Optimización en la Aproximación de Curvas NURBS con Algoritmos Genéticos Paralelos en un GPU

Leopoldo Noel Gaxiola Sánchez[a], Juan José Tapia Armenta[a], Julio César Rolón Garrido[a]

[a]*Centro de Investigación y Desarrollo de Tecnología Digital del Instituto Politécnico Nacional*
Av. del Parque 1310, Mesa de Otay, Tijuana, B.C., CP 22510
http://fiad.ens.uabc.mx/
[b]*CICESE, Laboratorio de Control, Departamento de Electrónica y Telecomunicaciones.*
Carr. Ensenada-Tijuana 3918, Zona Playitas, Ensenada, B.C., México, 22860.
lgaxiolas1100@alumno.ipn.mx, jtapiaa@ipn.mx, jcrolon@ipn.mx

2013 Published by $DIFU_{100}ci@$ http://www2.uaz.edu.mx/web/www/publicaciones
Selection and peer-review under responsibility of the Organizing Committee of the CICOMP-2013, www.cicomp.org

Resumen

En este trabajo se desarrollan dos versiones de algoritmos genéticos paralelos, un algoritmo genético paralelo simple y un algoritmo genético con modelo de islas; los dos algoritmos se implementaron en un GPU. El algoritmo genético paralelo basado en el modelo de islas que se propone en este trabajo tiene la característica principal de que no se hace migración entre las islas, en su lugar se crea una isla élite con los mejores individuos de cada una de las islas para compartir los mejores individuos. El algoritmo se aplica a un problema de aproximación de curvas NURBS a un conjunto de puntos de una imagen médica, con este algoritmo, además de reducir el tiempo de ejecución se obtiene soluciones más cercanas al óptimo que con los resultados obtenidos con el algoritmo genético paralelo simple.

Palabras clave: Algoritmos genéticos paralelos, Curvas NURBS, GPU.

1. Introducción

Los algoritmos genéticos (AGs) son técnicas de búsqueda y optimización inspirados en el proceso de la evolución natural de las especies [1]. En este caso, un individuo es una solución potencial del problema, y su representación se define de acuerdo al problema a resolver. Al igual que ocurre en la naturaleza, los AGs se basan en la supervivencia de los individuos más aptos de una población. El AG inicia con un conjunto de individuos que constituye una población que evoluciona de una generación a otra, a través de la creación de nuevos individuos con mejor función de aptitud y la eliminación en la población de individuos con baja función de aptitud.

En los AGs las poblaciones de individuos evolucionan mediante la aplicación de operadores genéticos, tales como selección, cruce y mutación, cuya funcionalidad e implementación depende del problema a resolver. Una de las cualidades principales de los algoritmos gené-

ticos es su facilidad de paralelización, ya que están basados en poblaciones con individuos independientes, de esta manera, el cálculo de la función de aptitud y de los operadores genéticos en un individuo no depende del cálculo en los demás individuos [2].

Típicamente, la población inicial se genera de manera aleatoria con distribución de probabilidad uniforme. El tamaño de la población inicial es importante porque influye en si el AG puede encontrar buenas soluciones y en el tiempo que se tarda en llegar a ellas. Si la población es demasiado pequeña, puede que no haya una muestra adecuada de individuos, y será difícil identificar buenas soluciones [2]. Si la población es demasiado grande, es necesario usar una gran cantidad de recursos de cómputo y el tiempo de procesamiento puede ser muy grande. En cada iteración del AG se crea una nueva generación de individuos en función de sus antecesores, teniendo más probabilidad de reproducirse los que tengan una mejor función aptitud.

Los AGs generalmente son capaces de aproximar la solución a un problema de optimización combinatoria en un tiempo razonable. Sin embargo, cuando se está cerca de la solución y se desea reducir el error de aproximación el tiempo de ejecución tiene un incremento muy grande. Como consecuencia, se han hecho múltiples esfuerzos para implementar AGs más rápidos, y una de las opciones más prometedoras es el uso de implementaciones paralelas, con lo que se puede obtener una reducción substancial del tiempo de procesamiento. En años recientes el desarrollo de procesadores gráficos más potentes ha tenido un fuerte impulso, como consecuencia es posible contar con plataformas de cómputo de alto rendimiento de bajo costo.

