Author copy. Cite this: J. Olivares, O. Zachariadis, N. Satpute and J. Gómez-Luna, "Optimum Vessel Segmentation," 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), Madrid, Spain, 2022, pp. 1-6, doi: 10.23919/CISTI54924.2022.9820200.

Segmentación Óptima de Vasos Sanguíneos.

Optimum Vessel Segmentation

Joaquín Olivares, Orestis Zachariadis
Universidad de Córdoba
España
olivares@uco.es

Nitin Satpute
Technology Innovation Institute
United Arab Emirates
nitin.satpute@tii.ae

Juan Gómez-Luna
ETH Zürich
Switzerland
juang@inf.ethz.ch

Resumen — **Una buena segmentación de vasos sanguíneos en imagen médica es crítica durante la cirugía. Proponemos un algoritmo para el cómputo paralelo del crecimiento de regiones (pSRG) basado en el cálculo del gradiente aplicando persistencia y bucles de cuadrícula (grid-stride loops - GSL). Nuestra propuesta pSRG para la segmentación de vasos 2D es más preciso y tienen un elevado valor de *dice score*. Siendo 1,9 veces más rápido que la implementación de referencia gracias a evitar transferencias innecesarias de memoria.**

Palabras Clave – segmentación; imagen médica; computación de altas prestaciones, GPU.

Abstract — **Good blood vessel segmentation in medical imaging is critical during surgery. We propose a parallelized region growth algorithm (pSRG) computing the gradient using Persistence and grid-stride loops. This approach avoid unnecessary memory transfers. As a result, in addition to faster computation, we obtain more accurate segmentation.**

Keywords – segmentation, medical imaging, high performance computing, GPU.

I. Introducción

En las imágenes médicas, la segmentación de vasos del hígado es una de las tareas más desafiantes durante cirugía hepática mediante laparoscopia [36]. Lo cual es extensible a cualquier cirugía sobre órganos blandos. Es por ello que la calidad de la segmentación es crucial. Por otra parte, las imágenes tomadas por tomografía computerizada (TC) o resonancia magnética (MR) de forma previa a la intervención no son útiles, ya que los tejidos se deforman al insuflar aire en el abdomen del paciente [37][38]. Por ello, es necesario tomar imágenes durante la operación, y procesar las mismas con rapidez. La computación de altas prestaciones permite incrementar la resolución de los resultados así como agilizar el proceso de cómputo [1],[2].

La computación de altas prestaciones permite realizar operaciones aritméticas complejas, las cuales destacan por requerir el procesamiento de un elevado volumen de datos [5][28]. La computación de altas prestaciones depende del tipo del dispositivo, siendo las tecnologías más comunes las GPU [9][34], y las FPGA. Éstas últimas permiten además operar con aritméticas y álgebras no convencionales [12], e incluso reconfigurar en caliente el dispositivo [13]. Aunque también se contempla como tal la optimización de dispositivos de baja capacidad de cómputo, a nivel de procesador multinúcleo [15], los cuales permiten hasta el procesamiento de pesadas redes neuronales profundas [20][23]. Los problemas a los que se pueden aplicar son muy variados, predominando el procesamiento de imágenes y vídeo, para cualquier tipo de entorno.

A. Crecimiento de una región sembrada (SRG)

El crecimiento de una región sembrada (*Seeded Region Growing* - SRG) es un enfoque ampliamente utilizado para la segmentación de vasos semiautomática. Delibasis *et al.* [3] propusieron una herramienta basada en una versión del algoritmo SRG, combinada con el conocimiento a priori de la forma requerida. SRG comienza con un conjunto de píxeles llamados semillas y crece una región uniforme y conectada de cada semilla. Los pasos clave para SRG son definir una o varias semillas y un criterio de clasificación que se basa en las propiedades de la imagen y la interacción del usuario [4]. SRG comienza a partir de una semilla y encuentra los puntos vecinos similares en función de criterios de umbral utilizado. La región crece si se cumplen dichos criterios. Los vecinos similares son nuevas semillas para la próxima iteración. Este proceso se repite hasta que la región no se pueda extender más. En la práctica, exige un alto costo computacional por la gran cantidad de datos dependientes que se procesan en SRG.

SRG es un proceso iterativo que se invoca de forma continua hasta que la región ya no puede crecer más. Cuando se implementa en GPU, requiere la terminación del kernel (es decir, la función ejecutada en la GPU) y el relanzamiento desde la CPU (*Kernel Termination and Relaunch* - KTRL) y transferencias de datos entre la CPU y la GPU [1],[4]. Nuestro principal objetivo es reducir estas transferencias de datos utilizando diferentes métodos de sincronización de GPU entre bloques (*inter-block GPU synchronization* - IBS), esperando como resultado una implementación paralela eficiente de SRG. IBS brinda flexibilidad para llevar a cabo todos los cálculos en la GPU al brindar visibilidad a los datos intermedios actualizados sin ninguna intervención de la CPU.

