Graph Cuts para Segmentación Óptima de Objetos: Análisis y Aplicaciones Prácticas

Basado en el trabajo de Boykov & Jolly (2001)

Javier Augusto Rebull Saucedo
Maestría en Inteligencia Artificial Aplicada
Instituto Tecnológico de Monterrey
Massachusetts, USA
A01795838@tec.mx

Juan Carlos Pérez Nava
Maestría en Inteligencia Artificial Aplicada
Instituto Tecnológico de Monterrey
Mexico City, Mexico
A01795941@tec.mx

Luis Gerardo Sánchez Salazar
Maestría en Inteligencia Artificial Aplicada
Instituto Tecnológico de Monterrey
California, USA
A01232963@tec.mx

Oscar Enrique García García
Maestría en Inteligencia Artificial Aplicada
Instituto Tecnológico de Monterrey
Mexico City, Mexico
A01016093@tec.mx

***Resumen*—La segmentación de imágenes es un paso fundamental en visión computacional. Este artículo analiza el método *Graph Cuts*, un enfoque de optimización global que formula la segmentación como un problema de energía mínima en un grafo. Explicamos cómo la interacción del usuario mediante semillas se combina con la información de píxeles vecinos para encontrar el corte óptimo (Min-Cut) que separa el objeto del fondo. Se presentan comparaciones con métodos alternativos (K-means, Mean Shift, Region Growing, Watershed, Superpixels) y se discuten ventajas, limitaciones y aplicaciones clave en edición de imágenes, segmentación médica y creación de datasets.**

***Index Terms*—Segmentación, Graph Cuts, Max-Flow/Min-Cut, Optimización Global, Visión Computacional**

I. Introducción

La segmentación consiste en dividir una imagen en regiones con significado, separando objetos de fondos. Este paso es esencial para que los sistemas de IA interpreten contenido visual y es especialmente crítico para la creación de datasets anotados en aprendizaje profundo, donde la anotación manual puede consumir entre 5-10 minutos por imagen [11].

Graph Cuts es un método interactivo propuesto por Boykov y Jolly [1] que ofrece una solución óptima y precisa, requiriendo solo una mínima guía del usuario para lograr resultados de alta calidad con garantías matemáticas de optimalidad global.

II. El Método Graph Cuts

II-A. Conceptualización

El método modela la imagen como un grafo $G = (V, E)$: una red de nodos y conexiones donde:

- **Nodos regulares**: Cada píxel de la imagen se convierte en un nodo.

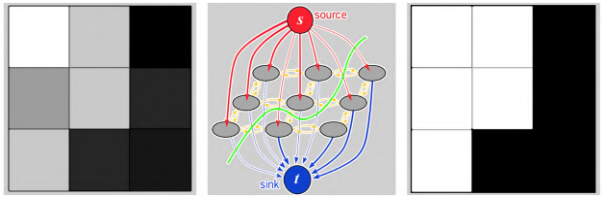


Figura 1: Transformación de imagen a grafo. Los píxeles se convierten en nodos conectados a los terminales s (objeto) y t (fondo). El corte mínimo define la segmentación.

- **Nodos terminales**: Se añaden dos nodos especiales: la Fuente (*Source* s, representa al "Objeto") y el Sumidero (*Sink* t, representa al "Fondo").
- **Aristas**: Conectan píxeles vecinos (n-links) y píxeles con terminales (t-links).

El objetivo es encontrar un "corte" que separe todos los nodos de objeto (conectados a s) de los nodos de fondo (conectados a t) minimizando una función de energía.

II-B. Formulación Matemática

La segmentación óptima minimiza la función de energía:

$$E(f) = \lambda \cdot R(f) + B(f) \tag{1}$$

donde $R(f)$ es el **término regional** que mide la coherencia con los modelos de apariencia, $B(f)$ es el **término de borde** que penaliza discontinuidades entre píxeles similares, y λ balancea ambos términos.

II-B1. Pesos de Terminal (t-links)

Los costos hacia los terminales se definen según las semillas del usuario y modelos probabilísticos:

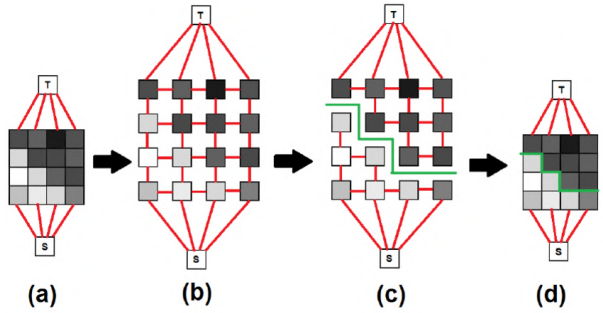


Figura 2: Pipeline interactivo: (a) Usuario marca semillas, (b) Se construyen modelos y pesos, (c) Max-Flow encuentra el corte óptimo, (d) Resultado final.

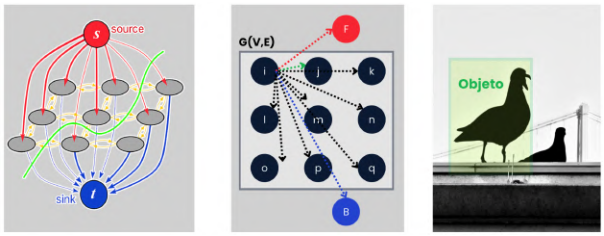


Figura 3: El algoritmo Max-Flow/Min-Cut encuentra el corte de costo mínimo que separa s de t, definiendo la segmentación óptima.

$$w(p,s) = \begin{cases} K & \text{si } p \in \text{Semilla Objeto} \\ -\lambda \log P(I_p|\text{Objeto}) & \text{en otro caso} \end{cases} \tag{2}$$

$$w(p,t) = \begin{cases} K & \text{si } p \in \text{Semilla Fondo} \\ -\lambda \log P(I_p|\text{Fondo}) & \text{en otro caso} \end{cases} \tag{3}$$

donde K es un valor suficientemente grande y $P(I_p)$ representa la probabilidad de que el píxel pertenezca a cada clase según modelos construidos de las semillas (típicamente histogramas o GMM).

II-B2. Pesos de Borde (n-links)

Los costos entre píxeles vecinos dependen de su similitud:

$$w(\{p,q\}) = \frac{1}{\text{dist}(p,q)} \cdot \exp\left(-\frac{\|I_p - I_q\|^2}{2\sigma^2}\right) \tag{4}$$

Esta fórmula penaliza cortes entre píxeles similares (alto peso) y facilita cortes en bordes naturales (bajo peso).

II-C. Algoritmo de Solución: Max-Flow/Min-Cut

Minimizar la energía en (1) es equivalente a encontrar el corte mínimo en G. Por el teorema de max-flow/min-cut [5], esto se resuelve eficientemente mediante algoritmos de flujo máximo. El algoritmo especializado de Boykov-Kolmogorov [2] aprovecha la estructura de grid de las imágenes para lograr tiempos de procesamiento de 2-5 segundos en imágenes típicas.

III. Comparación con Otros Métodos

Graph Cuts no es el único método de segmentación disponible. A continuación comparamos con alternativas prominentes:

III-A. K-means Clustering

Particiona píxeles en k grupos basándose en similitud de color [6]. **Ventajas**: Muy rápido y simple. **Desventajas vs. Graph Cuts**: No considera estructura espacial, produce bordes irregulares, sensible a inicialización, sin garantías de optimalidad global.

