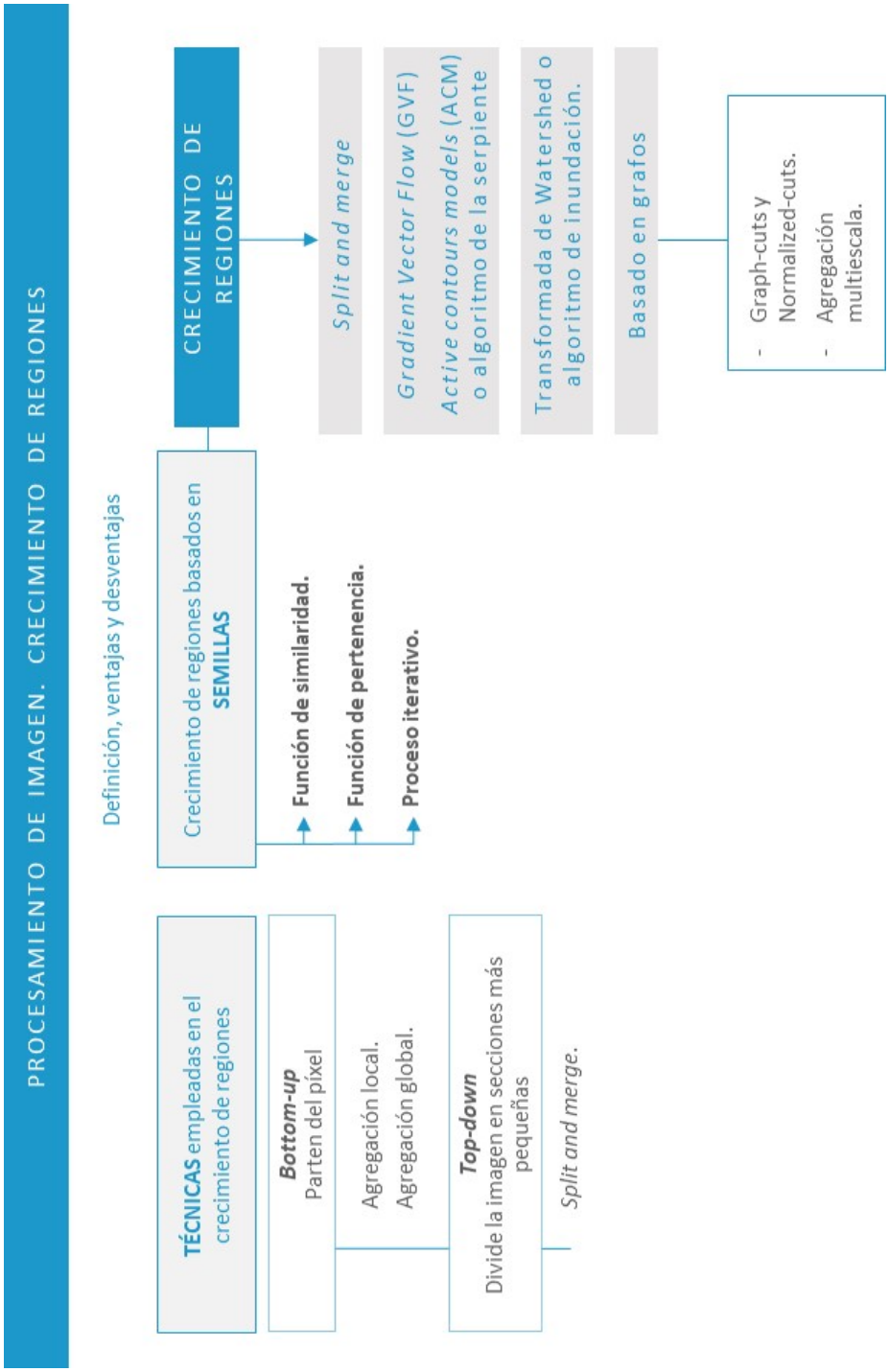
Visión Artificial

Tema 10. Procesamiento de imagen. Crecimiento de regiones

Índice



10.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

10.2. Segmentación y crecimiento de regiones

Como se ha visto en temas anteriores, la segmentación consiste en dividir una determinada imagen en regiones/segmentos de propiedades similares. Esas propiedades pueden venir definidas bien por el color, la textura o incluso por el hecho de estar acotados por contornos muy bien definidos.

El crecimiento de regiones es un algoritmo generalmente **no supervisado** (existe posibilidad de aplicar supervisión, pero no es lo habitual) que, partiendo de unas regiones concretas en la imagen, normalmente elegidas por una persona, encuentra los límites de dicha región donde el punto se encuentra.

En otras palabras, dada una región inicial, el algoritmo explora en la vecindad de dicha región y considera que aquellos píxeles más parecidos son de la misma región.

A continuación, un ejemplo de crecimiento de regiones. Inicialmente se parte de un punto y se expande alrededor de dicho punto hasta que la región encontrada empieza a no ser uniforme y el algoritmo se detiene.

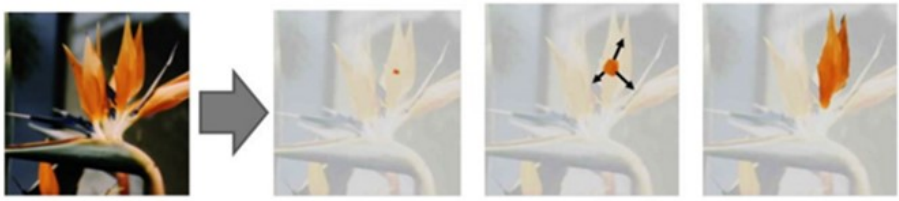


Figura 1. Ejemplo de crecimiento de regiones.

Por supuesto, este enfoque posee ventajas y desventajas. Dentro de las **desventajas** encontramos el hecho de que necesita una buena inicialización, es decir, requiere de intervención humana que indique dónde comenzar a crecer.

De hecho, este tipo de algoritmos se utiliza en biomedicina: imágenes biomédicas como ecografías, radiografías, etc., donde el médico indica qué región le interesa y el algoritmo encuentra las mejores fronteras de cada región.

Adicionalmente, como desventaja, nos encontramos con que este tipo de algoritmos son computacionalmente muy costosos por dos motivos:

- **Son algoritmos iterativos y no paralelizables.** Es decir, hasta que no acaba una iteración no puede comenzar la siguiente.
- Las **funciones** que exploran si un píxel debe pertenecer o no a una región son **complejas de evaluar y costosas de implementar,** lo que hace que estos algoritmos sean dependientes de estas funciones de evaluación.