En este artículo, proponemos un enfoque de GPU persistente, de bucle de cuadrícula (*grid-stride loop* - GSL) y basado en IBS, para evitar transferencias de memoria intermedias entre CPU y GPU. Además, nuestra implementación reduce el procesamiento sobre vóxeles (píxeles de una imagen tridimensional) innecesarios, proporcionando una aceleración significativa. El enfoque de subprocesos persistentes (*Persistent thread block* - PT) depende básicamente del número de bloques de subprocesos activos [6],[7]. Se propone calcular en paralelo varios SRG utilizando memoria

compartida (*shared memory*, en terminología de NVIDIA) en GPU, evitando transferencias intermedias entre la CPU y la GPU. Esto se inspira en el procesamiento paralelo en la región de interés estática (RoI) de mosaicos en la GPU. Comparemos la propuesta con una implementación KTRL para verificar la precisión de la segmentación de los vasos. Por lo tanto, las principales contribuciones de este artículo son mejorar el rendimiento de SRG al reducir los cálculos no deseados y evitar las transferencias de memoria intermedias entre la CPU y la GPU.

El resto del documento está estructurado de la siguiente manera. La Sección II resume los trabajos relevantes y el estado del arte con respecto a SRG. La Sección III explica los enfoques de GPU (KTRL y estático) para la implementación de SRG utilizando persistencia y GSL. La paralelización de SRG aplicada a la segmentación de vasos se analiza en la Sección IV. Los resultados de rendimiento y la comparación de SRG paralelos persistentes y GSL para la segmentación de vasos se presentan en la Sección V. Finalmente, la Sección VI concluye resumiendo las principales conclusiones del trabajo.

II. Estado del arte

Recientemente, se han realizado muchos trabajos sobre segmentación de imágenes que se basan en el modelo basado en *Snakes* [8], el flujo vectorial de gradiente (GVF) [10], y el modelo Chan-Vese basado en conjuntos de niveles [11].

Las Snakes se definen como un conjunto de puntos alrededor de un contorno [8]. Este método es inadecuado por los bordes lejanos y los muchos pequeños ruidos en la imagen [8]. El flujo vectorial de gradiente (GVF) es el campo vectorial que se produce mediante un proceso que suaviza y difunde un campo vectorial de entrada. Por lo general, se usa para crear un campo vectorial a partir de imágenes que apunta a los bordes del objeto desde una distancia. Es ampliamente utilizado en aplicaciones de análisis de imágenes y visión por computadora para seguimiento de objetos, reconocimiento de formas, segmentación y detección de bordes. En particular, se usa comúnmente junto con el modelo de contorno activo [10]. El costo de GVF es elevado al realizar un seguimiento del número y la distribución de puntos. Por otra parte, el modelo Chan-Vese se basa en conjuntos de niveles para la segmentación de imágenes [11]. En ausencia de bordes fuertes, el modelo de Chan-Vese propone una formulación basada en regiones para la segmentación de imágenes. El modelo Chan-Vese para contornos activos es un método potente y flexible que puede segmentar muchos tipos de imágenes. Pero entre todos, SRG es el algoritmo más simple y juega un papel esencial en la segmentación de imágenes médicas [1].

Smistad *et al.* [14] paralelizaron la SRG en segmentación de imágenes. El conjunto de datos de imágenes médicas se recorta antes del procesamiento. Después, la CPU asigna la memoria equivalente al tamaño recortado para copiar los datos a la imagen recortada en la GPU, y se calcula en paralelo la SRG. Es, por tanto, un enfoque de hilos no persistentes para la segmentación de vasos basada en SRG. Utilizaron una cola dinámica para SRG y cuando era necesario cambiar el número de subprocesos (debido a la expansión del borde de la región) implicaba reiniciar el kernel, lo que implica leer todos los valores de la memoria global nuevamente. Zhang *et al.* [16] implementó el crecimiento de región bidireccional utilizando una cola dinámica (pila). Jiang *et al.* [17] propuso un algoritmo mejorado de segmentación de vasos de crecimiento de región basado en ramificaciones utilizando una pila.

La implementación de SRG basada en GPU necesita una cola dinámica (pila). Las CPU proporcionan soporte de hardware para pilas, pero las GPU no [7]. Las tareas inherentes al sistema de colas son tediosas, la implementación en GPU de una pila requiere cambios continuos en las asignaciones de memoria, lo que a su vez requiere la invocación iterativa del kernel de GPU desde la CPU, en otras palabras, la terminación del kernel y el relanzamiento, como se explica en [18]. TREES (Task-parallel run-time system) [19], es un mecanismo dedicado a la invocación de núcleos de GPU desde la CPU de forma iterativa para actualizar la pila de máscaras de tarea.