III-B. Mean Shift

Busca modas en el espacio de características mediante ascenso por gradiente [7]. **Ventajas**: No requiere especificar número de clusters. **Desventajas vs. Graph Cuts**: Computacionalmente costoso ($O(n^2)$), parámetro de bandwidth crítico, tiende a sobre-segmentar.

III-C. Region Growing

Expande regiones desde semillas agregando píxeles vecinos similares [8]. **Ventajas**: Simple e intuitivo. **Desventajas vs. Graph Cuts**: Decisiones greedy locales, propenso a "fugas" a través de bordes débiles, sin optimalidad global.

III-D. Watershed

Trata la imagen como superficie topográfica e "inunda" desde mínimos locales [9]. **Ventajas**: Excelente para objetos tocándose. **Desventajas vs. Graph Cuts**: Sobre-segmentación severa, muy sensible al ruido, requiere pre-procesamiento intensivo.

III-E. Superpixels

Agrupa píxeles en pequeñas regiones homogéneas [10]. **Relación con Graph Cuts**: Son complementarios, no competidores. Los superpixels pueden usarse *junto con* Graph Cuts como pre-procesamiento, reduciendo la complejidad computacional.

III-F. Tabla Comparativa

La Tabla I resume las características clave de cada método.

Cuadro I: Comparación de Métodos de Segmentación

Método	Prec.	Veloc.	Interact.	Optimal.	Uso
Graph Cuts	**Exc.**	M. Alta	Sí	Global	Datasets
K-means	Baja	M. Alta	No	Local	Seg. rápida
Mean Shift	Media	Media	No	Local	Color
Region Growing	Media	Alta	Parcial	Local	Homogén.
Watershed	Alta	Media	Parcial	–	Objetos
Superpixels	Media	M. Alta	No	–	Pre-proc.

IV. Ventajas y Limitaciones

IV-A. Fortalezas Principales

1. **Optimización Global**: Garantiza encontrar el corte de costo mínimo, a diferencia de métodos greedy locales.
2. **Precisión en Bordes**: Captura detalles finos y produce bordes suaves naturalmente, típicamente logrando IoU > 0.95 [4].
3. **Interactividad Eficiente**: Solo 15-20 semillas bien colocadas son suficientes para resultados excelentes.

4. **Fundamentación Matemática**: Base teórica rigurosa en teoría de grafos y optimización.
5. **Flexibilidad**: Se extiende naturalmente a 3D, multi-etiqueta y video.

IV-B. Limitaciones

1. **Requiere Interacción**: No es completamente automático, impidiendo procesamiento masivo sin supervisión.
2. **Sensibilidad a Parámetros**: El parámetro λ debe ajustarse según la aplicación.
3. **Consumo de Memoria**: Para imágenes de alta resolución puede requerir técnicas especializadas.
4. **Bordes Difusos**: Asume bordes relativamente claros; elementos semi-transparentes son desafiantes.
5. **Velocidad**: 0.5-5 segundos por imagen, no apto para tiempo real sin optimizaciones.

V. Casos de Uso y Aplicaciones

V-A. Creación de Datasets para Machine Learning

Esta es quizás la aplicación más valiosa en la era del deep learning. Graph Cuts reduce el tiempo de anotación de 5-10 minutos a 30-60 segundos por imagen, logrando un **ahorro del 80-90 %** en costos mientras mantiene o mejora la calidad de las máscaras [12].

V-B. Segmentación Médica

Extensamente usado para delimitar órganos, tumores y vasos sanguíneos en imágenes médicas (MRI, CT, angiografías). Estudios clínicos demuestran reducción de tiempo de segmentación manual de 30-45 minutos a 3-5 minutos para volúmenes 3D completos [4].

V-C. Edición Profesional de Fotografías

GrabCut [3], basado en Graph Cuts, es la base de herramientas como "Select Subject" en Adobe Photoshop. Aplicaciones incluyen extracción de objetos para comercio electrónico, composición de imágenes y ajustes selectivos.

V-D. Otras Aplicaciones

- **Robótica y Vehículos Autónomos**: Percepción de objetos manipulables y segmentación de áreas transitables.
- **Imágenes Satelitales**: Monitoreo de deforestación, agricultura de precisión, respuesta a desastres.
- **Generación de Stickers**: Extracción automática de objetos con bordes limpios para contenido digital.

VI. Demo Interactiva en Google Colab

Para una demostración práctica de estos conceptos, invitamos al lector a ejecutar nuestro notebook interactivo en Google Colab, que permite cargar imágenes, dibujar semillas y aplicar Graph Cuts en tiempo real:

https://colab.research.google.com/drive/1h27LiBLhfs9OkNtox-L2tZjxePLyynRC

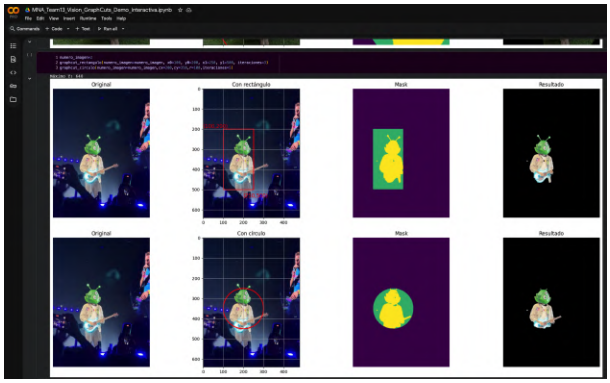


Figura 4: Demostración de la aplicación interactiva en Colab.

La implementación del sistema de segmentación basado en Graph Cuts se acompaña de una demostración interactiva desarrollada en Google Colab, diseñada para facilitar la experimentación y validación del algoritmo. Esta plataforma permite a los usuarios explorar las capacidades del sistema mediante interfaces intuitivas que combinan visualización en tiempo real con control paramétrico de las operaciones de segmentación.

El entorno interactivo proporciona tres niveles de funcionalidad: demostraciones básicas predefinidas, herramientas para creación de datasets personalizados, y un playground completamente configurable que permite definir regiones de interés mediante primitivas geométricas.

VI-A. Demostración de Segmentación Básica

La función `demo_segmentacion_basica()` ilustra el proceso fundamental de segmentación utilizando máscaras ground truth predefinidas. La implementación permite observar la efectividad del algoritmo al separar objetos de interés del fondo mediante la especificación explícita de regiones foreground y background.

La funcionalidad se extiende mediante selección interactiva de semillas, donde el usuario define manualmente puntos de foreground (verde) y background (rojo) directamente sobre la imagen. Este enfoque proporciona control fino sobre el proceso de segmentación sin requerir máscaras completas predefinidas.

VI-B. Creación de Dataset Personalizado

La función `demo_creacion_dataset()` permite generar conjuntos de datos customizados para entrenamiento o validación. Esta herramienta facilita la construcción de datasets anotados mediante la especificación de máscaras de segmentación, exportando tanto las imágenes originales como sus correspondientes ground truth en formato estructurado.