Como **ventaja** se encuentra el hecho de que el algoritmo, hasta cierto punto, es autónomo para encontrar las fronteras de las regiones de interés. Esta **autonomía** está basada en funciones de comparación y en funciones de maximización y minimización. Si esas funciones no están correctamente parametrizadas o definidas, el algoritmo nunca será capaz de encontrar una segmentación adecuada.

10.3. Técnicas empleadas en el crecimiento de regiones

Existen dos familias de técnicas para el crecimiento de regiones:

- Aquellas técnicas que se encargan de resolver el problema **partiendo de píxeles singulares** y encontrando regiones más globales. Llamadas *bottom-up:* parten del mayor grado de detalle (el píxel) y agrupan en función de él. Podemos decir que existe una división también dependiendo de cómo se realice la agregación, lo que da lugar a dos tipos de sub-familias:

  - Técnicas basadas en **agregación local,** es decir, para agregar únicamente nos fijamos en los vecinos más cercanos del píxel en cuestión (o de la región que se está agregando).

  - Técnicas basadas en agregación **global,** donde se tiene una visión global de toda la imagen y dos píxeles solo se unen bajo una misma región siempre y cuando a nivel global tenga sentido dicha agregación.

- Aquellas que se encargan de dividir la imagen en **secciones más pequeñas** de forma iterativa, viendo hasta qué punto dos regiones pueden unirse. Este algoritmo es conocido como *split and merge* y a estas técnicas también se denominan *top-down.*

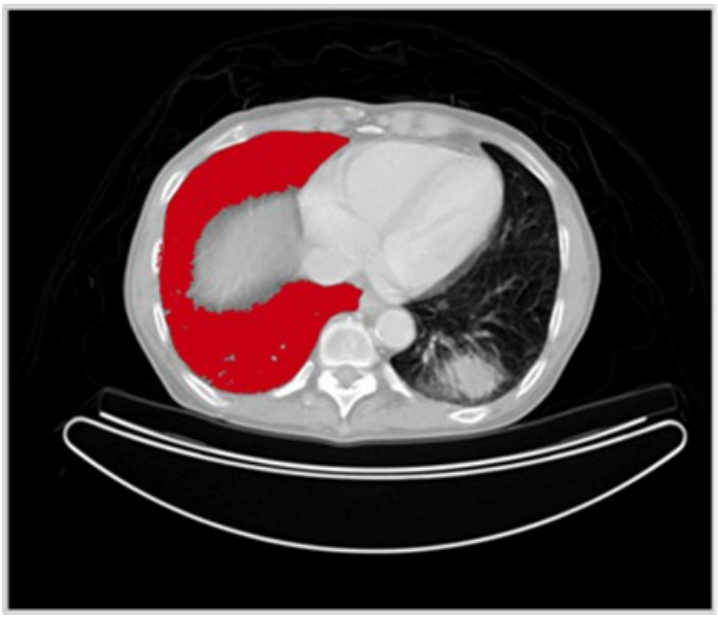


Figura 2. Resultado de un algoritmo de crecimiento de regiones (zona roja). Fuente: https://es.mathworks.com/matlabcentral/fileexchange/19084-region-growing

Como hemos comentado, el uso de este tipo de algoritmos está muy extendido en biomedicina y segmentación de imágenes médicas, donde no existe la posibilidad de obtener colores.

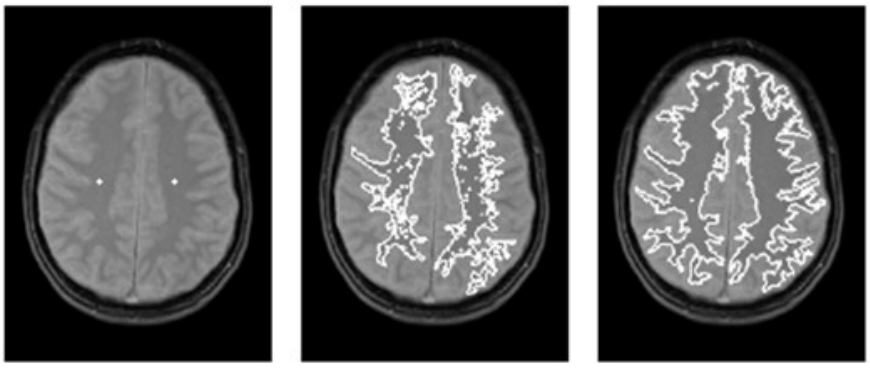


Figura 3. Evolución de un algoritmo bottom-up de crecimiento de regiones. Fuente: https://www.creatis.insa-lyon.fr/~grenier/?cat=8

Figura 4. Ejemplo de algoritmo split and merge. Fuente:
https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/hat/

En esta última imagen vemos que, partiendo de pequeñas divisiones de la imagen *(split)* se decide unir *(merge)* aquellas regiones con mayor parecido.

10.4. Crecimiento de regiones basado en semillas

El crecimiento basado en semillas es el más intuitivo y para ello hacen falta los siguientes ingredientes, que serán comunes al resto de métodos de crecimiento de regiones.

Inicialmente asumiremos que una semilla vendrá dada por un píxel. Otras alternativas contemplan el uso de una forma bien definida por el usuario de forma interactiva sobre la imagen, bien ya predefinida: un círculo, cuadrado, etc.

Por lo tanto, para trabajar con este método basado en semillas necesitamos:

- Un conjunto de píxeles: $p_1, p_2, \ldots, p_n$ que servirán como semillas.
- La **posición** de esos píxeles vendrá dada por la propia posición que ocupe dentro de la imagen, y las **propiedades** de cada píxel, por la función $I(p_i)$ donde se elige el nombre I, ya que comúnmente será la intensidad de color de dicho píxel. Otros valores de la función pueden ser la saturación u otras codificaciones de color que no sean las dadas por el criterio RGB.
- Una **función de similaridad** entre dos píxeles p_i y p_j. Esta similaridad vendrá dada por $S(p_i, p_j)$ y podrá aplicarse bien entre píxeles o bien entre regiones como posteriormente veremos.
- Una **función de pertenencia** a una región; dada $\pi_r(p_i)$ donde r hace referencia a la región en concreto y p_i a un píxel cualquiera. Esta función es 1 si $S(r, p_i) > T$, es decir, si la función de similaridad supera un cierto umbral. En cualquier otro caso, será 0. Dicho umbral se fijará de forma automática por el algoritmo o bien de forma manual, si así lo decide el usuario final.

Con lo cual, un algoritmo de crecimiento de regiones basado en semillas lo que intenta es encontrar el número mínimo de regiones que maximiza todas y cada una de las funciones de similaridad y de pertenencia dentro de cada región.

