maneja implícitamente cambios topológicos tales como la fusión y la separación.

Las *level sets* pueden usarse para la segmentación de imágenes empleando las características propias de imágenes, como intensidad media, gradiente, y bordes de la misma, dentro de la ecuación diferencial que rige las *level sets*. En un caso de uso típico, un contorno es inicializado por el usuario y es luego evolucionado hasta que se ajusta a la forma anatómica de la estructura de la imagen. Se han publicado numerosas implementaciones y variantes de este concepto básico. Puede consultarse una revisión de los mismos en el artículo de Sethian "*Level Set Methods and Fast Marching Methods*" (9)

La siguiente ecuación representa la ecuación diferencial genérica para una función de de *level sets*. Cada algoritmo de tipo *level-set* particularizará de una manera distinta dicha ecuación diferencial.

$$\frac{d}{dt}\psi = -\alpha A(x)\cdot\nabla\psi - \beta P(x)|\nabla\psi| + \gamma Z(x)\kappa|\nabla\psi|$$

Ecuación 1

Donde A representa un término de advección, P un término de propagación (expansión), y Z un modificador espacial para la curvatura media κ. Las constantes escalares α, β, γ introducen los pesos relativos de cada uno de los términos del movimiento de la interfaz. Los algoritmos de tipo *level-set* toman esta ecuación general como base, y podrán usar todos estos términos u omitir uno o más de ellos.

La mayoría de algoritmos de tipo *level-set* requieren dos imágenes como entradas, un modelo inicial $\psi(X, t = 0)$, y una imagen *feature image*, que puede ser bien la imagen que se desea segmentar o una versión pre-procesada. Se debe especificar el iso-valor que representa a la superficie Γ en el modelo inicial. La salida final individual de cada algoritmo level-set es la función ψ en el último instante de tiempo del algoritmo. Conviene resaltar que el contorno que representa la superficie Γ es la curva de nivel cero de la salida, y no el iso-valor que se especifica para el modelo inicial.

La solución Γ se calcula con precisión a escala sub-píxel. La mejor aproximación discreta de la superficie será por tanto el conjunto de posiciones de la rejilla más cercana a los cruces por cero de la imagen, como se muestra en la Ilustración 14. El filtro *ZeroCrossingImageFilter* disponible en *ITK* encuentra precisamente esas posiciones de la rejilla y se emplea para extraer la superficie.

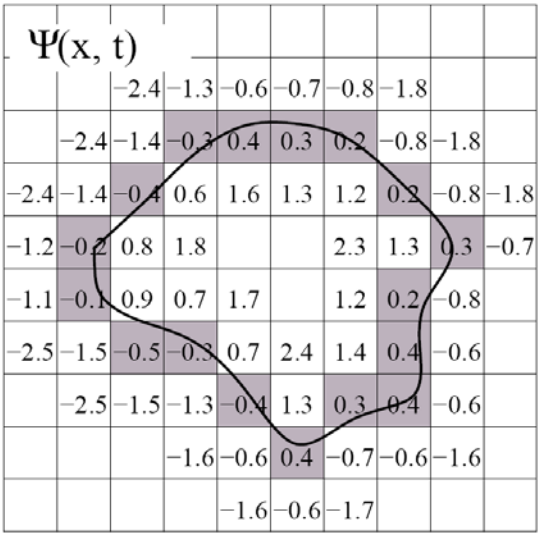


Ilustración 14: Posiciones más cercanas a los cruces por cero de la imagen

Existen dos consideraciones importantes al analizar el tiempo de procesamiento para cualquier tarea de segmentación *level-set*, a saber: El área de la superficie de la interfaz envolvente, y la distancia total que la superficie debe recorrer. Dado que las ecuaciones *level-set* son generalmente resueltas solamente en el ámbito de los píxeles cercanos a la superficie (con la excepción del algoritmo *fast-marching*), el tiempo que tarda cada iteración dependerá del número de puntos de la superficie. Esto implica que conforme la superficie crezca, la resolución de la ecuación se irá ralentizando proporcionalmente. Como la superficie ha de avanzar lentamente para evitar inestabilidades numéricas en la solución, la distancia que la superficie debe recorrer en la imagen dictamina el número total de iteraciones requeridas.

Una vez sentadas las bases teóricas necesarias, se pasa a detallar el funcionamiento del algoritmo *ThresholdLevelset*.

Los parámetros de entrada del algoritmo (excluyendo los triviales como imagen a segmentar y nombre del archivo de salida) son la semilla *Seed*, *InitialDistance*, *LowerThreshold* y *UpperThreshold*. Adicionalmente, el algoritmo admite el parámetro opcional *CurvatureScaling*. Atendiendo a (8), el algoritmo es muy poco sensible a la inicialización provista por el parámetro *InitialDistance*, cuyo valor usado para la evaluación es 5, que se corresponde con el usado por defecto en (8).