VI-C. Playground Interactivo

El sistema incluye un conjunto de funciones interactivas que permiten experimentar con diferentes configuraciones:

- `graphcut_rectangulo()`: Segmentación con ROIs rectangulares
- `graphcut_circulo()`: Segmentación con ROIs circulares
- `graphcut_custom()`: Múltiples ROIs combinados

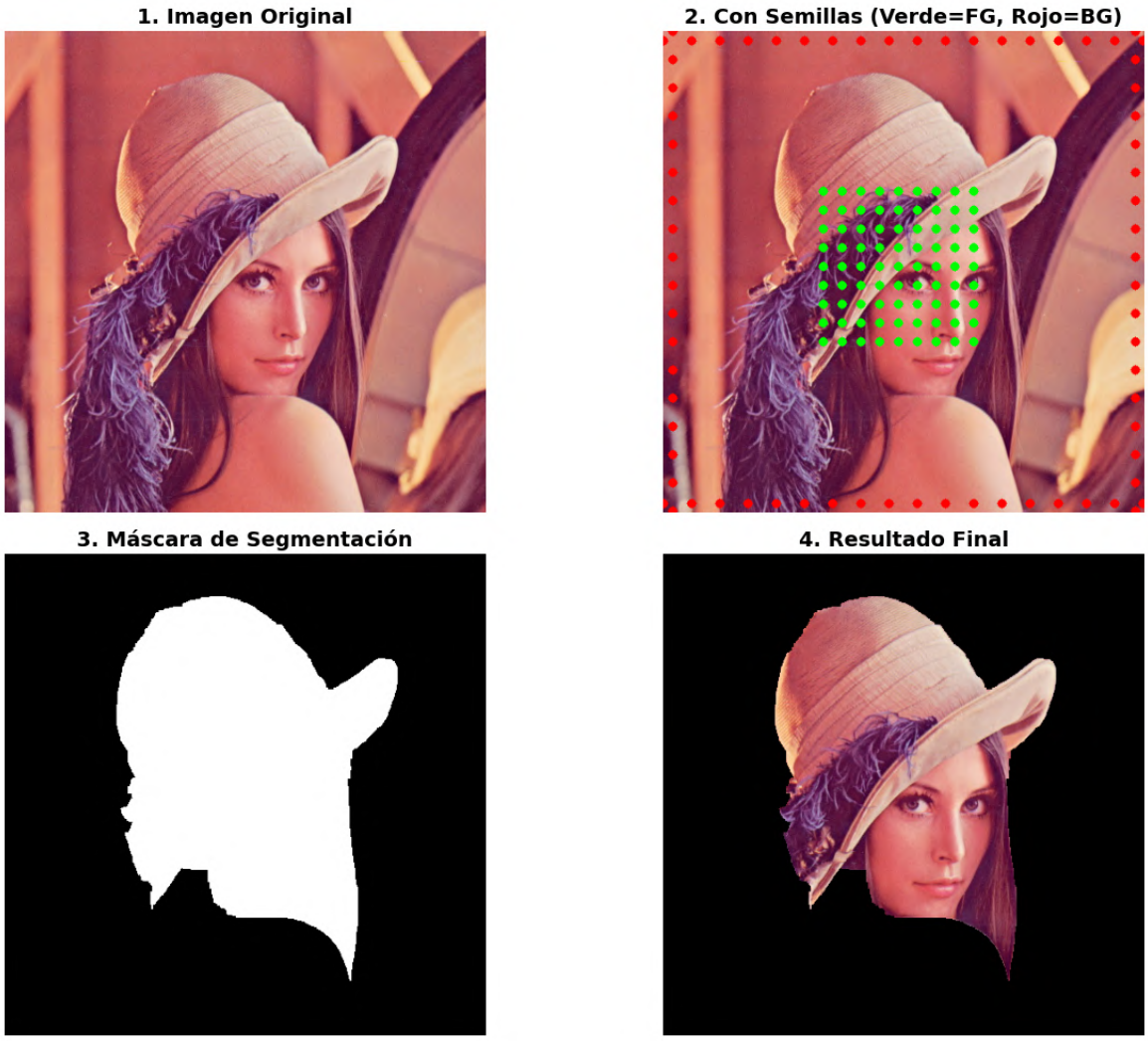


Figura 5: Segmentación del sombrero de Lena utilizando máscara ground truth. De izquierda a derecha: imagen original, máscara de referencia, resultado de segmentación.

Figura 6: Pipeline de segmentación interactiva mediante selección de semillas. (1) Imagen original, (2) semillas de foreground (verde) y background (rojo), (3) máscara de segmentación generada, (4) resultado final aplicado.

Los parámetros incluyen `numero_imagen`, coordenadas espaciales, radio para regiones circulares, y número de `iteraciones`. Las Figs. 7a-8d muestran resultados representativos con diferentes configuraciones de primitivas geométricas aplicadas a diversas imágenes del dataset.

Los ejemplos presentados demuestran la capacidad del sistema para adaptarse a diferentes escenarios de segmentación, desde objetos con bordes bien definidos hasta regiones complejas que requieren combinación de múltiples ROIs. La flexibilidad en la definición de primitivas geométricas permite al usuario iterar rápidamente sobre diferentes estrategias de segmentación sin modificar el código subyacente.

VII. Aplicación Interactiva en Python de Graph Cuts

Para reproducir este proyecto seguir las instrucciones del repositorio GitHub:

https://github.com/jrebull/MNA_Vision_Team13_GraphCuts

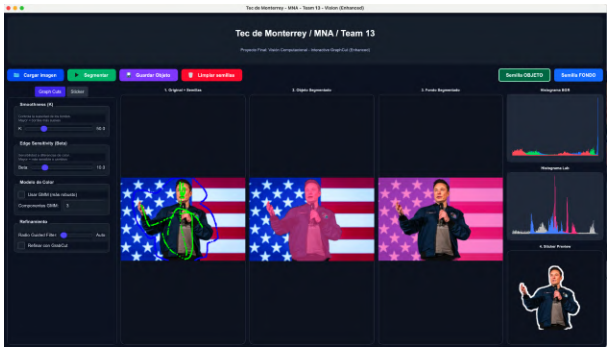


Figura 9: Demostración de la aplicación interactiva del proyecto.

Nuestra implementación de segmentación interactiva basada en Graph Cuts está diseñada como un sistema modular que integra algoritmos clásicos de visión computacional con una interfaz de usuario moderna e intuitiva. La arquitectura se divide en tres componentes principales: el motor de procesamiento (backend), la interfaz gráfica (frontend), y los módulos de refinamiento y post-procesamiento.

VII-A. Componentes Principales

VII-A1. Backend: Motor GraphMaker

El núcleo del sistema está implementado en Python, aprovechando bibliotecas especializadas para procesamiento de imágenes y optimización:

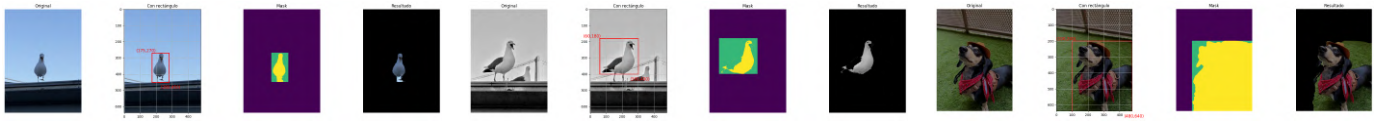
- **OpenCV (cv2)**: Manipulación de imágenes, conversión de espacios de color, filtros morfológicos
- **PyMaxFlow**: Implementación optimizada del algoritmo de Boykov-Kolmogorov para max-flow/min-cut
- **NumPy**: Operaciones matriciales de alto rendimiento
- **scikit-learn**: Gaussian Mixture Models (GMM) para modelado de color
- **PIL/Pillow**: Generación de stickers y renderizado de texto

La clase principal `GraphMaker` encapsula toda la lógica de segmentación, manteniendo el estado de la imagen, las semillas marcadas por el usuario, y los parámetros de configuración.