Este **proceso iterativo** posee las siguientes características:

- Al ser iterativo, es un proceso que posee gran coste computacional y cuya optimización debe realizarse, en muchos casos, de forma secuencial y no paralela.
- Solo considera regiones adyacentes, por lo que si dos píxeles no adyacentes, por función de similaridad S y de pertenencia pudieran pertenecer a una misma región, no serán unidos dentro de una misma región.
- Esto conduce en la gran mayoría de casos a una **sobresegmentación:** hay un número elevado de segmentos/grupos dentro de la imagen, y por ende, otro postprocesado para eliminar dicha sobresegmentación.
- Cada píxel tiene que tener una región, así que en muchos casos, algunos píxeles son forzados a pertenecer a una determinada región simplemente por estar en su adyacencia.
- Si la región a detectar no es homogénea y posee geometrías complicadas, este método no proporcionará un resultado preciso debido a que el algoritmo tenderá a crecer en todas direcciones, respetando la vecindad del píxel.
- Sin embargo, por su sencillez e interactividad permite encontrar una primera aproximación de la segmentación muy certera en aquellas imágenes donde el contraste sea muy alto.
- Además, suele servir como origen para otros algoritmos de crecimientos de regiones más específicos y complejos.

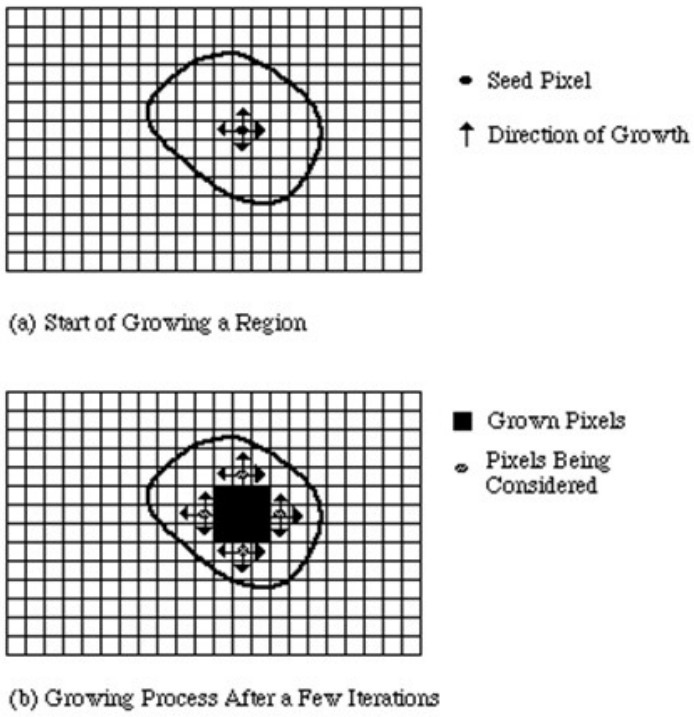


Figura 5. Comienzo del algoritmo de crecimiento de regiones basado en semillas. La dirección de crecimiento suele ser otro parámetro del algoritmo. Fuente: https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node35.html

10.5. Crecimiento de regiones basado en Split and Merge

Para explicar el algoritmo basado en *split and merge* voy a usar el siguiente esquema y cuatro subimágenes.

1	IMAGEN ORIGINAL *i*	Esta imagen es considerada en este método (y en este paso) como un segmento en sí mismo. La idea es ver si se pueden encontrar segmentos con funciones de similaridad y pertenencia más precisas.
2	DIVISIÓN EN CUATRO PARTES	Se divide la imagen en cuatro partes, evaluándose las funciones de similaridad y pertenencia en cada una. Se eligen cuatro en función del diseñador del algoritmo, aunque pueden escogerse el número de particiones que se desee. Para ser más óptimos computacionalmente, se suele dividir en cuatro y cada una a su vez en otras cuatro subpartes.

Figura 6. Pasos a seguir con el algoritmo *Split and Merge.*

En este ejemplo, únicamente el subconjunto I_4 no verifica las condiciones de similaridad y pertenencia. Esto quiere decir que la entropía dentro de dicho segmento o la variabilidad de intensidades hace que esté por debajo de un determinado umbral y por lo tanto haya que realizar de nuevo el proceso de dividir la imagen en cuatro partes. En este caso, únicamente en el segmento I_4 .

De las cuatro subimágenes que se han creado, los segmentos I_43 e I_44 tienen propiedades similares y por lo tanto se fusionan (merge). ¿Por qué se dividieron si después hay que fusionarlos? Simplemente por el funcionamiento del algoritmo. Primero realiza divisiones de forma automática, en este caso en cuatro, y posteriormente evalúa si dos segmentos deberían de seguir juntos o no.

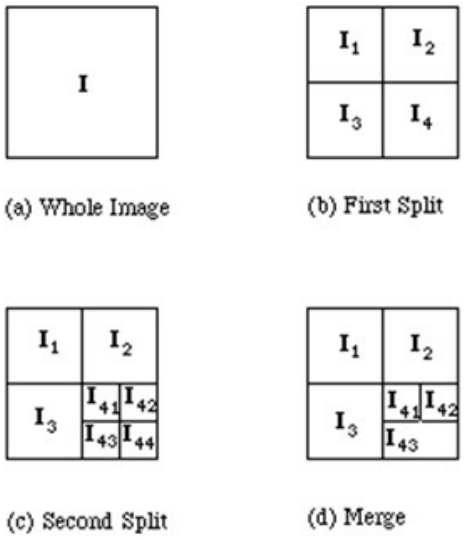


Figura 7. Esquema de pasos principales en el algoritmo split and merge. Fuente: https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node34.html

En la figura 7 vemos entonces que:

- La imagen original, todos los píxeles poseen la misma importancia (a).
- Se divide la imagen en partes iguales (b).
- Únicamente el segmento I_4 se subdivide en más subelementos debido a que dicho segmento es el menos uniforme (c).
- Los segmentos I_{43} e I_{44} se fusionan *(merge)* en uno debido a que tienen propiedades similares (d).

Este proceso de *split and merge* es «paralelizable», ya que el análisis se puede ejecutar de forma simultánea para los segmentos I_1 , I_2 , I_3 e I_4 .

Además, el coste computacional de este algoritmo va decreciendo a medida que va

avanzando, ya que los subconjuntos/segmentos cada vez son más pequeños en tamaño y hay menor número. Teóricamente se puede asegurar que el coste computacional de este algoritmo está acotado.

Para verlo más fácilmente, pondré un ejemplo con una imagen de 16x16 píxeles. En ella puede apreciarse que ya de por sí existen regiones con valores muy similares (valores iguales a 7).

