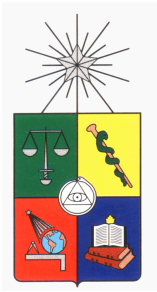


UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
DEPARTAMENTO DE INGENIERIA MATEMÁTICA

# SEGMENTACIÓN DE OBJETOS EN IMÁGENES DE MICROSCOPÍA MEDIANTE CONTORNOS ACTIVOS CON ADAPTABILIDAD TOPOLÓGICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN E INGENIERO CIVIL MATEMÁTICO

LUIS FELIPE OLMOS MARCHANT

PROFESOR GUÍA:
SR. ALEJANDRO MAASS SEPULVEDA

MIEMBROS DE LA COMISIÓN:
SRA. NANCY HITSCHFELD KAHLER
SR. STEFFEN HÄRTEL GRÜNDLER

SANTIAGO DE CHILE
JUNIO 2009

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN E
INGENIERO CIVIL MATEMÁTICO
POR: LUIS FELIPE OLMOS MARCHANT
FECHA: 22/06/2009
PROF. GUÍA: ALEJANDRO MAASS SEPULVEDA

# SEGMENTACIÓN DE OBJETOS EN IMÁGENES DE MICROSCOPÍA MEDIANTE CONTORNOS ACTIVOS CON ADAPTABILIDAD TOPOLÓGICA

El análisis y procesamiento de imágenes es un área con activo desarrollo en la matemática y ciencia de la computación. La motivación de esta memoria es desarrollar un método de segmentación de objetos biológicos en imágenes de microscopía que construya contornos de la más alta calidad y que requiera la menor interacción humana en su ejecución. Una familia de métodos exitosa en esta área está basada en un modelo de minimización de energía de curvas deformables bajo la influencia de fuerzas externas. En la presente memoria se trabajó en la implementación de uno de estos métodos y constó de dos partes.

En la primera parte se implementó el algoritmo *Topology Adaptive Snakes* o *T-Snakes*. La propiedad esencial de este método es que admite cambios topológicos en las curvas (fusiones, cortes y destrucción), lo que permite segmentar objetos con formas altamente complejas minimizando la interacción humana en el proceso.

En la segunda parte de este trabajo se implementó el algoritmo *Edge Preserving Gradient Vector Flow* (EPGVF) el cual genera los campos vectoriales que fuerzan a las curvas a evolucionar hacia los bordes de objetos en la imagen. Este método se comparó con su algoritmo predecesor, *Generalized Gradient Vector Flow* (GGVF), sin que EPGVF presentara ventajas convincentes en las imágenes probadas.

Los algoritmos implementados fueron aplicados para la segmentación de diversas estructucturas biológicas en imágenes de microscopía. La calidad de los resultados observados promete mejorar la cuantificación morfo-topológica de estas estructuras en diferentes áreas de la investigación y desarrollo.

# AGRADECIMIENTOS

Agradezco a mi gran familia y seres queridos, los cuales siempre me han proporcionado su apoyo incondicional y cariño.

También agradezco a Alejandro Maass por las recomendaciones dadas en el transcurso de esta memoria y doy un agradecimiento muy especial a Steffen Härtel por su disposición y guía en el desarrollo del presente trabajo.

Agradezco a todos los compañeros del laboratorio SCIAN en donde he trabajado esta memoria y también a todos mis compañeros del Departamento de Ingeniería Matemática, en especial a los integrantes de mi oficina "la cantina 434": Raul "Rául" Aliaga, Nicolás "N" Carreño, Ignacio "Chonak" Fantini, Cristóbal "Cris" Guzmán, Manuel "Monu" Larenas, Gustavo "Wuss" Navarro y Oscar "Puthh" Peredo.

Finalmente dedico este trabajo a la memoria de mi madre Ana María Marchant.

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

El análisis y procesamiento de imágenes es un área bien estudiada dentro de la matemática y ciencia de la computación. Esta disciplina ataca los problemas que existen en el proceso de extracción de información relevante de imágenes provenientes de observaciones en variadas áreas de la ciencia.

En el campo de la biofísica y la biología celular, el procesamiento de imágenes microscópicas ha permitido profundizar el estudio de estructuras biológicas, entregando información que no sería posible obtener en forma visual o a través de otra metodología. Ejemplos de esto son cuantificaciones de morfología, textura, topología y movimiento de células, tejidos y órganos ([HZKT+03], [HFM05], [DTHJM07]). Además, el procesamiento de imágenes ha contribuido en la obtención de información relevante en relación a reacciones enzimáticas que generan formas y estructuras únicas en distintos niveles de organización ([FHO02] , [HFM05], [DTHJM07]).

Como en todo experimento de observación, se requiere de procedimientos que permitan el mejor nivel de precisión con respecto a la cuantificación de estructuras celulares. Dentro del campo de la biofísica/biología celular, se estudian estructuras a nivel de microscopía óptica, donde la resolución en tres dimensiones (x,y,z) sólo se limita por la naturaleza intrínseca (física) de los fotones (ondas electromagnéticas). En este contexto aparece la microscopía confocal de fluorescencia, capaz de medir la fluorescencia de moléculas individuales a una resolución de 200-600 nm y estudiar la morfología, la topología y la dinámica de procesos biológicos a nivel sub-celular, celular y supra-celular.

El procesamiento de imágenes adquiridas a través de la microscopía confocal de fluorescencia ha cobrado relevancia para cuantificar la morfología, la topología y la dinámica de procesos en forma rigurosa. Esta memoria se concentra en el problema de la segmentación de estas imágenes, este es el procedimiento de extraer los contornos de regiones de interés (*region of interest*, ROIs). Para ello se aplicó una técnica perteneciente a la familia de modelos deformables, llamada *Contornos Activos* o *Snakes* [KWT88].

## 1.1. Contexto

### 1.1.1. Microscopía Confocal de Fluorescencia

La microscopía de fluorescencia se basa en la emisión y excitación (captación o absorción) de luz (ondas electromagnéticas). La microscopía confocal usa iluminación de barrido de rayo láser de alta precisión. Un sistema de espejos mueve al láser a través del objeto iluminando un punto (pixel) por vez. Se registran intensidades de emisión en cada punto del eje x-y del objeto observado y luego avanza en el eje z. De este modo se obtiene una pila de imágenes con información en 3 dimensiones (Ver Figura 1.1).

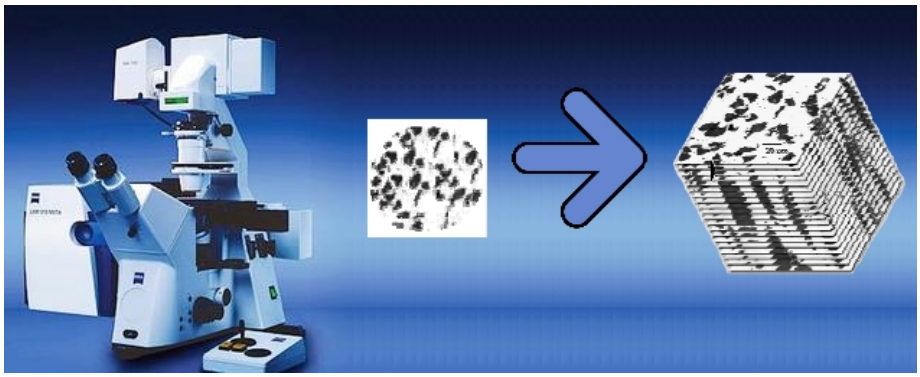


Figura 1.1: Esquema de la Microscopía Confocal, el microscopio obtiene imágenes del plano x-y en varias capas del eje z

Para poder identificar elementos de interés en las imágenes captadas, se les marca con estructuras submoleculares denominadas *fluoróforos*. Los fluoróforos tienen características bioquímicas que definen su adhesión a regiones dentro de las células que se quieran observar. La manera en que permiten identificar estas regiones (estructuras o organelos) es mediante la absorción de luz con ciertas longitudes de onda para luego emitir ondas con mayores longitudes que son captadas por el microscopio. De esta forma se obtienen imágenes en diferentes canales de color que marcan diferentes estructures celulares como la membrana y el citosól (Figura 1.2).

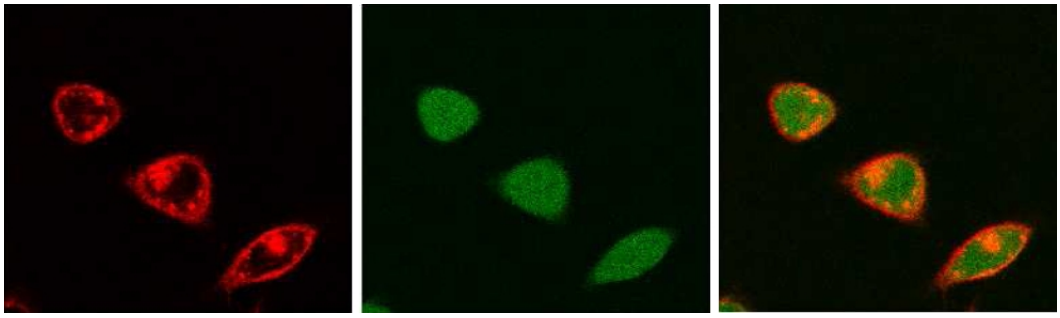


Figura 1.2: Visualización de una imagen obtenida por microscopía confocal en canales diferentes. La primera figura es el canal rojo que muestra membranas celulares marcadas con el fluoróforo $DiIC_{18}$. La segunda imagen muestra células marcadas con el fluoróforo calcina que marca el citosól. La tercera imagen muestra la superposición de ambos canales.

### 1.1.2. Contornos Activos (Snakes)

Segmentación es uno de los temas centrales en el procesamiento de imágenes que identifica diferentes objetos de interés (regions of interest, ROIs) dentro de la imagen. Una técnica para segmentar ROIs aplica la metodología de Contornos Activos o Snakes [KWT88](en este documento se ocupará el término snake o contorno indistinguiblemente).

La idea esencial de esta técnica es reducir el problema de encontrar el contorno de una ROI a encontrar la curva $\mathbf{x}(s) = (x(s), y(s))$ que resuelve el siguiente problema de minimización :

$$\min_{\mathbf{x} \in C^2([0,1];\mathbb{R}^2)} : \int_0^1 E_{int}(\mathbf{x}(s))ds$$

$E_{int}$ es la energía interna de la curva, depende de parámetros de elasticidad y rigidez. Las ecuaciones obtenidas para este sistema son posteriormente forzadas con fuerzas que guían la curva hacia los contornos de los objetos en la imagen.

## 1.2. Motivación

La motivación de esta memoria surge de la necesidad de LEO[1] (Laboratorio de Estudios Ontogénicos) en conjunto con SCIAN[2] (Scientific Image Analysis) de realizar análisis de imágenes obtenidas a través de microscopía confocal de fluorescencia.

[1] http://www.ontogenesis.cl
[2] http://www.scian.cl

Un ejemplo de lo anterior es el estudio de la morfogénesis[3] del órgano parapineal[4] del pez cebra (*Danio Rerio*). La evolución de este órgano parte con cerca de 20 células aparentemente desorganizadas que se van ordenando y migrando hacia un lado del cerebro [HJLC06], dando una asimetría al proceso (Ver Figura 1.3). Esta asimetría comienza por señales genéticas aún desconocidas y es clave en el desarrollo posterior del pez.

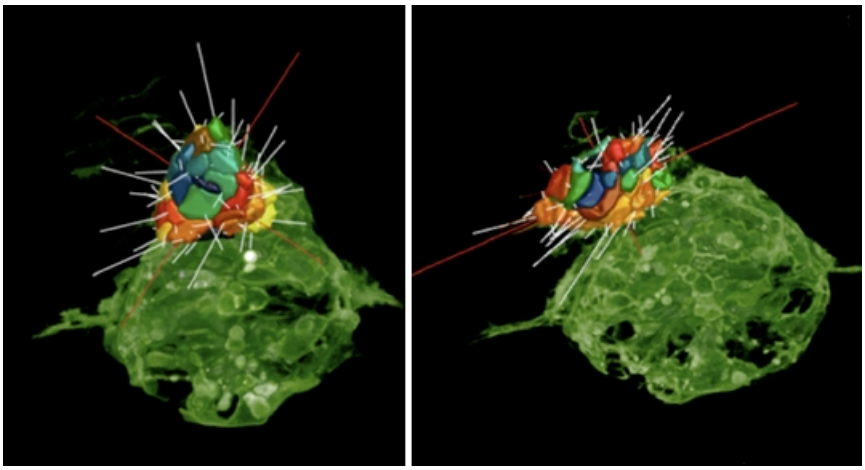


Figura 1.3: Morfogénesis del órgano parapineal. Las células migran desde el centro del cerebro hacia el lado izquierdo. Los modelos tridimensionales en la figura son construidos a partir de imágenes de microscopía confocal de fluorescencia. El primer cuadro es 34 horas post fecundación del pez, el segundo 38 horas post fecundación. Fuente : [Pal07]

Los modelos tridimensionales en la Figura 1.3, son construidos a partir de imágenes provenientes de microscopía confocal de fluorescencia usando el algoritmo de contornos activos implementado en [Jar07]. El procedimiento consiste primero en una segmentación en dos dimensiones y a partir de esto se construye un modelo tresdimensional. Sin embargo a veces la segmentación no es suficiente debido a la complejidad de las imágenes obtenidas y la segmentación debe ser hecha a mano en tabletas gráficas. Por lo tanto es de interés mejorar los métodos actualmente usados para minimizar la interacción humana en el proceso de segmentación, sin perder calidad en los modelos obtenidos.

## 1.3. Objetivos

### 1.3.1. Objetivo General

Aumentar la automatización en el proceso de segmentación de objetos en imágenes de microscopía confocal de fluorescencia mediante la mejora del algoritmo de contornos activos actualmente usado en SCIAN.

[3]Procesos de distribución de células a lo largo del desarrollo embrionario. Estos dan forma a tejidos, órganos y a la anatomía de un organismo

[4]Órgano que forma parte del cerebro y se desarrolla de forma muy temprana en los embriones del pez

### 1.3.2. Objetivos Específicos

- Implementar el algoritmo de *Edge Preserving Gradient Vector Flow* para mejorar el campo de gradientes actualmente usado.
- Implementar el algoritmo de *Topology Adaptive Snakes* y usarlo para obtener un método automatizado de segmentación.
- Realizar lo anterior de manera eficiente usando el lenguaje C y programación paralela donde sea posible.
- Incorporar las implementaciones al software de SCIAN.

# Capítulo 2

# Antecedentes

## 2.1. Contornos Activos (Snakes)

La idea general del método de Contornos Activos es el de crear un sistema "físico" en la imagen en el cual una cuerda (i.e. un continuo unidimiensional que posee elasticidad y rigidez) es llevada a bordes de objetos de interés contenidos en la imagen.

La formalización de este método primero considera un sistema en el cual la cuerda no está sometida a ningún tipo de cuerdas externas, así se obtienen las ecuaciones de la dinámica intrínseca de la curva (esto es, la que depende sólo de la elasticidad y rigidez). Posteriormente el sistema (homogéneo) obtenido es forzado con fuerzas construidas a partir de la imagen y que se encargan que la dinámica del sistema lleve la cuerda a los bordes de objetos relevantes en la imagen.

Se considera de aquí en adelante $I : \mathbb{R}^2 \to \mathbb{R}$ la función de intensidad de una imagen.

### 2.1.1. Teoría

Primero se plantea la dinámica intrínseca (i.e. no dependiente de la imagen) de una curva mediante un principio de minimización de energía:

$$\min_{\mathbf{x} \in C^2([0,1];\mathbb{R}^2)} : \int_0^1 E_{int}(\mathbf{x}(s))ds$$

Donde $C^2([0,1];\mathbb{R}^2)$ es el conjunto de funciones dos veces continuamente derivables de $[0,1]$ en $\mathbb{R}^2$. A continuación se explica en detalle la energía interna $E_{int}$.

#### Energía Interna

La energía interna $E_{int}$ consta de dos términos, el primero es la energía de elasticidad:

$$\alpha(\mathbf{x}(s))|\mathbf{x}'(s)|^2$$

donde $\alpha$ es la elasticidad local. A mayor $\alpha(\mathbf{x}(s))$ mayor contracción tendrá la curva en el punto $x(s)$. El segundo es la energía de tensión:

$$\beta(\mathbf{x}(s))|\mathbf{x}''(s)|^2$$

$\beta(\mathbf{x}(s))$ es la tensión local. A mayor $\beta(\mathbf{x}(s))$ el contorno tendrá menor capacidad de curvatura en el punto $\mathbf{x}(s)$.

En el desarrollo de esta memoria se considerarán elasticidades y tensiones homogéneas en el contorno, o sea que las funciones $\alpha$ y $\beta$ serán consideradas constantes:

$$\alpha(\mathbf{x}(s)) \equiv \alpha \quad \forall s \in [0,1]$$
$$\beta(\mathbf{x}(s)) \equiv \beta \quad \forall s \in [0,1]$$

**Condición de primer orden del problema de optimización**

El funcional a optimizar tiene la siguiente forma

$$\underset{\mathbf{x} \in C^2([0,1];\mathbb{R}^2)}{\mathbf{min}} \quad : \int_0^1 \alpha(\mathbf{x}(s))|\mathbf{x}'(s)|^2 + \beta(\mathbf{x}(s))|\mathbf{x}''(s)|^2 \, ds$$

Un punto estacionario del funcional satisface las ecuaciones de Euler-Lagrange (una versión un poco más general, ver el Teorema A.1.2) de lo cual se deduce el siguiente sistema desacoplado de ecuaciones diferenciales:

$$\beta\mathbf{x}^{(4)}(s) - \alpha\mathbf{x}''(s) = 0$$

Este problema también se puede pensar como encontrar la solución en régimen estacionario de:

$$-\gamma\frac{\partial \mathbf{x}}{\partial t}(t,s) + \beta\frac{\partial^4 \mathbf{x}}{\partial s^4}(t,s) - \alpha\frac{\partial \mathbf{x}^2}{\partial s^2}(t,s) = 0 \tag{2.1}$$

en donde $\gamma$ se puede ver como coeficiente de viscosidad del medio donde la snake está evolucionando.

**Fuerzas Externas**

Las fuerzas externas son construidas con datos provenientes de la imagen, son llamadas comúnmente *fuerzas de imagen*. Existen variados tipos de estas fuerzas, éstas se encargan de llevar el contorno a zonas de interés de la imagen como por ejemplo bordes o zonas de intensidad alta. En el presente trabajo se utilizarán dos: de inflación y de flujo vectorial de gradiente tal como se usa en [XP98].

- **Fuerza de Inflación** :

  Sea $\mathbf{n}$ la normal unitaria externa de la curva. Se define la fuerza de inflación $\boldsymbol{\rho}$ como:

  $$\boldsymbol{\rho}(\mathbf{x}(s)) = qF(I(\mathbf{x}(s)))\mathbf{n}(\mathbf{x}(s))$$

  donde $q$ es un parámetro que ajusta la intensidad de la fuerza y:

  $$F(x,y) = \begin{cases} 1 & \text{si } I(x,y) > T, \\ -1 & \text{si } I(x,y) \leq T. \end{cases}$$

  $T$ es un parámetro de umbral. Esta fuerza efectivamente infla el contorno si éste está situado en una zona de intensidad mayor que $T$ y lo desinfla de lo contrario (ver figura 2.1).

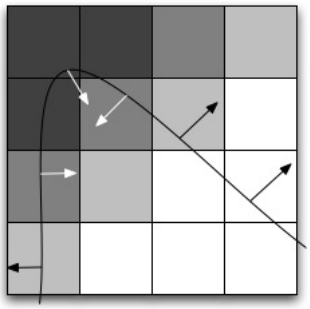


Figura 2.1: Fuerza de Inflación en una snake

- **Fuerza de flujo vectorial de gradiente** :

  Esta fuerza atrae el contorno a zonas de gradiente alto en la imagen, se denotará $\mathbf{f}$. Se ahondará con más detalle en la Sección 2.3.