VII-A2. Frontend: Interfaz PyQt5

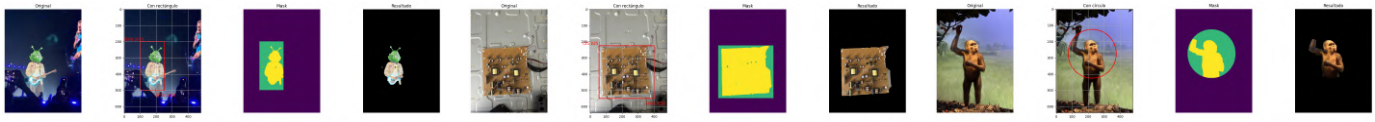
La interfaz de usuario está construida con PyQt5, proporcionando:

- **Sistema de tabs**: Organización de parámetros en categorías (Graph Cuts, Color Model, Post-Processing)
- **Canvas interactivo**: Marcado de semillas con selección de color (rojo para foreground, azul para background)

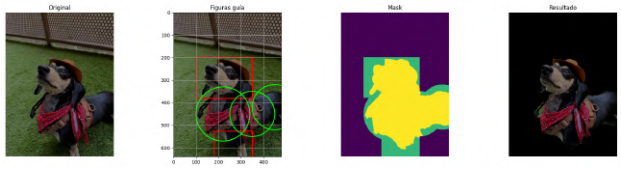


(a) Perro con sombrero usando formas guía (b) Paloma con ROI circular (c) Placa electrónica

Figura 7: Resultados del playground interactivo (parte 1). Cada imagen muestra: original, primitivas geométricas, máscara, y resultado final.



(a) Guitarrista con casco (b) Perro con rectángulo extenso (c) Gaviota sobre baranda



(d) Múltiples ROIs combinados

Figura 8: Resultados del playground interactivo (parte 2). Los ejemplos demuestran la versatilidad del sistema en diferentes escenarios de segmentación.

- **Visualización en tiempo real**: Cuatro vistas simultáneas (Original, Semillas, Modelo de Color, Resultado)
- **Controles de exportación**: Generación de máscaras binarias y stickers con transparencia

VII-A3. Módulos de Refinamiento

El sistema incorpora algoritmos complementarios para mejorar la calidad de la segmentación:

- **GrabCut**: Refinamiento iterativo basado en GMM
- **Guided Filter**: Suavizado de bordes preservando estructura
- **Morfología**: Operaciones de apertura/cierre para limpieza de máscaras

VII-B. Diagrama de Arquitectura

VII-C. Flujo de Datos

El pipeline de procesamiento sigue la secuencia:

1. **Carga de Imagen**: Conversión automática a espacio de color Lab para mejor separabilidad perceptual
2. **Marcado de Semillas**: El usuario marca píxeles representativos de foreground (rojo) y background (azul)
3. **Construcción de Modelos**: Creación de modelos probabilísticos Gaussianos o GMM a partir de las semillas
4. **Construcción del Grafo**: Generación de n-links (vecindad) y t-links (terminales) con pesos calculados
5. **Optimización**: Ejecución del algoritmo max-flow/min-cut para encontrar la segmentación óptima
6. **Refinamiento (Opcional)**: Aplicación de GrabCut para mejora iterativa
7. **Post-procesamiento**: Guided Filter para suavizado de bordes y operaciones morfológicas
8. **Exportación**: Generación de máscara binaria o sticker con transparencia

VIII. Fundamentos Técnicos

VIII-A. Algoritmo Graph Cuts

VIII-A1. Formulación del Problema

La segmentación se formula como un problema de minimización de energía sobre un grafo $G = (V, E)$ donde V contiene todos los píxeles de la imagen más dos nodos terminales especiales: source s (foreground) y sink t (background). La función de energía a minimizar es:

$$E(f) = \lambda \cdot R(f) + B(f) \tag{5}$$

donde $f : \mathcal{P} \to \{0, 1\}$ es la función de etiquetado que asigna cada píxel $p \in \mathcal{P}$ a foreground (1) o background (0).

VIII-A2. Término Regional (Data Term)

El término regional $R(f)$ mide qué tan bien cada píxel se ajusta a los modelos de foreground y background:

$$R(f) = \sum_{p \in \mathcal{P}} R_p(f_p) \tag{6}$$

donde:

$$R_p(f_p) = \begin{cases} -\log P(I_p|\text{foreground}) & \text{si } f_p = 1 \\ -\log P(I_p|\text{background}) & \text{si } f_p = 0 \end{cases} \tag{7}$$

Las probabilidades $P(I_p|\cdot)$ se estiman mediante modelos Gaussianos o GMM construidos a partir de las semillas marcadas por el usuario.

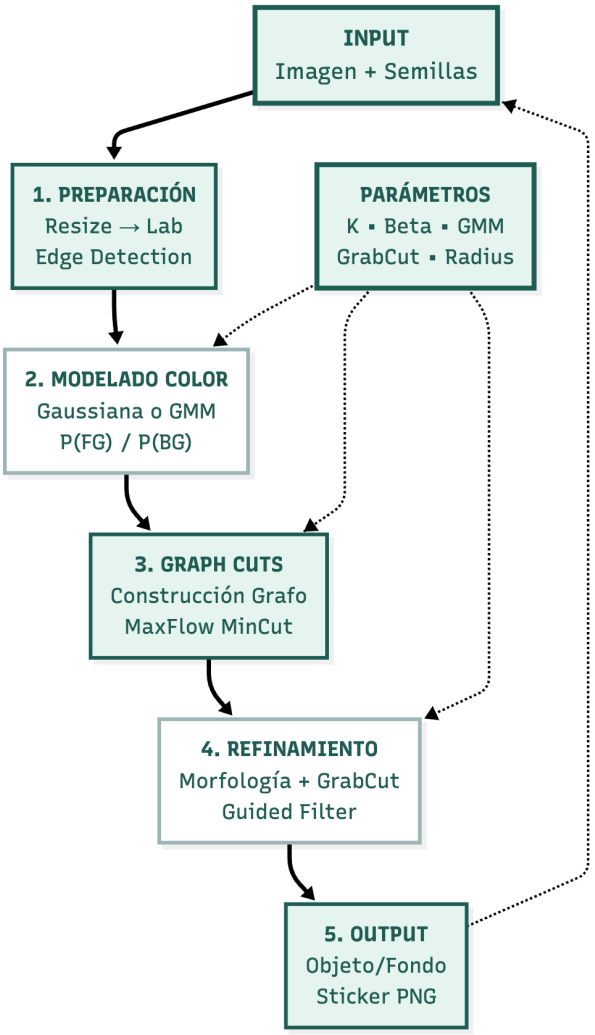


Figura 10: Arquitectura del sistema de segmentación interactiva. El flujo de datos comienza con la carga de imagen, seguido por marcado interactivo de semillas, construcción del grafo de energía, optimización mediante max-flow, y finalmente post-procesamiento opcional con GrabCut y Guided Filter.

