Visibilidad total para observadores desligados del terreno

A. R. Cervilla, S. Tabik, E. L. Zapata y L. F. Romero[1]

Resumen— **La cuenca visual de un punto situado en un territorio es la extensión del propio terreno que es visible desde dicho punto, teniendo en cuenta exclusivamente la propia orografía. El cálculo de cuencas visuales en Modelo Digital de Elevaciones (MDE) tiene un coste computacional muy elevado, ya que, a priori, cualquier de los N puntos del territorio es un obstáculo potencial para la visión directa entre otros dos puntos. Existen en la literatura numerosos algoritmos que permiten el cálculo de una cuenca visual en pocos segundos, pero a pesar de ello, ninguno de estos algoritmos se podría utilizar para el cálculo de las N cuencas visibles desde todos los posibles puntos de observación (cuenca visual total, CVT), ya que el coste computacional sería inmenso. En este trabajo presentamos el primer algoritmo que permite el cálculo de la CVT de un terreno de grandes dimensiones en tiempo razonable. A diferencia de otras aproximaciones al problema, el algoritmo no tiene restricciones, por lo que puede utilizarse para la elaboración de mapas de visibilidad de utilidad real.**

Palabras clave— **Cuenca visual. Visibilidad total. Listas enlazadas.**

I. Introducción

UN conocimiento preciso de la visibilidad en un territorio proporcionaría una herramienta de gran interés para diversas aplicaciones actuales en telecomunicaciones, planificación del medio natural, ecología, turismo, arqueología, etc. Así, se simplificaría la elaboración de algoritmos que requieren datos de visibilidad. Por ejemplo, el conocido problema de telecomunicaciones que busca dar una máxima cobertura visual a un territorio utilizando el menor número de puntos de observación [2], [7].

Se conoce como Cuenca Visual (CV) de un observador del terreno al área del terreno visible desde un determinado punto que, normalmente, se encuentra situado a una altura h sobre el terreno. Existen numerosos algoritmos que calculan la CV desde un único punto de vista o, como mucho, desde un número muy reducido de puntos de observación. En este contexto, la mayoría de los programas GIS, como ArcGIS 10 [6] o GRASS GIS 6.4 [8] incorporan módulos para el cálculo de la cuenca visual. Junto a ellos, existen numerosos algoritmos en la literatura con el mismo propósito [1], [4], [9], [5], [3], [14], [15]

Sin embargo, existen muy pocos algoritmos capaces de calcular la Cuenca Visual Total (CVT). En particular, existe una primera clase de metodos de CVT que utiliza mecanismos de simplificación de las CV, por lo que los resultados son igualmente, de calidad muy reducida, y en consecuencia, de poca utilidad. Es el caso del algoritmo de Llobera *et al* ([10]), en el que se limita el radio de cálculo, y el número de puntos. El resto de algoritmos de CVT en realidad se basan en el calculo de CV a ciertos punto.

Normalmente, estos métodos determinan la cuenca visual de un punto partir de un MDE del terreno, que a su vez está modelado por N puntos del territorio para los que se conoce su ubicación exacta en latitud, longitud y elevación. En el cálculo de la cuenca visual en un MDE, cualquier punto del terreno puede ser, a priori, un obstáculo para la visión de cualquier otro punto, y en consecuencia, un algoritmo ingenuo tendría un coste del orden $O(N^2)$.

A pesar de que existen algoritmos muy eficientes que reducen significativamente el coste del cálculo de una cuenca visual, realizar el cómputo de la totalidad de las cuencas visuales de un territorio [11]) tendría un coste mucho más elevado. De hecho, en [10] se califica como imposible el cálculo de la CVT. En particular, este coste sería superior a $O(N^2)$ si se procesa cada cuenca visual de forma independiente utilizando los algoritmos existentes para cuencas visuales, ya que todos tienen un coste superior a $O(N)$.