### 2.1.2. Forzamiento del Sistema Homogéneo

Debido a que no se puede asumir que las fuerzas externas son conservativas no se pueden incorporar directamente al problema de minimización de energía. Una manera clásica

de resolver esto es forzar el sistema homogéneo con estas fuerzas, esto es simplemente agregar estos términos al lado derecho de la ecuación 2.1:

$$-\frac{\partial \mathbf{x}}{\partial t}(t,s) + \alpha \frac{\partial^4 \mathbf{x}}{\partial s^4}(t,s) - \beta \frac{\partial \mathbf{x}^2}{\partial s^2}(t,s) = -\mathbf{f}(\mathbf{x}(t,s)) - \boldsymbol{\rho}(\mathbf{x}(t,s))$$

### 2.1.3. Resolución Numérica

La curva es discretizada en el tiempo y en el parámetro $s$, además es interpolada poligonalmente. Esta manera de atacar el problema es conocida como *snakes paramétricas* [1]. En cada paso temporal $t^n := n\Delta t$ la curva se representa por un conjunto de puntos $\{\mathbf{x}_i^n\}_{i=0}^m = \left\{ \begin{pmatrix} x_i^n \\ y_i^n \end{pmatrix} \right\}_{i=0}^m$. Definimos:

$$\mathbf{f}_i^n = \mathbf{f}(\mathbf{x}_i^n) \qquad \boldsymbol{\rho}_i^n = \boldsymbol{\rho}(\mathbf{x}_i^n)$$

$$\mathbf{w}_i^n = \mathbf{x}_{i-1}^n - 2\mathbf{x}_i^n + \mathbf{x}_{i+1}^n$$
$$\mathbf{z}_i^n = \mathbf{w}_{i-1}^n - 2\mathbf{w}_i^n + \mathbf{w}_{i+1}^n$$

$\mathbf{w}_i^n$ y $\mathbf{z}_i^n$ son aproximaciones de la segunda y cuarta derivada espacial de $\mathbf{x}$ respectivamente cuando el paso en $s$ es unitario (en este caso se puede asumir que el paso es unitario, ver apéndice). Se plantea el siguiente esquema de diferencias finitas *forward*:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \frac{\Delta t}{\gamma}\left(\alpha \mathbf{z}_i^n - \beta \mathbf{w}_i^n + \boldsymbol{\rho}_i + \mathbf{f}_i\right)$$

La intensidad de imagen ($I(x,y)$), necesaria para la fuerza de inflación, es interpolada bilinealmente para poder obtener su valor en coordenadas no enteras, también se hace lo mismo para cada una de las coordenadas de las fuerzas externas.

## 2.2. T-Snakes

En la resolución numérica del problema de snakes hay dos problemas adicionales que deben ser atendidos:

- **Cambios Topológicos**: En el apronte de snakes paramétricas se asume que la curva es continua, luego no pueden haber cambios topológicos (fusiones, divisiones, etc.) en una snake (ver Figura 2.2). Ésta capacidad es relevante si se quiere capturar ROIs no conexas.

[1] La otra forma es representar la snake como una curva de nivel de un función de dimensión mayor, ésta manera es conocida como *level set snakes.*

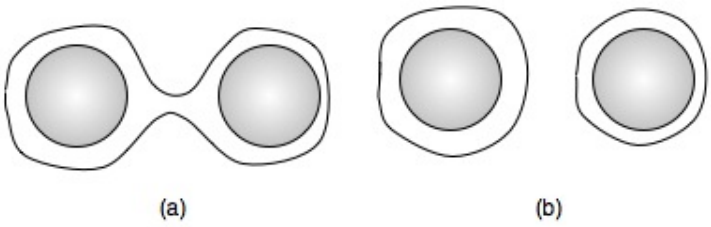



Figura 2.2: Ejemplo de un cambio topológico: si no se admiten estos cambios el modelo resultante de la evolución de la snake no será una segmentación apropiada de la imagen quedando como en la figura (a). En (b) se observa la evolución deseada

- **Muestreo**: El tiempo de procesamiento en la evolución de la snake es proporcional al número de puntos donde está siendo muestreada, por lo que el sobremuestreo debe ser evitado. Por otro lado si hay submuestreo de la snake, el modelo no está representando fielmente la curva . Por ello se requiere una estrategia robusta de muestreo de la snake para asegurar que la evolución del modelo sea fiel sin caer en el sobremuestreo (ver Figura 2.3).

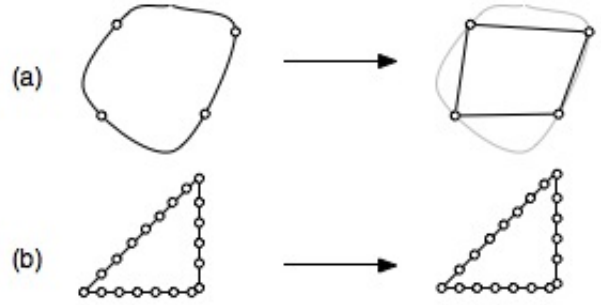



Figura 2.3: Problemas con el muestreo: Si hay submuestreo la representación de la curva no es lo suficientemente precisa (a). Si hay sobremuestreo no hay mucha ganancia y se necesita más tiempo de procesamiento (b).

En [MT00] se atacan ambos problemas simultaneamente con el esquema de *topology adaptive snakes* (snakes con adaptabilidad topológica), también conocidas como *T-snakes*. A continuación se explica esta metodología de manera general, los detalles técnicos se verán en el Capítulo 3.

### 2.2.1. Grilla

El muestreo de la snake es vía la intersección de ésta con una grilla, los puntos de intersección serán la representación de la snake. También se almacena en los nodos de la grilla un 1 o 0 dependiendo si este está dentro o fuera del área que encierra la snake (ver Figura 2.4).

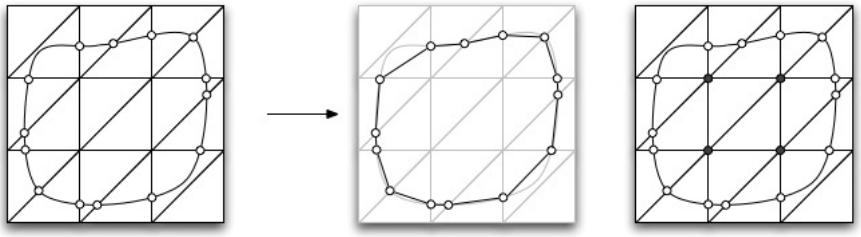


Figura 2.4: Uso de la grilla en T-snakes : Los puntos de intersección de la curva con la grilla forman el polígono que la representará, además ésta almacena un indicador de los nodos interiores a la snake (abajo)

La fineza de la grilla es un parámetro del modelo, debe ser lo suficientemente fina para que se tenga una representación razonable de la snake, en los ejemplos de [MT00] se usan del orden de la resolución de la imagen.

En este trabajo se usó una grilla *simplicial* (compuesta de simplex) tal como se realizó en [MT00].

### 2.2.2. Remuestreo Iterativo

El remuestreo se realiza cada $M$ (parámetro ajustable) pasos de la iteración en el tiempo del esquema numérico descrito en la Sección 2.1.3, los autores en [MT00] sugieren $M$ entre 5 y 10. A estas $M$ iteraciones se les denomina *paso de deformación*.

El remuestreo consta de dos fases, las cuales se detallan a continuación

#### Fase Uno

Después de hacer el paso de deformación los puntos de la curva casi seguramente no serán puntos de intersección con la grilla, para el remuestreo es necesario computar para cada segmento su intersección con la grilla. Esto se hace ocupando la siguiente estrategia de tipo *ray-casting*, la idea básica es lanzar un rayo desde un extremo del segmento e ir encontrando las intersecciones con la grilla hasta alcanzar el otro extremo (ver Algoritmo 1 y Figura 2.5).

Cada arco intersectado tendrá un *signo* que es $1$ si el primer vértice del arco cae dentro de la snake y $-1$ si no. El procedimiento ADD INTERSECTION se encarga de guardar la intersección si es que en el arco no se habían registrado intersecciones, o si se había hecho, sus signos son iguales. Esto servirá para descartar intersecciones cuando hay cambios topológicos (pues en este caso necesariamente una snake intersecta dos veces o más un arco). Además de ser guardada la intersección, el nodo que cae dentro de la snake se guarda en una cola para ser procesado posteriormente en la fase dos. El algoritmo 2 describe en detalle el procedimiento (también ver Figura 2.6).

**Algoritmo 1** Procesamiento de Intersecciones de un Segmento

```
1:  procedure GRID INTERSECTIONS(s, g)                ▷ Segmento s, Grilla g
2:      p_1 ← extremo 1 del segmento s
3:      p_2 ← extremo 2 del segmento s
4:      a_1 ← arco de g que intersecta a s más cercano a p_1
5:      a_2 ← arco de g que intersecta a s más cercano a p_2
6:      p ← punto de intersección en a_1
7:      a ← a_1
8:      repeat
9:          ADD INTERSECTION(a, p)
10:         a ← arco de g que intersecta a s más cercano a p en la dirección de s
11:         p ← punto de intersección encontrado en el paso anterior
12:     until a = a_2
13:     return i
14: end procedure
```

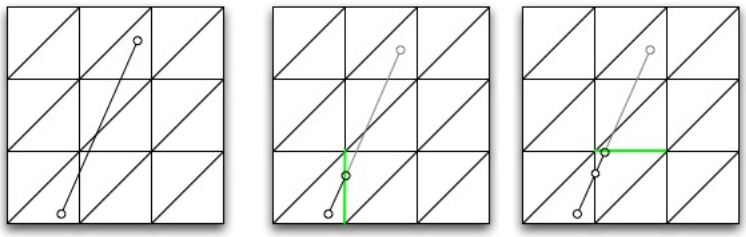


Figura 2.5: Algoritmo de intersección de un segmento con la grilla: Primero se busca el punto de intersección más cercano a un extremo (segundo cuadro) y luego se prosigue buscando el siguiente punto de intersección en la dirección del segmento (tercer cuadro)

**Algoritmo 2** Procesamiento de una intersección

```
1:  procedure ADD INTERSECTION(a, p)          ▷ Arco a, Punto de Intersección p
2:      sgn ← signo del primer vértice de a
3:      v ← vertice del a que queda adentro de la snake
4:      if a.sgn = 0 or a.sgn = sgn then   ▷ Si no había intersección o era del mismo signo
5:          a.sgn = sgn
6:          a.int = p
7:          ENQUEUE(v)
8:      else
9:          a.sgn = 0
10:         a.int = NULL
11:     end if
12: end procedure
```

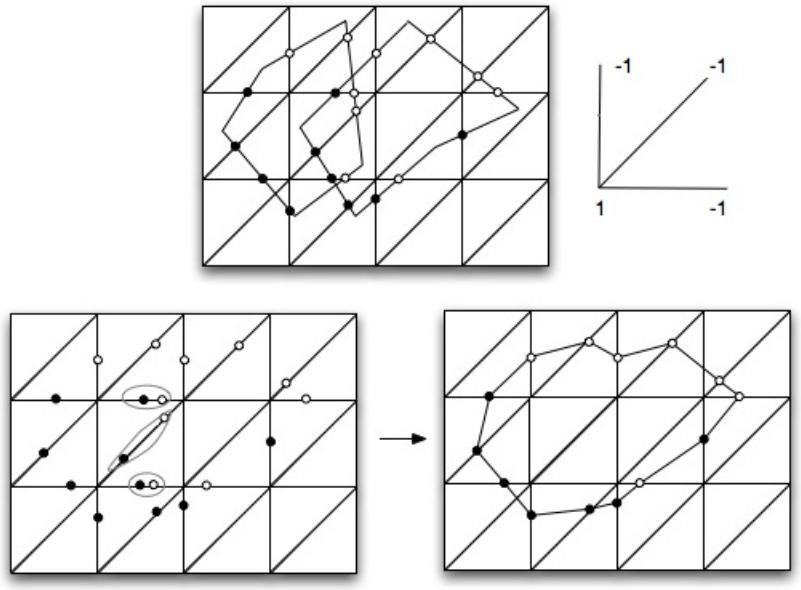



Figura 2.6: Un ejemplo del procesamiento de un cambio topológico: Dos polígonos se intersectan (arriba izquierda), las intersecciones con la grilla están pintadas negras si tienen signo $-1$ y blancas de lo contrario. El modo de seleccionar el signo (arriba derecha) es de acuerdo si el primer vértice de un arco cae o no dentro de la snake. Los arcos que tienen más de una intersección y éstas son de signo opuesto se descartan (abajo izquierda), para posteriormente en la fase dos realizar la reconstrucción apropiada del cambio topológico (abajo derecha).

### Fase Dos

En esta fase se actualiza la indicatriz del interior de la snake. Para ello los nodos encolados en la fase dos (sec. 2.2.2) son usados como semillas para un algoritmo de llenado. Este algoritmo es un recorrido *breadth-first* que no sigue si es que encuentra un nodo pintado o un arco con una intersección (ver Figura 2.7).

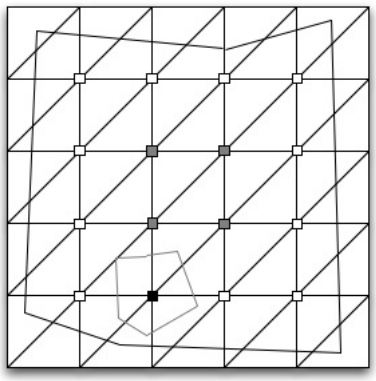


Figura 2.7: Actualización de la indicatriz, los nodos encolados están representados por cuadrados blancos, los nodos que ya habían sido marcados como interior por negro y los que se llenarán con el algoritmo *breadth first search* están marcados en gris.

### Rastreo de Contornos

Una vez terminada la fase dos se procede a descartar intersecciones invalidas, que son las que están en arcos con ambos extremos dentro de la snake, y con los que quedan se realiza un rastreo de los contornos detallado en el Algoritmo 3.

**Algoritmo 3** Rastreo de contornos

```
1:  procedure CONTOUR TRACE(q)        ▷ q cola de vértices de la grilla que son borde
2:      v ← POP(q)
3:      v_first ← v
4:      a ← algún arco adyacente a v que tenga una intersección
5:      a_first ← a
6:      repeat
7:          ADD SNAKE(a.inter)                          ▷ Agrega el punto a la snake
8:          a_aux ← NEXT(a)
9:          if a_aux.inter = NULL then
10:             (v, a) ← CHANGE VERTEX(v, a)
11:                                  ▷ Entrega el proximo vértice y arco a procesar
12:         else
13:             a ← a_aux
14:         end if
15:     until a = a_first and v = v_first
16: end procedure
```

### 2.2.3. Algoritmo T-Snakes

Juntando los procedimientos anteriores se puede dar un esquema general del algoritmo de T-snakes (ver algoritmo 4)

**Algoritmo 4** Algoritmo de T-Snakes

```
1:  procedure T-SNAKES(s_ini, d, r, it, M)
2:                                          ▷ s_ini snake inicial, d datos de la imagen,
3:                                  ▷ r resolución de la grilla, it numero de iteraciones,
4:                                          ▷ M, número de iteraciones de deformación
5:      g ← NEW GRID(r)
6:      q ← NEW QUEUE()
7:      PHASE 1(g, s_ini)                   ▷ Primero se proyecta la snake inicial sobre la grilla
8:      PHASE 2(g, q)
9:      snakes.add(CONTOUR TRACE(q) )               ▷ snakes es un contenedor de snakes
10:     for i = 0, i < it, i++ do
11:         for j = 0, j < M, j++ do
12:             for all s ∈ snakes do
13:                 DEFORM(s)
14:             end for
15:         end for
16:         for all s ∈ snakes do
17:             PHASE 1(g, s)
18:             PHASE 2(g, q)
19:             while q tenga elementos do
20:                 snakes.add(CONTOUR TRACE(q) )   ▷ snakes es un contenedor de snakes
21:             end while
22:         end for
23:     end for
24: end procedure
```

## 2.3. Flujo Vectorial de Gradiente

La idea esencial del Flujo Vectorial de Gradiente (*Gradient Vector Flow* ó GVF) es la de generar un campo de fuerzas que guíe la snake a zonas de gradiente de intensidad alto, que son las zonas de la imagen donde hay bordes.

Una primera aproximación para obtener un campo con esas características es usar:

$$\mathbf{u} = \nabla|\nabla I|^2$$

donde $I$ es la función de intensidad de la imagen. Esta función efectivamente es un campo atractivo hacia zonas de intensidad de gradiente alto, sin embargo tiene muy poco rango de captura, por lo que la snake no será atraída al contorno si es inicializada muy lejos de él (ver Figura 2.8).

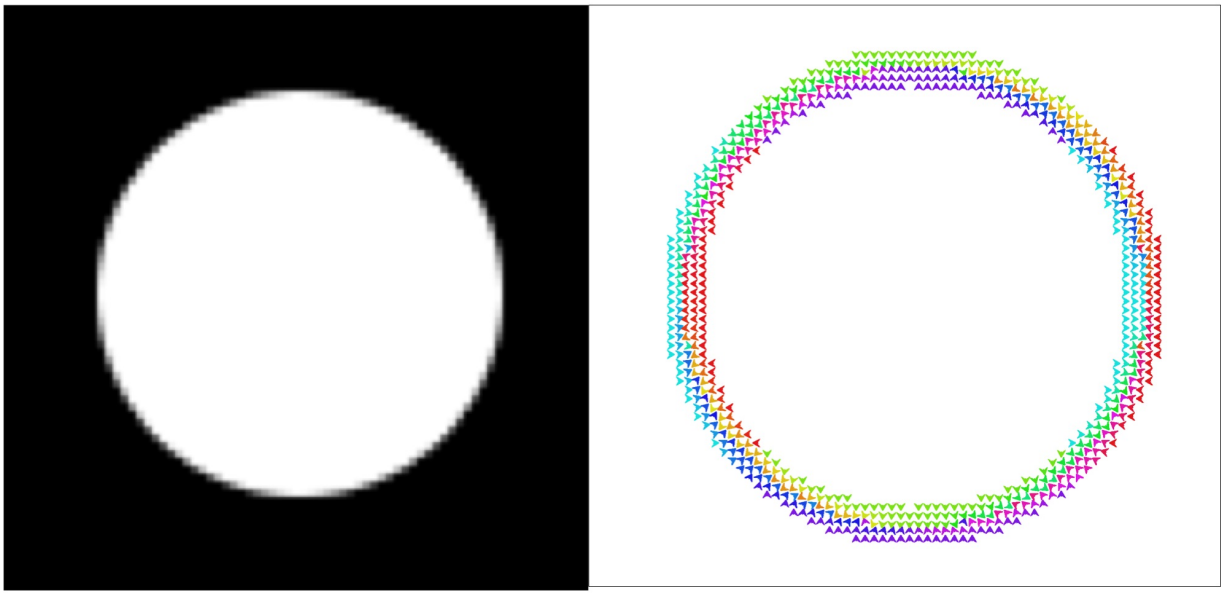


Figura 2.8: Ejemplo del campo $\nabla|\nabla I|^2$ de la imagen de un círculo. Vectores coloreados según dirección.

Una alternativa a este campo es convolucionar $I$ con una función gaussiana:

$$\mathbf{u} = \nabla|\nabla I * G_\sigma|^2$$
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Donde $\sigma > 0$. Esto hace que el gradiente se esparza dependiendo del tamaño de $\sigma$, a mayor sigma mayor esparcimiento. Esto es aún insuficiente para aumentar el rango de captura, sobre todo en imágenes de mayor tamaño (ver Figura 2.9).

GVF es una técnica que logra solucionar el problema anterior, a continuación se explica una generalización de GVF: GGVF (*Generalized Gradient Vector Flow*)[XP98].