Chen *et al.* [7] implementó en GPU una alternativa del sistema de colas (pila) que utiliza el lanzamiento dinámico del kernel, basada en lanzamientos libres mediante la técnica de reutilización de subprocesos. Esta técnica no requiere extensiones de hardware, y es directamente implementable en las GPU existentes. Al convertir el lanzamiento de subkernel en una función de programación independiente del soporte de hardware, el lanzamiento libre proporciona un enfoque alternativo para el lanzamiento de subkernel que se puede utilizar de manera beneficiosa en las GPU.

III. Paralelización de SRG

Se podría considerar a la GPU como una cuadrícula de bloques de hebras, también llamadas hilos, o subprocesos (*threads*). Una hebra es la unidad computacional más pequeña mapeada en los núcleos, y el bloque de hilos se mapea en los multiprocesadores (*streaming multiprocessors* - SM). Cada SM puede ser ocupado por más de un bloque. Las hebras de bloques independientes pueden acceder a los datos a través de la memoria compartida en el SM [21]. Para comunicar datos válidos entre los bloques, estos bloques persistentes deben sincronizarse utilizando IBS a través de la memoria del dispositivo. La persistencia implica un número máximo de bloques de subprocesos que pueden estar activos en el momento del cálculo, dependiendo de los recursos de GPU disponibles [6], [22].

En este trabajo, usamos enfoques basados en PT y memoria compartida para implementar el SRG. La memoria compartida y el SRG basado en bucle de cuadrícula (*Grid-stride Loop* – GSL) reducen las transferencias de memoria y los cálculos totales. Para cada subproceso realizado en paralelo en GPU, SRG comienza desde el subproceso semilla y encuentra vecinos similares que lo rodean, que se inician como nuevas semillas. El proceso de SRG se repite hasta que no se puedan encontrar nuevos vecinos similares. Normalmente, SRG se puede implementar en la GPU como una llamada al kernel recursiva o iterativa (KTRL) como se muestra en la Fig. 1. La llamada al kernel implica la invocación de una GSL. Los bloques se ejecutan en SM, y los subprocesos se ejecutan en núcleos.

SRG puede ser un proceso recursivo o iterativo. Las llamadas recursivas del kernel no pueden utilizar núcleos de GPU de manera eficiente debido a limitaciones de hardware [26]. KTRL penaliza el cómputo, ya que los elementos

innecesarios de la imagen forman parte de los cálculos y transferencias de memoria intermedias entre la CPU y la GPU.

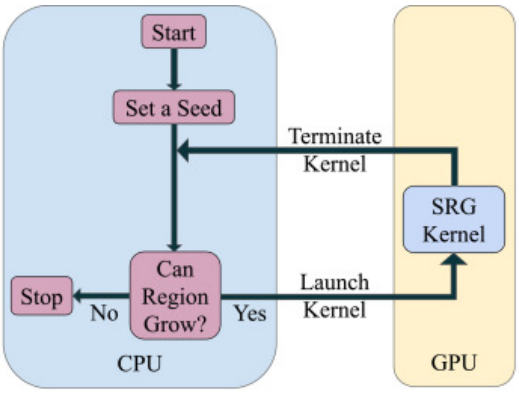


Figura 1. Implementación de SRG basado en KTRL.

Para evitar estos problemas, proponemos un enfoque diferente. SRG comienza desde la semilla, sin embargo, el control lo realizará la GPU. El kernel SRG se inicia si la región no se ha desarrollado por completo. Se necesita IBS para transferir datos válidos entre los bloques de subprocesos activos. El número de bloques de subprocesos activos (o bloques persistentes) en los SM está limitado debido a limitaciones de recursos [6], [21], [22]. El bucle GSL termina cuando la región no puede crecer más y el control regresa a la CPU o *host* como se muestra en la Fig. 2. Nuestra propuesta destaca al explotar el paralelismo utilizando persistencia e IBS como se detalla a continuación en los enfoques estático y dinámico.

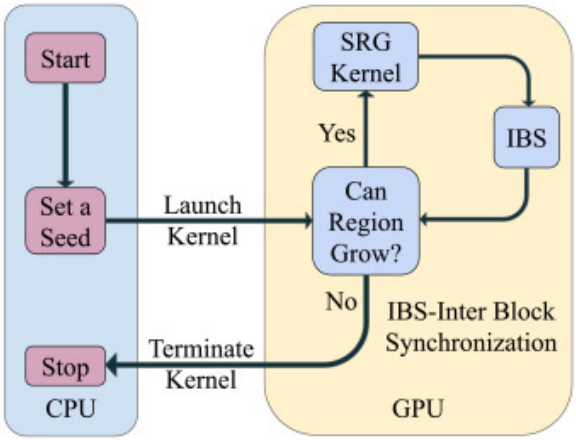


Figura 2. Propuesta de implementación SGL.