Por tanto, un algoritmo asumible debería, de alguna manera, considerar el problema globalmente. Dicho de otra forma, un elevado porcentaje de los cómputos realizados para el cálculo de la cuenca visual desde una determinada ubicación debe ser aprovechado para el cálculo de otras cuencas visuales, permitiendo una escalabilidad inferior a N. Esta aproximación ya fue utilizada en [18], donde se utiliza la propiedad que determina que si un punto $P1$ pertenece a la frontera de una región de visibilidad de P, entonces P pertenece a la frontera de una región de visibilidad de $P1$. Este algoritmo permite una reducción significativa del cálculo, aunque sólo se puede aprovechar si los observadores son solidarios (están ligados) al suelo (h=0).

Este trabajo presenta un algoritmo muy eficiente para el calculo de CVT desde observadores desligados. Las ideas claves son: i) utilizar una estructura de datos compacta y eficiente y ii) reducir el numero de operaciones aritméticas y accesos a memoria, en primer lugar, mediante una reducción de los puntos involucrados en los cálculos y mediante la extracción de una muestra estadística y, en segundo lugar, mediante la reutilización de la información.

II. El Modelo

El primer paso para simplificar los cálculo consiste en que, para cada punto de observación P, la cuenca visual sólo tiene en consideración los puntos que se encuentran en un número discreto S de líneas de observación (*los, line of sights*) que partiendo de P, se irradian al exterior en S direcciones privilegiadas (Figura 3). Cada línea de observación se co-

[1]Dpto. Arq. Comput., Univ. Málaga, felipe@uma.es

rresponde con un azimuth $\vec{s}$ de visión diferente. En la mayoría de los cálculos se utilizan $S = 360$ líneas, separadas exactamente por un grado. El segmento de observación s es, por tanto, considerado como una muestra estadística de los puntos comprendidos en el sector angular $s - 0.5, s + 0.5$. Por otro lado, y para mejorar la muestra, estos segmentos de observación incorporan a todos los puntos del MDE que se encuentran a una pequeña distancia del segmento, *i. e.*, tienen un cierto grosor, y se denominan bandas de observación $band(P, s)$

Cada banda de observación $band(P, s)$, junto a la que se encuentra en dirección opuesta $band(P, s^o)$, $s^o = s + 180$, constituyen el perfil del punto de observación en la dirección s: $prof(P, s)$. En un perfil se encuentra la información necesaria para calcular la visibilidad dentro de cada banda de observación. Debido al carácter prácticamente unidimensional del perfil de observación, las zonas de visibilidad se pueden representar como segmentos o tramos visibles. En la Figura 4, el punto P, desde una altura h, tiene dos segmentos visibles: $[P, P1]$ y $[P2, P3]$. Estadísticamente, cada uno de estos segmentos visibles representa a un anillo sectorial. Cada anillo sectorial visible está delimitado por un radio inferior (srs, *start of ring sector*) y un radio superior (ers, *end of ring sector*), siendo $nrs(P, s)$ el número total de anillos visibles en la banda s. La cuenca visual del punto P se calcula según la Fig. 1:

```
vs(P) = 0
for s = 0 to S − 1 do
   for i = 0 to nrs(P, s) − 1 do
      vs(P)+ = RSsurface(srs(P, s, i), ers(P, s, i))
   end for
   for i = 0 to nrs(P, s′) − 1 do
      vs(P)+ = RSsurface(srs(P, s, i), ers(P, s, i))
   end for
end for
```

Fig. 1. Cálculo de la Cuenca Visual

El cálculo de los anillos sectoriales se obtiene a partir de los perfiles (Algoritmo de la Fig. 2).

```
for s = 0 to S − 1 do
   for P = 0 to N do
      //Aquí, cada iteración calcula CV para un P
      create(band(P, s))
      rs(P, s) = computeForward(band(P, s))
      rs(P, s′) = computeBackward(band(P, s′))
   end for
end for
```

Fig. 2. Cálculo de Anillos Sectoriales

La eficiencia computacional del algoritmo propuesto se soporta en tres ejes. El primero de ellos es el propio muestreo estadístico mediante el cual las bandas de observación representan simplificadamente a los sectores. Mediante esta simplificación, que no es más que una variante de la técnica ray-tracing, conseguimos reducir la complejidad en el cálculo de CV de $O(N)$ a $O(S \cdot N^{1/2})$. La idoneidad del muestreo es fácilmente demostrable, ya que, en primer lugar, la rugosidad del suelo es prácticamente aleatoria en la inmensa mayoría de los terrenos, y