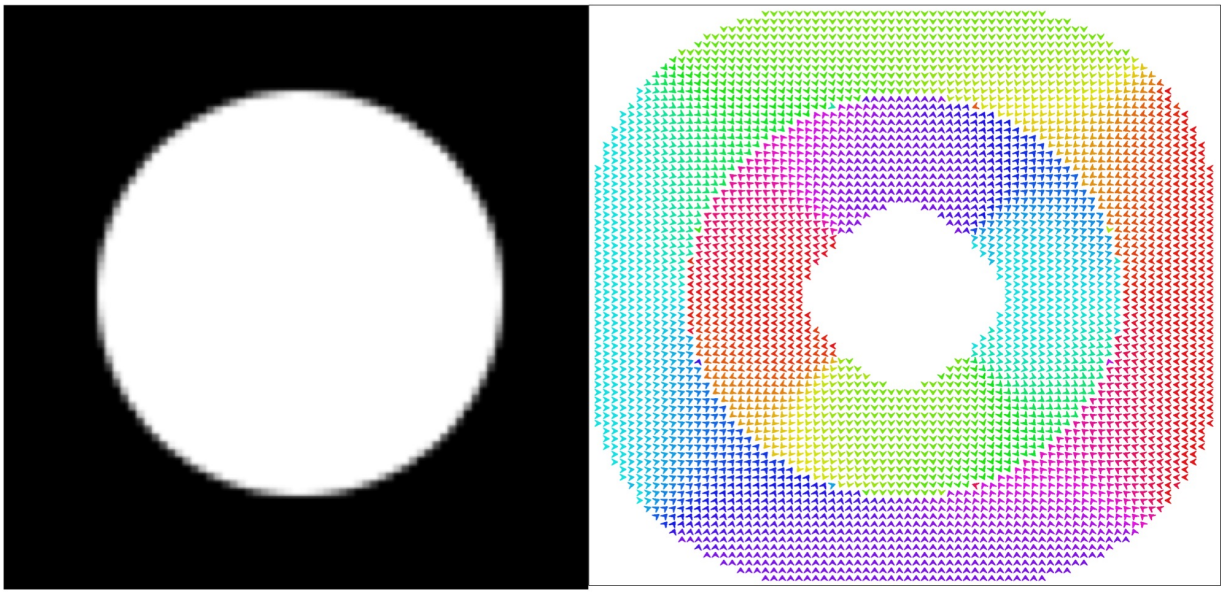


Figura 2.9: Ejemplo del campo $\nabla|\nabla I * G_1|^2$ de la imagen de un círculo. Vectores coloreados según dirección.

La idea es hacer un flujo de los campos mencionados anteriormente hacia zonas donde el gradiente es nulo o muy bajo. Para ello el campo buscado es una solución $\mathbf{u} = (u, v)$ del siguiente problema de minimización :

$$\min_{\mathbf{u}\in C^2(\Omega;\mathbb{R}^2)} \int_\Omega g(|\nabla f|)\left(|\nabla u|^2 + |\nabla v|^2\right) + h(|\nabla f|)\left(|u - f_x|^2 + |v - f_y|^2\right)\, dx$$

Donde :

- $\Omega \subseteq \mathbb{R}^2$ es el abierto que define la imagen.
- $f$ se denomina *mapa de borde* (*edge map*) y se puede tomar como $f = |\nabla I|^2$ ó $f = |\nabla I * G_\sigma|^2$ para algún $\sigma > 0$.
- $g$ y $h$ son funciones de clase $C(\Omega; [0, 1])$. Estas son funciones de peso, $h$ debe valer cero donde $|\nabla I|$ es pequeño y $g$ que es el complemento de $h$ se toma como $g = 1 - h$.

La zona donde $h$ no es cero es donde el gradiente es de tamaño considerable (donde hay bordes), en esa parte la minimización obligará a hacer más pequeños los términos $|u - f_x|^2$ y $|u - f_y|^2$, por tanto la solución $\mathbf{u}$ tenderá a parecerse a $\nabla f$ en esa parte de la imagen.

Por otro lado en la parte donde $g$ no es cero la minimización de $|\nabla u|^2$ y de $|\nabla v|^2$ tendrá un efecto difusivo (se puede pensar que es la minimización de energía calórica). Lo que hará difundir las direcciones de la zona de borde (donde $h$ es distinto de cero).

Este efecto difusivo se ve al determinar el sistema de ecuaciones en derivadas parciales de la condición de primer orden para una solución de este problema (ver Teorema A.2.2 en el caso $\mu = 0$) el cual es un sistema desacoplado de ecuaciones de advección-difusión:

$$\nabla g \cdot \nabla u + g\Delta u - h(u - f_x) = 0$$
$$\nabla g \cdot \nabla v + g\Delta v - h(v - f_y) = 0$$

Una mejora a GGVF son los denominados campos de flujo vectorial de gradiente que preservan bordes (*Edge Preserving Gradient Vector Flow* ó EPGVF) [LLF05]. Esta técnica ataca el problema de que los bordes de mayor intensidad al difundirse sobrepasan un borde que tiene intensidad menor. La mejora consiste en agregar un término que suaviza la solución en las direcciones paralelas a los bordes (y no en las perpendiculares a estas). El problema de minimización es similar al de GGVF:

$$\min_{\mathbf{u}\in C^2(\Omega;\mathbb{R}^2)} : \int_\Omega g(|\nabla f|)\left(|\nabla u|^2 + |\nabla v|^2\right) + h(|\nabla f|)\left(\mu\left((\nabla u \cdot \mathbf{p})^2 + (\nabla v \cdot \mathbf{p})^2\right) + |u - f_x|^2 + |v - f_y|^2\right) dx$$

donde :

$$\mathbf{p} = \begin{pmatrix} p^1 \\ p^2 \end{pmatrix} = \frac{1}{\sqrt{I_x^2 + I_y^2}} \begin{pmatrix} -I_y \\ I_x \end{pmatrix} \tag{2.2}$$

El campo $\mathbf{p}$ es paralelo a la dirección de los bordes, por lo que el término $\mu\left((\nabla u \cdot \mathbf{p})^2 + (\nabla v \cdot \mathbf{p})^2\right)$ en el funcional $J$ es la norma al cuadrado de la derivada direccional de $(u, v)$ en la dirección $\mathbf{p}$.

Una solución $(u, v)$ satisface el siguiente sistema de ecuaciones desacoplado (ver Teorema A.2.2 para la deducción):

$$\nabla g \cdot \nabla u + g\Delta u + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial u}{\partial x_j} + hp^ip^j\frac{\partial^2 u}{\partial x_i \partial x_j}\right) - h(u - f_x) = 0$$
$$\nabla g \cdot \nabla v + g\Delta v + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial v}{\partial x_j} + hp^ip^j\frac{\partial^2 v}{\partial x_i \partial x_j}\right) - h(v - f_y) = 0$$

La solución del sistema anterior puede verse como la solución en régimen permanente del sistema

$$\frac{\partial u}{\partial t} = \nabla g \cdot \nabla u + g\Delta u + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial u}{\partial x_j} + hp^ip^j\frac{\partial^2 u}{\partial x_i \partial x_j}\right) - h(u - f_x) \tag{2.3}$$

$$\frac{\partial v}{\partial t} = \nabla g \cdot \nabla v + g\Delta v + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial v}{\partial x_j} + hp^ip^j\frac{\partial^2 v}{\partial x_i \partial x_j}\right) - h(v - f_y) \tag{2.4}$$

### 2.3.1. Resolución Numérica

Como el sistema 2.3-2.4 es desacoplado y ambas ecuaciones son técnicamente iguales basta ver la resolución de una de ellas, digamos (2.3).

La alternativa escogida para la resolución numérica es un método de tipo Euler, es decir, un esquema de diferencias finitas *forward*.

Se asume de ahora en adelante que $\Omega$ es un rectángulo, se discretiza en rectángulos de lados $\Delta x$ y $\Delta y$. En el tiempo se discretiza en pasos $\Delta t$. El desarrollo completo se encuentra en la Sección B.1. Se definen:

$$
\begin{aligned}
x_i &= i\Delta x & g_{ij} &= g(|\nabla f(x_i, y_j)|) \\
y_j &= j\Delta y & h_{ij} &= h(|\nabla f(x_i, y_j)|) \\
t_n &= n\Delta t & p^1_{ij} &= p^1(x_i, y_j) \\
u^n_{ij} &= u(x_i, y_j, t_n) & p^2_{ij} &= p^2(x_i, y_j) \\
f_{ij} &= f(x_i, y_j)
\end{aligned}
$$

Con estas definiciones la iteración para la resolución numérica de la ecuación 2.3 es :

$$
\begin{aligned}
u_{ij}^{n+1} = u_{ij}^{n} + \Delta t \Bigg[ & \frac{(g_{i+1j} - g_{i-1j}) - (u_{i+1j}^{n} - u_{i-1j}^{n})}{4\Delta x^2} + \frac{(g_{ij-1} - g_{ij-1}) - (u_{ij+1}^{n} - u_{ij-1}^{n})}{4\Delta y^2} \\
& + g_{ij} \left( \frac{u_{i+1j}^{n} - 2u_{ij} + u_{i-1j}^{n}}{\Delta x^2} + \frac{u_{ij+1}^{n} - 2u_{ij} + u_{ij+1}^{n}}{\Delta y^2} \right) \\
& + \mu \Bigg\{ h_{ij}(p_{ij}^{1})^2 \frac{u_{i+1j}^{n} - 2u_{ij}^{n} + u_{i-1j}^{n}}{\Delta x^2} + h_{ij}(p_{ij}^{2})^2 \frac{u_{ij+1}^{n} - 2u_{ij}^{n} + u_{ij-1}^{n}}{\Delta y^2} \\
& + 2h_{ij}p_{ij}^{1}p_{ij}^{2} \frac{u_{i+1j+1}^{n} - u_{i+1j-1}^{n} + u_{i-1j-1}^{n} - u_{i-1j+1}^{n}}{2\Delta x \Delta y} \\
& + \left( \frac{h_{i+1j} - h_{i-1j}}{2\Delta x}(p_{ij}^{1})^2 + 2h_{ij} \frac{p_{i+1j}^{1} - p_{i-1j}^{1}}{2\Delta x} p_{ij}^{1} \right) \frac{u_{i+1j}^{n} - u_{i-1j}^{n}}{2\Delta x} \\
& + \left( \frac{h_{ij+1} - h_{ij-1}}{2\Delta y}(p_{ij}^{2})^2 + 2h_{ij} \frac{p_{ij+1}^{2} - p_{ij-1}^{2}}{2\Delta y} p_{ij}^{2} \right) \frac{u_{ij+1}^{n} - u_{ij-1}^{n}}{2\Delta y} \\
& + \left( \frac{h_{i+1j} - h_{i-1j}}{2\Delta x} p_{ij}^{1}p_{ij}^{2} + h_{ij} \frac{p_{i+1j}^{1} - p_{i-1j}^{1}}{2\Delta x} p_{ij}^{2} + h_{ij}p_{ij}^{1} \frac{p_{i+1j}^{2} - p_{i-1j}^{2}}{2\Delta x} \right) \frac{u_{ij+1}^{n} - u_{ij-1}^{n}}{2\Delta y} \\
& + \left( \frac{h_{ij+1} - h_{ij-1}}{2\Delta y} p_{ij}^{1}p_{ij}^{2} + h_{ij} \frac{p_{ij+1}^{1} - p_{ij-1}^{1}}{2\Delta y} p_{ij}^{2} + h_{ij}p_{ij}^{1} \frac{p_{ij+1}^{2} - p_{ij-1}^{2}}{2\Delta y} \right) \frac{u_{i+1j}^{n} - u_{i-1j}^{n}}{2\Delta x} \Bigg\} \\
& - h_{ij} \left( u_{ij}^{n} - \frac{f_{i+1j} - f_{i-1j}}{2\Delta x} \right) \Bigg]
\end{aligned}
$$

La resolución numérica de la ecuación 2.4 es completamente análoga a lo anterior.

# Capítulo 3

# Implementación

En esta sección se explica con un poco más de detalle técnico que en el capítulo de antecedentes (2) algunos aspectos relevantes de la implementación de los algoritmos de EPGVF y T-snakes.

## 3.1. Implementaciones Previas

La técnica de snakes ya ha sido implementada en SCIAN [Jar07]. Específicamente en aquel trabajo se implementó:

- El campo GGVF (ver Sección 2.3).
- El esquema de snakes clásico (ver Sección 2.1) en dos y tres dimensiones.
- Un esquema de reparametrización de la snakes para el caso dos dimensional. Este esquema consiste en mantener la cantidad de puntos por unidad de largo dentro de un rango fijo, si se sale de este rango entonces se remuestrea la snake para que esté en el intervalo deseado.
- Un esquema de interpolación de la snake es vía splines.

## 3.2. Herramientas Usadas

### 3.2.1. Software

Se usaron las siguientes herramientas de software

- El lenguaje C para la mayor parte de los algoritmos implementados.

- IDL 7.0 (Interactive Data Language). Este es el lenguaje en el cual está implementado SCIAN. En este trabajo se usó principalmente para visualizar los datos y para incorporar los algoritmos implementados a SCIAN en forma de librerías dinámicas (DLL's).
- OpenMP, librería para programación paralela en procesadores multinúcleo.

### 3.2.2. Hardware

Se usaron los siguientes computadores a lo largo de esta memoria.

- MacBook, de procesador Intel Core 2 Duo 2.4 GHz, 2 GB de RAM
- PC, de procesador Intel Core 2 Quad 2.4 GHz, 8 GB de RAM

## 3.3. Algoritmos Implementados

En esta sección se ven detalles técnicos relevantes de los algoritmos implementados.

### 3.3.1. EPGVF

#### Mapa de Borde

El mapa de borde $f$ escogido en la implementación es:

$$f(x,y) = |\nabla I * G_\sigma(x,y)|^2$$

con $\sigma > 0$ ajustable.

#### Funciones de Peso

Consideramos primeramente la función de Heaviside $H$ definida como:

$$H(t) = \begin{cases} 1 & t > 0 \\ \frac{1}{2} & t = 0 \\ 0 & t < 0 \end{cases}$$

Las funciones de peso estarán basadas en una aproximación suave de la función $H(t-\tau)$, donde $0 < \tau < 1$ es un parámetro ajustable. Para lograr el suavizado se relaja la función mediante otro parámetro $\delta$ que satisface $0 < \delta < \tau$ y $\tau + \delta < 1$ de forma que la

función vaya de 0 a 1 de manera continua en el intervalo $[\tau - \delta, \tau + \delta]$ . Más aún exigimos que la derivada sea continua. Se define entonces la función de Heaviside esparcida como $\mathcal{H}_{\tau,\delta}$ como (ver también figura 3.1) :

$$\mathcal{H}_{\tau,\delta}(t) = \begin{cases} 1 & t \geq \tau + \delta \\ -\frac{(t-\tau)^3}{4\delta^3} + \frac{3(t-\tau)}{4\delta} + \frac{1}{2} & \tau - \delta \leq t < \tau + \delta \\ 0 & t < \tau - \delta \end{cases} \tag{3.1}$$

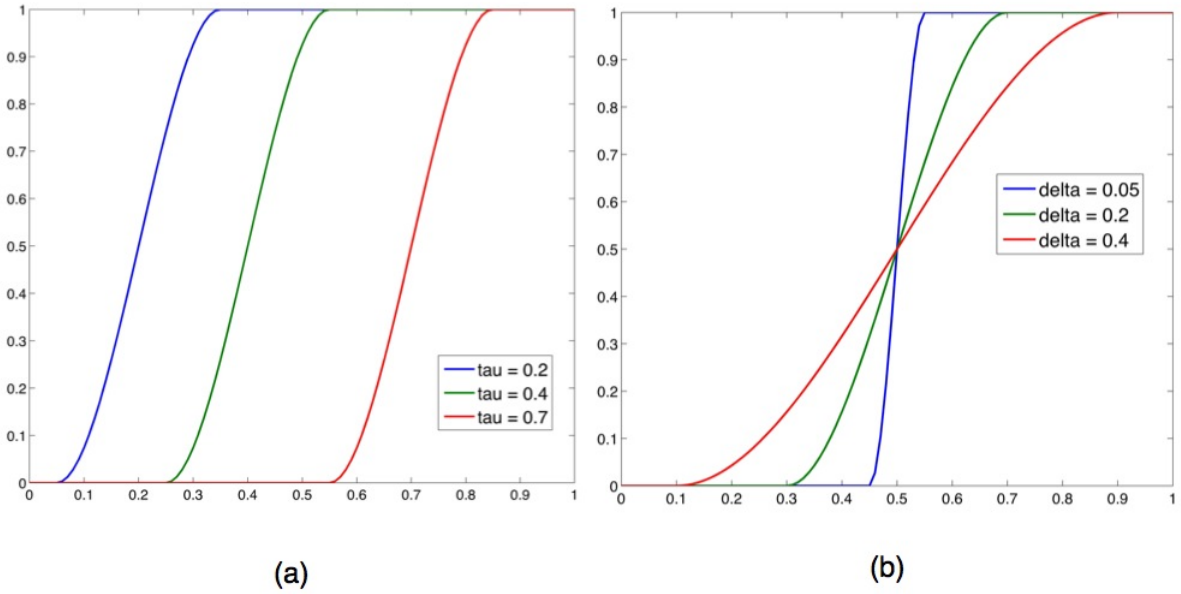



Figura 3.1: Función Heaviside esparcida $\mathcal{H}_{\tau,\delta}$ : (a) Para varios valores de $\tau$ que varía el punto en el cual empieza a cambiar de 0 a 1. $\delta$ está fijo en 0.15. (b) Para varios valores de $\delta$ que regula cuan rápido es el cambio de 0 a 1. $\tau$ está fijo en 0.5

Definimos entonces las funciones de peso:

$$g(f(x,y)) = 1 - \mathcal{H}_{\tau,\delta}(f(x,y))$$
$$h(f(x,y)) = \mathcal{H}_{\tau,\delta}(f(x,y))$$

Notemos que si $\delta \to 0$ entonces $\mathcal{H}_{\tau,\delta}(t) \to H(t-\tau)$ puntualmente, luego $h(x,y) \to H(f(x,y) - \tau)$, es decir es una función que vale 1 si $f(x,y) > \tau$ y 0 de lo contrario. Recordando que $h$ representa una función indicatriz de la zona de borde, $\tau$ se interpreta como el umbral de separación entre la zona de borde y la de difusión: si el mapa de borde se encuentra por arriba de $\tau$ en $(x,y)$ entonces este punto está en la zona de borde.

### Resolución Numérica Paralela

La implementación de este método se obtiene a partir de las iteraciones descritas en la Sección 2.3.1. En cada iteración es necesario actualizar dos arreglos bidimensionales $u^n$ y $v^n$ usando solamente datos de los arreglos del paso anterior $u^{n-1}$, $v^{n-1}$ y datos que pueden ser precalculados ($p^1$, $p^2$, $f$, $g$ y $h$) a partir de la intensidad de imagen $I$ y los parámetros del procedimiento, luego cada celda de $u^n$ y $v^n$ puede actualizarse independientemente de las otras. Sean $F_1$ y $F_2$ las funciones de actualización de las celdas de $u^n$ y de $v^n$, es decir:

$$u^n(i,j) = F_1(u^{n-1}, i, j, f, g, h, p^1, p^2)$$
$$v^n(i,j) = F_2(v^{n-1}, i, j, f, g, h, p^1, p^2)$$

El Algoritmo 5 muestra el método de resolución a un nivel de abstracción alto.

