Reorganización de matrices en algoritmos de barrido radial sobre Modelos Digitales del Terreno

Andrés Jesús Sánchez[1], Luis F. Romero[1], Siham Tabik[2], Gerardo Bandera[1]

Resumen— **Es muy frecuente, en los sistemas de información geográfica que trabajan con modelos digitales del terreno, el uso de algoritmos de barrido radial para el estudio de variables asociadas a parámetros cuya magnitud decrece con el cuadrado de la distancia, como las señales de radio, las ondas de sonido, o la propia visión humana. Sin embargo, dichos algoritmos están asociados a un acceso a las matrices de datos que, en la mayoría de los casos, aun siendo regular, derivan en un mal aprovechamiento de la localidad de la memoria. En este trabajo se muestra cómo la completa reorganización previa de las matrices de datos, en función de la dirección radial que corresponde, produce una considerable mejora del rendimiento, especialmente en algoritmos de elevada intensidad computacional. Sirva como ejemplo el cálculo de cuencas visuales totales, que es utilizada en este trabajo como caso de estudio. Por otra parte, la reestructuración matricial propuesta abre la puerta al uso intensivo de GPUs en muchos algoritmos para los que nunca se han considerado, por su irregularidad y baja eficiencia.**

Palabras clave— **Modelos Digitales del Terreno, arquitecturas heterogéneas CPU-GPU, Supercomputación, Visibilidad Total.**

I. Introducción

SON muy numerosos los algoritmos en los que el análisis de un Modelo Digital del Terreno, o MDT, se realiza mediante un barrido radial desde un punto situado en el propio territorio. Normalmente, a dicho punto de referencia se le denomina Punto de Observación, o POV (acrónimo del inglés *point of view*), ya que se considera que dicho punto tiene un especial interés. En muchos casos, el POV es emisor o receptor de señales cuya intensidad decrece con el cuadrado de la distancia (una señal emisora de radio, la visión de un observador sobre el terreno, o una fuente de sonido). Precisamente debido al decaimiento de la intensidad de las señales, este tipo de problemas se caracterizan porque la relación entre el POV y el resto del territorio se debilita con la distancia y, en consecuencia, la precisión requerida es más baja para los puntos más alejados.

Los algoritmos de barrido radial analizan la relación entre el POV y el resto del territorio mediante una discretización acimutal de los cálculos. Es decir, considerando al POV como vértice, el territorio se divide en sectores angulares horizontales, de forma similar a los cuadrantes de una rosa de los vientos. E independientemente del número de sectores (ya sean los 360 grados sexagesimales, o los 4 cuadrantes N, S, E y W) en todos estos problemas se considera que el eje del sector es el representante estadístico de todo el sector. Es obvio que la representatividad de dicho eje se reduce a a medida que nos alejamos del vértice, ya que el ancho del sector aumenta linealmente con el radio. Pero también resulta evidente que se reduce en la misma medida la necesidad de precisión, haciendo que este tipo de algoritmos sean especialmente adecuados para estos problemas. De hecho, algoritmos similares se pueden extender a problemas tridimensionales, como pueden ser las técnicas de *ray-tracing* en visión por computador, o en el renderizado de volúmenes.

A. Visibilidad del territorio

Quizás uno de los problemas más representativos de lo expuesto anteriormente sea el cálculo de la visibilidad de un observador ubicado a cierta elevación sobre el terreno. Un conocimiento preciso de la visibilidad en un territorio proporcionaría una herramienta de gran interés para distintas aplicaciones en diversos campos, tales como las telecomunicaciones, planificación del medio natural, ecología, turismo, arqueología, etc. Así, como ejemplo, la elaboración de algoritmos que determinen el menor número posible de puntos de observación necesarios para dar una máxima cobertura visual (o de telefonía) de un terreno, se simplificaría enormemente si se conociera a priori qué parte del territorio es visible desde cada punto [1].

Se conoce como Cuenca Visual (o *viewshed, VS*, del inglés) de un observador del terreno, al área del territorio visible desde un determinado punto que, normalmente, se encuentra situado a una altura h sobre el terreno. Existen numerosos algoritmos que calculan el VS desde un único punto de vista o, como mucho, desde un número muy reducido de puntos de observación. En este contexto, la mayoría de sistemas de información geográfica (GIS, por sus siglas en inglés) como ArcGIS [2], o GRASS GIS [3], e incluso aplicaciones de consumo masivo, como Google Earth, incorporan módulos para el cálculo de la cuenca visual. Además de ellos, existen numerosos algoritmos con el mismo propósito [4], [5], [6], [7], [8].