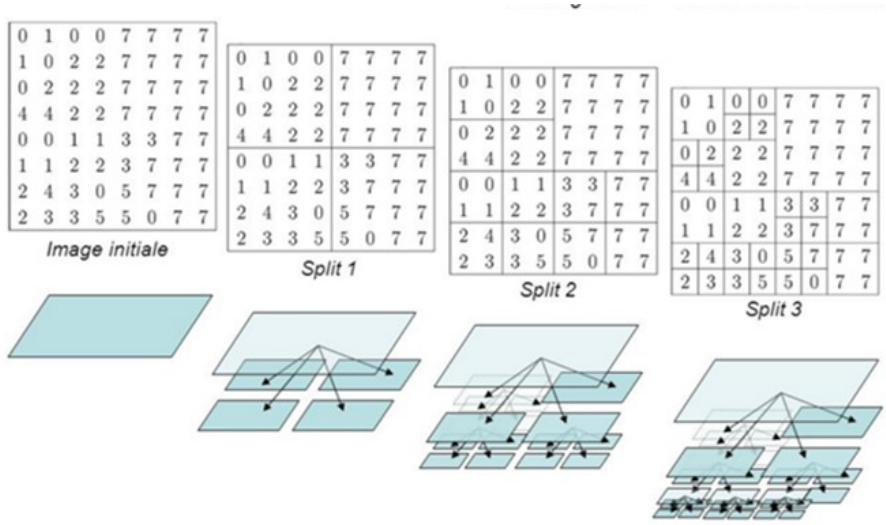


Figura 8. Máscaras correspondientes a los operadores de Prewitt para la identificación de bordes con orientación horizontal y vertical. Fuente: http://slideplayer.fr/slide/518235/

- En la primera iteración *(Split 1),* se divide la imagen también en cuatro partes. De todas ellas, hay una que ya no se dividirá más puesto que todos los píxeles son iguales entre sí.

- Del resto de regiones, se dividen de nuevo por la mitad y en una tercera división se aprecia cómo algunos subconjuntos con valores similares no se han dividido de nuevo. Es aquí donde entra en juego el umbral que veíamos en la segmentación basada en crecimiento de regiones con semillas. Si ese umbral se disminuye, el algoritmo es más estricto y por lo tanto producirá sobresegmentación.

- En paralelo, se puede ver cómo se va construyendo un grafo jerárquico que explica cómo la imagen se ha ido dividiendo y cómo cada segmento contiene varios subsegmentos. Ese grafo se utiliza en otras técnicas (como graph-cuts) para poder segmentar de una manera más precisa e incluso para poder unir segmentos parecidos.

Por último, es importante indicar que aunque el algoritmo *split and merge* es uno de los más empleados, tiene una **desventaja** muy importante y es que, una vez que dos segmentos han sido separados, jerárquicamente están separados en ramas diferentes, el algoritmo no evaluará si pueden ser fusionados en un mismo segmento.

Este proceso debe hacerse de forma posterior en un postprocesado que ha de recorrer todos los segmentos y evaluar la función de similaridad y pertenencia. Sin embargo, aunque esto puede parecer costoso, como el número de segmentos es pequeño, puede realizarse sin problema.

10.6. Crecimiento de regiones basado en Gradient Vector Flow (GVF)

Los métodos conocidos como *Gradient Vector Flow* (GVF en adelante) o *Active Contour Models* (ACM) suelen estar clasificados como métodos de segmentación basados en modelos u orientados a modelo y contorno. Sin embargo, pueden considerarse como métodos de crecimiento de regiones en un ámbito más amplio.

Esta familia de algoritmos se conoce popularmente como el **algoritmo de la serpiente,** puesto que simula el comportamiento de este animal a la hora de ir encontrando el contorno deseado.

Este tipo de métodos se basa en ir modificando el contorno de una determinada figura base (inicialmente suele ser un círculo) e irla deformando de forma iterativa hasta que se adapta a la forma del objeto o región que se quiere segmentar.

En este sentido, son muy dependientes de la inicialización y de las funciones de coste o de optimización, es decir, el criterio que siguen estas curvas para readaptarse al contorno buscado.

A continuación se presentan tres ejemplos de cómo funciona visualmente este algoritmo.

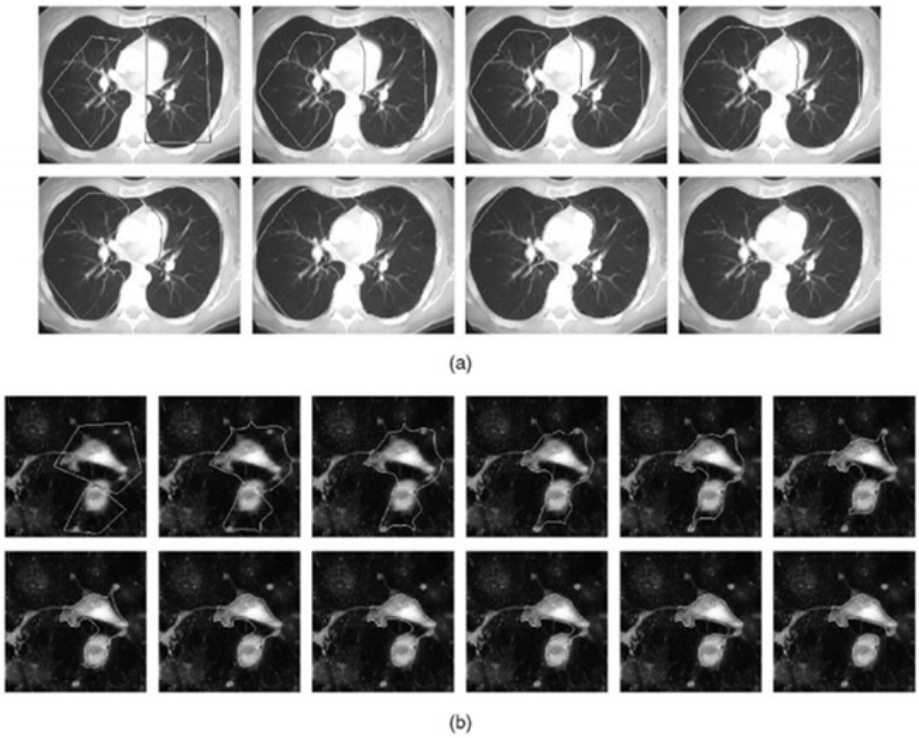


Figura 9. Funcionamiento del algoritmo GVF. Fuente: Paragios y Ramesh, 2004.

En la figura 9, (a) y (b) son imágenes biomédicas donde se aprecia como el algoritmo comienza con un contorno inicial y dicho contorno se va adaptando a la forma a segmentar.