VIII-A3. Término de Suavidad (Smoothness Term)

El término de borde $B(f)$ penaliza discontinuidades entre píxeles vecinos similares:

$$B(f) = \sum_{\{p,q\}\in\mathcal{N}} B_{\{p,q\}} \cdot \delta(f_p \neq f_q) \tag{8}$$

donde $\mathcal{N}$ representa las conexiones de vecindad y el peso $B_{\{p,q\}}$ se calcula como:

$$B_{\{p,q\}} = K \cdot \exp\left(-\frac{\|I_p - I_q\|^2}{2\sigma^2}\right) \cdot \frac{1}{\text{dist}(p,q)} \tag{9}$$

El parámetro K controla la importancia del término de suavidad, σ se estima como la desviación estándar de las diferencias de color en la imagen, y $\text{dist}(p,q)$ es la distancia euclidiana (1 para vecinos de 4-conectividad, $\sqrt{2}$ para diagonales).

VIII-A4. Equivalencia con Min-Cut

La minimización de $E(f)$ es equivalente a encontrar un corte mínimo en el grafo que separe s de t. Este problema se resuelve eficientemente mediante el algoritmo de max-flow de Boykov-Kolmogorov, que aprovecha la estructura de grid regular de las imágenes para lograr rendimiento superior a algoritmos genéricos.

VIII-B. Modelo de Color

VIII-B1. Espacio de Color Lab

Trabajamos en el espacio CIE Lab en lugar de RGB por varias razones técnicas:

- **Perceptualmente uniforme**: Distancias euclidianas en Lab aproximan diferencias perceptuales humanas
- **Separación luminancia-crominancia**: El canal L es independiente de a* y b*, facilitando el modelado
- **Mejor separabilidad**: Objetos con colores distintos se separan mejor en Lab que en RGB

VIII-B2. Modelos Probabilísticos

Implementamos dos opciones de modelado:

Gaussiana Simple: Para distribuciones unimodales:

$$P(I_p|\text{clase}) = \mathcal{N}(I_p; \mu, \Sigma) \tag{10}$$

donde μ es el vector de medias y Σ la matriz de covarianza estimados de las semillas.

Gaussian Mixture Model (GMM): Para distribuciones multimodales:

$$P(I_p|\text{clase}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(I_p; \mu_k, \Sigma_k) \tag{11}$$

donde K es el número de componentes (típicamente 3-5), π_k son los pesos de mezcla, y cada componente tiene su propia media μ_k y covarianza Σ_k.

El GMM es crítico para objetos multicolor (ej. chamarra tricolor) o cuando foreground y background tienen distribuciones complejas.

VIII-C. Post-procesamiento

VIII-C1. Guided Filter

El Guided Filter propuesto por He et al. (2013) refina los bordes de la segmentación mientras preserva estructura:

$$\text{Output}_p = a_k I_p + b_k, \quad \forall p \in \omega_k \tag{12}$$

donde ω_k es una ventana local centrada en k, y los coeficientes a_k, b_k se calculan minimizando la diferencia con la máscara original sujeto a un término de regularización. El radio de la ventana controla el grado de suavizado.

VIII-C2. GrabCut como Refinamiento

GrabCut se utiliza como paso de post-procesamiento iterativo que refina la segmentación inicial de Graph Cuts:

1. Se inicializa con la máscara obtenida de Graph Cuts
2. Se construyen GMMs de 5 componentes para foreground y background

3. Se itera entre optimización de GMMs y segmentación hasta convergencia (típicamente 5 iteraciones)

Este refinamiento es especialmente efectivo para bordes finos (cabello, plumas) y corrección de errores menores.

IX. Implementación - Código Principal

IX-A. Clase GraphMaker: Funciones Core

IX-A1. Carga y Preprocesamiento

`load_image_data(image_path, max_size=800)`: Carga la imagen y la redimensiona proporcionalmente si excede el tamaño máximo, manteniendo la relación de aspecto. Convierte automáticamente de RGB a Lab para procesamiento.

IX-A2. Gestión de Semillas

`add_seed(x, y, label, brush_size=3)`: Registra semillas marcadas por el usuario. El parámetro `label` es 1 para foreground (rojo) o 0 para background (azul). El `brush_size` permite marcar regiones en lugar de píxeles individuales, acelerando el proceso de marcado.

IX-A3. Construcción de Modelos

`_build_color_models()`: Extrae los píxeles correspondientes a las semillas y entrena los modelos probabilísticos:

- Si GMM está desactivado: calcula media y covarianza para distribuciones Gaussianas simples
- Si GMM está activado: usa scikit-learn's GaussianMixture para entrenar modelos con el número especificado de componentes

Incluye validación para casos degenerados (pocas semillas, semillas idénticas).

IX-A4. Modelo de Bordes

`_build_edge_model()`: Calcula el parámetro σ en la ecuación 9 como:

$$\sigma = \text{std}(\|I_p - I_q\|) \quad \forall \{p, q\} \in \mathcal{N} \tag{13}$$

Opcionalmente aplica detección de bordes Canny para identificar regiones con gradientes fuertes, permitiendo ajustar los pesos de n-links basándose en la presencia de bordes naturales.

IX-A5. Construcción del Grafo

`_build_graph()`: Genera el grafo de energía completo:

1. **Inicialización**: Crea grafo con PyMaxFlow de tamaño $n \times m$ (dimensiones de la imagen)
2. **T-links (terminal links)**: Para cada píxel p:
   - Si p es semilla de foreground: $w(p, s) = K_{\text{large}}$, $w(p, t) = 0$
   - Si p es semilla de background: $w(p, s) = 0$, $w(p, t) = K_{\text{large}}$
   - Si p no tiene semilla:

$$w(p, s) = -\lambda \log P(I_p|\text{foreground})$$
$$w(p, t) = -\lambda \log P(I_p|\text{background})$$

3. **N-links (neighbor links)**: Para cada par de píxeles vecinos $\{p, q\}$, se calcula el peso según la ecuación 9, multiplicado por el parámetro Beta.

IX-A6. Pipeline de Segmentación

`create_segment()`: Ejecuta el pipeline completo:

1. Validar que existan semillas de ambas clases
2. Construir modelos de color (`_build_color_models`)
3. Calcular parámetros de borde (`_build_edge_model`)
4. Construir grafo de energía (`_build_graph`)
5. Ejecutar max-flow/min-cut (PyMaxFlow)
6. Extraer etiquetas resultantes como máscara binaria
7. Aplicar refinamiento opcional (GrabCut, Guided Filter)
8. Retornar máscara final

IX-A7. Refinamiento con GrabCut

`_refine_with_grabcut(mask, iterations=5)`: Utiliza la implementación de OpenCV para refinamiento iterativo.

IX-A8. Suavizado de Máscara

`refine_mask(mask, radius)`: Aplica Guided Filter usando la imagen original como guía.

IX-B. Interfaz de Usuario (PyQt5)

IX-B1. Organización en Tabs

La interfaz separa los controles en tres categorías:

Tab 1 - Graph Cuts Parameters:

- Slider para K (Smoothness): rango 1-200
- Slider para Beta (Edge Sensitivity): rango 1-50
- Checkbox para habilitar detección de bordes Canny

Tab 2 - Color Model:

- Radio buttons: Gaussian vs GMM
- Slider para número de componentes GMM: 2-5
- Visualización de distribuciones (histogramas Lab)

Tab 3 - Post-Processing:

- Checkbox para GrabCut
- Slider para iteraciones de GrabCut: 1-10
- Slider para radio de Guided Filter: 5-50
- Checkbox para operaciones morfológicas

IX-B2. Canvas Interactivo

`ClickableLabel` hereda de QLabel e implementa eventos de mouse para trazado de semillas en tiempo real.