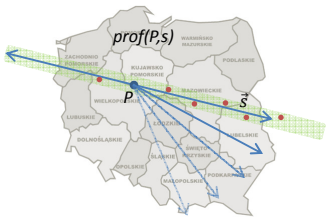

Fig. 3. La visibilidad en un punto se calcula en un conjunto discreto de S direcciones preferentes. Para cada punto P y dirección preferente $\vec{s}$, existe un perfil $prof(P, s)$ constituido por los puntos más próximos al segmento

en segundo lugar, la muestra es más abundante en las proximidades de punto de observación, donde precisamente es más necesaria la exactitud de los cálculos.

Los otros dos ejes de la eficiencia del algoritmo son, en primer lugar, el bajísimo coste de generación de la estructura de datos $prof(P, s)$, y en segundo lugar, la simplicidad del algoritmo de análisis de las bandas de observación (*computeForward/Backward*):

A. Perfil de P: ventana de puntos activos

Para implementar las bandas de observación podemos hacer dos suposiciones. La primera de ellas es que las bandas tengan un grosor fijo (en metros), y la segunda, que contenga un número fijo de puntos de observación. Aun siendo muy similares, y sin entrar en detalles, la primera opción es estadísticamente más homogénea, mientras que la segunda opción es computacionalmente más eficiente. En este trabajo se ha optado por la segunda opción, y tras exponerla, analizaremos las ventajas de nuestra decisión.

El perfil $prof(P, s)$ se implementará mediante una estructura de datos que contiene toda la información necesaria de los puntos que almacena. El perfil se caracteriza, en primer lugar, por un eje central que atraviesa uno y sólo uno de los puntos [1]. Dicho punto P se denomina punto de observación, o *point of view,* PoV para el cual se calcularan los segmentos de visibilidad. En la estructura, P tendrá, a cada lado, HW puntos. El número de puntos, BW= *2·HW+1 (Bandwidth)*, que determina el grosor de la banda, y la distancia de cada punto a PoV también son características de la banda de observación. Estos dos parámetros están íntimamente relacionados. Si la banda es muy estrecha, es decir, el número de puntos es reducido, la distancia entre dos puntos se puede aproximar por su diferencia de coordenadas en el eje s. Si la banda es muy ancha, dicha aproximación no será correcta, especialmente para los puntos más próximos. En la sección III se analiza este aspecto. En la mayoría de los cálculos realizados en este trabajo, N es 2000×2000, siendo $BW = 2001$, con $HW = 1000$ puntos a cada lado de P.

[1]En este trabajo se han utilizado MDEs caracterizados por mallas regulares cuadradas, aunque los resultados son igualmente válidos para otros tipos de mallas. En [17] se demuestra que para un conjunto S de direcciones privilegiadas elegido adecuadamente, no hay dos puntos que pertenezcan a un mismo segmento

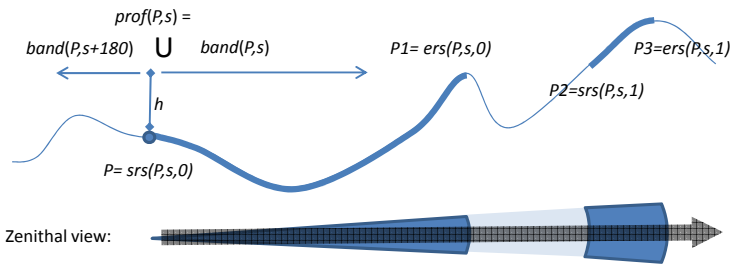

Fig. 4. Arriba: Segmentos de visibilidad para un observador en P desligado del terreno a una altura h. Los segmentos de visibilidad ([P,P1] y [P2,P3]) en la banda de observación s representan a dos sectores radiales. En la vista cenital, la banda es representada por una flecha oscura de grosor no nulo, que muestrea estadísticamente a una región angular.