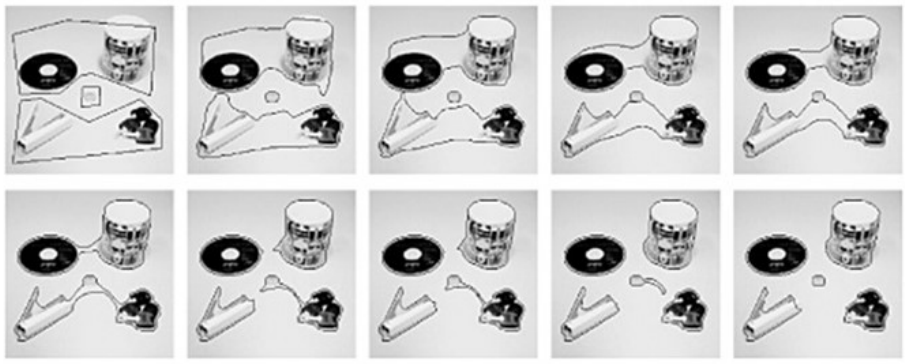


Figura 10. Funcionamiento del algoritmo GVF para la segmentación de varios objetos. Fuente: Paragios y Ramesh, 2004.

En este último ejemplo, la segmentación es relativamente sencilla debido a que el fondo de la imagen está muy diferenciado.

No obstante, aunque GVF y ACM suelen ser sinónimos en la literatura, se puede decir que GVF es una extensión de ACM o algoritmos de serpiente. La **diferencia** más significativa radica en que GVF convergen en las concavidades del contorno y no necesitan una inicialización tan cercana al contorno como en el caso de los algoritmos ACM.

De hecho, veamos la diferencia de forma matemática. El algoritmo de serpiente original (v) es un contorno bidimensional dinámico definido de forma paramétrica como $v(s) = [x(s), y(s)]$ donde $s \in [0,1]$ y que minimiza la siguiente función de energía:

$$E = \int_0^1 E_{\int}(v(s)) + E_{imagen}(v(s)) + E_{con}(v(s))\, ds$$

Donde:

- $E_{\int}$ denota la energía del contorno interior.
- E_{im} representa la energía debida a la intensidad propia de la imagen.
- Y E es una función de contorno que modela la parte exterior del contorno.

La definición de estas energías en modo de función queda fuera del ámbito de este tema, pero en literatura pueden encontrarse numerosos ejemplos de funciones para estas energías.

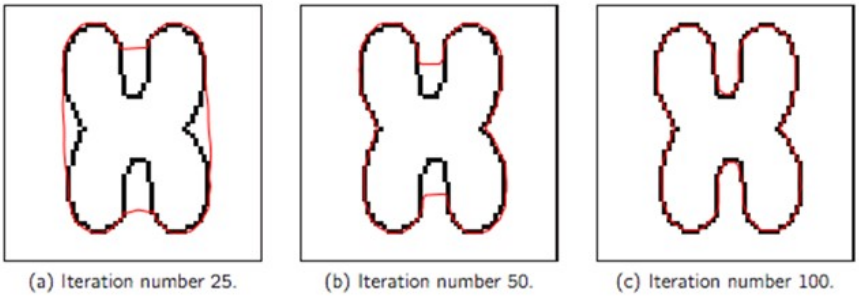


Figura 11. Esquema de funcionamiento del algoritmo GVF. Fuente: Cartas-Ayala, s. f.

En el caso de los métodos GVF, y aquí radica la diferencia con los algoritmos de serpiente, la función de energía E_{con} utiliza una superficie paramétrica en vez de una curva, es decir, es función de $[\ (x,y),v(x,y)]$ y no solo de $v(s)$.

Recomendamo0s leer lo siguiente:

Kass, M., Witkin, A. y Terzopoulos, D. (1987). Snakes: Active contour models. International *Journal of Computer Vision,* 1(4), 321–331. Recuperado de https://link.springer.com/article/10.1007/BF00133570

Leroy, B., Herlin, I. y Cohen, L. (1996). Multi-resolution algorithms for active contour models. En M. O. Berger et al. (Eds.). *ICAOS '96. Lecture Notes in Control and Information Sciences,* 219. Heidelberg: Springer Books. Recuperado de https://link.springer.com/chapter/10.1007/3-540-76076-8_117

Por último, presentamos una tabla comparativa entre GVF y ACM donde se presentan las ventajas y desventajas de ambos.

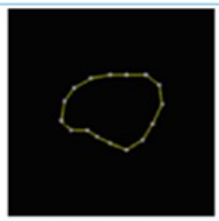
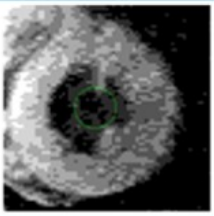
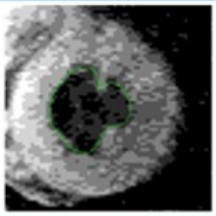
GVF vs ACM	Curva inicial	Curva final
El algoritmo de serpiente tradicional debe de inicializarse con un contorno muy similar al objeto a segmentar y aun así, no converge en las concavidades.		
El algoritmo GVF puede empezar muy alejado del contorno a segmentar, pero será capaz de converger en las concavidades.		
Incluso, si la inicialización de GVF atraviesa el contorno, será capaz de converger, a diferencia de ACM.		
GVF converge a contornos definidos de forma subjetiva, es decir, se aprecian a nivel visual, pero no están claramente definidos.		
GVF puede trabajar también con escalas de grises (aquí se aprecia cómo el algoritmo se adapta para encontrar las paredes del corazón en una imagen de resonancia magnética)		

Tabla 1. Comparación entre GVF y ACM. Fuente: Adaptado de http://www.iacl.ece.jhu.edu/static/gvf/

10.7. Crecimiento de regiones basado en Watershed

La transformada Watershed o segmentación basada en Watershed suele estar englobada dentro de la morfología matemática. Sin embargo, la motivación para ubicarla dentro de este tema se debe a que su funcionamiento está más cercano al crecimiento de regiones que a la propia morfología matemática.

La segmentación basada en Watershed se conoce popularmente como **algoritmos de inundación,** ya que utilizan el símil de la inundación de pantanos (o superficies) para explicar muy visualmente el algoritmo.

Con respecto a otras técnicas de segmentación, el objetivo de esta técnica es dividir en regiones la imagen de nivel de grises.

Generalmente una de las regiones se corresponde con el fondo de la imagen, que puede contener otros objetos fuera de interés, y el resto con los objetos o regiones que se pretende extraer. El objetivo último de esta técnica es **determinar los contornos** que definen dichos objetos.

El punto de partida es considerar que los contornos de una imagen se corresponden con las líneas donde el nivel de gris varía más rápidamente que en un determinado entorno vecino. En este punto, hay que hacer uso del **gradiente de la imagen,** que proporciona la posición de los píxeles donde la variación de intensidad es más pronunciada, muy similar al concepto de derivada matemática.

El uso de dicho gradiente nos permite asumir que los contornos de la imagen original se corresponden con las líneas de cresta de la imagen gradiente.