A. Enfoque estático

En el enfoque propuesto, aplicamos SGL sobre la región de interés (RoI estático) utilizando persistencia e IBS [6], [22]. La RoI corresponde a la imagen completa del hígado, la cual se mapea en la GPU como una GSL como se muestra en la Fig. 3b. La CPU invoca el kernel SRG en la GPU. Los bloques persistentes iteran a través de la imagen completa y la región de crecimiento desde la semilla en todas y cada una de las iteraciones de bloques persistentes sobre las teselas del mosaico de la imagen procesado en la GPU, como se muestra en la Fig. 3b. Los pasos de SRG en las Figs. 3c – f muestran la región del hígado en crecimiento. El kernel de SRG termina cuando la región crece por completo. Para cada hilo paralelizado en el bloque, si se encuentra semilla y no es el elemento límite del bloque, se calculan los elementos vecinos similares. La región se cultiva creando elementos vecinos similares como semillas nuevas.

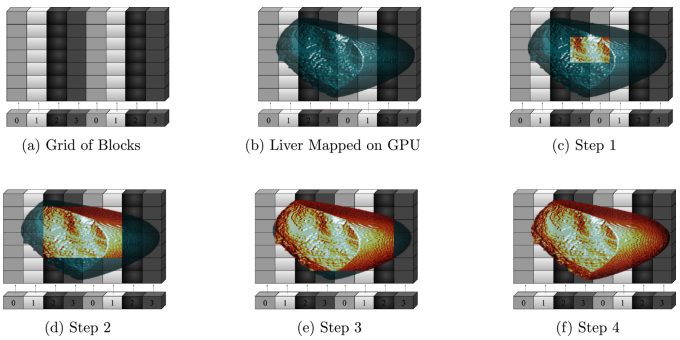


Figura 3. SRG usando persistencia y GSL.

En la Fig. 3 se muestran cuatro bloques persistentes (0, 1, 2, 3, con distintos tonos de gris). Los mosaicos con el mismo color se iteran por el mismo bloque persistente. En KTRL, estos mosaicos son procesados por los TB (*Thread block*s, bloques de hilos) de forma aleatoria. Se aplica un GSL mediante bloques persistentes sobre las teselas de la imagen completa del hígado (Paso 1 en la Fig. 3c). La región crece alrededor de la semilla que contiene elementos similares. IBS se aplica para comunicar datos válidos entre los bloques para el siguiente paso de SRG como se muestra en la Fig. 2b.

Los bloques persistentes iteran sobre la imagen del hígado y la región se vuelve a hacer crecer en el paso 2 como se muestra en la Fig. 3d. Se aplica IBS y se comunican datos válidos entre los bloques para que la región pueda crecer más como se muestra en las Fig. 3e y 3f. Después del paso 4 en la Fig. 3f, SRG se detiene porque la región no puede crecer más. Una vez se realiza el procesamiento de la imagen completa, todos los bloques persistentes se sincronizan globalmente a través de la barrera "Inter_Block_GPU_Sync ()" [27].

La diferencia entre el enfoque estático y dinámico de Satpute *et al.* [1] se puede explicar en términos de RoI estático y dinámico en las teselas de los mosaicos. En el enfoque estático, RoI permanece constante y SRG ocurre dentro del RoI constante hasta que la región no puede crecer más. Mientras que en el enfoque dinámico, SRG comienza dentro del RoI inicial. RoI aumenta e incluye más elementos de manera uniforme en todas las direcciones para el siguiente paso de SRG. SRG se lleva a cabo, el ROI aumenta y la región crece aún más. Por lo tanto, RoI cambia en cada paso de SRG hasta que la región no puede crecer más en un enfoque dinámico. En la siguiente sección, presentamos la segmentación de vasos 2D como una aplicación para SRG estático basado en RoI.

IV. Aplicación a la Segmentación de Vasos 2D

El algoritmo de segmentación 2D está inspirado en el algoritmo SRG basado en gradientes desarrollado por Rai y Nair [29]. Se basa en dos módulos importantes, el gradiente de la imagen, y SRG para la segmentación bidimensional paralela rápida de vasos a partir de imágenes hepáticas por tomografía computerizada (TC).

A. *Cálculo paralelo del gradiente de imagen*

Rai y Nair [29] presentaron una selección de criterios de homogeneidad y su impacto en la calidad de la segmentación utilizando SRG. En general, los criterios de umbral incluyen el contraste del objeto, el límite de la región, la homogeneidad de la región, los valores de intensidad y las características de la textura como la forma y el color. Además, incluimos funciones de costo basadas en valores de intensidad, y su dirección y magnitud de gradiente.

La función de costo explota ciertas características de la imagen alrededor de la semilla. La función de costo basada en gradiente requiere como parámetros la mayor magnitud de gradiente (max_g) y el gradiente mínimo (min_g) presente en la imagen. Las funciones de costo son:

$$(1)\ \text{Cost1} = g/(k*\text{max\_g}) \qquad 0 < \text{Cost1} < 1$$

$$(2)\ \text{Cost2} = (\text{max\_g} - g)/(\text{max\_g} - \text{min\_g}) \qquad 0 < \text{Cost2} < 1$$

Donde g es la magnitud del gradiente del píxel en consideración, y k es el parámetro constante que controla el crecimiento de la región. El píxel en consideración se agrega en la región de crecimiento si coincide con los elementos semilla, es decir, las funciones de costo especificadas por las Ecs. (1) y (2) se cumplen; de lo contrario, se excluye de la consideración.

