UNIVERSIDAD
DE MÁLAGA

Depto. de Arquitectura de Computadores

TESIS DOCTORAL

Algoritmos Multicore para el Cálculo de Parámetros de Visibilidad en Sistemas de Información Geográfica

Autor: Antonio Manuel Rodriguez Cervilla

Octubre 2018

Dirigida por:
Luis Felipe Romero Gómez,
Siham Tabik.

AUTOR: Antonio Manuel Rodríguez Cervilla

http://orcid.org/0000-0002-5035-964X

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

Dr. D. Luis Felipe Romero Gómez.
Catedrático del Depto. de Arquitectura de Computadores, Universidad de Málaga.

Dra. Dña. Siham Tabik.
Investigadora Ramón y Cajal Depto. de Ciencias de La Computación e Inteligencia Artificial, Universidad de Granada.

CERTIFICAN:

Que la memoria titulada "Algoritmos Multicore para el Cálculo de Parámetros de Visibilidad en Sistemas de Información Geográfica", ha sido realizada por D. Antonio Manuel Rodriguez Cervilla bajo nuestra dirección en el Depto. de Arquitectura de Computadores de la Universidad de Málaga y constituye la Tesis que presenta para optar al grado de Doctor en Ingeniería Mecatrónica.

Málaga, a Septiembre de 2018

Dr. D. Luis Felipe Romero Gómez.
Codirector de la tesis.

Dra. Dña.Siham Tabik.
Codirectora de la tesis.

A mi familia y amigos

Resumen

La mayoría de los algoritmos de visibilidad en Sistemas de Información Geográfica (SIG) se basan en el cálculo y análisis de las cuencas visuales desde un punto o a lo sumo desde un conjunto limitado de puntos. Dado un Modelo Digital de Elevaciones (MDE) y un observador situado en el modelo, su Cuenca Visual es la superficie del propio modelo que es visible desde el observador. La información proporcionada por los mapas de Cuencas Visuales (para un único observador) suele ser una representación bidimensional del terreno visible y es una información muy pobre y de poca utilidad, ya que sólo son válidos para un único punto de vista.

El cálculo del conjunto de Cuencas Visuales para todos los posibles observadores situados en el territorio proporcionaría una información mucho más valiosa, ya que permitiría determinar fácilmente la visibilidad de un observador que se desplaza (por ejemplo, un senderista), la superficie que se puede ver desde toda una región (por ejemplo, un lugar pintoresco), e incluso la superficie desde la que no se puede ver una parcela (por ejemplo, un vertedero). Además, dicha información sería más valiosa aún si los datos calculados tuvieran un carácter tridimensional (Volumen Visible) que realzaran la calidad de la cuenca visual. Es bien conocido que los puntos con mayor cuenca visual pueden no tener el mayor volumen visible; un índice de este tipo sería muy útil para muchas aplicaciones (turismo, urbanismo, estudios de hidrografía...). Además el conocimiento de donde posicionar un conjunto de observadores, para que estos cubran la mayor o toda la superficie de un terreno, es de vital importancia en muchas áreas (vigilancia contra incendios, vigilancia en zonas bélicas, posicionamiento de torres de telefonía, posicionamiento de observadores en carreras,...)

El objetivo de esta tesis consiste en el desarrollo de soluciones eficientes a problemas realistas de visibilidad. En concreto, nuestro objetivo principal ha sido el desarrollo de algoritmos precisos y escalables para el cálculo de los parámetros

de visibilidad en MDE de mediana y gran escala usando las arquitecturas de computadores actuales. Dentro de los parámetros de visibilidad nos hemos centrado en el cálculo de 1)Cuencas Visuales, 2)Visibilidad Volumétrica de un MDE.

Una vez calculados estos parámetros, nuestro propósito es utilizarlos para la resolución de problemas reales como, el cálculo del mínimo número de observadores que cubran el mayor área de un territorio. Las claves de los algoritmos propuestos son; 1) una estructura de datos adecuada, 2) alta reutilización de datos en la computación y lo más importante, 3) alta escalabilidad en los sistemas de computación de alto rendimiento.

Palabras Clave: SIG, MDE, Cuencas Visuales, Visibilidad Volumétrica, Algoritmos Paralelos, Sistemas Multi-Core.

Índice general

Índice de figuras

Lista de tablas

Capítulo 1

Introducción

El conocimiento preciso de la superficie del territorio que es visible desde una determinada posición es un problema que inquieta al ser humano desde su más remoto pasado, ya que dicho conocimiento proporciona una valiosísima cantidad de información que le ayuda a resolver, en consecuencia, una multitud de problemas esenciales. Así, la búsqueda del mejor emplazamiento para las edificaciones defensivas (fortalezas o torres de vigilancia, por ejemplo) es sin duda una de las situaciones en los que alcanzar una máxima visibilidad es prioritario. De hecho, los antropólogos coinciden en indicar que es el segundo condicionamiento físico más importante para establecer una población (después de la disponibilidad de agua, e incluso por delante del clima). Pero la defensa no es el único motivo, también es beneficioso conocer la visibilidad en situaciones menos vitales, ¿o a quién no le gustaría tener su residencia en el lugar con las vistas más privilegiadas de su entorno?

De forma paralela al desarrollo o evolución de las poblaciones humanas, ha evolucionado la forma en la que nos interesamos por el espacio que es visible en nuestro entorno. En este sentido, el vertiginoso ritmo de desarrollo tecnológico en el que estamos inmersos ha derivado en nuevas aplicaciones en las que el conocimiento del espacio visible, aún manteniendo un interés primordialmente estratégico, no está limitado solamente a oteadores humanos —necesariamente ligados al suelo—. En este sentido, la aeronáutica y las telecomunicaciones han revolucionado también las aplicaciones del cálculo de la visibilidad. Véanse si no, como ejemplos, las siguientes áreas de interés: i) En telecomunicaciones: la localización de antenas de radio, TV, transmisores de telefonía inalámbrica o móvil y receptores que cubran el mayor área posible; ii)

En materias medioambientales y vigilancia: colocación de torres en zonas con conflictos bélicos, paso migratorio de aves o simplemente zonas susceptibles de sufrir incendios forestales; en la planificación ambiental y evaluación de impacto ambiental de zonas degradadas; iii) en turismo y deportes: la determinación de rutas de senderismo para excursiones, o la vigilancia en las carreras de montaña [30, 21, 16], y finalmente por citar algunas aplicaciones más, la visibilidad se emplea también en arqueología, para la reconstrucción de vistas y sitios históricos [29].

También es interesante reseñar que el conocimiento de la visibilidad retroalimenta el desarrollo tecnológico, en la misma senda que ya lo hizo en la antigüedad, cuando la seguridad de una población estratégicamente situada era garantía de un mejor desarrollo para sus habitantes. Un simple ejemplo se me ocurre mientras redacto estas líneas de la introducción: el emplazamiento estratégico de las torres de telefonía móvil a lo largo de una vía férrea, facilita a los investigadores el teletrabajo incluso durante los desplazamientos en tren. Por último, quisiera indicar que la imaginación no tiene límites a la hora de buscar aplicaciones en las que el conocimiento del espacio visible es imprescindible. Un ejemplo es el cálculo de la trayectoria de drones que se podrían utilizar para vigilar la mayor superficie posible de un territorio, minimizando el recorrido de los mismos (aplicable en la vigilancia de incendios). Incluso se podría utilizar la visibilidad como herramienta de geoposicionamiento, de forma que un vehículo puede averiguar el lugar en el que se encuentra mediante la comparación de la cuenca visual con parámetros previamente calculados. Intuimos que podría dar la solución al problema de geoposicionamiento extraterrestre (vehículos lunares o marcianos) donde las redes GPS no están disponibles.

1.1. Visibilidad

Según la Real Academia Española (RAE), la palabra visibilidad tiene dos acepciones muy parecidas:

1. Cualidad de visible. Elemento que se puede ver.

2. Mayor o menor distancia a que, según las condiciones atmosféricas, pueden reconocerse o verse los objetos.

A pesar de que las definiciones que nos proporciona la RAE – como, por otra parte, debe ser – son excesivamente genéricas, y en consecuencia insuficientemente

rigurosas para requerimientos científicos, ya dejan entrever, mediante la incorporación de parámetros como distancia y condiciones atmosféricas, que subyace una fuerte componente cuantificable en las mismas.

En el ámbito científico, donde es un requisito ser lo más preciso posible, para la primera acepción se podría definir la visibilidad de un observador fijo – en un punto de la superficie de la Tierra (o cualquier otro astro de estudio)– como la distancia a la que un objeto de tamaño y aspecto arbitrario deja de ser visible para el observador, a causa de la turbidez del aire o del agua. Por tanto, el trabajo que aquí se presenta es mucho más afín a la segunda acepción de la RAE, que de la primera de la que parte.

Se puede aun precisar más el grado de la definición de visibilidad para su implementación computacional; podemos definir la visibilidad de forma matemática entre un sujeto y un objeto separados por cierta distancia, como una variable booleana que adquiere el valor 1 si el objeto es visible por el sujeto y 0 si no lo es. De forma que si el valor es 1 entre el sujeto y el objeto no existe ningún obstáculo que impida la visión directa entre ellos, y el valor 0 en caso contrario. También, en éste mismo sentido, sería posible definir la visibilidad del sujeto para todo objeto en el área de estudio, en cuyo caso sería necesaria la combinación de parámetros espaciales (superficies, volúmenes, etc.) y booleanos.

La visibilidad sobre la superficie de un astro de grandes dimensiones, como un planeta, se ve afectada principalmente por el relieve del mismo. Es decir, por su orografía que está formada por elementos como las montañas, barrancos, valles, mesetas, etc. También existen factores externos al relieve de los astros que llegan afectar a la visibilidad en gran medida, como son las condiciones atmosféricas y ambientales de la superficie a estudiar. Si el espacio visible presenta un alto grado de polución o se producen fenómenos meteorológicos como la niebla, lluvia, nieve, etc., la visibilidad disminuirá. Otro factor que afecta a la visibilidad en un astro es el tamaño del mismo a la hora de realizar cualquier estudio, no es lo mismo la visibilidad en el planeta Tierra, que en Marte. Si ambos planetas tuviesen la superficie totalmente lisa y libres de obstáculos, entonces la visibilidad sería distinta debido al radio de curvatura que presenta cada uno de los planetas y el planeta con mayor radio presentaría una mayor visibilidad.

Aparte, y por si no fueran ya suficientes las variables que afectan al estudio de la visibilidad, las personas que la estudian han de tener en cuenta otros parámetros que le afectan, como la luz diurna, ya que no existirá la misma visibilidad de día que de noche y, por ejemplo, en vigilancia nocturna, es necesario el uso de equipos especiales para poder tener visibilidad. También la visibilidad sobre la superficie se ve afecta por la localización de elementos

obstáculos naturales o artificiales, como pueden ser árboles, edificios y puentes entre otros. Estos obstáculos reducirán la visibilidad de un observador que desea tener información detrás de estos elementos, al impedir captar información más allá de ellos. Así pues con todo lo mencionado en el presente apartado, se muestra que la visibilidad es un problema altamente complejo y que va más allá de lo simple que parece tras examinar su definición.

El presente trabajo, ante la cantidad de variables que afectan a la visibilidad, la trata teniendo en cuenta solo el relieve de la superficie de un astro (casi siempre la Tierra, como es obvio). Es decir que se descarta los efectos atmosférico, ambientales y si se considera que siempre hay luminosidad suficiente (como si fuese de día o que los equipos de medida pueden tener una captación suficiente aunque sea de noche). Además, se trata la visibilidad como una variable directa, que no tiene por que tener en cuenta los problemas de reflexión y refracción en la búsqueda de soluciones, ya que el cálculo de la visibilidad siempre se van a aplicar sobre la superficie de un medio que mantiene relativamente constante el índice de refracción, como es en caso del aire, de la superficie terrestre. Con estas consideraciones se han excluido intencionadamente toda componente temporal por un claro motivo: la superficie de de los planetas (como el nuestro) poseen una cierta deriva de los continentes o superficie que es prácticamente estática, pero que hace cambiar el relieve con el paso de las décadas. A pesar de ello, existen sobrados motivos para la incorporación de la variable tiempo en muchos de nuestros resultados: observadores (sujetos) que se desplazan, edificios que se levantan, árboles que crecen o puentes que se elevan, por citar sólo unos pocos casos. La mayor parte de nuestro trabajo se ha centrado en el estudio de la visibilidad entre observadores fijos situados en un territorio estático, ya que los resultados son fácilmente extrapolables a situaciones dinámicas, como la visibilidad de un senderista que camina por un parque natural.

1.1.1. Formalización matemática

Hasta ahora se ha tratado el problema de la visibilidad desde una perspectiva conceptual en la presentación del mismo. Pero para poder trabajar con la visibilidad de forma matemática y obtener resultados, es necesario darle un formalismo matemático al principal problema que nos atañe: el cálculo de la visibilidad de un territorio desde un observador.

Comencemos, sin embargo, con un concepto más elemental, como la visibilidad entre dos puntos:

Sean dos puntos A y B pertenecientes a una superficie S y, sea $h(x)$ la función

que representa la elevación (sobre el nivel del mar) de cada punto intermedio localizado a lo largo del segmento X que une A y B sobre la superficie S. Entonces se dice que B es visible desde A —y viceversa—, si se cumple que todos los puntos de la superficie que hay entre A y B, están situados por debajo de la recta que los une. Es decir que no existe ningún obstáculo a lo largo recta que une los puntos A y B que impida la visión entre los puntos. Esta definición queda recogida mediante la siguiente expresión matemática:

$$M \cdot (x - A) - (h(x) - h(A)) \geq 0, \forall x \in [A, B] \tag{1.1}$$

siendo M, la pendiente del segmento $A - B$

$$M = \frac{h(B) - h(A)}{B - A} \tag{1.2}$$

En la figura 1.1, se presenta dos funciones superpuestas en la misma figura. La primera función es $h1(x)$ representada por una línea continua de color azul y la segunda función $h2(x)$ está representada por la línea punteada de color rojo. Cada una de los funciones representa una hipotética superficie, pertenecientes a dos posibles terrenos distintos. En ambas funciones se desea saber si los puntos a distancia $x = A$ y $x = B$ de las dos funciones son visibles entre si. Se puede observar que las dos funciones son muy parecidas, y solo hay un pequeño tramo en el que la superficie de la función $h2(x)$ se eleva por encima de la función $h1(x)$ y además, dicha elevación sobrepasa la altura de la línea que une a los puntos A y B. De forma que A y B no son visibles entre sí, para el hipotética superficie $h2(x)$; algo que sí ocurría en el caso de la función $h1(x)$, donde no hay ningún obstáculo que impida la visibilidad entre A y B.

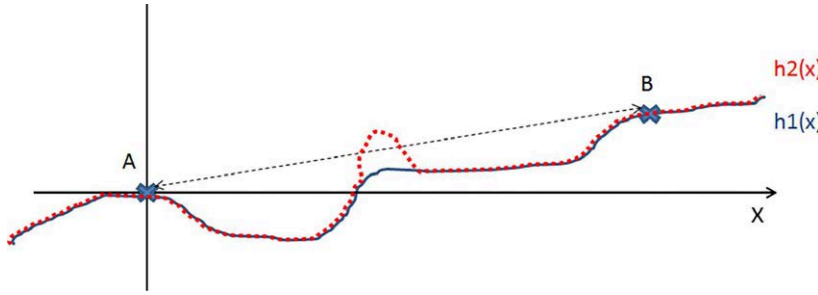


Figura 1.1: Curvas de Visibilidad de superficies similares.

De forma matemática la visibilidad de un punto sobre una superficie, se le asigna valor 0 a un punto si no es visible y 1 si es visible. De esta forma, dentro

de una superficie S, la cual se compone de una serie de puntos o localizaciones $p(x, y)$, la función Visibilidad (Vi) de un observador fijo o Punto de Interés (PI), se define matemáticamente bajo la siguiente expresión:

$$Vi_{PI}(x,y)_S = \begin{cases} 0 & \text{si p(x,y) no es visible desde PI} \\ 1 & \text{si p(x,y) es visible desde PI} \end{cases} \forall p(x,y) \in S \qquad (1.3)$$

A la hora de obtener la visibilidad sobre terrenos de naturaleza continua, surge la necesidad de realizar una descomunal cantidad de cálculos, ya que no es posible obtener una función que modele fielmente la superficie de un astro y por tanto, no se puede hacer un estudio funcional de una superficie, sino que es necesario el estudio de cada uno de los puntos que forman la superficie. Al ser las superficies de naturaleza continua, el número real de puntos a abarcar es infinito, por lo que a lo largo de la historia se han implementado soluciones que discretizan las superficies de estudio y las computan bajo sistemas informáticos.

1.1.2. La altura del observador en la Visibilidad

Nos referimos en este apartado a la altura a la que el observador o equipo de observación se posiciona sobre la superficie de estudio, de manera que no se encuentra pegado al suelo. Para una persona posicionada de pie sobre un terreno, la altura de Visibilidad es la distancia entre sus ojos y el plano de la superficie que pisa.

De partida, es normal pensar que el valor de visibilidad no sera igual si el observador se encuentre pegado a la superficie de estudio, a que se encuentre ligeramente sobre la superficie o incluso bajo la misma. En la formalización mostrada hasta ahora para el cálculo de la visibilidad de un solo observador posicionado en algún lugar de la superficie de un astro se ha considerando, que el mismo, está ligado a la superficie, es decir que su altura es 0.

En el cálculo de los parámetros de visibilidad, la altura tiene una elevada influencia, por su impacto en los resultados de los parámetros de visibilidad que se midan. Por ejemplo: estar sobre una torre vigilancia, en un helicóptero, usando un dron, entre otros casos, hará que el observador tenga más visibilidad y que sea también más fácil ser observado desde otras posiciones. Al estar en una posición elevada, los posibles obstáculos de las líneas de visión —entre observador y observado— pueden ser salvados, aumentando la visibilidad, y por tanto, la distancia visual máxima alcanzable.

Hay casos en los que la variación de altura presenta una gran efecto sobre los resultados de la visibilidad. Este echo se debe a que en estos casos una ínfima variación del valor de la altura puede hacer que la visibilidad del observador varíe descomunalmente. El ejemplo más sencillo de como puede afectar la altura respecto a la superficie en la visibilidad, se da en las superficies horizontales totalmente planas, como puede ser la superficie del mar o un extenso lago. Un observador posicionado en el centro de un lago con enorme extensión y con altura $h < 0m$, tendría una visibilidad nula, ya que todos los puntos de la superficie tienen una altura superior a el y solo captaría los punto más cercanos; sin embargo a poco que su altura supere la superficie del lago, aunque sean sólo un par de centímetros, la visibilidad varía enormemente, ya que el observador puede ver toda la superficie del lago.

Ahora bien, todos los ejemplos presentados hasta ahora han sido de aumento de la altura del observador, pero también está el caso de la disminución de la altura, como son los casos de agujeros o zunchos militares y submarinos entre otros ejemplos. El estar bajo la superficie permitirá al observador ser menos visible e incluso nada visible para otro observador, pero a la vez su visibilidad sera menor. Situación que se solventa mediante el uso de algún instrumento del tipo periscopio que permita elevar su altura de observación, para poder obtener una mayor visibilidad.

1.1.3. Alcance de la Visibilidad

Además de la altura, otro factor importante es la distancia máxima o alcance máximo que posee un observador, ya que no todos los observadores presentan el mismo alcance potencial de visibilidad. Este es un aspecto importante, ya que un observador podría ver o captar a otro observador, pero no viceversa. El caso más sencillo se ve en las imágenes de satélite: un satélite puede ver o captar a una persona saliendo de un edificio, pero dicha persona no puede alcanzar a ver al satélite por encontrarse este a varios cientos o miles de kilómetros de este, lo cual excede el alcance del ojo humano. El alcance máximo, además de los obstáculos viene determinado por las capacidades del observador: No es lo mismo que el observador sea una persona, cuya visibilidad alcanza en condiciones normales unas cuantas decenas de kilómetros, a que sea un sofisticado equipo de vigilancia o telescopio, con capacidad para ver a cientos o miles de kilómetros.

Generalmente, en la realización de estudios de visibilidad sobre la superficie terrestre se considera que el alcance máximo de un observador va a estar siempre limitado por el horizonte, es decir, la línea hipotética que aparentemente separa

la superficie terrestre con la bóveda celeste.

Como se ha podido apreciar, la cantidad de información para el cálculo de los parámetros de visibilidad suele ser exageradamente elevada. De forma que para poder trabajar con la información de los terrenos en equipos informáticos, la información debe ser digitalizada, como se ha mostrado anteriormente. Con el fin de realizar dicha tarea y poder llevar a cabo una computación de los datos surgieron hace algunas décadas una serie de herramientas, las cuales se denominan Sistemas de Información Geográfica o simplemente SIG en español (o en inglés GIS, acrónimo de *Geographical Information system*).

1.2. SIG y los modelos digitales del terreno

Bajo el nombre de Sistemas de Información Geográfica (SIG), aparecen distintas definiciones. Pero la más utilizada en la bibliografía, define SIG como el conjunto de herramientas software y hardware que, junto a los procesos adicionales que se implementan, permiten la organización, almacenamiento, manipulación, análisis, modelización y presentación de datos procedentes del mundo real y que están vinculados a una referencia espacial o terreno. Es decir, SIG es todo software y hardware que permite la utilización y gestión de datos superficiales de cualquier astro celeste, para aportar soluciones a problemas en los que intervienen dichos datos. De esta manera un SIG puede ser utilizado para investigaciones científicas dentro de muchas áreas. Como son la gestión de los recursos, la gestión de activos, la arqueología, la evaluación del impacto ambiental, la planificación urbana, la cartografía, la sociología, la geografía histórica, la colonización y estudio de satélites y planetas, por nombrar algunos de los campos en los que se pueden usar. Por ejemplo, un SIG podría permitir a grupos militares encontrar los caminos de mínima visibilidad dentro de un territorio, para no ser descubiertos por el enemigo [22], al atravesarlo o pueden ser utilizados por una empresa de construcción para ubicar una nueva zona residencial y aprovechar las ventajas de las vistas y cobertura de ese área sobre una zona de mercado con escasa competencia.

Los SIG permiten la gestión de toda la información espacial de una zona de estudio, ya que permite separar la información en diferentes capas y almacenarlas independientemente para poder trabajar con cada una ellas por separado. De forma que se facilita la generación de nuevas capas o la simple obtención de parámetros, que de otra forma sería muy difícil. En la figura 1.2, se muestra un ejemplo de distribución de capas en SIG, donde hay una separación de distintos tipos posibles de capas información a partir del mundo real. Se puede observar en

la figura la superposición de capas, y como una de las capas presenta información hidrográfica y otra información urbana.

Figura 1.2: Posibles división en capas de información de un SIG.

La primera utilización real de los SIG en el mundo probablemente se remonta a la década de los 60 en Ottawa (Canadá). Por lo cual, los SIG se pueden considerar relativamente jóvenes en comparación con los primeros estudios de visibilidad, que se remontan con seguridad a las primeras civilizaciones, como la egipcia que ya posicionaba sus monumentos y ciudades según orientaciones de astros celestes y considerando las distancias entre elementos del relieve, como ríos y montaña.

Los SIG han ido evolucionando paralelamente con los sistemas informáticos, ya que los SIG se basan tanto en la capacidad de procesamiento, como la de almacenamiento de estos para la obtención de soluciones al mundo real. Esta evolución paralela ha hecho posible hoy en día la resolución de problemas que hace apenas unas décadas era impensable poder resolver de forma eficiente y con errores mínimos. Es más, los cálculos de parámetros de visibilidad que se desarrollan en este trabajo, cuyos resultados hubieran tardado varios años de cómputo intenso hace apenas un par de décadas, se obtiene en cuestión de horas con los medios informáticos de hoy en día. Por ejemplo, algunos autores como [15] consideraban que el cálculo de la visibilidad alcanzable en todo punto de un territorio de gran extensión, donde los volúmenes de datos pueden ser de unos 10^9 puntos de estudio se necesitarían años ininterrumpidos de cálculo. Sin embargo, el presente trabajo mostrará que es capaz de computar dicha cantidad de puntos es menos de 15 horas.

1.2.1. Evolución de los SIG

Como ya se mencionó anteriormente, los primeros SIG se remontan a 1962, año en el que el Departamento Federal de Silvicultura y Desarrollo Rural Canadiense en Ottawa desarrolló el llamado Sistema de información Geográfica de Canadá (Canadian Geographic Information System, CGIS). Este sistema fue utilizado para almacenar, analizar y manipular datos recogidos para el Inventario de Tierras de Canadá (Canada Land Inventory, CLI) —una iniciativa orientada a la gestión de los vastos recursos naturales del país con información cartográfica relativa a tipos y usos del suelo, agricultura, espacios de recreo, vida silvestre, aves acuáticas y silvicultura, que era similar a los mapas catastrales españoles actuales—. El principal responsable del desarrollo fue Roger Tomlinson [38].

Algunas de las principales características de este primer sistema fueron: i) el uso escala de 1 : 50,000, ii) añadir un factor de clasificación para permitir el análisis de la información que permitía superponer capas de información, iii) realizar mediciones y iv) llevar a cabo digitalizaciones y escaneos de datos. Asimismo, soportaba un sistema nacional de coordenadas que abarcaba todo el continente americano. Una codificación de líneas en "arcos" y que almacenaba los atributos de cada elemento y la información sobre su localización en archivos separados. Como consecuencia de esto, Tomlinson está considerado como "el padre de los SIG's actuales" y en particular por el empleo de información geográfica convergente estructurada en capas, lo que facilita su análisis espacial. El CGIS estuvo operativo hasta la década de los 90. Fue desarrollado como un sistema basado en una computadora central y su fortaleza radicaba en que permitía realizar análisis complejos sobre un conjunto de datos que abarcaban todo el continente Americano.

En 1964, Howard T. Fisher en la Universidad de Harvard constituyo un grupo de trabajo para la Computación Gráfica y Análisis Espacial, donde se desarrollaron una serie de importantes conceptos teóricos en el manejo de datos espaciales, produciendo código software y sistemas germinales, tales como SYMAP, GRID y ODYSSEY, los cuales sirvieron como base para posteriores desarrollos comerciales tanto en el ámbito publico como privado de todo el mundo.

En 1977 en Fort Collins, Western Energy and Land Use Team (WELUT) y el Servicio de Pesca y Vida Silvestre de Estados unidos empezaron a desarrollar el proyecto Map Overlay and Statistical System (MOSS). Este es el predecesor de GRASS y empezó a ser usado en 1979.

En 1982 el Laboratorio de Investigación de Ingeniería de la Construcción del Ejército de los Estados Unidos (USA-CERL) desarrolla GRASS como

herramienta para la supervisión y gestión medioambiental de los territorios. En paralelo a GRASS, las compañias M&S Computing (más tarde Intergraph), Environmental Systems Research Institute (ESRI) y CARIS (Computer Aided Resource Information System) emergen como proveedores comerciales de software SIG. Incorporaron con éxito muchas de las características de CGIS, combinando el enfoque de primera generación de sistemas de información geográfica relativo a la separación de la información espacial y los atributos de los elementos geográficos representados con un enfoque que organiza y estructura estos atributos en bases de datos.

A continuación en la década de los 90 se inicia una etapa de difusión comercial para profesionales independientes, donde los sistemas de información geográfica empezaron a difundirse a nivel del usuario doméstico gracias al desarrollo de los ordenadores personales. A principios del presente siglo XXI, ha surgido un rápido crecimiento en los diferentes sistemas de información geográfica. Además, se ha visto restringido el número de plataformas, lo que hace más factible la continuación en el desarrollo de las mismas y su mantenimiento. Los usuarios de la primera década del siglo XXI comenzaron ha exportar el concepto de visualización de datos SIG a Internet, mediante visores web. Actualmente se puede observar una expansión en el número de desarrollos de software SIG de código libre. En este caso, a diferencia del software comercial, se suele abarcar una gama más amplia de sistemas operativos, permitiendo ser modificados para llevar a cabo tareas específicas, como la cartografía catastral del ministerio de hacienda español, con separaciones de zonas urbanas, y no urbanas mediante distintas cartografías.

Una consecuencia del elevado número de aplicaciones de los SIG, en actividades con componentes espaciales sobre cualquier tipo de terreno, ya sea marítimo o no, terrestre o extraterrestre, es la aparición de nuevas tecnologías que han incidido de manera decisiva en la evolución de los SIG. Algunas de ellas se han popularizado de tal forma que el público general, fuera del ámbito investigador, ha oído hablar de ellas y muchos son los que las utilizan. Un ejemplo es el famoso programa Maps de Google, Lanzado en 2005, y el cual la mayoría de las personas conocen. Esta es, en definitiva, la situación actual de los sistemas de información geográfica en la que constantemente se desarrollan nuevas aplicaciones, que mejoren tanto la precisión, eficiencia y la exactitud de los resultados en la solución a distintos problemas.

Además de observar un enorme número de aplicaciones para los SIG, se debe tener en cuneta la gran cantidad de formas de presentar los datos de entrada y salida. Esto requiere una estandarización del formato de los datos y de normas de transferencia, para poder trabajar de forma sencilla y eficiente en todo el mundo. Esta situación no es tarea fácil, ya que cada organización quiere

imponer su formato, de forma que todavía en la actualidad no se ha llegado a un total acuerdo. Por otra parte, además de las formas de representación de las soluciones en SIG, uno de los mayores problemas en el almacenamiento de los datos en los SIG, son los sistemas de coordenadas que se emplean en cada base de datos, ya que no hay un estándar común. Existe gran variedad de sistemas de coordenadas de referencia y proyecciones, de manera que un mismo punto presenta distinto valor de coordenadas, en función del sistema de coordenadas que ha sido seleccionado. Esta situación hace que algunas aplicaciones desarrolladas hasta la fecha solo sean validas, para resolver problemas en ciertas regiones o territorios y no se puede exportar la solución empleada a otras regiones de manera sencilla. Adicionalmente hace necesario que la persona que trabaje con SIG, deba tener ciertos conocimientos básicos en sistemas de coordenadas y sea capaz de pasar de un sistema de coordenadas a otro y pueda comprobar que no haya errores por la conversión entre sistemas.

1.3. Sistemas de Coordenadas

La presente tesis busca proporcionar soluciones para todos los sistemas de coordenadas planas. Por ello, la persona que use cualquier algoritmo de visibilidad decidirá que sistema de coordenadas empleará y deberá tener en cuenta los posibles errores qué se cometen por usar un sistema u otro en la obtención de resultados.

Los conocimientos básicos para poder trabajar sobre sistemas de coordenadas en SIG se resumen a continuación:

1. La localización de un punto sobre una superficie nos la proporciona sus coordenadas. Dichas coordenadas pertenecen a un sistema de coordenadas concreto, con un tipo de coordenadas especifico, como pueden ser coordenadas esféricas, cilíndricas o cartesianas entre otras. Todos los sistemas de coordenadas de navegación y posicionamiento sobre planetas y satélites naturales, cuya forma superficial del astro se puede aproximar por la de un geoide, como la Tierra o la Luna, parten de la aproximación a un elipsoide para la representación de las coordenadas del mismo. Dicho elipsoide queda definido básicamente por dos parámetros, que son la longitud del semieje mayor (a) y el aplastamiento (f) que relaciona el semieje mayor del elipsoide con el menor (b), siendo $f = \frac{a}{a-b}$.

   Es necesario enfatizar que la presente tesis se centra en el estudio sobre el planeta Tierra, pero las soluciones implementadas son extrapolables a

cualquier otro planeta o satélite natural de forma similar a un geoide, como puede ser el planeta marte, sobre el que existe un gran interés por su colonización y se han enviado varias sondas espaciales y equipos rovers para su estudio, como el famoso Curiosity o también llamado Mars Science Laboratory.

En el caso de la Tierra existen una gran cantidad de diferentes elipsoides para la representación de las coordenadas de posicionamiento. Por ejemplo, resulta que la mayoría de los mapas oficiales en España usan el elipsoide $ED50$, mientras que en Estados Unidos usan el $WGS84$. Los parámetros del elipsoide $ED50$ son semieje mayor $a = 638388m$ y aplastamiento $f = 297$, mientras que en el elipsoide $wGS84$ es semieje mayor $a = 6378137m$ y aplastamiento $f = 298,257223563$. Si se eligen dos puntos con las misma coordenadas en los dos sistemas $WGS84$ y $ED50$, se pueden encontrar uno respecto al otro desplazados varios cientos de metros sobre la superficie terrestre.

Si se mezclasen por error, datos de los dos sistemas de coordenadas sin la previa conversión de un sistema a otro, para analizar algún parámetro en SIG, entonces el análisis sería completamente erróneo. Por esta razón, la persona que use datos procedentes de varias bases, debe saber cuales son los sistemas de coordenadas usados en cada una de las bases de datos y si es posible, cómo pasar de uno a otro sistema mediante las correspondientes ecuaciones. Decimos que si es posible, porque puede que haya partes de la superficie que un sistema de coordenadas no cubra, mientras que otro sistema sí lo hace.

Como nota informativa, cabe decir que en Europa, desde 2014, se está haciendo un esfuerzo para que todos los países europeos usen el elipsoide $ETRS89$, por lo que muchas bases están en proceso de cambio y es más común todavía usar el $ED50$ en España que está más consolidada. Es más, los modelos utilizados en la presente tesis están referenciados bajo el elipsoide $ED50$, de forma que si queremos superponer las imágenes obtenidas por el algoritmo desarrollado sobre los mapas que proporciona la herramienta de Google Earth, es necesario la conversión previa de los datos del elipsoide de partida $ED50$ al elipsoide $WGS84$. Por otro lado decir que el 23 de mayo del 2018 la Comunidad Rediam presento el primer MDE con coordenadas referenciadas al elipsoide $ETRS89$. Esta comunidad es la principal responsable de normalizar, integrar y difundir la información sobre el medio ambiente andaluz y la que nos ha proporcionado los MDEs con los que se ha trabajado en la presente tesis.

Los modelos de los elipsoides empleados en las coordenadas son, a veces,

muy complejos, ya que los elipsoides no tienen en cuenta solamente los parámetros de forma de un elipsoide normal, sino que algunos modelos contemplan posibles variaciones de la forma de la Tierra, como pueden ser las derivaciones tectónicas de los continentes que son del orden de centímetros anuales.

2. Hay varias formas de proporcionar o dar las coordenadas, pero dos son las más habituales:

   a) *Coordenadas Geográficas* con parámetros de identificación latitud (Norte y Sur) y longitud (Este y oeste).

   Los parámetros latitud y longitud se miden en grados sexagesimales desde el centro de la Tierra. La latitud mide el ángulo entre el punto deseado y el plano Ecuatorial, de forma que el plano Ecuatorial es latitud 0. La latitud es siempre positiva y se mide de 0 a 90 grados, de forma que los puntos al norte del ecuador son de latitud Norte de $(0 - 90)$, mientras que los puntos del sur son de latitud Sur de $(0-90)$. Las líneas de igual latitud se llaman paralelos y forman círculos paralelos al plano del ecuador. La longitud se define como el ángulo a lo largo del Ecuador desde cualquier punto de la Tierra. A la hora de definir una línea o longitud inicial ha habido cierta controversia, pero se ha generalizado que la longitud que pasa por el observatorio de Greenwich, en Londres sea considerada como origen. Las líneas de igual longitud son los meridianos y van de 0 a 180 grados de Este a Oeste.

   La superficie terrestre bajo este tipo de coordenadas se trata como una esfera. En la figura 1.3 se muestra una imagen de Google Earth donde se aprecia la división en cuadrantes por el solapamiento de meridianos y paralelos. Esta imagen es la representación del elipsoide $WGS84$, utilizada por Google para todos sus sistemas de posicionamiento y mapas.

   Las Coordenadas Geográficas son ampliamente usadas en navegación marítima y posicionamiento GPS, pero no tanto en estudios científicos debido a que sus coordenadas son angulares y no es directo el cálculo de distancia en unidades de longitud.

   b) *Coordenadas Planas*, son las proyecciones de los elipsoides sobre planos. Este tipo de coordenadas permite trabajar sobre papel en coordenadas del tipo (x, y), como si fuesen coordenadas cartesianas, con distancias en metros o kilómetros.

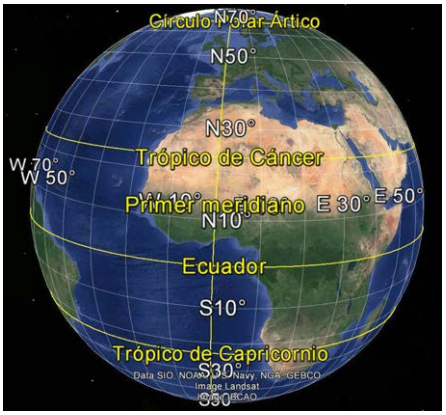


Figura 1.3: Mapa del mundo latitud y longitud, basado en el elipsoide WGS84.

Trabajar directamente con coordenadas de tipo esférico o elíptico suele ser complejo, por esta razón se realizan proyecciones, del espacio a tratar, a lo que serían superficies planas que son más sencillas de manejar.

Hay varios tipos de coordenadas planas, pero las más utilizadas son las *Coordenadas UTM*. UTM son las siglas de Universal Transverse Mercator, que utilizan la proyección de Mercator centrada en el ecuador para la generación de las coordenadas planas.

En la figura 1.4 se muestra un ejemplo de la proyección de Mercator sobre la línea del ecuador. Una vez realizada la proyección cilíndrica se genera el mapa plano de las coordenadas. En la figura 1.5 se muestra las zonas que comprenden Europa y África en las coordenadas UTM referenciadas al elipsoide $WGS84$, bajo la proyección de Mercator centrada en el ecuador.

Al generar el mapa plano UTM referenciado al elipsoide $WGS84$ se observa como los meridianos se proyectan sobre el plano, con una separación proporcional a la del modelo, así hay equidistancia entre ellos. Sin embargo los paralelos se van separando a medida que nos alejamos del Ecuador, por lo que al ir llegando a los polos las deformaciones serán infinitas. Esto hace que sólo se represente con precisión la región entre los paralelos $84°Norte$ y $80°Sur$, sin llegar a los 90° que cubre la Tierra completamente. Por consiguiente, hay una zona tanto del polo norte como del sur, donde las coordenadas UTM referenciadas al elipsoide $WGS84$ no pueden representarse, y por ello,

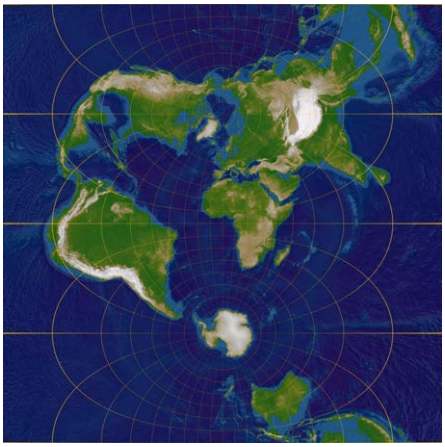


Figura 1.4: Mapa del mundo en proyección transversal de Mercator bajo el elipsoide WGS84, centrado sobre el meridiano 45^{o}E y el ecuador.

para estas zonas no se puede realizar ningún análisis bajo el elipsoide $WGS84$, siendo necesario el uso de otro elipsoide o proyección para poder realizar análisis de estas zonas.

Bajo las coordenadas UTM referenciadas al elipsoide $WGS84$ se divide la Tierra en bandas para la latitud y en husos en la longitud. Los husos tienen 6° de longitud cada uno, lo que da lugar a un total de 60 husos que empiezan en el huso 1 el cual comienza en 180°$Oeste$. En la figura 1.5 se muestran 22 husos de los 60, los que van del número 26 al 44. Por otro lado las bandas son 20, que dividen los husos directamente. Cada banda tiene 8° de latitud, excepto la última más al Norte que añade 4 grados más llegando a alcanzar los 84°$Norte$. Los husos se representa por un número del 1 al 60, mientras que las bandas por una letra del abecedario ingles, donde se elimina la letras I y O para no ser confundidas con ningún número. Las bandas normales van de la letra C a la X dejando las letras iniciales A, B, Y y Z para la proyección independiente de los polos. Se dice independiente porque bajo este sistema de coordenadas los datos de localización no guardan la misma escala que los demás.

La combinación de un huso y una banda se le llama zona. España en coordenadas UTM se localiza en los husos 29 y 30 y bandas S y T. Andalucía se encuentra principalmente en la zona $30 - S$.