En este trabajo se resuelve un problema de optimización combinatoria por medio de algoritmos genéticos paralelos (AGPs). El problema consiste en encontrar la combinación de posiciones y coeficientes de los pesos de los puntos de control de una curva NURBS (del inglés, *Non-Uniform Rational B-spline*) que generarán la mejor aproximación posible a un conjunto de puntos que describen un contorno en imágenes médicas de resonancia magnética (puntos dato). La complejidad computacional de este problema es NP-Completo [3]. El espacio solución de este problema consiste en una gran cantidad de combinaciones de las posiciones y los coeficientes de los pesos de los puntos de control de la curva paramétrica, por lo tanto no es apropiado usar un algoritmo de búsqueda exhaustiva. Para este fin se desarrollaron dos versiones de algoritmos genéticos paralelos implementados en un GPU, un algoritmo genético paralelo simple y un algoritmo genético paralelo basado en el modelo de islas. Se hace una comparación entre ellos y se muestran los resultados obtenidos.

2. Curva B-spline racional no uniforme (NURBS)

Una curva NURBS está completamente determinada por sus puntos de control, lo cual implica que la curva cambia de manera predecible conforme lo hacen sus puntos de control [4]. A esto se le conoce como la propiedad de soporte local y permite que el movimiento de un punto de control sólo afecte a la curva localmente. Una curva NURBS está definida como [5]:

$$C(t) = \frac{\sum_{i=1}^{n+1} w_i P_i N_{i,k}(t)}{\sum_{i=1}^{n+1} w_i N_{i,k}(t)} \qquad a \le t \le b \tag{1}$$

donde:
w_i son factores de ponderación o pesos,
P_i son los puntos de control,
$N_{i,k}$ son las funciones base de orden k (grado $k-1$).

Las funciones base son determinadas por la recurrencia:

$$N_{i,1}(t) = \begin{cases} 1 & \text{si } x_i \le t < x_{i+1} \\ 0 & \text{en otro caso} \end{cases} \tag{2}$$

$$N_{i,k}(t) = \frac{t - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(t) + \frac{x_{i+k} - t}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(t) \tag{3}$$

donde el vector de nodos está determinado por $X = (x_0, \ldots, x_{n+k+1})$ y satisface la relación $x_i \le x_{i+1}$.

La parametrización tiene el objetivo de asociar las coordenadas (x, y) de los puntos dato a un parámetro de t en el dominio de la curva, es decir es una función de $\Re^2 \to \Re$. Si los puntos a ser aproximados son $Q_0, \ldots Q_n$, entonces se deben encontrar los parámetros $\tau_0, \ldots \tau_n$, uno para cada punto dato. En este trabajo se utiliza el método de parametrización de longitud de cuerda (del inglés, *chord length*) que se define por medio de la siguiente recurrencia [6]:

$$\tau_0 = 0 \tag{4}$$

$$\tau_i = \tau_{i-1} + \frac{\|Q_i - Q_{i-1}\|}{\sum_{i=1}^{n+1} \|Q_i - Q_{i-1}\|} \tag{5}$$

donde τ_{i+1} representa la suma de las distancias de los puntos dato anteriores y Q_i los puntos dato, por lo que cada valor τ_i representa un punto dato en el dominio de

la curva.

El vector de nodos se construye en base al método de parametrización de longitud de cuerda de la siguiente manera [4]:

$$x_{i+k-1} = \frac{i(\tau_{i+1} - \tau_i)}{n-k+1} + \tau_i \qquad 0 \leq i \leq k-1 \qquad (6)$$

3. Algoritmos genéticos paralelos

Los algoritmos genéticos paralelos surgen ante la necesidad de cómputo requerida por problemas de extrema complejidad cuyo tiempo de ejecución al utilizar los algoritmos genéticos secuenciales es una limitante [7]. Es por eso que se busca la manera de poder adaptar este tipo de heurísticas a distintas configuraciones de cómputo paralelo. La aplicación de los algoritmos genéticos paralelos tiene como finalidad dividir un problema en varios sub-problemas y resolverlos de manera simultánea en varios procesadores, lo cual permite mejorar el desempeño y la calidad de búsqueda de los algoritmos genéticos.

En general cuando se paraleliza un algoritmo su comportamiento es el mismo que el del algoritmo secuencial. Sin embargo este no es necesariamente el caso en la paralelización de algoritmos genéticos. En la estructura de un algoritmos genético paralelo una tarea puede dividirse en sub-tareas de forma que exista una distribución equilibrada de actividades; los miembros de las poblaciones de un algoritmo genético puede ser divididos en sub-poblaciones que son distribuidos en diferentes procesadores mediante mecanismos de comunicación y control que ayudarán a generar una estructura sólida a nivel de los algoritmos genéticos en un ambiente paralelo.

Existen varias formas de paralelizar un algoritmo genético. La primera y más intuitiva es la global, que consiste básicamente en paralelizar la evaluación de la función de aptitud de los individuos manteniendo una sola población. Una mejor opción de paralelización de los algoritmos genéticos es dividir la población en subpoblaciones que evolucionan por separado e intercambian individuos cada cierto número de generaciones [7]. A continuación se describen cada una de las etapas de un algoritmo genético.