IX-B3. Sistema de Visualización

Cuatro paneles muestran simultáneamente:

1. **Original**: Imagen sin modificar
2. **Seeds**: Imagen con semillas superpuestas (rojo/azul)
3. **Color Model**: Visualización 2D de las distribuciones en espacio Lab
4. **Result**: Máscara binaria o composición RGBA

Cuadro II: Guía de Parámetros del Sistema

Parámetro	Rango	Default	Usar BAJO	Usar ALTO
K (Smoothness)	1-200	50	Bordes definidos, objetos sólidos	Fondos texturados, bokeh
Beta (Edge Sens.)	1-50	15	Colores similares (evita fragmentar)	Alto contraste cromático
GMM Components	2-5	3	Colores uniformes (cielo, paredes)	Objetos multicolor complejos
Guided Filter	5-50	15	Imágenes nítidas, bordes claros	Fondos desenfocados, bokeh
GrabCut Iters.	1-10	5	Casos fáciles (velocidad)	Bordes difíciles (calidad)

X. Parametrización del Sistema

X-A. Manual de Parámetros

X-B. Recomendaciones por Tipo de Imagen

X-B1. Fondos Uniformes

K=30-50, Beta=18-25, GMM=Off, GrabCut=On

X-B2. Fondos Texturados

K=70-90, Beta=12-16, GMM=On (4 comp), GrabCut=On

X-B3. Colores Similares

K=40-60, Beta=8-12, GMM=On (4-5 comp), GrabCut=On

X-B4. Fondos con Bokeh

K=80-95, Beta=10-15, GMM=On (3-4 comp), Guided Filter=25-40

X-B5. Imágenes Médicas

K=70-85, Beta=14-18, GMM=Off, GrabCut=On, Marcado denso

XI. Evaluación Experimental: Casos de Estudio

Para validar la efectividad y robustez de nuestra implementación, realizamos evaluación exhaustiva con nueve casos representando el espectro completo de dificultad.

XI-A. Caso 1: Alto Contraste con Fondo Uniforme

Imagen: Presidente Trump (Perfil)
Dificultad: Nivel 1 de 5 (Muy Fácil)

XI-A1. Caracterización

Objeto con tonos cálidos sobre fondo azul uniforme. Alto contraste cromático elimina ambigüedad.

XI-A2. Parámetros

K=50.0, Beta=16.0, Radio=15, GMM=No, GrabCut=Sí

Figura 11: Pipeline caso de alto contraste.

XI-A3. Análisis

Segmentación exitosa al primer intento con marcado mínimo. Este caso establece la línea base de rendimiento óptimo.

Figura 12: Resultado caso de alto contraste.

XI-B. Caso 2: Objeto en Vuelo

Imagen: Águila en el Aire
Dificultad: Nivel 2 de 5 (Fácil)

XI-B1. Caracterización

Ave con plumaje detallado contra cielo uniforme.

XI-B2. Parámetros

K=40.0, Beta=21.0, GMM=No, GrabCut=Sí

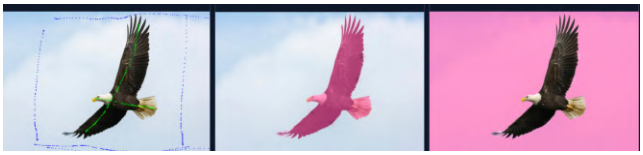


Figura 13: Pipeline objeto con geometría compleja.

Figura 14: Segmentación águila.

XI-C. Caso 3: Similitud Cromática Moderada

Imagen: Sombreros sobre Madera
Dificultad: Nivel 3 de 5 (Medio)

XI-C1. Parámetros

K=40.0, Beta=21.0, GMM=No, GrabCut=Sí

Figura 15: Pipeline similitud cromática.

Figura 16: Resultado sombreros.

XI-D. Caso 4: Fondo Heterogéneo

Imagen: Retrato OscarRAG
Dificultad: Nivel 3 de 5 (Medio-Alto)

XI-D1. Parámetros

K=50.0, Beta=10.0, Radio=25, GMM=No, GrabCut=Sí

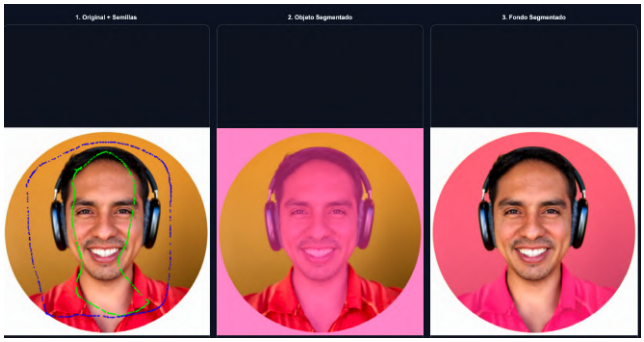


Figura 17: Pipeline retrato con fondo complejo.

XI-E. Caso 5: Pelaje Color Similar

Imagen: Perro Corgi (Simba)
Dificultad: Nivel 4 de 5 (Difícil)

XI-E1. Parámetros

K=42.0, Beta=9.0, Radio=15, GMM=Sí (4 comp), GrabCut=Sí

XI-F. Caso 6: Fondo Bokeh

Imagen: Presidente Maduro
Dificultad: Nivel 4 de 5 (Difícil)

XI-F1. Parámetros

K=86.0, Beta=11.0, Radio=25, GMM=Sí (4 comp), GrabCut=Sí

Figura 18: Resultado retrato.

Figura 19: Pipeline alta similitud cromática.

Figura 20: Resultado Corgi.

Figura 21: Pipeline fondo bokeh.

Figura 22: Resultado con bokeh.

XI-G. Caso 7: Escena de Playa

Imagen: Humano en Playa (Joss)
Dificultad: Nivel 4 de 5 (Difícil)

XI-G1. Parámetros

K=55.0, Beta=16.0, Radio=15, GMM=Sí (4 comp), GrabCut=Sí

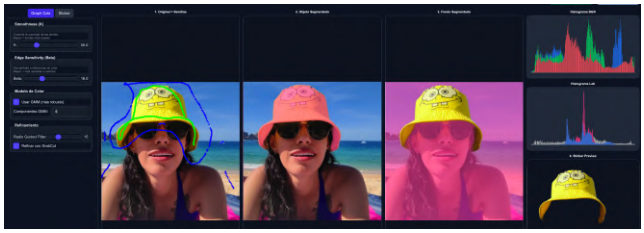


Figura 23: Pipeline escena de playa.

Figura 24: Resultado playa.

XI-H. Caso 8: Colores Idénticos

Imagen: Pez Payaso en Anémona
Dificultad: Nivel 5 de 5 (Extremo)

XI-H1. Parámetros

K=89.0, Beta=9.0, GMM=Sí (5 comp), GrabCut=Sí

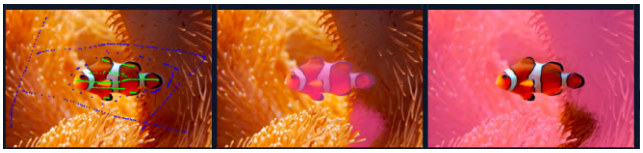


Figura 25: Pipeline caso extremo.

Figura 26: Resultado pez payaso.