Para dar las coordenadas UTM de un punto de la superficie, se indica

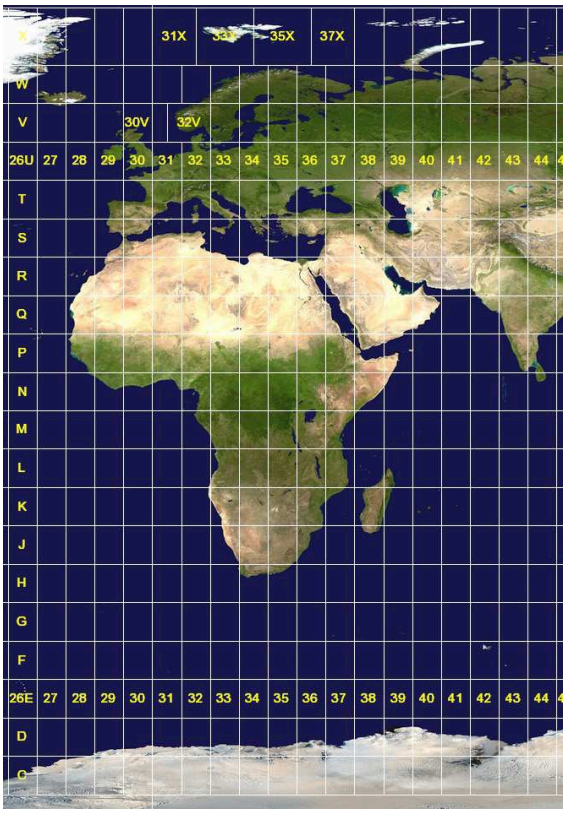


Figura 1.5: Coordenadas UTM.

en primer lugar la zona mediante el huso y la banda, y a continuación las coordenadas Norte-Este dentro de la zona. De esta forma, por ejemplo, las coordenadas UTM y Geográficas del faro del puerto de Málaga en España, que es una antigua edificación llamada la Farola, son: zona $30-S$ y $4064089{,}00mN$, $373665{,}00mE$ para sus coordenadas UTM bajo el elipsoide $WGS84$. Mientras que sus Coordenadas Geográficas son latitud $36{,}714096N$ y longitud $4{,}414556E$

Una vez sabido que existen diferentes formas en las que un mismo punto de un territorio está identificado o localizado por diferentes valores de coordenadas, dependiendo del sistema de coordenadas empleado, es importante saber como usar la información en los SIG. Para ello hay que tener claro el comportamiento básico de un SIG. Los SIG son bases de datos con información geográfica (datos

alfanuméricos) que se encuentra almacenados en sistemas informáticos, de forma que cada dato o conjunto de datos están asociados por un identificador. De esta manera, señalando un objeto de la base de datos se conocen fácilmente sus atributos. Entre los atributos están las coordenadas del punto o puntos del SIG. Además, en algún lugar de la base de datos debe estar indicado cual es el sistema de coordenadas de referencia. Raramente una base de datos SIG no incorpora la información de la base del sistema de referencia, pero si se da el caso no queda más remedio que preguntar al propietario de la base de datos SIG, sobre qué sistema de utilizado en la base de datos. El trabajar con bases de datos permite una gestión sencilla de la información por capas, haciendo que cada una de las capas incorpore un atributo. Así, como se ha mostrado anteriormente, cada capa puede tener información independiente que caracteriza un terreno, como pueden ser los edificios, los accidentes costeros, la vegetación, la hidrografía y carreteras entre otras muchas capas.

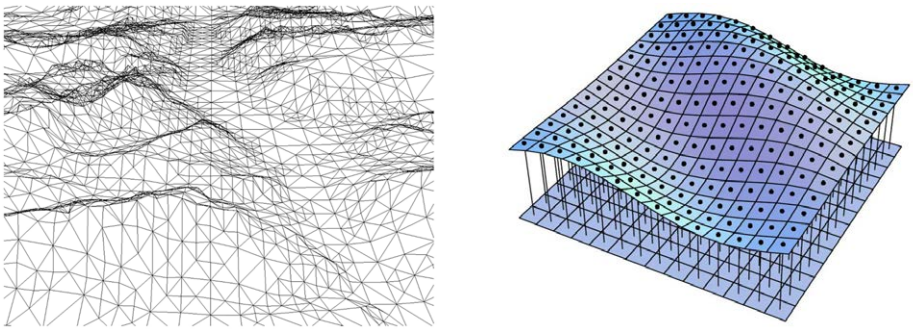
En el estudio de la visibilidad de terrenos, las capas con mayor importancia y más utilizadas, son las llamadas modelos digitales de superficie (MDS). Los MDS representa la superficie en $3D$ de un territorio de forma espacial mediante una estructura numérica de datos, la cual se almacena de forma que se puede representar de forma gráfica o usar la información en la resolución de problemas. Para que el lector no se vea desconcertado con otros términos, es necesario aclarar que en la biografía aparecen autores que llaman a los MDS, modelos digitales del Terreno (MDT). Existen varios tipos de MDS, pero los más utilizados en los algoritmos de visibilidad son los modelos digitales de elevaciones (MDE), en ingles (DEM).

En la mayoría de los casos, el término Modelo Digital de Superficie se refiere a la superficie de la Tierra e incluye todos los objetos que ésta contiene, como pueden ser edificios, monumentos, arboles... En cambio un MDE, representa la superficie del suelo desnudo y sin ningún objeto, como la vegetación o los edificios. Los MDE representan normalmente las alturas o cotas de un terreno, dentro un sistema de coordenadas concreto y con unos valores de resolución en los parámetros acotados.

1.4. MDEs

Dentro de los MDE, hay quien los divide en varios tipos de representación o tipos de modelos, pero probablemente los más utilizados son los mapas de alturas y las redes triangulares irregulares, en ingles triangulated irregular network (TIN). Los primeros son mallas de n-puntos regulares de tipo raster (cada punto y sus

vecinos están separados por la misma distancia), donde cada punto de la malla se caracteriza por sus coordenadas de localización (x,y) y altura (h), produciendo una representación discreta de la superficie de estudio. Mientras que los segundos, los TIN, son una representación continua de la superficie de un terreno mediante la unión de triángulos con diferentes inclinaciones y tamaños. Los TIN utiliza una base de datos vectorial, siendo cada vector guardado un vértice de uno o varios de los triángulos irregulares que representan la superficie continua del terreno a estudiar. La información de cada vector es la coordenadas son la posición(x, y) y la altura h de cada vértice. Así que la información que se almacena, tanto de los MDE de altura como de los TIN es parecida, ya que ambos guardan la información de localización de puntos y altura. Las principales diferencias están en la representación y en que las distancias entre los puntos del MDE de altura es constante, mientras que en los TIN no tiene porque serlo e incluso lo habitual es que las distancias sean variables, ya que las posiciones suelen atender más al tipo de orografía del terreno (terreno plano o variante), para buscar la mayor aproximación a la superficie real, con el menor número de vértices o triángulos posibles. En la figura 1.6 se puede ver un ejemplo del muestreo de los dos tipos de MDE sobre dos terrenos distintos. Se puede ver como los TIN proporcionan una mayor aproximación a la superficie del terreno, al dividir la superficie en triángulos y ser de distinto tamaño, mientras que los modelos de alturas solo muestrean la altura de los puntos de la superficie.



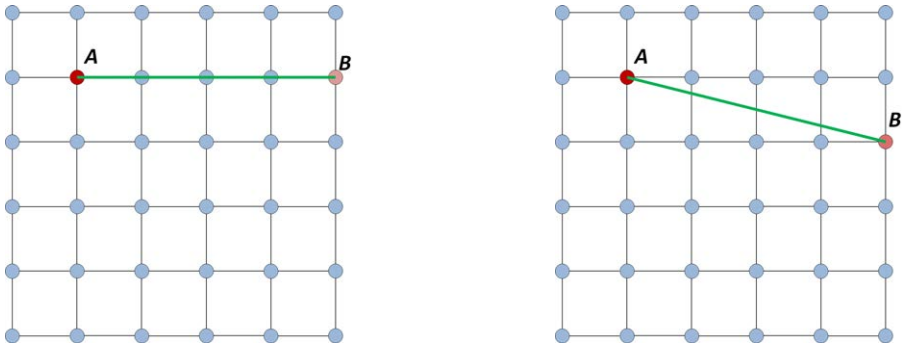
(a) MDE tipo TIN

(b) MDE tipo Mapa de Alturas

Figura 1.6: Ejemplo de dos de los posibles tipos de MDE.

En el presente trabajo se ha optado por el uso de los MDEs de alturas en vez de usar los TIN. Esta elección ha sido tomada por mantener los MDEs de alturas una estabilidad en las localizaciones de los puntos dentro de las mallas, que permite simplificar ciertas operaciones, al ser constante las distancias entre

todos los puntos del MDE. Por ejemplo, dos MDE de alturas pertenecientes a dos terrenos distintos, pero con el mismo número de puntos n, la misma forma de malla y la misma resolución, guardan la misma distancia entre todos los puntos, de forma que al superponer los dos MDEs de alturas resulta que los puntos de un modelo se encuentran directamente sobre los puntos del otro y se pueden realizar de forma directa operaciones entre los MDEs sin mucho esfuerzo. Algunas de las operaciones que se pueden realizar son de comparación o de resta para ver que MDE presenta mayores alturas y así medir el parecido de los relieves de dos territorios. Sin embargo, esta facilidad en la realización de algunas operaciones entre distintos MDE de alturas no se da para los modelos TIN, ya que al no tener mantenerse la estabilidad en las distancia entre los vértices, resulta muy difícil que el vértice de un TIN se encuentre en la misma posición relativa que el vértice de otro TIN. Por tanto, si se desea realizar operaciones entre dos modelos TIN, pertenecientes a distintos terrenos, es necesario el cálculo de la altura de los puntos que se localizarían en las mismas posiciones que los vértices del modelo con el que se compara. Este echo agrava el coste computacional sobre la realización de operaciones entre distintos TIN frente al uso de MDE de alturas.



(a) (b)

Figura 1.7: Ejemplos de línea que une dos puntos A y B distintos, sobre el mismo MDE.

Otro factor para declinarse por la utilización de los MDEs de alturas frente a los TIN, es que gracias a la regularidad en las mallas de los MDE de alturas se puede reducir el espacio del almacenamiento necesario de los modelos en el interior de los equipos informático, que los usan. Esto se debe a que no es necesario el almacenamiento de las coordenadas exactas de todos los puntos del MDE de alturas, sino que es suficiente con tener el conocimiento de las coordenadas exactas de uno de los puntos perteneciente al MDE de alturas a estudiar y saber

en que orden se han guardado los puntos dentro del fichero que los contiene junto con la resolución o distancia entre los puntos; ya que se puede calcular fácilmente las coordenadas de cualquier otro punto de análisis del MDE a estudiar. De esta forma se ahorra espacio en memoria al no ser necesario almacenar las coordenadas de cada punto. Simplemente, con solo almacenar en un fichero las alturas de los puntos del MDE y la coordenada del primer punto o de un punto de referencia, se tiene toda la información del MDE sin la necesidad de almacenar, para cada punto del MDE, sus coordenadas. Todo ello permite que sean más fácil de manipular y efectuar cálculos con ellos, así como la implementación de algoritmos más eficientes.

Un ejemplo aclaratorio de los descrito en el párrafo anterior es la suposición de estudiar un MDE de alturas con tamaño $N = 1000x1000$ puntos y resolución de $10x10m^2$ que se almacena en un fichero siguiendo el orden de izquierda a derecha y de arriba abajo, donde el primer punto almacenado tiene coordenadas UTM (0m Norte, 0m Este) y se quiere saber las coordenadas del punto almacenado en la posición 500 y 1000 del fichero. Resulta que el punto 500 está a una distancia de 500 puntos a la derecha del primero y el 1000 a una distancia de 1 punto por debajo del primero, ya que la primera fila del MDE corresponde a las posiciones 0 a la 999 en el fichero y la segunda fila del MDE va la posición 1000 a la 1999. Por tanto, el punto 500 está a una distancia de $500x10 = 5000$ metros del primero y sus coordenadas UTM serian por tanto (0m Norte, 5000m Este). En el caso del punto 1000 la distancia es de 10 metros por debajo, de forma que sus coordenadas UTM serian (-10m Norte, 0mEste).

Cuando se trabaja con un MDE tipo raster de alturas es muy importante tener en cuenta que no se dispone de toda la información de la superficie del terreno que representa, sino una muestra representativa de la misma realizada mediante el muestreo de la superficie, de forma que si, por ejemplo, se dispone de un MDE (M) de un terreno, el cual tiene una resolución de muestreo $R \times R\ metros^2$ de la superficie. Y hay dos puntos A y B pertenecientes al MDE, para los que se desea calcular la visibilidad entre ellos. Lo más sencillo para comprobar si son visibles entre si, es ver que ningún punto que se localiza sobre la línea que une A y B supera dicha línea de visión. En la figura 1.7a se observa un MDE, donde sobre la línea entre los puntos A y B existen una serie de puntos que permiten comprobar si son o no visibles A y B. Ahora bien, al tratarse los MDEs de alturas, como una red de puntos regular, puede darse el caso que la línea que une A y B no tenga ningún punto o que tenga muy pocos. Ver figura 1.7b. Este echo hace posible que dos puntos que en el mundo real no son visibles, se puedan considerar visibles, por no tener información del espacio entre ellos. Ante un caso así, la solución más obvia al no disponer de más datos, sería interpolar los puntos cercanos a la línea

entre A y B. De forma que el número de puntos sobre la línea que une A y B, siempre sea el mismo. Las interpolaciones que se suelen usar en los algoritmos que calculan parámetros de visibilidad con este tipo de modelos suelen ser la bilineal, cubica o vecinos cercanos.

Se podría pensar que por los problemas de falta de puntos en los MDEs de alturas, es mejor el uso de MDE tipo TIN, donde la información es más completa al ser planos delimitados los que conforman el modelo y por tanto, este problema no se da. Pero el MDE TIN no es totalmente real, al igual que sucede con los MDE de alturas; ya que se hace aproximación de la superficie real por medio de los planos que conforman los triángulos irregulares, de forma que los planos de los triángulos presentan una información parecida a la interpolación de los MDEs de alturas. La única forma de que un modelo sea más real es mediante el aumento de la resolución a niveles cuasi-infinito del número de puntos, caso imposible de tratar en un equipo informático. Por tanto, el cálculo de la visibilidad, en SIG, presenta casi siempre un error insalvable por este motivo.

1.5. Parámetros de Visibilidad en MDEs

Existen un número elevado de parámetros de Visibilidad que se pueden obtener usando MDEs, pero hemos optado por mostrar en este apartado solo algunos de los parámetros de visibilidad más relevantes en SIG, para los cuales el presente trabajo ha desarrollado algoritmos eficientes que los calculan tanto de forma secuencial como paralela. Estos parámetros se pueden agrupar según la dimensión o número de dimensiones del problema a tratar. Es decir pueden ser: 1)unidimensionales, que buscan resultados de visibilidad en segmentos y rectas; 2)bidimensionales, realizan estudios de superficies visibles para determinados puntos de un espacio y son los más habituales; 3) por último los parámetros de visibilidad pueden ser tridimensionales que calculan volúmenes, algo más sofisticados de computar, pero que añaden más información a los resultados.

La mayoría del software para el estudio de la visibilidad en SIG, que aparece en la bibliografía, calcula una aproximación de la superficie que podría ver un solo observador dentro de un terreno o en parte de el. De forma, que las mayoría de los algoritmos de visibilidad aportan soluciones a problemas bidimensionales, mientras que hay un menor número de soluciones a problemas tridimensionales y muy pocos trabajos tratan el problema unidimensional. En verdad el problema unidimensional apenas tiene complejidad de cálculo y el número de aplicaciones que lo utilizan es muy bajo, como mostraremos a lo largo de la presente memoria.

Los algoritmos existentes en la bibliografía, que proporcionan soluciones para aplicaciones en 2D, se pueden clasificar en dos categorías en función de la aplicación de destino [23]. La primera categoría se basa en el concepto de campos ISOVIST o ISOVIST-2D, introducidas en [4], para calcular las áreas visibles en pequeños espacios urbanos y de construcción. Los métodos basados en ISOVIST se utilizan sobre todo en la rasterización en tiempo real, en la visualización interactiva de espacios cerrados y pequeñas zonas urbanas [35, 33, 13]. La segunda categoría utiliza el concepto de Cuenca Visual o Viewshed en pequeños y medianos terrenos y paisajes naturales [37, 36, 23]. Este segundo tipo es el que atañe al presente trabajo, por usar primordialmente los MDE de alturas. Y aunque el algoritmo desarrollado en el presente trabajo permita calcular las Cuencas Visuales, estas se puede utilizar también en el cálculo de visibilidad 2D en las áreas urbanas.

La mayoría de los algoritmos basados en Cuenca Visual y ISOVIST calculan las áreas visibles para un solo observador, como mucho, para un número de observadores reducido [2, 5, 18, 20, 7, 14, 26, 25]. Sin embargo, muy pocos trabajos proponen soluciones para la computación de la Cuenca Visual Total [37]. Esta limitación se debe, entre otras razones, al elevado coste computacional de los métodos empleados y también al hecho de que se basan en una simplificación excesiva del problema geométrico, aportando soluciones con baja exactitud y lentas de obtener. Esto ha hecho que el presente trabajo dedique especial atención a este problema, para la obtención de soluciones eficientes en la computación de las Cuencas Visuales.

Un ejemplo de las Cuencas Visuales de un observador se puede ver en la figura 1.8. En dicha figura se muestra la Cuenca Visual del observador *POV* perteneciente a un terreno montañoso. En gris se presenta la superficie del terreno visible por el observador o lo que es lo mismo la Cuenca Visual del observador y en rojo la superficie no visible por el observador, pero que se encuentra dentro de los limites visibles del observador. Es decir la zona roja no es visible por el observador, pero está dentro de los limites de alcance del observador. Si el observador no presenta limites en la distancia a la que puede llegar a alcanzar la visibilidad, entonces el limite lo impone el horizonte. Recordemos que el horizonte, para un observador terrestre en una dirección, es el lugar donde se unen la superficie terrestre con el cielo. En la figura, toda la superficie que excede los limites del horizonte está excluida de la Cuenca Visual por exceder los limites de la visibilidad del observador en cada dirección, por esta razón, dicha superficie mantiene los colores que representa el relieve del terreno y no están en color rojo o gris.

Una de las características de las Cuencas visuales, es que presentan la propiedad de reciprocidad de la visibilidad. Esta propiedad consiste simplemente

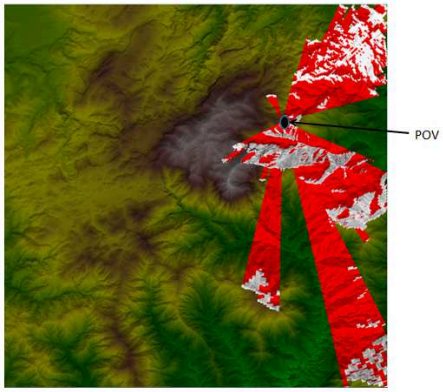


Figura 1.8: Cuenca visual del punto POV con coordenadas UTM X=4063890 e Y=324530, sobre la zona 30N, en Málaga, España. La zona gris muestra las áreas visibles desde el punto POV, dentro del terreno de estudio, mientras que la zona roja son las áreas invisibles entre dos áreas visibles para dicho punto. Las demás partes del terreno presentan colores según la orografía del mismo.

en que si un observador principal localizado en un punto puede ver cierta área de un terreno, entonces, desde ese área del terreno, otro observador pegado al suelo y con las mismas características de visión que el observador principal (alcance de visibilidad bajo las mismas condiciones) puede ver al observador principal y viceversa. Esta propiedad es muy útil y proporciona soluciones a un alto número de aplicaciones; por ejemplo, si se desea posicionar un escenario para un concierto al aire libre en una zona acotada de la que se dispone de su MDE de alturas, resulta que las Cuencas Visuales permiten buscar que punto de la zona presenta mayor visibilidad con altura del escenario, y a la vez es más visible desde todos los puntos del MDE, lo que hace que sea más visible el escenario por los asistentes.

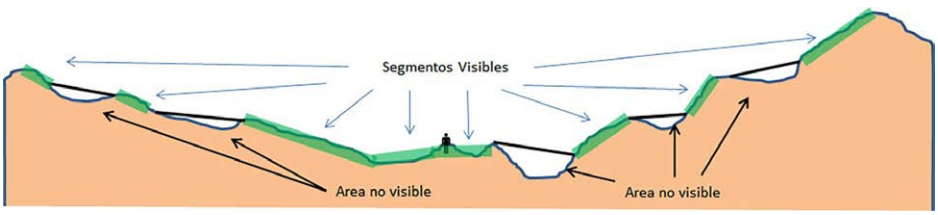


Figura 1.9: Segmentos Visibles que conforman la Cuenca Visual o área visible desde un observador en un hipotético terreno.

La forma de realizar la computación de las Cuencas Visuales conlleva una serie de pasos que se salen de una introducción, por lo que en el la sección 3.1, se muestra como es la computación de las Cuencas Visuales en más detalle y una definición mas exacta de la misma. Lo que si es importante tener claro es el concepto de las Cuencas Visuales para no confundirlo con otros parámetros de visibilidad. En la figura 1.9, se muestra el corte hecho a un hipotético terreno en una sola dirección, de forma que, mediante el corte se obtiene una función o curva bidimensional de la superficie donde se localiza un observador al que se le calcula su Cuenca Visual. Al analizar la curva, resulta que hay zonas visibles por el observador y otras no. Las zonas no visibles se dan cuando al trazar una recta desde el observador, hay parte de la superficie que obstaculizan la línea de visión entre el observador y la zona del relieve no visible. Por otro lado, las zonas visibles son la concatenación de puntos visibles del terreno sobre la curva, esta concatenación se ha destacado en color verde sobre la figura y generan segmentos visibles que son cada uno de los tramos de color verde. La suma del área correspondiente a los segmentos visibles es la Cuenca Visual en dicha curva para el observador.

Las Cuencas Visuales aportan soluciones a una gran multitud de aplicaciones, ya que, por ejemplo, pueden ayudar en la tarea de calcular la cobertura en los Sistemas de Telecomunicaciones, y por tanto, pueden ayudar ha eligir las mejores posiciones para la instalación del mínimo número de antenas. En la tarea de posicionar las antenas de los Sistemas de Telecomunicaciones, ha sido interesante ver el hecho de que hasta no hace mucho la elección de las localizaciones para la instalación de antenas dependía de criterios propios del equipo de ingeniería y no de estudios que contemplasen, las Cuencas Visuales o otro parámetro de visibilidad. Los criterios propios en la colocación de antenas, solían ser, la elección de la montaña o edificio más altos que permitía solapar lo menos posible las coberturas de las antenas. Es decir que las posiciones se encontraban sobre los limites de alcance máximo de una antena transmisora respecto alcance de otra antena transmisora, para no solapar demasiado las capacidades de cobertura y ahorrar el número de antenas. Una vez instaladas las antenas, se media la cobertura que proporcionan las antenas instaladas, pero casi en ningún momento se realizaban simulaciones previas de cobertura que tuviese en cuenta las características del relieve. Esta metodología no era eficiente, ya que daba lugar a zonas cercanas a las antenas con cobertura de señal nula por el mero hecho de estar desplazada la antena unas pocas decenas de metros sobre una localización que proporcionaría mejores prestaciones. La solución a este problema no solía ser el desplazamiento de la antena, sino la colocación de una nueva antena que cubriera la zona de interés; de forma que al instalarse nuevas antenas

se aumentaba el coste de instalación y sobre todo, el del mantenimiento de las instalaciones de Telecomunicaciones.

El no usar métodos realmente eficientes en el posicionamiento de las antenas en los sistemas de telecomunicaciones ha afectado a las coberturas y costes de instalación y mantenimiento de los mismos. El grado de afectación no ha sido el mismo para todos los sistemas de telecomunicaciones. Y la forma de solucionar los problemas de cobertura por no ubicar las antenas en los lugares óptimos no siempre los ha tenido que resolver la empresa suministradora del servicio. Por ejemplo, en el caso de la Televisión por radiodifusión se ha solido posicionar las antenas emisoras en las montañas o puntos más altos de las ciudades o terrenos a cubrir y se orientaban las antenas de los equipos receptores hacia las antenas de emisión. Si desde las localizaciones de los usuarios receptores no se capta con suficiente intensidad la señal de las antenas emisoras, entonces, les resulta necesario a los usuario receptores pagar el sobrecoste que les permita obtener una señal adecuada mediante la instalación estructuras pasivas, como mástiles que permita la elevación de la antena receptora para tener visibilidad de la emisora y la colocación de elementos electrónicos (activos), del tipo de filtros y amplificadores de señal que les permite una mejor recepción de la señal emisora. Otro ejemplo de los sistemas de telecomunicación donde no ha sido eficiente el posicionamiento de las antenas, es en las primeras instalaciones de telefonía móvil, cuya instalación de antenas usa una división de la superficie a cubrir en celdas. Las primera instalaciones no tenía en consideración el relieve del terreno o los obstáculos que se presentan (edificios, puentes, montañas ...), sino simplemente el tamaño de la zona a cubrir y el alcance de los equipos de telecomunicaciones. Esta situación hacía que hubiese zonas que necesitaban cobertura y no la poseían, por la colocación inadecuada de las antenas al mantener la topología en celdas. Las empresas de telecomunicaciones resolvían este problema con la instalación se antenas de menor alcance dentro de las celdas, de forma que se hacia una nueva división de las celdas en subceldas. Esta solución, por una parte, aumentaba la cobertura y el número de teléfonos móviles que eran capaces de gestionarse en las celdas y apenas repercutía directamente para el bolsillo del usuario, pero aumentaba en gran medida los costes de instalación y mantenimiento de las redes de telefonía móvil por parte de la empresa.

La Cuenca Visual es un problema considerado en $2D$, por ser su resultado en $2D$, pero el análisis que se hace de los datos de un MDE como entrada es en $3D$, ya que se analiza las alturas de los puntos de una superficie. Por tanto, en la solución hay una perdida de información, debido a que el análisis no es solo de las posiciones de los puntos, sino también las alturas, para poder determinar si son o no visibles los puntos de un terreno.

Una aproximación más realista de la visibilidad, consiste en abordar el problema en tres dimensiones y no pararse solo en el problema $2D$,ya que la visibilidad en $3D$ da una mayor información que por ejemplo, la Cuenca Visual que solo da información de la superficie visible. Por ello en el presente trabajo se ha utilizado un nuevo parámetro llamado **Visibilidad Volumétrica** de un observador.

La Visibilidad Volumétrica se define como el volumen de aire contenido en el espacio delimitado en la parte superior por las líneas o plano del horizonte visible y en la parte inferior por los segmentos visibles de igual forma que las Cuencas Visuales. Si se usa la misma muestra bidimensional de superficie que la de las Cuencas Visuales mostrado en la figura 1.9 y se calculan la Visibilidad Volumétrica, entonces se obtiene como resultado el mostrado en la figura 1.10. En dicha figura se puede apreciar como quedaría limitado el volumen visible de un observador mediante la zona coloreada en tonos con degradado de amarillo y rojo. Este degradado es en función de la distancia al observador, siendo el tono amarillo para el volumen visible más cercano y el color rojo para el más lejano. Se puede observar en la figura como está delimitado el volumen visible entre las líneas del horizonte del observador para la parte superior y en la parte inferior por los segmentos visibles más las líneas que unen los segmentos visibles. Los segmentos visibles son las líneas de color verde más gruesa, al igual que pasaba con las Cuencas Visuales de la figura 1.9. Este volumen visible, es la Visibilidad Volumétrica que podría ver un observador si solo se estudiara esta curva, donde el observador miraría solo en una dirección y su opuesta.

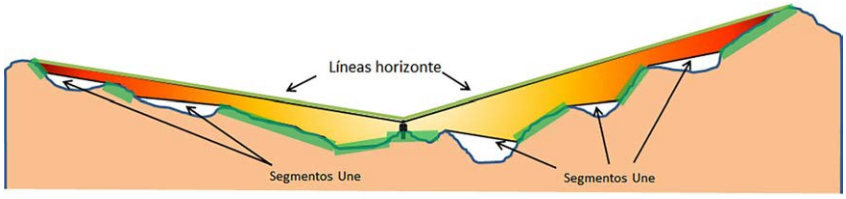


Figura 1.10: Volumen Visible sobre una hipotética superficie, donde se muestran los limites inferior y superior del volumen visible.

La Visibilidad Volumétrica de un observador en un perfil, se aleja un poco de la representación real sobre un terreno tridimensional. En la figura 1.11 se puede observar la Visibilidad Volumétrica en $3D$ de un observador (POV) en un hipotético terreno. En dicha figura se ve la posición del observador y el contorno que forma los puntos del horizonte que capta el observador en todas las direcciones del terreno, representado mediante la línea azul. Cada una de las línea rojas de la

figura corresponde a cada una de las líneas trazadas de horizonte, si se dividiera el espacio en n-sectores equiespacidados desde el observador.

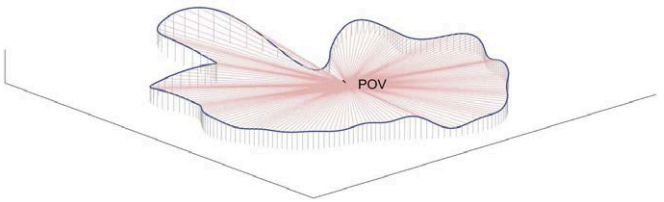


Figura 1.11: Visibilidad Volumétrica del observador POV dentro de un hipotético terreno.

Hay quien piensa que el cálculo de la Visibilidad Volumétrica no aporta nada nuevo, que no lo hicieran ya las Cuencas Visuales. Pero la realidad se equivoca, ya que la Visibilidad Volumétrica da información de capacidad o volumen, muy útil en aplicaciones de paisajismo, arqueología, hidrografía, etc., donde la Cuenca Visual es incapaz de aportar soluciones. La Cuenca Visual como se ha mencionado anteriormente solo proporciona información de la superficie que puede ver o no un observador, lo cual hace que fácilmente existan áreas de un terreno de estudio que presentan la misma Cuenca Visual desde un observador y sin embargo posean relieves totalmente distintos, que para ciertas aplicaciones pueden ser problemáticas o ventajosas. En la figura 1.12 se muestra cómo dos zonas diferentes de un terreno, medidas desde el mismo observador(POV), presenta la misma superficie o área visible de $1hm^2$ en la medida de su Cuenca Visual. Sin embargo, resulta que las Visibilidades Volumétricas de las dos zonas son muy distintas, debidas a la diferencia en la topología del relieve donde se encuentra cada una de las zonas visible a estudiar. En el caso de la zona señalada con color rojo, corresponde a una zona de montaña con grandes variaciones de altura, lo cuál hace que tenga una elevada Visibilidad Volumétrica, $167hm^3$ medida desde el observador POV. Por otro lado, la zona de color marrón, presenta un relieve casi plano con baja variación de altura, lo que hace que su Visibilidad Volumétrica sea pequeña, $4hm^3$. De este ejemplo se puede extraer que las zonas con idénticas Cuencas Visuales pero mayor Visibilidad Volumétrica tendrán un relieve más pronunciado, menos plano. Estas zonas son más interesantes para paisajismo, por ofrecer mayor variación en las formas del paisaje y serán menos idóneas por norma general para la construcción de zonas urbanas, por necesitar un mayor gasto

en la adaptación del terreno y instalación de infraestructuras —alcantarillado, luz, agua, sistemas de telecomunicaciones...—, que si fuese plano el suelo y por consiguiente presentase una menor Visibilidad Volumétrica. Aunque puede haber cierto número de personas que prefiera vivir en zonas con alta Visibilidad Volumétrica, sin importar el incremento del precio, ya que dichas zonas serán menos propensas a inundaciones al tener mayor desnivel y presentaran mejores vistas. Por otro lado, la Visibilidad Volumétrica presenta soluciones a otros problemas como puede ser la búsqueda de puntos del cauce de los ríos que sean mejores para la construcción de presas de agua, ya que la Visibilidad Volumétrica nos da una medida de capacidad.

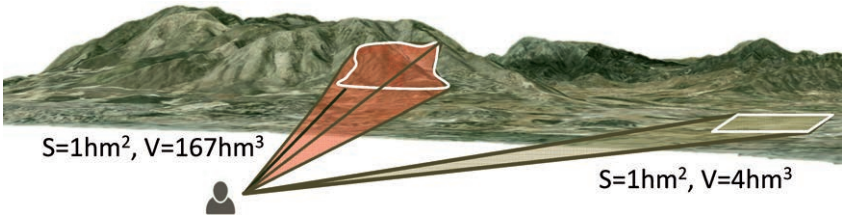


Figura 1.12: Visibilidad Volumétrica de un observador(POV) posicionado a $6km$ de dos áreas (roja y verde) con la misma cuenca visual de $1hm^2$, pero con diferentes Visibilidades Volumétricas de $167hm^3$ y $4hm^3$.

La complejidad de cálculo de la Visibilidad Volumétrica es superior a la de las Cuencas Visuales, como cabria de esperar. Muy pocos estudios calculan la Visibilidad Volumétrica, y los que lo hacen, apenas la calculan para un observador o un conjunto pequeño de varios observadores en pequeñas áreas urbanas representadas por su MDE [33, 35]; dichos trabajos se centran más bien en alcanzar una buena visualización en tiempo real, ya que persiguen el cálculo de la Visibilidad Volumétrica sobre un observador que se desplaza en una pequeña área.

Además, tras una revisión bibliográfica de trabajos existentes en este área, no se ha identificado ninguna solución para Visibilidad Volumétrica en un MDE con un tamaño de terreno más grande que el de una ciudad. La mayoría de los trabajos se centran en terrenos de tamaño pequeño, como un edificio, una urbanización o una ciudad sin llegar donde solo se computan unos cientos de miles de puntos. Por lo que en MDEs con un tamaño de millones de puntos, como los que trata el presente trabajo, nadie lo ha hecho antes. Esto nos hace ser pioneros en el cálculo de la Visibilidad Volumétrica en MDE de gran extensión y con una alta resolución, donde el número de puntos del MDE va desde varios millones de puntos más de mil millones de puntos, como puede ser el MDE de

Andalucía si se trata con una resolución de $10 \times 10m^2$, resultando un MDE con 1500 millones de puntos.

A partir de los parámetros de Cuencas Visuales y Visibilidad Volumétrica no es complicado el cálculo de otros parámetros, para solucionar problemas, tales como el número mínimo de observadores o torres que permiten dar cobertura a la mayor superficie de un territorio, el cálculo de las trayectorias con mínima visibilidad o con máxima por donde un equipo de control debería o no pasar dependiendo de si desea ver la máxima superficie o no ser visto por el enemigo, y una infinidad más de aplicaciones posibles. Esto ha hecho que el presente trabajo haya perseguido en todo momento la creación de una herramienta, que permita resolver tanto el problema de Cuencas Visuales y Visibilidad Volumétrica de un solo observador, como de un conjunto de observadores y incluso de todos los puntos de un MDE. El resultado de este último punto son los mapas de Cuencas Visuales totales y de Visibilidad Volumétrica total, muy útiles en la colocación de antenas de telecomunicaciones, posicionamiento de torres de vigilancia forestales, búsqueda de sitios con mejores vistas, etc. Además, se ha generando el primer mapa de Visibilidad Volumétrica total de un MDE de gran extensión como es el de Andalucía con mil quinientos millones de puntos bajo una resolución de $10x10m^2$.

1.6. Contribuciones de la Tesis

En el presente trabajo se han desarrollado e implementado nuevos algoritmos para cálculo las Cuencas Visuales y la Visibilidad Volumétrica de todos los puntos de un terreno representado por un MDE, independientemente del tamaño que tenga. En particular, se desarrolla un sencillo algoritmo para aumentar la eficiencia computacional en las computadoras modernas con arquitectura multi-core. Concretamente, las contribuciones han sido:

- Un algoritmo para el cálculo de las Cuenca Visuales totales de todos los puntos de un territorio.
- Un algoritmo novedoso para el cálculo de la Visibilidad Volumétrica.
- Una implementación paralela eficiente de ambos algoritmos en sistemas multi-core.
- Una solución para el cálculo de la visibilidad total en terrenos de grandes dimensiones con una alta resolución.

La metodología desarrollada procesa los MDE en una o dos etapas, en función de las dimensiones del MDE. Los modelos de mayores dimensiones (MDE grandes) necesitan ser particionados en modelos de menor tamaño (MDE medianos). La relación entre el tamaño del modelo (y las estructuras de datos asociadas) con los tamaños de los distintos niveles inferiores de la jerarquía de memoria es el criterio utilizado para diferenciar entre modelos grandes o medianos. Este aspecto se discute en el siguiente capítulo, donde también se muestra un marco de trabajo paralelo.

Por lo que sabemos, el Algoritmo desarrollado es el primero para el cálculo de los mapas de índice de Visibilidad Volumétrica y Cuencas Visuales total sobre MDE de gran tamaño.

Además abordamos otros problemas, como son 1) calcular el número mínimo de observadores que hacen falta para cubrir un territorio dependiendo de las necesidades del problema, ya que no es lo mismo cubrir un territorio en el ámbito de las telecomunicaciones que en el ámbito de la vigilancia contra incendios o otro tipo, 2) aportar soluciones que dan un valor añadido en el campo del senderismo y el paisajismo gracias al cálculo de la Visibilidad Volumétrica, 3) otras aplicaciones, como el cálculo de horizontes y la detección de zonas aisladas que son precursoras para el cálculo de trayectorias optimas a recorrer por un equipo o sistema de vigilancia se muestra en los apéndices de la presente memoria.

1.7. Estructura de la memoria

El presente trabajo se ha estructurado de la siguiente forma: En primer lugar se ha realizado una introducción donde se han mostrado los problemas reales que se van a resolver de una forma eficiente; además se ha presentado el estado del arte de los Sistemas Información Geográfica junto con un estudio de la situación actual de los problemas de Cuencas Visuales y Visibilidad Volumétrica. El capítulo 2 describe detalladamente los algoritmos desarrollados para el cálculo de parámetros de visibilidad sobre MDE de tamaño medio y grande. Además, se muestra la herramienta desarrollada para el posicionamiento del mínimo número de observadores que cubran la mayor superficie posible de un terreno. El capítulo 3 muestra con más detalle las variaciones del algoritmo desarrollado para las distintas aplicaciones que se han estudiado, como son la Cuenca Visual total, Visibilidad Volumétrica total y posicionamiento del mínimo número de observadores. Por último, se describen las principales conclusiones y trabajos futuros en el capítulo 4.

Capítulo 2

Cálculo de la visibilidad

El objetivo del presente capítulo es mostrar el algoritmo que se ha desarrollado para facilitar el cálculo de los parámetros de visibilidad más relevantes que se pueden obtener a partir de un MDE, aprovechando al máximo las capacidades de cómputo de los sistemas informáticos actuales. Una vez presentado el funcionamiento del algoritmo en su forma secuencial, se muestra una versión del mismo, modificado para la ejecución paralela, que permite explotar la potencia de cálculo de los sistemas *multi-core* y *multi-socket*.

De todos los parámetros de visibilidad que se pueden obtener de un MDE, el presente trabajo se centra en dos, los cuales previamente se han descrito en el capítulo de introducción y que son: 1) la Cuenca Visual o Viewshed-2D, ampliamente estudiada en la bibliografía, en la que introducimos un novedoso parámetro no tratado hasta ahora, 2) la Visibilidad Volumétrica (Viewshed-3D) de todos los puntos de un MDE. A partir de estos dos parámetros se pueden obtener soluciones para una gran cantidad de aplicaciones, tales como: el cálculo de mínimo número de observadores que pueden dar cobertura a un territorio y el cálculo de caminos o senderos con máxima visibilidad, para la creación de rutas paisajisticas.

El cálculo de cada uno de los parámetros de visibilidad, consiste en la gestión ordenada de un conjunto de formulas matemáticas sencillas que se aplican siguiendo un determinado orden sobre los datos proporcionados por los MDEs objeto de estudio. La complejidad, y en consecuencia la potencia de los algoritmos de visibilidad no recae tanto sobre la complejidad de las formulas utilizadas, como si en la gestión eficiente de los datos que intervienen junto las formulas

matemáticas, ya que el volumen de datos a tratar es enorme y normalmente se tiene que iterar muchas veces sobre los mismos datos. Por tanto, esta gestión de datos es la responsable de obtener los resultados de los parámetros deseados en más o menos tiempo y con mayor o menor error respecto al valor real. Además, la forma en la cual se realiza la gestión de los datos por parte del algoritmo va a permitir que el mismo pueda ejecutarse en paralelo de una manera más o menos sencilla, o que solo pueda realizarse una implementación secuencial, ya que si los distintos pasos que componen el algoritmo guardan una gran dependencia entre ellos, entonces resulta imposible hacerlos concurrentes de forma eficiente. Esta última circunstancia es la que se da en la mayoría de los algoritmos de visibilidad desarrollados hasta la fecha, de forma que no se pueden hacer paralelos de forma eficiente y por tanto, estos algoritmos desaprovechan las enormes capacidades de cálculo de los sistemas informáticos actuales.