El algoritmo responsable del cálculo de la visibilidad mediante un barrido radial es bastante simple. Como se ha expuesto anteriormente, es necesaria una discretización previa del territorio que rodea al observador en un número ns de sectores acimutales (en lo sucesivo, utilizaremos sectores de 1^o y $ns = 360$

[1]Dpto. de Arquitectura de Computadores, Universidad de Málaga, e-mail: ajsanchez@ac.uma.es, felipe@uma.es, gbandera@uma.es

[2]Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, e-mail: siham@ugr.es

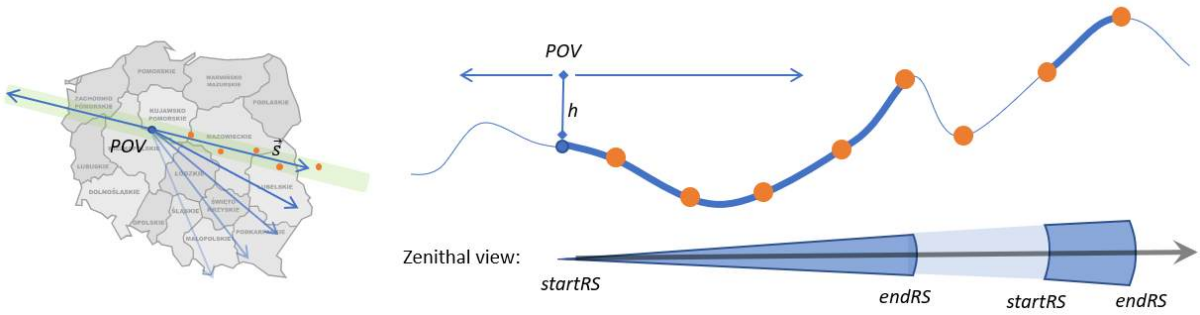

Fig. 1. Barrido radial para calcular la visibilidad.

para simplificar). Para cada uno de los sectores s elegidos, y para cada sentido del eje que pasa por el sector, se calcula su visibilidad seleccionando puntos próximos al eje como representantes de cada sector ($selectAxisPointSet$, suponemos el MDT representado con una malla regular cartesiana):

Algoritmo 1 Cálculo de la cuenca visual de un punto POV.

```
function viewshed(i, j, POVheight)
global point POV = MDT[i, j];
POV.h += POVheight;
pointSet axis = selectAxisPointSet(POV, s);
float VS = 0;
for s = 0, (ns/2 − 1) do
   VS += linearViewshed(axis, true); //forward
   VS += linearViewshed(axis, false);//backward
end for
VS *= (π/ns); //Papus theor. scaling
endfunction
```

donde i, j y $POVheight$ representan las coordenadas y elevación del punto de observación POV (el observador se sitúa ligeramente por encima del terreno); $axis$ es el conjunto de puntos que representan al sector (normalmente próximos al eje, según se muestra en rojo en la figura 1) y VS es la cuenca visual.

La cuenca visual (normalmente se mide la superficie visible) en un determinado sector, y para un observador situado sobre el eje del sector, se obtiene calculando la visibilidad de cada uno de los puntos restantes del eje, mediante un recorrido desde el más próximo al más lejano en el sentido correspondiente:

Algoritmo 2 Cálculo de VS para un punto en el eje de un sector.

```
function linearViewshed(axis, forward)
global bool visible = true;
global float max = -∞; // Max. angle
global pointSet visibleSet = emptySet;
T = axis.POV
while T != (forward)?axis.last : axis.first do
   pointViewshed(axis.POV,T);
   T = (forward)?axis.next() : axis.prev();
end while
return visibleSet.measure();
endfunction
```

donde $visibleSet$ es el conjunto de puntos del eje que son visibles desde POV, y T el punto destino. Obviamente, un punto destino es visible desde el punto origen si su altitud angular es superior a la de todos los puntos más próximos, y que han sido previamente recorridos y analizados. Sin embargo, es frecuente utilizar la tangente, en lugar del propio ángulo, para evitar la ejecución de algunas operaciones trigonométricas costosas e innecesarias, como se presenta en el Algoritmo 3.