Dado un sector s, y con el objeto de simplificar la generación de $prof(P, s)$, hemos utilizado una estrategia que consiste en calcular la CV de cada punto del territorio siguiendo el orden de la coordenada $s^{\perp}$ (la dirección perpendicular a s). De esta forma, el conjunto $prof(P, s)$ es prácticamente idéntico al del punto procesado previamente (en cada iteración, la estructura gana un punto por el lado $+s^T$ y pierde otro en el lado $-s^T$). En particular, la estrategia que se propone afecta al orden en el que se procesan las iteraciones del segundo bucle del Algoritmo 2. En nuestros cálculos, s=0 se corresponde con un azimuth inicial 90.001º (dirección este), por lo que las bandas s y s^o se refieren a las observaciones hacia poniente y levante desde cualquier punto. El diferencial 0.001 es el que permite que no haya nunca dos o más puntos en un mismo segmento. Para ese mismo sector, los puntos son procesados de norte a sur (prácticamente), empezando por el nodo situado en el extremo noreste.

El procesamiento en este orden, y para un tamaño fijo del perfil nos garantiza que:

- los puntos incluidos en la ventana activa son consecutivos
- en cada perfil, sólo un nodo difiere respecto al anterior

Entre los BW puntos de la banda, hay cinco puntos especiales o nodos *centinela*: el punto central, P, más los cuatro puntos que delimitan la propia banda: (Figura 8):

- P El punto central de la banda, i.e., PoV.
- $PO = P - HW$ Es el punto más antiguo dentro de la banda y el de menor índice $s^{\perp}$ o menor coordenada $s^{\perp}$
- $PN = P{+}HW$ Punto más reciente, con mayor índice $s^{\perp}$
- PL Es el punto más alejado (geográficamente) de PoV en la dirección s
- PF Es el punto más alejado de PoV en la dirección s^o

Según el ordenamiento propuesto, el algoritmo para el cálculo del perfil de un punto $Prof(P, s)$ debe calcular el índice de PoV, eliminando y añadiendo un nuevo nodo para mantener constante BW:

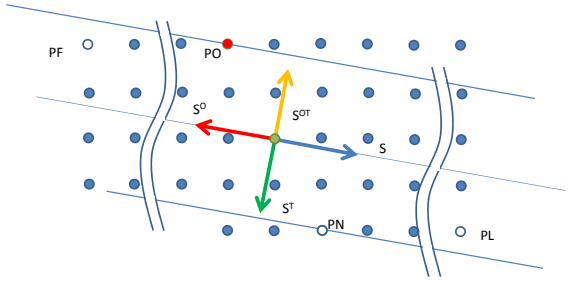

Fig. 5. Direcciones de ordenamiento de los puntos y nodos centinela de la banda de observación

function: create(prof(P,s))
  remove $P - HW$
  insert $P + HW + 1$
  $P = P + 1$ //Update PoV in order $s^{\perp}$
  $identify(PL, PF)$

Fig. 6. Cálculo del perfil

El algoritmo 6 sólo es válido si se conoce el perfil del punto anterior. Aunque esta es la situación más frecuente, es necesario inicializar la ventana con los primeros BW *puntos*. En este trabajo hemos utilizado una estrategia simple que consiste en que durante las primeras HW iteraciones, la función *create* añade dos puntos por iteración, sin eliminar ninguno, en una etapa de transición inicial, mientras que en las últimas HW iteraciones, sólo se eliminan puntos sin añadir nuevos. El bucle interno del algoritmo 2 queda así:

for $P = 0$ to N **in order** $s^{\perp}$ **do**
  if $P < HW$ **then**
    $create(band(P, s), init_phase)$
  else if $P >= N - HW - 1$ **then**
    $create(band(P, s), clos_phase)$
  else
    $create(band(P, s), stationary)$
  end if
  $rs(P, s) = computeForward(band(P, s))$
  $rs(P, s^o) = computeBackward(band(P, s^o))$
end for

Fig. 7. Cálculo de la visibilidad en un sector

B. *Procesamiento del perfil: cálculo de los anillos sectoriales*

Mientras que el recorrido de los puntos se realiza en la dirección $s^{\perp}$, el cálculo de los segmentos

visibles dentro del perfil $prof(P, s)$ se debe realizar en las direcciones s y s^o para las bandas de observación $band(P, s)$ y $band(P, s^o)$ respectivamente. La organización interna de los datos dentro del perfil es un aspecto crítico a tener en cuenta, y en el que no podemos dejar aparte la arquitectura del computador para alcanzar un rendimiento razonable.