**Algoritmo 5** Resolución numérica paralela de EPGVF

**procedure** EPGVF($I$, $\tau$, $\delta$, $\mu$, $\sigma$, $n_{\text{max}}$, $e_{\text{min}}$)
  Calcular $f$, $g$, $h$, $p^1$ y $p^2$ a partir de los datos y parámetros del procedimiento.
  $e \leftarrow 0$ ▷ $e$ es la variación entre $(u, v)$ y $(u_{\text{next}}, v_{\text{next}})$
  $u \leftarrow 0$ ▷ las matrices $u$, $v$, $u_{\text{next}}$ y $v_{\text{next}}$ son inicializadas en 0
  $v \leftarrow 0$
  $u_{\text{next}} \leftarrow 0$
  $v_{\text{next}} \leftarrow 0$
  **for** $n = 1 \dots n_{\text{max}}$ **do**
    **for all** $(i,j) \in \{1, \dots, x_{\text{max}}\} \times \{1, \dots, y_{\text{max}}\}$ **do**
      Asignar $(i,j)$ a algún thread libre $t$
      ▷ Inicio thread $t$
      $u_{\text{next}}(i,j) \leftarrow F_1(u, i, j, f, g, h, p^1, p^2)$
      $v_{\text{next}}(i,j) \leftarrow F_2(v, i, j, f, g, h, p^1, p^2)$
      LOCK($e$) ▷ Bloqueamos $e$ para evitar *data racing*
      $e \leftarrow e + (u_{\text{next}}(i,j) - u(i,j))^2 + (v_{\text{next}}(i,j) - v(i,j))^2$
      UNLOCK($e$)
      ▷ Fin thread $t$
    **end for**
    $u \leftarrow u_{\text{next}}$
    $v \leftarrow v_{\text{next}}$
    **if** $e \leq e_{\text{min}}$ **then**
      BREAK
    **end if**
  **end for**
**end procedure**

### 3.3.2. T-Snakes

#### Estructura de datos de la grilla

Tal como se mencionó en la Sección 2.2.1, la grilla usada en la implementación del esquema de reparametrización de las T-Snakes está compuesta de *simplexes*.

Las estructuras de datos básicas de la grilla son las que representarán los vértices, arcos y caras (los polígonos que forman los vértices). Estas estructuras están construidas de manera que pueden acceder a sus estructuras adyacentes explícitamente, por ejemplo cada arco posee una lista de punteros a la estructura de los vértices de sus extremos y una lista de punteros de las caras adyacentes a él (ver Figura 3.2).

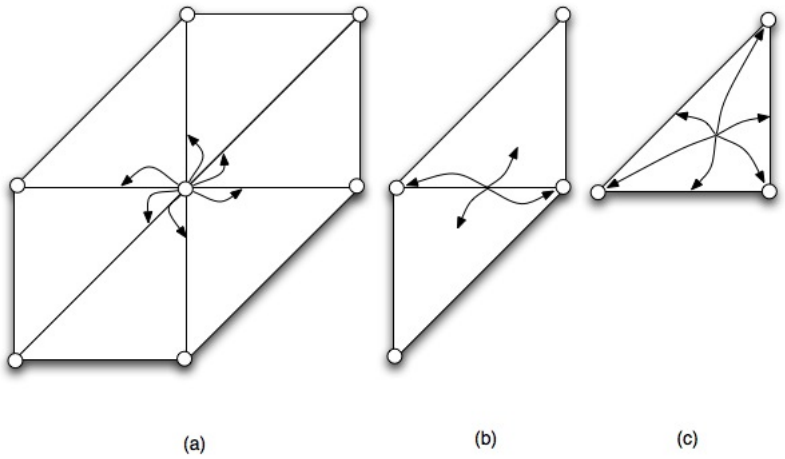



Figura 3.2: Punteros de cada estructura básica de la grilla. (a) La estructura del vértice posee punteros a cada uno de los arcos a los que es adyacente. (b) Un arco posee un puntero a los vértices que lo forman y las caras adyacentes a él. (c) Una cara posee punteros a los vértices que la definen y los arcos adyacentes a ella.

Las estructuras además poseen variables relevantes para la ejecución del algoritmo de reparametrización en las T-snakes. Finalmente la estructura de datos de la grilla contiene una lista de vértices, una de arcos y una de caras junto con otros parámetros relevantes para el algoritmo de T-snakes.

A continuación se muestran en detalle los campos de cada estructura:

- **Vértice**
  - *x*: Coordenada x del vértice.
  - *y*: Coordenada y del vértice.
  - *e*: Punteros a arcos adyacentes. Es un arreglo fijo de seis celdas.
  - *ind*: Valor de la indicatriz. Uno si el vértice está dentro de la snake, cero si no.
- **Arco**
  - *v*: Punteros a vértices adyacentes. Es un arreglo fijo de dos celdas
  - $f$: Punteros a caras adyacentes. Es un arreglo fijo de dos celdas
  - *inter_x*: Variable que almacenará la coordenada x de una eventual intersección de la snake con la grilla.
  - *inter_y*: Variable que almacenará la coordenada y de una eventual intersección de la snake con la grilla.
  - *inter_sign*: El signo de la intersección (ver Sección 2.2.2).
  - *type*: Tipo de arco. Puede ser vertical, diagonal u horizontal.
- **Cara**
  - *v*: Punteros a vértices adyacentes. Es un arreglo fijo de tres celdas.
  - *e*: Punteros a arcos adyacentes. Es un arreglo fijo de tres celdas
  - *type*: Tipo de cara. Puede ser normal o rotada.
- **Malla**
  - *vlist*: Punteros a todos los vértices.
  - *elist*: Punteros a todos los arcos.
  - *flist*: Punteros a todas las caras.
  - *nx*: Ancho en vértices de la malla.
  - *ny*: Alto en vértices de la malla.
  - *x_max*: Ancho en pixeles de la imagen.
  - *y_max*: Alto en pixeles de la imagen.

- *hx*: Paso horizontal de la malla ($hx = x\_max/(nx-1)$).
- *hy*: Paso vertical de la malla ($hy = y\_max/(ny-1)$).
- *nvertices*: Número total de vértices.
- *nedges*: Número total de arcos.
- *nfaces*: Número total de caras.
- *m*: Pendiente de los arcos diagonales ($m = hx/hy$).

En el proceso de reparametrización no se agregan ni se eliminan vértices, arcos ó caras por lo tanto sólo fue necesario implementar métodos de creación y destrucción de las estructuras. Los campos de las estructuras se acceden directamente.

### Algoritmo de comparación de números de punto flotante

El algoritmo de reparametrización requiere calcular los puntos de intersección entre un segmento y la grilla, y ésto a su vez requiere comparaciones de igualdad entre números de punto flotante. La comparación nativa ($==$) en números de punto flotante es una comparación bit por bit lo cual tiene una utilidad muy limitada.

Es por esto que se requiere una rutina de comparación de números de punto flotante robusta. Una primera idea es fijar un nivel de error $\varepsilon$ y decidir si dos números $x$ e $y$ son iguales si es que su error absoluto es menor que $\varepsilon$:

$$cmp(x,y) = 1 \Leftrightarrow |x-y| \leq \varepsilon$$

El problema de este apronte es que no hay $\varepsilon$ adecuado para todos los rangos de magnitud. Por otro lado si hacemos algo similar con el error absoluto:

$$cmp(x,y) = 1 \Leftrightarrow \frac{|x-y|}{\text{máx}(|x|,|y|)} \leq \varepsilon$$

o equivalentemente:

$$cmp(x,y) = 1 \Leftrightarrow |x-y| \leq \varepsilon \ \text{máx}(|x|,|y|)$$

aún hay problemas para números con módulo menor que 1, pues $\varepsilon \ \text{máx}(|x|,|y|)$ puede ser mucho más pequeño que $\varepsilon$, por lo que puede ser muy difícil que dos números sean considerados iguales. Una solución robusta a este problema es una estrategia mixta (ver [Eri04], pág 443):

$$cmp(x,y) = 1 \Leftrightarrow |x-y| \leq \varepsilon \ \text{máx}(1,|x|,|y|) \quad ,$$

la cual elimina los problemas de cada una de las dos estrategias mostradas y provee un algoritmo robusto de comparación. En la implmementación del presente trabajo se uso $\varepsilon = 10^{-9}$.

# Capítulo 4

# Resultados y Discusión

## 4.1. EPGVF

### 4.1.1. Comparación con GGVF

Se usará la imagen de la Figura 4.1 para analizar la comparación de los aprontes de GGVF y EPGVF. Consta de un círculo con intensidad alta y un rectángulo con intensidad media-baja. Ambas figuras están cercanas y se agregó ruido gaussiano para poder exponer las ventajas de EPGVF frente a GGVF.

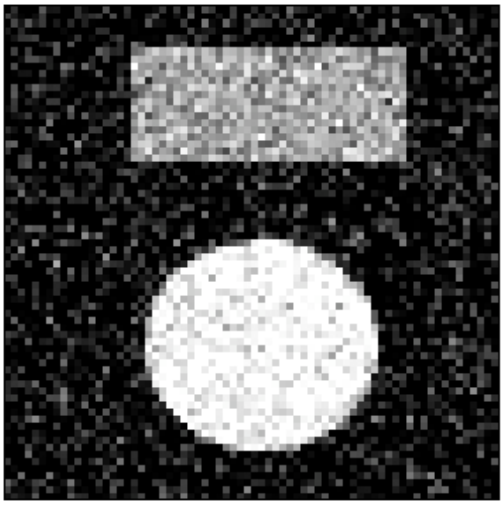


Figura 4.1: Imagen Original para la Comparación de GGVF y EPGVF

Las funciones de peso que ocupan los GGVF están definidas como:

$$g(x,y) = e^{-|\nabla I(x,y)|/K} \tag{4.1}$$

$$h(x,y) = 1 - e^{-|\nabla I(x,y)|/K} \tag{4.2}$$

con $K > 0$ un parámetro ajustable. A mayor $K$ se necesitará una mayor intensidad de gradiente de imagen ($|\nabla I|$) en el punto $(x, y)$ para considerar ese punto como uno en la zona de bordes.

Primero se calcula el GGVF con $K = 10$ y se escoge *edge map* $|\nabla I|^2$. Se observa en la Figura 4.2 (b) que debido al alto valor de $K$ hace que la zona de borde (donde $h$ no es cero) esté razonablemente delimitada pero con un muy bajo valor de $h$ especialmente el caso del rectángulo. Es por esto que en el GGVF el borde del círculo "devora" al del rectánglo por efecto de la difusión (Figura 4.2 (c)).

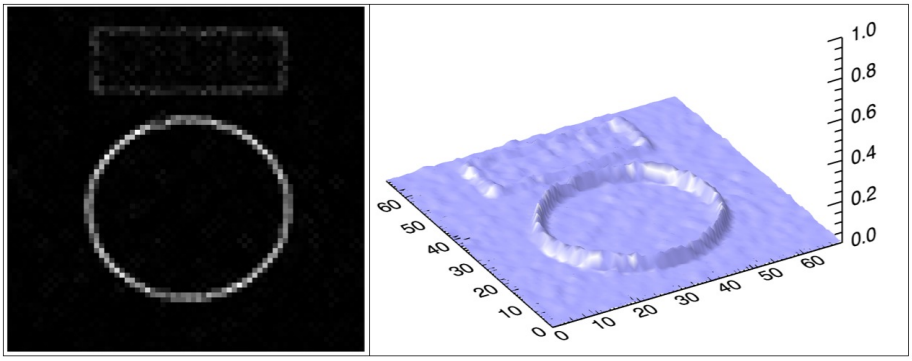
Un posible apronte para solucionar este problema es disminuir el valor de $K$, esto realza las zonas de borde asignándoles un mayor valor de $h$, pero también zonas de ruido (que tienen gradiente pequeño) pasan como zona de borde debido a la naturaleza exponencial de $h$. Estos efectos se observan en las Figuras 4.3 (b) y (c) donde se calculó el GGVF con $K = 1$. El ruido perturba el cálculo y crea un GGVF atractivo a bordes falsos y mas aún, se distingue muy levemente entre los contornos del rectángulo y círculo. Es importante remarcar que la atracción a bordes falsos es particularmente pernicioso para varios usos posteriores en las snakes que pueden ser inicializadas en el borde de la imagen, y este efecto hace que la segmentación esté muy lejos de lo que se debería lograr.

Una solución conocida para suavizar el ruido es convolucionar la imagen con una gaussiana. Esto se puede implementar en el apronte de GGVF cambiando el *edge map* $f$ por $|\nabla I * G_\sigma|^2$. Se calculó el GGVF ahora con este *edge map* con $\sigma^2 = 1$. El efecto del suavizado se puede observar en la Figura 4.4 (a) en el cual se observa aminorado el ruido comparado con la Figura 4.3 (a). Este efecto también se observa en la Figura 4.4 (b) donde se observa que $h$ distingue mejor la zona de borde, sin embargo esta zona "engorda". Este efecto hace que en el GGVF 4.4 (c) vuelva a pasar lo que en en el primer caso: el borde del círculo "devora" al del rectángulo (pero menos que en el primer ejemplo).

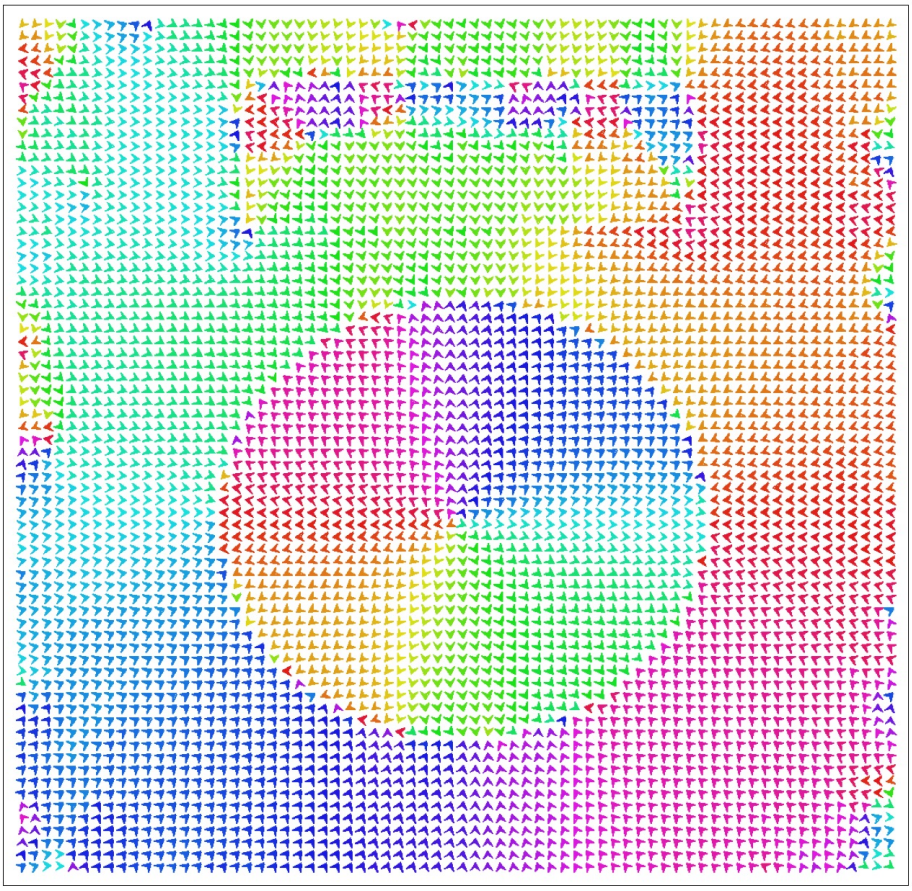
Razonando similarmente como en el segundo ejemplo se disminuye $K$ a $0.1$. El resultado hace que efectivamente se logre una separación en el campo de las figuras, sin embargo nuevamente el ruido sea considerado como borde, con los mismos efectos indeseados que en el segundo ejemplo. Esto se puede observar en la Figuras 4.5 (b) y (c).

Luego no se puede encontrar un $K$ que permita construir un campo que satisfaga la separación de las figuras y la robustez hacia el ruido. Por otra parte con EPGVF esto se puede lograr en este caso con $\mu = 2.0$, $\tau = 0.05$, $\delta = 0.03$ y el *edge map* $|\nabla I * G_1|^2$. Esto se puede observar en la Figura 4.6

Finalmente en la Figura 4.7 se compara EPGVF con GGVF en una imagen de microscopía de lípidos. Se puede observar en (b) que existen perturbaciones en el campo debido a que el ruido es considerado como zona de borde. En (c) se puede apreciar que no hay perturbaciones como en el caso anterior.

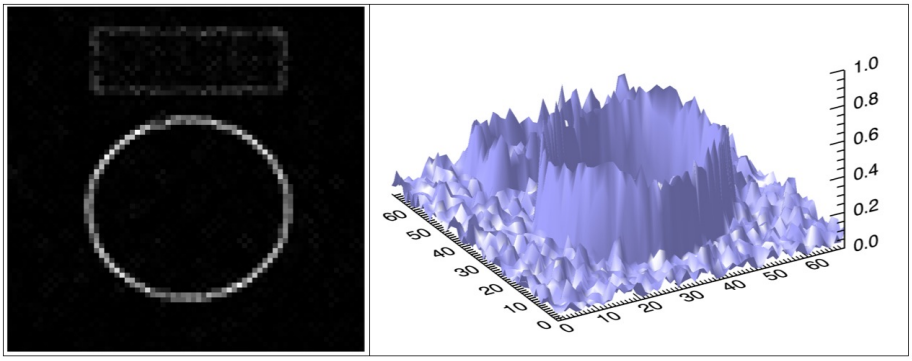



(a) (b)

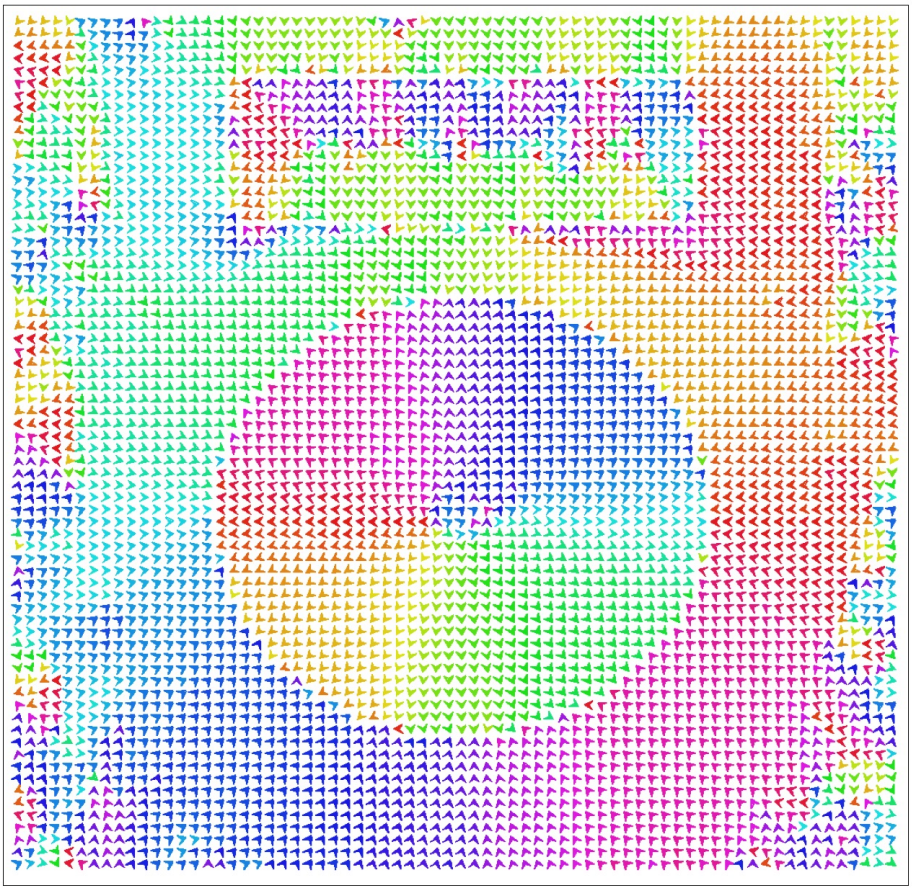


(c)

Figura 4.2: (a) El *edge map* $|\nabla I|^2$. (b) La función de peso en la zona de borde: $h = 1 - e^{-|\nabla I|/10}$. (c) GGVF normalizado, vectores coloreados según su dirección.