A la hora de redactar la presente memoria, hemos separado la gestión de los datos de nuestros algoritmos de visibilidad por una parte, y por otra parte las formulas matemáticas empleadas en el cálculo de algunos de los principales parámetros y aplicaciones que se analizan en la presente tesis. Se ha optado por esta forma de presentación para intentar hacer más comprensible el trabajo realizado y porque los algoritmos desarrollados presentan dos partes (la gestión y formulación) bien diferenciadas. Además los algoritmos desarrollados parten de una gestión básica, para luego alcanzar el desarrollo de sus variantes paralelas, que es lo que se muestra en el presente capítulo. Por otro lado, en el capítulo 3, se muestran en detalle las expresiones matemáticas que permiten el cálculo de cada parámetro y aplicación tratados, así como los principales resultados de los experimentos realizados.

Antes del desarrollo de nuestro algoritmo, se ha realizado un exhaustivo estudio sobre el estado del arte en la computación de parámetros de visibilidad y se ha podido comprobar la existencia de una gran variedad de algoritmos([2, 14, 28, 18, 6, 5, 25],...). Este estudio nos ha permitido comprobar que la inmensa mayoría de los trabajos previos calculan solo un parámetro de visibilidad para un solo punto sobre un MDE representativo de un terreno de tamaño relativamente pequeño, al no superar la extensión de los terrenos a estudiar las decenas de hectáreas de superficie. Para el problema de analizar un parámetro de visibilidad sobre un conjunto numeroso de puntos, como si estos fuesen observadores, o sobre todos los puntos de un MDE, el número de trabajos encontrados en la bibliografía es muy bajo. Además, los trabajos que realizan el cálculo de un parámetro de visibilidad para un conjunto de observadores sobre un MDE, lo hacen principalmente de forma secuencial y los que lo hacen de forma paralela simplemente repiten la misma metodología usada en el cálculo de un solo punto

para el cálculo del conjunto de puntos a los que se desean calcular la visibilidad. Resulta una tarea bastante ardua y lenta el aplicar la misma metodología de un solo punto en al análisis de un conjunto de puntos de estudio, al no incluir mejoras que puedan reducir el tiempo de cálculo, como por ejemplo, hacer posible que los algoritmos puedan de unas iteraciones a otras aprovechar datos y operaciones ya realizadas con anterioridad sin la necesidad de volver a repetir cálculos idénticos sobre los mismos datos. Por tanto, existe una clara necesidad de crear un algoritmo que pueda reaprovechar al máximo las operaciones y que además realice el cálculo eficiente de los parámetros de visibilidad para todos los puntos de un MDE y no solo para un punto o conjunto de puntos. Por otro lado, existe también la necesidad de poder tratar MDEs de cualquier tamaño de puntos y extensión de superficie. Estas necesidades son las que han llevado al presente trabajo a buscar soluciones para calcular parámetros de visibilidad sobre MDEs de cualquier tamaño de puntos, aprovechando al máximo las capacidades de los sistemas computacionales actuales, de forma que se arrojen los resultados deseados en el menor tiempo y con la mayor precisión posibles.

Antes de poder computar de forma eficiente todos los puntos de un MDE, es necesario comprender como se realiza el cálculo de parámetros de visibilidad de un solo observador o punto, y situarse en la complejidad inherente de este tipo de problemas, para poder llegar a visualizar la potencia del algoritmo desarrollado en esta tesis.

2.1. Visibilidad para un solo punto de un MDE

Recordamos que en la bibliografía aparecen varias metodologías que permiten el cálculo de algún parámetro de visibilidad para un solo punto u observador que se encuentra localizado dentro de un terreno representado mediante su MDE. La principal diferencia entre las distintas metodologías que emplean cada algoritmo de cálculo de visibilidad, reside en la forma y el orden en los que se recorren los puntos del MDE a tratar. En este sentido, una de las metodologías más extendida en los SIG realiza una división, en un número de partes fijo, de todo el espacio de estudio en torno al observador. Cada una de las divisiones sigue una forma radial y con direcciones equiangulares desde el observador al extremo del MDE de partida.

El análisis de la visibilidad se hace en cada una de las divisiones de manera independiente, de forma que los puntos de una de ellas no se consideran en el análisis de las otras. Esta forma de analizar el problema de visibilidad es conocido como *enfoque por sectores o sectorial*, en la que cada una de las partes es un

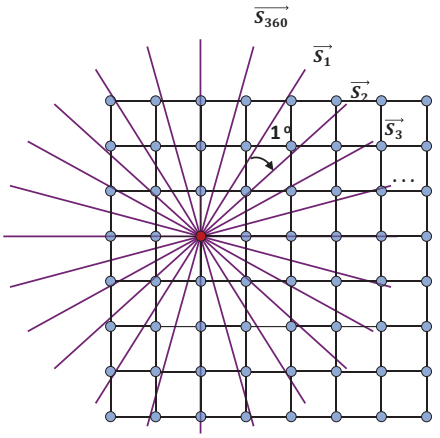


Figura 2.1: La visibilidad de un observador (punto de color oscuro) es calculado a partir del análisis en las 360 direcciones equidistantes que conforman los vectores $\vec{S}_i$ con $i = 1, 2..., 360$.

sector con dirección $\vec{S}_i$. El caso más habitual es realizar la división del espacio en 360 partes separadas un grado entre si, siendo la dirección de cada división representada por un vector $\vec{S}_i$ con $i = 0, 2, ..., 359$, véase la Figura 2.1.

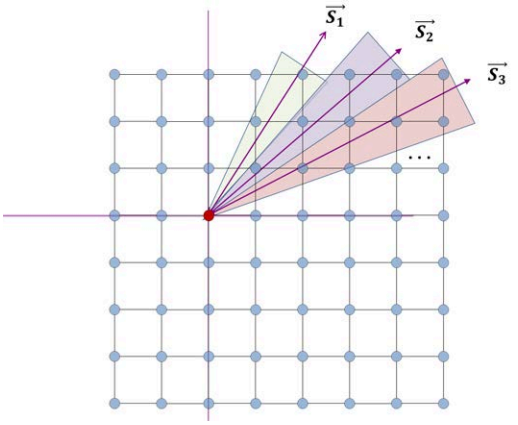


Figura 2.2: 3 sectores (área sombreada) de un observador (punto en rojo) con direcciones $\vec{S}_1$, $\vec{S}_2$ y $\vec{S}_3$.

El vértice de cada sector es el propio observador y la amplitud del ángulo de apertura en grados de cada sector es igual al cociente de dividir 360 grados entre el número de divisiones o sectores. Una vez dividido el espacio en sectores se identifican algunas propiedades. La primera es que el número de puntos por sector a analizar, y por tanto, los valores de visibilidad de cada sector son independientes entre si. Otra característica que se cumple para un sector S_i cualquiera, es la existencia de pocos puntos en las distancias cercanas al observador, mientras que habrá una gran cantidad de puntos para distancias lejanas como se puede ver en la figura 2.2, donde además se muestran 3 sectores con áreas sombreadas y con direcciones principales $\vec{S}_1$, $\vec{S}_2$ y $\vec{S}_3$.

El enfoque o análisis de puntos de un MDE por sectores se introdujo por primera vez para calcular la reflexión bidireccional en [5]. En dicho trabajo se usa una estrategia parecida a la del cálculo de los horizontes de [24]. El conjunto de puntos a considerar en cada sector, con dirección $\vec{S}_i$, es fundamental para la precisión y el tiempo de computo de los parámetros de visibilidad sobre un MDE. Los autores del articulo reducen significativamente el número de operaciones y por tanto, el tiempo de computo al realizar un decremento del número de puntos empleados a la hora de calcular el parámetro de visibilidad sobre un sector. Si por ejemplo, sólo tienen en cuenta los puntos que se localizan sobre la línea central del sector y descartan el resto de puntos del sector que no se localizan sobre la misma, resulta obvio que apelando a la estadística, esta metodología ahorra enormemente el número de operaciones a realizar en el cálculo de la visibilidad, siempre y cuando la longitud máxima de análisis supere el equivalente a una decena de puntos, ya que como se ha mencionado antes cada sector presenta forma triangular, de manera que a largas distancias desde el observador, manteniendo el ángulo de apertura, el número de puntos dentro del arco que se describe dentro del sector es considerablemente alto, mientras que para distancias cercanas al observador el número de puntos dentro del arco del sector es muy bajo. Por tanto, al considerar solo los puntos de la línea central cuando las distancias supera las decenas de puntos hace que se produzca una significativa reducción del número de puntos a considerar. En este caso, se esta considerando que los puntos sobre la línea central del sector es una muestra estadística significativa para obtener un buen resultado del análisis de todo un sector. Ahora bien, lo que no muestran con suficiente detalle los autores es la posibilidad de que los resultados no sean suficientemente precisos e incluso totalmente erróneos. Este hecho es consecuencia de la posible perdida de análisis de puntos relevantes para el cálculo de la visibilidad, debido a que estos puntos no se encuentren sobre la línea central del sector y por tanto, puede producirse una perdida enorme en la precisión de los resultados. Esta perdida no es constante, depende de la resolución

del MDE, del número de sectores, de la longitud del sector, de si caen o no sobre la línea central del sector puntos relevantes y de la rugosidad del terreno, ya que si el terreno es totalmente plano no se cometerá prácticamente error alguno si se compara los resultados con las mejores aproximaciones de cálculo. Pero si el terreno es muy rugoso, puede darse el caso que el error cometido sea enorme y el resultado obtenido no se parezca en nada al valor real.

A los algoritmos que solo tienen en cuenta los puntos que se localizan sobre la línea central del sector, o muy próximos a ella, se les conoce, en los SIG, como algoritmos de Línea de Visión Directa o simplemente Línea de Visión (en ingles *Line-Of-Sight (LOS)*). El mecanismo de los algoritmos LOS son un punto de partida del algoritmo desarrollado en el presente trabajo, por lo que se abordará con más detalle en los siguientes apartados.

Otros trabajos han mejorado la precisión en el cálculo de los horizontes mediante un pequeño aumento del volumen de datos a analizar dentro de cada sector y no solo considerar los puntos sobre la línea central del sector. Estos trabajos han añadido los puntos más próximos al observador y los más próximos a la línea central del sector para el cálculo de los horizontes [37, 34]; o bien han realizado algún tipo de interpolación que permita tener un número de puntos suficientes sobre la línea central del sector [15]. Además, han realizado el cálculo mediante el análisis de todos los puntos del sector para la obtención de resultados más reales y de esta manera validar el resultado de sus algoritmos. Tras la realización de los experimentos, los autores de estos trabajos, muestran que sus aproximaciones mejora considerablemente la precisión de los resultados frente a los algoritmos solo LOS. Por otro lado, el introducir puntos cercanos a la línea central del sector solo produce un pequeño incremento en el número de operaciones y por tanto, el tiempo de ejecución del algoritmo apenas aumenta. En estos trabajos, la comprobación de los resultados se ha realizado principalmente mediante la comparación de experimentos visuales, donde han comprobado que si se calcula la visibilidad usando todos los puntos de los sectores (método exhaustivo), se obtienen resultados similares a los obtenidos usando su propia metodología, mientras que para algunos terrenos los resultados de los algoritmos "solo LOS" difieren bastante del análisis exhaustivo. De esta forma demuestran que su solución es una buena aproximación del cálculo de los horizontes y superan las aproximaciones de los algoritmos LOS al ofrecer resultados más parecidos a la computación de todos los puntos de un MDE.

Por otro lado, los trabajos [37, 34] han medido los tiempos de ejecución de distintos algoritmos demostrando que la complejidad de cálculo por el número de operaciones a realizar, si se usa todos los puntos del sector, es enorme, y que el tiempo de cómputo es muy levado frente a los algoritmos de visibilidad que solo

tienen en cuenta los puntos sobre y entorno a la línea central del sector.

2.1.1. Complejidad de los algoritmos de un único observador

El tiempo de ejecución de los algoritmos de visibilidad, que se ejecutan sobre equipos informáticos, es uno de los mejores indicativos de la complejidad del problema a tratar. Dicho tiempo de ejecución se ve afectado principalmente por dos factores:

- Número de operaciones a realizar.
- Requisitos de memoria necesarios para la computación.

Número de operaciones a realizar

Las operaciones que se usan en el cálculo de los parámetros de visibilidad son sencillas, al ser estas mayoritariamente comparaciones, sumas, restas, multiplicaciones y divisiones. Sin embargo, el número de operaciones que se deben realizar suele ser extremadamente elevado. Por ejemplo, si se desea realizar el estudio de algún parámetro de visibilidad para un solo observador localizado dentro de un MDE de alturas cuadrado, con una dimensión de malla de $2000 \times 2000 = 4$ millones de puntos, resulta, que si se desea calcular el parámetro teniendo en cuenta toda la superficie del MDE y no se utiliza ninguna estrategia de simplificación del número de operaciones, entonces es necesario como mínimo la comparación del observador con cada uno de los puntos del MDE, por lo que se deben realizar 4 millones de evaluaciones entre el observador y cada uno de los puntos, para comprobar si los puntos son o no visibles entre si. Este número de evaluaciones no es ni siquiera representativo del coste computacional en el análisis de los parámetros de visibilidad. Para cada pareja puntos no basta con realizar la comparación directa entre el observador y un punto de estudio, sino que es necesario la generación previa de un perfil de puntos, entre el observador y el punto a analizar, que permita comprobar si el punto es o no visible. Por tanto, se deben generar casi 4 millones de perfiles de puntos, donde la longitud de los perfiles no es siempre la misma, sino que depende de la distancia a la que se encuentre el punto de estudio respecto al observador. De esta forma, la distancia y por tanto, la cantidad de puntos por perfil va desde 1 punto para el caso más cercano y $2000 - 1 = 1999$ puntos para el caso mas lejano, sobre el MDE de 4 millones de puntos presentado. El caso más pequeño se da cuando

el punto de análisis es uno de los puntos vecinos al observador y el caso más costoso se da cuando el observador se localiza es una de las esquinas de la malla del MDE y el punto a analizar en la esquina opuesta, habiendo ,por tanto, una distancia de una fila de puntos menos uno. Bajo esta forma de análisis donde se estudia un perfil para cada punto se pueden llegar a necesitar más de cinco mil millones de comparaciones ($\sum_{i=1}^{i=1999} i \cdot (2i+1)$), para obtener el valor de un parámetro de visibilidad de un solo punto u observador sobre un MDE de 4 millones de puntos. Por tanto, el número de comparaciones real supera en más de 3 ordenes de magnitud la primera aproximación. Por otro lado, en el cálculo un parámetro de visibilidad es necesario el uso de más operaciones como sumas, restas, divisiones, etc., en la determinación de si un punto es visible o no, por lo que se puede observar que el número de operaciones real es inmenso y que por solo el número de operaciones a realizar, en los problemas de visibilidad, se requieren de equipos con mucha capacidad de computo, si se quiere alcanzar soluciones en tiempos razonables (que no sean años de computación ininterrumpida).

Todo lo mostrado en el presente apartado sobre la complejidad de cálculo de parámetros de visibilidad ha sido un resumen del caso de hacer la computación sin ninguna mejora. Ahora bien la complejidad computacional debida al número de operaciones de los algoritmos de visibilidad de un solo observador ha sido mejorada significativamente en los años, gracias al continuo desarrollo de nuevos algoritmos que reducen su complejidad. Si se parte de un MDE de tamaño N, resulta que la complejidad es de orden $O(N^2)$ para el cálculo de visibilidad de un solo punto del MDE, por necesitar operar con cada uno de los puntos del MDE. Atallah [2] realiza el cálculo del horizonte utilizando un sistema informático con un solo núcleo, sobre el que ejecuta un algoritmo de divide y vencerás para computar la envolvente superior de N segmentos en el plano, este algoritmo presenta una complejidad computacional $O(\alpha(N)logN)$ donde α es la inversa de la función AcKerman, siendo una función de crecimiento lento, casi constante. Hersberger [18] mejora el algoritmo de Atallah, alcanzando una complejidad de $O(NlogN)$. De Floriani and Magillo [14, 9] utilizan un algoritmo probabilístico que calcula el horizonte para un solo punto en diferentes resoluciones de terrenos triangulados en tiempo $O(NlogN)$. El método aproximado del enfoque sectorial de los algoritmos LOS, donde se divide el horizonte en S sectores y en cada sector determina la elevación del horizonte considerando solamente los puntos del terreno que están en la línea central del sector[1], reduce la complejidad a $O(S(\sqrt{N}))$. Este último método es el que más reduce el número de operaciones aunque no presente la mejor precisión en los resultados. El algoritmo desarrollado

[1]En la mayoría de los casos, este tipo de algoritmos considera la interpolación de los puntos más cercanos al eje, muestreados con una frecuencia similar a la resolución del MDE de partida. En otros casos se consideran los puntos inmediatamente adyacentes al eje.

en el presente trabajo sigue esta filosofía, con lo que de partida la complejidad del algoritmo desarrollado es $O(S(\sqrt{N}))$, donde, como S va a ser una constante, se puede decir que la complejidad es $O(\sqrt{N})$ para la computación de la visibilidad de un solo punto, sin poder por el momento reducirse mucho más, si se desea obtener unos resultados que no se alejen demasiado de la realidad. Todas estas complejidades presentadas, son para el estudio de la visibilidad sobre un solo punto o observador, de forma que, si desea computar la visibilidad para todos los puntos de un MDE, la complejidad se ve multiplicada por N.

Un ejemplo real de la reducción de operaciones que se lleva a cabo mediante el enfoque sectorial de los algoritmos *LOS*, se puede realizar sobre el mismo MDE de 4 millones de puntos, donde hacían falta más de billones de comparaciones si no se aplica ninguna optimización. Si sobre este MDE se desea calcular algún parámetro de visibilidad sobre un único observador, localizado en el centro del MDE, con la metodología sectorial de los algoritmos LOS y considerando 360 sectores, donde solo es necesario el análisis de un solo perfil de puntos por sector. Resulta que la longitud de cada uno de los perfiles puede ser de unos 1000 puntos, ya que al tener el MDE un tamaño de 2000×2000 puntos, un observador en el centro del MDE dista como máximo de 1000 puntos de los bordes. Por tanto, el número de comparaciones pasaría a estar entorno las 360000 comparaciones. Lo que lleva a una posible reducción en más de 4 ordenes de magnitud del número de operaciones.

En la reducción de la complejidad mediante el número de operaciones, existen autores que usan un número de sectores inferior a 360. Sin embargo hay evidencias de que la precisión de los resultados se degrada significativamente, si el número de sectores es menor y la metodología no considera todos los puntos contenidos en el sector, es obvio reconocer que el error resultante en algunos casos resulta ser significativamente grande; con lo cual no se proporciona una solución suficientemente realista. Por otro lado, se podría pensar en aumentar el número de sectores más allá de los 360, de manera que pueda proporcionar un resultado más fiel. Esta última opción aumentaría el coste computacional y aunque el resultado proporcionando sería más preciso, puede resultar innecesario tal nivel de precisión para la mayoría de las aplicaciones. Por ejemplo, en aplicaciones de telefonía móvil y de visión por un observador humano, la señal de radio y la visión humana decrecen con el cuadrado de la distancia, de forma que hay situaciones que para varios kílometros de distancia no es necesario tener mucha precisión, como se verá en el capítulo 3. Por tanto, es necesario un compromiso entre en número de sectores y la precisión deseada en los resultados.

Requisitos de memoria necesarios para la computación

El segundo factor que afecta al tiempo de ejecución para el cálculo de la visibilidad es la cantidad de memoria necesaria por los algoritmos de visibilidad para poder llevar a cabo la computación. Este factor es de gran relevancia sobre la computación de la visibilidad, ya que en la mayoría de los casos la cantidad de fallos en memoria caché que se producen es tan elevado, que afecta más a los tiempos de computación que las propias ejecuciones de las operaciones. Todo ello radica en el hecho de que las necesidades de memoria en la implementación y ejecución de los algoritmos de visibilidad son normalmente grandes, por las necesidad de mantener en memoria una gran cantidad de datos. Por ejemplo, si se está estudiando un MDE de 1500 millones de puntos donde cada punto esta representado por una variable de tipo float de 4 bytes, entonces sería conveniente tener un array en caché de unos $5,6$ Gbytes para datos de entrada. Por otro lado, es también conveniente tener otro array de tamaño similar o superior para los datos de salida, más variables auxiliares para comparaciones y otras operaciones intermedias, lo que hace que probablemente sean necesarias varias decenas de Gbytes de espacio en memoria caché. Esta cantidad de memoria raramente se puede encontrar en un computador, y muchos menos en sus cachés.

En resumen el problema de las implementaciones de los algoritmos de visibilidad reside en el hecho de que los datos a analizar junto a las variables y estructuras de datos necesarias en la computación de los algoritmos no caben dentro de la memoria caché de los procesadores actuales a usar. Esta situación hace que el número de fallos de memoria caché se eleve de forma proporcional a la complejidad algorítmica por el número de puntos del MDE a tratar, y por tanto, si no se quiere que el tiempo de cómputo necesario también se incremente de forma proporcional por los fallos en caché, hay que buscar alternativas que permitan reducir este problema.

Aunque se ha presentado un sencillo ejemplo donde se muestra la necesidad de aprovechar eficientemente la jerarquía de memoria, es precipitado aun hacer un análisis más preciso, ya que, aunque el tamaño total necesario depende en parte del primer factor (tamaño de los datos de entrada), también depende del número y tamaño de las variables que se emplean y de las velocidades de la memoria del sistema informático sobre el que se compute. Por estos motivos hemos optado por que el lector pueda observar la limitación de este factor de memoria de forma precisa más adelante en la presente memoria, gracias a la presentación de la estructura desarrollada para el algoritmo del presente trabajo, en el cálculo de los parámetros de visibilidad. Así como un análisis de rendimiento, donde se puede apreciar en cuanto afecta la tasa de fallos de caché en la computación

del algoritmo de visibilidad desarrollado, ya que los autores de otros algoritmos no han tenido en cuenta este factor. Cualquier ejemplo que se ponga de este factor sin la base de un algoritmo concreto es muy abstracto y poco preciso. Lo que no se deja para después es la observación de que a día de hoy, los algoritmos de visibilidad se encuentra principalmente limitados por la memoria (memory-bound), por ser la capacidad de cómputo de los sistemas *multi-core* muy elevada, pero la capacidad de la memoria caché muy limitada para este tipo de problemas.

2.1.2. Validación del análisis sectorial

A la hora de diseñar un nuevo algoritmo bajo el enfoque sectorial, es necesario tener la certeza de que este tipo de métodos son suficientemente validos. En la bibliografía, se ha hablado mucho de la validez de los algoritmos basados en análisis sectorial y del número de sectores a utilizar. Pero aunque es una metodología que lleva más de una década usándose, nadie ha realizado un estudio en detalle, ya que la validación de otros autores, simplemente se ha basado en la comparación del resultado final obtenido para un punto, con el de otros algoritmos para el mismo punto, y ninguno ha usado un conjunto significativo de puntos en el análisis de los resultados. Este hecho nos ha parecido una falta de rigurosidad por muchos de los trabajos anteriores y hemos querido comprobar que efectivamente el análisis sectorial es una buena aproximación para los problemas de visibilidad reales.

En primer lugar, recordamos que la sectorización de la visibilidad consiste en el cálculo de la misma, para un punto de observación POV, mediante el cálculo de la visibilidad en cada uno de los S sectores circulares resultantes de una partición acimutal del MDE de estudio. Dentro de cada sector, el problema se resuelve de forma unidimensional, es decir, se considera que toda la superficie contenida en el sector (2D) se representa por un segmento, cuya dirección es la misma que la del sector (1D) (la bisectriz del ángulo o línea central de sector), ver figura 2.3. En las siguientes líneas, utilizamos el concepto de perfil de puntos para denominar la curva de elevaciones de los puntos que se encuentran en la bisectriz del sector. Intuitivamente, el perfil sería la curva resultante de la intersección del terreno por el plano vertical que pasa por la bisectriz. En la figura 2.4 se muestra un ejemplo de un perfil de superficie que se puede obtener para un observador sobre un MDE, donde la zona roja de la figura representa la orografía de la superficie, a la cual se ha realizado el corte por la bisectriz de un sector de estudio, obteniendo el perfil mostrado.

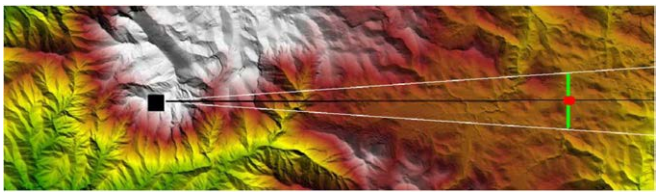


Figura 2.3: Sector de un observador. El observador es el cuadrado negro y el sector es el área entre las líneas blancas. La línea negra es la representación unidimensional del sector, siendo el punto rojo la representación de todo el arco de color verde, en el estudio de la visibilidad.

Figura 2.4: Perfil de puntos de un observador, en uno so los posibles sectores de estudio sobre un MDE.

La cuestión más relevante en la validación del estudio sectorial, es demostrar con rotundidad la afirmación de que el eje central de un sector (representado por los puntos que están sobre el eje central y los cercanos al mismo) es una muestra estadística fiable de todo el sector. Es decir, que la visibilidad de todo el sector se puede extrapolar a partir de los datos del perfil en el eje central del sector. Si no se cumple esta condición, entonces resultaran poco realistas los resultados de los algoritmos sectoriales y no tendrán sentido su uso en la resolución de problemas de visibilidad que busquen precisión en los resultados.

El principal factor que afecta en la validez del estudio sectorial, es la dependencia existente entre el relieve del terreno y la validez de los resultados. Concretamente, el factor del relieve que más afecta es la rugosidad. En las grandes llanuras (y cómo no, en el mar), es bastante probable que la altura o elevación en un punto de la bisectriz de un sector, no difiera mucho respecto de las alturas de los demás puntos que se encuentra sobre el arco del sector con misma distancia desde el observador. Es decir, que en llanura, los puntos que se encuentran sobre el arco que puede trazar un punto que se encuentra sobre la bisectriz de un sector al hacer la rotación horizontal a izquierda y derecha desde el observador y con limites los propios del sector, tienen valores similares en todo el arco. En este caso,

queda claro que los puntos de la línea central de un sector, si son una muestra significativa del sector de estudio. Sin embargo, en territorios muy escarpados, con repentinos cambios de elevación, como puede ocurrir, por ejemplo, en Monument Valley (Arizona), los Tepuys de Venezuela o en ciudades plagadas de rascacielos, como Dubái, es mucho más probable que la elevación en un punto no sea un valor representativo de todo el arco y menos cuando la distancia al observador sea grande, por lo que resulta necesaria una medida de lo abrupto que es un terreno. Existen numerosos estadísticos en la literatura para medir la rugosidad de un territorio [19, 11]. Aunque, paradójicamente los parámetros más fiables de rugosidad no se han calculado para superficie terrestre, sino para superficies extraterrestres por la necesidad de aterrizar las naves espaciales, en el lugar correcto sin el riesgo de que se dañen en el aterrizaje de las mismas sobre la superficie de otros planetas, con la correspondiente pérdida económica de enorme magnitud [17].

Además de la rugosidad del terreno, la amplitud del arco o tamaño del ángulo del sector circular influye en la validez de la aproximación de la línea central. Esta amplitud pone limites a cuál puede ser el número máximo de sectores a analizar para el estudio de la visibilidad alrededor de un punto, siendo siempre el número máximo de sectores igual a 360 dividido por la amplitud de cada sector en grados y viceversa. A una mayor amplitud del arco, la representatividad de su valor central puede perder valía por haber más cantidad de superficie o puntos sobre el arco que representa el punto central de dicho arco. Igualmente, la representatividad del punto central no es la misma cuando los puntos están cerca, a cuando están lejos del observador, siendo mayor la representatividad cuando los puntos del eje están más próximos al observador por estar más cerca del eje central, que para distancias lejanas, donde puede haber miles de puntos dentro del arco que se localiza a la misma distancia desde el observador hasta ellos, y en promedio, más lejos del eje.

Para corroborar que el eje central es una muestra estadística fiable se ha realizado un experimento, en un lugar bastante rugoso de nuestro planeta: en el parque de la Sierra de la Nieves. La Sierra de las Nieves se localiza en la provincia de Málaga, dentro de Andalucía. Este espacio se sitúa, en el ámbito geológico de contacto entre los complejos Bético, Penibético y la unidad denominada Campo de Gibraltar. Presenta un característico relieve con escarpes de falla y cabalgamientos, así como mantos de corrimiento; todas estas estructuras presentan además una dirección dominante Noreste-Suroeste, por lo que, las mayores rugosidades de sus perfiles se presentan en la dirección perpendicular Noroeste-Sureste.

Para el experimento se han elegido 1000 puntos como si fueran observadores,

dentro de un MDE que cubre la zona Noroeste de la Sierra, y se han elegido tres sectores para el cálculo de uno de los parámetros de visibilidad, en concreto su Cuenca Visual, la cual se mostrara con todo detalle en el apartado 3.1, del capitulo siguiente. Los sectores elegidos presentan direcciones trazadas hacia el Sureste con valores de ángulo ($S = 136^o, 158^o y 179^o$) desde Norte. Además se han considerado tres ángulos de apertura para los sectores(0,5^o, 1^o y 2^o). La elección de ángulo de apertura ha sido al estudio de trabajos anteriores donde demuestran —de varias formas— que la pérdida de precisión es irrelevante cuando el número de sectores S es superior a 180 sectores, de manera que la amplitud del ángulo es de 2^o o menos.

Por otro lado, en la validación del experimento se ha considerado el hecho de que en los problemas de visibilidad, la precisión requerida es decreciente con el cuadrado de la distancia, como suele suceder en un gran número de aplicaciones (muy habitual en las observaciones visuales o en trasmisiones de radiofrecuencias). Este hecho hace que se sea benévolo en los experimentos y se ha limitado la longitud del radio de estudio en los sectores a 25km dentro de un MDE del terreno seleccionado con $10 \times 10m^2$ de resolución, que es más que suficiente para muchas de las aplicaciones. En resumen, bajo este experimento se analiza un total de $3 \times 3 \times 1000 = 9000$ sectores (se han elegido 1000 puntos para los cuales se calcula la visibilidad solo 3 sectores con 3 ángulos de apertura distintos por cada uno de las direcciones de los sectores).

En cada punto de observación y para cada dirección de sector citados, se ha calculado su Cuenca Visual de dos formas :

1. Mediante extrapolación de los valores de visibilidad obtenidos en el eje central del sector. En este caso, sólo hay que trazar un segmento o línea central al sector. Esta línea define una curva de elevaciones del terreno, obtenido a partir de la interpolación de los puntos del MDE más próximos al segmento. Sobre esta curva se calcula la Cuenca Visual. En la figura 2.5, se muestra un ejemplo de un sector para un observador al que se le aplicaría esta primera forma. Donde en la línea central del sector en color verde se muestra unos puntos en color azul que son los puntos que se obtiene por la interpolación de los puntos más cercanos del MDE a la bisectriz del sector y que conformaran la curva de elevaciones. A la que se estudiara que zonas son visibles y se extrapola a todo el sector obteniendo las Cuencas Visuales. Con esta metodología se hace cálculo de las Cuencas Visuales sobre 9000 líneas centrales, ya que se analizan 3 sectores con 3 ángulos de apertura distintos sobre 1000 puntos como observadores.

2. Mediante el cálculo exacto de la visibilidad de cada uno de los puntos dentro

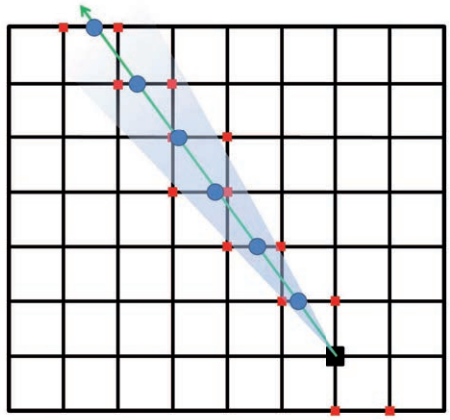


Figura 2.5: Sector de un observador. El observador es el cuadrado negro y el sector es el área dentro de la zona sombreada. La línea verde es la bisectriz del sector.

del sector. En este caso, el cálculo es mucho más costoso, ya que hay que trazar un segmento entre el observador y cada punto, para determinar su visibilidad. Al igual que en el caso anterior, donde se trabajaba con perfil sobre la línea central del sector, en cada uno de estos segmentos se traza un perfil de elevación consistente en la interpolación de los puntos más cercanos al segmento. Una vez obtenido el perfil, se pasa a estudiar si son o no visibles los puntos por el observador, para a continuación calcular la Cuenca Visual y compararla. Bajo esta metodología se analizan los puntos de 9000 sectores, mientras que con la anterior se analizaba solo los puntos de un eje central por sector.

Para comparar las diferencias entre las dos metodologías[2], a los resultados obtenidos por uno u otro método se le ha realizado la diferencia relativa respecto a la media entre los dos valores de las Cuencas Visuales calculadas.Los valores se han representado mediante histogramas. Así, cuando hay mayor coincidencia entre las Cuencas Visuales del sector usando una o otra metodología exhaustiva, el histograma se acumula a la izquierda (0 % de diferencia). Si el histograma se

[2]En todas las operaciones se presupone que el observador está situado a $1,5$ metros sobre el suelo, salvo que expresamente se especifique otro valor. Además, en el cálculo exacto se aplica una interpolación bilineal para la determinación de las alturas intermedias de los perfiles entre cada punto y el observador.

desplaza a la derecha (100 %), la diferencia es elevada. Los histogramas se han agrupado según el ángulo de sector de estudio ($S = 136^o, 158^o y 179^o$), según la distancia que alcanza la Cuenca Visual, y según el ángulo de apertura del sector. El la figura 2.6 se pueden ver los histogramas para el sector $S = 158^o$ donde cada una de las filas representa los histogramas de cada ángulo de apertura(0,5^o, 1^o y 2^o) y dentro de la fila cada uno es referente a la distancia de la visibilidad.

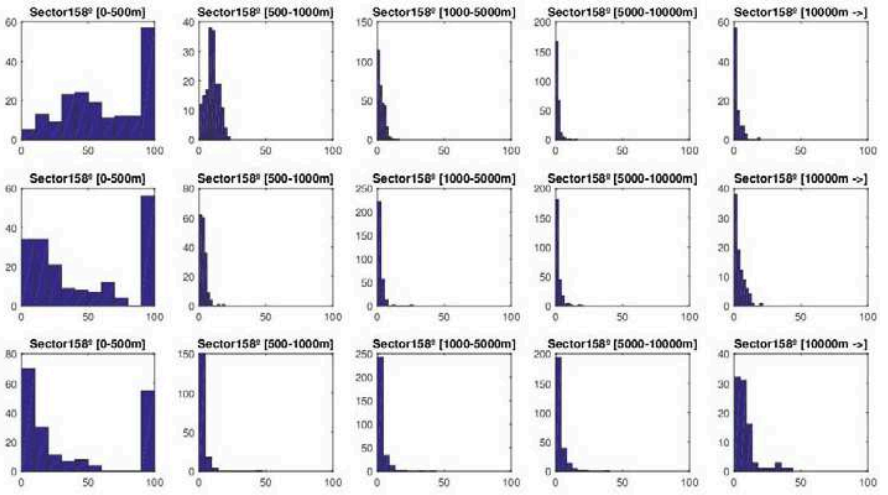


Figura 2.6: Histogramas de la diferencia relativa obtenido a partir de 1000 puntos en un solo sector de análisis. Sector estudiado con ángulo 158^o. Las filas corresponde con los ángulos de apertura de 0,5^o, 1^o y 2^o de arriba abajo.

Los resultados obtenidos, cuando la Cuenca Visual se extiende más allá de $1km$, corroboran claramente que el eje central de un sector es una muestra fiable de todo el sector, ya que una inmensa mayoría de los valores difieren menos de un 15 % prácticamente en cualquier caso. Este valor es considerablemente bueno. Para el rango $500 - 1000m$ también los resultados son buenos, pero resultan especialmente óptimos para sectores de 1^o respecto a 0,5^o y 2^o. Esta situación era de esperar para el caso del sector de 2^o de apertura, por lo explicado anteriormente respecto a la representatividad del punto central, ya que, cuanto más ancha es la amplitud del sector menor es la representatividad del eje central. Pero no tanto en el caso de 0,5^o, donde una vez analizados los casos particulares, se descubre que la explicación a esta situación se encuentra en la pobreza de la muestra de los valores empleados para el cálculo por el segundo método. Así, la visibilidad se calcula para apenas 5 o 6 puntos por el segundo método, mientras que en el

primero se incluyen valores de varias decenas de puntos cercanos al eje central, haciendo más fiable el valor obtenido. Resulta altamente curioso que en los puntos más próximos, el segundo método falla estrepitosamente, ya que la muestra de puntos en cada sector es excesivamente pobre, creando la necesidad de calcular la Cuenca Visual para el entorno próximo mediante algún algoritmo específico. Esta situación se ve enormemente afectada por la resolución del MDE, ya que, si se reduce la resolución de $10 \times 10m^2$ a $20 \times 20m^2$ el número de puntos cercanos en el sector con la misma apertura es menor de la mitad.

Además de los cálculos aquí comentados para la representatividad del eje central bajo ángulos de $0{,}5^o$, 1^o y 2^o, se han hecho cálculos para sectores de mayor ángulo (4^o o más), donde las diferencias ya empiezan a ser importantes, y la representatividad del eje central deja de cumplirse en terrenos muy abruptos. Tras los cálculos realizados se puede concluir que la elección optima es seleccionar 360 sectores equiangulares de 1^o.

Hay que reconocer que tanto el método de la bisectriz del sector, como el método exhaustivo, no dejan de ser aproximaciones de la solución real al problema.

2.1.3. Errores de los Algoritmos de Visibilidad

Bajo el marco de estudio de cualquier algoritmo SIG, basado en la utilización de MDEs, tanto de alturas como TIN, resulta importante tener en cuenta que ningún algoritmo dará un valor totalmente real del parámetro de visión que se desea calcular. Habrá algoritmos que proporcionen una aproximación mayor o menor al valor real dependiendo de la metodología del algoritmo y de las características de la superficie que analice el mismo. Por ejemplo, si la superficie a analizar es totalmente plana, el resultado real y el calculado podrán llegar a ser prácticamente iguales, al ser éstos lo más parecidos posible. En la computación de la visibilidad, el hecho de que la superficie de estudio sea totalmente plana es una situación que apenas se da, ya que en los terrenos totalmente planos la visibilidad de un observador es toda la superficie, por no haber ningún obstáculo que dificulte la visibilidad del observador.

El único interés que tiene el estudio de visibilidad sobre una superficie totalmente plana es poder corroborar el buen o mal comportamiento de los algoritmos, ya que en zonas que se conoce perfectamente cual debe ser su resultado, es fácil detectar fallos básicos de los algoritmos implementados. Lo habitual es la realización del cálculo de cualquier parámetro de visibilidad sobre terrenos que presentan cierta rugosidad e incluso alta rugosidad, para ser

analizados mediante herramientas SIG.

Tras analizar un elevado número de algoritmos de visibilidad en SIG, se puede concluir, que el resultado de todos los algoritmos de visibilidad basados en MDE pasados, actuales y posiblemente futuros, nunca arrojaran un resultado exacto al real ni en el caso de una superficie sin obstáculos. Esto es debido a tres hechos que afectan a todos los algoritmos que usan datos del mundo real y se computan en sistemas informáticos:

- El primer hecho viene generado por la naturaleza de los datos usados en los SIG, ya que al pasar del mundo real al digital se esta realizando una discretización en función de los niveles de digitalización empleados. Esto se traduce en errores, al igual que sucede con todo sistema que utiliza muestreo de variables continuas. El error de digitalización es dependiente del número de bits usados y del tipo de aproximación a los niveles de digitalización empleado en la discretización de los datos para la representación real.

- El segundo hecho es la resolución del muestreo de la superficie a estudiar o número de puntos del MDE. Está claro que si hay más información de un terreno menor sera el error que se cometa y viceversa, a menor número de puntos se podrá cometer un error mayor. En la figura 2.7, se presentan tres perfiles del mismo terreno, donde el primer perfil corresponde a un hipotético terreno de estudio, mientras que el segundo y tercer perfil de la figura son muestreos del mismo terreno a distintas resoluciones $R = 100m$ y $R/10 = 10m$; las unidades de la figura son en metros y se observa como en ninguno de los muestreos se asemejan al original. Lo que se si observa es como según se aumenta la resolución mejora el parecido al terreno original, pero aunque el número de puntos de muestreo tienda casi el infinito, siempre se cometerá un pequeño error por estar pasando de un espacio continuo de naturaleza fractal que es la superficie terrestre, a otro discreto con valores escalonados.