Para este doble ordenamiento de los puntos existe una estructura de datos muy apropiada que permite el análisis de los elementos en ordenamientos no correlacionados: la lista doblemente enlazada mediante una matriz de nodos. En las siguientes secciones se analizan dos opciones para la organización interna de la lista.

B.1 Los nodos se ordenan en memoria en orden $s^{\perp}$

En esta opción, la inserción de un nuevo punto P+HW y la eliminación de P-HW-1 se realiza en un solo paso, ya que basta convertir la matriz de nodos en una lista circular, con un puntero cíclico indicando la posición de la cabecera de la lista, que simultáneamente es la propia cola. Por contra, el cálculo del perfil necesita recorrer los enlaces desde P a PL en la dirección s, y desde P a PF en la dirección s^o, y en sentido negativo desde P hasta el otro extremo (PF). En este caso, el lazo interno del algoritmo accede a memoria de forma desordenada, utilizando los enlaces de la lista enlazada. Cada nodo de la lista es una estructura (*struct node*)constituida por:

- *node.id*, identificador del punto
- *node.d*, coordenada en la dirección s
- *node.h*, elevación del punto
- *node.os*, número de orden en la dirección S
- *node.next*, Nodo más próximo en sentido S
- *node.prev*, Nodo más próximo en sentido S^o

Uno de los inconvenientes de este sistema es que la inserción de nodos en la función $create(prof)$ necesita conocer en qué lugar de la lista enlazada se produce la inserción. Para ello, es necesario recorrer la lista para encontrar la ubicación exacta, utilizando el parámetro *node.os*. Aunque dicha búsqueda se puede realizar en el propio bucle del cálculo de los perfiles (rutinas $computeForward$ y $computeBackward$ del algoritmo 2), en este trabajo se ha realizado una importante optimización que consiste en precalcular la ubicación de los nodos insertados dentro de la lista. Dicho lugar de inserción se declara como un nuevo nodo centinela a añadir a los 5 existentes (PI, o punto de inserción). Los N puntos de inserción precalculados se guardan en un archivo ya que, cualquier otro territorio que utilice una malla de estructura y dimensiones idénticas puede reutilizar dicha información.

Así, en este trabajo, se han calculado las CVT al MDE de Andalucía compuesto de 416 hojas. Todos las hojas tienen las mismas dimensiones (2000×2000) por lo que los 4000000 índices de los puntos PI son idénticos, y sólo fue necesario el cálculo en una hoja.

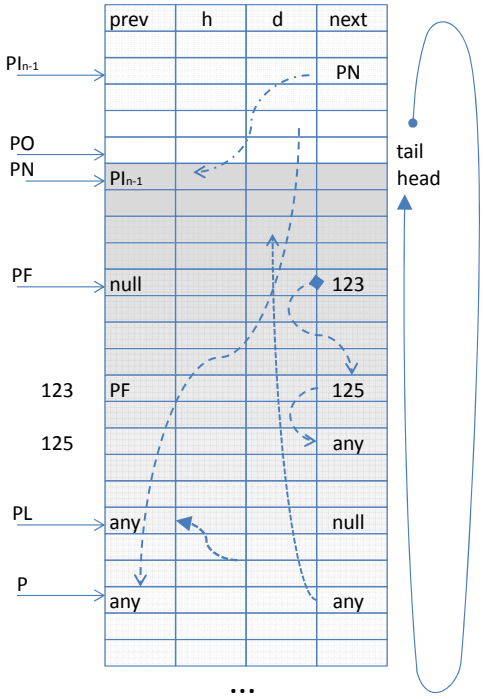

Fig. 8. En la ventana de puntos activos de esta figura, los puntos están ordenados según su posición en memoria desde el punto más antiguo, de inferior coordenada $s^{\perp}$ (PO), con fondo claro, hasta el más reciente (PN) con fondo oscuro. El punto P se encuentra exactamente en el centro de la estructura circular. Por otra parte, el doble enlazado (a través de los punteros *prev* y *next*), mantiene un ordenamiento de los nodos desde PF hasta PL. El primer tramo, de PF a P, es la banda $s^{\perp}$: $band(P, s^T)$ (línea de rayas). El segundo tramo, con línea de puntos, recorre la banda en sentido s: $band(P, s)$. A priori, PO, PN, y el punto de inserción, PI pueden estar en cualquier lado, s o $s^{\perp}$, del punto P.