3.1. Representación de los individuos

Una consideración importante que se debe tener al diseñar un AG es definir una representación para los individuos de la población que modelan la solución del problema [1]. Los individuos son representados por su cromosoma, que contiene una cadena de genes que simbolizan los datos que queremos optimizar. Algunas formas de representar el cromosoma de cada individuo son, por medio de una cadena binaria, una lista de números. En este trabajo cada gen del cromosoma de los individuos contiene tres valores, dados por la posición (x, y) y el peso de cada punto de control de la curva NURBS.

3.2. Función de aptitud

La función de aptitud es la que determina el potencial de solución que tienen los individuos dentro de la población [1] En este trabajo la función de aptitud está determinada en base a la sumatoria de las distancias de cada punto dato y su correspondiente punto de la curva, entre menor sean estas distancias mejor será la solución que ofrece el individuo y está dada por:

$$\lambda = \sum_{i=1}^{n} \sqrt{(x_i + \hat{x}_i)^2 + (y_i + \hat{y}_i)^2} \qquad (7)$$

donde (x_i, y_i) son los puntos de los datos y $(\hat{x}_i, \hat{y}_i)$ son los puntos de la curva.

El problema de definir la función de aptitud debe considerarse cuidadosamente para que se pueda alcanzar una buena aproximación al problema planteado. Si se elige mal una función aptitud o se define de manera inexacta, puede que el AG sea incapaz de encontrar una solución al problema, o puede acabar resolviendo un problema equivocado.

3.3. Operador de selección

Por medio del operador de selección se escogen los individuos que aplicarán el operador de cruce. Los individuos seleccionados heredarán sus características a la siguiente población de individuos de las posibles soluciones [1]. El operador de selección no produce puntos nuevos en el espacio de búsqueda, sino que determina qué individuo dejará descendencia y en qué cantidad lo hará para la próxima generación. Existen varios métodos de selección que se utilizan en los algoritmos genéticos, como por ejemplo selección por ruleta, selección por torneo, selección proporcional.

En esta investigación se implementa selección por torneo en el algoritmo genético paralelo simple, en la cual se elige aleatoriamente una pequeña muestra de la población y de ella se selecciona el individuo con mejor valor en su función aptitud (todos los números aleatorios

que se utilizan en este trabajo se generan con una distribución uniforme). Para el algoritmo genético paralelo con modelo de islas se selecciona un individuo de manera aleatoria de la misma isla o de la isla élite, ya que el tener una isla élite se tiene una mayor probabilidad de seleccionar un individuo de mayor función aptitud, este tipo de cruce se propone en este trabajo debido a la ventaja que le da el tener la isla élite al algoritmo. En ambos caso solamente se selecciona un individuo para ser cruza ya que el otro individuo está definido por el hilo ya que en ambos algoritmos cada individuo esta mapeado a un hilo en el GPU.

3.4. Operador de cruce

Para mejorar los individuos de la población se utiliza el operador genético de cruce. Para llevar a cabo la operación de cruce, se seleccionan dos individuos para ser combinados y el individuo resultante tiene la posibilidad de reemplazar a uno de los padres o al individuo con menor aptitud en la población [1]. El nuevo individuo va a sustituir a uno de los padres siempre y cuando tenga mejor función aptitud. En caso de que el nuevo individuo no sea mejor que los padres se puede intentar sustituirlo por el peor individuo de la población. El proceso está orientado a las sub-regiones del espacio de búsqueda, donde se supone que existe una solución óptima. Existen varias formas de aplicar el operador de cruce tales como cruce por un único punto, cruce por puntos múltiples, cruce uniforme, entre otras.

En este trabajo cada individuo de la población hace cruce con otro individuo escogido por el operador de selección en cada iteración y se utiliza el cruce por dos puntos, en donde dos posiciones en la cadena de genes de los cromosomas de ambos padres se seleccionan aleatoriamente. Uno de los padres aporta la información que queda fuera del rango entre las dos posiciones y el otro proporciona la información que queda dentro de las dos posiciones.

3.5. Operador de mutación

El operador de mutación crea un individuo realizando algún tipo de alteración, usualmente pequeña, en un individuo de la población escogido de forma aleatoria con una probabilidad pequeña [1]. La mutación tiene el objetivo de dispersar la búsqueda del algoritmo ya que da origen a individuos con material genético nuevo. En este trabajo, en el proceso de mutación algunos genes del cromosoma del individuo se seleccionan aleatoriamente para ser reemplazados por otros valores, que son seleccionados de manera aleatoria dentro del rango de las soluciones posibles.