B. *Segmentación paralelizada de los vasos sanguíneos*

Proponemos paralelizar la segmentación de vasos como se muestra en la Fig. 4. El usuario selecciona una o varias semillas, que junto con la imagen se transfieren a la GPU. El kernel del dispositivo calcula el gradiente de la imagen en paralelo como se discutió en la sección anterior aplicando IBS.

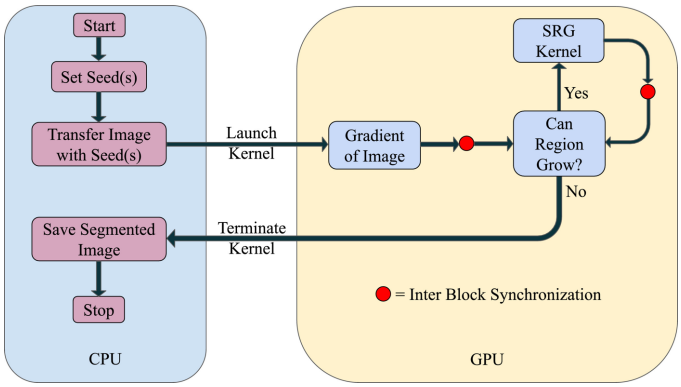


Figura 4. Cómputo paralelo de la Segmentación de vasos.

Aplicamos el algoritmo SRG utilizando como funciones de costo las basadas en gradiente mostradas en las Ecs. (1) y (2). Para cada píxel en paralelo, el píxel en consideración invoca al núcleo SRG cuando se satisfacen las funciones de costo. Las semillas se actualizan después de aplicar IBS, y las funciones de costo basadas en gradiente se verifican nuevamente para nuevos píxeles. Este proceso continúa hasta que no se forman nuevas semillas, es decir, no se agregan nuevos píxeles a la región de crecimiento.

El kernel finaliza y el control vuelve a la CPU cuando la región crece por completo. La imagen segmentada se transfiere a la CPU, y el proceso de segmentación se detiene. Esta implementación de GPU evita la llamada iterativa del kernel SRG desde la CPU.

V. Evaluación Experimental

El equipo utilizado es un Intel (R) Core (TM) i7-7700HQ CPU @ 2.80GHz RAM 24 GB, NVIDIA GPU 1050 (RAM 4GB), mientras el entorno software consistió en OpenCL 1.2 (ref. [30]) y CUDA Toolkit 10.1.

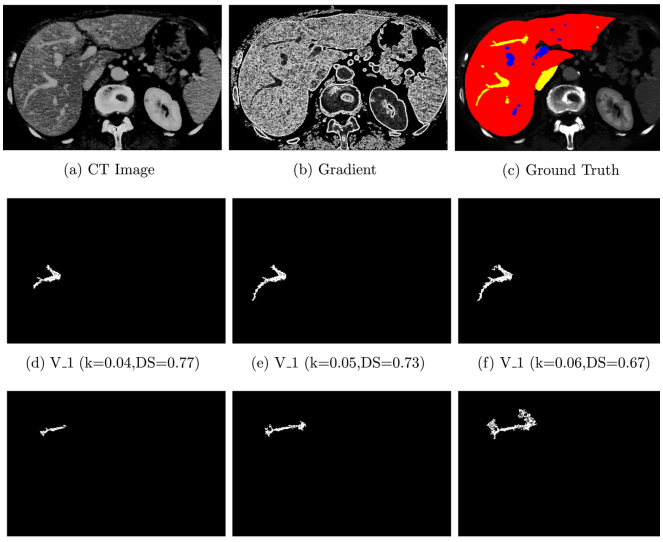
A. *Segmentación paralelizada de los vasos sanguíneos*

Los datos del hígado para el trabajo de investigación se han obtenido del Centro de Intervención de la Universidad de Oslo, Noruega [31]. El personal médico proporcionó los datos de certeza (*ground truths* - *GT*) para la segmentación de vasos. Cabe destacar que la valoración de la información visual es siempre subjetiva, y depende en gran medida de la experiencia de los expertos [24]. Por ello, cuanto más precisa sea la segmentación, tanto mejor será dicha valoración y repercutirá en la cirugía laparoscópica [25]. Las imágenes proceden de tomografía computarizada (TC). Para calcular el GT, las imágenes se preprocesan mediante una aplicación desarrollada localmente con 3D Slicer para mejorar los vasos [32]. La Tabla 1 muestra información sobre las imágenes utilizadas, incluyendo el número total de rebanadas (*slices*), especificando cuales incluyen vasos utilizados para la experimentación de un volumen particular (*Vessel slices*).