(a) (b)



(c)

Figura 4.3: (a) El *edge map* $|\nabla I|^2$. (b) La función de peso en la zona de borde: $h = 1 - e^{-|\nabla I|}$. (c) GGVF normalizado, vectores coloreados según su dirección.

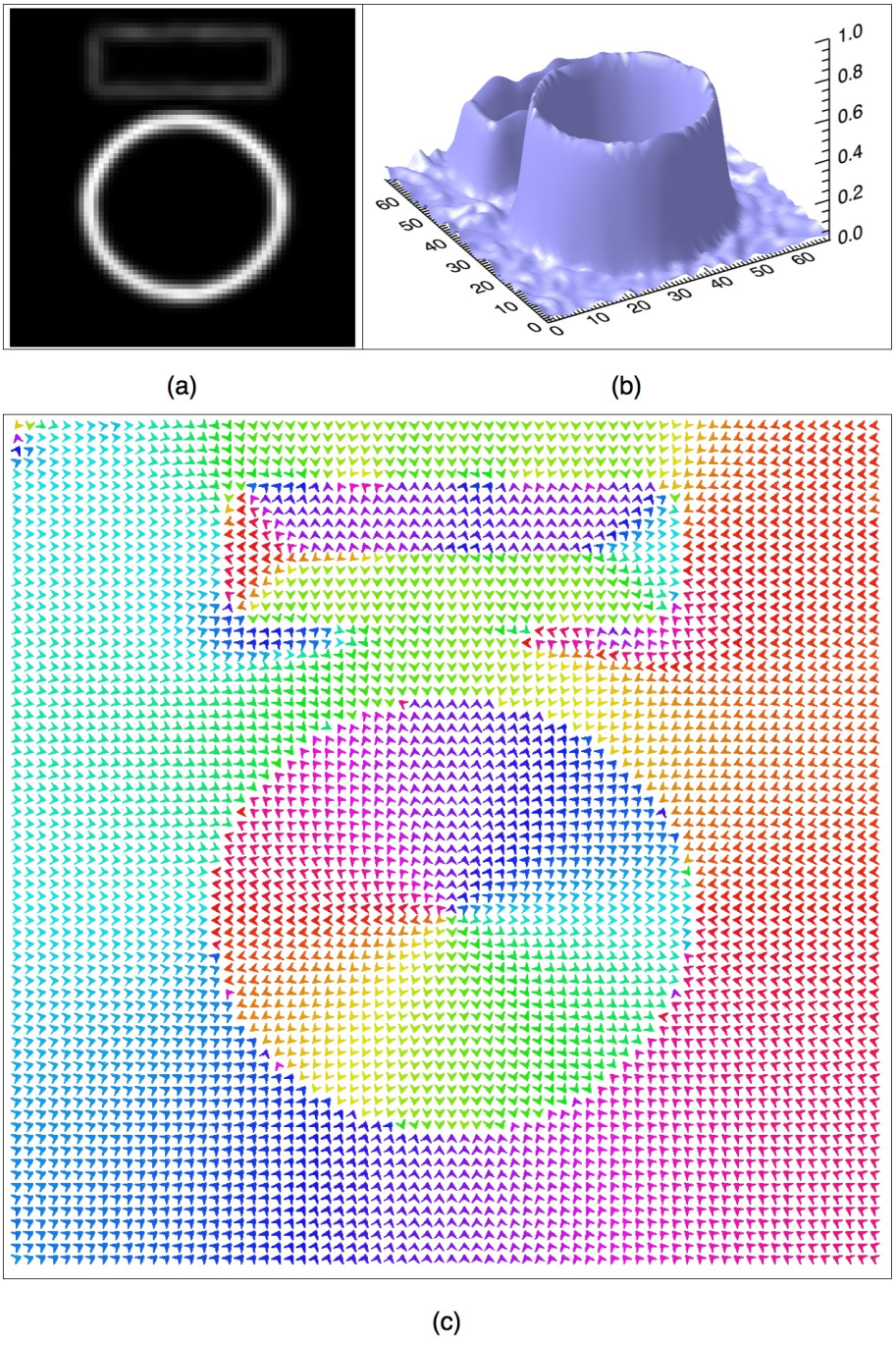
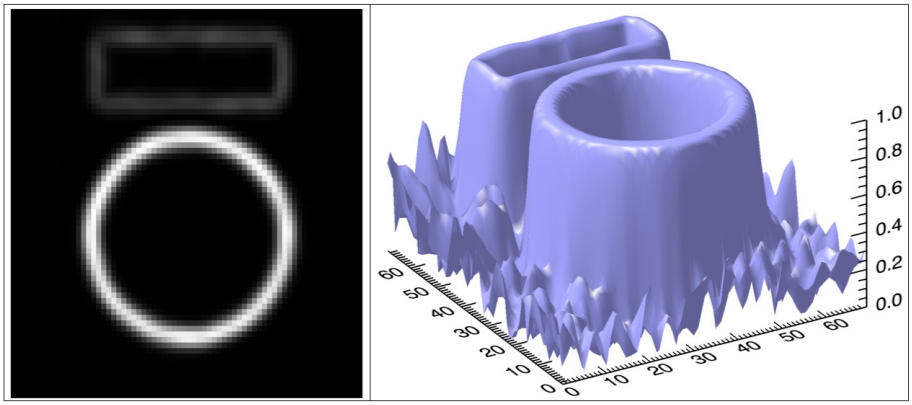
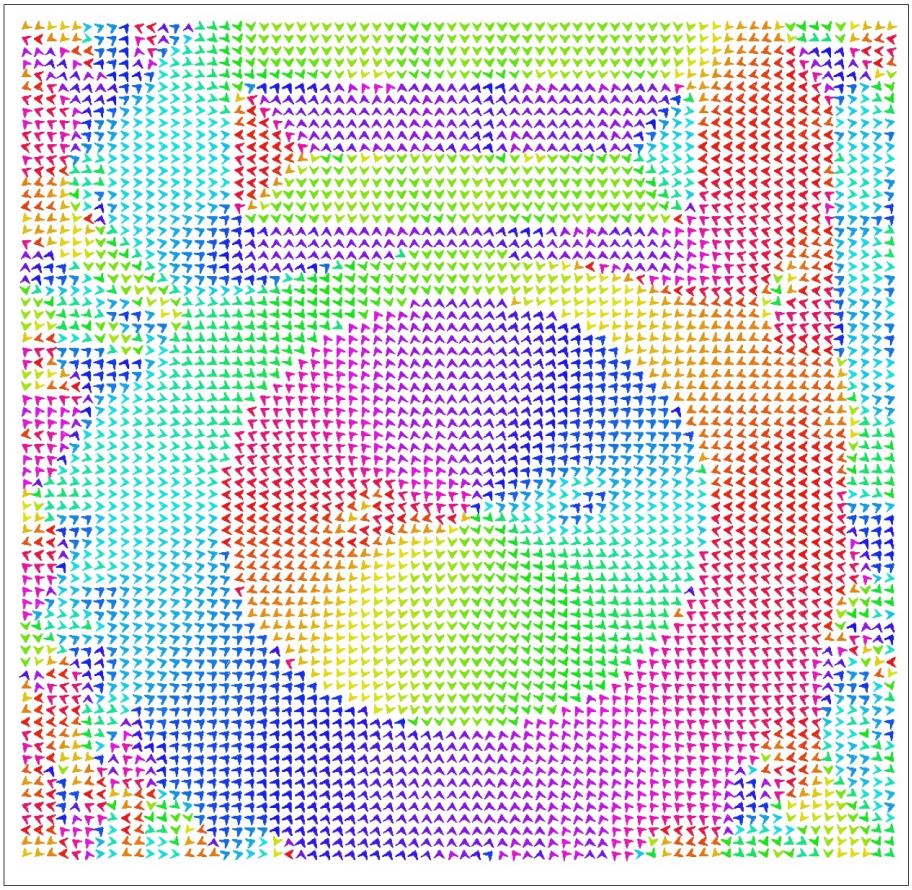



Figura 4.4: (a) El *edge map* $|\nabla I * G_1|^2$. (b) La función de peso en la zona de borde: $h = 1 - e^{-|\nabla I * G_1|}$. (c) GGVF normalizado, vectores coloreados según su dirección.

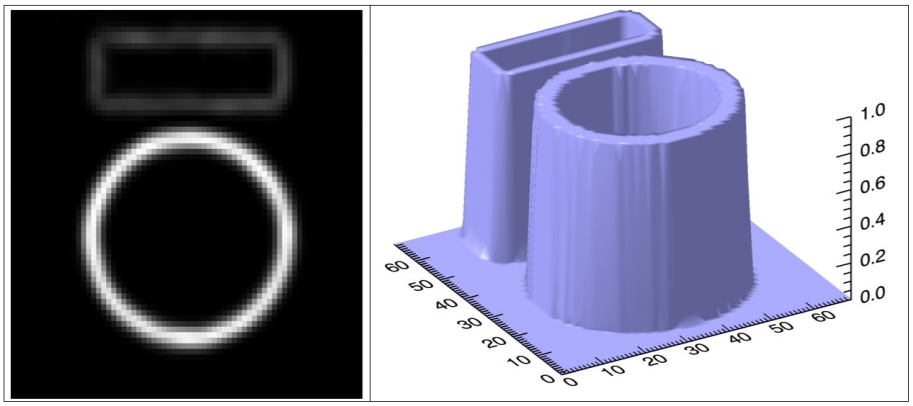



(a) (b)

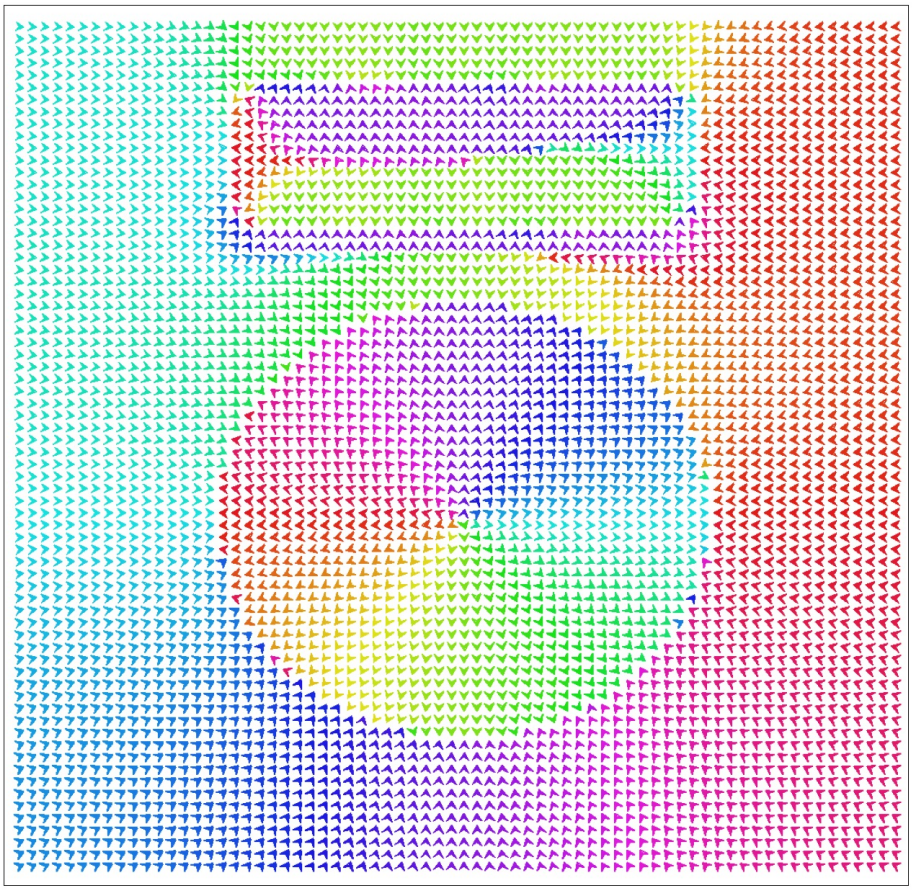


(c)

Figura 4.5: (a) El *edge map* $|\nabla I * G_1|^2$ ($\sigma^2 = 1$). (b) La función de peso en la zona de borde: $h = 1 - e^{-|\nabla I * G_1|/0.1}$. (c) GGVF normalizado, vectores coloreados según su dirección.




(a) (b)



(c)

Figura 4.6: (a) El *edge map* $|\nabla I * G_1|^2$ . (b) La función de peso en la zona de borde: $h = 1 - \mathcal{H}_{0.05,0.03}(|\nabla I * G_1|^2)$ . (c) EPGVF normalizado, vectores coloreados según su dirección.

.

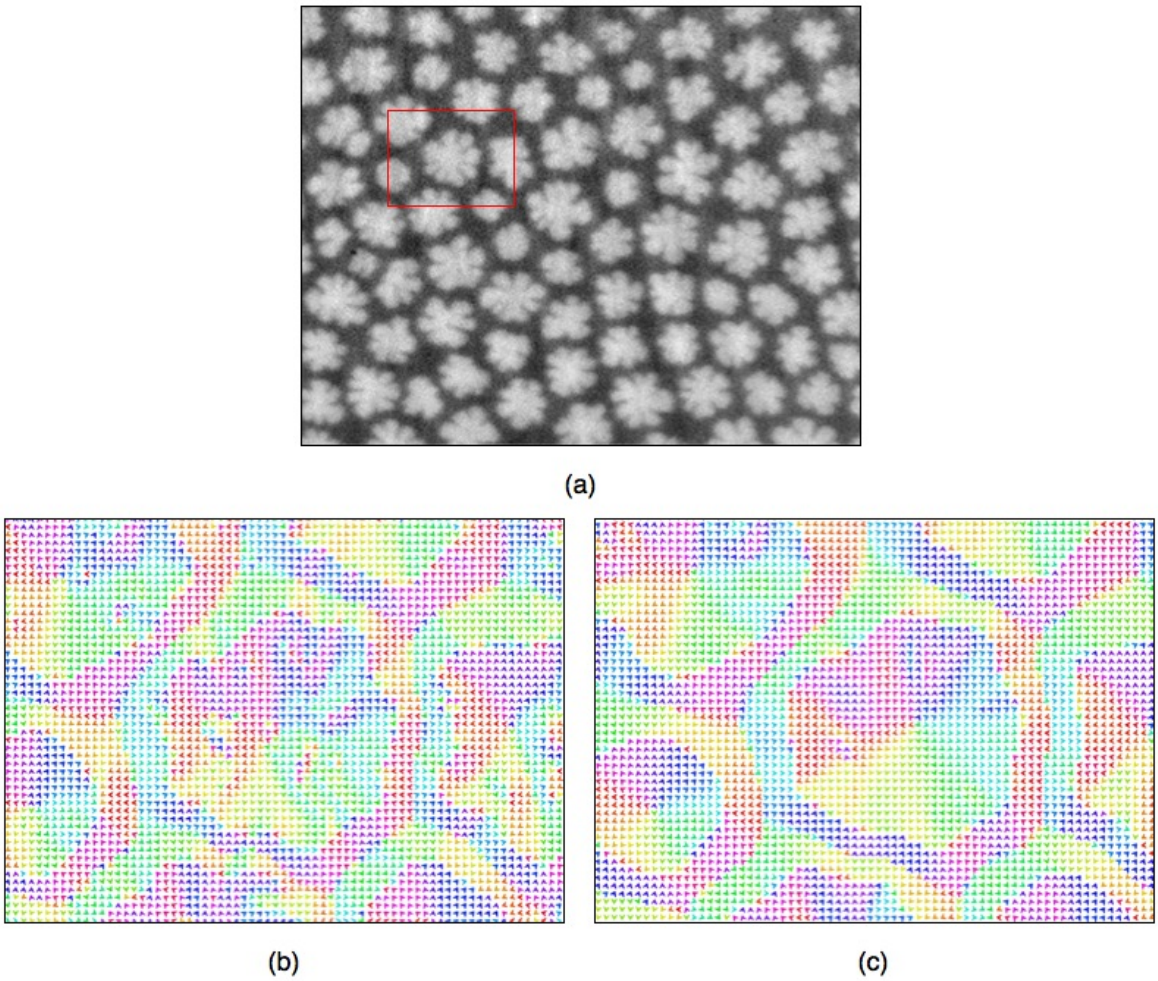



Figura 4.7: Comparación de EPGVF y GGVF en una imagen de lípidos: (a) La imagen original. (b) GGVF normalizado de la zona marcada en rojo en (a). (c) EPGVF normalizado de la zona marcada en rojo en (a). Vectores coloreados por orientación.

### 4.1.2. Discusión

Una observación en [XP98] menciona que en la ecuación de la cual se deducen los GGVF los términos:

$$\nabla g \cdot \nabla u$$
$$\nabla g \cdot \nabla v$$

no influyen considerablemente en el campo obtenido, por lo que se descartan de la ecuación a resolver dando la formulación de los GGVF. Motivado por ello se realizaron pruebas para ver la influencia en el EPGVF de este término y de los términos:

$$\mu \sum_{i=1}^{2} \sum_{j=1}^{2} \left( \frac{\partial (h p_i p_j)}{x_i} \frac{\partial u}{\partial x_j} + h p_i p_j \frac{\partial^2 u}{\partial x_i \partial x_j} \right)$$
$$\mu \sum_{i=1}^{2} \sum_{j=1}^{2} \left( \frac{\partial (h p_i p_j)}{x_i} \frac{\partial v}{\partial x_j} + h p_i p_j \frac{\partial^2 v}{\partial x_i \partial x_j} \right)$$

Notemos que al eliminar estos términos queda simplemente la resolución de GGVF pero con las funciones de peso de EPGVF. Los resultados obtenidos en las pruebas que se han hecho resolviendo la ecuación sin estos términos son que éstos no influyen en resultado final. Por lo que el único efecto que causa la mejora de EPGVF respecto de GGVF son las funciones de peso. Una posible explicación es que la función exponencial no permite diferenciar bien las zonas de bordes por su crecimiento brusco, por otro lado la función usada en el EPGVF (basada en la aproximación de la Heaviside $\mathcal{H}_{\tau,\delta}$) tiene un crecimiento mucho más controlable (cúbico y dependiente de dos parámetros) lo que permite separar bien las zonas de borde. En la Figura 4.8 se muestra el cálculo de GGVF con las funciones de peso del EPGVF y no se observa mayor diferencia con respecto a los resultados expuestos en la Figura 4.6.