Más aún, una vez conocido el punto de inserción, es innecesario el almacenamiento de los parámetros *node.os* y *node.id*, por lo que la estructura de datos se simplifica, almacenando exclusivamente los punteros de la lista enlazada (short integer), la elevación h (unsigned short, para modelos de precisión 0.1 metros o superior), y la coordenada *node.d* (float). Ello permite almacenar toda la información (aproximadamente 20KB) en la L1 de datos de la mayoría de los procesadores.

Finalmente, es necesario indicar que la eficiencia de este algoritmo se basa en la estabilidad de la estructura de datos, que hace que la dimensión de la ventana de observación sea constante, independientemente del punto analizado como del sector. Esto es debido a que se optó por una estructura de tamaño constante en lugar de una banda de ancho fijo, que hubiera introducido una ligera irregularidad prácticamente inapreciable en los resultados numéricos, pero muy significativos en coste computacional.

B.2 Los nodos se ordenan en memoria en orden S

En este caso, el bucle interno (rutinas $computeForward$ y $computeBackward$ del algoritmo 2), responsable de casi la totalidad del

tiempo de ejecución, accede de forma lineal a memoria, mientras que la inserción y eliminación de nuevos elementos implicaría una reubicación completa de la información en la memoria.

Los experimentos realizados demuestran que esta organización es claramente menos eficiente, incluso en arquitecturas con cache de datos L1 inferior al tamaño del perfil, debido a que el beneficio en el acceso a memoria en en lazo interno se diluye completamente en las dos estructuras propuestas debido a la operación de división en el núcleo del algoritmo (Fig. 9), necesaria para el cálculo de la tangente. Por esta razón, la actualización del perfil, mucho más costosa en este segundo modelo, es determinante. En lo sucesivo, se utilizará la estructura de datos descrita en la Sección II-B.1.

C. Algoritmo de cálculo de los anillos sectoriales

El algoritmo de cálculo de los anillos sectoriales entre P y PL queda de la siguiente forma:

```
function: computeForward(band(P,s))
  d0 = P.d;
  h0 = P.h;
  bool visible = false
  float max = -∞ // Max angle
  currN = P
  while currN! = PL do
    currN = currN.next;
    kernel(d0, h0, currN.d, currN.h, &max, &visible)
  end while

function: kernel(d0, h0, d, h, &max, &visible)
  float angle = (h − h0))/(d − d0);
  if angle > max then
    if !visible then
      store_srs(d)
    end if
    visible = true;
    angle = max;
  else
    if visible then
      store_ers(d)
      nrs[P, s] + +;
    end if
    visible = false
  end if
```

Fig. 9. Cálculo de la cuenca visual de un punto en el sector s

En el sentido opuesto, s^o, los cálculos son similares, aunque la lista enlazada se recorre en sentido contrario, desde P hasta PF, utilizando los enlaces $node.prev$.

Por otra parte, el algoritmo implementado finalmente realiza un tratamiento diferenciado de los cuatro puntos más próximos a P en cada una de las bandas, ya que por su proximidad, y debido a la desviación relativa respecto al eje de la banda, pueden afectar mucho a la calidad de los resultados finales. En el algoritmo 9, afecta especialmente a los valores iniciales, como por ejemplo a los valores iniciales de $currN$ y el ángulo máximo max.