- El tercer hecho no procede de la obtención de los datos sino del procesamiento de los mismos, y es debido a los errores de redondeo de los sistemas de computación en la realización de las operaciones aritméticas durante la ejecución de los algoritmos. En este sentido pruebas realizadas en sistemas *multi-core* distintos, tanto en diferentes CPU como en diferentes GPU, han dado distintas soluciones para ejecuciones del mismo problema, empleando siempre el mismo algoritmo y mismo tipo de datos con igual número de bits. Las diferencias entre los resultados de un sistema a otro han sido pequeñas, de menos de un uno por ciento del valor. Pero el único

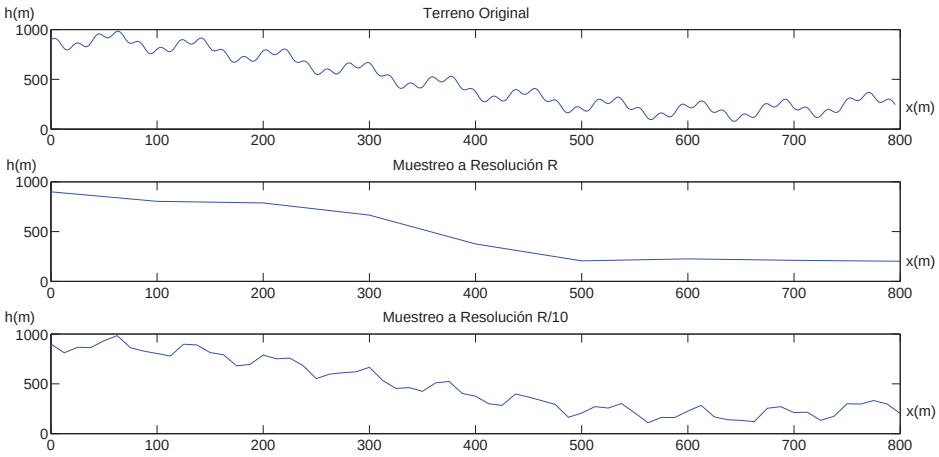


Figura 2.7: Perfil de un hipotético terreno, y sus equivalentes tras realizar un muestreo a distintas resoluciones, con base de resolución $R = 100$ *metros*.

factor que hace que sean diferentes es la estrategia de redondeo a la hora de hacer las operaciones en los distintos sistemas cuando existe operaciones que los decimales se salen del rango permitido por el tipo de datos usado.

Una vez mostrada la imposibilidad de conseguir que el resultado sea exactamente igual al real, ya que siempre se va a cometer un error. Queda la posibilidad de que el resultado obtenido mediante un algoritmo sectorial presente una mayor o menor variación con respecto al al valor real, en función de:

- La cantidad de puntos del MDE en el sector, bien mediante la resolución empleada para el MDE o por la interpolación de otros puntos.
- El número de sectores elegido para el cálculo.
- Del número de bits que usa el tipo de datos empleados en todos los cálculos.
- Del error de redondeo en las operaciones con equipos informáticos, por la arquitectura de los sistemas informáticos.
- De si se usa o no interpolación entre puntos cercanos a las líneas de los perfiles de estudio.
- Otros factores dependientes del tipo de algoritmo, como el radio máximo que se emplea o tipos de interpolaciones.

En conclusión estamos ante un problema de enorme complejidad y de resultado siempre aproximado, por lo que conseguir un producto suficientemente fiel para usarlo en un conjunto amplio de aplicaciones, es todo un éxito. Tras estas revelaciones no es de extrañar que haya autores [12, 31, 32] en la bibliografía, que consideran validos un resultado respecto a otro, si la diferencia en el cálculo del parámetro de visibilidad, como puede ser las Cuencas Visuales o Viewshed, difieren hasta el 50 % para el mismo punto u observador en el mismo MDE. Esta consideración no se comparte por los autores del presente trabajo, ya que para nosotros considerar que una variación de un 50 % es valida puede llegar a dar soluciones inadecuadas en varias de las aplicaciones de los SIG, como por ejemplo, la colocación del menor número de torres de vigilancia.

2.2. Visibilidad Total de un MDE

Todo lo descrito hasta ahora se refiere a la computación de algún parámetro de visibilidad para un solo punto o observador. Además se ha mostrado la efectividad del cálculo de los parámetros de visibilidad mediante el enfoque sectorial. Pero es a partir de ahora , cuando vamos a presentar el verdadero corazón del trabajo de esta tesis: un algoritmo que permite la computación eficiente de parámetros de visibilidad para todos los puntos de un MDE de alturas.

Se define como Visibilidad Total de un MDE de alturas con N puntos, al resultado de calcular un parámetro de visibilidad para todos los puntos del MDE, sin ninguna restricción limitadora de superficie o radio de estudio, salvo los propios limites del tamaño de superficie que abarcan los distintos MDE a estudiar. Es decir, que en el estudio de la visibilidad los puntos de los bordes del MDE no tendrán información de puntos de otros MDE, aunque, dichos puntos se encuentran muy cercanos físicamente. Esto hace que en los puntos de los bordes solo tengan información relevante en los sectores con direcciones interiores al MDE y su valor para las otras direcciones se verán truncados y arrojaran valores no validos.

Sobre los problemas de Visibilidad Total, apenas hay trabajos realizados, como ya se ha mencionado con anterioridad. Además hay algunos autores como [23] que aunque calculan la visibilidad para todos los puntos de un MDE, limitan el radio de estudio a unos cientos de metros. Esta solución, para muchas aplicaciones, es insuficiente por la necesidad de contemplar unos radios mayores. La limitación en la metodología de [23] reside en que, si se aumenta el radio de estudio, entonces se aumenta cuadráticamente el número de puntos a estudiar y los tiempos de cómputo se disparan enormemente de forma que no se obtienen los resultados en

unos tiempo razonables, ya que el algoritmo planteado consiste en la extrapolación de la solución de un punto a varios, sin introducir ninguna mejora en el cálculo de la visibilidad de un conjunto de puntos, ni hacer una gestión eficiente de la memoria, que contemple los fallos de caché que se producen en la misma.

La falta de algoritmos eficientes para el cálculo de Visibilidad Total, ha inspirado a que el algoritmo desarrollado en el presente trabajo se centre en este problema.

Antes de seguir con el propio algoritmo, es necesario presentar la clasificación realizada en el presente trabajo de los MDEs en función del tamaño de puntos de la malla que los representa, ya que el presente trabajo se ha visto en la obligación de realizar dicha clasificación de los MDE para poder tratar de forma eficiente los problema de visibilidad en las arquitecturas informáticas actuales. En la clasificación aludida, difiere los MDEs según el número de puntos que componen la malla del MDE, en la representación de un terreno y no en el tamaño de la superficie que cubre, ya que con los que trabaja los algoritmos es con un número de puntos. Si el número de puntos del MDE es muy elevado resulta imposible que la información de todos los puntos junto con las variables necesarias en la computación quepan en los distintos niveles de la memoria caché de un procesador, por exceder los limites de la misma. Por esta razón se hace la clasificación de forma que permita implementar el algoritmo sin problemas de almacenamiento en memoria, gracias a que los principales datos usados durante la ejecución del algoritmo, se adecuan a los tamaños típicos de los distintos niveles dentro de la jerarquía de memoria del equipo. Como se verá más adelante en el presente capítulo, MDEs con tamaño de 2000×2000 puntos (o menos) permiten que el algoritmo desarrollado tenga los datos más importantes dentro de la caché de nivel uno de los procesadores de gama media-alta; como consecuencia de ello se considera MDEs de tamaño grande si su dimensión supera este valor y medio si su tamaño es inferior o igual al mismo. En lo sucesivo, todos los cálculos en MDE de tamaño medio se utilizaran estos valores.

La gran mayoría de los trabajos relacionados con el presente, han realizado el estudio sobre mallas de puntos de pequeños territorios y con resoluciones no muy altas. Ningún trabajo hasta la fecha ha tratado el problema para una extensión de grandes dimensiones, como podría ser toda la superficie de la comunidad autónoma de Andalucía en una malla de $10 \times 10m^2$ de resolución, con un total de 1500 millones de puntos. La principal razón de esta situación es que si se extrapolase los tiempos presentados por algunos algoritmos relacionados a un MDE extremadamente grande, el tiempo de ejecución sería del orden de decenas de años de cómputo ininterrumpido.

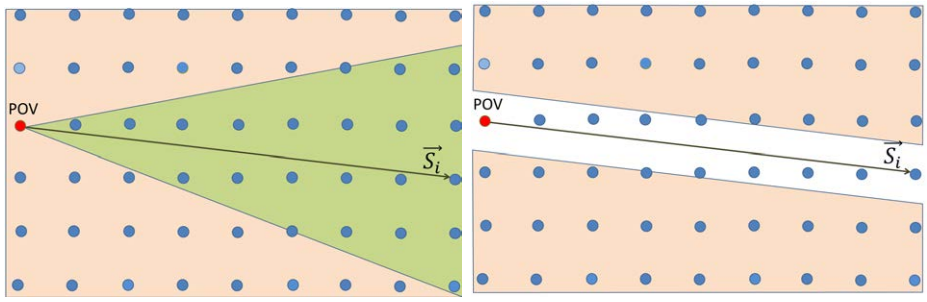
Para alcanzar una solución que permita el cómputo de MDEs de cualquier tamaño en unos tiempos razonables, se ha desarrollado en primer lugar un eficiente algoritmo capaz de computar la visibilidad de MDEs de tamaño medio, para a continuación extender el algoritmo a un segundo nivel donde sea capaz de analizar MDEs de tamaño grande sin limitaciones de tamaño de malla, que permita la explotación de las características de una implementación paralela y la ejecución sobre sistemas informáticos de alto rendimiento con arquitecturas *multi-core* y *multi-socket*, de forma altamente eficiente. Por esta razón, se ha dividido el presente apartado en las siguientes secciones: 1) *Algoritmo para MDE de tamaño medio, secuencial*, para aplicar a MDE de pequeño o mediano tamaño; 2) *Algoritmo para MDE de tamaño medio, paralelo*; 3) *Algoritmo paralelo para MDE de tamaño grande*; y 4) *Marco paralelo Algoritmos de Visibilidad SIG*. Esta última sección contiene una aclaración del paralelismo y como este no solo es aplicable al algoritmo aquí tratado sino para la mayoría de los algoritmos de visibilidad en SIG;

2.2.1. Algoritmo secuencial para MDE de tamaño medio

En esta sección se describe: i) el funcionamiento básico del algoritmo; ii)la estructura de datos esencial implementada, que se ha denominado BOS; dicha estructura permite una mayor eficiencia en la implementación del algoritmo para el cálculo de la visibilidad; iii) la gestión dinámica de la estructura BOS; iv) el cálculo de un parámetro atómico de visibilidad: el segmento visible elemento base (anillos sectoriales visibles) para el cálculo de parámetros de visibilidad y v) resumen de como es la implementación completa del algoritmo en pseudocódigo.

En la sección 2.1, se mostró como algunos autores han estudiado el problema del cómputo de la visibilidad por sectores y cómo estos han implementado principalmente dos estrategias diferentes. La primera estrategia hace un uso de los puntos que están sobre la línea central del sector, o una interpolación de los puntos que se encuentran próximos a la línea central en la dirección del sector($\vec{S_i}$), ya que puede darse el caso, que sobre la línea central del sector no haya ningún punto, produciendo un resultado altamente inexacto, por la exclusión de puntos relevantes en el cálculo de la visibilidad. Un ejemplo de esta situación se presenta en la figura 2.8a, donde se localiza un observador *POV* dentro de un MDE al que se desea calcular un parámetro de visibilidad. El sector de estudio con dirección $\vec{S_i}$ es la zona del MDE en tono verdoso y en cuya bisectriz se observa la línea central del sector que va desde el observador hasta el final del MDE. Se puede ver en la figura como solo hay un punto sobre la línea central del sector, mientras que hay puntos no muy alejados de la línea central y pertenecientes al sector

que no se tendrían en cuenta, por lo que se aplica un criterio de proximidad e interpolación de los puntos cercanos a la línea central del sector, en este tipo de algoritmos. La segunda estrategia de la bibliografía es usar todos los puntos del sector; por tanto, en el ejemplo de la figura 2.8a se analizaría todos los puntos de la zona verdosa para el cálculo de la visibilidad. Esta estrategia, requiere en la mayoría de los casos, un número extremadamente elevado de operaciones, que puede hacer necesario un tiempo de cómputo tan grande, que no se considere adecuado emplear esta estrategia para obtener el valor del parámetro deseado. Ésta es la razón por lo que la primera estrategia es la que se suele usar en la mayoría de los estudios aunque los resultados arrojados solo sean aproximaciones.



(a) (b)

Figura 2.8: Ejemplo de análisis sectorial y mediante *Band-Of-Sight* para el mismo observador POV.

El algoritmo desarrollado en el presente trabajo introduce una nueva metodología en el análisis de los MDE, ya que en lugar de usar los puntos del MDE de alguna de las dos formas como lo hacen las dos estrategias presentadas en el párrafo anterior, se usa una aproximación intermedia consistente en una ventana de puntos que se ha llamado Banda de Visión, del ingles *Band-Of-Sight* (BOS). En la figura 2.8b se muestra un ejemplo de análisis con BOS. En dicho ejemplo se busca obtener la visibilidad para el sector $\vec{S_i}$, sobre el mismo observador (POV) localizado en el mismo MDE que se empleo en el ejemplo del párrafo anterior de la figura 2.8a. Se observa en el ejemplo una banda con fondo blanco y orientación $\vec{S_i}$. Dicha banda es la ventana BOS y los puntos interiores a la misma son los que utilizaría nuestro algoritmo para el análisis de alguno de los parámetros de visibilidad del observador POV. Además en la figura 2.8b se puede observar, al igual que en la figura 2.8a, la línea central del sector que va desde el observador de estudio POV al extremo opuesto según la dirección $\vec{S_i}$ del sector.

Para la computación de un parámetro de Visibilidad Total sobre un MDE de análisis, el algoritmo desarrollado en la presente tesis realiza una serie de barridos sobre todos los puntos del MDE a estudiar mediante las iteraciones de dos bucles anidados. Una simplificación de estos dos bucles se muestra en el pseudocódigo 2.1, donde aparecen dos bucles *for*. Además se puede observar en dicho pseudocódigo como el primer bucle (externo) hace un barrido sobre los sectores a analizar. El número de iteraciones de este primer bucle es típicamente igual al número total de sectores (nsec=360). En el segundo bucle, interno, se realiza la parte mayoritaria de las operaciones del algoritmo, ya que una vez fijado por el bucle exterior el sector de análisis, el bucle interno recorre todos los puntos del MDE a estudiar y para cada uno de ellos calcula el parámetro de visibilidad deseado, solo para los puntos del sector preseleccionado por el bucle externo. De forma que el número de iteraciones que realiza el bucle interno es equivalente al número total de puntos del MDE (N). La realización del cálculo del parámetro de visibilidad deseado dentro del segundo bucle, se realiza mediante la llamada a la función *ComputarVisibilidad(POV,s)*. En esta función es donde se realizan las operaciones matemáticas necesarias para computar el parámetro de visibilidad deseado del observador POV.

Pseudocódigo 2.1: Bucles Anidados Básico.

```
for s=0 to s<nsec do
  for POV=0 to POV<N do
    ComputarVisibilidad(POV,s);
  endfor
endfor
```

Pseudocódigo 2.2: Bucles Anidados.

```
for s=0 to s<nsec/2 do
 Ordenar_Puntos(s);

 so=s+180;
 for POV=0 to POV<N do
  Actualizar(BOS);
 /*Visibilidad en dirección S_i*/
  v(POV,s)=ComputarVisibilidad(POV,s);
 /*Visibilidad en dirección -S_i*/
  v(POV,s)=ComputarVisibilidad(POV,so);
 endfor
endfor
```

En el pseudocódigo 2.2 se presenta una descripción más completa de la implementación del algoritmo desarrollado, donde se puede observar, como el primer bucle presenta un número total de iteraciones igual a la mitad del número total de sectores de estudio ($\frac{nsec}{2} = \frac{360}{2} = 180$ iteraciones). Este hecho se debe a que la ventana BOS en cada iteración del bucle interno posee la información de los puntos del MDE para el análisis de dos sectores sobre el observador POV. En concreto la información contenida por la ventana BOS para un observador POV, son los puntos del MDE que se localizan como máximo a una cierta distancia

trasversal de la línea central de los sectores con dirección normal de estudio $\vec{S_i}$, y con dirección puesta $\vec{S_i^o}$ o $-\vec{S_i}$. La distancia transversal es la que indica lo ancho que es la ventana BOS, ya que si la ventana es estrecha la distancia sera pequeña y viceversa. Al tener información de los puntos en las direcciones $\vec{S_i}$ y $-\vec{S_i}$, se calcula la visibilidad para cada uno de los sectores mediante la llamada a la función *ComputarVisibilidad(POV,s)* para el sector con dirección normal y *ComputarVisibilidad(POV,so)* para el sector opuesto, cuya dirección es la misma que el normal más un giro de 180 grados.

Además de seleccionar el sector en el bucle externo, se realiza una ordenación de todos los puntos del MDE dependiente del sector de estudio. Esta ordenación es una de las bases del algoritmo desarrollado, ya que indica el orden el el cual se van a recorrer todos los puntos del MDE para hacer el barrido del sector fijado. El orden en el cual se recorren los puntos es dependiente del sector de estudio y se obtiene mediante la llamada a la función *Ordenar_Puntos(s)*. Terminada la ordenación, se entra en el segundo bucle, donde en cada iteración además de seleccionar el observador o punto al que se calcula la visibilidad POV, se actualiza la estructura de datos que almacena los puntos de la ventana *Band-Of-Sight* mediante la llamada a la función *Actualizar(BOS)*, que usa los datos obtenidos mediante la función *Ordenar_Puntos(s)* y otros que se calculan en la propia función *Actualizar(BOS)* .

Es en la actualización de la estructura BOS es donde reside la clave de la gestión de la memoria y los datos por nuestro algoritmo, permitiendo reducir significativamente los tiempos de ejecución del algoritmo desarrollado frente a otros algoritmos. Sin esta estructura y su correcta gestión, dentro de la función *ComputarVisibilidad()* no se podría calcular el parámetro deseado de forma eficiente, ya que es dentro de la *ComputarVisibilidad(POV,s)*, donde se recorre los puntos de la ventana BOS para la obtención del parámetro deseado sobre un observador POV en la dirección del sector de estudio impuesta por la variable s.

Es importante tener claro que con esta metodología no se obtiene el valor total de la visibilidad para un punto, en todos los sectores, hasta que no se finaliza los dos bucles del algoritmo. Y al finalizar los dos bucles del algoritmo, el resultado es la visibilidad total del MDE de estudio. Si se desease calcular exclusivamente el valor de visibilidad para un observador, bastaría con poner una condición que evitara realizar la función *ComputarVisibilidad()* si el punto seleccionado no es el deseado, de forma que en el bucle interior solo realiza las tareas de actualización de la estructura BOS y el análisis de la visibilidad para el punto seleccionado, mientra que el exterior hace el recorrido de todos los sectores. Por tanto, se puede apreciar como el algoritmo desarrollado esta pensado para optimizar el cálculo de la visibilidad total de un MDE pero no para la visibilidad de un solo punto.

Se podría pensar, de que sería más eficiente usar otra forma de anidamiento en los bucles para realizar el recorrido de los puntos y sectores. Es decir, que el bucle exterior sea el encargado de recorrer todos los puntos del MDE, y una vez seleccionado un observador el bucle interior calcule para dicho observador el parámetro de visibilidad, haciendo el recorrido por cada uno de los sectores, en vez de elegir un sector y recorrer todos los puntos. Esta otra forma de anidamiento permitiría seleccionar el observador de forma más rápida y haría más sencillo la computación de la visibilidad para un solo punto arbitrario. Sin embargo, la técnica propuesta permite que se puede reutilizar mucha de la información de la estructura de datos que contiene la ventana BOS en iteraciones consecutivas del bucle interior, ya que observadores próximos comparten más de la mitad de los puntos entre sus ventanas BOS. Un ejemplo, de este hecho se presenta en la figura 2.9, donde en las dos imágenes de la figura se observan como son las ventanas BOS para dos observadores *POV* distintos sobre el mismo MDE y mismo sector de estudio. Se puede observar en la figura como los observadores son vecinos y como sus ventanas BOS son casi iguales a diferencia de dos puntos. Esta situación permite que la ejecución del algoritmo para la computación de visibilidad total sea mucho más rápida, si se fija primero el sector y luego se analiza para cada punto su visibilidad, ya que las estructura BOS va a necesitar una menor modificación con el anidamiento elegido, que con el uso de otro tipo de anidamiento, haciendo que se reduzca el número de fallos en caché y permitiendo optimizar el uso de la jerarquía de memoria, al reutilizar la mayoría de los datos de una iteración a otra del bucle interior.

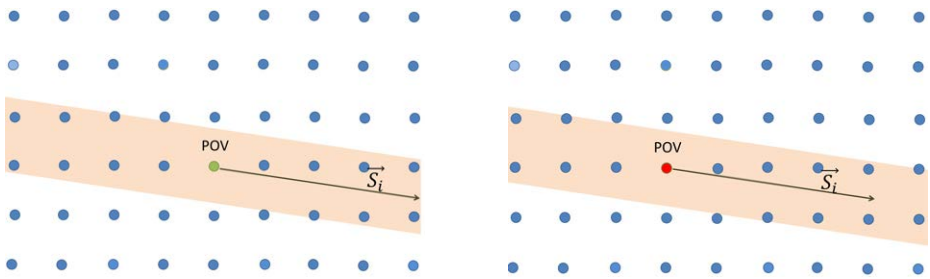


Figura 2.9: Ejemplo de dos posibles ventanas Band-Of-Sight de dos distintos observadores POV.

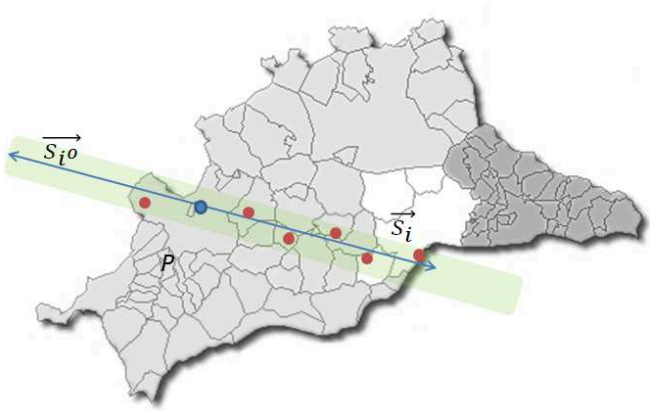
Volviendo a la implementación del algoritmo presentando en el pseudocódigo 2.2, la función *ComputarVisibilidad()* realiza el cálculo de los parámetros de visibilidad deseados, sobre un punto seleccionado *POV*, en

tres fases para un determinado sector s fijado por el bucle exterior.

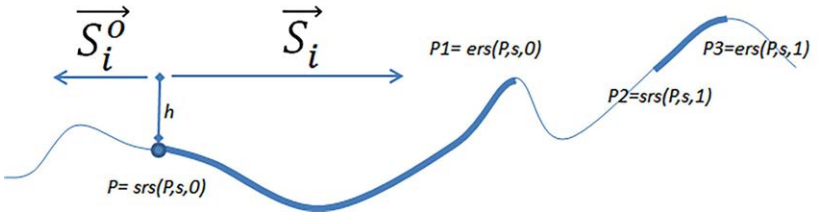
En la primera fase se realiza principalmente operaciones de gestión de datos donde para cada una de las direcciones $\vec{S_i}$ se construye un perfil de puntos en el plano vertical sobre la línea central del sector perteneciente al observador POV. El perfil es realizado a partir de los datos del MDE localizadas en la estructura BOS y que conforman la ventana de visión para el punto POV. Estos datos han sido introducidos en la estructura BOS previamente por la función *Actualizar(BOS)* siguiendo un orden preestablecido por la función *Ordenar_Puntos(s)* del bucle exterior. El perfil creado a partir de los datos de las estructura BOS es equivalente al perfil de puntos de los algoritmos LOS, pero con la información de la ventana BOS, necesaria para el cálculo de los parámetros de visibilidad en un sector y su opuesto. A continuación, en la segunda fase se analizan los puntos del perfil para determinar si dichos puntos son visibles o no por el observador. Las secuencias de puntos colindantes visibles que se obtienen sobre el perfil creado, conforman tramos o segmentos visibles, que quedan perfectamente delimitados por el primer punto del segmento visible y por el último punto del segmento. La última fase de la función *ComputarVisibilidad()* se calcula el parámetro deseado con la información de los segmentos visibles o no visibles y una serie de operaciones matemáticas correspondientes al parámetro que se desea calcular.

Un resumen gráfico de las dos primeras fases de la función *ComputarVisibilidad()* se pueden ver en la composición de figuras 2.10. En dicha composición se presenta primeramente la figura 2.10a donde existe un punto o observador P (en color azul) perteneciente a un terreno, sobre el que se quiere calcular el parámetro de visibilidad en los sectores S_i y $-S_i$. El territorio la figura 2.10a pertenece a la provincia de Málaga, España. Para calcular el parámetro deseado sobre P se estudian los puntos(en color rojo) de la banda o ventana BOS (sombreada en verde). Estos puntos en la computación se localizan dentro de la estructura BOS y van a permitir la generación de un perfil de superficie como el de la figura 2.10b, mediante la proyección de la altura de los putos sobre el eje central siendo esta una representación vertical del plano de estudio. Por medio del perfil de puntos se puede determinar los segmentos visibles del sector S_i, estos segmentos son presentados en la figura 2.10b con un grosor de línea mayor y delimitados por los puntos P y $P1$ para el primer segmento y $P2$ y $P3$ para el segundo segmento visible. Aunque en los segmentos visibles suele haber varios puntos más, no suele ser necesaria la información de los mismos. La figura 2.10c muestra una reconstrucción en 2D, sobre el plano horizontal, del resultado del cómputo de las Cuencas Visuales o Viewshed, para el sector de estudio. Es decir, que dicha figura presenta la superficie visible

para un observador P en el sector estudiado, utilizando el concepto de los anillos sectoriales visibles, donde la zona en azul oscuro es la superficie visible o anillo-sectorial visible como tal y la zona más clara es la zona no visible sobre el sector de estudio. Una explicación detallada de los anillos sectoriales y su generación se muestra en la sección 2.2.2 del presente capítulo.



(a) BOS para el punto P sobre un MDE



(b) Perfil de una BOS para el punto P con una Altura h

(c) Generación de anillos sectoriales del perfil de la figura (b) en la dirección $\vec{S}_i$

Figura 2.10: Computación de la visibilidad, BOS, perfil y anillos sectoriales sobre un punto en dirección $\vec{S}_i$.

La implementación eficaz del algoritmo sectorial, basado en dos bucles, necesita el uso de una organización de datos apropiada en la *Band-Of-Sight*

que permita obtener resultados de forma eficiente. En las siguientes subsecciones se muestra la organización y gestión de los datos pertenecientes a los puntos introducidos en la *Band-Of-Sight*.

La banda de visibilidad, BOS

El presente trabajo ha denominado ventana de visión, *Band-Of-Sight* o simplemente ventana BOS a una ventana rectangular de puntos dinámica con la misma orientación que la dirección principal $\vec{S_i}$ del sector de estudio. Dicha ventana es dinámica, al ir desplazándose por todo el MDE de estudio, según se realizan las iteraciones del algoritmo; de forma que realiza un barrido por todos los puntos del MDE para poder computar los parámetros de visibilidad deseados.

En cada iteración del bucle interno del algoritmo, la ventana se ve modificada seleccionando en cada iteración un nuevo punto de observación, hasta completar la totalidad del modelo, y se adecua la misma al observador de estudio según la dirección del sector que se está analizando ($\vec{S_i}$). El barrido de puntos del MDE, como se verá más tarde, se realiza en la dirección perpendicular al sector de estudio ($\vec{S_i^T}$). La ventana BOS como ya se ha mencionado, debe de contener los puntos tanto en la dirección $\vec{S_i}$ como en la dirección opuesta $\vec{S_i^o}$, partiendo en cada dirección desde un observador hasta llegar a los bordes del MDE en las direcciones de análisis. Se puede, por tanto, decir que la ventana *Band-Of-Sight* es equivalente al ensanchamiento de la línea central de un sector y su opuesto para en cálculo de la visibilidad de un solo observador. En la figura 2.11, se muestra un ejemplo de ventana BOS donde se presenta una malla de puntos correspondiente a un hipotético MDE. La ventana BOS corresponde a la parte sombreada de la figura en tono azul y limitada por dos líneas también en color azul. Sobre la ventana se muestra como sería el perfil de superficie o orografía del terreno para el área que ocupa la ventana BOS.

Además de la orientación de la ventana BOS para cada sector de estudio, el tamaño de la ventana es otra de las principales características de la misma, ya que este parámetro va a determinar por un lado la precisión de los resultados y por otro cuanto espacio de memoria ocupara la estructura que contenga la ventana BOS para poder ser procesada.

Al tamaño de ventana se le ha llamado *bw*, del ingles *bandwith*, y su dimensión debe ser capaz de abarcar el número suficiente de puntos en una dirección $\vec{S_i}$ y su opuesta $-\vec{S_i}$ o $\vec{S_i^o}$ respecto al punto al que se le calcula la visibilidad. Al estar tratando con un MDE resulta que un conjunto de puntos es equivalente a una superficie sobre el modelo, de forma que podríamos limitar el tamaño de la

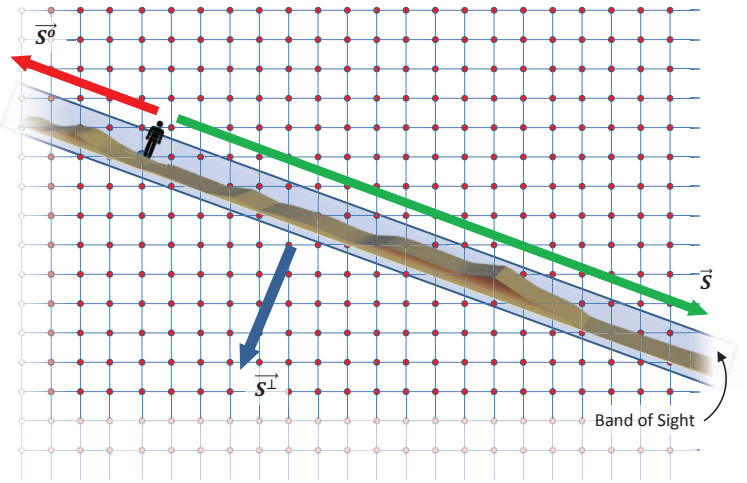


Figura 2.11: Los puntos visibles de un observador en dirección $\vec{S}$ y su opuesto $\vec{S^o}$ son calculados mediante el análisis de los bw puntos de la BOS (el área sombreada).

ventana BOS por su número de puntos, en vez de basarnos en la superficie que los mismos representan sobre un MDE. Por ejemplo, si el MDE tiene una resolución de $10x10m^2$ y se quiere tener una ventana BOS que cubra una superficie de $1km^2$, resulta que es similar a tener una $bw = 1Km^2/10 \times 10m^2 = 100 \times 100 = 10000$ puntos

El número de puntos que debe contener la ventana, bw, debe ser del orden $\sqrt{n}$, siendo n el número total de puntos de un MDE. La elección de este tamaño de ventana es consecuencia de que la línea central de un sector de estudio puede presentar, en el mayor de los casos $\sqrt{n}$ puntos sobre ella, al estar trabajando con MDE cuadrados de tamaño $\sqrt{n} \times \sqrt{n}$. Esta situación, de número de puntos máximo en la línea central de un sector, se da exclusivamente en los casos que el punto de estudio este en uno de los bordes del MDE siendo el sector de estudio completamente perpendicular al borde más cercano al punto o cuando el punto de estudio se localiza en una de las esquinas del MDE y el sector de estudio es bien diagonal a los bordes o paralelo a ellos; ya que de esta forma, la línea central sector tiene el equivalente a una fila o columna completa de puntos del MDE, que presenta un tamaño de $\sqrt{n}$. Para otras localizaciones de los puntos de análisis

sobre un MDE de alturas y para otras orientaciones de los sectores de estudio, el número de puntos sobre la línea central del sector es inferior a $\sqrt{n}$, siendo mayoritariamente nulo el número de puntos entre el observador y un punto final. En la figura 2.12, se muestran los puntos que conforman el MDE de un terreno y cuyo tamaño es de $n = 10x10 = 100$ puntos. Además en la figura se han dibujado lo que podrían ser varias de las líneas centrales de distintos sectores de estudio para un observador localizado en el extremo superior izquierdo del MDE. Esta figura, permite corroborar todo lo mencionado en el presente párrafo, en cuanto que los puntos de los bordes son los únicos que presentan algunos sectores donde sus líneas centrales alcanzan una cantidad de puntos igual a $\sqrt{n}$, cuando el sector de estudio es perpendicular al borde del MDE y en el caso de puntos de las esquinas como el de la figura cuando es diagonal.

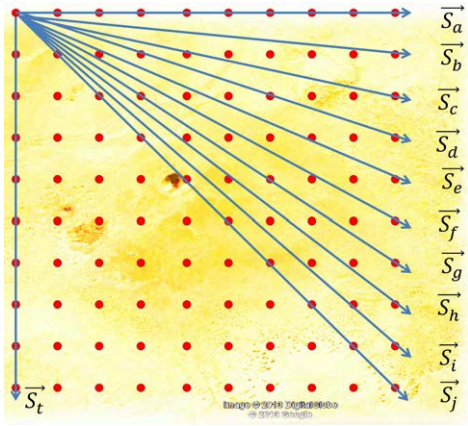


Figura 2.12: Varias líneas centrales de varios sectores.

Si el tamaño de ventana seleccionado es mayor o igual a $\sqrt{n}$, se está asegurando que se realizara el análisis de los sectores con un mayor número de puntos que en el mejor de los casos considerados por los algoritmos LOS. De esta forma, la ventana BOS permitirá no perder la información de puntos relevantes y por tanto, se alcanzara la obtención de un resultado suficientemente preciso tanto en los casos en los que los puntos se encuentren o no sobre la línea central del sector. Si el tamaño *bw* es demasiado bajo con respecto a la dimensión del MDE, la calidad del resultado disminuye. Siempre hay que tener en cuenta que si *bw* es demasiado alto se corre el riesgo de que la estructura que contiene la ventana BOS no quepa totalmente en la caché del procesador en el que se realice la computación, con lo cual se producirían un elevado número de fallos en memoria, produciendo, por tanto, perdida en ciclos de computación, por traer de nuevo el

dato a la caché desde una de las distintas capas de la jerarquía de memoria del equipo informático sobre la que se guarda la estructura BOS.

Se ha comprobado experimentalmente con el algoritmo desarrollado, que tamaños de la ventana BOS entre el 50 % y el 200 % de $\sqrt{n}$ proveen resultados validos para la mayoría de las aplicaciones de visibilidad, en un tiempo razonable de pocos minutos. El limite de tamaño inferior dependerá de la precisión deseada en los resultados, ya que si es muy bajo no se tiene información suficiente para una buena aproximación, mientras que el limite superior de la ventana BOS dependerá principalmente del tamaño de memoria del sistema donde se ejecute y del tiempo de ejecución dispuesto a tolerar en la obtención de los resultados, ya que se tardara más tiempo al analizar más puntos. Por otro lado, si el tamaño de la ventana es muy alto se corre el riesgo de que puntos cercanos al observador y que se encuentran en otros sectores de análisis, los computen el algoritmo como si pertenecieran al propio sector, lo que produciría errores leves en los resultados, ya que, aunque no son puntos pertenecientes al sector, sus valores de alturas no difieren normalmente mucho por la suavidad del terreno. Un estudio más profundo puede determinar cuál es el tamaño optimo en función del ángulo de apertura de los sectores, aunque para ello habría que disponer de datos reales de visibilidad, lo que actualmente es inviable [40]. Por este motivo, utilizamos como límite de superior del 200 % de $\sqrt{n}$.

Una vez seleccionados los puntos del MDE para el cálculo de la visibilidad de un observador en una dirección y su opuesta mediante la ventana BOS con tamaño fijo bw, estos puntos permiten la generación de un perfil de puntos para el estudio de la visibilidad.

El perfil de puntos que se obtiene a partir de la ventana BOS en una dirección de estudio, es generado mediante el recorrido de los puntos dentro de la ventana en dirección del sector de estudio, desde el observador hasta el extremo de la ventana en esa dirección, siguiendo el orden de los puntos más cercano al más lejano del observador. La figura 2.13 muestra un ejemplo, donde se presenta parcialmente un MDE sobre el que se calcula la visibilidad de un observador (POV), dentro de una ventana BOS delimitada por dos líneas paralelas en color azul con dirección de estudio ($\vec{S}_i$). En dicha figura se ve el orden de recorrido de los puntos, para la posterior generación del perfil, según dicho orden.

Un pequeño resumen gráfico de la generación del perfil de puntos citado en el párrafo anterior se presenta en la figura 2.14, donde se puede observar como los puntos recorridos dentro de una ventana BOS son proyectados sobre la línea central del sector. Esta proyección es el paso previo a la generación del perfil vertical y el orden en el cual los puntos son proyectados sobre la línea central,

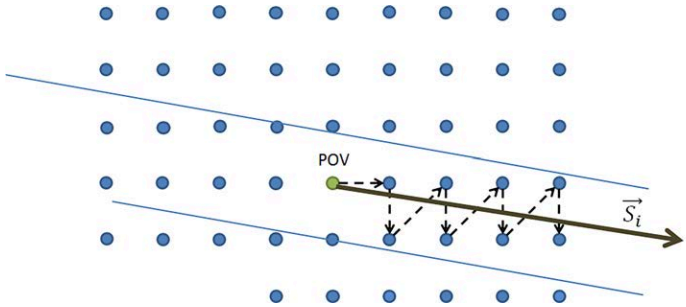


Figura 2.13: Forma de recorrer los puntos de una BOS para la generación de un perfil de puntos del observador POV según la dirección $\vec{S_i}$ del sector.

se realiza acorde a la distancia real de los puntos respecto al observador, de forma que la misma distancia que hay del observador al punto real, que entre el observador y la proyección del mismo sobre la línea central del sector. Una vez realizada la proyección de los puntos sobre la línea central del sector el siguiente paso es la generación del perfil de puntos en el plano vertical, para ello se usa la información de la altura de cada uno de los puntos y el resultado que se puede obtener es un perfil como el que se muestra en la parte inferior de la figura 2.14. En dicha figura se puede observar como el perfil de estudio generado está compuesto por los puntos de la ventana BOS que se encuentran a una distancia concreta de la línea central del sector, delimitada dicha distancia por el tamaño *bw* elegido para la ventana BOS. En las figura 2.15 se muestra una situación con un tamaño de ventana *bw* mayor que en la figura 2.13, donde se aprecia como la ventana es más ancha y como ha variado el orden en el que se recorren los puntos debido a que estos se ordenan para la generación del perfil dentro de la ventana BOS según la distancia que presenta el punto de la ventana respecto al observador (POV). Otro detalle que se puede extraer de la figura 2.13 es que el primer punto del nuevo perfil que se genera, siguiendo el orden de cercanía al observador POV, estaría con toda seguridad fuera de un sector con un ángulo de apertura de un grado, al igual sucede con los puntos 4 y 6. De esta forma, se añade información de puntos de otros sectores que, dependiendo de la rugosidad del terreno y de la resolución del MDE, pueden variar más o menos los resultados respecto del valor real. Este hecho demuestra que bajo la sectorización por *Band-of-Sight*, el tamaño de la ventana BOS no debe ser muy grande para no cometer el error de introducir puntos de otros sectores vecinos en el análisis de puntos cercanos al observador.

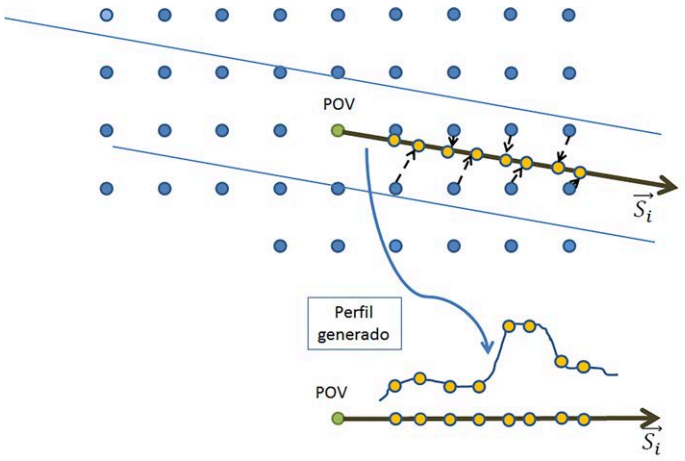


Figura 2.14: Perfil generado sobre la línea central de la Banda de Visión.