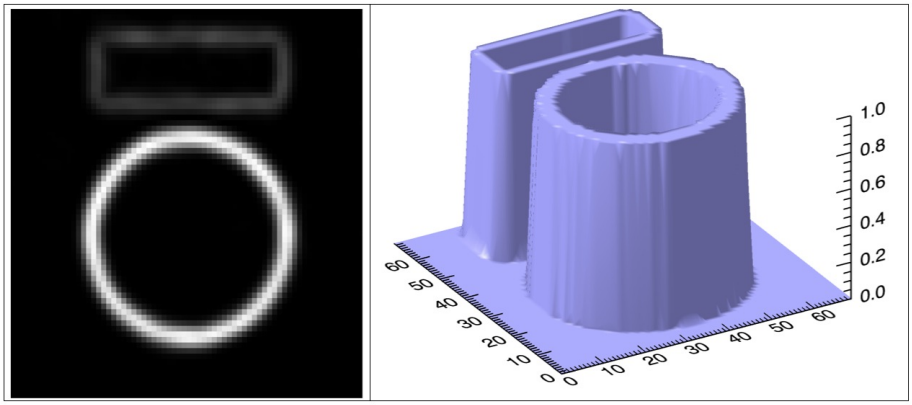
## 4.2. T-Snakes

### 4.2.1. Comparación con la implementación actual

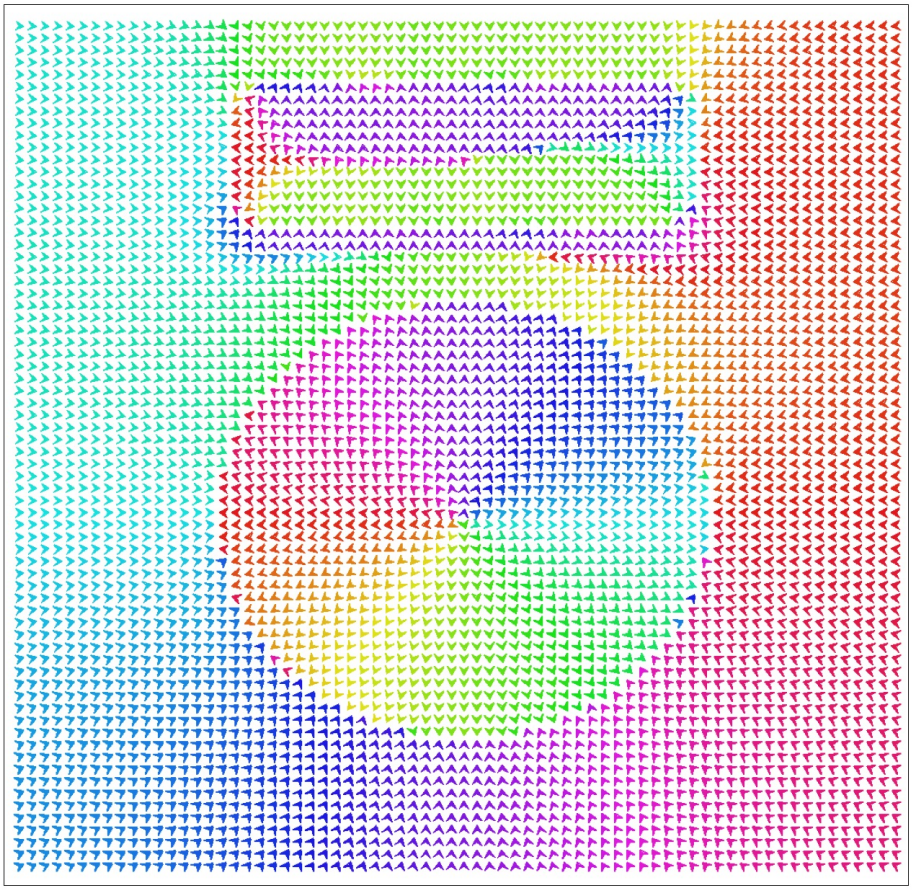
Se comparará la implementación de T-Snakes con respecto a la implementación actual de snakes en SCIAN.

En las siguientes pruebas se usará GGVF pero con las funciones de peso del EPGVF como la fuerza que guiará la snake hacia los bordes.

En la Figura 4.9 se observa la evolución de una snake normal en la imagen del rectángulo y círculo usada en la Sección 4.1. La falta de fuerza de inflación (deflación en este caso)




(a) (b)



(c)

Figura 4.8: (a) El *edge map* $|\nabla I * G_1|^2$ ($\sigma^2 = 1$). (b) La función de peso en la zona de borde: $h = 1 - \mathcal{H}_{0.05,0.03}(|\nabla I * G_1|^2)$. (c) GGVF normalizado, vectores coloreados según su dirección.

hace que la snake converja a un equilibrio que está lejos aún de ser una segmentación adecuada de la figura. En cambio con el uso de T-Snakes 4.10 con fuerza de inflación la snake se acerca a un punto en el cual se forma un cuello y posteriormente se separa en dos snakes que evolucionan independientemente gracias al manejo de cambios topológicos que provee el método.

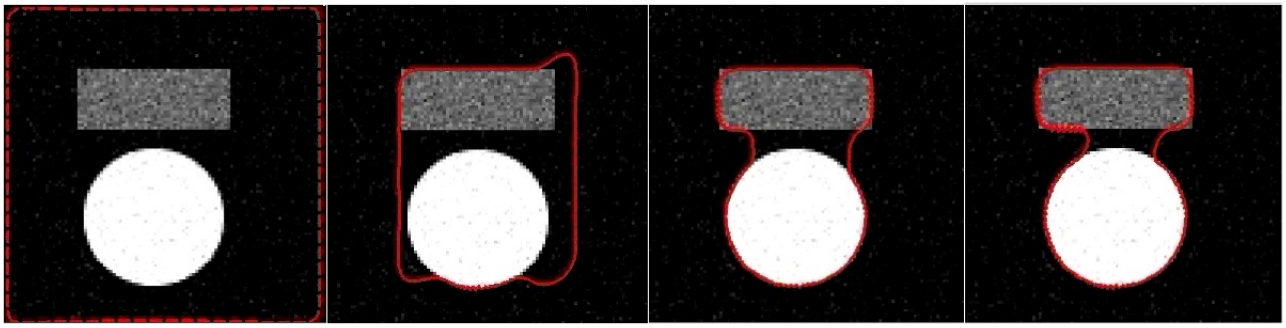


Figura 4.9: Segmentación de un círculo y un rectángulo usando snakes normales

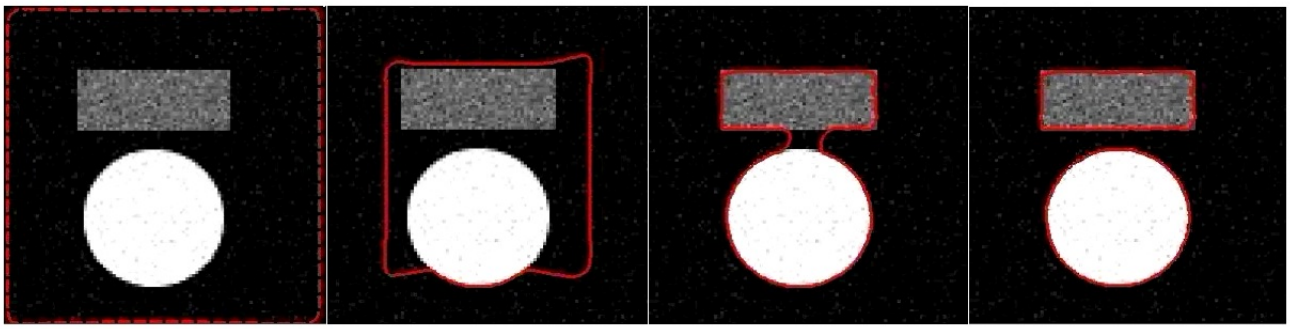


Figura 4.10: Segmentación de un círculo y un rectángulo usando T-Snakes

En la Figura 4.11 se observa un fenómeno similar al anterior, la falta de fuerza de inflación hace que la snake no pueda seguir por los "tubos" de la imagen. Por otro lado las T-Snakes (Figura 4.12) logran evolucionar como corresponde y además se manejan los cambios topológicos apropiadamente.

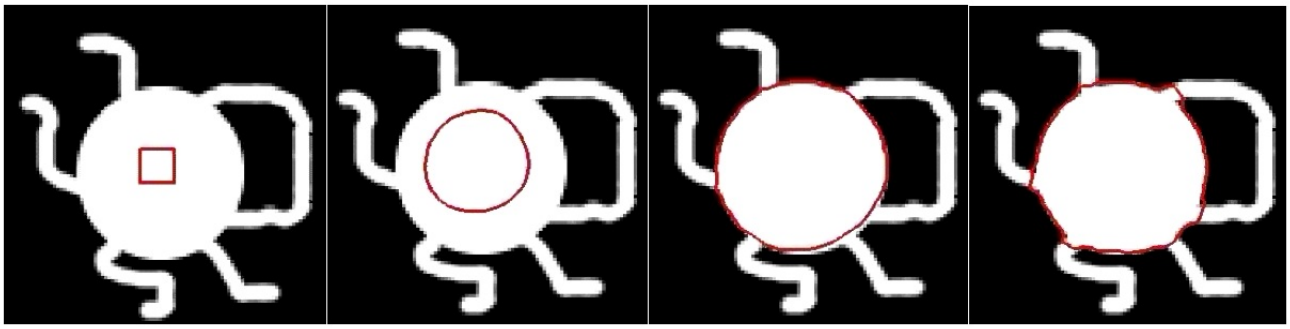


Figura 4.11: Evolución de una snake clásica en una imagen con un objeto de topología compleja

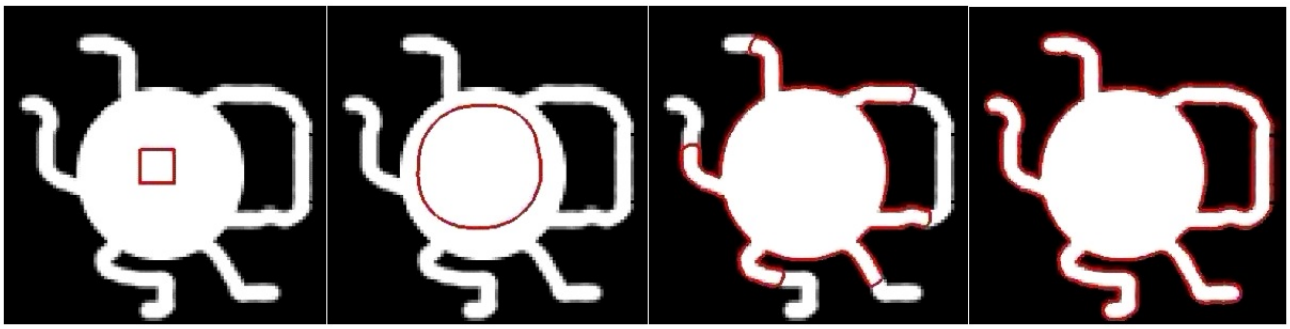


Figura 4.12: Evolución de una T-Snake en una imagen con un objeto de topología compleja

### 4.2.2. Discusión

En los ejemplos recién expuestos se observa la ventaja más importante de las T-Snakes con respecto a las snakes clásicas: adaptabilidad topológica, y esto se traduce en el posibilidad de fusionar y cortar snakes. Además éstas permiten construir una estrategia para lograr una segmentación automática: se "siembran" snakes en toda la imagen a intervalos regulares y se les deja evolucionar. Gracias a la adaptabilidad topológica estas se fusionaran, desaparecerán o cortarán logrando así capturar todos los contornos de objetos en la imagen. Ejemplos de esta estrategia se verán en la Sección 4.3

## 4.3. Ejemplos de Aplicación

En la Figura 4.13 se muestra el GGVF (con las funciones de peso de EPGVF) obtenido de una imagen de microscopía de lípidos. La segmentación usando este campo y T-Snakes se observa en la Figura 4.14, en la cual se demuestra la flexibilidad del apronte ya que para llegar a la segmentación final se requiere manejar una serie de cambios topológicos (cortes) en la snake.

También se aplica a esta imagen la estrategia de sembrado de snakes (ver Figura 4.15). Se inicializan snakes circulares de 12 pixeles de radio. El resultado es una segmentación equivalente al obtenido en el primer ejemplo pero se requieren muchos menos pasos de deformación antes de llegar a segmentar todos los objetos (80 pasos de deformación contra 800).

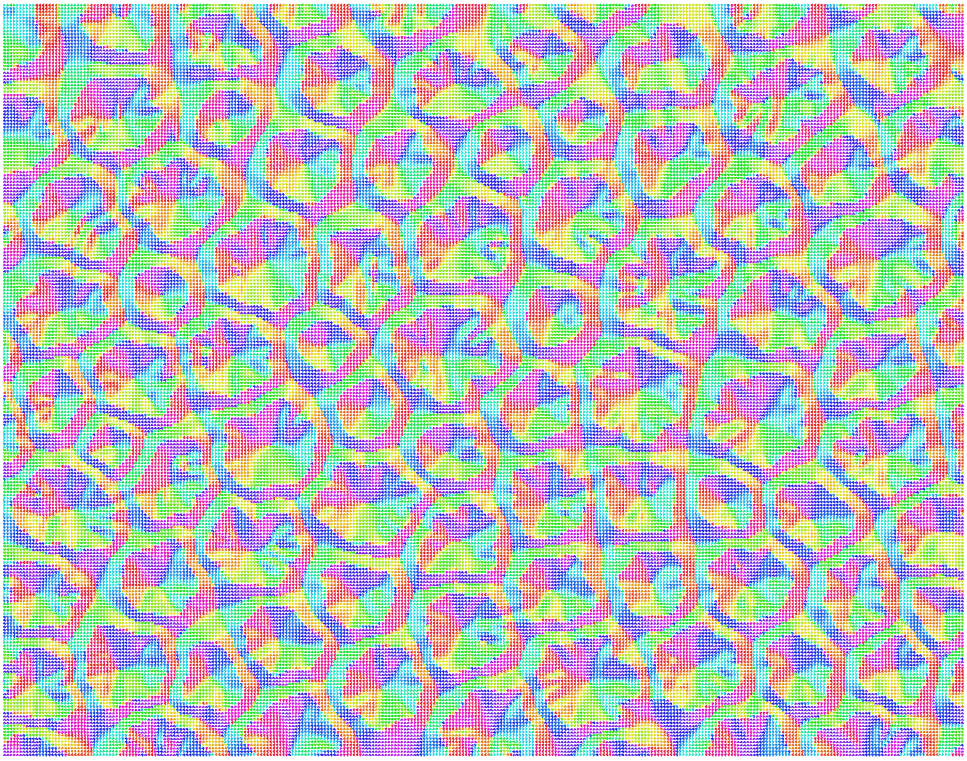


Figura 4.13: GGVF de una imagen de Lípidos: Los parámetros para las funciones de peso fueron : $\tau = 0.05$, $\delta = 0.025$. El paso temporal para la resolución numérica fue $\Delta t = 0.1$. Los vectores están coloreados por el ángulo que poseen.

Otro ejemplo de aplicación es en una imagen de una tubulina acetilada. Las tubulinas son proteínas que conforman microtúbulos, éstos son observados en la imagen de la Figura 4.17. Las T-Snakes logran segmentar de manera satisfactoria la imagen, a pesar de lo delgada que es la tubulina.

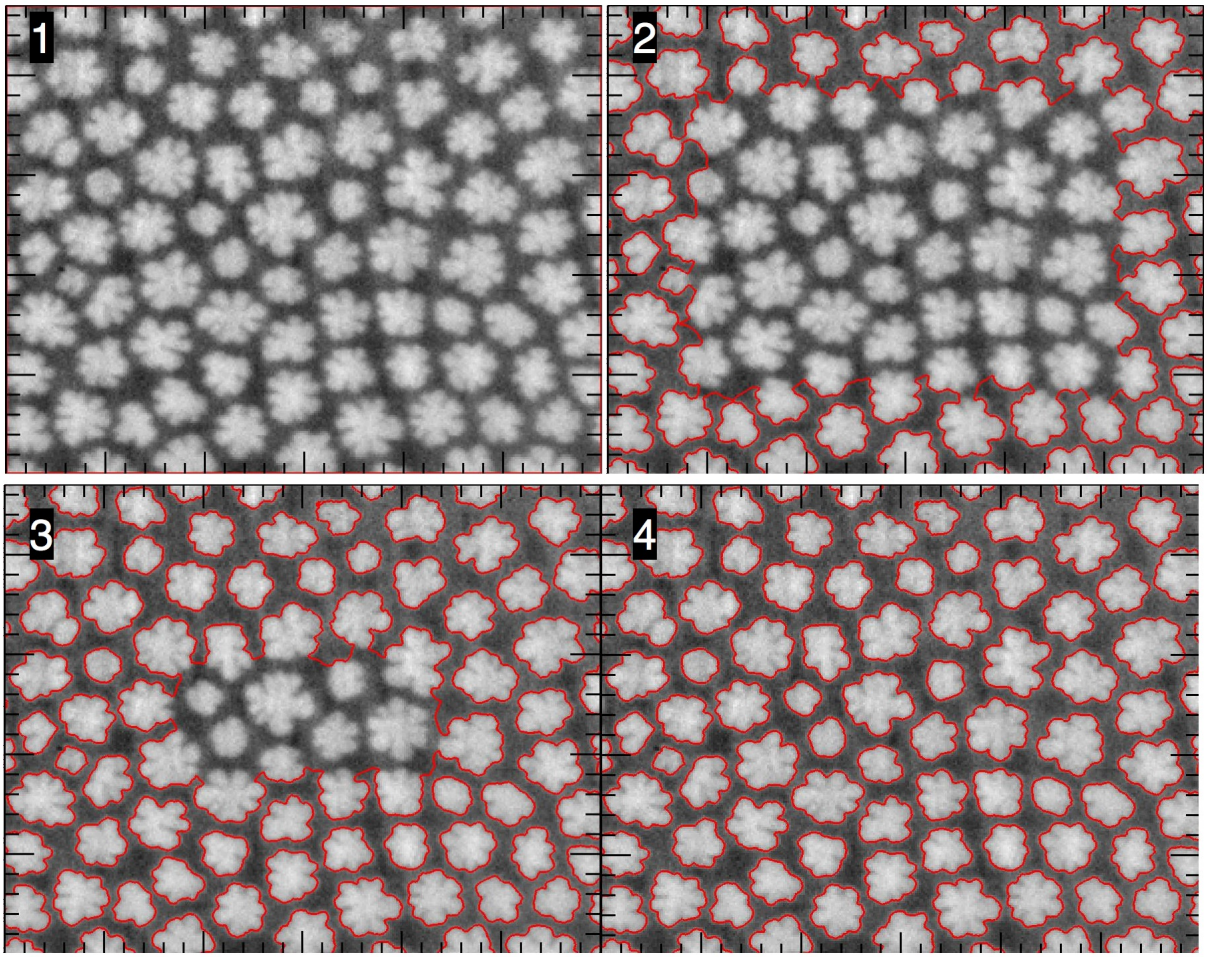



Figura 4.14: Segmentación de una imagen de lípidos vía T-Snakes: Los parámetros usados fueron : $m = 3$, $p = 20.5$, $q = 20.0$, $T = 85$, $a = 10.0$, $b = 5.0$, $\Delta t = 0.002$. La malla fue escogida de resolución $50 \times 50$.