Algoritmo 3 Cálculo de la visibilidad de un punto T desde POV.

```
function pointViewshed(POV, T)
float dist =distance(POV,T);
float angle = (T.h − POV.h)/dist;
visible = angle > max
if(visible) visibleSet.add(T);
endfunction
```

Una optimización evidente para este algoritmo, que no solo reduce accesos a memoria sino también cálculos, ha sido utilizada en [9], y consiste en almacenar sólo los puntos en los que empieza y finaliza una secuencia en la que todos los puntos son visibles desde POV, y a los que denominan *segmentos visibles* o *anillos sectoriales*, por la figura geométrica que se obtiene al extrapolar el segmento a todo el sector (véase la figura 1). En este caso, el algoritmo quedaría de esta forma:

Algoritmo 4 Visibilidad de T desde P mediante anillos sectoriales.

```
function pointViewshed(POV, T)
float dist = distance(POV,T);
float angle = (T.h − POV.h)/dist;
bool prevVisible = visible
visible = angle > max
bool startRS =!prevVisible & visible;
if(startRS) dist0 = dist;
bool endRS = prevVisible & !visible;
if(endRS) VS += dist ∗ dist − dist0 ∗ dist0;
endfunction
```

donde $startRS$ y $endRS$ son las variables que se encargan de indicar si durante el barrido a lo largo del eje se ha encontrado o abandonado una secuencia de puntos visibles. Por otro lado, se ha añadido una

variable global *dist*0 en la que se almacena la distancia entre POV y el primer punto de un tramo visible. Este valor se utiliza para calcular la superficie del anillo sectorial cuando aparezca el último punto del tramo, mediante la diferencia de cuadrados. Recordemos que el área de un anillo sectorial de 1º de apertura es $(\pi/360)\cdot(R^2-r^2)$.

II. Complejidad de los algoritmos de barrido en los MDT

Los algoritmos de barrido sectorial pueden reducir significativamente la complejidad de los cálculos inherentes a un determinado problema. Si, por ejemplo, intentamos calcular la elevación media de un territorio, podríamos reducir la complejidad desde $O(\mathrm{N})$ a $O(\mathrm{s}\cdot\mathrm{N}^{1/2})$, siendo N el tamaño del MDT, medido en puntos. Teniendo en cuenta que los MDT superan ampliamente los varios millones de puntos y que la discretización sectorial en raras ocasiones supera el grado de precisión, estamos hablando de una reducción en las operaciones que oscila entre uno y tres órdenes de magnitud [10]. Para el cálculo de la visibilidad, y teniendo en cuenta que, a priori, cualquier punto del territorio podría ser obstáculo para la visibilidad entre dos puntos cualesquiera del mismo, la complejidad del problema (ya optimizado) es $O(\mathrm{N}^{1.5})$, mientras que la del algoritmo basado en la discretización radial, como hemos visto, es de $O(\mathrm{s}\cdot\mathrm{N}^{0.5})$.

Aún así, estamos hablando de un número bastante elevado de operaciones que hacen que la supercomputación o el paralelismo sean altamente recomendables para estas aplicaciones. De hecho, uno de los problemas que hasta ahora se consideraban inabordables es el de la visibilidad total, que describiremos a continuación.

A. Visibilidad total de un MDT

Hasta hace apenas 5 años, nunca se había considerado abordable el problema del cálculo de la cuenca visual desde todos los puntos del territorio representado por un MDT (es decir, con los N puntos de observación). El motivo era evidente: en el mejor de los casos, el cálculo del VS de un solo POV en un territorio, representado por unos pocos millones de puntos, ya requería, al menos, de varios segundos e incluso minutos de CPU.