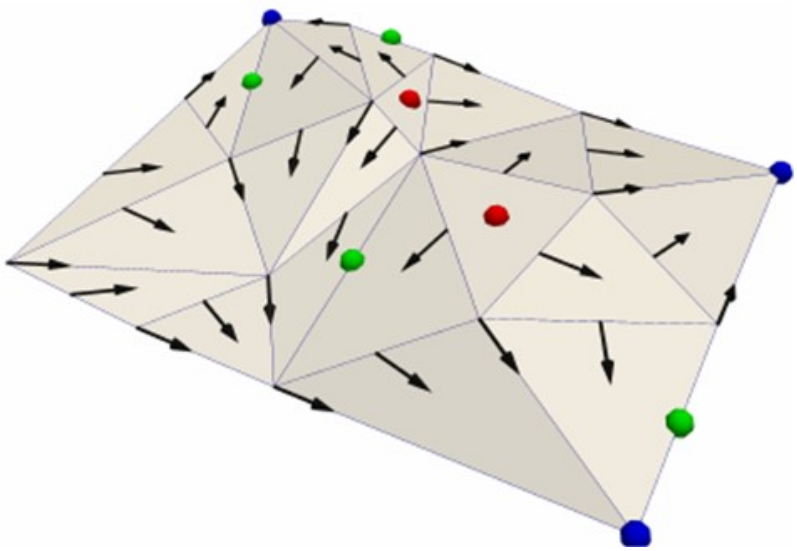


Figura 12. Ejemplo visual de gradiente de una imagen. Fuente: https://www.researchgate.net/figure/Example-of-a-discrete-gradient-vector-field-on-a-triangulated-terrain-Paired-simplices_fig1_261674491

El concepto de Watershed se basa en visualizar una imagen en tres dimensiones (3D): dos coordenadas espaciales frente a niveles de gris. En esta interpretación topográfica, de ahí que estos algoritmos estén dentro del campo de la morfología matemática (nacieron dentro del mundo de la topografía), se considerarán tres tipos de puntos:

- Puntos que corresponden a mínimos locales.
- Puntos en los que, si se coloca una gota de agua, esta cae con certeza en un único mínimo: los puntos que verifican esta condición se denominan **puntos Watershed** o *catchment basis.*
- Puntos en los que el agua caería con igual probabilidad en más de uno de estos mínimos: los puntos que verifican esta condición forman las **líneas Watershed o líneas de cresta.**

El objetivo principal de los algoritmos de segmentación es alcanzar estas líneas divisorias o de cresta.

La idea básica es la siguiente: supongamos que se hace un pequeño agujero en cada mínimo local y que todo el relieve topográfico es inundado desde abajo. El agua va subiendo e inundando las cuencas. Cuando el agua de dos cuencas está a punto de juntarse, se construye un dique *(dam)* para evitar la fusión. La inundación continúa y llega a un punto en que solo se ve la parte de arriba de los diques por encima de la línea de agua. Las líneas de Watershed forman un camino conexo, dando por lo tanto bordes continuos entre las regiones. Este ejemplo puede verse en la imagen a continuación.

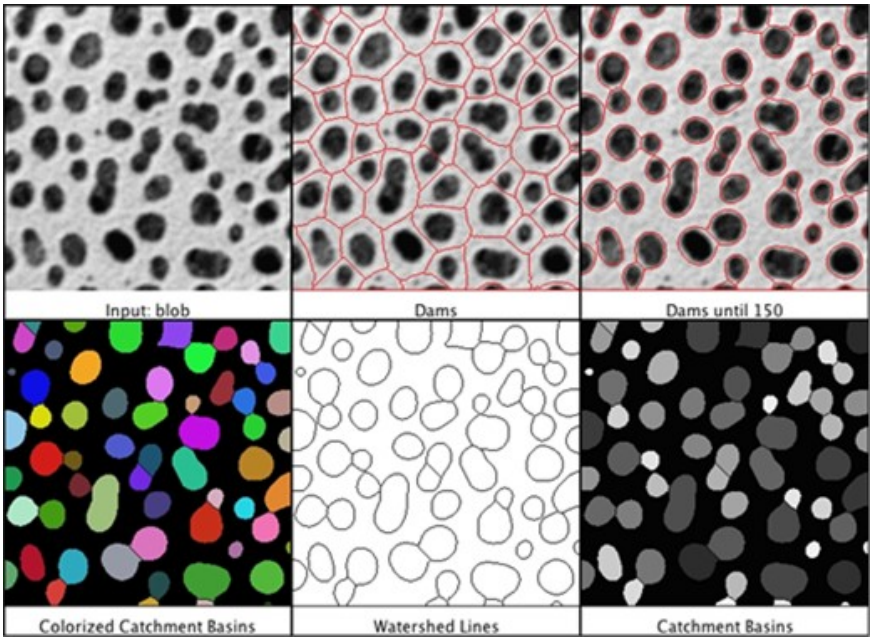


Figura 13. Pasos más comunes en la ejecución de la segmentación basada en Watershed. Fuente: http://bigwww.epfl.ch/sage/soft/watershed/

Ideas clave

La segmentación basada en Watershed produce normalmente **sobresegmentación** y puede deberse a diferentes aspectos como la diferencia de texturas o patrones, ruido o incluso cambios en las tonalidades del color. Para ello, lo más habitual suele ser:

- Eliminar los contornos irrelevantes una vez realizado el Watershed.
- Modificar la imagen gradiente de tal forma que las regiones de depresión o valles se correspondan únicamente con los objetos deseados.
- Imponer marcadores como mínimos de la imagen gradiente.
- Suprimir todos los demás mínimos del gradiente (los irrelevantes) rellenando los correspondientes valles.
- Preservar las líneas de cresta más importantes de la imagen gradiente, localizadas entre los marcadores.

10.8. Crecimiento de regiones basado en grafos

Por último, presentamos un apartado dedicada a los algoritmos de crecimientos de regiones basadas en **grafos.** Estos algoritmos entienden la imagen como un grafo con cierta conectividad entre los píxeles y con unas **propiedades de conectividad:** funciones de similaridad, peso, distancia, etc.

En función de esas propiedades, el algoritmo itera el grafo eliminando las conexiones más débiles (menos parecidas) y potenciando las más fuertes.

Dentro de esta familia se encuentran los siguientes algoritmos principales:

- **Graph-cuts** y **Normalized-cuts:** dado un grafo, eliminan aquellas conexiones, también conocidas como *edges,* entre los píxeles (llamados en la literatura como *nodes),* que son más débiles.
- **Agregación multiescala:** partiendo de un grafo, crean diferentes niveles de segmentación donde el algoritmo va encontrando qué subsegmentos (subgrafos) pertenecen a un mismo segmento. Muy similar al concepto de *split and merge,* pero utilizando métodos como *graph-cuts* para realizar el *split.*

A continuación, se presenta un esquema resumido de ambos algoritmos.