TABLA I. Conjunto de datos para experimentar

#Vol	#Slices	Vessel Slices
10504	59	7
18152	139	5
23186	87	6
28059	59	6

Las variaciones en la segmentación de vasos con el parámetro k aplicando nuestra implementación paralela SRG (pSRG) se muestran en la Fig. 5. La entrada al algoritmo es un corte (*slice*) de hígado de TC (Fig. 5a). El gradiente de la imagen de TC de entrada y la verificación (*ground truth*) para la segmentación se muestran respectivamente en las Fig. 5b y 5c. Para evaluar la calidad de la segmentación de vasos, se utilizan el Coeficiente de similitud de dados (*Dice similarity coefficient* - DS), y la Precisión [33]. El DS mide la similitud entre el *Ground truth* y la salida segmentada. Si son idénticos (es decir, contienen los mismos elementos), el coeficiente es 1.0, disminuyendo en función del número de elementos comunes hasta un valor de 0.0 cuando no hay ningún elemento en común. La Precisión describe el número de detecciones positivas con respecto al *Ground truth*. De todos los elementos que están segmentados en una imagen dada, el número de estos elementos que realmente tenían una anotación de *Ground truth* coincidente se puede llamar precisión.



(a) CT Image (b) Gradient (c) Ground Truth

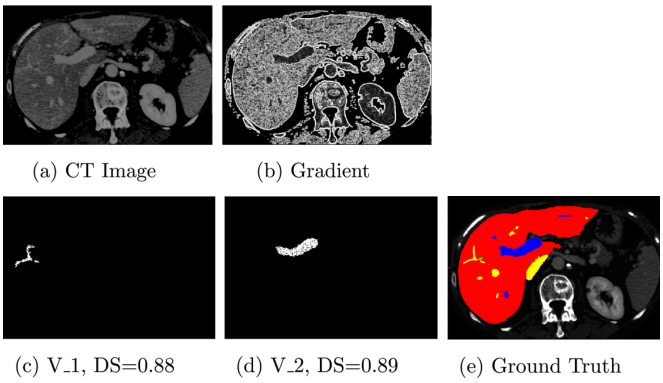
(d) V_1 (k=0.04,DS=0.77) (e) V_1 (k=0.05,DS=0.73) (f) V_1 (k=0.06,DS=0.67)

(g) V_2 (k=0.04,DS=0.54) (h) V_2 (k=0.05,DS=0.60) (i) V_2 (k=0.06,DS=0.37)

Figura 5. Variaciones en la segmentación de vasos en el primer *slice*.

Mostramos dos vasos segmentados para diferentes valores del parámetro k: 0.04, 0.05 y 0.06. Correspondiéndose el primero con las Fig. 5d, 5e, y 5f. Así, para una precisión de 0.04 se obtiene una puntuación alta en DS (0,77). Mostramos la segmentación del segundo vaso del mismo corte en las Fig. 5g, 5h, y 5i. La segmentación más precisa se obtiene para k=0.05 obteniendo un coeficiente de similitud de 0,60.

Mostramos la calidad de la segmentación en otro corte de TC (Fig. 6a), su gradiente (Fig. 6b), la segmentación de dos vasos (Fig. 6c y 6d), y el *Ground truth* (Fig. 6e). Analizamos que los vasos se segmentan con mayor precisión debido a un mejor valor del coeficiente de similitud de los dados cuando el parámetro k toma el valor 0.05.



(a) CT Image (b) Gradient

(c) V_1, DS=0.88 (d) V_2, DS=0.89 (e) Ground Truth

Figura 6. Segmentación del segundo corte de hígado.

La aceleración promedio para la segmentación de vasos mediante nuestra implementación pSRG es 1,9× más rápida que la implementación de KTRL. Se considera la aceleración de la segmentación del vaso para el caso de mayor precisión y mejor valor de puntuación de DS.

B. Comparativa con otros trabajos

Otros trabajos sobre segmentación de imágenes se basan en *snakes* [8] y Chan-Vese [11]. Validamos el rendimiento analizando 24 cortes obtenidos de 4 hígados diferentes, los cuales contienen 72 vasos. La Tabla II muestra la comparación de los resultados promedio de DS y Precisión obtenidos.