III. Resultados

A. Calidad de los resultados

Los resultados obtenidos en el cálculo de la CV puntual han sido comparados en 30 ubicaciones diferentes con los de las dos aplicaciones más utilizadas en los sistemas de información geográfica, ArcGIS y GRASS, así como con el único modelo de CVT existente en la literatura. Por las limitaciones de éste último, la comparación se ha realizado con observadores ligados al suelo ($h = 0m$). Los datos obtenidos demuestran una gran semejanza respecto a ArcGIS, que en todo caso es inferior al 5%, lo cual es más que suficiente dada la inestabilidad numérica del problema subyacente (ligerísimas variaciones en la posición del *pov* pueden provocar inmensas diferencias en el tamaño de la cuenca visual). En este problema, usualmente, se consideran aceptables los resultados del mismo orden de magnitud [12]. La Figura 10 muestra además cómo las CV calculadas con nuestro algoritmo son muy similares a las de los otros tres modelos en un determinado punto del MDE de Andalucía.

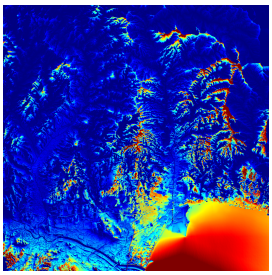

Fig. 11. CVT de la Hoja 1113A (MDE Andalucía), a $10 \times 10m$. Coordenadas UTM (zona 30S) X=360000, Y=4080000.

Estas semejanzas, que se cumplen en todo tipo de ubicaciones, e incluso con observadores no ligados al suelo (en las aplicaciones que lo permiten) garantizan la validez de los resultados obtenidos, y en consecuencia, del Mapa de Visibilidad calculado.

Por otra parte, los experimentos realizados con tamaños diferentes para la ventana de puntos activas (BW) en el rango 50% a 200% respecto a $\sqrt{N}$ apenas afectan a la calidad de resultados. A partir de valores fuera de este rango, se produce una clara degradación de los mismos. Un estudio más profundo puede determinar cuál es el tamaño óptimo en función del ángulo, aunque para ello habría que disponer de datos reales de visibilidad, lo que actualmente es inviable [13]. Algo parecido ocurre con otros parámetros como S, el número de sectores. Un rango de valores entre 90 y 360 es más que suficiente para garantizar la fiabilidad de los resultados sin penalizar excesivamente el rendimiento. Hay que tener en cuenta que el coste de cálculo es directamente proporcional a S aunque, por otra parte, la escalabilidad con este parámetro es perfecta.

La Figura 11 muestra un Mapa de Visibilidad para una hoja del MDE de Andalucía a 10m. Este mapa de visibilidad está elaborado para un observador situado a 2 metros y es el primer mapa de visibilidad elaborado para un territorio sin restricciones respecto a la ubicación del observador y sin radio límite de observación. Con una escala de colores, apreciable en la versión online de este trabajo, se muestra la visibilidad de los puntos en un rango 0–200 km^2.

(a) (b) (c) (d)

Fig. 10. Cuenca visual del punto UTM (zone 30S) coordenadas, X=368800 Y=4070140, en un MDE de $10 \times 10m$, según (a) El modelo propuesto en este trabajo, (b) el modelo propuesto en [32] (c) r.los bajo GRASS y, (d) Viewshed tool bajo ArcGIS.

B. Tiempos de ejecución

El cálculo de la CV de un sólo punto en un MDE de 2000×2000 puntos, usando el algoritmo propuesto es 0.0032 segundos, claramente inferior a los resultados de los trabajos comparados, que son de 0.0063 segundos para [18], mientras que Viewshed tool y GRASS necesitaron 10 y 155 segundos respectivamente, bajo las mismas condiciones. Estos resultados demuestran, por un lado, que los algoritmos existentes para el cálculo de la CV son ineficaces en el cálculo de CVT; y por otro lado, que el algoritmo propuesto en este trabajo es claramente más eficiente que el único algoritmo existente para CVT, sin las restricciones de éste.

IV. Conclusiones

Este trabajo presenta una importante contribución para los Sistemas de Información Geográfica consistente en la elaboración del primer Mapa de Visibilidad Total para observadores no ligados al suelo. Para su elaboración, ha sido necesario el desarrollo de un algoritmo que aproveche al máximo la localidad en el acceso a la información, para solventar así los problemas de complejidad inherentes a los cálculos que requiere el problema a resolver.