Si el MDT está representado por una malla cartesiana de puntos de dimensiones *dimx* y *dimy*, el problema se podría expresar de la siguiente forma:

Algoritmo 5 Cálculo de la cuenca visual total.

```
for i = 0, dimy − 1 do
  for j = 0, dimx − 1 do
    tvs[i,j] = viewshed(i, j, POVheight)
  end for
end for
```

donde *tvs*, del inglés *total viewshed* es una matriz, con las mismas dimensiones del MDT, en la que se guarda la extensión del VS de cada punto, considerado como POV del territorio. Un ejemplo de los resultados obtenidos se muestran en la figura 2, donde cada celda del MDT rasterizado indica, mediante un mapa de colores, si los puntos del territorio tienen muy baja visibilidad (gama de azules), o ésta es muy elevada (gama de rojos). La complejidad inherente del problema es en este caso, de $O(\mathrm{N}^3)$, ya que resolvemos N veces un problema de $O(\mathrm{N}^2)$, mientras que la del algoritmo basado en la discretización radial subiría hasta $O(\mathrm{s}\cdot\mathrm{N}^{1.5})$, lo que justifica que nunca se hubiera abordado (o al menos, con MDTs de grandes dimensiones).

Sin embargo, trabajos recientes han demostrado que, mediante técnicas *multigrid*, y ciertas optimizaciones de memoria es posible hacer los cálculos de la visibilidad total para MDTs de gran tamaño en tiempos razonables [11], [12], demostrándose incluso la enorme utilidad de dichos algoritmos para la ubicación de torres de observación [13]. Aún así, estos algoritmos trabajan con cifras descomunales para los accesos a datos y las operaciones en punto flotante, por lo que solo se ha podido tratar adecuadamente el problema gracias a una gestión eficiente de los accesos a memoria.

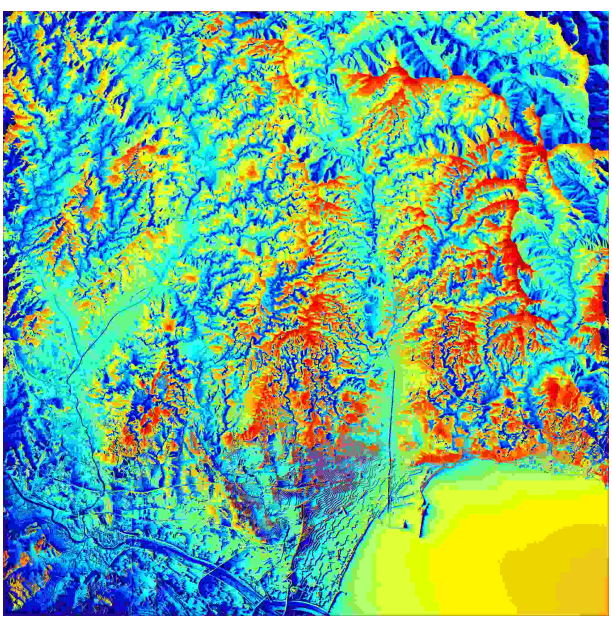

Fig. 2. Mapa de cuencas visuales totales en el entorno del municipio de Málaga, España.

B. Reutilización de datos en la banda de visibilidad

El conjunto de puntos que se tienen en cuenta para calcular la cuenca visual (representados por la estructura *axis* en el algoritmo 1) es denominado por los autores Cervilla *et al*[14] como Banda de Visibilidad, o BoS (del inglés *band of sight*). En otros algoritmos (véase, por ejemplo, Franklin *et al*[1]) se utilizan conceptos similares, aunque siempre hacen referencia a los puntos próximos al eje, o al resultado de la interpolación de los mismos. En la figura 1, a la izquierda, la banda de visibilidad se muestra en color verde. Es obvio que la reutilización de los datos en la banda de visibilidad, en la que concurren los puntos del MDT

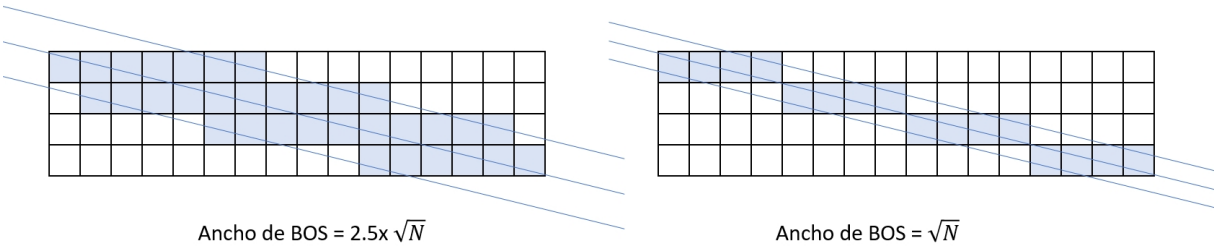

Fig. 3. Nodos que contribuyen a la BOS, con distintos anchos para la banda.