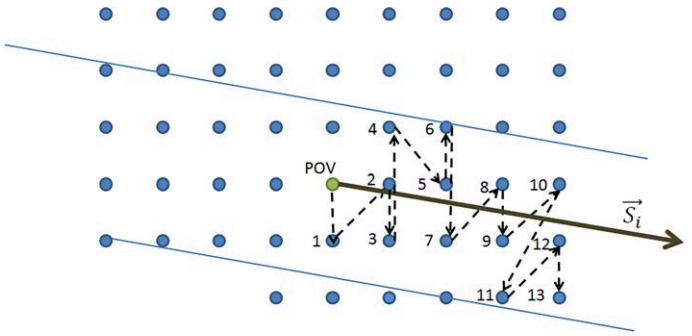


Figura 2.15: Segundo recorrido para banda más ancha.

Comportamiento dinámico de la BOS

Para llegar a comprender la estructura interna de la BOS es necesario previamente conocer, como evoluciona dinámicamente durante la ejecución del bucle interno asociado a un determinado sector.

Como dijimos anteriormente, el bucle interno recorre todos los puntos del MDE en dirección perpendicular (S^T) al sector de análisis (S). En cada iteración, el POV se sitúa exactamente en el centro de la BOS medido según la dirección transversal del sector de estudio(S^T). Para ello, conviene que los puntos vayan internamente orientados en esa dirección. Pero el cálculo de la visibilidad de cada

POV, necesita analizar los puntos de la BOS en dirección longitudinal, desde el propio POV hasta los extremos de la banda. Para ello, conviene que los puntos estén internamente orientados en esa dirección.

La estructura de datos que mejor se adecua a estos, aparentemente contradictorios, requerimientos es la lista doblemente enlazada, ya que permite el análisis de los elementos en ordenamientos no correlacionados, como veremos a continuación.

La BOS como lista enlazada

Para poder operar con los puntos de la ventana BOS dentro de un sistema informático, es necesario que la información de los mismos se almacene dentro de la memoria de forma eficiente. Dicha tarea se lleva a cabo mediante la implementación de una estructura de datos, que permite la gestión eficiente de la memoria donde se almacena la ventana BOS. La estructura desarrollada consiste, como se ha descrito antes, es una lista doblemente enlazada, la cual posee una cola circular. Esta estructura facilita la gestión dinámica en el tratamiento de los datos referentes a los puntos de estudio, ya que permite el ordenamiento de los puntos según mejor convenga, al poder introducir los puntos dentro de la estructura siguiendo un orden (S o S^T) y después recorrer los puntos dentro de la estructura con el mismo orden o con otro, ya que los punteros de la lista permiten seguir un orden y la posición en memoria de cada elemento de la lista otro. Cada elemento de la estructura es un nodo con la información de cada uno de los puntos de la ventana de visión y los enlaces necesarios para el elemento siguiente y el anterior de la lista. En la figura 2.17 se puede ver un esquema visual de la forma que tendría la estructura BOS.

La información que se almacena en cada uno de los nodos son 4 parámetros de cada punto, de forma que cada nodo esta formado por 6 componentes, que se describen a continuación:

1. **nodo.idx**: Es el identificador del punto (idx). Este parámetro es exclusivo de cada punto, y su valor es el orden del punto según la lectura del fichero de entrada representativo del MDE a estudiar. Es decir el valor de idx del primer punto es $idx = 0$, del segundo punto $idx = 1$ y así sucesivamente. Los puntos del MDE están ordenados dentro del fichero de oeste a este y de norte a sur de forma que el primer punto es el superior izquierdo del MDE y el siguiente punto por debajo del primero ($idx = 0$) en un MDE de dimensiones $m \times mpuntos$ es $idx = m$. De esta forma, el siguiente punto de la primera columna del MDE que se localiza debajo del punto con identificador

$idx = m$ es el punto con la identificador $idx = 2xm$ dentro del fichero. Por otro lado, el identificador idx permite calcular la ubicación geográfica de cualquier punto conociendo las coordenadas del primer punto del MDE y la resolución del MDE. Es fácil pasar del identificador idx, a las coordenadas planas UTM gracias a que la malla del MDE y la distancia de puntos permanece constante. Por ejemplo, se dispone de un MDE con tamaño de $n = 2000 \times 2000$, el primer punto tendría $idx = 0$, el siguiente a la derecha $idx = 1$ y el siguiente para abajo sera $idx = 2000$ por leerse el fichero por filas y tener la primera fila identificador del 0 al 1999. De esta forma, si la resolución es $R = 10 \times 10m^2$ y las coordenadas UTM del primer punto son $(406000N, 36000E)$, entonces las coordenadas UTM de cualquier punto con identificador idx es igual a $406000 + (RedondeoInferior(\frac{idx}{\sqrt{n}} \cdot)\sqrt{R})$ N , $36000(Resto(\frac{idx}{\sqrt{n}}) \cdot \sqrt{R})$ E.

2. **nodo.d**: El segundo parámetro es la coordenada o distancia d del punto. Y es la resultante de la rotación del eje de referencia del MDE en la dirección del sector, tomando como origen el punto con $idx = 0$. Si X y Y son las coordenadas UTM normales de los puntos, X^s y Y^s representa las coordenadas rotadas o referenciadas al sector s siendo X^s el eje de referencia principal, por lo que el parámetro d es el valor de la coordenada X^s del punto. En la figura 2.16 se muestra como es la rotación de los ejes para un sector con dirección $\vec{S_i}$ y como es la variable d.

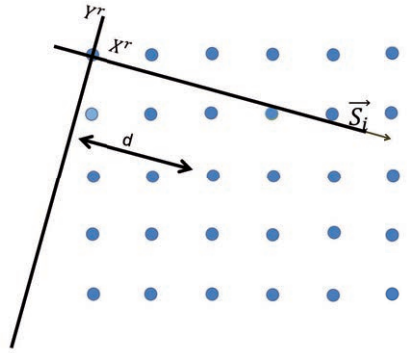


Figura 2.16: Rotación coordenadas en función del sector.

Este parámetro d, permite calcular fácilmente el orden en el que van a recorrer los puntos dentro de la ventana, ya que basta con recorrerlos desde el observador hasta el punto que presenta una mayor d. Por otro lado, si

la ventana es estrecha (número de puntos reducido frente al tamaño total del MDE) resulta que la distancia entre dos puntos se puede aproximar por la diferencia de sus coordenadas X^s, es decir restando directamente sus valores de d.

3. **nodo.h**: La altura del punto respecto al nivel del mar y en metros (h). Como se puede comprender es el parámetro más importante en la generación de los perfiles de puntos y del cálculo de la visibilidad.

4. **nodo.oa**: Es el último de los parámetros almacenados, llamado oa, representa el orden auxiliar con el cual se pueden recorrer los nodos. Si los nodos son guardados en memoria según el orden S^T entonces oa tendrá el valor correspondiente al número de orden en dirección S. Si los puntos son guardados en memoria según el orden S entonces oa tendrá el valor según el orden S^T.

5. **nodo.sig**: es un puntero con la información de cual es el nodo siguiente, según el orden de la dirección del sector S.

6. **nodo.prev**: es un puntero con la información de cual es el nodo anterior, por lo que sigue la dirección $-S$.

Siempre hay que tener en cuenta que los puntos del modelo son barridos según la dirección $\vec{S^T}$ del sector de estudio y los puntos dentro de la ventana se deben recorrer según la dirección $\vec{S}$ para la obtención del perfil a analizar. Si se elige que los nodos estén ordenados en memoria según el orden S, entonces el bucle interno, responsable de casi la totalidad del tiempo de ejecución, accede de forma lineal a la memoria, mientras que la inserción de nuevos elementos implica una reubicación completa de la información en la memoria para poder mantener el orden S. Sin embargo, si los elementos se introducen en memoria siguiendo el orden S^T, entonces la inserción y eliminación de un nuevo elemento es muy sencilla, ya que al ser la lista de cola circular, basta con eliminar el último elemento de la lista y introducir uno nuevo. El problema de esta segunda metodología reside en la necesidad de encontrar la ubicación exacta de cada nuevo elemento según la dirección S, por lo que hay que recorrer la lista utilizando el paramento $nodo.oa$, pero no es necesario la modificación de más información que la de los punteros, por lo que este ordenamiento resulta más eficiente que el primer caso.

Por tanto, los puntos son ordenados acorde a su posición en memoria siguiendo el orden S^T, desde el punto o nodo más antiguo PO hasta el más nuevo PN. De forma que en la fase regular, tras pasar la fase de llenado de la estructura, en cada iteración se inserta un nuevo punto en la estructura según se realiza el barrido

del MDE en la dirección S^T al sector de estudio. La inserción del nuevo punto consiste en machacar la información del nodo más antiguo, con la inserción de la información del siguiente punto, pasando a estar ubicado el nuevo nodo el la posición que ocupaba el punto más antiguo y moviendo la cabeza y cola de la lista enlazada. Además de la inserción de la información del nuevo punto se calculan las posiciones de los punteros *sig* y *prev* para mantener el orden de recorrido de los puntos según la dirección del sector.

El observador POV, al que se le calcula la visibilidad en una dirección y opuesta, está situado en el centro de la estructura circular. Los punteros *prev* y *sig* mantiene el orden de los nodos según la dirección del sector de estudio y van desde PF a PL, de forma que, si se parte del punto POV y se hace el recorrido de los nodos con los puntero *sig*, se llega hasta el nodo PL en dirección S y si se hace el recorrido con los nodos *prev* se alcanza el nodo o punto PF.

Anteriormente se mostró cómo la ventana, y por tanto, la estructura BOS, debía tener un número de puntos del orden $bw = \sqrt{n}/2$.

Para un MDE de 2000×2000 la estructura BOS debería tener un tamaño de unos 2000 puntos para cumplir con la recomendación de que bw debe ser del orden de $\sqrt{n}$, por otro lado, para que exista un punto central bw debe ser impar; así, $bw = 2 \cdot hw + 1$. Con estas dos premisas, un buen tamaño para la estructura BOS es de $bw = 2001$ puntos, con lo cual hw sería 1000. Ahora bien, pruebas realizadas muestran que si $bw = 1001$ la estructura BOS ocupa un tamaño aproximado de unos 16 KBytes, que cabría en la caché de datos de nivel 1 de una cpu actual típica montada en un equipo sobremesa. Esta reducción (de $\sqrt{n}$ a casi $\sqrt{n}/2$) apenas afecta a la precisión de los resultados, pero mejora significativamente los tiempos de ejecución.

Una vez presentada la estructura BOS, qué datos se deben almacenar dentro de ella y cual debe ser su tamaño, se pasa a mostrar como es la gestión de la estructura.

Gestión de la estructura BOS

La gestión de la estructura que contiene la ventana BOS busca que se adecúe lo más rápido posible el contenido de la estructura a las necesidades de cómputo del algoritmo de visibilidad. Esta tarea se consigue gracias a que el número de cambios dentro de la estructura es el menor y más sencillo posible de una iteración a otra del bucle interior del algoritmo desarrollado.

La estructura de datos representada por la lista enlazada se encuentra vacía

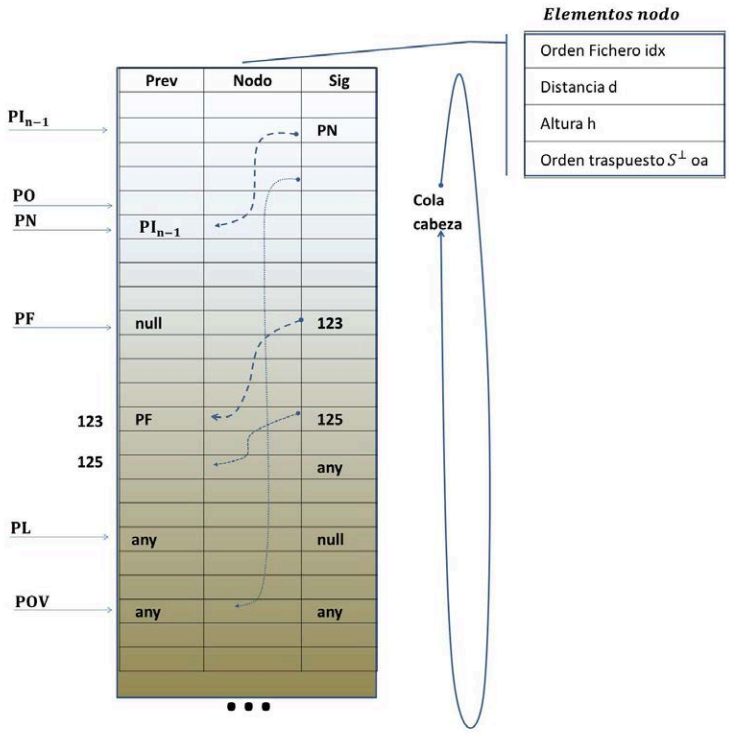


Figura 2.17: Estructura BOS con los puntos activos, donde PO es el punto más antiguo y PN el más reciente. Se mantiene el doble enlazado de la lista por los punteros *Prev* y *Sig*, manteniendo un ordenamiento de los puntos desde PF hasta PL.

al inicio del cálculo de los parámetros de visibilidad. Es decir, la estructura no se llena antes de comenzar con la ejecución del algoritmo, sino que es en la propia ejecución cuando se realiza el llenado de la estructura, en concreto se realiza en las primeras iteraciones del algoritmo (iteraciones del bucle interior); de esta forma, no se pierde ciclos de computación en llenar previamente la estructura.

Estas primeras iteraciones, es lo que se ha denominado fase de llenado, donde se realiza la introducción de un único punto en la estructura en la primera iteración del bucle interior del algoritmo, y se realiza la introducción de dos nuevos puntos en la estructura por cada una de las siguientes iteraciones del bucle interior, hasta llenar totalmente la estructura. De esta forma, se mantiene

siempre el observador POV en el centro de la estructura, y la estructura crece dinámicamente de manera controlada hasta el tamaño bw predefinido.

En la ejecución del algoritmo, una vez está llena la estructura y sin haber llegado a pasar por la misma todos los puntos del MDE, se cambia de la fase de llenado a la denominada fase o estado regular. En esta fase, por cada iteración del bucle interior del algoritmo se añade solo un nuevo punto y se elimina otro que resulta ser el punto más antiguo dentro de la estructura según se va realizando el barrido del MDE. De forma que la estructura BOS se comporta en esta fase como una estructura *FIFO*. Al solo eliminar y insertar un punto, el coste computacional es muy bajo, ya que basta con eliminar el elemento más antiguo de la lista e introducir uno nuevo, que se sabe a priori cual va a ser, por haber calculado a priori como es el barrido de los puntos fuera del bucle interior. De forma que una vez introducido el nuevo punto el la localización del más antiguo de la estructura basta con modificar los enlaces entre los nodos que conforman la estructura para tener la ventana BOS correspondiente al nuevo observador de estudio y de esta forma, volver al proceso de generación del perfil de puntos para el análisis de la visibilidad. Es previsible, por tanto, que se produzca un fallo cache por iteración en esta fase y además, es perfectamente previsible saber a que dato de memoria afecta.

Una vez han pasado por la estructura todos los puntos del MDE y por tanto, se han computado la visibilidad para casi todos los puntos del MDE para el sector de análisis específico, la estructura BOS se encuentra en la última fase, llamada de vaciado. En esta fase, faltaran hw puntos a los que calcular su visibilidad y se produce un desalojo de los puntos que se encuentran en la estructura de una forma similar a la fase de llenado, ya que en cada iteración se eliminan dos puntos, correspondientes a los más antiguos. La realidad es que estos puntos no son eliminados de la estructura, sino que simplemente no se reconocen al realizar la reordenación de los enlaces de los nodos que se recorren la estructura para luego generar el perfil de puntos.

El usar esta estrategia de llenado, desplazamiento y vaciado permite, por una parte, no perder tiempo en realizar un llenado previo a la computación y por otro lado, que una vez llenada la estructura, en cada iteración del bucle interno se elimine el punto más antiguo de la estructura BOS, que es además el punto más lejano a la línea central del sector según la dirección $-S^T$, siendo la dirección S las dirección del sector de estudio y S^T la dirección en la que se realiza el barrido de todos los puntos del MDE a analizar.

Otro de los puntos fuertes del algoritmo desarrollado es que el barrido de los puntos del MDE, se realiza siempre en dirección S^T cuando se está realizando

el análisis de la visibilidad para todos los puntos en una pareja de sectores con direcciones S y $-S$. Es decir que para el cómputo de la visibilidad en cada sector de estudio por medio del algoritmo desarrollado, el orden de entrada y salida de los puntos sobre la estructura de datos BOS es dependiente del sector y no sigue el orden natural del fichero de entrada como podría esperarse. Por ejemplo, si se estuviese analizado un sector con dirección noroeste-sureste $\vec{S_{ns}}$ el cálculo de la visibilidad se realiza empezando por el punto superior derecho o el punto más al nordeste del MDE, en vez del primer punto del fichero según el orden natural, que es el punto al noroeste o superior izquierdo. El punto superior derecho se encuentra en el borde del MDE y si se analiza este punto en un sector con dirección de ángulo 1^o resulta que, por encontrarse en el borde del MDE, no tiene puntos del MDE a analizar, ya que los puntos más cercanos en la direcciones de estudio corresponderían a puntos localizados fuera del MDE, tanto para la dirección del sector como para la dirección opuesta. En la siguiente iteración del barrido de puntos, según S^T, se agregan dos puntos quedando el siguiente punto de análisis en el centro. En una nueva iteración y siguientes el proceso es el mismo, quedando el punto (POV) sobre el que se computa su visibilidad en el centro de la lista. Esta forma de analizar los datos facilita la generación del perfil vertical del punto de estudio POV y el introducir los datos dentro de la estructura BOS. En la fase de desplazamiento el punto más antiguo de la estructura es $PO = POV - hw$ y el más nuevo $PN = POV + hw$, siguiendo el orden de introducción dentro de la lista, y siendo POV el punto de observación o de estudio, que es el único que estará justo en el centro de la banda. La aportación más notable de esta estrategia es que en cada iteración nos permite calcular la visibilidad recorriendo fácilmente todos los puntos de la estructura siguiendo los enlaces de la lista. Además siempre que se recorra un MDE cuadrado y de tamaño fijo, tenemos que para cada sector, el orden en que se recorre los puntos se mantiene de una ejecución a otra. De forma que sobre cualquier otro MDE con el mismo número de puntos y el mismo tipo de malla o distribución de puntos, se tendrá el mismo recorrido, aunque la resolución de los MDE sea distinta. Lo que cambiará de un análisis a otro sera las posiciones reales de los puntos de los MDE y las alturas de los puntos del MDE, que harán que los resultados finales sean distintos. Por tanto, bajo esta premisa (mantener de unas ejecuciones a otras el orden en el que se recorren los puntos de un MDE) si previamente para cada sector se calcula como va a recorrerse los puntos según el barrido S^T y se guarda la ordenación o forma de recorrido en un archivo, no habrá que volver a realizar el cálculo del orden de inserción de los puntos en la estructura, con lo que el coste computacional para la ordenación de los puntos resulta ser prácticamente nulo.

La figura 2.18 muestra un ejemplo de ventana BOS sobre un MDE, en el

cual se realiza el barrido de puntos para el sector de estudio S. La ventana de la figura corresponde a la fase regular del barrido del MDE, es decir cuando la estructura esta llena y en cada iteración entra y sale un punto de la misma. En la figura se puede ver todos los puntos más relevantes desde la ventana, tales como el observador o punto de estudio (POV), el nuevo punto (PN) que entraría en la nueva iteración y el punto más antiguo (PO) saldría según el orden S^T y los puntos más alejados desde el observador en la dirección del sector y su opuesta (PL y PF).

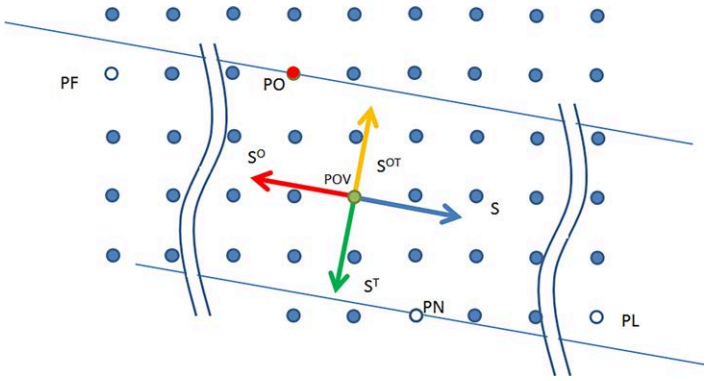


Figura 2.18: Direcciones de ordenamiento y puntos más relevantes.

Una vez estén los puntos dentro de la estructura BOS se pasa a extraer la información relevante para realizar el cálculo de la visibilidad. El primer paso es la localización de los segmentos visibles, con los cuales se pueden fácilmente obtener los llamados anillos sectoriales visibles por el observador, que limitan las zonas de la superficie visible dentro de cada uno de los sectores.

2.2.2. Cálculo de los anillos sectoriales visibles

Tras tener la información, de los puntos de la ventana de visión, almacenada dentro de la estructura BOS, se dispone del del perfil vertical del sector de análisis para un observador (POV) ; el siguiente paso del algoritmo es determinar que puntos del perfil son o no visibles y de esta manera, determinar cuales son los segmentos visibles del perfil del observador. Para saber si un punto P_j es visible o no por un observador POV, se puede usar el método de los algoritmos LOS.

En la figura 2.19, se puede apreciar un ejemplo de un hipotético perfil de puntos para un sector de estudio S_i con dirección $\vec{S}_i$, donde se encuentra el

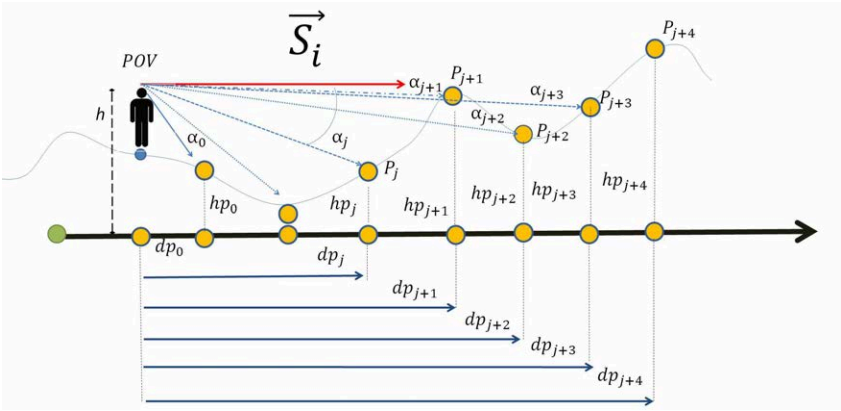


Figura 2.19: Visibilidad o no de los puntos de estudio.

observador POV con altura h, al que se desea realizar el cálculo de su visibilidad dentro del perfil. La altura h del observador POV es la suma de la altura del observador respecto la superficie terrestre más la altura de la superficie respecto al nivel del mar. Además, se puede apreciar en la figura 2.19 la información de las alturas (hp_j) y de las distancias al observador (dp_j) para cada uno de los puntos del perfil. Esta información es la única necesaria para el cálculo de los segmentos visibles, ya que permite determinar si los puntos son o no visibles para el observador POV, bajo el método de los algoritmos LOS. Este método con la información de las alturas y distancias calcula los denominados ángulos de visión (α_i). El ángulo de visión de un punto (P_i) localizado sobre un perfil vertical de estudio, es el ángulo que forma la línea que une el punto con el observador respecto el plano horizontal del observador. Los algoritmos LOS dicen que un punto P_i es visible si su ángulo de visión (α_i) es mayor o igual al ángulo de visión del último punto visible anterior en el perfil de estudio. Para poder llevar a cabo esta metodología de comprobación de que puntos son o no visibles, se considera que siempre el punto más cercano al observador en la dirección de estudio es visible. Además, la situación de que el primer punto del perfil siguiente al observador sea siempre visible por el mismo, no es una suposición, sino un hecho, ya que el observador vera siempre el punto más cercano al el por no haber otro punto que obstaculice la visibilidad.

Siguiendo con el ejemplo de la figura 2.19, se puede entender como actuá la metodología de los algoritmos LOS. El punto P_j del perfil posee un ángulo de visión α_j respecto al plano horizontal que contiene el observador POV y resulta

que este punto es visible por no haber nada que obstaculiza la visión directa entre el observador y el punto P_j, situación que se corrobora visualmente al igual que se haría matemáticamente al ser α_j mayor a todos los ángulos de visión anteriores. De igual modo el siguiente punto P_{j+1}, es visible al ser el valor de ángulo α_{j+1} mayor a α_j, que era el ángulo del último punto anterior visible. Sin embargo, el punto P_{j+2} no es visible por estar obstaculizado por el punto anterior, hecho que se corroboraría matemáticamente al poseer ángulo α_{j+2} menor a α_{j+1}. El punto P_{j+3} tampoco es visible, porque aunque no es obstaculizado por el punto P_{j+2}, si está obstaculizado su visión respecto del observador por el punto P_{j+1}, hecho que se justifica matemáticamente al ser el ángulo α_{j+3} mayor que α_{j+2}, pero menor que α_{j+1}, que es el ángulo del último punto anterior visible.

Con el ejemplo mostrado en el párrafo anterior queda demostrado como el cálculo de visibilidad, en los algoritmos LOS, recae en el ángulo de visión (α). De forma sencilla se puede calcular el ángulo α de los puntos de un perfil para un observador mediante el uso de las reglas trigonométricas. Siendo el ángulo α la *arcotangente* del cociente entre la diferencia de alturas y distancias entre el observador en el punto de análisis. De forma que el numerador es la diferencia entre la altura del punto de análisis(hp_j) y la altura del observador (h), y el numerador es la distancia del punto de análisis al observador. Luego la expresión matemática del ángulo de visión es:

$$\begin{aligned} \alpha &= \arctan \frac{\Delta h}{\Delta d} \\ \Delta h &= hp_j - h \\ \Delta d &= dp_j \end{aligned} \tag{2.1}$$

Para comprobar si un punto es o no visible mediante la metodología del ángulo de visión, no es necesario en realidad saber el valor exacto del ángulo α con todos sus decimales, sino que es suficiente con saber si en ángulo del punto de estudio actual es mayor, igual o menor al ángulo del último punto visible anterior. Teniendo en cuenta esta consideración y que los ángulos que forman los puntos del perfil presenta valores entre -90 y 90 grados decimales, pero nunca llegan a los extremos de los 90 o -90 grados, ya que nunca un punto del perfil va a estar en la misma posición del observador, donde se daría el caso de valores extremos; entonces no es necesario calcular en ángulo α, sino que se puede usar la relación directa entre $\alpha = \arctan \frac{\Delta h}{\Delta d}$ y la simple división $\frac{\Delta h}{\Delta d}$, ya que resulta que si el valor de la división aumenta, aumenta el del ángulo de visión y si disminuye también disminuye el valor del ángulo de visión. Por tanto, para saber si el ángulo de un

punto es mayor o igual al del último punto visible, basta con saber si el cociente $\frac{\Delta h}{\Delta d}$ es mayor o igual a la del último punto visible. Con esta metodología se ahorra cómputo de CPU, por no realizar el cálculo de la *arcotangente* que consume más ciclos. Además para la división se puede utilizar operaciones nativas de baja precisión, dada la naturaleza del algoritmo.

A la división $\frac{\Delta h}{\Delta d}$ correspondiente al cálculo del ángulo α se le ha llamado ángulo-equivalente en el presente trabajo, para distinguirlo del ángulo de visión real α, resultante de aplicar la *arcotoangente* sobre dicha división.

Un conjunto de puntos visibles consecutivos forman lo que se denomina segmento visible en la dirección de estudio. Normalmente un observador para cada sector con dirección $\vec{S_i}$ va a presentar varios segmentos visibles, donde cada segmento está delimitado por el primer (SRS_{k_i}) y último (ERS_{k_i}) punto del segmento. Los puntos SRS_{k_i} y ERS_{k_i} se llaman de apertura y cierres respectivamente, siendo el identificador k_i el número del segmento visible dentro un sector S_i.

La búsqueda de los segmentos visibles sobre un perfil se realiza de forma computacional mediante la aplicación de unas sencillas condiciones o reglas sobre los puntos del perfil construido para el sector de análisis. Estas reglas son:

- Si el punto P_j es visible y el punto anterior P_{j-1} no lo es, entonces se abre un nuevo segmento visible k_i, siendo el punto P_j el punto de apertura SRS_{k_i} del nuevo segmento visible.

- Si el punto P_j es visible y el anterior P_{j-1} es también visible, entonces se está dentro de una zona visible y se guarda en ángulo-equivalente del punto P_j, por ser este ahora el último punto visible.

- Si el punto P_j no es visible y el punto anterior P_{j-1} tampoco es visible, entonces se está dentro de una zona no visible y no se realiza ninguna acción.

- Si el punto P_j no es visible y el anterior P_{j-1} si es visible, entonces el punto $Pj-1$ es el último punto visible del segmento visible actual y pasa a ser punto de cierre ERS_{k_i} del segmento actual k_i.

- Si el punto P_j es visible, el punto anterior P_{j-1} es también visible y pero P_j es el último punto del perfil en la dirección de análisis, entonces el punto P_j es un punto de cierre ERS_{k_i} del segmento k_i.

- Si el punto P_j es visible, el punto anterior P_{j-1} no es es visible, pero P_j es el último punto del perfil en la dirección de análisis, entonces el punto P_j

es a la vez punto de apertura SRS_{k_i} y punto de cierre ERS_{k_i} del segmento k_i.

En la ejecución del algoritmo se almacenan los puntos SRS_{k_i} y ERS_{k_i}, para, por un lado poder hacer representación gráfica de la visibilidad del observador analizado y por otro su utilización en posteriores cálculos. El guardar la información de los segmentos visibles hace que se tenga también la información de los segmentos no visibles o conjunto de puntos consecutivos no visibles, ya que estos están limitados por los puntos de cierre de los segmentos visibles y los puntos de comienzo de los siguientes segmentos visibles. Esta información es importante cuando el objetivo sea la búsqueda de las zonas menos visibles, como es el caso de buscar zonas aisladas o zonas de difícil visibilidad para esconder equipos militares en conflictos bélicos.

Hasta ahora los segmentos visibles tienen un carácter unidimensional por el propio perfil sobre el que se han calculado. Pero sin embargo en la realidad los segmentos visibles representan la superficie visible sobre el sector que se está analizando, y este tiene un carácter bidimensional. Es decir un segmento visible determina una superficie visible del sector que se está analizando, donde las longitudes de las superficies visibles dentro del sector son las longitudes de los segmentos visible mientras que la anchura de la superficies son los arcos o longitud de arcos de los comienzos y final de los segmentos visibles del propio sector, como se puede ver en la figura 2.20. Cada una de las superficies visibles dentro de un sector se le llama anillo-sectorial visible por no ser un anillo completo de las longitudes de los comienzos y finales de los segmentos, sino una porción del mismo correspondiente al ángulo de apertura del sector. La figura 2.20 presenta un ejemplo de los anillos sectoriales en la dirección $\vec{S}$ para un observador POV sobre un MDE. Además, en la figura se observa que hay tres anillos sectoriales delimitados cada uno por sus puntos iniciales SRS_k en color verde y sus puntos finales ERS_k en color carne.

El cálculo del área de un anillo-sectorial se puede realizar directamente mediante el uso de la fórmula matemática para el cálculo de áreas de anillos circulares. De forma que si el anillo-sectorial comienza en SRS, termina en ERS y el ángulo de los anillos sectoriales es de un grado, entonces el área que encierra cada anillo sectorial es la diferencia de las áreas del circulo delimitado por ERS menos el área del circulo delimitado por SRS dividido entre 360 para obtener el equivalente del arco de un grado y multiplicado por el ángulo de apertura en grados del sector(Ap):

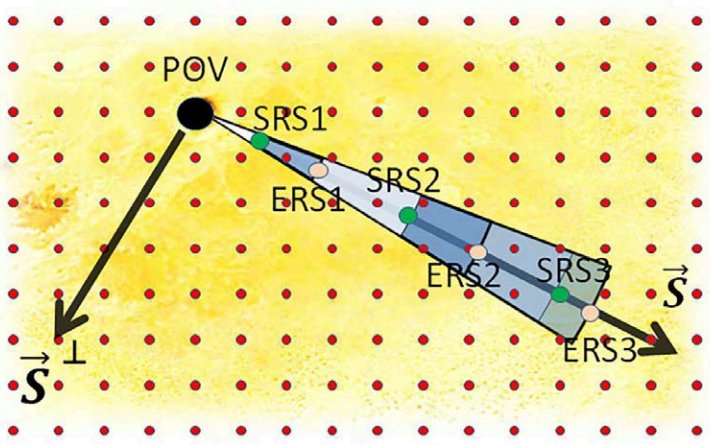


Figura 2.20: Anillos sectoriales del observador POV en la dirección $\vec{S}$ con los puntos de comienzo del anillo SRS_K y de final ERS_k.

$$A_{RS} = \frac{Ap \cdot pi \cdot ERS^2 - SRS^2}{360} \tag{2.2}$$

Una vez encontrados los segmentos visibles, que se traducen en anillos sectoriales, únicamente queda calcular los parámetros de visibilidad deseados por el usuario y que se muestran en los apartados del capítulo 3.

Implementación algoritmo secuencial para MDE de tamaño medio

Se han contado hasta ahora las claves de la implementación del algoritmo desarrollado en su versión no-paralela mediante el pseudocódigo simplificado del mismo, y cómo se realiza la gestión eficiente de los datos y el cálculo de las zonas visibles de la superficie de un terreno mediante el análisis de los anillos sectoriales. Para la compresión total del algoritmo desarrollado queda describir como se han unido todas las partes. Y esta es la tarea que se lleva a cabo en la presente sección, donde se plasma todo lo visto en las secciones anteriores, bajo la implementación del algoritmo desarrollado.

El pseudocódigo 2.3 muestra el resumen de la implementación completa del algoritmo secuencial. Donde el primer paso de todo el proceso de cómputo es la lectura de los datos del fichero que contiene el MDE. A continuación se pasa a la reserva en memoria del espacio necesario tanto para la estructura BOS de tamaño *bw*, como para las demás estructuras y variables auxiliares. Todo ello se realiza dentro de la función *Crear_BOS_va(bw)*. Una vez pasado este punto se comienza con la parte de computación del algoritmo de visibilidad mediante la ejecución de los dos bucles anidados. En el primer bucle(externo) se hace el barrido de los sectores de análisis. El número de iteraciones de este bucle es la mitad del número de sectores(minsec=360/2 = 180), ya que en cada iteración del segundo bucle (interno) se estudio la visibilidad para un sector y el opuesto.

El siguiente paso en el bucle externo es obtener el orden en el cual se van a recorrer los puntos del MDE tras la selección del sector de análisis, ya que el barrido de los puntos se realiza en dirección traspuesta ($\vec{S^T}$) a la dirección del sector ($\vec{S}$). Este orden de recorrido de los puntos se realiza mediante la llamada a la función *Ordenar_Puntos()*. Tras saber el orden de recorrido de los puntos se ejecuta el barrido mediante el segundo bucle *for*, que en cada iteración selecciona un punto al cual se le calcula la visibilidad.

Tras seleccionar un punto mediante el bucle interno al cual calcular la visibilidad se actualiza la estructura BOS para contener los puntos del sector de estudio y el opuesto mediante la llamada a la función *Actualizar_BOS()*. Esta función simplemente realiza la gestión de la estructura mediante la introducción e eliminación de puntos y la asignación de los valores a los punteros para poder recorrer la ventana en un sentido o otro según la dirección del sector fijado por el primer bucle. Es decir que dentro de esta función se realiza el segundo ordenamiento de los puntos mediante los punteros, para poder generar los perfiles en la dirección del sector y su opuesta. Este ordenamiento consume ciclos de computación, pero se puede precalcular antes y guardarlo en un fichero, ya que siempre que se seleccione un sector sobre un MDE de mismo tamaño de puntos, estos se recorrerán de igual forma.

Una vez está actualizada la estructura BOS se pasa a realizar el cálculo de la visibilidad en un sector y luego en su opuesto mediante la función *ComputarVisibilidad()*. Esta función trabaja con los parámetros del punto seleccionado *POV*, los datos dentro de la estructura BOS y el número de sector, para saber si calcula la visibilidad de un sector o su opuesto. La función *ComputarVisibilidad()* calcula que puntos de la ventana de visión son o no visibles, para determinar cuales son los segmentos visibles de cada sector y en el caso que sea necesario calcular los anilos-sectoriales y los parámetros de visibilidad impuesto por el programador, como las Cuencas Visuales y Visibilidad

Volumétrica para el punto POV en el sector s prefijado.

Una vez finalizado los dos bucles se dispondrá de los resultados de visibilidad deseados para todos los puntos del MDE, los cuales a continuación serán almacenados para su posterior tratamiento y visualización. Por último, se hace una limpieza en memoria de los recursos empleados en la ejecución del algoritmo por si se desea hacer nuevas ejecuciones en un espacio limpio, usando para ello la función *deleteBOSandaux*(). Tras todo este trabajo el algoritmo ha finalizado.

Pseudocódigo 2.3: Algoritmo básico de visibilidad.

```
LeerMDE(higth);//Lectura del MDE de alturas

Crear_BOS_va(bw);//Generar la estructura BOS y variables auxiliares
for s=0 to s<=minsec do // Recorrido de sectores
 //Ordenar los puntos del MDE según la dirección de barrido
 Ordenar_Puntos(s);
 //Barrido para el cálculo de visibilidad de cada punto del MDE
 for POV=0 to POV<N do
  Actualizar(BOS);//Actualización de la estructura BOS

  so=s+180;//sector opuesto
  ComputarVisibilidad(POV,s);// sector normal
  ComputarVisibilidad(POV,so);//sector opuesto
 endfor
endfor

Almacenar(parámetros calculados);

deleteBOSandaux();//Liberar espacio usado
```

Dentro de la función *ComputarVisibilidad()* se localiza el núcleo del cómputo matemático del algoritmo, ya que todo lo realizado hasta este punto por el algoritmo han sido operaciones de gestión y ordenación de datos, mientras, que en esta función se realiza todas las operaciones necesarias para el cálculo de la visibilidad. Por una parte, esta función analiza que puntos son visibles mediante la metodología de los ángulos equivalentes y por otra parte se calcula los anillos sectoriales junto otros parámetros de visibilidad, mediante la aplicación de las operaciones matemáticas correspondientes. En el pseudocódigo 2.4 se muestra la implementación simplificada de la función *ComputarVisibilidad()*.

La función *ComputarVisibilidad()* se divide en dos partes una de recorrido de los puntos de la estructura BOS y otra de cálculo de los parámetros deseados. Esta última parte de cálculo la realiza dentro de la función *kernel()* en el pseudocódigo 2.4, que es donde se hacen las verdaderas operaciones matemáticas. El recorrido de los puntos de la ventana de visión guardados dentro de la estructura BOS es dependiente del sector que se analiza, si este es en dirección normal, la variable contadora de sectores s sera inferior a 180, y si es en dirección opuesta su valor sera superior a 180 al ser $so = s + 180$. Si el sector que se

Pseudocódigo 2.4: Pseudocódigo función ComputarVisibilidad().

```
ComputarVisibilidad(POV,s)
  float d_POV=POV.d;//Posición del observador
  float h_POV=POV.h;//Altura del observador
  bool visible_segmento=false;
  float maxanguloanterior= -∞ //Máximo ángulo anterior

  currP=POV;
  if s<180 then
    while currP != PL do//recorrido puntos dirección normal
      currP=currP.sig;
     Kernel(d_POV,h_POV,currP.d,currP.h,&max,&visible_segmento);
    endwhile
  else
    while currP != PF do//recorrido puntos dirección opuesta
      currP=currP.prev;
      Kernel(d_POV,h_POV,currP.d,currP.h,&max,&visible_segmento);
    endwhile
  endif
end ComputarVisibilidad


Kernel(d_POV,h_POV,d_P,h_P,&maxanguloanterior,&visible)
  float anguloequivalente = (h_P-h_POV)/(d_P-d_POV);
  bool es_visible = anguloequivalente >= maxanguloanterior;
  bool empieza_segmento = es_visible && !visible_segmento;
  bool fin_segmento = !es_visible && visible_segmento;

 if empieza_segmento then
    //Incremento del número de anillo-sectorial visibles
    Incremento(n_anillo_sectorial);
    //Almacenamiento del comienzo del segmento visible en un array
    Almacena(d_P);
  endif

  if fin_segmento then
    //Almacenamiento del final del segmento visible en un array
    Almacena(d_P);

    //Cálculo de los anillos sectoriales véase ecuación 2.2
      .....

    //Cálculo de otros parámetros de visibilidad deseados
      .....
  endif

  //Guarda le estado visible punto, para la siguiente iteración
  visible_segmento=es_visible;/
  //Actualización del máximo ángulo para próxima iteración
  maxanguloanterior = max(anguloequivalente,maxanguloanterior)
end Kernel
```

analiza es en dirección normal entonces se recorren los puntos de la estructura BOS mediante los puntero *sig* hasta alcanzar el último punto en la dirección normal que es el punto *PL* y si el sector que se analiza es el opuesto entonces se

realiza el recorrido usando los punteros *prev* hasta alcanzar el punto *PF*, que es el último en la dirección opuesta. Para ello se recorren mediante un bucle *while* los elementos de la estructura siguiendo los punteros *sig* o *prev* dependiendo de si el sector que se analiza es en dirección normal o opuesta. El elemento de la estructura, al cual se analiza en cada una de las iteraciones su visibilidad por medio del bucle *while*, se denominado el el pseudocódigo como *currP*.