Graph-cuts

Para Graph-cuts se presenta el esquema típico de funcionamiento:

- En primer lugar, se obtiene la imagen original.
- En segundo lugar, se crea el grafo obteniendo para cada conexión entre píxeles (nodos dentro del grafo) una función de similaridad o parecido. La vecindad puede ser sencilla, como en este caso, o compleja si consideramos conectividades con nodos/píxeles más lejanos.
- Se buscan aquellas conectividades más débiles o que poseen una similaridad menor. El corte del grafo por esos puntos más débiles es lo que otorga el nombre de Graph-cuts al algoritmo.
- Por último, se convierte el grafo en imagen de nuevo, esta vez con una frontera que refleja la segmentación realizada.

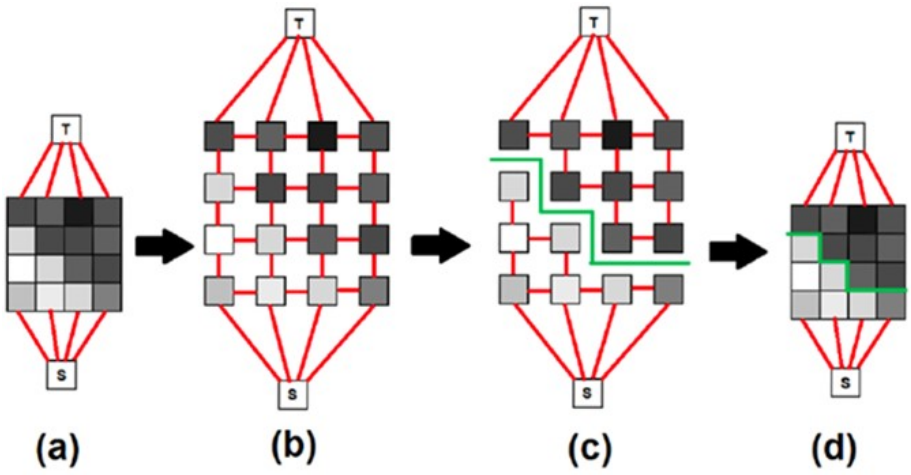


Figura 14. Funcionamiento simplificado de graph-cuts.

Agregación multiescala

El funcionamiento es el siguiente:

- Se parte de la imagen original, donde cada píxel representará un nodo.
- Se realiza, en función de un criterio de similaridad, una agrupación basada en los nodos más próximos. Esta agregación de por sí ya es una segmentación, aunque proporciona resultados muy pocos precisos.
- Una vez agrupados los nodos en segmentos, se obtiene un **representante** (normalmente el valor medio) de dicho segmento, lo que conduce a una nueva escala (nueva capa) con menos nodos, pero cada nodo representando a un subconjunto de nodos iniciales.
- Se vuelve a repetir la agregación, esta vez para los nodos representantes y se itera este proceso hasta que, o bien se alcanza un número determinado de segmentos, o bien se verifica una condición de calidad, es decir, que los segmentos creados, independientemente del número que sean, tienen unas propiedades bien definidas.
- Finalmente, la imagen se segmenta deshaciendo las agrupaciones realizadas por cada uno de los niveles.

De esta manera, si existen regiones parecidas entre ellas, pero separadas de forma espacial en la imagen, el algoritmo encontrará la manera de asociarlas a un mismo segmento.

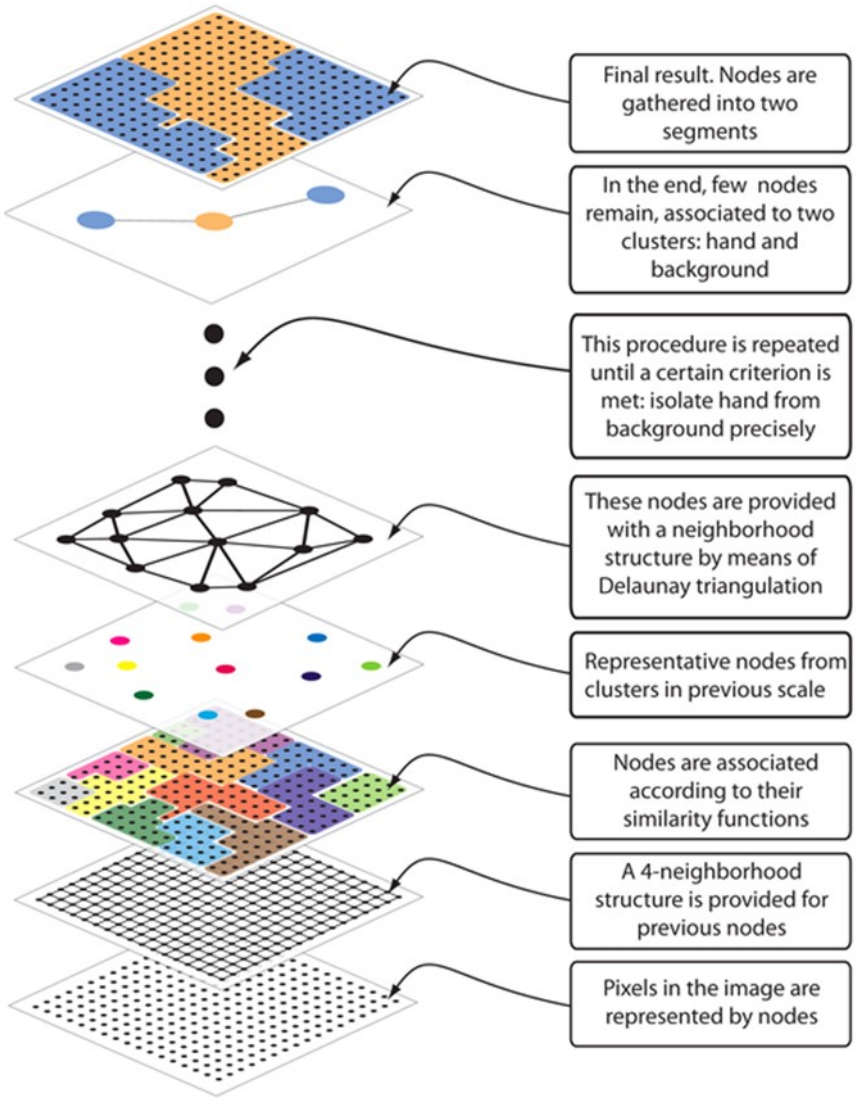


Figura 15 Procedimiento de agregación multiescala comenzando por la imagen original (parte inferior) y terminando con la imagen segmentada (parte superior).