alineados en la dirección del sector elegido, es la clave de la optimización de los algoritmos de visibilidad total, pues precisamente se puede reutilizar dicha estructura para calcular la cuenca visual de todos los puntos alineados en la dirección de un mismo sector. En ese sentido se produce la principal aportación de dichos autores en [11]. Como se ha mencionado anteriormente, consiste en el máximo aprovechamiento de la memoria. Para ello, su algoritmo garantiza que sólo se va a producir un fallo de cache antes de ejecutar todas las operaciones aritméticas implicadas en el cálculo de la visibilidad de un único punto en un determinado sector. Para alcanzar esa considerable reutilización de los datos, han invertido las operaciones: en lugar de seleccionar un punto y calcular su cuenca visual, que sería la forma más natural de proceder (según se propone en el algoritmo 5), han intercambiado los bucles para que en primer lugar se seleccione el sector, y en segundo lugar se realice el barrido de puntos (Stewart, [10]). Gracias a ello, han conseguido resultados muy sorprendentes, que no son más que una consecuencia de la elevada intensidad operacional de su algoritmo que, para un modelo digital extenso (que cubre la superficie de Andalucía, con 10x10m de precisión), y en el que el número promedio de anillos sectoriales en cada dirección es 12, resulta ser IO=257 $flops/byte$ [14].

Sin embargo, el algoritmo que proponen tiene una clara limitación: la altísima secuencialidad del bucle que hace el barrido sobre los puntos. Sin entrar en los complicados detalles de la gestión de la estructura BoS, se puede afirmar que es imposible averiguar el estado de la estructura —para un punto de observación POV— sin conocer el valor de la estructura en el punto anterior. Esto no supone un impedimento para explotar sobradamente el paralelismo en multicomputadores o multiprocesadores, ya que la propia parcelación del territorio, por un lado, y el procesamiento sectorial, por otro, proporcionan miles de hebras que pueden ejecutarse en paralelo. Sin embargo, los experimentos llevados a cabo en GPUs y Xeon-Phi, arquitecturas en las que un grano más fino es posible, no han sido tan satisfactorios [14].

También existen limitaciones a la vectorización. Si se observan los algoritmos 4 y 5, que representan el núcleo principal en el cálculo de la visibilidad, existe una operación de división que es claramente determinante para el coste de CPU. Dado que no se necesita una alta precisión para los cálculos (errores en el resultado de un 20% son perfectamente admitidos [15]), la operación de división podrían ser perfectamente sustituida por el cálculo de recíprocos con baja precisión (por ejemplo, con el juego de instrucciones AVX de intel) vectorizando el lazo del algoritmo 2.

III. Reorganización de la malla de puntos

Obsérvese la figura 3, en la que se muestra una fracción de la banda de visibilidad (BoS, según la nomenclatura de [9]) para un sector $s = 14^o - 346^o$. Para esa dirección, sólo los puntos de la malla en color oscuro son considerados en los cálculos. En algunos algoritmos de barrido sectorial, como en [5], es la distancia al eje la que determina el número de puntos de la banda, mientras que para Cervilla *et al*, es el número de puntos contenidos en la estructura. En la figura se muestran los valores para dos casos particulares para el tamaño de BoS.

El extenso estudio estadístico realizado en [14] demuestra que el tamaño de la estructura no es un factor determinante, mientras sea del orden $N^{0.5}$. Son además muchos los estudios que determinan que en la mayoría de los problemas de barrido radial (y como consecuencia de la pérdida de precisión con la distancia) el método utilizado para la interpolación de los datos del MDT al eje del sector tampoco resulta ser un factor determinante en la precisión de los resultados[1].

A. Resumen de la propuesta

Teniendo en cuenta las limitaciones expuestas, en este trabajo se propone una completa reorganización de los datos que permita la explotación de todo el paralelismo existente sin perjudicar la precisión de los resultados. En particular, el algoritmo propone:

- Utilizar el método de barrido de Stewart [10], o lo que es lo mismo, colocar el lazo que recorre los sectores en el exterior al ser éste el único modelo que garantiza la reutilización de los datos alineados en una dirección.