XI-I. Caso 9: Imagen Médica

Imagen: Mamografía
Dificultad: Nivel 5 de 5 (Extremo)

XI-I1. Parámetros

K=75.0, Beta=16.0, Radio=15, GMM=No, GrabCut=Sí

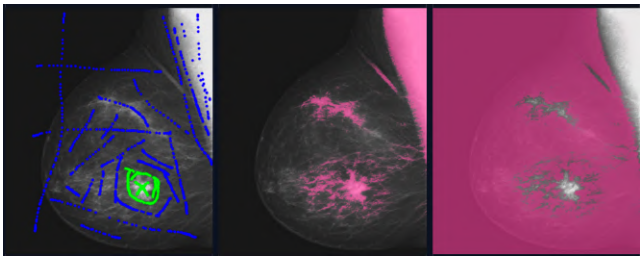


Figura 27: Pipeline imagen médica.

XI-J. Tabla Comparativa de Casos

XII. Aplicación: Exportación de Stickers con Transparencia

Una vez completada la segmentación, el sistema ofrece funcionalidad integrada para exportar los objetos segmentados como stickers digitales con canal alpha completo, listos para uso en aplicaciones de mensajería, diseño gráfico, o e-commerce.

Cuadro III: Análisis Comparativo de los Nueve Casos

Caso	Nivel	K	Beta	Radio	GMM	GrabCut	Desafío
Trump	1/5	50.0	16.0	15	No	Sí	Ninguno
Águila	2/5	40.0	21.0	–	No	Sí	Geometría fina
Sombreros	3/5	40.0	21.0	–	No	Sí	Similitud moderada
OscarRAG	3/5	50.0	10.0	25	No	Sí	Fondo heterogéneo
Simba	4/5	42.0	9.0	15	Sí (4)	Sí	Solapamiento alto
Maduro	4/5	86.0	11.0	25	Sí (4)	Sí	Bokeh
Joss	4/5	55.0	16.0	15	Sí (4)	Sí	Multicolor
Pez	5/5	89.0	9.0	–	Sí (5)	Sí	Colores idénticos
Mamografía	5/5	75.0	16.0	15	No	Sí	Sin color

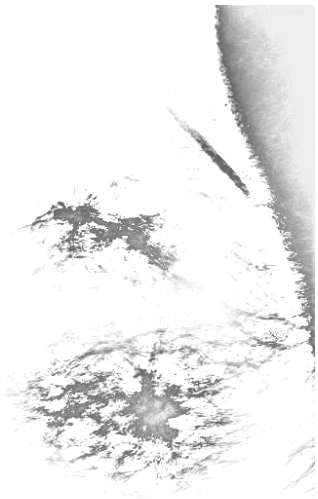


Figura 28: Resultado mamografía.

Figura 29: Sticker de OscarRAG con texto personalizado y borde decorativo blanco.

Figura 30: Sticker del Presidente Trump con transparencia completa, listo para uso en mensajería.

XII-A. Proceso de Generación

El flujo de trabajo para crear un sticker es directo y automatizado:

1. El sistema toma la máscara de segmentación obtenida (binaria: 0=background, 255=foreground)
2. Convierte la imagen original de RGB a RGBA agregando un canal alpha
3. Aplica la máscara al canal alpha: píxeles de foreground se vuelven opacos (alpha=255), píxeles de background transparentes (alpha=0)
4. Opcionalmente añade un contorno decorativo mediante dilatación morfológica de la máscara
5. Renderiza texto personalizado usando PIL/Pillow con fuentes TrueType
6. Exporta el resultado final como archivo PNG con compresión óptima

XII-B. Ejemplos de Stickers Generados

XII-C. Casos de Uso

Los stickers generados son directamente aplicables en:

- **Aplicaciones de mensajería**: Creación de paquetes personalizados para WhatsApp, Telegram, LINE, Slack
- **Marketing digital**: Assets promocionales rápidos sin necesidad de diseñadores
- **E-commerce**: Fotografía de productos sobre fondo transparente para catálogos en línea
- **Composiciones gráficas**: Elementos recortados para montajes fotográficos o diseño editorial
- **Redes sociales**: Contenido visual para Instagram Stories, memes, y publicaciones creativas

XIII. Retos Técnicos y Soluciones

XIII-A. Challenges Identificados

XIII-A1. Colores Similares

Solución: Beta bajo (8-12) + GMM (4-5 comp) + marcado denso.

XIII-A2. Fondos Texturados

Solución: K muy alto (70-95) fuerza coherencia regional.

XIII-A3. Bordes Finos

Solución: GrabCut iterativo + Guided Filter amplio.

XIII-A4. Escala de Grises

Solución: K alto, marcado denso, sistema como interpolador guiado.

XIII-B. Importancia del Marcado Manual

Relación empírica:

$$N_{\text{semillas}} \approx 10 \times \text{Nivel}^{1,5} \tag{14}$$

Nivel 1: 10 semillas. Nivel 5: 150 semillas.

XIV. Conclusiones

XIV-A. Efectividad del Sistema

El sistema maneja exitosamente desde casos triviales hasta extremos mediante parametrización apropiada.

XIV-B. Ventajas

Flexibilidad de ajustar K, Beta, GMM y post-procesamiento permite adaptación a cada imagen.

XIV-C. Limitaciones

- Requiere interacción humana
- Sensible a calidad de semillas
- Desafíos con solapamiento perfecto
- Escala de grises reduce efectividad

XIV-D. Trabajo Futuro

1. Deep learning para sugerencia de parámetros
2. Marcado asistido por IA
3. Integración con SAM
4. Procesamiento GPU

XV. Conclusión

XV-A. Aprendizajes del Proyecto

Graph Cuts transforma el problema de segmentación en uno de optimización en grafos, ofreciendo garantías de optimalidad global que la mayoría de métodos alternativos no proporcionan. A lo largo de este proyecto, comprobamos empíricamente que su capacidad para producir bordes precisos con mínima interacción del usuario lo hace invaluable en aplicaciones donde la calidad es crítica.

Nuestra experiencia práctica con 9 tipos distintos de imágenes —desde casos triviales como el perfil de Trump con fondo uniforme hasta casos extremadamente complejos como el pez payaso en anémona o la mamografía médica— reveló que **no existe una configuración universal de parámetros**. El éxito de la segmentación depende críticamente de:

- **Comprensión del contenido**: Identificar si los colores objeto/fondo son similares (Beta bajo) o contrastantes (Beta alto).
- **Análisis de textura**: Fondos complejos requieren K alto para ignorar ruido; fondos uniformes permiten K bajo para mayor precisión.
- **Marcado estratégico de semillas**: En casos difíciles (Simba, mamografía, pez), la cantidad y distribución de semillas es tan importante como los parámetros algorítmicos.
- **Trade-offs calidad-velocidad**: GMM + GrabCut ofrecen 20 % mejor calidad a costa de 3x tiempo de procesamiento.

El desarrollo de la interfaz parametrizable demostró ser crucial. La capacidad de ajustar K (1-200), Beta (1-50), componentes GMM (2-5), y el radio del Guided Filter en tiempo real permitió **experimentación iterativa** que aceleró nuestro entendimiento del algoritmo exponencialmente comparado con implementaciones de caja negra.