Dentro de la función *kernel()* lo primero que se calcula es es ángulo-equivalente del punto que se está analizando, llamado en el pseudocódigo *anguloequivalente*, mientras que a el ángulo-equivalente del último punto anterior visible se ha llamado *maxanguloanterior*. Para calcular *anguloequivalente* simplemente se realiza el cociente entre la diferencia de alturas entre el observador(h_POV) y el punto que se analiza (h_P) en el numerador y la distancia entre el observador y el punto (d_P-d_POV) en el denominador.

A la hora de darle un valor inicial a la variable *maxanguloanterior* existen principalmente dos posibilidades o estrategias. La primera es darle el valor del primer punto de análisis en el perfil, siendo necesario para ello un cálculo previo al recorrido de los puntos del perfil y comenzar en el segundo punto. La segunda y la cual se ha implementado por su sencillez y validez es darle a *maxanguloanterior* un valor altamente negativo del que se sabe que a priori el ángulo-equivalente nunca va alcanzar dicho valor. De esta forma, se asegura que el ángulo-equivalente del primer punto es mayor al valor inicial del *maxanguloanterior* y por tanto, este primer punto se computa como visible, que es la situación real como se mostró en la sección 2.2.2.

La variable booleana que indica si el punto actual *currP* es visible o no se ha llamado *es_visible* en el pseudocódigo 2.4. Otra variable booleana importante es *visible_segmento* que indica si el punto anterior era o no visible por el observador (*POV*). Con estas dos variables booleanas y la aplicación de las reglas expuestas en la sección 2.2.2 para determinar los puntos de comienzo y final de segmento visible, se comprueba si el punto es comienzo, final o no de segmento visible. Si es comienzo de segmento la variable booleana *empieza_segmento* pasara a tener valor *True* y si no es comienzo su valor es *False*. Algo similar sucede con la variable booleana *fin_segmento* que presentara un valor *True* cuando el punto sea final del segmento visible. En el caso de que las variables *empieza_segmento* y *fin_segmento* sean *False* el algoritmo no realizara ningún proceso con el punto que se está analizando y se pasara a analizar otro punto del perfil. Sin embargo, si la variable *empieza_segmento* es *True*, que indica que el punto es inicio de segmento visible, entonces se incrementa el contador de anillos sectoriales visibles para el observador que se está analizando y se proceda al almacenamiento de la posición del punto que abre el nuevo segmento visible. Por otro lado, si el punto es final

de segmento de forma que *fin_ segmento* es *True*, las acciones que se realizan principalmente son cerrar el segmento visible y almacenar la posición del punto. Además de estas acciones, se calcula el valor de la superficie correspondiente al anillo-sectorial del segmento visible, y demás parámetros de visibilidad que se desean calcular, ya que al disponer del la dimensión de un segmento visible cerrado se puede calcular todos los parámetros de visibilidad del mismo.

Con el trabajo realizado hasta aquí se finaliza la implementación secuencial del algoritmo desarrollado que calcula los segmentos visibles para todos los puntos de un MDE en todas las direcciones y va a permitir la computación de los principales parámetros de visibilidad.

2.2.3. Algoritmo paralelo para MDE de tamaño medio

Hasta ahora, se ha estudiado el problema de visibilidad para MDEs que la presente tesis ha denominado de tamaño medio (2000 $\times$ 2000puntos o menos) y de forma cuadrada, demostrando que se trata de un problema de elevada complejidad, para el cual existen soluciones. Además se ha mostrado el desarrollado del algoritmo realizado, que proporciona una solución secuencial al problema de la visibilidad. Ahora bien, la computación algoritmo secuencial desarrollado puede llegar a necesitar para un MDE de tamaño medio más de una hora de ejecución en un equipo de sobremesa. Este tiempo de computación hace interesante poder hacer paralelo el algoritmo y de esta forma reducir los tiempos aprovechando la capacidad de cómputo de los sistemas *multi-core* y *multi-socket*.

A la hora de crear, modificar o implementar un algoritmo paralelo aparecen dos elementos esenciales de forma general. El primero es la descomposición del problema en partes o problemas más pequeños de resolver y el segundo es la elección del paradigma de comunicación en programación paralela.

En la computación paralela la descomposición del problema en problemas más pequeños permite resolver cada pequeño problema de forma simultanea y casi por separado, de forma que cada *core* de un sistema *multi-core* resuelve uno de los pequeños problemas. Básicamente existen dos tipos de descomposición, que son:

1. *Descomposición de dominio*, es una de las técnicas más usadas en paralelismo de datos. Los datos de estudio son divididos en partes de aproximadamente el mismo tamaño. Estas partes son computadas de forma paralela en diferentes procesadores. Cada procesador trabaja sólo con la parte de datos que le son asignados, de forma que puede necesitar la comunicaciones con otros procesadores para intercambiar datos. Si el

intercambio de datos es muy alto puede resultar que el algoritmo paralelo tarde más que el secuencial, hecho que puede hacer inadecuado el uso de este tipo de descomposición en el paralelismo si existe una alta dependencia de datos en las operaciones.

Por otro lado, el paralelismo de datos permite mantener un flujo único de control y seguir el modelo SPMD (*single program multiple data*).

La estrategia de descomposición de dominio puede no ser la más eficiente cuando se presente tiempos muy diferentes en el cómputo de las partes asignadas, ya que puede darse el caso de haber hilos parados mientras otros continúan computando. Por tanto, esta descomposición es apropiada cuando la descomposición de los datos permite que el tiempo de cómputo de cada uno de los hilos sea parecido y no exista la necesidad de una excesiva comunicación entre los hilos por la computación de los datos asignados, gracias a la independencia del análisis de los datos de entrada y ejecución.

2. *Descomposición funcional*, también llamada paralelismo de tareas (reparto de tareas). En esta descomposición el problema divide el conjunto total de operaciones o tareas en un gran número de tareas más pequeñas o subtareas (muchas más tareas que procesadores disponibles) y las subtareas son asignadas a los procesadores disponibles. Tan pronto como un procesador termina una subtarea, recibe otra subtarea hasta que queden resueltas todas.

   El paralelismo de tareas se implementa sobre un paradigma de maestro y esclavos. El proceso maestro va asignando las subtareas a los procesos esclavos, recogiendo los resultados producidos y asignando subtareas restantes.

Los paradigmas de comunicación en la programación paralela, buscan que los resultados de hacer paralelos los algoritmos mediante la descomposición sean correctos. Existen dos paradigmas principales a la hora programar de forma paralela:

1. Programación paralela mediante memoria compartida (espacio de direcciones único). En este paradigma los distintos hilos de ejecución pueden acceder a un espacio de direcciones único, con variables globales compartidas, además de haber variables propias no compartidas.

   Este paradigma es apropiado para arquitecturas de memoria compartida, aunque el espacio de direcciones único también puede ser simulado sobre arquitecturas de memoria distribuida.

2. Programación paralela mediante paso de mensajes. En este el programador debe hacer una división del trabajo y los datos a ejecutar entre varios procesos y debe gestionar como tiene que ser la comunicación de los datos entre los procesos, los cuales no comparten ningún espacio de memoria.

   Este paradigma es flexible en cuanto a su implementación, si bien requiere un mayor trabajo del programador.

   Puede ser empleado en arquitecturas de memoria distribuida e incluso en redes de ordenadores, y también en arquitecturas de memoria compartida, aunque en ese caso los diferentes procesos utilizan partes de memoria independientes y separadas.

El algoritmo del presente trabajo no permite una descomposición de los datos de entrada de un MDE, ya que realiza el cómputo total de todos los puntos del MDE y no se puede realizar esta operación por conjuntos de datos. Sin embargo si permite una división de las operaciones, por ser el número de las mismas extremadamente alto, como se puedo ver en el resumen de la complejidad de los algoritmos de visibilidad en el apartado 2.1.1. Todo ello lleva a que la descomposición que se realice sea de tareas para poder computar de forma eficiente un MDE de altura y de tamaño medio.

La primera idea para hacer paralelo el algoritmo es a través de los dos bucles *for*, ya que en programación las partes de un código que mejor se suelen hacer paralelas son los bucles y las operaciones de reducción. Bajo esta estrategia cada hilo de ejecución del programa paralelo podría ejecutar una iteración de algunos de los bucles.

Al analizar los dos bucles se ve que el bucle con mayor número de iteraciones es el interno, con un número de iteraciones igual al número de puntos del MDE a analizar. De manera que si se hace paralelo el bucle interno se podría llegar a poder generar millones de hilos de ejecución. Ahora bien cada iteración del bucle interno guarda una enorme dependencia con la anterior por el uso de la estructura BOS, de forma que una iteración no puede realizarse sin la actualización de la estructura por la iteración anterior. Esta dependencia en la iteraciones del bucle interior hace imposible el paralelismo, ya que seria necesario una modificación total de la estructura de datos o bien la creación de tantas estructuras como hilos en ejecución se pongan a funcionar, lo cual excedería la capacidad de memoria cache. Por tanto, queda descartada la opción de hacer paralelo el bucle interior por lo que le toca estudiar hacer paralelo el bucle exterior que recorre los sectores.

En el caso de hacer paralelo el bucle externo resulta que hay una independencia casi total en el uso de los datos y operaciones a realizar,

simplemente seria necesario la creación de una estructura de datos BOS por cada iteración del bucle exterior que junto con la creación de algunas de las principales variables que intervienen en el bucle interno, permiten la total independencia de ejecución de los hilos de ejecución. Este hecho, permite que sea fácil de hacer paralelo en bucle externo y se pueda llegar a ejecutar de forma paralela hasta un total de $\frac{número\ sectores}{2} = 180$ hilos sin modificar apenas el código secuencial. Simplemente hay que agregar las directivas necesarias para hacer paralelo el bucle *for* que recorre los sectores y re ordenar el código para que las variables y estructuras dependientes de cada hilo se generen interiores al hilo.

Después de ver las partes del código del algoritmo que resultan más fácil de hacer paralelo se puede elegir un paradigma de programación paralela para el algoritmo. La mejor opción resulta ser el paradigma de memoria compartida, ya que al ser totalmente independiente el cálculo de cada pareja de sectores entre si, no es necesario una sincronización entre los hilos de ejecución y la memoria compartida. Algo que no se puede olvidar es la necesidad de poner limites en la escritura de los resultados, para poder mantener la coherencia de los mismos y que sean correctos. Es decir que se ejecute de forma atómica la escritura de resultados sobre las variables generales.

En el pseudocódigo 2.5 presenta un resume de la implementación paralela del algoritmo desarrollado. Donde se puede apreciar el paralelismo hecho en el bucle exterior que recorre los sectores. Las líneas de código en color rojo del pseudocódigo son las modificaciones hechas respecto la versión secuencial presentada en el pseudocódigo 2.3. La primera modificación que se realiza en la versión paralela es pasar la llamada a la rutina *Crear_BOS_va()* al interior del bucle exterior convertido en paralelo. De esta manera, para cada iteración del bucle que recorres sectores se crea una estructura BOS independiente junto las variables auxiliares necesarias para realizar la computación por medio de un hilo de ejecución. Toda la estructura y ejecución del bucle interno se mantiene igual en la versión paralela que en la secuencial. Tras finalizar en bucle interno y haberse calculado los parámetros de visibilidad para un sector sobre todos los puntos, se pasa al almacenamiento de los resultados calculados por el hilo y a la eliminación de todas las variables y estructuras empleadas por el hilo. Y ya por último tras la ejecución completa de los dos bucles se pasa al almacenamiento final de los parámetros de visibilidad y a la liberación del espacio usado.

Pseudocódigo 2.5: Algoritmo Paralelo.

```
LeerMDE(higth);//Lectura del MDE, datos alturas
//Recorrido paralelo de sectores y su opuestos
Paralell for s=0 to s<minsec do

//Generar estructura BOS y variables auxiliares para cada hilo
```

```
  Crear_BOS_va(bw);
  Ordenar_Puntos(s);
  //Barrido de los del MDE cada punto pasa a ser un observador
  for POV=0 to POV<N do
    Actualizar(BOS);
    so=s+180;//sector opuesto al normal
    ComputarVisibilidad(POV,s);// sector normal
    ComputarVisibilidad(POV,so);//sector opuesto
  endfor
  /*Almacenar datos calculados por el hilo*/
  Almacenarsector(parámetros calculados);
  /* Eliminar estructura y variables propias del hilo */
  deleteBOS();

endfor//end Paralell
//Almacenar datos finales
 Almacenar(parámetros calculados);
/* Eliminar todas las variables y datos intermedios*/
 deleteandaux();
```

Bajo esta ejecución paralela el algoritmo puede reducir en más de dos ordenes de magnitud el tiempo cómputo de la visibilidad, siempre que el equipo informático lo soporte. De forma que, si antes se tardaba más de una hora, con la ejecución paralela se pasa a unos pocos minutos.

2.2.4. Algoritmo paralelo para MDE de tamaño grande

Hasta ahora se ha mostrado el desarrollo de un nuevo algoritmo que busca la optimización y la eficiencia en el cálculo de parámetros de visibilidad sobre MDEs de tamaño medio. Ahora bien, si se quiere hacer el estudio de una superficie de grandes dimensiones representada por un MDE de alturas con una alta resolución; cómo puede ser el caso de analizar un territorio tan extenso como el de Andalucía con $10 \times 10m^2$ de resolución. Resulta que el tamaño del MDE a tratar es de $50000 \times 30000 = 1500$ millones de puntos y por tanto, la malla del MDE que la representa es 375 veces más grande que el caso de los MDEs de tamaño medio abordados hasta ahora, por lo que, si se realiza una ejecución de forma similar a todo lo hecho hasta ahora, seria necesario que algunas de las variables empleadas también tuviesen un tamaño 375 veces superior.

El presente trabajo no ha ejecutado el cálculo de la visibilidad para MDE de tamaño $50000 \times 30000 = 1500$ millones de puntos utilizando la misma metodología que la empleada en el caso de MDE de tamaño mediano. Experimentos realizados con MDE de tamaño 16 veces superior al MDE mediano han tardado más de 160 veces el tiempo respecto del mediano, haciendo que el tiempo de cómputo supere las 20 horas frente a los pocos minutos del MDE de tamaño mediano. Si se extrapola esta situación de usar al de 375 veces superior es de esperar que el

tiempo de CPU se incremente en miles de veces el tiempo, ya que los fallos en caché se disparan y por tanto, la comprobación podría tardar uno o varios meses.

Ante la imposibilidad de los algoritmos actuales, e incluso del nuevo algoritmo desarrollado de realizar los cálculos en unos tiempos razonables, se plantea en la presente sección una solución eficiente que, utilizando, parte del algoritmo desarrollado en su forma paralela, reduce los tiempos significativamente y proporciona unos resultados validos para un gran número de aplicaciones, donde aportan soluciones los principales parámetros de visibilidad que son estudiados en la presente tesis.

La clave de toda estrategia que utilice el algoritmo desarrollado reside en que la estructura BOS junto las variables auxiliares en la computación pueda ser almacenadas en los niveles más bajos de la jerarquía de memoria de los procesadores. Es decir, que ocupen el menor espacio de memoria para que en las caché de los procesadores se eviten al máximo número de fallos. Esta situación no es nada fácil de obtener a priori, ya que si se parte de lo expuesto anteriormente en otras secciones (ver sección 2.2.1), sobre un MDE que es 375 veces mayor que el caso de MDE de tamaño mediano resulta necesario que la estructura BOS presente un tamaño de puntos de al menos $\sqrt{375} \approx 19{,}4$ veces mayor que en el caso de MDE de tamaño mediano. De forma que el tamaño recomendable de la estructura BOS para un MDE con 1500 millones de puntos deben, en principio, ser unos 38000 puntos, y por tanto, requeriría de espacio aproximado de $640KBytes$ en la L1. Por otro lado, las limitaciones vienen también por el tamaño de las demás variables que intervienen, ya que, por ejemplo, el tamaño del fichero de entrada del MDE pasa a ser unos 6 GBytes, cuando para MDE de tamaño medio era de unos 16 MBytes. Al igual sucede con los resultados de visibilidad de la computación del algoritmo que pasan a ocupar del orden de GBytes.

La problemática de la necesidad o aumento de datos por la estructura BOS en el tratamiento de MDE de tamaño grande se puede abordar si se limita el radio de estudio al considerar que puntos lejanos no aportan información de interés en un gran Número de aplicaciones de la visibilidad, como telefonía, torres de vigilancia, etc. Por ejemplo, siguiendo con el caso de un MDE de tamaño 1500 millones de puntos y con resolución de $10 \times 10m^2$, resulta que la longitud que llega a cubrir la ventana BOS para una pareja de direcciones opuestas puede alcanzar los 500 kilómetros de longitud. Este valor excede las distancias de alcance máximo de la mayoría de las aplicaciones de visibilidad, como es el caso de las torres de telefonía móvil donde las distancias suelen ser del orden de las decenas de kilómetros e incluso menos. Es más, la propia curvatura de la tierra hace innecesario el cálculo de la visibilidad más allá de los 200 Kilómetros, ya que la superficie de la tierra actuá como obstáculo.

La duda ahora es seleccionar un tamaño de ventana BOS, y en consecuencia el de estructura que permita tener un alcance acorde con las necesidades de las aplicaciones que se plantean resolver. La solución no es sencilla, ya que cualquier limite puede representar una perdida de información en mayor o menor medida para ciertas aplicaciones. Ante este marco de trabajo la presente tesis ha buscado que el limite del radio de visibilidad permita el cálculo de los parámetros de visibilidad atendiendo a criterios de la obtención de soluciones de forma más rápida, precisa y abarcando el mayor número de aplicaciones disponibles sobre un MDE de gran tamaño.

Si se regresa a la problemática sobre los MDEs de tamaño mediano, de la cuál se sabe el buen funcionamiento, resulta que para MDEs de tamaño 2000×2000 puntos y resolución $10 \times 10m^2$ la longitud máxima de radio de un observador localizado en el centro del MDE es de unos 10 kilómetros. Estos 10 Kilómetros resulta un valor más que suficiente para muchas de las aplicaciones de visibilidad. Por lo que una idea para poder computar de forma rápida un MDE de tamaño grande es la limitación del radio máximo a 10 kilómetros, de forma que la estructura BOS puede tner el mismo tamaño que la usada en los MDE de tamaño mediano. Además, poner este radio máximo permite la descomposición del MDE de gran tamaño en MDEs de tamaño mediano que se pueden calcular de forma independiente y después ser combinados los resultados para obtener el resultado del MDE de tamaño grande con limitación de radio.

Las ventajas de la descomposición del MDE de tamaño grande en MDEs de tamaño medio no queda solo en lo expuesto en el apartado anterior sino que, si se logra una computación independiente de cada uno de los MDEs de tamaño medio entonces se puede agregar otro nivel de paralelismo al problema, ya que cada nuevo MDE de tamaño medio se puede ejecutar dentro de un sistema *multi-core* diferente por la descomposición de los datos de entrada haciendo que se reduzca los tiempo de ejecución. Por tanto, si se analizara un MDE como el de Andalucía con $50000x30000$ puntos bajo la partición en MDEs de tamaño mediano, se obtendría un total de 375 MDEs de $2000x2000$ puntos, los cuales cada uno se puede ejecutar en de forma paralela en un sistema *multi-core*. Si además se tiene en cuenta que la forma paralela del cómputo del MDE de tamaño mediano permitía la paralización en 180 hilos del bucle de recorrido de los sectores visto en la sección2.2.3, entonces se puede llegar a generar 67500 hilos de computación independientes. Esto permite que si se dispone de un sistema informático con 67500 cores CPU y suficientes recursos de memoria se puede tardar menos de diez minutos en computar la visibilidad de todo el MDE de 1500 millones de puntos, con limitación de radio.

En la figura 2.21 se muestra una partición de Andalucía en cuadrículas o MDE

de tamaño 2000×2000.

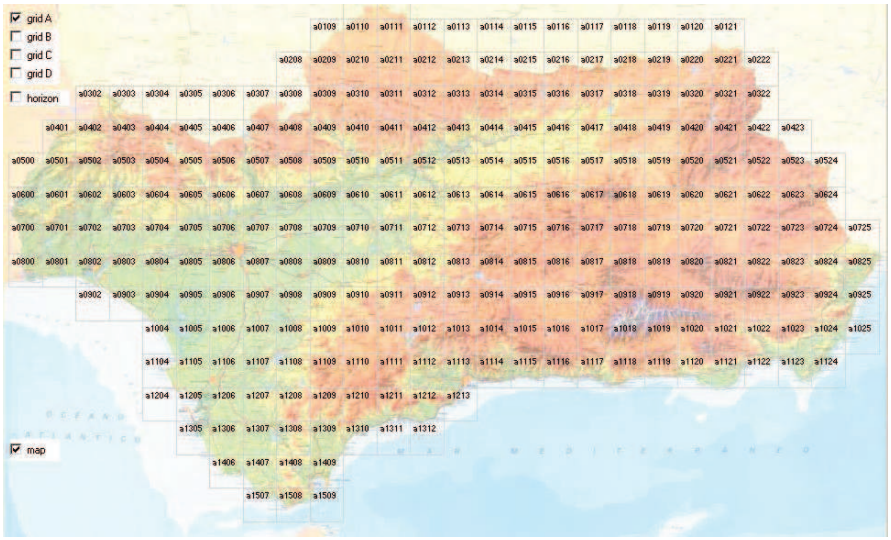


Figura 2.21: Partición de Andalucía en MDEs de tamaño 2000×2000 puntos.

Al aplicar la división del MDE de tamaño grande en MDEs de tamaño medio y computarlos cada uno de ellos de forma independiente surge un problema que afecta enormemente a la validez de los resultados, y hace que estos sean prácticamente inválidos para muchos de los puntos de los MDEs de tamaño medio. Este nuevo problema es lo que el presente trabajo a denominado *efecto borde de la combinación de los MDEs medianos*. El efecto borde consiste en que puntos cercanos a los bordes o laterales de los MDEs de tamaño mediano no pueden ser analizados correctamente por no disponer de la información de otros MDEs vecinos, ya que puntos localizados físicamente dentro del radio máximo de estudio asignado no pueden ser analizados por localizarse en otros MDEs distintos. Bajo esta situación los puntos de los laterales del los MDEs de tamaño medio solo podrán presentar valores reales para el análisis de los sectores cuyas direcciones sean hacia el interior del MDE y no se produce el efecto borde. Mientras que para los sectores con direcciones hacia el exterior del MDE presenta soluciones truncadas por no poder utilizar la información de otros MDEs de tamaño mediano como se ha indicado.

Para reducir el problema del efecto borde y que las soluciones sean validas para todos los sectores se ha empleado una solución heurística desarrollada en el trabajo [36]. La solución de este trabajo divide el MDE de tamaño grande en

MDEs de tamaño medio que se solapan parcialmente. A cada uno de estos nuevos MDE de tamaño medio se le denomina malla de puntos. A la hora de hacer el estudio de visibilidad sobre los puntos de las mallas, no se analizan todos los sectores para los puntos sino un conjunto de sectores que depende de la posición del punto sobre la malla. En concreto lo que se hace es una nueva división de la malla en cuatro sub-mallas o cuadrículas independientes de tamaño un cuarto del original y a los puntos pertenecientes a cada sub-malla se le calcula la visibilidad para el mismo conjunto de sectores. Los conjunto de sectores que utiliza [36] son cuatro y cada uno tiene un cuarto del número de sectores totales. Es decir que si se usan un total de 360 sectores para cada cuadricula se analizan 90 sectores cuyas direcciones van siempre hacia el interior de la malla del MDE. Es decir se analizan los sectores cuya dirección no necesitan de la información de los puntos de otros MDEs de tamaño mediano para los puntos del borde de la cuadricula que a su vez pertenecen a los bordes de las mallas o MDE de tamaño medio en los que se ha dividido el MDE de tamaño grande.

En realidad el cómputo descrito en el párrafo anterior no evita el efecto borde, ya que si el radio máximo de visibilidad hace que se supere por los puntos de la cuadricula las dimensiones del MDE, entonces se está produciendo otra vez el efecto borde, aunque esta vez afecta a distancias mayores. Parece que el efecto borde no se puede eliminar, pero la realidad es que si se aplica la técnica multigrid descrita en [36], donde el radio máximo alcanza los 200 Kilómetros, que es la máxima distancia permitida por la curvatura de la tierra. Ahora bien si se limita el radio máximo a la distancia entre los dos puntos más alejados de una cuadricula, y se aplica el cómputo del párrafo anterior entonces el efecto borde es casi despreciable, ya que en los sectores que se analiza nunca necesitaran información de puntos de otros MDE de tamaño medio y se evita el tener que implementar la técnica multigrid.

En la figura 2.22 se puede ver las cuadrículas, su numeración y cuales sería las direcciones que tienen los sectores de estudio en dichas cuadrículas; donde se ha realizado la siguiente partición de cuadrícula y asignación de conjunto de direcciones:

- Cuadrícula 1, el conjunto de direcciones es $\vec{S_i}$ con $i = 1, 2...., 90$, en color rojo.
- Cuadrícula 2, el conjunto de direcciones es $\vec{S_i}$ con $i = 91, 92...., 180$, en color verde.
- Cuadrícula 3, el conjunto es $\vec{S_i}$ con $i = 271, 272...., 360$, en color naranja.
- Cuadrícula 4, el conjunto es $\vec{S_i}$ con $i = 181, 182...., 270$, en color azul.

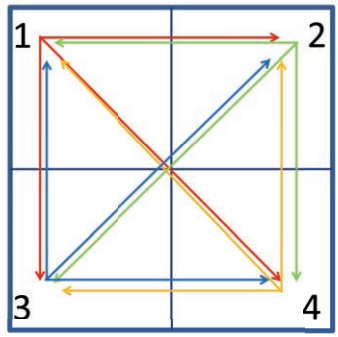


Figura 2.22: Partición de una malla en cuatro cuadrículas.

Nunca hay que olvidar que en cada cuadrícula se calcula la visibilidad en un conjunto limitado de sectores, pero se considera los datos de toda la malla.

Tras computar todas las cuadriculas con el limite del radio máximo, el efecto borde se ha eliminado, pero ahora los puntos de cada una de las cuadriculas solo tienen la información de visibilidad de un conjunto de 90 sectores de un grado y no los 360 sectores necesarios para tener la visibilidad en todas las direcciones. La mejor solución para tener la información de todos los sectores alrededor de un observador es la generación de tres nuevas particiones del MDE de tamaño grande en MDEs de tamaño medio de igual tamaño que la primera partición. Estas nuevas particiones se solaparan en el espacio con la primera partición del MDE de tamaño grande pero estarán desplazadas una cuadricula respecto la primera. De esta forma las mismas cuadriculas de puntos en cada una de las particiones estarán en diferente posición sobre los distintos MDEs de tamaño medio generados. Así podremos calcular a cada cuadricula un conjunto de sectores sin que aparezca el efecto borde, porque dependiendo del MDE de tamaño medio al que pertenezca, se calculara un conjunto de sectores o otro a la cuadricula hasta obtener todos los sectores necesarios.

Al final de la computación de todos los MDEs de tamaño medio desplazados una cuadricula se obtiene, para los puntos de cada cuadrícula, la información en todas las direcciones, que se sumara y se obtendrá el valor final del MDE de tamaño grande con limitación de radio.

En la figura 2.23 muestra como se obtiene el resultado completo de los 360 sectores para una de las cuadriculas (cuadricula 1) de un MDE de tamaño mediano, representado por el nombre $MDEi, j$ *sumado*. Se ve que son necesarios

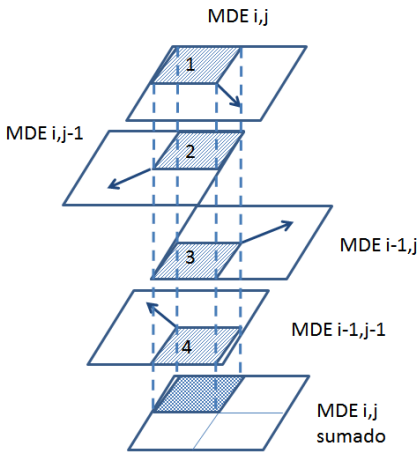


Figura 2.23: Suma cuatro cuadriculas para la generación de una con todos los sectores.

el cómputo de 4 MDEs medianos desplazados, el primero es el $MDEi, j$ que corresponde con el MDE de estudio del cual se obtienen los datos de los sectores de 1 a 90, el segundo es el $MDEi, j-1$ que corresponde al MDE desplazado su estudio una cuadrícula a la izquierda, de forma que aporta los sectores de 91 a 180, el tercero es $MDEi-1, j$, desplazado una cuadrícula hacia arriba para sumar la cuadricula 3 con los sectores de 181 a 270. Por último el $MDEi-1, j-1$ que es el anterior desplazado una cuadrícula a la izquierda para sumar los sectores que faltan hasta los 360 para cada uno de los puntos de la cuadrícula.

En el caso de dividir el MDE de tamaño grande en MDEs de tamaño medio de 2000×2000 puntos resulta que cada una de las cuadriculas que componen el MDE es de tamaño 1000×1000 puntos. Y una vez realizada la partición de las cuadrículas queda también delimitado el radio máximo de estudio para evitar el efecto borde. Este radio en el caso de tener cuadrículas de tamaño 1000×1000, es de 1000 puntos; ya que, si por ejemplo, se está realizando el análisis en la cuadrícula 1 donde se estudian los sectores con direcciones de $\vec{S_i} = 1 \;\; a \;\; 90$, la distancia mínima de los puntos de dicha cuadrícula al borde del MDE mediano es de 1000 puntos y se presenta en los puntos de los bordes de la cuadrícula que no comparten localización con los de los bordes del MDE mediano, sino que se encuentran a una distancia de no menos de 1000 puntos de los bordes del MDE mediano en las direcciones de estudio. La longitud real del radio limite para evitar el efecto borde con esta metodología es el producto de la resolución del MDE por

el número de puntos que evita el efecto borde. De forma que si se está trabajando con MDEs de 2000×2000 puntos de tamaño y una resolución de $10 \times 10m^2$ el radio límite con el que se puede trabajar es de $10m \times 1000 = 10Km$ para evitar el efecto borde. Todos estos tamaños y limites de radio por el tamaño de las cuadriculas son totalmente compatibles con el tamaño de ventana BOS por lo que bajo el algoritmo desarrollado en la presente tesis no seria necesaria ninguna limitación en la estructura que contiene la ventana BOS.

La distancia de 10km es suficiente en muchas aplicaciones como telecomunicaciones de corta distancia, paisajismo, vigilancia de incendios...

La estrategia descrita permite que no aparezcan problemas con el efecto borde de los MDE, pero se presenta la necesidad de trabajar con cuatro veces el número original de mallas, ya que para generar el valor de visibilidad en todas las direcciones de un observador perteneciente a una de las cuadrículas hay que estudiar un total de cuatro MDE medianos, desplazados una cuadrícula cada uno y a continuación sumarlos, como se puede observar en en ejemplo de la figura 2.23.

La situación de necesitar cuatro MDEs desplazados una cuadrícula para generar una cuadrícula completa, hace que el número total de MDEs de tamaño mediano a analizar se multiplique por cuatro, de forma que si, por ejemplo, antes se tenia que analizar 375 MDEs que no se solapaban ahora se pasa a analizar 1500 MDEs que sí lo hacen. Ahora bien cada uno de los nuevos MDEs que se solapan parcialmente se computa en un tiempo inferior por no realizarse el cálculo de todos los sectores en todas las cuadriculas. Por otro lado, como se mantiene la independencia de la computación entre los distintos MDEs de tamaño mediano parcialmente solapados se puede computar cada MDE parcialmente solapado en un *core* distinto, lo que permite aumentar el nivel de ejecución paralela si hay suficientes recursos en el sistema informático que lo computa. Por tanto, si el equipo pudiese ejecutar de forma simultanea unos $4 \times 67500 = 27000$ hilos de cómputo entonces el algoritmo daría solución en menos de la mitad de tiempo y sin efecto borde, ya que el tiempo de cómputo de cada uno de los MDE de tamaño medio parcialmente solapados se reduce en más de la mitad.

Una vez presentada como es la estrategia que emplea la presente tesis para la computación de un MDE de tamaño grande exponemos cómo es el proceso básico para su implementación. Ésta consta de 4 pasos esenciales que en la implementación real pueden ser más o los mismos pero más complejos:

1. División del MDE de tamaño grande en MDEs medianos parcialmente solapados.

2. Barrido de todos los MDE de tamaño medio y computación de cada uno

de ellos.

3. Combinación de resultados de las cuadriculas en MDE de tamaño medio que no se solapan con todos los sectores de análisis.

4. Combinación de los resultados de cada uno de los MDEs de tamaño medio en un MDE de tamaño grande.

La división del MDE de tamaño grande se puede hacer previamente a la ejecución del algoritmo y no en la propio algoritmo de forma que cada uno de los MDEs de tamaño mediano parcialmente solapados se guardan en un fichero para su posterior estudio. Esta metodología ahorra tiempo de cómputo si es necesario hacer varias ejecuciones del algoritmo y si se quiere ser más selectivo en el estudio de visibilidad de zonas de interés sobre el MDE de tamaño grande y no sobre todo el MDE, ya que simplemente es seleccionar los ficheros con los datos de entrada deseados para su computación. Además, si se hace la partición previa y no se dispone de un equipo con un número de cores tan alto para ejecutar todos los hilos, pero si de varios equipos independientes que no se comunican, entonces bajo la presente estrategia se puede fraccionar el estudio del MDE grande entre los distintos equipos. Esto se hace gracias a que el estudio de cada MDE de tamaño medio es independiente y al final del proceso basta con recoger los resultados de cada equipo y combinarlos. Por esta razón, ha sido la estrategia empleada en la presente tesis, ya que en el desarrollo de la presente tesis no hemos podido acceder a un sistema informático con 27000 cores CPU y que además, lo tuviésemos disponible durante un periodo de tiempo de al menos un día. Lo que si se ha llegado a realizar son computación sobre sistemas informáticos con cientos de cores-CPU compartido por varios usuarios que se encolan en el lanzamiento de sus implementaciones.

Volviendo al algoritmo, resulta que la parte principal a modificar en el algoritmo de MDE de tamaño medio para la computación del MDE de tamaño grande es la correspondiente a la función *ComputarVisibilidad()*. En esta se limita el estudio o análisis de los sectores para cada punto, dependiendo de a que cuadricula del MDE de tamaño medio pertenece y la demás parte del algoritmo apenas se modifica, ya que simplemente se va realizando por parte del del algoritmo la lectura de los MDEs de tamaño medio desplazados que han sido previamente guardados en ficheros.

En pseudocódigo 2.6 se muestra el resumen de como quedaría la implementación del algoritmo para analizar los sectores dependiendo de a que cuadricula pertenezca el punto. La parte de código en color rojo son las modificaciones echas para la computación por cuadriculas. Donde lo primero es la

creación de un tercer bucle tipo *while* para recorrer los MDEs de tamaño medio que se quieran computar, de forma mientras no se haya terminado de computar los MDE se realizara la elección de un nuevo MDE por la función *elegMDE()* que pasara el nombre del fichero a la función *leerMDES()* para su lectura y paso al núcleo del algoritmo. Este núcleo de algoritmo es prácticamente el mismo que el del caso de computar un solo MDE de tamaño medio, ya que el barrido de sectores y puntos. La diferencia estriba, en que una vez seleccionado el punto en el bucle interior se comprueba a que cuadricula pertenece mediante la función *cuadricula()* y el valor de obtenido se guarda en la variable *cu*, que sera pasada a la función *ComputarVisibilidad()*, para que esta la tenga en cuenta.

La función *ComputarVisibilidad()* con la información de a que cuadricula pertenece el observador POV y de que sector se esta analizando permite o no que se realicen el barrido de la ventana de visión para el observador. La elección de si se estudia un sector o no se determina en el pseudocódigo por las nuevas lineas en color rojo dentro de la función función *ComputarVisibilidad()*, que son dos textitif, los cuales permiten realizar el barrido de la estructura si se cumple las condiciones de las cuadriculas y los sectores.

Pseudocódigo 2.6: Algoritmo Paralelo para cálculo de cuadriculas.

```
elegMDE(MDE) //Seleccionar primer MDE de tamaño medio desplazado a computar
compute=(MDE!=ultimoMDE)
while (compute) do //Barrido de MDEs de tamaño medio

  leerMDEs(MDE); //Lectura de MDEs de tamaño medio solapados parcialmente
  Paralell for s=0 to s<minsec do //Barrido de sectores de forma paralela
   Crear_BOS_va(bw);
   Ordenar_Puntos(s);

   for POV=0 to POV<N do //Barrido de los N puntos del MDE
     /*Comprobar cuadricula a la que pertenece POV*/
     cu=Cuadricula(POV)
     Actualizar(BOS);
     so=s+180;//sector opuesto al normal
     ComputarVisibilidad(POV,s,cu);// sector normal
     ComputarVisibilidad(POV,so,cu);//sector opuesto
   endfor

   Almacenarsector(parámetros calculados);//Almacenar datos calculados
   deleteBOS();//Eliminar estructura y variables propias del hilo
  endfor //end Paralell
 //Almacenar datos finales
  Almacenar(parámetros calculados);
  //Eliminar todas las variables y datos
  deleteandaux();
  elegMDE(MDE) //Seleccionar siguiente MDE a computar
endwhile


ComputarVisibilidad(POV,s,cu)
  float d_POV=POV.d;//Posición del observador
```

```
  float h_POV=POV.h;// Altura del observador
  bool visible_segmento=false;
  float maxanguloanterior= -∞ //Máximo ángulo anterior

  currP=POV;
  if s<180 then
    if (cu==1 && s<90 )or (cu==2 && s>=90) then
    while currP != PL do
        currP=currP.sig;
        Kernel(d_POV,h_POV,currP.d,currP.h,&max,&visible_segmento);
      endwhile
    endif
  else
    if (cu==3 && s>=90) or (cu==4 && s<90) then
    while currP != PF do{
        currP=currP.prev;
    Kernel(d_POV,h_POV,currP.d,currP.h,&max,&visible_segmento);
      endwhile
    endif
  endif
end ComputarVisibilidad
```

Para obtener el resultado final del MDE de tamaño grande hay que combinar los resultados del cálculo de los parámetros de visibilidad sobre todos los MDEs medianos cuadriculados o parcialmente solapados en los que se basa nuestra metodología. Esta operación es bastante sencilla de implementar como se ha mostrado en la generación una cuadricula que contenga la información del parámetro de visibilidad en todos los sectores, ya que basta con ir sumando la información de los MDE de tamaño medio parcialmente solapados. Pero la realización de esta sencilla tarea es difícil de llevar a cabo por el volumen de datos con el que se puede estar tratando. Si por ejemplo, se parte del tan mencionado MDE de Andalucía con 1500 millones de puntos hay que generar 1500 MDEs de tamaño mediano que después hay que combinar. De forma que por un lado hay que disponer de variables con capacidad de almacenar el MDE de tamaño grande y por otro todos los MDEs de tamaño medio. Una variable que pueda almacenar solo los valores finales de los parámetros de visibilidad de todos los puntos de un MDE de tamaño 1500 millones necesita un espacio de unos 6Gbytes. Por otro lado, tener abierto los 1500 MDEs de tamaño medio de unos 2000×2000 necesita tener 1500 variables con 16 Mbytes de tamaño cada una para después ir realizando las sumas, es decir hace falta al menos 24 Gbytes si se quieren sumar a la vez la información de todos los puntos. Hay que recordar siempre que estos tamaños de memoria son solo para la computación del valor final de los parámetros de cada punto, ya que, si por ejemplo, se deseara almacenar de forma conjunta todos los segmentos visibles para su posterior representación gráfica se necesitaría de variables con un tamaño superior al centenar de **Terabyte**. El gestionar directamente variables con estos tamaños es enormemente complicado por la incapacidad de que quepan en las cachés de los procesadores. Por tanto,

surge la necesidad de hacer por fases la combinación de los MDEs de tamaño medio cuadriculados. En concreto se ha hecho en dos fases:

- Primera fase: la generación de los MDEs de tamaño mediano completos con los 360 grados a partir de los MDEs basados en cuadriculas. Para obtener un MDE de tamaño mediano hay que operar con en resultado de visibilidad en 9 MDEs medianos cuadriculados que son cada uno de los MDEs que comparten alguna cuadrícula con el original. Esto se hace por medio de cuatro operaciones:

  1. Apertura de cada fichero de visibilidad calculado de cada MDE mediano cuadriculado.
  2. Identificación de las cuadriculas.
  3. Combinación o suma de las cuadriculas con los mismo puntos de estudio pero en distintas direcciones.
  4. Generación del nuevo MDE de tamaño medio en un fichero con los resultados.