[1]Entre los métodos de interpolación que se han considerado en [14] están la interpolación bicúbica, el kriging (llamado así por el nombre de su creador, Daniel Krige), y el algoritmo de proximidad de Bresenham. El kriging se diferencia de otros métodos más habituales (como las interpolaciones bilineal o bicúbica) al asumir que la altitud es una variable regionalizada. Esta hipótesis supone que existe una correlación espacial entre la elevación de un lugar y la de su entorno (esencialmente, que existe un cierto patrón), y por tanto puede deducirse un valor de elevación a partir de los valores del entorno de acuerdo a unas funciones homogéneas en toda el área.

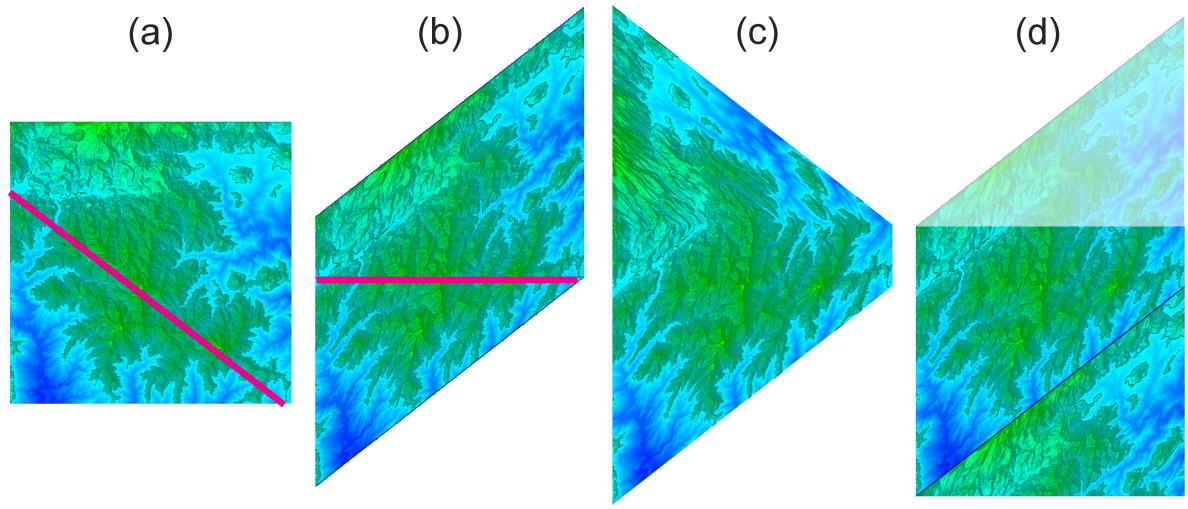

Fig. 4. Fases de redistribución de las matrices: a) Matriz original; b) Matriz reorganizada para 38°; c) Matriz compactada; y d) Matriz compactada para GPU.

- Dada una dirección del sector, construir simultáneamente todas las bandas de visibilidad que atraviesen en paralelo el MDT. Esto implica la creación de un nuevo MDT del territorio y que, por las diferentes longitudes de las bandas, tendría un aspecto más o menos sesgado.
- Simplificar el método de interpolación para el cálculo de las bandas paralelas utilizando el algoritmo de Bresenham, que se usa habitualmente para el rasterizado de líneas. Éste se elige por su velocidad, así como por la suficiente fidelidad para el problema considerado (que no requiere gran precisión en la distancia).
- Compactar la información, para que, en caso de recurrir al uso de GPUs para el procesamiento de los datos, el acceso a la estructura, así como las computaciones realizadas sean lo más regulares posibles, y reduzcan al máximo los condicionales.

El algoritmo resultante tras la reestructuración de la matriz es el siguiente:

Algoritmo 6 Cálculo de la cuenca visual total mediante reestructuración matricial de un MDT.

```
function totalviewshed(POVheight)
float VS = 0;
for s = 0, (ns/2 − 1) parallel do
  float sectorVS = 0;
  sMDT=skew(MDT, s); //rewrite MDT
  compact(sMDT);
  for i = 0, √N − 1 parallel do
    sectorVS+ =linearViewshed(i, true);
    sectorVS+ =linearViewshed(i, false);
  end for
  // Skewing and Papus theo. scaling:
  VS+ = (k ∗ (π/ns) ∗ sectorVS)
end for
endfunction
```

donde la funcion *skew* es la que redistribuye los datos del MDT en una nueva matriz, y k es una constante para cada sector, debido a la deformación del modelo digital sesgado. En el algoritmo, se ha simplificado a $\sqrt{N}$ el tamaño del lazo que recorre las filas de la estructura sesgada, aunque en la práctica este número oscila entre $dimx$ y $dimx + dimy$ (entre 0 y 45°), entre $dimx + dimy$ y $dimy$ (para el rango 45° a 90°), entre $dimy$ y $dimy + dimx$ (para el rango 90° a 135°), y entre $dimy + dimx$ y $dimx$ (para el rango 135° a 180°).