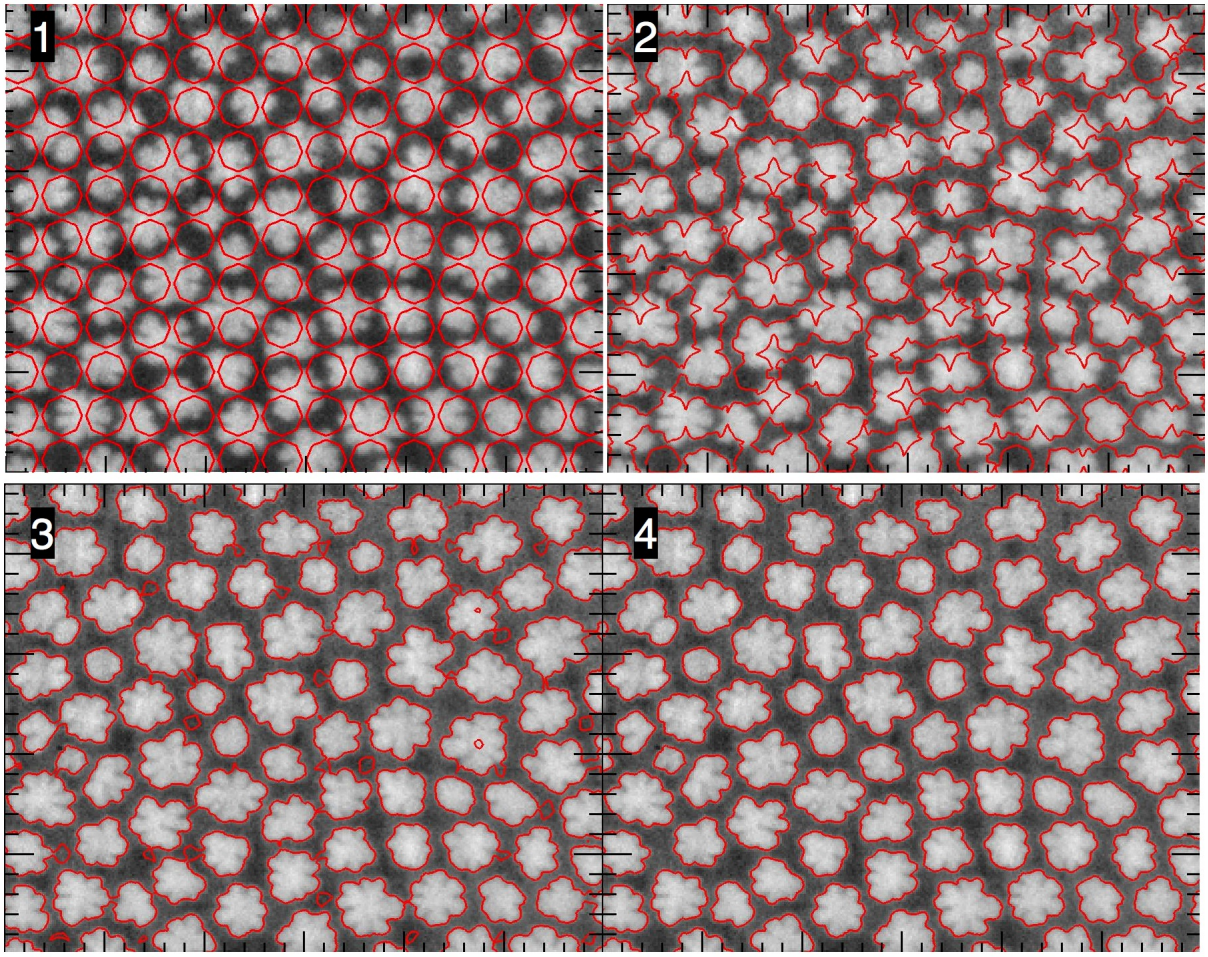



Figura 4.15: Segmentación de una imagen de lípidos vía T-Snakes con la estrategia de "sembrado" de snakes: Los parámetros usados fueron : $m = 5$, $p = 21$, $q = 20.0$, $T = 130$, $a = 12.0$, $b = 7.0$, $\Delta t = 0.002$. La malla fue escogida de resolución $300 \times 300$.

En esta imagen también se muestra la estrategia de "sembrado" de snakes (ver Figura 4.18) inicializando con snakes circulares de radio 12 pixeles. Nuevamente el número de iteraciones requerido para la convergencia es mucho menor. En la estrategia de inicializar una sola snake desde afuera la convergencia demoró del orden de 800 pasos de deformación mientras que en este caso se llegó sólo en 160 pasos de deformación. Además se segmenta un pequeño agujero en el lado izquierdo de la tubulina, el cual la primera estrategia no logra captar.

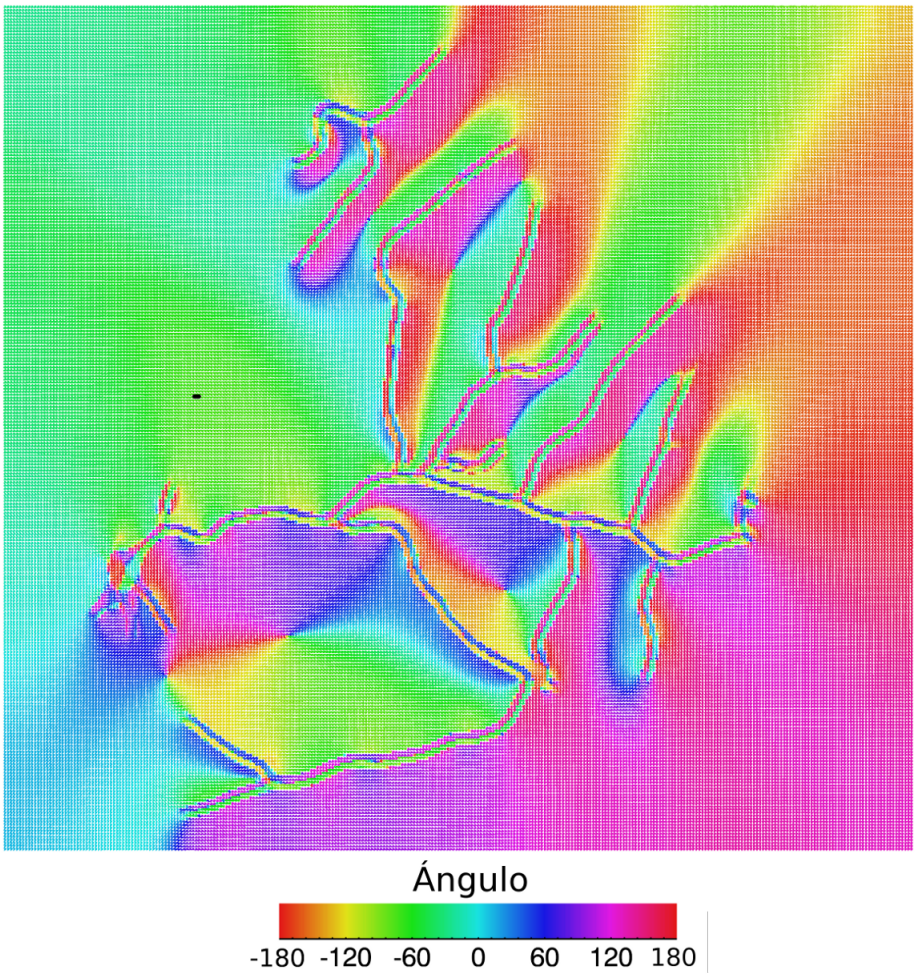



Figura 4.16: GGVF de una imagen de Tubulinas: Los parámetros para las funciones de peso fueron : $\tau = 0.05$, $\delta = 0.025$. El paso temporal para la resolución numérica fue $\Delta t = 0.1$. Los vectores están coloreados por el ángulo que poseen.

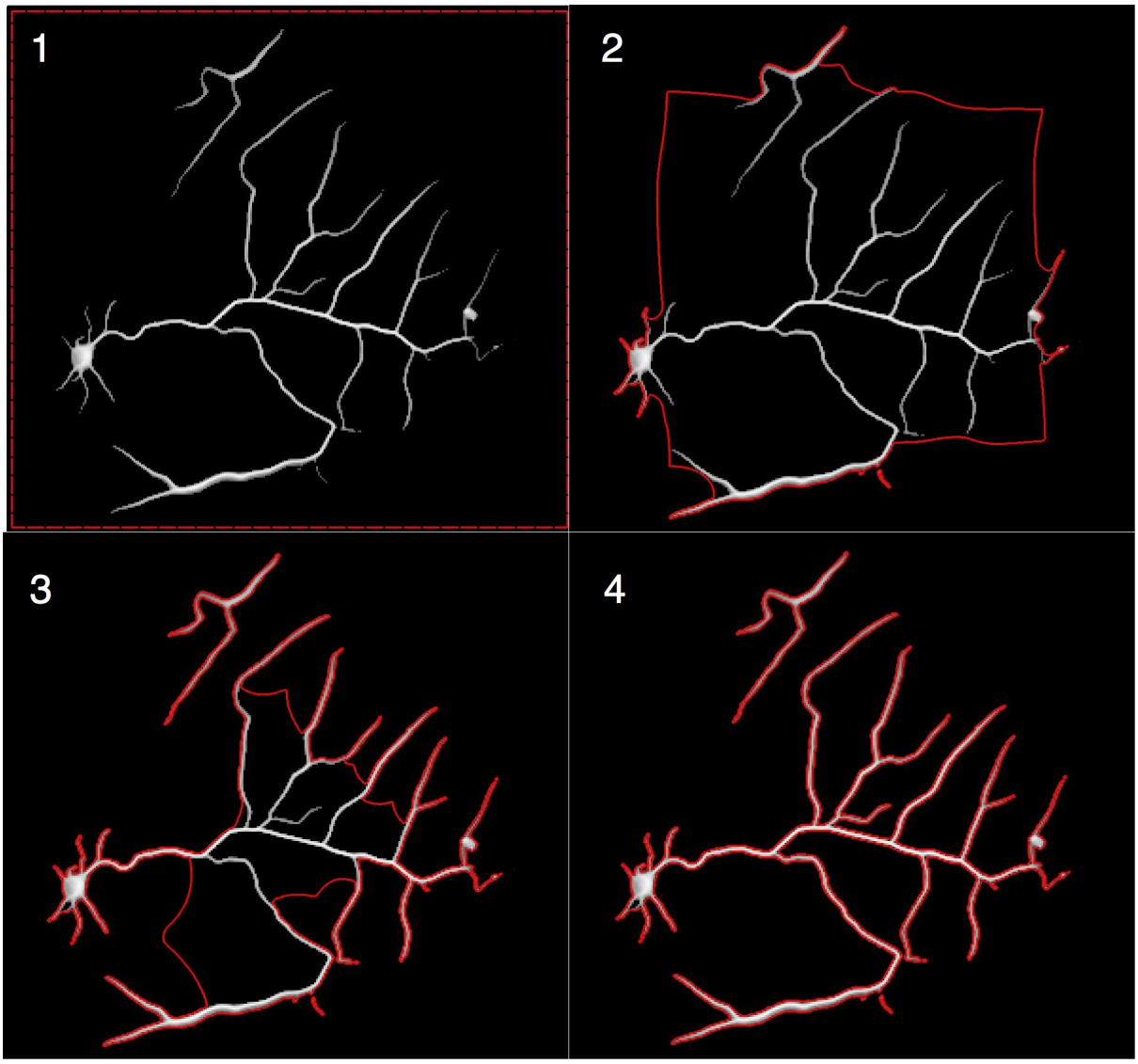



Figura 4.17: Segmentación de una imagen de Tubulinas vía T-Snakes: Los parámetros usados fueron : $m = 5$, $p = 20.5$, $q = 20.0$, $T = 15$, $a = 12.0$, $b = 7.0$, $\Delta t = 0.002$. La malla fue escogida de resolución $300 \times 300$.

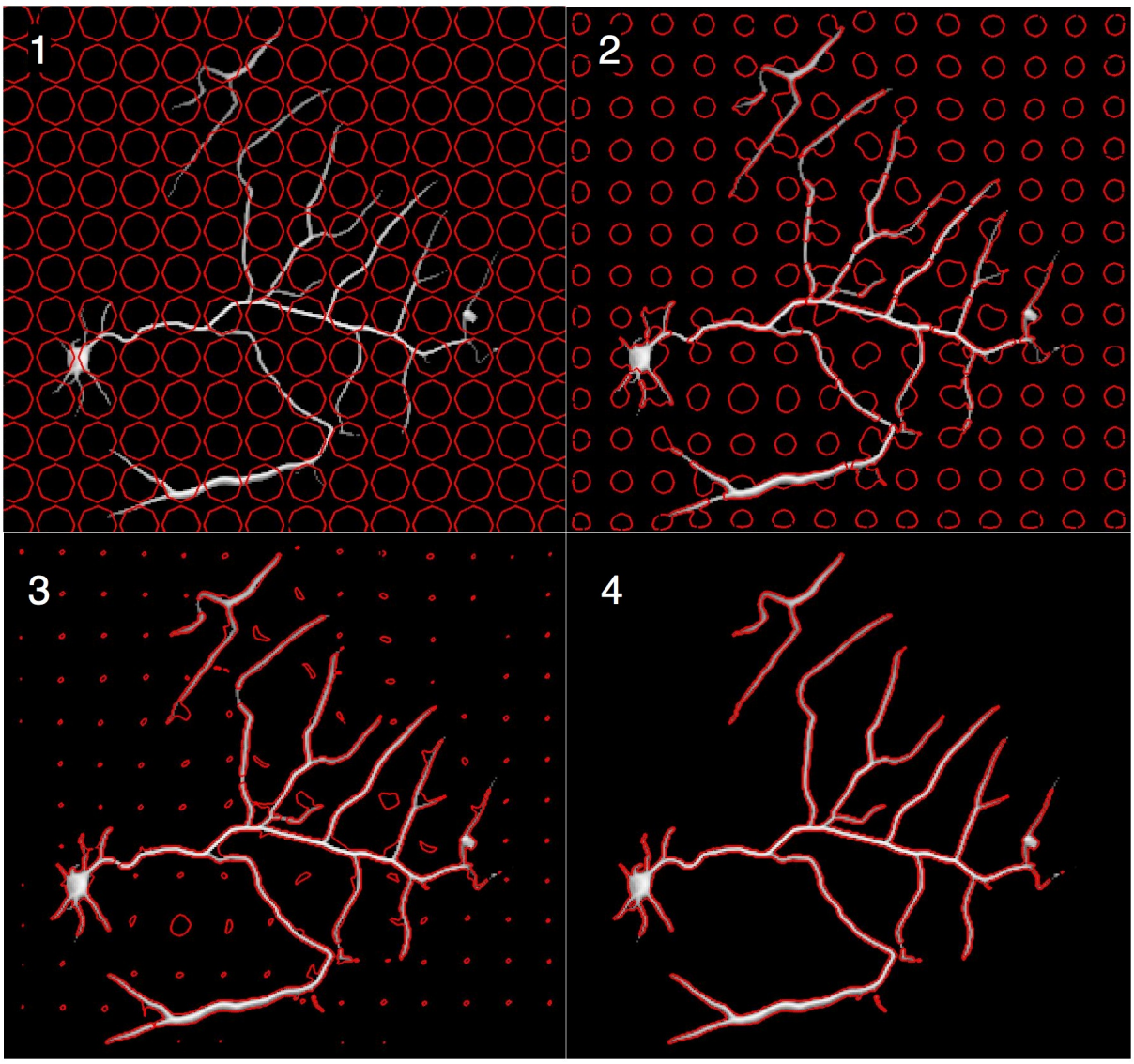



Figura 4.18: Segmentación de una imagen de Tubulinas vía T-Snakes con la estrategia de "sembrado" de snakes: Los parámetros usados fueron : $m = 5$, $p = 20.5$, $q = 20.0$, $T = 15$, $a = 12.0$, $b = 7.0$, $\Delta t = 0.002$. La malla fue escogida de resolución $300 \times 300$.

# Capítulo 5

# Conclusiones y Trabajo a Futuro

## 5.1. Conclusiones

En el presente trabajo se ha logrado implementar satisfactoriamente el algoritmo de EPGVF mostrando con los ejemplos y datos probados que la mayor diferencia con GGVF es la elección de una función de peso más flexible y no los términos extras en la ecuación de EPGVF. Lo anterior junto con la elección de usar la imagen convolucionada con una función gaussiana hace que las zonas de difusión y de bordes queden bien definidas, lo que se traduce en la creación de un campo con mayor atracción hacia los contornos y que evita la difusión de bordes ficticios creados por la presencia de ruido.

También se implementó el algoritmo de T-Snakes, estrategia que permite segmentar objetos de mayor complejidad gracias al manejo de cambios topológicos que posee.

Ambos algoritmos fueron aplicados a imágenes de microscopía confocal de fluorescencia logrando una segmentación adecuada de estas imágenes. Además se usó la estrategia de sembramiento de snakes para lograr un método de segmentación que requiere muy poca interacción humana y que logra segmentar todos los objetos de una imagen.

Las implementaciones fueron realizadas en C para aprovechar mejor las capacidades del hardware, en el caso de EPGVF fué realizada de manera paralela y así poder usar procesadores multinúcleo los cuales son ubicuos hoy en día. Los algoritmos han sido incorporados a SCIAN en forma de una librería dinámica usando los puentes disponibles en IDL para llamadas a funciones externas.

## 5.2. Trabajo a Futuro

### 5.2.1. T-Snakes

El trabajo realizado dejó muchos caminos abiertos por los cuales se puede continuar el desarrollo del algoritmo de T-Snakes. Los principales son:

- Estudiar e implementar un criterio de parada para el algoritmo (como el mencionado en [MT00] que usa el concepto de "calor" en un segmento de snake). Esto es necesario para obtener un método de segmentación con un mayor nivel de automatización y además mejorar la rapidez del algoritmo pues se lograría evitar que se hagan cálculos en snakes que no se moverán más.
- Estudiar e implementar la paralelización del paso de deformación, la fase uno y dos del algoritmo de T-Snakes para aumentar la eficiencia y rapidez en la ejecución.
- El estudio e implementación de T-Snakes en tres dimensiones (*T-Surfaces*, ver [MT00]). Esto es aplicable en la construcción de modelos tridimensionales a partir de los *stacks* de imágenes obtenidos a partir de microscopía confocal de fluorescencia.

### 5.2.2. EPGVF

El desarrollo de esta parte de la memoria también dejó problemas abiertos para trabajar en el futuro:

- La realización de un análisis numérico acabado del método usado para la resolución del sistema de ecuaciones de EPGVF. En particular es de interés obtener una cota del tipo Courant-Friedrichs-Lewy para el método y así poder determinar de manera automática el paso temporal a usar en la resolución de la EDP.
- La realización de un análisis matemático riguroso del sistema de ecuaciones de EPGVF para saber porqué los términos descartados no afectan la solución de manera significativa o en su defecto saber bajo que condiciones si lo hacen.
- Un problema presente en el modelo es la determinación de sus parámetros ($\mu$, $\tau$ y $\delta$). Para obtener un campo útil para la segmentación se debe afinar muy bien los parámetros y no hay valores que sirvan de manera general, pues es un problema que depende mucho de los datos. La determinación automática de estos parámetros es entonces un tema relevante para obtener sistemas con la menor interacción humana posible. En [DF07] se estudia una solución automática a este problema, usando métodos estadísticos, considerando además los parámetros de elasticidad y curvatura de la segmentación vía snakes.

# Apéndice A

# Cálculo de Variaciones

## A.1. Ecuaciones de Euler-Lagrange

Se deducen las ecuaciones de Euler-Lagrange para el caso en que el lagrangiano depende también de la segunda derivada temporal.

**Lema A.1.1.** *Sea* $c \in C([a,b];\mathbb{R})$ *tal que :*

$$\int_a^b c(t)h(t)\,dt = 0$$

*para toda función* $h \in C^1([a,b];\mathbb{R})$ *con* $h(a) = h(b) = h'(a) = h'(b) = 0$. *Entonces* $c \equiv 0$

*Demostración.* Basta probar el lema para el caso $a = 0$, $b = 1$ pues con una transformación afín se obtiene cualquier otro caso. Consideremos para $0 < \delta < \frac{1}{2}$:

$$h_\delta(t) = \begin{cases} 0 & t < 0 \\ f_\delta(t) & 0 \le t < \delta \\ 1 & \delta \le t < 1-\delta \\ g_\delta(t) & 1-\delta \le t < 1 \\ 0 & 1 \le t \end{cases}$$

con:

$$f_\delta(t) = \begin{cases} 0 & t < 0 \\ -\frac{2}{\delta^3}t^3 + \frac{3}{\delta^2}t^2 & 0 \le t < 1 \\ 1 & 1 \le t \end{cases}$$

$$g_\delta(t) = 1 - f_\delta(t-(1-\delta))$$

Se puede verificar que:

- $f_\delta$, $g_\delta$ y $h_\delta$ son funciones de clase $C^1([0,1];\mathbb{R})$,
- $h_\delta$ converge puntual y monótonamente a 1 cuando $\delta \to 0$,
- Para $n > 2$ se tiene que $w_n = ch_{\frac{1}{n}}$ satisface $w_n(0) = w_n(1) = w_n'(0) = w_n'(1) = 0$.

Luego por hipótesis:

$$\int_0^1 c(t)w_n(t)\,dt = 0$$

Además como $w_n$ converge puntualmente a $c$, $cw_n$ converge puntualmente a $c^2$ y por ser funciones continuas en un compacto la familia $cw_n$ está acotada uniformemente por una función constante $K$ que es integrable en $[0,1]$.