10.9. Referencias bibliográficas

Cartas-Ayala, A. (Sin fecha). Gradient Vector Flow Snakes. Manuscrito presentado para su publicación. Recuperado de http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/cartas.pdf

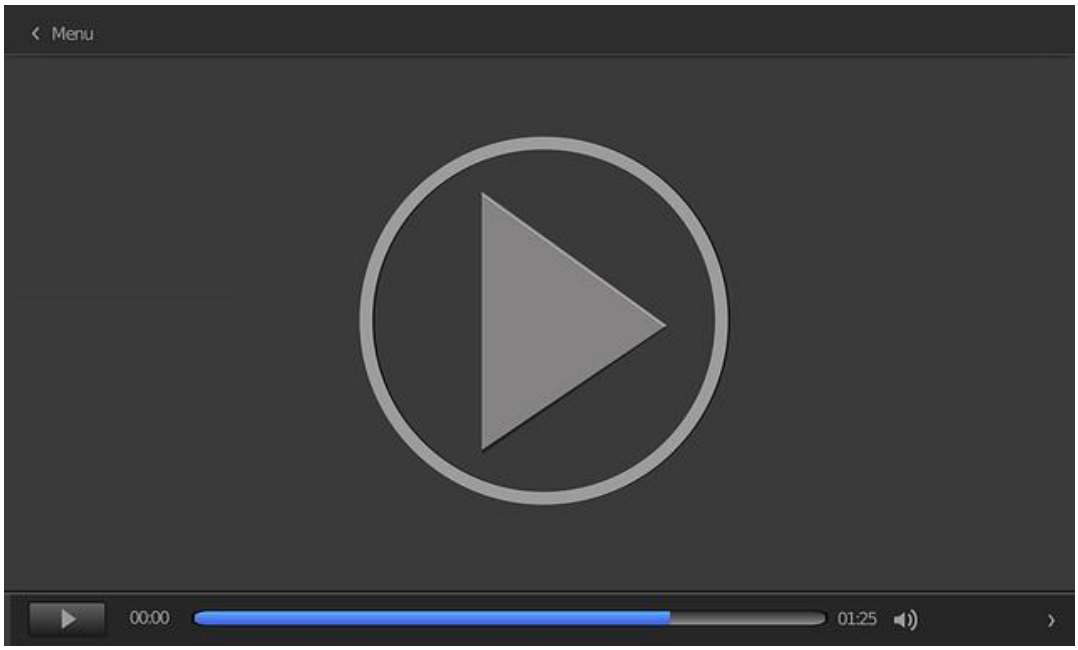
González, R. C. y Woods, R. E. (2008). *Digital image processing*. New Jersey: Pearson Education.

Paragios, N. y Ramesh, V. (2004). Gradient Vector Flow Fast Geometric Active Contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(3), 402-407. Recuperado de https://www.researchgate.net/publication/8337757_Gradient_Vector_Flow_Fast_Geometric_Active_Contours

Watershed

UniHeidelberg. (2013, mayo 21). *5.1 Watershed | Image Analysis Class 2013* [Vídeo]. YouTube. https://www.youtube.com/watch?v=A0CoHAOhrWg

Vídeo en el que profundiza en la transformada Watershed y se da una explicación detallada del algoritmo que no se cubre en este tema.



Accede al vídeo:

https://www.youtube.com/embed/A0CoHAOhrWg

Segmentation by Weighted Aggregation

Sharon, E., Galun, M., Basri, R. y Brandt, A. (2004). *Hierarchical Adaptive Texture Segmentation or Segmentation by Weighted Aggregation (SWA).* Departament of CS and Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel. http://www.cis.upenn.edu/~jshi/GraphTutorial/Tutorial-ImageSegmentationGraph-cut4-Sharon.pdf

Con el fin de profundizar sobre la agregación multiescala, este documento presenta otro enfoque similar al presentado en este tema.

Test

1. La segmentación basada en crecimiento de regiones es un proceso:

   A. Supervisado, necesita tener segmentaciones ya realizadas para parametrizar adecuadamente los diferentes modelos

   B. No supervisado, como todo método de segmentación

   C. Semisupervisado, ya que generalmente depende de una inicialización muy orientada y de una parametrización de los criterios a maximizar/minimizar.

2. La segmentación basada en crecimiento de regiones es más costosa computacionalmente que la segmentación basada en color con k-means:

   A. Verdadero

   B. Falso

3. El uso de semillas para el crecimiento de regiones es:

   A. La única solución posible para inicializar estos algoritmos.

   B. Un método sencillo de inicialización.

   C. Útil siempre que no haya más de tres semillas en una imagen.

4. Los algoritmos de GVF pueden usarse en imágenes 3D siempre y cuando:

   A. Sean imágenes biomédicas.

   B. Se defina una función de energía que considere tres dimensiones.

   C. No existe una implementación GVF para 3D, únicamente tiene sentido para imágenes 2D.

5. La gran mejora que proporciona GVF en comparación con ACM es:

   A. Son más robustos a la inicialización.

   B. Utilizan funciones de energía menos complejas.

   C. Ninguna de las anteriores.

6. La sobresegmentación que proporciona Watershed puede eliminarse mediante:

A. Un procesado posterior eliminando las líneas Watershed menos importantes.

B. Evitando filtrados que al suavizar la imagen eliminen los detalles necesarios para una buena segmentación.

C. Todas las anteriores.

7. En una imagen de 32x32 píxeles, con todos los píxeles en blanco, ¿cuántas veces se haría el proceso *split* del algoritmo *split and merge?*

A. Siempre se realiza al menos una vez.

B. En este caso ninguna, ya que todos los píxeles son iguales.

C. Se realizará una vez y después se hará un *merge* dejando la imagen segmentada igual que la imagen original.

8. Supongamos que tenemos un tablero de ajedrez de 8x8 píxeles y cada píxel es una casilla, luego será blanca o negra alternativamente, ¿cuántos segmentos resultarán?

A. Dependerá de la función de similitud, pero mínimo un segmento y máximo 64 segmentos.

B. En ningún caso podrán ser más de 8 segmentos.

C. Serán 64 segmentos, independientemente de la función de similitud.

9. En la agregación multiescala, ¿qué proceso es el que consume más tiempo?

A. La creación de los segmentos basada en las distancias y la similitud.

B. La creación del grafo y los cortes una vez obtenido el grafo.

C. El procesamiento de la primera escala, que es la que más nodos tiene y la que más información contiene.

D. Todas las anteriores.

10. ¿Qué algoritmo elegirías si tuvieras que ser interactivo e incorporar modificaciones por parte de un usuario experto humano?

A. Ninguno, este tipo de algoritmos no están pensados para ser interactivos con los seres humanos.

B. Graph-cuts y el crecimiento de regiones basados en semillas. Estos dos métodos pueden incorporar inicializaciones por parte de los expertos humanos.

C. Todos los métodos vistos en este tema, incluida la agregación multiescala.

D. Es mejor que el humano no intervenga en la segmentación.