- Segunda fase: la unión de todos los resultados de MDEs de tamaño medio, para generar la solución del MDE de tamaño grande o mapa completo de visibilidad a partir de los MDEs de tamaño medio anteriores. Un primer enfoque sería generar directamente el fichero de visibilidad completo mediante la creación de un array que contenga todos los puntos y a continuación guardarlo en un fichero. Esta metodología resulta un poco costosa en tiempo, debido a que si se cogen los resultados de los 375 mapas de visibilidad para generar un único mapa de visibilidad es necesario que el array final tenga un tamaño de $375 \times 2000 \times 2000 \times 4 = 6 GBytes$ en memoria por un lado y por otro lado, la memoria necesaria en la apertura de cada uno de los ficheros a combinar es de $16 Mbytes$. Ahora bien si el objetivo final es la generación de una imagen del mapa de visibilidad para poder estudiar de forma visual y rápida los resultados, entonces el problema se ve de nuevo incrementado por necesitar ahora dos estructuras en memoria de tamaño una $6 Gbytes$ para el fichero de resultados y otra estructura de un tamaño igual o superior para fichero de imagen de salida aunque después una vez generada la imagen se comprima el fichero en algún formato de imagen más compacto.

Con todo el trabajo presentado hasta este punto se ha mostrado como el algoritmo desarrollado puede obtener un parámetro de visibilidad deseado de cualquier superficie representada por un MDE de cualquier tamaño, en unos

tiempos razonables y con una alta precisión gracias a la capacidad del algoritmo para poder ser implementado en los sistemas *multi-socket* y/o *multi-core*.

2.3. Marco Paralelo para Algoritmos de Visibilidad

De una forma menos visible, el trabajo realizado no solo ha aportado una solución eficiente para dar resultados de visibilidad, sino que también ha desarrollado un marco de paralelismo valido para la computación de la mayoría de los algoritmos de visibilidad que traten de analizar MDE de tamaño grande. Este marco paralelo se encuentra en su mayoría inmerso entre las líneas de esta memoria y puede resultar difícil de verlo claramente, por lo que se considera imprescindible resumirlo en esta sección. De esta forma, se tiene una mejor compresión del mismo y lugar en la memoria donde poder consultarlo rápidamente.

El marco paralelo que mostramos a continuación es valido para cualquier algoritmo de cálculo de visibilidad en SIG que use un MDE de alturas y esté basado en el estudio de los parámetros en análisis sectorial de los datos con limitación del radio máximo de cobertura. El radio máximo va a estar siempre limitado por dos factores: uno, las necesidad de la aplicación para que se realice el estudio del parámetro que se computa; y dos, el tamaño del MDE que se estudia, ya que puede que este sea muy pequeño con lo cual no habría problema o que sea muy grande lo que hace necesario la división del MDE grande en MDE de tamaño mediano.

El marco paralelo se ha dividido en niveles, los cuales pueden ser o no empleados sobre el algoritmo de cálculo, siendo cada nivel independiente de aplicación aunque dependiente en el proceso. Es decir si un nivel se aplica no tiene porque aplicarse otro, pero la aplicación de uno de ellos hace que los otros se puedan aplicar de forma más o menos optima. Una representación gráfica del marco paralelo se muestra en la figura 2.24. Dicha figura resume el marco paralelo desarrollado para cualquier algoritmo SIG basado en sectores y que es dependiente del número de puntos del MDE a tratar.

Como se muestra en la figura 2.24 el marco paralelo se ha dividido en tres niveles, donde el primer nivel o nivel más alto es la aplicación de un método de resolución *multigrid*, que deja paso a una partición de datos por bloques que comprende el segundo nivel o nivel medio y un paralelismo de grano fino a través de la ejecución paralela de la computación sectorial en el nivel más bajo o tercer

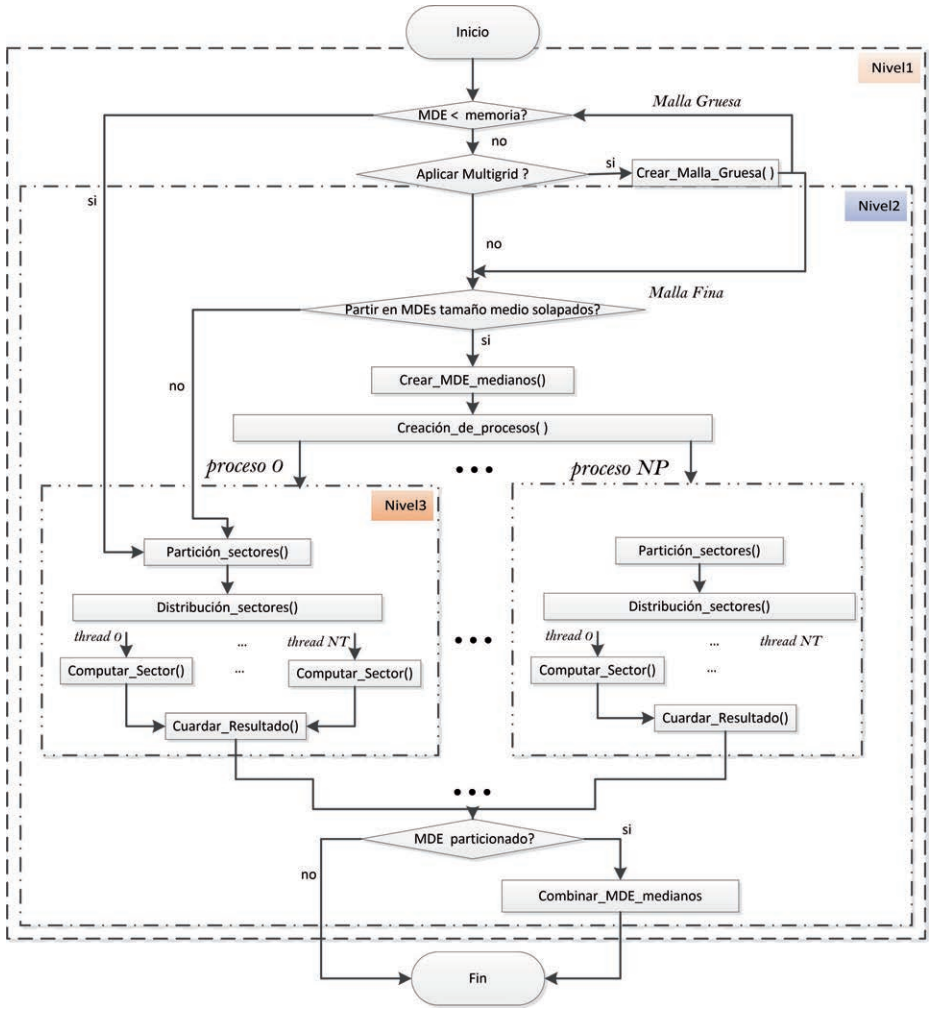


Figura 2.24: Diagrama de computación paralela de MDEs de cualquier dimensión.

nivel. A continuación se describe cada uno de los niveles con más detalle.

2.3.1. Nivel 1: Aproximación multi-resolución

El nivel uno no aporta paralelismo, pero es imprescindible tenerlo en cuenta, ya que una variación en la resolución del MDE puede hacer que los datos quepan

en los niveles inferiores de la jerarquía de memoria de los procesadores utilizados. Al aplicar el cambio de resolución, por un lado se aporta una mejora considerable en los tiempos de cómputo y por otro lado, permite aumentar el radio máximo de análisis en los casos que no importe una perdida de precisión de los resultados.

Este nivel tiene en cuenta que en la mayoría de las aplicaciones SIG, el análisis de la visibilidad es determinar si un punto A es visible respecto otro punto B mediante la utilización de alguna función que lo confirme. A estas funciones se les denominan funciones punto a punto (en ingles point-to-point function) por realizar principalmente comparaciones punto punto. Es interesante ver en la resolución de este tipo de problemas cómo no siempre se requiere la misma precisión en las soluciones, hecho que permite modificar la resolución de los datos de entrada de forma que se analicen menos puntos y que estos quepan mejor en memoria y permitan el ahorrando tiempo de cómputo. Por ejemplo, si se estudia puntos lejanos sobre un observador humano, resulta que la fidelidad del ojo humano es una función que se atenúan o reducen de forma inversa al cuadrado de la distancia en la mayoría de los problemas de visibilidad, de igual forma sucede con las ondas de radio. Basado en el hecho de que estos algoritmos necesitan menos precisión en las zonas más alejadas, la mayoría de los autores que resuelven este problema proponen utilizar una rejilla de resolución más baja para los puntos más lejanos en el caso que se desee tener un mayor alcance. Esta idea algunos autores la denominan generación de una malla de resolución baja, también llamada malla gruesa.

Otra de las razones de la generación de una malla gruesa es debido a la naturaleza o tipo de datos de entrada, ya que puede resultar más o menos relevante la precisión del cálculo. Por ejemplo, si deseamos calcular la cuenca visual en puntos que se encuentran sobre el mar, no es necesaria mucha precisión, ya que toda la superficie se encuentra a la misma altura. Sin embargo, si es el cálculo de Cuencas Visuales sobre áreas urbanas, nos interesara una mayor resolución de los MDE.

La actuación de este nivel se puede resumir, en la realización de un análisis de las necesidades de la aplicación que se quiere resolver junto con las características de los MDE de entrada y las características del sistema informático que realizara la computación. Si el sistema cuenta con suficientes recurso para dar una solución con las características deseados no se aplicara este nivel de cambiar resolución y se pasa directamente al último nivel o de cálculo. Si no es el caso se comprueba si una solución con menos resolución es valida para la aplicación que se está tratando. Si es valida la solución con menos resolución se aplica la creación de un nuevo MDE de menor resolución y si no es valida la solución con menor resolución se pasa al siguiente nivel.

2.3.2. Nivel 2: Partición de datos

Se ha mostrado en la sección 2.2.4 un ejemplo de aplicación de este nivel del marco-paralelo, donde interesa dividir un MDE de tamaño grande en varios de MDE de tamaño mediano, para poder calcular de forma eficiente los parámetros de visibilidad deseados. El único inconveniente de la aplicación de este nivel es la reducción en la máxima distancia visible calculable, para no cometer errores de efecto borde, pero, sin embargo, permite realizar el cálculo de cada uno de los bloques o MDE resultantes de la partición en distintos cores, de forma que se ha dividido el cálculo del MDE principal en procesos más elementales y eficientes de computar, aplicando la descomposición de los datos en MDE más pequeños.

Este nivel es muy recomendable cuando la estructura de datos requerida por el algoritmo (como la implementación basada en el hull-tree [34, 36]) no encajan en la jerarquía de memoria penalizando enormemente el tiempo de cómputo. En este punto, es tarea del programador analizar y comprobar cual es la la mejor división acorde con los recursos del equipo con el que se realizan los cálculos.

Un punto muy importante es que en el caso de trabajar con un sistema informático compuesto por diferentes unidades *multi-core*, *multi-socket* con distintos recursos, la partición del MDE no tiene por que ser uniforme sino que puede presentar distinto tamaño de MDE particionados acorde con los recursos de cada sistema informático que lo compute. Cuando se realiza particiones no iguales deben ser estas lo más regulares posibles y obedecer a criterios geográficos con el fin de minimizar los requerimientos de comunicación entre los procesos que computan cada una de las particiones. De esta forma, a la hora de unir los resultados de las particiones o subbloques la tarea sera más sencilla y la espera de unos subbloques a otros menor. Es decir que si por ejemplo, el tamaño de cuadricula de los MDE de menor tamaño de la partición es de 500×500 puntos, entonces un buen tamaño para las cuadriculas de los MDE más grandes de la partición podría ser de 1000×1000 puntos, y de esta forma sería más sencillo la combinación de resultados si además las particiones han sido realizadas acorde a criterios geográficos donde si es necesario la superposición, ésta es inmediata. La implementación de este nivel de partición se ha realizado con un modelo de paso de mensajes (MPI);

Una vez asignado a cada proceso cual va a ser el MDE particionado que va a analizar, queda simplemente computarlo en el nivel tres.

2.3.3. Nivel 3: Paralelización por sectores

Este nivel es exclusivo para algoritmos SIG basados en el análisis de cálculo sectorial. Este nivel presenta un alto nivel de paralelismo, ya que hace paralelo el cálculo del parámetro de estudio gracias a la computación de los sectores en diferentes cores.

Este nivel del marco de paralelismo puede ser fácilmente implementado en sistemas de alto rendimiento basados tanto en memoria compartida como distribuida gracias a que el estudio de cada sector es independiente y solo hay que compartir la información una vez finalizados los cálculos que obtienen la información resultante de cada sector.

Capítulo 3

Aplicaciones

Los parámetros de visibilidad y sus aplicaciones en SIG han sido ampliamente estudiados, como demuestra la bibliografía, y a fecha de hoy, el número de aplicaciones es muy alto y no deja de crecer. El desarrollo de nuevas aplicaciones SIG hace necesaria la mejora de equipos electrónicos tanto estáticos (computadores), como móviles(satélites, drones, teléfonos móviles....), para poder ser capaces de proporcionan soluciones a una emergente infinidad de problemas reales, que hace algunas décadas eran impensables, y que a fecha de hoy facilitan la vida de millones de personas. Por ejemplo, en el caso de equipos móviles, se pueden destacar los drones, tan de moda en los últimos años y usados en un amplio número de aplicaciones como son: vigilancia, mensajería, grabación de eventos... En el caso de equipos estáticos, está la mejora de los sistemas informáticos, que cada vez presentan más potencia de cálculo y permiten la resolución de problemas con un coste mayor computacional, permitiendo la resolución de problemas en menor tiempo.

El presente capítulo muestra el potencial del algoritmo descrito en el capítulo 2 para desarrollar cuatro mapas novedosos útiles para varias aplicaciones reales. En concreto, el algoritmo desarrollado permite calcular los siguientes parámetros:

1. Cuencas Visuales para todos los puntos de un MDE de tamaño mediano y grande, el llamado Total-Viewshed.
2. Visibilidad Volumétrica para todos los puntos de un MDE de tamaño mediano y grande, llamado Total-Volumetric-Visibility.

Estos parámetros pueden ser usados para múltiples aplicaciones:

1. Cálculo de múltiples observadores para Máxima cobertura de un MDE de tamaño mediano.

2. Otras aplicaciones: El cálculo de los horizontes de todos los puntos de un DEM, el cálculo de las zonas aisladas de un terreno, el cálculo de la trayectoria óptima de un drone.

La implementación del algoritmo desarrollado, así como la optimización del cálculo de los distintos parámetros y aplicaciones, se ha realizado utilizando el lenguaje de programación *C++* y se ha ejecutado en diferentes sistemas multi-core, para comprobar la coherencia de los resultados y los tiempos de computo. Se ha utilizado el lenguaje *C++* por el rendimiento del código nativo generado, frente a otras soluciones(*Java*, *C#*), y por la existencia de librerías que permiten la implementación eficiente del algoritmo tanto de forma secuencial, como la implementación paralela. En este último caso son ampliamente utilizadas en la bibliografía librerías como openMP y MPI que son altamente compatibles en la implementación de programas que se deseen ejecutar sobre diferentes arquitecturas multi-core y multi-socket.

3.1. Cálculo de las Cuencas Visuales

La Cuenca Visual, también llamada Viewshed, Cuencas Visuales 2D o Viewshed 2D, es el parámetro más estudiado en los SIG para la resolución de múltiples problemas de visibilidad en grandes territorios.

La definición general de las Cuencas Visuales se mostró en el apartado 1.5 en la introducción de la presente tesis. Para poder calcular las Cuencas Visuales es necesario conocer primero el parámetro de las Cuencas Visuales Lineales.

3.1.1. Cuenca Visual Lineal

La Cuenca Visual Lineal tiene carácter unidimensional como se puede extraer de su propio nombre, y afecta exclusivamente al segmento que se extiende entre un observador y un punto final de observación. Tanto el observador como el punto final se localizan sobre la superficie de un determinado terreno. La Cuenca Visual Lineal mide qué zonas de un segmento son o no visible por un observador localizado en un extremo del segmento. Toda la superficie del terreno que no pertenece al segmento de análisis es despreciada en el cálculo de la Cuenca Visual Lineal. Por tanto, la información del terreno a estudiar se ve reducida y el número

de operaciones es pequeño en comparación con la Cuenca Visual normal o 2D que suele analizar toda la superficie de un terreno.

De manera formal, la Cuenca Visual Lineal entre un observador **O** y un punto final P de un terreno se define como el conjunto de puntos $x \in [O, P]$ del segmento $O-P$ tal que la función $Visibilidad(x)$ es igual a 1; siendo la función de $Visibilidad(x)$ exclusiva del segmento $[O, P]$ y exclusiva también del observador **O** a una determinada altura de observación. La función $Visibilidad_{obs}(x)$ indica si un punto x es o no visible por un observador(obs), presentando la función un valor 0 si el punto x no es visible por el observador y 1 si lo es. Todos los puntos visibles por el observador dentro del segmento $[O, P]$ forman tramos visibles y la suma de todos los tramos conforman la Cuenca Visual Lineal en un segmento $[O, P]$ para un observador **O**. La medida de la Cuenca Visual Lineal es longitudinal al medir los tramos de segmentos visibles y se puede expresar en metros lineales visibles.

En la figura 3.1 se muestra un ejemplo de una posible función $Visibilidad(x)$, obtenida a partir de la curva o perfil de superficie $C(x)$ que se muestra en la misma figura. $C(x)$ es un segmento de superficie extraído de un terreno T dado, en el cual se localizan los puntos **O** y P que delimitan un segmento. En dicho segmento se analiza la Cuenca Visual Lineal para un observador localizado en 0, de forma que es necesario el cálculo de la función $Visibilidad(x)$ pertenece exclusivamente al observador posicionado en 0 para el segmento de estudio seleccionado.

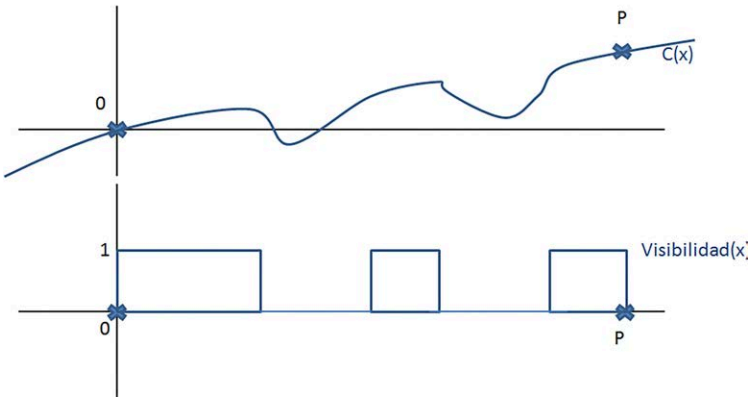


Figura 3.1: Perfil Curva $C(x)$ y Visibilidad para un segmento $[0, P]$ con observador en 0.

La obtención de la función $Visibilidad(x)$ a partir de la curva de superficie $C(x)$ se puede realizar fácilmente mediante el uso de la misma estrategia empleada en el cálculo de los segmentos visibles de los anillos sectoriales visibles, presentada

con anterioridad en la sección 2.2.2. Es decir usando la metodología de los algoritmos de Lineas-de-Visión, que si recordamos permite comprobar sobre un perfil de puntos, que puntos son o no visibles por un observador localizado en cualquier posición dentro del perfil. Como la curva de superficie $C(x)$ es equivalente al perfil de puntos del cálculo de los anillos sectoriales, se puede extrapolar completamente el problema y de esta forma obtener la función $Visibilidad(x)$.

Una vez obtenida la función $Visibilidad(x)$ para un observador, mediante la metodología de los algoritmos de Lineas-de-Visión, el siguiente paso es calcular el valor de la Cuenca Visual Lineal. La expresión matemática que permite el cálculo del valor de la Cuenca Visual Lineal para el observador 0 (CVL(0)) en un segmento continuo $[0, P]$ de superficie con perfil o curva $C(x)$ se define como:

$$CVL(0) = \int_0^P Visibilidad(x)_{C(x)} dx \tag{3.1}$$

La expresión presentada para el cálculo de la Cuenca Visual Lineal (CVL) está orientada a trabajar con espacios continuos. Por tanto, no es valido usar esta expresión directamente en la realización de estudios sobre MDEs de alturas que son de naturaleza discreta. Es necesario una discretización de la expresión matemática continua de la Cuenca Visual Lineal que permita la ejecución de problemas sobre equipos informáticos donde se usa principalmente información discreta, llamándose a esta nueva variable Cuenca Visual Lineal discreta (CVL_d).

Para el cálculo de la (CVL_d) de la curva de perfil $C(x)$ y a la función de $Visibilidad(x)$ se extraen sus homólogos discretos $C_d(n)$ y $Visibilidad_d(n)$ mediante la realización de un muestreo a las curva $C(x)$ y a la función $Visibilidad(x)$. Este muestreo se realiza a priori con la misma resolución que tendría el MDE a analizar o de partida.

En la figura 3.2 se presentan $C_d(n)$ y $Visibilidad_d(n)$ resultantes del muestreo de $C(x)$ y $Visibilidad(x)$ respectivamente. Si se extrajesen $C_d(n)$ y $Visibilidad_d(n)$ directamente del MDE, posiblemente se obtendrían unos muestreos o aproximaciones distintas, ya que la extracción directa de la curva $C_d(n)$ sobre el MDE de partida no es igual al muestreo sobre el espacio continuo directamente. Por ejemplo, si sobre un MDE se selecciona un punto 0 y un punto P arbitrario, el segmento que los une no siempre presentará el mismo número de puntos interiores al segmento 0-P, aunque sea la distancia entre el punto 0 y el punto P la misma. Esta situación puede sorprender, pero un ejemplo de esta situación se puede observar en la figura 3.3, donde se han trazado sobre

un hipotético MDE las lineas que une el punto observador 0 con una serie de puntos finales que van de $P1$ a $P13$. La longitud de cada uno de los segmentos entre el observador y un punto final es muy parecida, pero sin embargo la cantidad de puntos en cada uno de los segmentos y la distancia entre los puntos interiores de los segmentos difieren bastante entre ellos. Como se observa en la figura 3.3, el número de puntos internos es muy irregular, incluso utilizando una banda de puntos adyacentes. Con esta situación es muy complejo extraer de un MDE un perfil de puntos adecuado para el cálculo de la CVL_d, ya que los MDE no poseen toda la información de la superficie a tratar, sino una muestra significativa que puede presentar perdida de información necesaria para una curva $C_d(n)$. Si se quiere reducir este error manteniendo la resolución de muestreo de forma que todos los segmentos de igual distancia presente el mismo número de puntos, entonces es necesaria la aplicación de alguna técnica de interpolación que proporcione los puntos intermedios de los segmentos con una buena aproximación del valor real de la superficie que representara la curva $C_d(n)$.

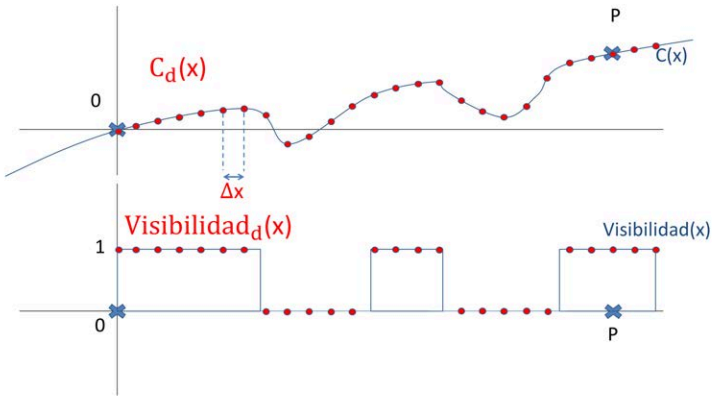


Figura 3.2: Curva continua $C(x)$, versión discreta $C_d(n)$ y las funciones de Visibilidad continuas y discreta para un segmento $[0, P]$ con observador en 0, pertenecientes a un MDE sobre un terreno T.

Una vez obtenida la curva $C_d(n)$ necesaria para obtener la nueva expresión de CVL_d es suficiente con transformar la integral de la función continua de $Visibilidad(x)$ del caso de computo de la CVL en el sumatorio del producto de la función discreta de visibilidad ($Visibilidad_d(n)$) por la distancia existente entre dos puntos consecutivos ($\Delta d(n, n-1)$) del segmento $[0, P]$. Por tanto, la expresión con la que se puede trabajar con equipos informáticos queda de la siguiente forma:

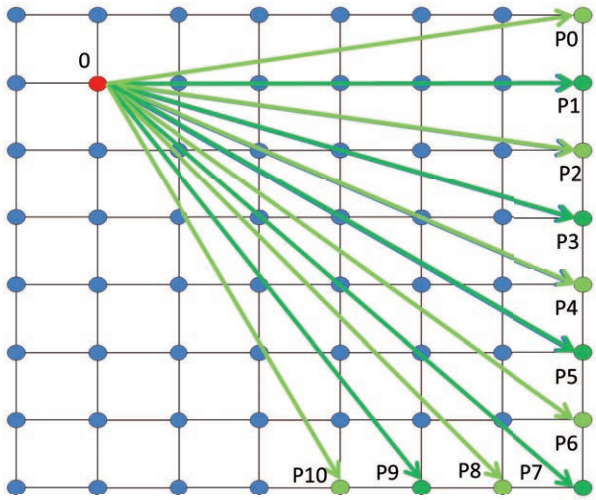


Figura 3.3: Conjunto de segmentos que unen un observador 0 con un conjunto de puntos ($P0....P10$) a los que calcular la Cuenca Visual lineal de cada segmento. Todo ello sobre la malla de puntos de un hipotético MDE.

$$CVL_d(0) = \sum_{n=1}^{n=P} Visibilidad_d(n)\Delta d(n, n-1) \tag{3.2}$$

Es muy importante tener en cuenta que la Cuenca Visual Lineal proporciona soluciones a un número reducido de aplicaciones en la vida real. En la mayoría de los casos las soluciones que aporta la Cuenca Visual Lineal se limitan a problemas en los que es conveniente realizar algún tipo de control o vigilancia sobre segmentos rectos de una superficie. Es necesario en este tipo de problemas el conocimiento de las Cuencas Visuales Lineales, ya que las variaciones en la orografía del terreno hacen que existan zonas de los segmentos a analizar que no sean visibles por un observador a una determinada altura, mientras que si pueden serlo simplemente con variar un poco la altura o la posición sobre el segmento que se está analizando.

En el mundo real es necesario un conocimiento visual del terreno mayor que el que proporciona la Cuenca Visual lineal. Por este motivo, se usan más otros parámetros como la Cuenca Visual 2D que sí aporta más soluciones a los problemas reales de hoy en día. No es de extrañar que los parámetros de estudio

2D y 3D en SIG son los mayoritariamente usados en los problemas de visibilidad, quedando relegado los análisis en una dimensión a muy pocos casos. Además, de las soluciones de visibilidad en 2D y 3D se pueden extraer las soluciones para una sola dimensión, aunque con un coste computacional más elevado que hacer el análisis directamente en una dimensión. Hasta la fecha la gran mayoría de aplicaciones y publicaciones de visibilidad en SIG realizan los estudios en 2D, por ser más sencillo de computar que 3D y abarcar un gran número de aplicaciones donde no es transcendental el conocimiento en 3D para poder implementarse. Por esta razón el trabajo realizado se ha centrado principalmente en el computo eficiente de la Cuenca Visual 2D.

3.1.2. Cuencas Visuales 2D o Viewshed

La Cuenca Visual 2D o simplemente Cuenca Visual de un observador (POV), posicionado dentro de un terreno (T), se define como la superficie de T, visible por POV en todas las direcciones posibles. De forma que si la posición del observador es (x_{POV}, y_{POV}) y la función de visibilidad para dicho observador en todo el terreno es $Visibilidad_{POV}(P)$, siendo P cada uno de los puntos $(P(x, y))$ del terreno T; resulta que la Cuenca Visual de POV, es la superficie que cubre todos los puntos $P(x, y) \in T$, tal que $Visibilidad_{POV}(P) = 1$.

De la definición de Cuenca Visual 2D se ve claramente que es la extrapolación a dos dimensiones de la Cuenca Visual Lineal de carácter unidimensional. En la figura 3.4 se puede apreciar la Cuenca Visual 2D de un observador POV, donde las zonas de color gris representan los puntos o superficie visibles desde el punto POV, es decir el área que conforma la Cuenca Visual 2D, dentro del terreno de estudio. Por otro lado, están las zonas de color rojo, que caracteriza a la superficie invisible entre dos áreas visibles o entre un área visible y el observador directamente. Por otro lado, en la figura 3.4 se presenta zonas con tonalidades que van del color verde al blanco. Dichas zonas no son visibles desde del observador, ni se encuentran entre zonas visibles y el observador, sino que están más lejos de las áreas visibles del observador, por lo que son simplemente zonas del terreno de estudio no visibles por el observador, o que excede la capacidad de visión del observador. Mientras que las Cuencas Visuales Lineales se miden en metros lineales, las Cuencas Visuales 2D tiene una dimensión de superficie y sus unidades son normalmente metros cuadrados visibles.

El cálculo de la Cuenca Visual 2D es un trabajo matemáticamente sencillo; basta con realizar la extrapolación del problema de las Cuencas Visuales Lineales de una dimensión a las dos dimensiones de una superficie, para calcular la Cuenca

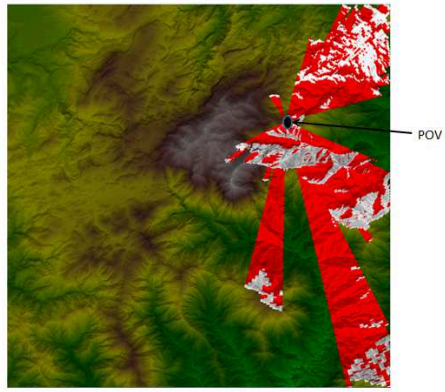


Figura 3.4: Cuenca Visual de un punto aleatorio POV sobre un MDE.

Visual 2D visible por un observador POV en un terreno T.

Las dos dimensiones del problema de las Cuencas Visuales 2D permite usar coordenadas cartesianas o polares para su estudio. En la mayoría de aplicaciones, se calcula la visibilidad circular con radio máximo de alcance, para un observador. Además, se emplea mayoritariamente una metodología de estudio basada en sectores circulares, donde las coordenadas más convenientes son las polares(ρ, θ). De manera que la expresión matemática sobre una superficie de naturaleza continua, para el cálculo de la Cuenca Visual 2D(CV) de un observador (POV), basada en coordenadas polares, queda definida por la siguiente expresión:

$$CV(POV) = \int_{0^o}^{360^o} \int_{0}^{lmax} Visibilidad(P(\rho, \theta)) d\rho d\theta \tag{3.3}$$

La doble integral, en coordenadas polares, hace un barrido de toda la superficie de estudio, como si se tratara de un barrido por sectores de una superficie, ya que la integral interna se asemeja a barrer la superficie de un sector desde el observador hasta el extremo del sector o del MDE y la externa barre todo el conjunto de sectores entre 0 y 360 grados.

La expresión presentada para la Cuenca Visual 2D, es solo valida para el caso de espacio continuo, por lo que, al igual que sucede con las Cuencas Visuales Lineales, es necesario la discretización del problema para poder computar los MDEs en equipos informáticos. La discretización de la expresión para las Cuencas Visuales en 2D es más compleja que en el caso de las Cuencas Visuales Lineales,

ya que, por una parte, estamos tratando de una doble integral y por otro lado, la función $Visibilidad(P)$ es mucho más compleja. Una forma sencilla de realizar la discretización de la doble integral del espacio continuo, es generar un doble sumatorio discreto con los puntos del MDE a estudiar. De esta manera, una posible expresión matemática para el cálculo de las Cuencas Visuales 2D discreta sería:

$$CV_d(POV) = \sum_{sector=0}^{sector=ns} \sum_{P=1}^{P=N} Visibilidad_{POV}(P) \cdot Area(P) \tag{3.4}$$

A priori la expresión de $CV_d(POV)$ hace un barrido de todos los puntos de un MDE, siendo el punto de partida el observador POV. El barrido del MDE como se muestra en la expresión 3.4 se realiza seleccionado en primer lugar un sector usando el sumatorio exterior; y dentro de cada sector se recorre todos los puntos del MDE que se localizan dentro del sector. Una vez sumada el área correspondiente a todos los puntos de un sector se realiza la misma operación para otro sector hasta terminar el barrido de todos los sectores de estudio. Ahora bien, cada punto P del MDE tiene una área propia ($Area(P)$) tanto si es visible como si no, de forma que las suma de todas las áreas visibles por un observador POV es la Cuenca Visual Discreta. Este hecho se refleja en la expresión de la $CV_d(POV)$ como el sumatorio de cada una de las áreas de los puntos del MDE multiplicado por la visibilidad del punto ($Visibilidad_{POV}(P)$).

En la realización del barrido de todos los puntos, si un punto P es visible por el observador POV la función $Visibilidad_{POV}(P)$ presentará el valor uno, cero en caso contrario. Este comportamiento de la función $Visibilidad_{POV}()$ es el mismo que en el caso de la Cuenca Visual Lineal, pero la función $Visibilidad_{POV}()$ correspondiente a las Cuecas Visuales 2D es más compleja de determinar que en el caso de las Cuencas Visuales Lineales, ya que no basta con un único perfil de puntos para analizar la Cuenca Visual 2D de un observador sobre todos los puntos de un MDE, sino que hace falta trazar un perfil de puntos desde el observador a cada uno de los puntos del MDE y estudiar si el punto es o no visible. Este problema, que consiste en comprobar si un punto P de un MDE es visible o no por un observador POV perteneciente al mismo MDE, se estudio en la sección 2.1 de la memoria.

La ejecución del cálculo de la visibilidad por sectores, como ya se mostró en el apartado 2.2.1, permite calcular de una forma eficiente la superficie visible, que viene perfectamente determinada por los llamados anillos sectoriales presentados

en la sección 2.2.2, donde se calculan los anillos sectoriales de un observador para todas las direcciones ($\vec{S}_i$) sobre un terreno. Los anillos sectoriales en una dirección de estudio, determinan el área visible de un observador en dicha dirección y por tanto, la Cuenca Visual 2D de un observador en una dirección de estudio coincide con los anillos sectoriales; siendo la suma de todos los anillos sectoriales en una dirección igual al valor de la Cuenca Visual del observador en dicha dirección de estudio. Por tanto, se puede decir con rotundidad que una vez delimitados los comienzos y finales de todos los anillos sectoriales para un observador (POV) en todas las direcciones o sectores de estudio, se obtiene fácilmente la Cuenca Visual para dicho observador, al ser esta igual a la suma de las áreas de todos los anillos sectoriales visibles para ese observador.

Bajo esta situación, donde las Cuenca Visual es la suma de los anillos sectoriales aparece una nueva forma de calcularlas y por tanto, de presentarla matemáticamente. En esta nueva forma, si se llama $rs_n; n = 1, 2....e_{rsS_i}$ al número de anillos sectoriales de un observador (POV), en una dirección $\vec{S}_i$, resulta que la Cuenca Visual (CV_d) en todas las direcciones ($\vec{S}_i; i = 1...nsec$) para un observador, dentro de un MDE de naturaleza discreta queda expresada por la ecuación 3.5. Con esta expresión queda bien determinadas las Cuencas Visuales para un observador sobre un MDE de partida. De esta forma, para alcanzar las Cuencas Visuales Totales de un MDE solo es necesario aplicar la expresión 3.5 a cada uno de los puntos del MDE.

$$\boldsymbol{CV_d(POV)} = \sum_{\boldsymbol{S_i=1}}^{\boldsymbol{S_i=nsec}} \sum_{\boldsymbol{rs_n=1}}^{\boldsymbol{rs_n=e_{rsS_i}}} \boldsymbol{A_{RS}}$$

$$CV_d(POV) = Cuenca\ \ Visual\ \ del\ \ punto\ \ POV$$
$$S_i = N\acute{u}mero\ \ de\ \ Sector$$
$$nsec = N\acute{u}mero\ \ total\ \ de\ \ sectores$$
$$rs_n = N\acute{u}mero\ \ de\ \ anillos\ \ sectoriales$$
$$e_{rsS_i} = \acute{U}timo\ \ segmento\ \ visible\ \ dentro\ \ del\ \ sector$$
$$A_{RS} = \acute{A}rea\ \ de\ \ cada\ \ anillo\ \ sectorial\ \ visible$$

(3.5)

A la hora de calcular las Cuencas Visuales 2D de un observador suelen aparecer circunstancias que hacen variar el número de puntos a computar en el análisis de la visibilidad sobre la superficie representada por un MDE de estudio. En la mayoría de los casos las circunstancias suelen reducir el número total de puntos a examinar en la computación de visibilidad y en muy pocos casos los

aumentan el número de puntos. Por ejemplo, se quiere estudiar un para metro de visibilidad sobre la superficie de un territorio del cual resulta que la superficie representada por un MDE de partida excede los limites del mismo; entonces bajo en este caso, se debe descartar en el análisis los puntos del MDE que se encuentren fuera de la zona de interés.

Otra situación típica de limitación en el número de puntos del cálculo de las Cuencas Visuales 2D, es la imposición de un limite en el radio máximo de alcance para el parámetro visibilidad deseado. Este echo se debe principalmente a que los equipos de visión (como ojo humano, cámaras de vigilancia, etc...) tienen alcances máximos o distancias máximas de visión, por lo que no tiene sentido en muchas ocasiones la computación de todos los puntos del MDE de partida, como puede ser el caso de algunos radares de control aéreo y marítimo. Otra posible situación que se puede dar es que una zona de un territorio este ya cubierta por un observador o equipo previo, de forma que se busca que otro observador o equipo no tenga en cuenta para su estudio de visibilidad esta zona. Esta situación, se suele dar en sistemas de vigilancia y de telecomunicaciones, donde zonas de un terreno ya cubiertas por un equipo o antena, no deberían de solaparse con la zonas de otro equipo o antena. Además de estas limitaciones físicas y políticas existen otras, como por ejemplo, intereses arqueológicos. Las limitaciones arqueológicas consideran el estado en el que se el encontraba la superficie terrestre en antiguas civilizaciones, las cuales podían encontrarse sumergidas en su época y ahora estar localizadas en un desierto o viceversa.

Todos los casos de variación del número de puntos en los análisis de la visibilidad presentados, han sido de reducción del número de puntos. Sin embargo, existen algunos casos que un MDE no contiene la suficiente información para el área que se desea estudiar, lo que hace que sea necesario aumentar el número de puntos. Suelen presentarse en este sentido dos situaciones. La primera situación es que un MDE no abarca toda una zona de interés, por lo que se requiere el uso de dos o más MDE para cubrir dicha zona, de forma que se incrementa el número de puntos. La segunda situación que se puede dar, es que la información del MDE de partida no es suficiente, por la resolución del mismo, ya que se quiere tener más precisión en los resultados. Por tanto, se necesita generar un nuevo MDE con mayor resolución. Las formas de obtener el nuevo MDE con resolución aumentada son mediante la medida física de más puntos directamente sobre el terreno de estudio o realizando algún proceso de interpolación que permitirá generar un nuevo MDE con mayor resolución.