XV-B. Retos Técnicos Superados

Los desafíos más significativos no provinieron del algoritmo Graph Cuts en sí, sino de su integración con tecnologías complementarias:

1. **Dependencias inconsistentes**: El módulo `opencv-contrib-python` para Guided Filter no está disponible universalmente, requiriendo fallbacks robustos.
2. **Gestión de memoria**: Imágenes de alta resolución (¿3000px) saturaban la memoria durante construcción del grafo, motivando el auto-downsampling a 900px.
3. **Balance precision-performance**: GrabCut mejora bordes pero añade latencia considerable; implementamos como opción configurable.
4. **Casos límite**: Imágenes en escala de grises (mamografía) pierden la ventaja del espacio de color Lab, exponiendo limitaciones inherentes del método.

La implementación de la función de generación de stickers con PIL demostró la versatilidad del pipeline: la misma máscara de segmentación sirve tanto para análisis científico como para aplicaciones creativas.

XV-C. Relevancia en 2025

Aunque Graph Cuts fue publicado en 2001, este proyecto confirmó su relevancia contemporánea. En la era del aprendizaje profundo, Graph Cuts continúa siendo esencial especialmente en:

- **Creación de datasets**: Anotación rápida y precisa de imágenes para entrenar modelos supervisados.
- **Casos con datos limitados**: Donde no hay suficientes ejemplos para entrenar redes neuronales profundas.
- **Aplicaciones médicas**: Donde la explicabilidad y garantías matemáticas son requisitos regulatorios.
- **Refinamiento de predicciones**: Como post-procesamiento de salidas de CNNs para mejorar bordes.

El futuro promete la integración híbrida donde CNNs generen semillas automáticamente y Graph Cuts refine los resultados, combinando lo mejor de ambos paradigmas: la capacidad de generalización del deep learning con las garantías de optimalidad de métodos basados en grafos.

XV-D. Reflexión Final del Equipo

Este proyecto nos enseñó que **la teoría matemática elegante no garantiza resultados prácticos inmediatos**. Graph Cuts tiene una fundamentación impecable en teoría de grafos y optimización combinatoria, pero su aplicación efectiva requiere intuición sobre el dominio del problema, experimentación

sistemática con parámetros, y comprensión profunda de las limitaciones inherentes.

La experiencia de enfrentarnos a fallos iniciales —como el pez donde solo se capturaban las semillas, o Simba donde se incluía todo el piso de madera— fue invaluable. Cada fallo nos forzó a revisar nuestras suposiciones, entender mejor los términos de energía, y apreciar por qué ciertos parámetros funcionan en ciertos contextos.

El trabajo colaborativo fue esencial: debuggear código, diseñar la UI, experimentar con parámetros y documentar resultados requirieron perspectivas complementarias. **Ningún miembro del equipo podría haber completado este proyecto individualmente con la misma calidad**.

Finalmente, el proyecto reforzó una verdad fundamental en visión computacional: **no existen algoritmos perfectos universales**. Graph Cuts es extraordinario para ciertos problemas y mediocre para otros. Saber cuándo aplicarlo y cuándo buscar alternativas es tan importante como dominar su implementación.

XV-E. Conclusiones Individuales

Javier Augusto Rebull Saucedo - Arquitecto de Software & Integración

Mi contribución se centró en la arquitectura del sistema y la integración de componentes. Diseñar la separación entre `GraphMaker.py` (backend) y la UI de PyQt5 (frontend) siguiendo principios de separación de responsabilidades fue crucial para la mantenibilidad. La implementación del sistema de parámetros ajustables mediante sliders y checkboxes transformó una herramienta académica en un laboratorio interactivo de aprendizaje.

El reto más significativo fue manejar la conversión entre espacios de color (BGR $\rightarrow$ Lab $\rightarrow$ visualización RGB) sin perder información ni introducir artefactos. Aprendí que la ingeniería de software sólida —manejo de errores, validación de inputs, fallbacks robustos— es tan importante como el algoritmo subyacente para crear herramientas útiles en producción.

Juan Carlos Pérez Nava - Investigador Algorítmico & Optimización

Mi enfoque estuvo en comprender profundamente la teoría de Graph Cuts e implementar mejoras algorítmicas. La transición de un modelo Gaussiano simple a GMM (Gaussian Mixture Models) fue mi contribución principal, permitiendo segmentar objetos con distribuciones de color complejas que el modelo simple fallaba sistemáticamente (como el pez payaso).

Investigar la literatura científica —especialmente los papers de Boykov & Jolly (2001) y Rother et al. (2004) sobre GrabCut— y traducir ecuaciones matemáticas a código Python funcional fue extraordinariamente educativo. Comprendí visceralmente que β no es solo un ”parámetro de sensibilidad”sino el inverso de la varianza esperada en diferencias de color, con implicaciones directas en cómo el grafo penaliza discontinuidades.

El proyecto me enseñó que la optimización prematura es contraproducente: primero implementar correctamente, luego optimizar los cuellos de botella identificados con profiling.

Luis Gerardo Sánchez Salazar - Especialista en UX/UI & Visualización

Lideré el diseño de la interfaz de usuario y la estrategia de visualización. La decisión de mostrar 4 vistas simultáneas (semillas, objeto, fondo, histogramas) más el preview de sticker fue fundamental para el flujo de trabajo iterativo. Diseñar un sistema de dibujo interactivo que funcionara correctamente con imágenes redimensionadas —mapeando coordenadas del widget a coordenadas de la imagen original— requirió geometría cuidadosa.

La implementación del sistema de stickers con bordes personalizables y texto PIL fue particularmente gratificante, demostrando que aplicaciones técnicas pueden tener outputs creativos y divertidos. Los histogramas Lab complementando los BGR fueron mi idea para ayudar a usuarios a entender por qué trabajamos en ese espacio de color.

Aprendí que una UI bien diseñada no es solo estética, sino que reduce dramáticamente la curva de aprendizaje y hace que características complejas (GMM, GrabCut) sean accesibles mediante simples checkboxes.

Oscar Enrique García García - Ingeniero de Calidad & Experimentación

Mi rol consistió en testing sistemático y experimentación con diversos tipos de imágenes. Curé el dataset de 9 imágenes abarcando diferentes niveles de dificultad, documenté meticulosamente los parámetros óptimos para cada caso, y creé el manual de recomendaciones basado en características de la imagen.

Ejecutar cientos de segmentaciones variando K, Beta, y GMM sistemáticamente reveló patrones que no eran obvios teóricamente: por ejemplo, que Beta > 20 rara vez es útil excepto en casos con contraste extremo, o que GMM con 5 componentes usualmente overfitea excepto en casos muy específicos como el pez.

El debugging de casos fallidos —especialmente Simba capturando el piso, y la mamografía produciendo resultados inconsistentes— me enseñó debugging científico: aislar variables, documentar reproducibilidad, y entender que algunos problemas no tienen ”solución”sino solo ”mitigaciones mediante marcado más cuidadoso”.

XVI. Agradecimientos

Los autores expresamos nuestra gratitud a nuestros profesores del curso de Visión Computacional en la Maestría en Inteligencia Artificial Aplicada del Tecnológico de Monterrey: Dr. Gilberto Ochoa Ruiz, MIP Ma. del Refugio Melendez Alfaro, y M. en C. Jose Angel Martinez Navarro, por su guía experta y dedicación durante el trimestre de Septiembre 2025.

REFERENCIAS

[1] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proceedings Eighth IEEE International Conference on Computer Vision (ICCV)*, vol. 1, pp. 105–112, 2001.

[2] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[3] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[4] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[5] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, pp. 281–297, 1967.

[7] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[8] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[9] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[11] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

[12] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

Equipo 13

Maestría en Inteligencia Artificial Aplicada

Tecnológico de Monterrey

16 de Noviembre de 2025