4. Unidades de procesamiento gráfico

La Unidad de Procesamiento Gráfico (GPU, del inglés *Graphics Processing Unit*) es un dispositivo de cómputo capaz de ejecutar una gran cantidad de procesos en paralelo, a través del uso de hilos (*threads*, en inglés). La GPU funciona como un coprocesador en una computadora de propósito general cuyo procesador central es un CPU (del inglés *Central Processing Unit*) [8].

El modelo de programación utilizado en los procesadores gráficos según la taxonomía de Flynn es el modelo SIMD (del inglés, *Single Instruction Multiple Data*), en este modelo todas las unidades de procesamiento ejecutan la misma instrucción y operan sobre diferentes datos. Esto reduce la complejidad en el diseño de la unidad de control y permite que se dedique un mayor espacio en el procesador gráfico para el procesamiento de datos.

La ventaja de utilizar un procesador gráfico en lugar de un procesador de propósito general consiste en que, en aquellos casos en los que es posible paralelizar las operaciones que se efectúan sobre un conjunto de datos, la GPU supera ampliamente a la CPU [9].

La función destinada para ejecutarse en la GPU se llama *kernel*. Cuando una función *kernel* se invoca desde el CPU, se indica el número de bloques e hilos que van a ejecutarla [8]. Cada bloque de hilos se ejecuta por los núcleos de un multiprocesador. Dependiendo de la cantidad de multiprocesadores de la GPU, el *kernel* puede ser completado en una, o varias re-distribuciones de bloques de hilos a los multiprocesadores.

5. Implementación de algoritmos genéticos paralelos en una GPU

En esta sección se explica cómo se implementa un algoritmo genético paralelo en una GPU. Por lo que se describe a detalle las dos versiones de algoritmos genéticos paralelos que se implementaron. Es importante resaltar que la cantidad de hilos es igual a la cantidad de individuos, ya que cada individuo de la población se mapea en un hilo del GPU.

5.1. Algoritmo genético paralelo simple

En el algoritmo genético paralelo simple (AGPS), sólo hay una población, en la que la evaluación de los individuos y los operadores genéticos se paralelizan de forma explícita [2]. Puesto que sólo hay una población,

Tabla 1. **Algoritmo 1**. Seudocódigo para el AGPS

```
main( ) {
1. asignación de memoria en el CPU y en el GPU
2. calcular parametrización de los puntos dato
3. copia de memoria del CPU a GPU
4. hilos = NUM_IND
5. bloques = NUM_ISLAS
6. GenPobIni <<<bloques,hilos>>>(parametros)
7. while condición_terminar == FALSO do
8.    AGPS<<<bloques,hilos>>>(parametros)
9. end while
10. copia de memoria de población del GPU a CPU
11. obtener mejor solución
12. liberar memoria del GPU y CPU
}
_ _global_ _GenPobIni( ){
13. GenPob(P)
14. Evaluar(P)
}
_ _global_ _AGPS( ){
15. seleccionar al mejor individuo
16. SelecPadres(P)
17. P' ← CrucePadres(P)
18.   if prob_mut <5 % then
19.   P' ← Mutacion(P)
20. end if
21. Evaluar(P')
22. if Aptitud(P') < Aptitud(P) then
23.   P = P'
24. end if
}
```

la selección considera a todos los individuos y cada individuo tiene oportunidad de aparearse con cualquier otro.

El AGPS es un método relativamente fácil de implementar y puede obtenerse una mejora significativa con respecto a la versión secuencial, si los costos de comunicación no dominan a los costos de procesamiento. Al algoritmo genético paralelo simple también se le conoce como algoritmo genético paralelo panmítico, pues se cuenta con un solo depósito de material genético (*gene pool*), es decir, con una sola población.

La implementación del AGPS en la GPU se describe en el seudocódigo del Algoritmo 1. El primer paso es asignar memoria en el CPU y en la GPU, luego se calcula en el CPU la parametrización de los puntos de los datos para llevarlos al dominio de la curva, después se copia de la memoria del CPU a la memoria del GPU los valores de la parametrización y los demás datos que se necesitan.

Enseguida se llama al *kernel GenPobIni* donde cada hilo genera el individuo que va a procesar, esto es, para cada punto dato se generan las coordenadas de la posición y los pesos de su correspondiente punto de control. Las coordenadas de la posición se obtienen de manera aleatoria dentro de un círculo con centro en el punto dato y un radio de 10 unidades de distancia. Se propone el valor del peso como un número aleatorio con distribución uniforme en un intervalo de (0,1) con incremento de 0.1 que se asigna a cada punto de control. Al final en este *kernel* se evalúa la función aptitud para cada individuo, la cual es la sumatoria de la distancia que existe entre los puntos de la curva y los puntos dato. Para evaluar la función aptitud es necesario generar la curva NURBS que se define por los puntos de control y los pesos de cada individuo.