ThresholdLevelSet tiene como objeto definir un rango de valores de intensidad que clasifican al tipo de tejido de interés. Ese rango lo usa como base para estimar el término de propagación en la Ecuación 1 de la función *level-set*. Usando la técnica de segmentación por *level sets*, la suavidad de la superficie

puede constreñirse para evitar el derrame o *leaking* que es común en los esquemas *connected-component.*

El término de propagación P de la Ecuación 1 se calcula a partir de las distintas entradas *FeatureImage* g, *UpperThreshold* U y *LowerThreshold* L según la siguiente expresión:

$$P(x) = \begin{cases} g(x) - L & if\ g(x) < \dfrac{U-L}{2} + L \\ U - g(x) & en\ otro\ caso \end{cases}$$

La Ilustración 15 muestra la función del término de propagación. Los valores de intensidad en g que se encuentran entre L y H arrojan valores positivos a P, mientras que las intensidades fuera de este rango arrojan valores negativos a P.

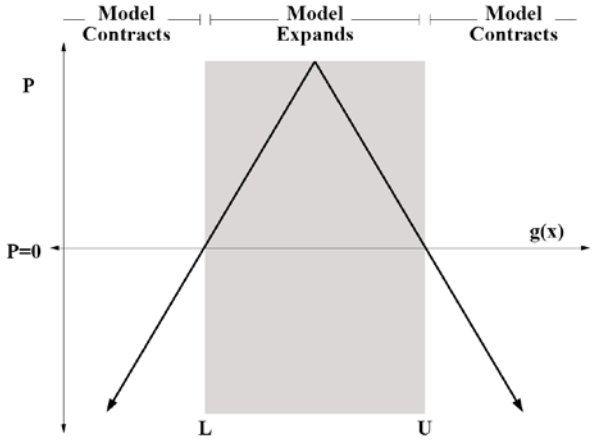


Ilustración 15: Término de propagación

El algoritmo *ThresholdLevelSet* toma dos entradas, como se representa en la Ilustración 16. La primera es un conjunto inicial de curvas de nivel. La segunda es la imagen *feature* g. En la mayoría de aplicaciones, este filtro no requiere pre-procesado en la imagen de entrada. Generalmente no se requiere suavizado de la imagen de entrada para generar soluciones razonables, aunque en algún caso pueda preferirse llevarlo a cabo. La Ilustración 16 muestra cómo se construye el algoritmo. La superficie inicial se genera usando el filtro *fast-marching* (8).

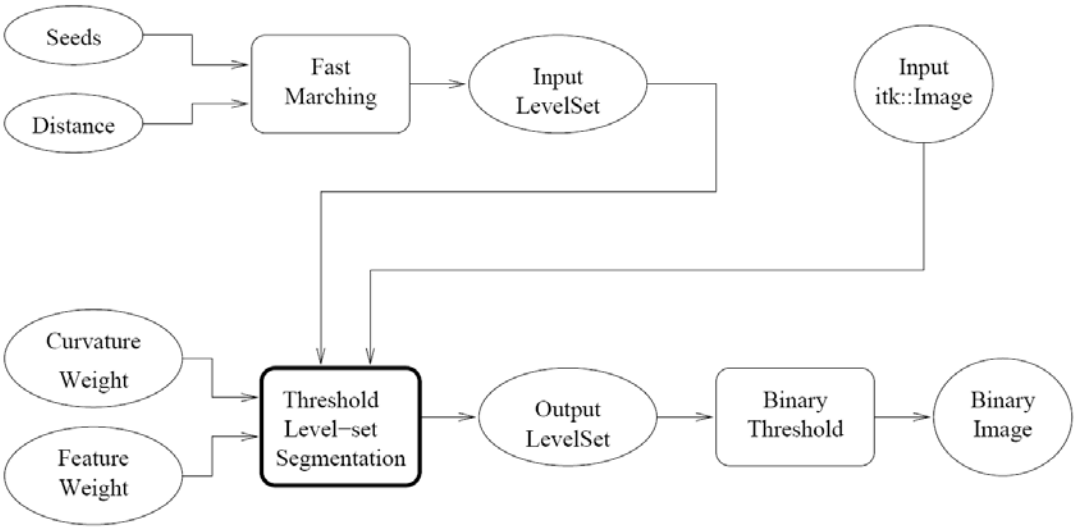


Ilustración 16: Diagrama de bloque del algoritmo ThresholdLevelSet