TABLA II. COMPARATIVA EN TÉRMINOS DE DS Y PRECISIÓN

#Vol	#Vessel Slices	#Total Vessels	Chan-Vese [11]		Snake [8]		pSRG	
			DS	Prec	DS	Prec	DS	Prec
10504	7	21	0.78	0.82	0.77	0.81	0.85	0.83
18152	5	15	0.75	0.85	0.78	0.83	0.82	0.87
23186	6	18	0.77	0.83	0.81	0.82	0.84	0.86
28059	6	18	0.72	0.81	0.76	0.78	0.81	0.84

Nuestra propuesta pSRG para la segmentación de vasos no solo es más rápida sino también más precisa. La precisión de la segmentación depende del parámetro ' k '. Los médicos tienen el conocimiento para decidir qué segmentación es más precisa. Permitir ajustar el valor de k es una ventaja que ofrece a los médicos flexibilidad para ajustar la precisión de la segmentación del vaso. De hecho, a veces resulta difícil encontrar un mismo valor de k idóneo para diferentes vasos en un mismo corte de TC. El rango de k puede variar entre 0,03 y 0,12 para una mejor precisión de la segmentación de vasos.

VI. CONCLUSIÓN

En este artículo, proponemos la implementación paralela de SRG (pSRG) basada en persistencia y GSL. Para obtener una aceleración significativa, debemos aprovechar el paralelismo mediante el uso de persistencia e IBS. Es deseable obtener algoritmos que requieran la menor sincronización posible. Los algoritmos más rápidos en GPU son los que encajan perfectamente en el modelo de programación de GPU, donde los bloques son independientes entre sí y no requieren sincronización [35].

El problema surge cuando no se puede evitar la llamada iterativa del kernel. Incurre en transferencias de memoria de la CPU a la GPU cuando se utiliza KTRL para sincronizaciones globales. Terminar un kernel y reiniciarlo de nuevo incurre en transferencias de datos de la CPU a la GPU y viceversa. Consumiendo mucho tiempo de cómputo.

Utilizando IBS junto con la persistencia podemos mapear el algoritmo completo en la GPU manteniendo la sincronización. El control regresa a la CPU solo cuando la tarea del kernel ha terminado. La CPU lanza un kernel en la GPU, la GPU lo ejecuta y los resultados finales se copian en la CPU, evitando toda comunicación de datos intermedia.

Nuestra propuesta pSRG para la segmentación de vasos 2D es preciso y tienen un elevado valor de DS. Es además 1,9 veces más rápida que la implementación KTRL.

AGRADECIMIENTOS

El trabajo cuenta con el apoyo del proyecto Navegación de tejidos blandos de alto rendimiento (HiPerNav). Este proyecto ha recibido financiación del programa de investigación e innovación Horizonte 2020 de la Unión Europea en virtud del acuerdo de subvención nº. 722068. Agradecemos al Centro de Intervención, Hospital Universitario de Oslo, (Noruega) el proporcionar las imágenes de TC y la información *Ground truth* para la validación clínica de la segmentación de vasos.

REFERENCIAS BIBLIOGRÁFICA

[1] N. Satpute, et al. "GPU acceleration of liver enhancement for tumor segmentation" Comput. Methods Programs Biomed., 184 (2020), p. 105285

[2] R. Palomar, et al. "High-Performance Computation of Bézier Surfaces on Parallel and Heterogeneous Platforms," International Journal of Parallel Programming, vol. 46, pp. (2018) 1035–1062.

[3] K.K. Delibasis, et al "A novel tool for segmenting 3d medical images based on generalized cylinders and active surfaces" Comput. Methods Programs Biomed., 111 (1) (2013), pp. 148-165

[4] E. Smistad, et al. "Medical image segmentation on GPUs–a comprehensive review" Med. Image Anal., 20 (1) (2015), pp. 1-18

[5] O. Zachariadis, et al. "Accelerating Sparse Matrix-Matrix Multiplication with GPU Tensor Cores". Computers Methods and Programs in Biomedicine, 193, (2020) 105431.

[6] K. Gupta, J.A. Stuart, J.D. Owens "A study of persistent threads style GPU programming for GPGPU workloads" Innovative Parallel Computing-Foundations & Applications of GPU, Manycore, and Heterogeneous Systems (INPAR 2012), IEEE (2012), pp. 1-14

[7] G. Chen, X. Shen "Free launch: optimizing GPU dynamic kernel launches through thread reuse" Proceedings of the 48th International Symposium on Microarchitecture, ACM (2015), pp. 407-419

[8] S. Roy, S. Mukhopadhyay, M.K. Mishra "Enhancement of morphological snake based segmentation by imparting image attachment through scale-space continuity" Pattern Recognit., 48 (7) (2015), pp. 2254-2268

[9] R Palomar, et al. "Parallelizing and Optimizing LIP-Canny Using NVIDIA CUDA" Int. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (2010), pp. 389-398

[10] E. Smistad, A.C. Elster, F. Lindseth "Real-time gradient vector flow on GPUs using OpenCL" J. Real-Time Image Process., 10 (1) (2015), pp. 67-74

[11] N Satpute, et al. "Accelerating Chan–Vese model with cross-modality guided contrast enhancement for liver segmentation," Computers in Biology and Medicine, vol. 124, (2020) 103930.

[12] J. Olivares "A low cost architecture for variable block size motion estimation" Journal of Signal Processing Systems 68 (1), (2012) pp.127-138.