El elemento más importante del algoritmo es una estructura de datos compacta que consiste en una lista enlazada de tamaño fijo y reducido espacio en memoria. La alta eficiencia del algoritmo de generación de esta estructura, así como el del utilizado para el análisis de la misma no solo es la clave del rendimiento del modelo propuesto, sino que además pueden ser aprovechados en problemas computacionales muy diferentes al propuesto en este trabajo, pero en los que, igualmente, mallas regulares son barridas en ángulos diferentes a los ejes del propio mallado. Sirvan como ejemplos los frentes de onda (*wavefronts*) utilizados en la codificación de video h264 o los empleados en los algoritmos de reconstrucción por tomografía axial.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación, a través del Programa Juan de la Cierva y del Proyecto TIN2010-16144, y por la Junta de Andalucía, a través del Proyecto de Excelencia TIC-8260.

Referencias

[1] Atallah M., 1983. Dynamic computational geometry. Proceeding of the 24th IEEE Symposium on the Foundations of Computer science, 92–99.

[2] Ben-Shimo Y., Ben-Moshe B., Ben-Yehezkel Y., Dvir A. and Segal M., 2007. Automated antenna positioning algorithms for wireless fixed-access networks. Journal of Heuristics, 13(3), 243–263.

[3] Cabral B., Max N., and Springmeyer R., 1987. Bidirectional reflection functions from surface bump maps. ACM SIGGRAPH Computer Graphics 21, 273–281.

[4] De Floriani L., and Magillo P., 1994. Visibility Algorithms on Triangulated Terrain Models. International Journal of Geographic Information Systems, 8(1), 13–41.

[5] De Floriani L., Falcidieno B. , Nagy G. and Pienovi C. , 1989. Polyhedral Terrain Description Using Visibility Criteria, Institute for Applied Mathematics, National Resemvh Council, Technical Report (17).

[6] ESRI(Environmental Systems Research Institute), 2010. ARCGIS Software. Version 9.3.

[7] Franklin R. and Ray C. K., 1994. Higher isn't necessarily better: Visibility Algorithms and Experiments. Advances in GIS Research: Sixth International symposium on spatial Data handling, 751–770.

[8] GRASS Development Team, 2011. Geographic Resources Analysis Support System (GRASS) Software, Version 6.4.1. Open Source Geospatial Foundation. http://grass.osgeo.org. [Accessed 28/05/2012]

[9] Hershberger J., 1989. Finding the upper envelope of n line segments in O(n log n) time, Information Processing Letters, 33(4), 169–174.

[10] Llobera, M., 2003. Extending GIS-based visual analysis: the concept of visualscapes. International Journal of Geographical Information Science , 17(1), 25–48.

[11] Llobera, M., Wheatley, D., Steele, J., Cox, S. and O. Parchment, 2010. Calculating the inherent visual structure of a landscape (total viewshed) using high-throughput computing. In Niccolucci, F. & Hermon, S. (Eds.) Beyond the artifact: Digital Interpretation of the Past, CAA 04. Budapest: Archaeolingua, 146–151.

[12] Maloy M. A. and Dean D. J., 2001. An accuracy assessment of various GIS-based viewshed delineation techniques. Photogrammetric Engineering and Remote Sensing, 67(11), 1293–1298.

[13] Wechsler S. P. and Kroll C. N., 2006. Quantifying DEM Uncertainty and its Effect on Topographic Parameters. Photogrammetric Engineering and Remote Sensing, 72(9), 1081–109.

[14] Miller, D. 2001. A Method for Estimating Changes in the Visibility of Land Cover. Landscape and Urban Planning, 54(1), 93–106.

[15] Miller, D.R. Brooker, N.A. and Law, A.N.R., 1995. The Calculation of a Visibility Census for Scotland. In: Proceedings of the ESRI Annual Conference, May 1995, Redlands. CA. USA. [online]. Available from: http://bit.ly/tcvcs [Accessed 28/05/2012]

[16] Tabik S., Romero L. F., http://bit.ly/ProfMA, (Video).

[17] Tabik S., Romero L. F., and Zapata E. L., 2011. High Performance Three-horizon Composition Algorithm for large scale terrains. International Journal of Geographical Information Science, 25(4), 541-555.

[18] Tabik S., Romero L. F., and Zapata E. L., 2011. Simultaneous Computation of Total-Viewshed on Large High Resolution Grids. International Journal of Geographical Information Science, In press.