Los individuos generados en el *kernel GenPobIni* son almacenados en una variable llamada población vieja *P*, donde cada hilo tiene espacio para guardar las posiciones (x, y) y los pesos w de los puntos de control. Después entra en un ciclo que procesa el *kernel AGPS*, en general el criterio de paro está determinado por el número de iteraciones o alcanzar un cierto valor de la función aptitud, en este trabajo se eligió que termine cuando se alcanza un cierto número de iteraciones.

El *kernel AGPS* lo primero que hace es seleccionar al mejor individuo de toda la población con el fin de mantenerlo y no sustituirlo si no es por otro mejor, para ello se aplica un algoritmo de reducción [8], luego se selecciona un individuo con el que se hará cruza por medio del operador de selección de torneo, el cual compara dos individuos y selecciona el que tenga mayor función aptitud. Como se puede apreciar, todos los hilos entran a hacer el cruce en cada iteración. Luego se aplica el operador de cruce entre los dos padres. Posteriormente se aplica el operador de mutación, cada individuo tiene una probabilidad de 5 % de ser mutado. En la mutación cada uno de los genes del individuo que se alterarán es cambiado por un punto aleatorio dentro de un círculo de radio 5 y para el peso se sigue con las mismas consideraciones que cuando se genera la población inicial.

El individuo generado por cruce se guarda en la variable *P'* que representa la población nueva, donde cada hilo tiene espacio para guardar las posiciones (x, y) y los pesos w de los puntos de control del individuo generado. En caso de que el hilo entre también a hacer mutación, el individuo generado entra directamente en la población vieja sin reemplazar al mejor individuo.

Al final, en el *kernel AGPS*, a los individuos generados se les evalúa su función aptitud y en caso de que el individuo de la población nueva tenga una mejor aptitud que el individuo de la población vieja, se sustituye. Cuando las iteraciones terminan se copia la población vieja al CPU. Luego se obtiene la mejor solución generada por el AG y finalmente se liberan las memorias del CPU y el GPU.

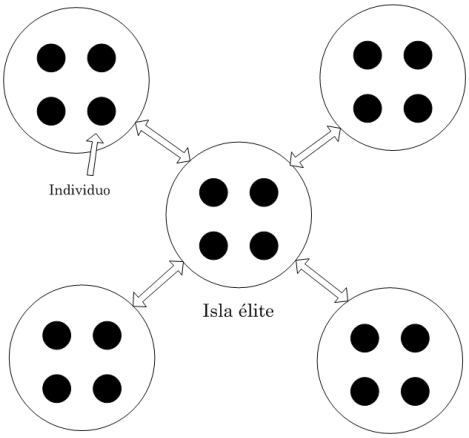


Figura 1. Organización de los individuos de la población del AGPI.

5.2. Algoritmo genético paralelo con modelo de islas

En el algoritmo genético paralelo con modelo de islas (AGPI) la población se divide en subconjuntos, también conocidos como poblaciones locales (en inglés se les conoce como *demes*), que evolucionan de manera aislada, aunque intercambian individuos con la frecuencia que se indica en el algoritmo [2]. A este intercambio de individuos se le llama migración, y se considera como un nuevo operador genético. El AGPI que se propone en este trabajo tiene una modificación, en lugar de migrar los individuos entre las islas, se construye una isla con los mejores individuos de cada una de las subpoblaciones, denominada isla élite. Esta isla de mejores individuos es compartida con todas las demás islas. En la Fig. 1 se puede observar la organización de los individuos de la población del AGPI.

La implementación del AGPI en la GPU es muy similar a la manera que se implementa el AGPS. La diferencia en este caso es que como se manejan subpoblaciones aisladas es necesario mapear las islas en los bloques de hilos, por lo que cada bloque representa una isla. En el Algoritmo 2 se presenta el pseudocódigo para el AGPI, el proceso es el mismo que en el algoritmo del AGPS hasta la ejecución del *kernel GenPobIni*.