[13] L. Medina, et al. "A comparison of FPGA and GPGPU designs for Bayesian occupancy filters". Sensors vol. 17 (11), 2599 (2017).

[14] E. Smistad, A.C. Elster, F. Lindseth "GPU accelerated segmentation and centerline extraction of tubular structures from medical images" Int. J. Comput. Assisted Radiol. Surg., 9 (4) (2014), pp. 561-575

[15] FJ. Rodriguez‐Lozano, et al. "Non-Invasive Forehead Segmentation in Thermographic Imaging". Sensors vol. 19, 4096 (2019).

[16] X. Zhang, X. Li, Y. Feng "A medical image segmentation algorithm based on bi-directional region growing" Optik, 126 (20) (2015), pp. 2398-2404

[17] H. Jiang, et al. "A region growing vessel segmentation algorithm based on spectrum information" Comput. Math. Methods Med., 2013 (2013)

[18] T.C. Pessoa, et al. "A GPU-based backtracking algorithm for permutation combinatorial problems" Algorithms and Architectures for Parallel Processing, Springer International Publishing, Cham (2016), pp. 310-324

[19] B.A. Hechtman, A.D. Hilton, D.J. Sorin, "TREES: a CPU/GPU task-parallel runtime with explicit epoch synchronization, arXiv:1608.00571. (2016)."

[20] FJ. Rodriguez‐Lozano, et al. "Benefits of ensemble models in road pavement cracking classification", Computer-Aided Civil and Infrastructure Engineering. (2020) 1-25.

[21] V. Vineet, P.J. Narayanan "CUDA cuts: Fast graph cuts on the GPU" 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (2008), pp. 1-8,

[22] S. Xiao, W.C. Feng "Inter-block GPU communication via fast barrier synchronization" 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS) (2010), pp. 1-12,

[23] G Almeida, AR Figueira, J Lencart, JMRS Tavares, "Segmentation of male pelvic organs on computed tomography with a deep neural network fine-tuned by a level-set method", Computers in Biology and Medicine 140, (2022) 105107.

[24] A Beghdadi, et al. "A critical analysis on perceptual contrast and its use in visual information analysis and processing" IEEE Access 8, (2020) pp-156929-156953

[25] A Teatini, et al. "Influence of sampling accuracy on augmented reality for laparoscopic image-guided surgery" Minimally invasive therapy & allied technologies, (2020) pp. 1-10

[26] X. Tang, et al. "Controlled kernel launch for dynamic parallelism in GPUs" 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA) (2017), pp. 649-660,

[27] N Satpute, et al. "Fast parallel vessel segmentation," Computer Methods and Programs in Biomedicine, vol. 192, (2020) 105430.

[28] P San Juan, et al. "Low precision matrix multiplication for efficient deep learning in NVIDIA Carmel processors" The Journal of Supercomputing, (2021) pp. 1-13

[29] G. Rai, T. Nair, "Gradient based seeded region grow method for ct angiographic image segmentation", arXiv:1001.3735. (2010).

[30] J.E. Stone, D. Gohara, G. Shi "OpenCL: a parallel programming standard for heterogeneous computing systems" IEEE Des. Test, 12 (3) (2010), pp. 66-73, 10.1109/MCSE.2010.69

[31] Å.A. Fretland, et al. "Laparoscopic versus open resection for colorectal liver metastases" Ann. Surg., 267 (2) (2018), pp. 199-207

[32] R. Naseem, et al. "Cross-modality guided contrast enhancement for improved liver tumor image segmentation" IEEE Access vol. 9, (2021) pp. 118154 – 118167.

[33] Y. Zhao, et al "Knowledge-aided convolutional neural network for small organ segmentation" IEEE J. Biomed. Health Inf., 23 (4) (2019), pp. 1363-1373, 10.1109/JBHI.2019.2891526

[34] H martinez, et al "A framework for genomic sequencing on clusters of multicore and manycore processors" The International Journal of High Performance Computing Applications. Vol. 32 n. 3 (2018) pp. 393-406

[35] O. Zachariadis, et al "Accelerating B-spline Interpolation on GPUs: Applica-tion to Medical Image Registration" Computer Methods and Programs in Biomedicine, vol. 193, 105431, 2020.

[36] A Benedetti, et al. "Risk Factors of Positive Resection Margin in Laparoscopic and Open Liver Surgery for Colorectal Liver Metastases: A New Perspective in the Perioperative Assessment : A European Multicenter Study" Annals of Surgery, Volume 275, Number 1, 25 (2022), pp. e213-e221(9)

[37] A. Teatini, et al. "Use of stereo-laparoscopic liver surface reconstruction to compensate for pneumoperitoneum deformation through biomechanical modeling". VPH2020-Virtual Physiological Human. (2020).

[38] H. Luo, et al. "Unsupervised learning of depth estimation from imperfect rectified stereo laparoscopic images", Computers in Biology and Medicine, (2021) 105109.