Para visualizar gráficamente el método, la figura 4 muestra la redistribución de filas y columnas de un MDT con las elevaciones correspondientes al Parque Nacional Sierra de la Nieves, en Málaga. A la izquierda (a) se muestra la matriz original de datos (supondremos en éste ejemplo que, en memoria, se almacenan contiguos los datos de una misma latitud; es decir, el lazo externo va de norte a sur, y el interno de oeste a este).

Supongamos que se ha elegido la dirección 128°-322° para el cálculo de la cuenca visual total. La figura 4(b) muestra gráficamente el modelo sesgado (representado por la estructura sMDT en el algoritmo 6). Utilizando el algoritmo de Bresenham, se han proyectados todos los segmentos paralelos de la figura 4(a) en 4(b) de forma que el tamaño de ambas estructuras (el número de elementos no nulos) es idéntico. En la nueva matriz, a diferencia de la original, todos los datos en la dirección de búsqueda están en la misma fila y, por lo tanto, el acceso a memoria es secuencial.

Tras la reescritura de la matriz, ésta puede ser compactada de dos formas: en la figura 4(c), todos los datos de la matriz sesgada se han desplazado a la primera columna. Por otro lado, en la figura 4(d), además, se han reubicado los datos en otra columna, con el objetivo de compactar aún más la información (el triángulo del territorio en color claro se ha eliminado y aparece reubicado abajo). Obsérvese que en la figura 4(d), muchos algoritmos, como TVS pueden operar sin la necesidad de condicionar la longitud de la fila.

IV. Resultados

Se han comparado los resultados obtenidos en el cálculo de la cuenca visual total entre el algoritmo de Cervilla *et al* (*tvs*) y el que se propone en este trabajo, *sdem*, del inglés *skewed digital elevation model*. Se consideran, para los cálculos, observadores situados a 1.5 metros de elevación sobre un MDT de la Sierra de las Nieves de 2000x2000 puntos y 10x10m de precisión. Se ha elegido esta zona por su representatividad, ya que contiene zonas muy llanas, y otras muy montañosas y escarpadas. Los cálculos se han ejecutado en tres sistemas multiprocesador basados en los procesadores Intel Xeon E5-2698v3, a 2.30GHz, Intel core I5-6500, a 3.20GHz, y AMD Ryzen5-2600 a 3.4GHz, con 32, 8 y 6 cores respectivamente.

En el caso del algoritmo *tvs* se ha elegido un tamaño $\sqrt{N}+1$ para la banda, de forma que la estructura BoS coincide prácticamente con la interpolación de Bresenham (véase la figura 3, derecha). Por este motivo, los resultados obtenidos con ambos algoritmos son numéricamente idénticos.

En lo que respecta al número de cores, se ha aplicado paralelismo al bucle externo (que barre los sectores) y en ambos casos la escalabilidad es lineal (como cabía esperar, ya que los sectores son completamente independientes y los cálculos asociados a los distintos sectores tienen carga muy equilibrada, por motivos estadísticos).

Respecto al tiempo empleado por cada core en cada sector, que al fin y al cabo es el parámetro que realmente mide la calidad y eficiencia de los algoritmos propuestos, las tres arquitecturas ofrecen tiempos de ejecución que rondan los 60 segundos para el algoritmo *tvs*, de los que aproximadamente el 10% se emplea en el reordenamiento de puntos. Por su parte, el algoritmo *sdem* requiere menos de un segundo para redistribuir la matriz, y reduce en más de un 25% el tiempo de cálculo del sector (bajando hasta 30-35 segundos, según la arquitectura), debido sobre todo a la regularidad del lazo que recorre la banda de visibilidad.