En el *kernel GenPobIni* los individuos se crean de la misma manera como se crearon en el AGPS, con la diferencia de que después de la evaluación de la función aptitud se selecciona el mejor individuo de cada isla y se migra a la isla élite, para esto fue necesario asignar dos variables en memoria compartida un arreglo para guardar todas las aptitudes de la isla para utilizarlo en el algoritmo de reducción y una variable para guardar el identificador del hilo del mejor individuo, con el fin de que solo los individuos de una isla puedan acceder a

Tabla 2. **Algoritmo 2**. Seudocódigo para el AGPI

```
main( ){
1. asignación de memoria en el CPU y en el GPU
2. calcular parametrización de los puntos dato
3. copia de memoria del CPU a GPU
4. hilos = NUM_IND
5. bloques = NUM_ISLAS
6. GenPobIni<<<bloques,hilos>>>(parametros)
7. while condición_terminar == FALSO do
8.    AGPI<<<bloques,hilos>>>(parametros)
9. end while
10. copia de memoria de isla élite del GPU a CPU
11. obtener mejor solución
12. liberar memoria del GPU y CPU
}
_ _global_ _GenPobIni( ){
13. GenPob(I)
14. Evaluar(I)
15. seleccionar al mejor individuo de cada isla
16. MigrarIndElite(E)
}
_ _global_ _AGPI( ){
17. SelecPadres(I,E)
18. I'← CrucePadres(I,E)
19. Evaluar(I')
20. if Aptitud(I')<Aptitud(I) then
21.    I = I'
22. end if
23. if prob_mut <5 % then
24. I'← Mutacion(I)
25. end if
26. seleccionar al mejor individuo de cada isla
27. MigrarIndElite(E)
}
```

estas variables y mantener las islas aisladas.

Los individuos generados en el *kernel GenPobIni* son almacenados en la variable *I* que representa la población vieja. Además hay una variable *E* llamada élite, en donde hay un espacio asignado para guardar el mejor individuo de cada isla. Después se entra en un ciclo que procesa al *kernel AGPI*, que termina cuando se alcanza un cierto número de iteraciones. Al momento de hacer el cruce solo se selecciona aleatoriamente un individuo de la misma isla *I* o de la isla élite *E*. Después, a los individuos generados se les evalúa con la función de aptitud. En caso de que el individuo de la población nueva *I'* tenga mejor aptitud que el individuo de la población vieja *I*, se sustituye.

Luego se aplica la mutación de manera muy similar a la manera que se aplicó en AGPS, con la diferencia de que ahora los individuos generados tienen la excepción de reemplazar al mejor individuo de la isla. Al final del *kernel AGPI* se selecciona el mejor individuo de cada isla y se migra cada uno isla élite. Cuando las iteraciones terminan sólo se copia la isla élite al CPU. Luego se obtiene la mejor solución generada por el AG y finalmente se liberan las memorias del CPU y el GPU.

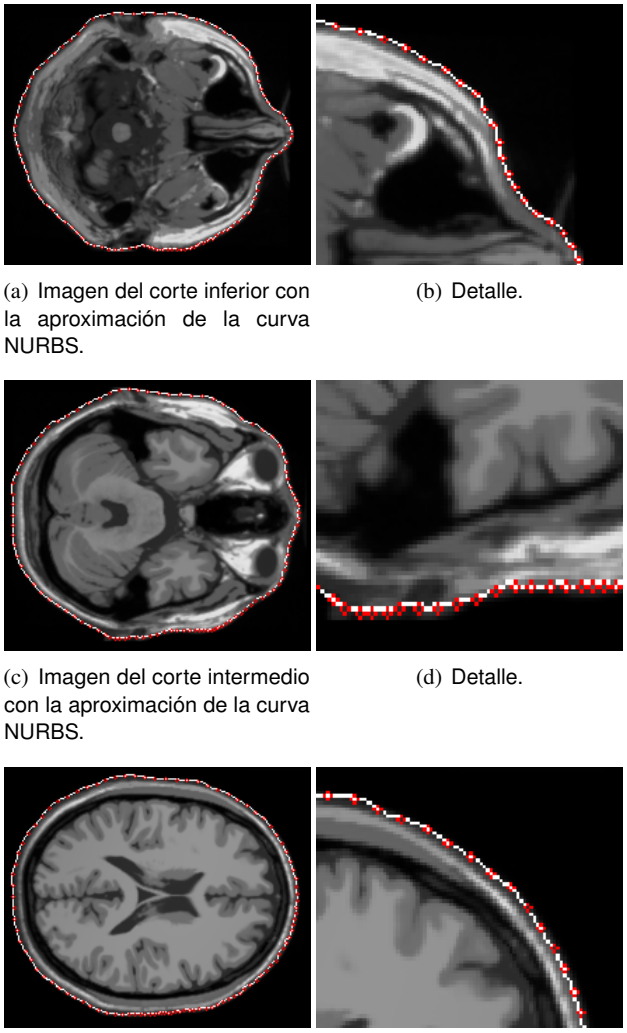
6. Resultados

La implementación del algoritmo de aproximación se realiza en el sistema operativo GNU/Linux y lenguaje CUDA C en una computadora con un CPU de intel i7 de 2.8 GHz. El algoritmo se procesa en un GPU GeForce GTX 670.