Luego por el teorema de convergencia dominada de Lebesgue se tiene que :

$$0 = \lim_{n->\infty} \int_0^1 c(t)w_n(t)\,dt = \int_0^1 c(t)^2\,dt$$

y por lo tanto $c = 0$ ctp, pero por ser continua se tiene $c \equiv 0$ ■

**Teorema A.1.2.** *Sea $L \in C^1(\mathbb{R}^N \times \mathbb{R}^N \times \mathbb{R}^N;\mathbb{R})$. Si $x \in C^2([0,T];\mathbb{R}^N)$ es una solución regular del problema :*

$$(\mathcal{P}) : \min_{\substack{x\in C^2([0,T];\mathbb{R}^N)\\ x(0)=x_i,\ x(T)=x_f}} J(x)$$

*donde $J$ está definido como:*

$$J(x) = \int_0^T L(x(t),\dot{x}(t),\ddot{x}(t))\,dt$$

*entonces $x$ satisface la siguiente ecuación :*

$$\frac{\partial L}{\partial x} - \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} + \frac{d^2}{dt^2}\frac{\partial L}{\partial \ddot{x}} = 0$$

*Demostración.* Sea $\nu$ una función de clase $C^1(\mathbb{R};\mathbb{R}^n)$ tal que $\nu(0) = \nu(T) = \nu'(0) = \nu'(T) = 0$. Sea $x$ una solución al problema $(\mathcal{P})$. Se define:

$$\phi(\lambda) = J(x + \lambda\nu)$$

Luego $\phi$ tiene un mínimo en 0 por lo tanto:

$$\phi'(0) = 0$$

Calculando:

$$\phi'(0) = \frac{d}{d\lambda} J(x + \lambda\nu)|_{\lambda=0} = \frac{d}{d\lambda} \int_0^T L(x(t) + \lambda\nu(t), \dot{x}(t) + \lambda\dot{\nu}(t), \ddot{x}(t) + \lambda\ddot{\nu}(t))\, dt|_{\lambda=0}$$

como $L$ es una función de clase $C^1$, por el teorema de convergencia dominada se puede pasar la derivada dentro de la integral y aplicando la regla de la cadena se obtiene:

$$0 = \phi'(0) = \int_0^T \frac{\partial L}{\partial x}(x(t), \dot{x}(t), \ddot{x}(t))\nu(t) + \frac{\partial L}{\partial \dot{x}}(x(t), \dot{x}(t), \ddot{x}(t))\dot{\nu}(t) + \frac{\partial L}{\partial \ddot{x}}(x(t), \dot{x}(t), \ddot{x}(t))\ddot{\nu}(t)\, dt$$

Integrando por partes y ocupando el hecho de que $\nu(0) = \nu(T) = \nu'(0) = \nu'(T) = 0$ se llega a la ecuación :

$$\int_0^T \left( \frac{\partial L}{\partial x}(x(t), \dot{x}(t), \ddot{x}(t)) - \frac{d}{dt}\frac{\partial L}{\partial \dot{x}}(x(t), \dot{x}(t), \ddot{x}(t)) + \frac{d^2}{dt^2}\frac{\partial L}{\partial \ddot{x}}(x(t), \dot{x}(t), \ddot{x}(t)) \right) \nu(t)\, dt = 0$$

Como $\nu$ es arbitrario se puede aplicar el Lema A.1.1 y se obtiene el resultado ■

## A.2. Marco Variacional de GGVF y EPGVF

Se usará en la demostración del próximo teorema el siguiente resultado de cálculo vectorial:

**Teorema A.2.1** (Fórmula de Integración por Partes)**.** *Sea $u$ un campo escalar de clase $C^1(\mathbb{R}^n; \mathbb{R})$, $\mathbf{v}$ un campo vectorial de clase $C^1(\mathbb{R}^n; \mathbb{R}^n)$ y $\Omega$ un abierto de frontera suave por partes cuya normal exterior es $\nu$. Entonces:*

$$\int_\Omega \nabla u \cdot \mathbf{v}\, dx = \int_{\delta\Omega} u\mathbf{v}\cdot\nu\, d\sigma - \int_\Omega u\nabla\cdot\mathbf{v}\, dx$$

**Teorema A.2.2.** *Sea $\mathbf{u} = (u, v)$ una solución al problema:*

$$(\mathcal{P}) : \min_{\mathbf{u}\in\mathcal{C}^2(\Omega;\mathbb{R}^2)} J(\mathbf{u})$$

*donde :*

$$J(\mathbf{u}) = \int_\Omega g(|\nabla f|)\left(|\nabla u|^2 + |\nabla v|^2\right) + h(|\nabla f|)\left(\mu\left((\nabla u\cdot\mathbf{p})^2 + (\nabla v\cdot\mathbf{p})^2\right) + |u - f_x|^2 + |v - f_y|^2\right) dx$$

- $\Omega$ *es un abierto acotado de frontera regular*
- $f$ *es una función de clase* $C^1(\mathbb{R}^2;\mathbb{R})$.
- $g$ *y* $h$ *son funciones de clase* $C^0(\mathbb{R}^2;\mathbb{R})$.
- $\mu > 0$.
- $\mathbf{p} = \begin{pmatrix} p^1 \\ p^2 \end{pmatrix}$ *es un campo de clase* $C^1(\mathbb{R}^2;\mathbb{R}^2)$

*Definiendo* $(x_1, x_2) = (x, y)$ *se tiene que* $\mathbf{u} = (u, v)$ *satisface el sistema de ecuaciones:*

$$\nabla g \cdot \nabla u + g\Delta u + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp_ip_j)}{x_i}\frac{\partial u}{\partial x_j} + hp_ip_j\frac{\partial^2 u}{\partial x_i \partial x_j}\right) - h(u - f_x) = 0$$

$$\nabla g \cdot \nabla v + g\Delta v + \mu \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp_ip_j)}{x_i}\frac{\partial v}{\partial x_j} + hp_ip_j\frac{\partial^2 v}{\partial x_i \partial x_j}\right) - h(v - f_y) = 0$$

*Demostración.* El funcional $J$ se puede separar en la suma de dos funcionales que dependen sólo de $u$ y de $v$ , y que además son a valores positivos. Esto implica que si $\mathbf{u} = (u, v)$ minimiza $J$ entonces $u$ y $v$ minimizan los funcionales que les corresponden en la descomposición de $J$. Además por simetría de estos funcionales, basta analizar sólo uno de estos.

Sea $u$ una función de clase $C^2(\mathbb{R}^2;\mathbb{R})$ que minimiza:

$$J_1(u) = \int_\Omega g(|\nabla f|)|\nabla u|^2 + h(|\nabla f|)\left(\mu\left((\nabla u \cdot \mathbf{p})^2\right) + |u - f_x|^2\right)dx$$

y para $w \in C_0^\infty(\mathbb{R}^2;\mathbb{R})$ se define:

$$\phi(\lambda) = J_1(u + \lambda w)$$

Como $\mathbf{u}$ minimiza $J_1$ se tiene que:

$$\phi'(0) = 0 \tag{A.1}$$

Calculando:

$$\begin{aligned}\phi'(0) &= \frac{d}{d\lambda}\, J_1(u + \lambda w)|_{\lambda=0} \\ &= \frac{d}{d\lambda}\int_\Omega g\left(|\nabla f|\right)\left(|\nabla u + \lambda\nabla w|^2\right) + h(|\nabla f|)\left(\mu\left|(\nabla u + \lambda\nabla w)\cdot\mathbf{p}\right|^2 + |u + \lambda w - f_x|^2\right)dx\,\Big|_{\lambda=0}\end{aligned}$$

por teorema de convergencia dominada se puede pasar la derivada dentro de la integral:

$$\begin{aligned}\phi'(0) &= \int_\Omega 2g(|\nabla f|)\left((\nabla u + \lambda\nabla w)\cdot\nabla w\right) + 2h(|\nabla f|)\left(\mu\,\mathbf{p}\cdot(\nabla u+\lambda\nabla w)\,\nabla w\cdot\mathbf{p} + (u+\lambda w - f_x)w\,\right)dx\,\Big|_{\lambda=0}\\ &= \int_\Omega 2g(|\nabla f|)\left(\nabla u\cdot\nabla w\right) + 2h(|\nabla f|)\left(\mu\left((\mathbf{p}\cdot\nabla u)(\mathbf{p}\cdot\nabla w)\right) + (u-f_x)w\,\right)dx\end{aligned}$$

y aplicando integración por partes (Teorema A.2.1) se tiene que:

$$\phi'(0) = 2\int_\Omega \left(-\nabla\cdot(g(|\nabla f|)\nabla u) - \mu\,\nabla\cdot(h(|\nabla f|)(\mathbf{p}\cdot\nabla u)\,\mathbf{p}) + h(|\nabla f|)(u-f_x)\right)w\,dx$$

Ahora por A.1 se obtiene la ecuación integral:

$$2\int_\Omega \left(-\nabla\cdot(g(|\nabla f|)\nabla u) - \mu\,\nabla\cdot(h(|\nabla f|)(\mathbf{p}\cdot\nabla u)\mathbf{p}) + h(|\nabla f|)(u-f_x)\right)w\,dx = 0$$

Como ésto es válido para cualquier $w \in C_0^\infty(\mathbb{R}^2;\mathbb{R})$ y $u$ es continua se obtiene finalmente:

$$-\nabla\cdot(g(|\nabla f|)\nabla u) - \mu\,\nabla\cdot(h(|\nabla f|)(\mathbf{p}\cdot\nabla u)\mathbf{p}) + h(|\nabla f|)(u-f_x) = 0$$

Con un poco de manipulación algebraica y definiendo $(x_1, x_2) = (x, y)$ la ecuación anterior es equivalente a (se omiten evaluaciones de $g$ y $h$):

$$\nabla g\cdot\nabla u + g\Delta u + \mu\sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial u}{\partial x_j} + hp^ip^j\frac{\partial^2 u}{\partial x_i\partial x_j}\right) - h(u-f_x) = 0$$

Por un argumento análogo:

$$\nabla g\cdot\nabla v + g\Delta v + \mu\sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\partial(hp^ip^j)}{x_i}\frac{\partial v}{\partial x_j} + hp^ip^j\frac{\partial^2 v}{\partial x_i\partial x_j}\right) - h(v-f_y) = 0$$

lo cual concluye la demostración ■

# Apéndice B

# Análisis Numérico

## B.1. Diferencias Finitas y EPGVF

Se explicita en esta sección la aproximación por diferencias finitas de la ecuación 2.3, se definen:

$$
\begin{aligned}
x_i &= i\Delta x & g_{ij} &= g(|\nabla f(x_i, y_j)|) \\
y_j &= j\Delta y & h_{ij} &= h(|\nabla f(x_i, y_j)|) \\
t_n &= n\Delta t & p^1_{ij} &= p^1(x_i, y_j) \\
u^n_{ij} &= u(x_i, y_j, t_n) & p^2_{ij} &= p^2(x_i, y_j) \\
f_{ij} &= f(x_i, y_j)
\end{aligned}
$$

Se aproximan las derivadas por las siguientes diferencias finitas:

$$
\begin{aligned}
\frac{\partial u}{\partial x}(x_i, y_j, t_n) &\approx \frac{u^n_{i+1j} - u^n_{i-1j}}{2\Delta x} & \frac{\partial u}{\partial y}(x_i, y_j, t_n) &\approx \frac{u^n_{ij+1} - u^n_{ij-1}}{2\Delta y} \\
\frac{\partial^2 u}{\partial x^2}(x_i, y_j, t_n) &\approx \frac{u^n_{i+1j} - 2u^n_{ij} + u^n_{i-1j}}{\Delta x^2} & \frac{\partial^2 u}{\partial y^2}(x_i, y_j, t_n) &\approx \frac{u^n_{ij+1} - 2u^n_{ij} + u^n_{ij+1}}{\Delta y^2} \\
\frac{\partial u}{\partial t}(x_i, y_j, t_n) &\approx \frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta t} & \frac{\partial^2 u}{\partial x \partial y}(x_i, y_j, t_n) &\approx \frac{u^n_{i+1j+1} - u^n_{i+1j-1} + u^n_{i-1j-1} - u^n_{i-1j+1}}{2\Delta x \Delta y}
\end{aligned}
$$

Estas aproximaciones son las más simples con error de segundo orden.

Las aproximaciones de las derivadas de $f$, $g$ y $h$ son las mismas que las del $u$. Notemos que la ecuación 2.3 es equivalente a :

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial g}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial g}{\partial y}\frac{\partial u}{\partial y} + g\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \mu\left(h(p^1)^2\frac{\partial^2 u}{\partial x^2} + 2hp^1p^2\frac{\partial^2 u}{\partial x \partial y} + h(p^2)^2\frac{\partial^2 u}{\partial y^2}\right.\\
&+ \left(\frac{\partial h}{\partial x}(p^1)^2 + 2h\frac{\partial p^1}{\partial x}p^1\right)\frac{\partial u}{\partial x} + \left(\frac{\partial h}{\partial y}(p^2)^2 + 2h\frac{\partial p^2}{\partial y}p^2\right)\frac{\partial u}{\partial y}\\
&+ \left(\frac{\partial h}{\partial x}p^1p^2 + h\frac{\partial p^1}{\partial x}p^2 + hp^1\frac{\partial p^2}{\partial x}\right)\frac{\partial u}{\partial y} + \left.\left(\frac{\partial h}{\partial y}p^1p^2 + h\frac{\partial p^1}{\partial y}p^2 + hp^1\frac{\partial p^2}{\partial y}\right)\frac{\partial u}{\partial x}\right)\\
&- h(u - \frac{\partial f}{\partial x})\end{aligned}$$

Luego aproximando por diferencias finitas:

$$\begin{aligned}\frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta t} &= \frac{(g_{i+1j} - g_{i-1j}) - (u_{i+1j}^n - u_{i-1j}^n)}{4\Delta x^2} + \frac{(g_{ij-1} - g_{ij-1}) - (u_{ij+1}^n - u_{ij-1}^n)}{4\Delta y^2}\\
&+ g_{ij}\left(\frac{u_{i+1j}^n - 2u_{ij}^n + u_{i-1j}^n}{\Delta x^2} + \frac{u_{ij+1}^n - 2u_{ij}^n + u_{ij+1}^n}{\Delta y^2} + \right)\\
&+ \mu\left\{h_{ij}(p_{ij}^1)^2\frac{u_{i+1j}^n - 2u_{ij}^n + u_{i-1j}^n}{\Delta x^2} + h_{ij}(p_{ij}^2)^2\frac{u_{ij+1}^n - 2u_{ij}^n + u_{ij-1}^n}{\Delta y^2}\right.\\
&+ 2h_{ij}p_{ij}^1p_{ij}^2\frac{u_{i+1j+1}^n - u_{i+1j-1}^n + u_{i-1j-1}^n - u_{i-1j+1}^n}{2\Delta x\Delta y}\\
&+ \left(\frac{h_{i+1j} - h_{i-1j}}{2\Delta x}(p_{ij}^1)^2 + 2h_{ij}\frac{p_{i+1j}^1 - p_{i-1j}^1}{2\Delta x}p_{ij}^1\right)\frac{u_{i+1j}^n - u_{i-1j}^n}{2\Delta x}\\
&+ \left(\frac{h_{ij+1} - h_{ij-1}}{2\Delta y}(p^2)^2 + 2h_{ij}\frac{p_{ij+1}^2 - p_{ij-1}^2}{2\Delta y}p_{ij}^2\right)\frac{u_{ij+1}^n - u_{ij-1}^n}{2\Delta y}\\
&+ \left(\frac{h_{i+1j} - h_{i-1j}}{2\Delta x}p_{ij}^1p_{ij}^2 + h_{ij}\frac{p_{i+1j}^1 - p_{i-1j}^1}{2\Delta x}p_{ij}^2 + h_{ij}p_{ij}^1\frac{p_{i+1j}^2 - p_{i-1j}^2}{2\Delta x}\right)\frac{u_{ij+1}^n - u_{ij-1}^n}{2\Delta y}\\
&+ \left.\left(\frac{h_{ij+1} - h_{ij-1}}{2\Delta y}p_{ij}^1p_{ij}^2 + h_{ij}\frac{p_{ij+1}^1 - p_{ij-1}^1}{2\Delta y}p_{ij}^2 + h_{ij}p_{ij}^1\frac{p_{ij+1}^2 - p_{ij-1}^2}{2\Delta y}\right)\frac{u_{i+1j}^n - u_{i-1j}^n}{2\Delta x}\right\}\\
&- h_{ij}(u_{ij}^n - \frac{f_{i+1j} - f_{i-1j}}{2\Delta x})\end{aligned}$$

# Bibliografía


[DF07] Runhong Deng and M.D. Fox, *Parametric optimization for epgvf snake using anova and taguchi method*, March 2007, pp. 108–109. 48

[DTHJM07] L. De Tullio, S. Härtel, J. Jara, and Fanani M.L., *The initial surface composition and topography modulate sphingomyelinase-driven sphingomyelin to ceramide conversion in lipid monolayers*, Cell Biochemistry and Biophysics **47** (2007), 169–177. 1

[Eri04] Christer Ericson, *Real-time collision detection (the morgan kaufmann series in interactive 3-d technology) (the morgan kaufmann series in interactive 3d technology)*, Morgan Kaufmann, December 2004. 28

[FHO02] M. L. Fanani, S. Härtel, and R.G. Oliveira, *Bidirectional control of sphingomyelinase activity and surface topography in lipid monolayers*, Biophysical Journal **83(6)** (2002), 3416–3424. 1

[HFM05] S. Härtel, M. L. Fanani, and B. Maggio, *Shape transitions and lattice structuring of ceramide-enriched domains generated by sphingomyelinase in lipid monolayers.*, Byophisical Journal **88** (2005), 287–304. 1

[HJLC06] S. Härtel, J. Jara, C.G Lemus, and M.L. Concha, *3d morpho-topological analysis of asymmetric neuronal morphogenesis in developing zebrafish*, Computational Modelling of Objects Represented in Images. Fundamentals, Methods and Applications: Proceedings of the International Symposium CompIMAGE 2006 (J. M. R. S. Tavares and J. R. M. Natal, eds.), Taylor & Francis, October 2006, pp. 215–220. 4

[HZKT+03] S. Härtel, M. Zorn-Kruppa, S. Tikhonova, P. Heino, M. Engelke, and H. Diehl, *Staurosporine-induced apoptosis in human cornea epithelial cells in vitro*, Cytometry **8** (2003), 15–23. 1

[Jar07] J. Jara, *Contornos activos en tres dimensiones para la segmentación de estructuras biológicas*, http://www.scian.cl/portal/globals_file.php?CS=2885&ID=1087361407.5591&D=ON, 2007, Proyecto de Título de Ingeniero Civil en Informática, Universidad Austral de Chile. 4, 22

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, *Snakes: Active contour models*, International Journal of Computer Vision **1** (1988), 321–331. 1, 3

[LLF05] Chunming Li, Jundong Liu, and Martin D. Fox, *Segmentation of external force field for automatic initialization and splitting of snakes*, Pattern Recognition **38** (2005), no. 11, 1947 – 1960. 19

[MT00] Tim McInerney and Demetri Terzopoulos, *T-snakes: Topology adaptive snakes*, Medical Image Analysis **4** (2000), no. 2, 73 – 91. 10, 11, 48

[Pal07] K. Palma, *Análisis morfo -topológico de la migración asimétrica de órgano parapineal en el desarrollo embrionario de pez cebra (danio rerio)*, 2007, Unidad de Investigación. Facultad de Medicina. Universidad de Chile. 4

[XP98] Chenyang Xu and Jerry L. Prince, *Generalized gradient vector flow external forces for active contours*, Signal Processing **71** (1998), 131–139. 8, 17, 37