V. Conclusiones

Los resultados obtenidos en este trabajo demuestran que, en algoritmos que trabajan sobre MDT (o en general, modelos bidimensionales o tridimensionales basados en mallas cartesianas) utilizando un barrido radial, la reestructuración de los datos de entrada mediante interpolación de los mismos en la dirección del sector de análisis, un procedimiento cuya complejidad es $O(s \cdot N)$, es una opción altamente recomendable cuando el problema a resolver sea de complejidad superior. El motivo principal no es solo que el alineamiento de datos en memoria reduce los fallos de cache, sino que la preparación de los datos permite que las computaciones de los segmentos paralelos del sector se puedan ejecutar en paralelo.

Además, el número de hilos que es posible crear es proporcional a $\sqrt{N}$, por lo que son claros candidatos para su ejecución en arquitecturas *manycore*.

Como consecuencia de ello, en futuros trabajos se implementará una versión heterogénea, en la que los distintos sectores se ofrezcan a modo de granja de tareas sobre las que CPUs y GPUs seleccionen su trabajo usando un algoritmo de asignación por *work-stealing* o similar.

Agradecimientos

Este trabajo ha sido cofinanciado por el Ministerio de Educación y Formación Profesional, a través del Proyecto TIN2016-80920-R; por la Junta de Andalucía, a través del Proyecto de Excelencia TIC-8260 y por la Universidad de Málaga, a través del I Plan Propio Integral de Docencia y el Proyecto *usmartdrive* del I Programa Smart-Campus.

Referencias

[1] Wm Randolph Franklin and Clark Ray, "Higher isn't necessarily better: Visibility algorithms and experiments," in *Advances in GIS research: sixth international symposium on spatial data handling*. Taylor & Francis Edinburgh, 1994, vol. 2, pp. 751–770.

[2] ESRI, *ARCGIS Software. Version 9.3*, Environmental Systems Research Institute, 2010.

[3] M. Neteler, M.H. Bowman, M. Landa, and M. Metz, "GRASS GIS: a multi-purpose Open Source GIS," *Environmental Modelling & Software*, vol. 31, pp. 124–130, 2012.

[4] Mikhail J Atallah, "Dynamic computational geometry," in *Foundations of Computer Science, 1983., 24th Annual Symposium on*. IEEE, 1983, pp. 92–99.

[5] Leila De Floriani and Paola Magillo, "Visibility algorithms on triangulated digital terrain models," *International Journal of Geographical Information Systems*, vol. 8, no. 1, pp. 13–41, 1994.

[6] Brian Cabral, Nelson Max, and Rebecca Springmeyer, "Bidirectional reflection functions from surface bump maps," in *ACM SIGGRAPH Computer Graphics*. ACM, 1987, pp. 273–281.

[7] David R Miller, Neil A Brooker, and Alistair NR Law, "The calculation of a visibility census for scotland," in *Proceedings of the ESRI annual conference*, 1995.

[8] David Miller, "A method for estimating changes in the visibility of land cover," *Landscape and Urban Planning*, vol. 54, no. 1, pp. 93–106, 2001.

[9] AR Cervilla, S Tabik, EL Zapata, and LF Romero, "Visibilidad total para observadores desligados del terreno," *Jornadas Sarteco*, 2012.

[10] A James Stewart, "Fast horizon computation at all points of a terrain with visibility and shading applications," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 4, no. 1, pp. 82–93, 1998.

[11] S. Tabik, A.R. Cervilla, E.L. Zapata, and L.F. Romero, "Efficient data structure and highly scalable algorithm for total-viewshed computation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 304–310, 2015.

[12] A.R. Cervilla, S. Tabik, J. Vias, M. Merida, and L.F. Romero, "Total 3d-viewshed map: Quantifying the visible volume in digital elevation models," *Transactions in GIS*, 2016.

[13] A.R. Cervilla, S. Tabik, and L.F. Romero, "Siting multiple observers for maximum coverage: An accurate approach," in *Proceedings of the 2015 International Conference on Computational Science, ICCS2015*, 2015.

[14] A. Cervilla, *Algoritmos Multicore para el Cálculo de Parámetros de Visibilidad en Sistemas de Información Geográfica*, Tesis Doctoral. Universidad de Málaga, 2019.

[15] Siham Tabik, Emilio Zapata, and Luis F. Romero, "Simultaneous computation of total viewshed on large high resolution grids," *International Journal of Geographical Information Science*, vol. 27, no. 4, pp. 804–814, 2013.