Se presentan algunos experimentos para optimizar la aproximación de una curva NURBS a un conjunto de puntos. En los experimentos se utilizaron imágenes de resonancia magnética de diferentes cortes, tomadas de [10]. Las imágenes tienen un tamaño de 217×181 píxeles con 8 bits de profundidad. Para estos experimentos se utilizó el algoritmo de aproximación con optimización de parámetros por el algoritmo genético paralelo con modelos de islas. La entrada y la salida de las imágenes se manejan por medio de la biblioteca de funciones para visión por computadora OpenCV, acoplada con CUDA.

Los puntos que se obtienen de la imagen son generados por el algoritmo de segmentación que se desarrolló en el trabajo de tesis [11], el cual emplea el método de modelos deformables.

En todos los experimentos se usaron 512 hilos por bloque, un porcentaje de mutación del 5 %, a cada individuo seleccionado para la mutación se le muta el 30 % de los genes y las curvas NURBS usadas son de orden 3 (grado 2). El tiempo (σ) es el tiempo que tarda en ejecutarse el programa completo y el error de la aproximación (ϵ) es el valor de la función aptitud del mejor individuo que proporciona el algoritmo genético (el mejor individuo es la solución que presenta la distancia mínima entre los puntos a aproximar y la curva NURBS). Ver Ec. (7).



(a) Imagen del corte inferior con la aproximación de la curva NURBS.

(b) Detalle.

(c) Imagen del corte intermedio con la aproximación de la curva NURBS.

(d) Detalle.

(e) Imagen del corte superior con la aproximación de la curva NURBS.

(f) Detalle.

Figura 2. Resultado de la aproximación.

Para los experimentos de la Fig. 2, se utilizaron imágenes de resonancia magnética. En estos tres experimentos los parámetros que se utilizaron fueron 1200 iteraciones, el número de puntos que describen el contorno exterior de la imagen médica es de 100 y el número de bloques de hilos igual a 80. La imagen de la Fig. 2(a) es el corte inferior, la imagen de la Fig. 2(c) es el corte intermedio y la imagen de la Fig. 2(e) es el corte superior. En las Figs. 2(a), 2(c) y 2(e) se observa la aproximación final que se obtuvo y en las Figs. 2(b), 2(d) y 2(f) se observa una ampliación de una parte de la

imagen para mostrar la forma en que las curvas NURBS se ajustan a los puntos dato.

En el experimento de la Fig. 2(a) los resultados de la aproximación que se obtuvieron fueron $\epsilon = 10.89$ y $\sigma = 1959.30\ s$. En la Fig. 2(b) se observa que la curva pasa exactamente por los puntos de dato de esa sección, ya que describen una curvatura suave que puede ser descrita por el grado de ese tramo de la curva.

Los resultados de la aproximación obtenidos para la Fig. 2(c) fueron $\epsilon = 10.76$ y $\sigma = 1960.65\ s$. En la Fig. 2(d) se observa que la curva no pasa exactamente por los puntos dato esto se debe a que el contorno es muy irregular en esa área por lo que el grado de la curva que se utiliza no es suficiente para interpolarlos, esta es una limitación del algoritmo de aproximación que se desarrolló.

En el experimento de la Fig. 2(e) los resultados de la aproximación que se obtuvieron fueron $\epsilon = 12.90$ y $\sigma = 1961.22\ s$. En la Fig. 2(f) se puede ver que la curva pasa exactamente por los puntos de datos, ya que en esa sección el contorno es suave y puede ser descrito por ese tramo de la curva.

6.1. Comparación de algoritmos genéticos paralelos

Se hace una comparación entre las implementaciones del algoritmo genético paralelo simple (AGPS) y el algoritmo genético paralelo con modelo de islas (AGPI). Se utilizaron los puntos de la imagen de resonancia magnética del corte inferior Fig. 2(a). Los parámetros definidos que se utilizaron para ambos algoritmos son porcentaje de mutación = 5 %, hilos por bloque = 512, cantidad de puntos a mutar de cada individuo = 30 %, número de puntos de la imagen médica = 100 y orden de la curva = 3, es necesario aclarar que en el caso del AGPS se procesa una sola población es decir 512 individuos equivalentes a la población de una isla.

Los resultados de los experimentos del AGPI se pueden ver en la Tabla 3 y los resultados del AGPS se pueden ver en la Tabla 4. Para los experimentos del AGPI los parámetros que se varían el número de iteraciones y el número de isla, mientras que para los experimentos del AGPS sólo se varía el número de iteraciones ya que la población es constante para este caso. Cada uno de los resultados mostrados en la tabla es el resultado de la media de 10 pruebas realizadas. Se observa que el AGPI obtiene mejores soluciones de aproximación que el AGPS esto se debe a la ventaja que ofrece el tener una isla élite para compartir los mejores individuos entre las islas, además de que se tienen varias islas que evolucionan al mismo tiempo por lo que tiene un campo de exploración mucho mayor y se necesita un tiempo menor para encontrar una solución mejor. Al tener una sola población se hace necesario que se incremente el número de iteraciones para obtener el mejor resultado provocando que el tiempo necesario para encontrar una solución sea muy grande y el tener una sola población produce que el campo de exploración no sea tan grande como el caso del AGPI.

Tabla 3. Algoritmo genético paralelo con modelo de islas (AGPI) en el GPU.

Iteraciones		Número de islas		
		40	60	80
300	ϵ	25.25	25.23	24.81
	$\sigma\ (s)$	248.12	384.89	490.23
600	ϵ	18.64	18.05	16.27
	$\sigma\ (s)$	490.86	768.83	977.27
900	ϵ	15.25	15.1	13.93
	$\sigma\ (s)$	739.54	1151.47	1464.34
1200	ϵ	13.74	12.86	11.56
	$\sigma\ (s)$	986.87	1534.31	1954.4
1500	ϵ	13.37	12.16	11.45
	$\sigma\ (s)$	1237.53	1919.61	2442.79

Tabla 4. Algoritmo genético paralelo simple (AGPS) en el GPU.

	Número de iteraciones			
	10000	20000	40000	60000
ϵ	25.15	16.44	14.07	12.44
$\sigma\ (s)$	832.72	1645.55	3593.67	4940.27

7. Conclusiones

En este trabajo se implementaron algoritmos genéticos paralelos en un GPU para la optimización de la aproximación de una curva NURBS a un conjunto de puntos de una imagen médica. Se desarrollaron dos versiones de algoritmos genéticos paralelos, un algoritmo genético paralelo simple y un algoritmo genético paralelo con modelo de islas.

El algoritmo genético paralelo con modelo de islas da mejores resultados que el algoritmo paralelo simple, debido a que da una mejor aproximación y lo hace en la mitad del tiempo del que le toma al algoritmo genético paralelo simple. Esta ventaja se la proporciona el tener la isla élite que ayuda al programa a dar mejores resultados ya que se comparten los individuos más aptos entre todas las islas, y además ayuda a hacer más rápida la migración ya que de otra manera se tendría que hacer migración entre las islas. También, al evolucionar simultáneamente varias islas de individuos contribuyen a que el campo de exploración de soluciones sea mayor.

Al implementar el algoritmo genético paralelo con modelo de islas en el GPU se aprovecha su arquitectura al mapear las islas y los individuos en los bloques de hilos y los hilos, respectivamente. También la manera en la que se lleva a cabo la selección para hacer el cruzamiento es distinta a como se hace comúnmente en los algoritmos genéticos secuenciales, ya que al mapear un individuo a cada uno de los hilos se tiene la seguridad de que ese individuo entra al operador de cruce y el otro individuo se escoge de manera aleatoria, esta forma de seleccionar al otro individuo se debe a la característica esencial de este algoritmo que es la isla élite que aporta una mayor cantidad de mejores individuos dentro de la población. Además, en este algoritmo se asegura que en cada iteración se crea una generación completa, porque todos los individuos hacen cruzamiento en cada iteración.

Agradecimientos

Se agradece el apoyo recibido por parte del proyecto SIP-20130489.

Referencias

[1] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, 2008.
[2] G. Luque and E. Alba, *Parallel Genetic Algorithm: Theory and Real World Applications*. Springer, 2011.
[3] T. H. Cormen *et al*, *Introduction to Algorithms*, Third edition, MIT Press, 2009.
[4] D. F. Rogers, *An Introduction to NURBS*, Morgan Kaufmann Publishers, 2001.
[5] L. Piegl and W. Tiller, *The NURBS Book*, Springer, 1997.
[6] J. J. Fang and C. L. Hung, "An improved parametrization method for B-spline curve and surface interpolation," *Comput.-Aided Des.*, Vol. 45, No. 6, pp. 1005-1028, 2013.
[7] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithm*, Kluwer Academic Publishers, 2000.
[8] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, 2010.
[9] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2010.
[10] C. Cocosco *et al*., "BrainWeb: Online Interface to a 3D MRI Simulated Brain Database," *Third Intl. Conf. on Functional Mapping of the Human Brain*, Vol. 5, No. 4, Copenhagen, Denmark, May 1997, p. S425.
[11] R. Alvarado, "Procesamiento de imágenes aplicando modelos deformables con computo de alto rendimiento," Tesis de maestría, Centro de Investigación y Desarrollo de Tecnología Digital del IPN, 2012.