3.1.3. Implementación secuencial de las Cuencas Visuales Totales para un MDE de tamaño medio

La implementación final del algoritmo de Cuencas Visuales Totales parte del Algoritmo de Visibilidad secuencial presentado en el apartado2.2.1 del capitulo anterior. En concreto se implementa todo el algoritmo presentado en dicho apartado bajo el lenguaje C++ y se modifica solo una parte del mismo para implementar las Cuencas Visuales Totales 2D.

La parte que se modifica solo afecta a la función $Kernel()$ que es llamada dentro de la función *ComputarVisibilidad()*, dejando el resto del algoritmo tal como estaba; ya que como se describe en la sección 2.2.2 dentro de la función $Kernel()$ se delimita los anillos sectoriales los cuales facilitan el cálculo de las Cuencas Visuales. En el pseudocódigo 3.1 se muestra como el cálculo de las Cuencas Visuales se realiza mediante la introducción de una única línea de código (mostrada en color rojo) dentro de la función $Kernel()$. Esta línea de código aprovecha que se esta realizando el cierre de los segmentos visibles, para calcular el área del anillo-sectorial de dicho segmento visible y incrementar el valor de las Cuencas Visuales del observador (CV_{POV}).

El algoritmo desarrollado hace uso de dos bucles anidados, de forma que tras añadir la línea de código que permiten calcular las Cuencas Visuales, se obtiene el valor de la Cuenca Visual de un observador para un sector y su opuesto al finalizar cada una de las iteraciones del bucle *for* interno. Al cambiar de iteración del bucle interno se cambia de observador y se mantiene los sectores de estudio. Mientras que al cambiar de iteración de bucle exterior se cambia tanto de observador como de conjunto de sectores. Lo importante es que al finalizar el recorrido del bucle externo que realiza el barrido de los sectores se obtiene las Cuencas Visibles Totales de un MDE de tamaño medio.

El algoritmo desarrollado almacena el resultado de las Cuencas Visuales Totales en un array de flotantes con el mismo número de elementos que puntos posee el MDE de partida y donde cada uno de los elementos almacena la superficie de las Cuencas Visuales del punto que representa. Para poder hacer posteriores análisis, este mapa de visibilidad se almacena en un fichero binario con extensión .flt que presenta un tamaño de 16 MB, si el MDE de partida tiene un tamaño de 2000×2000 puntos; que como se lleva mostrando a lo largo de todo trabajo es el tamaño que se ha considerado como MDE de tamaño medio. El tamaño de 16 MB se debe al tipo de datos usado para representar y almacenar el valor de la Cuenca Visual de cada punto; en concreto se almacenara mediante una variable o número de tipo float con 4 *byte* de longitud.

Pseudocódigo 3.1: Kernel de Cuencas Visuales.

```
Kernel(d_POV,h_POV,d_P,h_P,&maxanguloanterior,&visible)
  float anguloequivalente = (h_P-h_POV)/(d_P-d_POV);
  bool es_visible = anguloequivalente >= maxanguloanterior
  bool empieza_segmento = es_visible && !visible_segmento
  bool fin_segmento = !es_visible && visible_segmento

  if empieza_segmento then
     //Incremento del número de anillo-sectorial visibles
     Incremento(n_anillo_sectorial)
     //Almacenamiento del comienzo del segmento visible en un array
     Almacena(d_P);
     //Almacena la distancia de comienzo del anillo-sectorial visible
     d_SRS = d_P-d_POV;
  endif

  if fin_segmento then
     //Almacenamiento del final del segmento visible en un array
     Almacena(d_P);
     //Almacena la distancia del fin del anillo-sectorial visible
     d_ERS = d_P-d_POV;

    //Cálculo y incremento de la Cuenca Visual del  observador POV
     CV_POV += (d_ERS^2 - d_SRS^2) * (pi/nsec);
  endif

   //Guarda le estado visible del punto, para la siguiente iteración
   visible_segmento=es_visible;/
  //Actualización del máximo ángulo-equivalente para próxima iteración
   maxanguloanterior = max(anguloequivalente,maxanguloanterior)
end Kernel
```

Los mapas de visibilidad se pueden pasar a imágenes de mapa de visibilidad en escala de colores y ser estos mapas guardados bajo algún tipo de formato de imagen para su posterior estudio visual. Es valido cualquier formato de imagen como puede ser el *.png*, *.jpg*, *.bmp*, ... El estudio de los mapas en formato imagen con escalas de colores permite realizar una comparación visual de forma rápida. Además, permite a cualquier persona sin necesidad de unos conocimientos previos en visibilidad comprender los resultados e incluso llegar a poder utilizar la información resultante. Por otro lado, los ficheros de estas imágenes presentan normalmente un tamaño inferior a la mitad del tamaño en $MBytes$ que los ficheros .flt de partida. Por ejemplo, de todos los formatos de imagen existentes a los que se puede pasar los ficheros $.flt$, se ha seleccionado el formato *.png* en el presente trabajo, bajo el cual los ficheros resultantes tienen unos tamaños entre los $1,5MB$ a casi los $3MB$. Estos tamaños han dependido de si la imagen presenta más o menos uniformidad en sus colores y por tanto, resultados de visibilidad.

Cómo se ha llevado a cabo la conversión o paso de un fichero de resultados $.flt$ a una imagen o mapa *.png* se describe en el apéndice B. Lo único que detallamos

en el presente párrafo es que se realiza una conversión directa entre en valor del las Cuencas Visuales de los puntos y el color que representa dicho valor. Por ejemplo, un punto con un valor de Cuencas Visuales muy alto, le corresponde un color rojo oscuro y un valor bajo un color azul oscuro. Una de las actividades más importante que debe llevar a cabo cualquier programador que implemente un algoritmo de Cuencas Visuales y la posterior conversión de los resultados a imagen de mapas, es la selección de las paleta o escala de colores necesaria para representar los valores de los parámetros de visibilidad de la manera más clara posible; ya que si no elige una paleta adecuada se generara una imagen poco entendible, la cual puede llevar a errores en el análisis de resultados. Por ejemplo, si la escala seleccionada es pequeña puede que se vean truncados valores altos haciendo que no se pueda distinguir con claridad los puntos con mayor visibilidad de otros que tienen algo menos de visibilidad que los primeros. Además, la misma situación puede pasar con los rangos demasiado bajos si no se selecciona una buena escala de colores y rango de valores. Habrá situaciones que sea mejor elegir una escala logarítmica para la paleta o escala de colores, porque el rango de valores es muy grande y variable, de forma que una escala lineal no da una buena imagen de estudio, por la incapacidad de tener un amplio abanico de colores que represente los valores.

En la realización del cálculo de las Cuencas Visuales Totales mediante el uso del algoritmo desarrollado, aparece una información muy útil que puede ser almacenada para posteriores estudios: los comienzos y finales de todos los segmentos visibles de cada uno de los observadores sobre los puntos del terreno al que representa el MDE de partida. Esta información es la Cuenca Visual detallada del observador.

El almacenar la información de los comienzos y finales de todos los segmentos visibles en el barrido sectorial es una tarea recomendable si se piensa que más adelante sera preciso generar el mapa de Cuencas Visuales de un punto o conjunto de puntos para ver la zona de cobertura o visión de los mismos. Si no es necesaria esta información es mejor no almacenarla por la cantidad de memoria necesaria y el incremento de los tiempos de computo por realizar el almacenamiento de un mayor número de variables, que además suelen ser de gran tamaño. Por tanto, si no se almacena los segmentos visibles, se ahorra espacio en memoria y tiempo de procesamiento. Pruebas realizadas en el presente trabajo utilizando MDE de tamaño medio han arrojado ficheros de más de un centenar Gbytes de espacio. Por ejemplo, en el caso concreto de estudio del MDE de tamaño medio de la ciudad de Málaga, con coordenadas UTM(zona 30S), N=4080000, E=360000 y resolución $10 \times 10m^2$, ha generado un fichero de unos 121 Gbytes de espacio. El tamaño del fichero con la información de todas las Cuencas Visuales de un MDE

de tamaño medio depende del número de anillos sectoriales que pueda presentarse en el cálculo de las cuencas visuales, de forma que este número no es para nada constante, sino que depende en gran medida de la rugosidad del terreno, ya que a mayor rugosidad el número de anillos sectoriales o segmentos visibles suele ser mayor, por lo que el espacio en memoria requerido también es superior.

3.1.4. Resultados de la implementación secuencial

A la hora de verificar los resultados del algoritmo desarrollado, se ha realizado el cálculo de la Cuenca Visual para 30 localizaciones diferentes de un observador perteneciente a un MDE de tamaño medio, que cubre una parte de la provincia de Málaga con resolución $10 \times 10m^2$. Las coordenadas UTM de dicho MDE son zona 30S y la coordenada del primer punto superior izquierdo es 4080000N, 360000E. A este MDE le hemos llamado MDE de la ciudad de Málaga para distinguirlo de otros y cada vez que se haga referencia al mismo no indicar sus coordenadas UTM.

Los resultados obtenidos para las 30 localizaciones diferentes mediante el algoritmo desarrollado, se han comparado con dos de las aplicaciones más utilizadas en los Sistemas de Información Geográfica, que son ArcGIS y GRASS. También se han comprobado los resultados con el único modelo de Cuencas Visuales Totales encontrado en la literatura [37]. Por las limitaciones de este último, la comparación de los resultados se ha realizado con los observadores ligados al suelo, es decir con altura $h = 0m$ sobre la superficie del terreno. Los valores obtenidos guardan una gran semejanza respecto a los alcanzados mediante ArcGIS. Donde en todos los casos las diferencias han sido inferiores al 5 %. Esto es más que suficiente dada la inestabilidad numérica del problema subyacente (ligerísimas variaciones en la posición del observador pueden provocar enormes diferencias en el tamaño de la Cuenca Visual). En el cálculo de las Cuencas Visuales se consideran usualmente aceptables los resultados del mismo orden de magnitud, como se vio en la sección 2.1.3. Para comprobar la validez del algoritmo desarrollado en todos los ámbitos, se han comparado las soluciones con observadores no ligados al suelo (en las aplicaciones que lo permiten). Siendo las diferencias de las soluciones inferiores al 8 %. Esto garantiza la validez de los resultados obtenidos, y en consecuencia la del algoritmo desarrollado en la presentes tesis para el cálculo de los mapas de Cuencas Visuales.

En la figura 3.5 se presenta un ejemplo de computación de las Cuencas Visuales sobre un punto utilizando todos los algoritmos descritos en el párrafo anterior. En concreto se realiza el análisis sobre un punto con coordenadas UTM

zona 30S, N=4070140 y E=368800. Este punto se encuentra dentro del MDE de tamaño medio de la ciudad de Málaga con resolución de $10 \times 10m^2$.

(a) (b)

(c) (d)

Figura 3.5: Resultado de diferentes herramientas para el cálculo de la Cuenca Visual del punto con coordenadas UTM(zona 30S), N=4070140, E=368800, sobre el MDE de la ciudad de Málaga: (a) resultado algoritmo desarrollado en esta tesis, (b) resultado del modelo propuesto en el articulo [37], (c) resultado con la herramienta GRASS y (d) salida Viewshed tool bajo la herramienta ArcGIS.

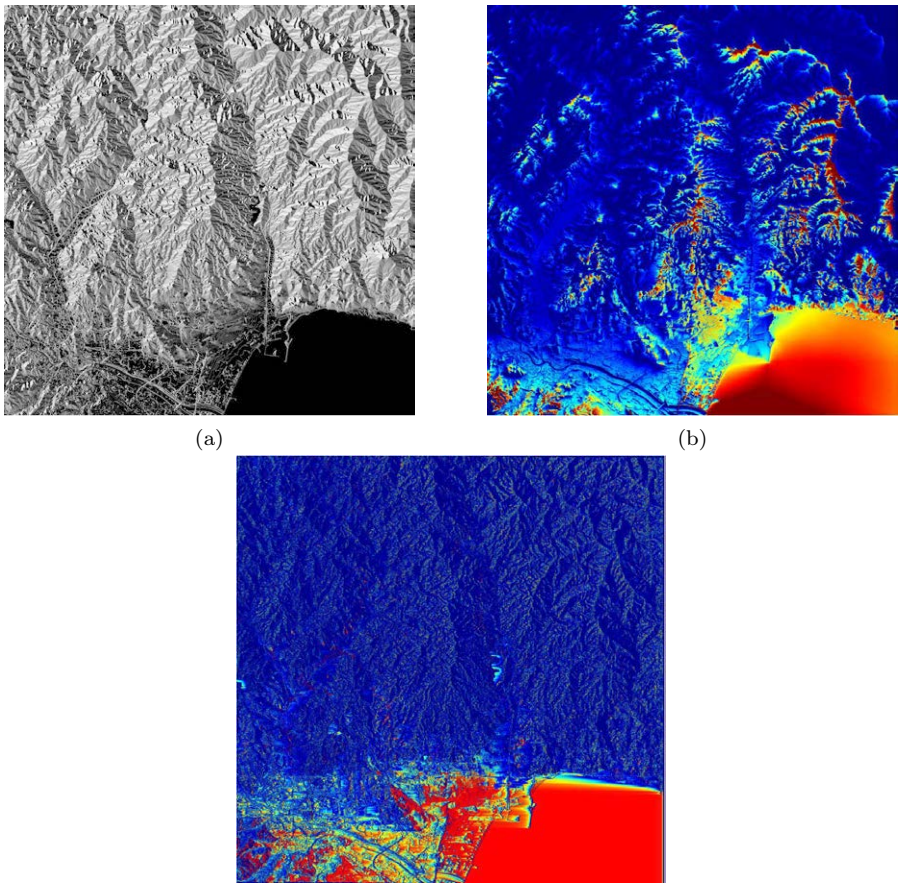
Tras observar las imágenes de la figura 3.5 se puede comprobar visualmente como la solución resultante del algoritmo desarrollado (figura 3.5a) es parecida a las soluciones aportadas por los otros algoritmos, con los que se ha comparado, para el mismo punto de estudio.

En la figura 3.6a se muestra en tonos de blanco y negro el MDE de tamaño medio de la ciudad de Málaga. Los puntos en color negro son representativos de los puntos con menor altura respecto el nivel del mar y los puntos más blancos representan los de mayor altura. De forma que se ve claramente como la zona inferior derecha, que es totalmente negra, pertenece al mar Mediterráneo que baña las costa de la ciudad de Málaga. Al presente MDE se le ha calculado las

Cuencas Visuales Totales para observadores situados a 2 metros de altura sobre la superficie del terreno y se ha obtenido el mapa de Cuencas Visuales Totales de la figura 3.6b, que es uno de los primeros mapas de visibilidad sin restricciones respecto a la ubicación del observador y sin radio límite de observación sobre MDEs de tamaño medio. La representación del valor de la Cuenca Visual de un punto sobre el mapa se ha realizado en escala de colores, que van en un rango de 0 km^2 a 200 km^2 de superficie visible en todas las direcciones para los observadores. Los colores van desde el azul oscuro, representativo de los observadores con bajo valor de Cuenca Visuales (cercanos a $0km^2$), al color rojo para los puntos con alto valor de Cuencas Visuales (cercanos a $200Km^2$). Por último, la figura 3.6c muestra un mapa con la diferencia en el cálculo de las Cuencas Visuales Totales, cuando los observadores se encuentran localizados a 2 metros de altura respecto el suelo y a 0 metros o pegado a el. En este mapa comparativo, se ha modificado la escala de colores, de forma que los valores representados van desde 0 Km^2 a 50 km^2, siendo el color rojo para los puntos que presenta una mayor diferencia de sus Cuencas Visuales (diferencia cercana a $50Km^2$), por la variación de 2 metros de altura del observador respecto del suelo. El color azul oscuro del mapa de diferencias indica que los puntos presentan diferencias mínimas (diferencia cercana a $0Km^2$) por la variación de 2 metros de la altura el observador. Con esta última figura de diferencias del mismo MDE pero con los observadores posicionados a distintas altura, queda demostrado el potencial del algoritmo desarrollado para calcular las Cuencas Visuales de cualquier punto y a cualquier altura a partir de un MDE de alturas. Por otro lado, esta figura de diferencias demuestra como afecta la altura de un observador en el cálculo de la visibilidad, ya que el mismo punto por la variación de altura al que se ha sometido cambia hasta en $50km^2$ su capacidad de ver más superficie.

Tiempos de Ejecución de la implementación secuencial

Una vez comprobado la validez de los resultados de las Cuencas Visuales, es muy importante medir la eficiencia de cálculo del algoritmo desarrollado mediante el conocimiento de los tiempos de computación y la comparación con otros algoritmos. Experimentalmente, se ha calculando las Cuencas Visuales del MDE de tamaño medio de la ciudad de Málaga. Para la ejecución con el algoritmo desarrollado se ha usado una ventana BOS de tamaño $bw = 1001$ puntos, que se ha ejecutado sobre un equipo de sobremesa, cuyas características se puede ver dentro de la sección A.1, del apéndice. Además sobre el mismo equipo se ha ejecutado los diferentes algoritmos o herramientas para el cálculo de las Cuencas Visuales.



(a) (b)

(c)

Figura 3.6: (a) MDE de la ciudad de Málaga, coordenadas UTM(zona 30S), N=4080000, E=360000, y resolución de $10 \times 10m^2$. (b) Cuencas Visuales Totales con observador a 2 metros altura, color rojo para puntos de máxima Cuencas Visuales y azul oscuro para mínimas. (c) mapa comparativo de las Cuencas Visuales medidas entre observadores a 2 y 0 metros sobre la superficie.

El tiempo empleado en el cálculo de la Cuenca Visual de un sólo punto en el MDE de 2000 × 2000 puntos, usando el algoritmo realizado, ha sido 0.0032 segundos. Este tiempo ha sido claramente inferior a los resultados obtenidos con los demás algoritmos comparados. Que han sido de 0,0063 segundos para la computación mediante el algoritmo del articulo [37], mientras que las

herramientas Viewshed tool ArcGIS y GRASS han necesitado de 10 y 155 segundos respectivamente, bajo las mismas condiciones. En el caso de calcular el Mapa de Cuencas Visuales Totales, se tarda con el algoritmo desarrollado 12836 segundos y con el trabajo [37] unos 25500 segundos, lo que son $3,56$ y $7,08$horas respectivamente. Con los otros dos algoritmos ni siquiera se ha intentado, ya que a priori se necesitaría más de un mes e incluso más de un año de computo ininterrumpido para obtener las Cuencas Visuales de todos los puntos de un MDE de tamaño medio.

Estos tiempos demuestran que los algoritmos existentes para el cálculo de la Cuencas Visuales son bastante lentos y por tanto ineficaces en el cálculo de Cuencas Visuales Totales. Además otro lado, los tiempo demuestran que el algoritmo desarrollado es una solución altamente útil en el computo de la visibilidad sobre MDE de tamaño medio, ya que proporciona soluciones precisas en unos tiempos razonables.

3.1.5. Implementación paralela de las Cuencas Visuales Totales para un MDE de tamaño medio y resultados

Como se ha podido ver en la sección anterior los tiempos de cálculo necesarios para la computación de las Cuencas Visuales Totales son extremadamente elevados y aunque el algoritmo desarrollado aporte soluciones en unos tiempos más que razonables, si los comparamos con los demás algoritmos de la bibliografía, resulta interesante reducir lo máximo posible los tiempos de computación, ya que se esta hablando todavía de la necesidad de más de tres horas de ejecución en la obtención de un mapa de Cuencas Visuales totales. Por ello tiene un gran valor el poder hacer paralelo de forma eficiente el algoritmo de visibilidad desarrollado, ya que gracias a la paralización se reducen los tiempos de computo totales.

Como se mostró en la sección 2.2.3 del presente trabajo, no resulta excesivamente complejo hacer paralelo el algoritmo desarrollado, ya que se puede hacer paralelo el primer bucle exterior del algoritmo, gracias a la estructura empleada. Recordamos que este bucle exterior realiza el barrido de los sectores y hacer paralelo este bucle apenas modifica el código original, por la gran independencia existente en la computación sobre distintos sectores. Los únicos datos que comparten las distintas iteraciones del bucle exterior son solo de lectura (el MDE de partida), y una variable de escritura común para almacenar el resultado final. Por tanto, una iteración del bucle exterior no necesita la ejecución de otra iteración previa para la computación de cualquier sector. Sin embargo la computación del bucle interior es dependiente del sector y de iteraciones previa del

mismo bucle, ya que por ejemplo, al principio del bucle interno la estructura BOS esta vaciá y empieza una fase de llenado, para seguir con una de desplazamiento y terminar con una de vaciado. Esta gestión de la estructura BOS hace imposible ejecutar una iteración del bucle interno del algoritmo si no se han realizado previamente todas las anteriores anteriores a la actual. Por tanto, la iteración actual tiene que espera la finalización de la anterior y es prácticamente imposible hacer paralelo el bucle interior de una forma eficiente.

Tras la realización de numerosos experimentos bajo la implementación paralela del algoritmo y la comparación de los mismos con las ejecuciones secuenciales, se ha comprobado que los resultados para las Cuencas Visuales Totales son los mismo en tanto que la ejecución paralela como la secuencial se realice bajo el mismo sistema o equipo informático. Es decir, que en los experimentos realizados ha aparecido que los resultados de la ejecuciones del algoritmo implementado difieren si se ejecuta el algoritmo sobre distintos equipos informáticos.

La variación de los resultados encontrada por la computación del algoritmo en distintos sistemas informáticos es bastante reducida y no se da para todos los puntos del MDE de partida. La variación de los resultados se da principalmente en los puntos con mayor visibilidad y afecta de forma general a menos del $0,003\,\%$ (113 puntos de 4 millones de puntos) de los puntos de un MDE de tamaño medio y además las diferencias en en valor de las Cuencas Visuales ha sido inferior a $16m^2$ sobre puntos con visibilidad superior a los $100Km^2$. Como el número de puntos afectados es muy bajo, y además las variaciones son ínfimas en comparación con el valor total de visibilidad, consideramos que los resultados son validos tanto en unos equipos como otros. Tras analizar las causas de las variaciones de los diferentes experimentos realizados entre equipos, hemos llegado a la conclusión que estas diferencias en los resultados obtenidos están asociadas a los procesos de redondeo sobre las operaciones en punto flotante de las distintas arquitecturas de CPU sobre las que se ha ejecutado el algoritmo desarrollado, ya que el número de decimales en algunas operaciones es bastante alto.

Al ser los resultados de la versión paralela y secuencial idénticos cuando se ejecuta sobre el mismo sistema informático, resulta que la parte realmente importante de este apartado es el estudio del efecto en los tiempo de la ejecución paralela con distinto número de cores y el rendimiento que se alcanza bajo la versión paralela.

Tiempos de Ejecución de la implementación paralela

Ante las limitaciones de realizar ejecuciones sobre un ordenador de sobremesa, que suele poseer un número de cores bajo, se ha ejecutado el algoritmo paralelo en uno de los nodos del supercomputador Picasso del Centro de Supercomputación de la Universidad de Málaga. El nodo usado es un equipo HP DL980G7 con 80 cores. Todas las características del nodo se pueden ver en el apéndice A.2.

La ejecución del algoritmo secuencial dentro de uno de los nodos HP DL980G7 del supercomputador Picasso ha arrojado un tiempo medio de cálculo por sector de casi 26 segundos para el cálculo de las Cuencas Visuales Totales del MDE de la ciudad de Málaga. Este valor representa un 10 % del rendimiento teórico máximo (ver sección D), lo cual resulta razonable, ya que las operaciones en punto flotante están dominadas por las divisiones. El tiempo total empleado por un core para calcular el mapa de Cuencas Visuales Totales en todos los sectores es de unos 4706 $segundos = 1,3\ horas$ de media, lo que supone casi un tercio del tiempo respecto del equipo de sobremesa del apartado 3.1.4.

Empleando siempre el mismo MDE de tamaño medio de partida (MDE de la ciudad de Málaga), se ha realizado una serie de pruebas donde se comprueba la escalabilidad de la versión paralela del algoritmo. Para ello se ha realizado el cálculo de las Cuencas Visuales Totales y la generación de sus mapas correspondientes mediante la realización de ejecuciones con distinto número de cores. El número de cores seleccionados para realizar las pruebas de ejecución paralela han sido 1, 2, 4, 8, 16, 32, 64 y 80. Para cada uno de los números de cores seleccionados se han realizado diez ejecuciones, a las cuales se les han medido los tiempos de ejecución. De los tiempos obtenidos se han seleccionado para cada número de cores, ocho de las diez ejecuciones, descartando la ejecución que más tiempo ha tardado y la que ha tardado menos. A cada conjunto de ocho ejecuciones se le ha hecho la media de los tiempo dando como resultado los tiempos de la tabla 3.1. Y además se han calculado el factor de mejora (speed-up) y la eficiencia. Con estos datos se ha construido la figura 3.7, donde se presenta por un lado el tiempo de ejecución en función del número de cores y por otro lado, el speed-up ó factor de mejora del rendimiento de las versiones paralelas.

El speed-up($S(n)$) y la eficiencia ($E(n)$) nos permiten evaluar el comportamiento de nuestro algoritmo en cuanto al cálculo paralelo. Recordamos que el speed-up se define como:

$$S(n) = \frac{T(1)}{T(n)} \tag{3.6}$$

Siendo $T(1)$ es el tiempo de ejecución de un solo core o secuencial y $T(n)$ el tiempo de ejecución de n cores.

Y la eficiencia se define como:

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)} \tag{3.7}$$

Como bien es sabido tanto el speed-up como la eficiencia dan una idea de la ocupación de los cores y del nivel de paralelización de un algoritmo. Si el valor del speed-up es cercano al número de cores, entonces la eficiencia sera cercana a su valor máximo ($E(n) = 1$).

Número de cores	1	2	4	8	16	32	64	80
Media Tiempos Ejecución	4705,3	2486,7	1301,0	623,7	295,8	160,9	80,9	76,8
Speed-up($S(n)$)	1	1,89	3,61	7,54	15,96	29,22	58,14	62,47
Eficiencia($E(n)$)	1	0,94	0,91	0,94	0,99	0,91	0,89	0,78

Tabla 3.1: Media de tiempos de ejecuciones paralelas en segundos.

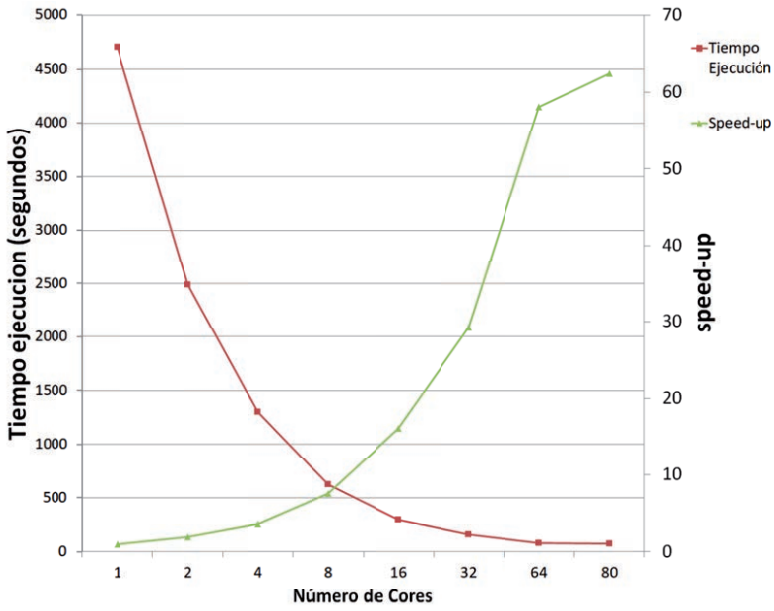


Figura 3.7: Tiempos de ejecución y speed-up de nuestro algoritmo en la generación de un mapa de Cuencas Visuales Totales, sobre un MDE de tamaño mediano.

Tras estudiar tanto la figura 3.7 como la tabla 3.1 se puede apreciar que el algoritmo escala de una manera excelente en su forma paralela, cuando el número de cores presenta valores bajos, ya que la reducción en el tiempo de ejecución y la variación sobre el valor del speed-up es prácticamente proporcional al número de cores que emplea la ejecución. Por ejemplo, si nos fijamos en la ejecución con 2 cores el tiempo pasa a ser cercano la mitad de un solo core y el speed-up cercano al doble (1,89); si nos fijamos en 16 cores, el tiempo es cercano a un dieciseisavo y el speed up casi dieciséis (15,96). Ahora bien, si se compara el tiempo de la ejecución 80 cores, resulta que el speed-up que arroja es aproximadamente de 62. Además, se observa en la figura 3.7 como apenas hay salto del speed-up de la ejecución de 64 cores a la de 80. Todos estos datos pueden llegar ha hacer pensar que el algoritmo no es escalable para valores altos del número de cores, pero en realidad es una consecuencia del número de iteraciones del bucle exterior, que es el que se hace paralelo. En concreto el bucle exterior posee siempre un número de 180 iteraciones, resultando que si se realiza una ejecución con un número de cores que no es divisor entero de 180, entonces habrá cores que realicen más iteraciones del bucle *for* que otros cores que permanecerán parados mientras los primeros trabajan, haciendo que el rendimiento global baje. Para corroborar este hecho se ha realizado pruebas con 15, 30 y 60 cores, que son números divisores de 180 y que un solo nodo HP DL980G7 permite ejecutar. Tanto los tiempo obtenidos como las eficiencias que arrojan dichos tiempos se muestran en la tabla 3.2 junto con los valores de las ejecuciones realizadas con 16, 32 y 64 cores. Como se puede observar en la tabla 3.2 los tiempos para ejecuciones con de 15, 30 y 60 son parecidos a las ejecuciones con 16, 32 y 64 cores, lo que hace que la eficiencia de las ejecuciones con un número de cores divisor de 180 sea mayor.

Número de cores	15	16	30	32	60	64
Media Tiempos Ejecución	296,6	295,8	162,6	160,9	81,22	80,9
Eficiencia($E(n)$)	1,05	0,99	0,94	0,91	0,95	0,89

Tabla 3.2: Media de tiempos de ejecución paralela en segundos.

La conclusión de este apartado lleva a afirmar que el algoritmo desarrollado es altamente paralelo y que con un equipo equipo con 180 cores y suficientes recursos de memoria se puede obtener el mapa de Cuencas Visuales Totales de un MDE de 2000×2000 puntos en menos de un minuto, ya que, aunque no se ha podido realizar una ejecución con un equipo de 180 cores, si se ha conseguido obtener el mapa de Cuencas Visuales Totales en 1,25 minutos con de 60 cores. Por tanto, si el equipo posee 3 veces más cores con igual o más potencia de cálculo, el tiempo de computo sera menos de la mitad del tiempo obtenido con la ejecución de 60 cores del nodo Picasso.

3.1.6. Implementación paralela de las Cuencas Visuales Totales para un MDE de tamaño grande

Este apartado se ha centrado en realizar el cálculo de las Cuencas Visuales Totales para el MDE de Andalucía con resolución de $10 \times 10m^2$. Dicho MDE presenta un tamaño de $50000 \times 30000 = 1500$ *millones* de puntos, de ahí que se llame MDE de tamaño grande. Para poder realizar la computación de las Cuencas Visuales Totales, de una forma eficiente, se debe usar la estrategia para el cálculo de parámetros de visibilidad sobre MDEs de tamaño grande presentado en la sección 2.2.4, junto con las ecuaciones usadas en la implementación de las Cuencas Visuales Totales vistas en la sección 3.1.2 del presente capítulo. Como ya se mostró en la sección 2.2.4, tratar un MDE de estas dimensiones con la misma estrategia que para los MDE de tamaño medio, es una tarea imposible.

Según la estrategia desarrollada en la sección 2.2.4 del capitulo anterior, para la computación de un MDE de tamaño grande con unos 1500 millones de puntos, se hace previamente una división en 1500 MDEs de tamaño medio ($2000 \times 2000 = 4$ *millones* de puntos) que se solapan parcialmente. En concreto la parte mínima o número de puntos que se solapan o comparten los MDEs de tamaño medio, es igual al tamaño de una cuadricula que en este caso es de $1000 \times 1000 = 1000000$ puntos por ser el MDE de tamaño medio elegido 2000×2000 puntos. A su vez el máximo número de puntos que pueden llegar a solaparse o ser los mismos pero en dos MDEs distintos es el equivalente a 2 cuadriculas, es decir 2 millones de puntos. A cada uno de los MDEs de tamaño medio parcialmente solapados se le calcula las Cuencas Visuales Totales solo las direcciones o sectores con dirección hacia el interior del MDE de tamaño medio. De esta forma, como ya se vio en la sección 2.2.4, el análisis permite asegurar la ausencia del efecto borde en la computación con un radio máximo igual a una línea de la cuadricula. Al ser las cuadriculas de 1000×1000 puntos y la resolución del MDE de $10 \times 10m^2$, el radio esta limitado a $1000 \times 10 = 10$ Kilómetros.

El fichero que contiene el MDE de 1500 millones de puntos de partida ocupa cerca de 6 GBytes de memoria y los 1500 MDEs de tamaño medio en los que se ha dividido llegan a ocupar casi $23,5$ Gbytes de memoria. Además, los resultados van a necesitar tanto espacio como los MDEs de entrada tras la computación de las Cuencas Visuales Totales de los 1500 MDEs de tamaño medio que después se deben sumar y combinar para llegar ha generar el MDE de tamaño grande final. Tras conocer el tamaño aproximado de las necesidades de memoria para los datos de entrada y salida se puede comprender como para llevar a cabo la computación del MDE de tamaño grande es necesario disponer de más de un par de cientos de GBytes de memoria principal para poder manejar los datos de entrada salida

junto con las variables auxiliares que intervienen en el proceso de cálculo. Una vez vistas las enormes necesidades de memoria principal para este tipo problema, queda más que demostrado que la computación de las Cuencas Visuales Totales de un MDE de tamaño grande necesita más recursos que los que puede dar un simple equipo de sobremesa con dos cores y 6 GBytes de memoria RAM, para obtener los resultados en unos tiempos razonables y no tener que esperar a más de un mes de computación ininterrumpida.

En el presente trabajo para computar el MDE de tamaño grande, se ha vuelto a emplear los nodos HP DL980G7 del supercomputador Picasso, que poseen 80 cores y 2TBytes de memoria RAM por nodo. En concreto se ha realizado la computación con cuatro nodos HP DL980G7, que permite realizar la computación de los MDEs tamaño medio de forma eficiente y rápida.

Para llegar a realizar la implementación del cálculo de las Cuencas Visuales Totales del MDE de Andalucía se han adaptado los 4 pasos de la implementación básica, para la computación de parámetros de visibilidad sobre MDE de tamaño grande, que se mostró en la sección 2.2.4, a los recursos de computación de los nodos del supercomputador Picasso. Los 4 pasos quedan de la forma que se enumera a continuación:

1. Paso 1. División del MDE de tamaño grande en 1500 MDEs de tamaño medio, desplazados.

2. Paso 2. La computación paralela, usando el algoritmo desarrollado con limitación de sectores de estudio de MDEs de tamaño medio en 4 nodos HP DL980G7, de forma que cada nodo computa 375 MDEs de tamaño medio.

3. Paso 3. Suma de los resultados de los 1500 MDEs de tamaño medio parcialmente solapados y con computación parcial de sectores, en 375 MDEs de tamaño medio que no se solapan y que tienen la información de la visibilidad para los todos puntos en todos los sectores.

4. Paso 4. Obtención de las imágenes de Cuencas Visuales Totales y fichero de resultado de la Cuencas Visuales Totales del MDE de tamaño grande.

El paso 1, para ejecutarlo sobre los nodos HP DL980G7 no hay que modificar nada sobre lo explicado en la sección 2.2.4, simplemente es generar 1500 MDEs de tamaño medio, desplazados. En el paso 2 se ha hecho un reparto de los MDEs a computar por cada nodo previamente a la ejecución. Es decir se han elegido de los 1500 MDEs de tamaño medio cuales va a computar cada uno de los nodos

de una forma fija. El realizar el reparto a priori y no dinámicamente dentro de la propia ejecución ha sido principalmente por dos motivos. El primero por la dificultad en algunas pruebas de tener totalmente disponible los cuatro nodos del supercomputador Picasso, que es compartido por varias decenas de investigadores a la vez y la forma en la que entran las ejecuciones de los distintos investigadores es por orden según cantidad de recursos se reservan y tiempo en cola, lo que hace que si se pide muchos recursos puede tardar días en la entrada de la ejecución del programa dentro de los nodos. Y el segundo motivo es la imposibilidad de que cada nodo pueda ejecutar de forma paralela más de un MDE de tamaño medio de forma eficiente, ya que el número de cores de cada nodo es de 80 cores y por tanto no supera los 180, que es el número máximo de paralelismo para un MDE de tamaño medio; de forma que en un nodo solo puede computarse a la vez un MDE de tamaño medio de forma eficiente y por tanto hay que esperar la finalización de la computación de un MDE de tamaño medio antes de comenzar con la computación del siguiente.

En la realización del paso número 3, se obtiene MDEs de tamaño medio que no solapa la información de los sectores de los puntos con la información de otros MDEs, pero que los puntos dentro de cada MDE tiene la información de las cuencas Visuales en todos los sectores para cada punto. Para la generación de cada uno de los 375 MDEs que no solapan información de sectores, basta con con abrir 9 MDEs de tamaño medio parcialmente solapados y sumar la información para cada punto correspondiente a cada conjunto de sectores para tener la información de todos los sectores para ese punto. Esta tarea se ha implementado de forma paralela, ya que no se modifican los MDE de tamaño medio de partida, sino que se generan unos nuevos con la información de todos los sectores. El nivel de paralización que se puede alcanzar es tantos hilos como MDE independientes compongan el MDE de tamaño grande, en el presente trabajo este número es de 375, por lo que se ha lanzado un bucle *for* con tantos hilos como MDE de tamaño medio resultantes finales.

El último paso es la combinación de los MDE de tamaño medio que no se solapan en el MDE de tamaño grande que se desea obtener. Esta tarea se ha realizado de forma secuencial, ya que es una tarea de escritura sin computación ninguna donde se abre los archivos y se van concatenando la información según el orden de los distintos MDE de tamaño medio. El problema que se presenta en esta tarea es el tamaño de las variables a usar, que aunque básicamente el trabajo es una concatenación de archivos, el tener un gran número de los mismos abiertos hace que resulte una tarea pesada, para el equipo informático, ya que hay leer la información de 375 ficheros que ocupan en total un tamaño de unos $6GBytes$ para guardarla en un único fichero con un tamaño de $6GBytes$. Además,

si posteriormente se desea obtener el mapa o imagen de Cuencas Visuales Totales en formato *.png* que ocupa un espacio cercano a los 1.2GBytes, la cual se puede bajar su resolución llegando a guardarse es una imagen de tamaño inferior a 2Mbytes, para tener una simple representación de la misma.

El trabajar con ficheros de tamaños de GBytes sobre equipos de sobremesa para la realización de ciertos estudios es una tarea bastante ardua, ya que estos equipos no suelen tener los recursos suficientes para dicha tarea. Ahora bien si lo que se busca es solamente el análisis de la imagen final la situación cambia, ya que existe otra forma de obtener la imagen final del MDE de tamaño grande. Esta forma consiste que en el paso 3 en vez de obtener un fichero, con la información del MDE de tamaño medio que no se solapa, se obtiene solamente la imagen del mapa de Cuencas Visuales Totales de cada MDE que no se solapa. De forma que se pueden concatenar directamente los 375 mapas de Cuencas Visuales totales, para obtener el mapa del MDE de tamaño grande. En el presente trabajo se ha hecho esta concatenación mediante el uso de la función *cat* en linux, que permite concatenar ficheros de formato imagen de una manera sencilla. Además si se obtienen previamente los mapas de Cuencas Visuales Totales de cada MDE de tamaño medio que no se solapa, se pueden detectar errores en la computación antes de que acabe el algoritmo, ya que si el primer mapa de Cuencas Visuales Totales de tamaño medio se produce un error, seguramente los siguientes mapas se repita en mismo error.

3.1.7. Resultados para Cuencas Visuales Totales para un MDE de tamaño grande

Una vez realizada todo la computación se ha obtenido el Mapa de Cuencas Visuales Totales de Andalucía con radio máximo de visibilidad de $10Km$ y con altura de 0 metros sobre en nivel del mar. El Mapa en cuestión se presenta en al figura 3.8. La escala de colores representan valores de 0 Km^2 a $200Km^2$. Los puntos con máxima Cuencas Visuales están en color rojo mientras que los de mínima en color azul. Se puede observar en la figura como los puntos costeros y los marítimos cercanos a la costa, presentan en general una alta visibilidad y como puntos marítimos más alejados de la costa su visibilidad es nula. Este hecho simplemente justifica que el radio máximo de visibilidad es de 10km lo que hace que puntos alejados más de 10km de la costa bajo la computación realizada no alcanzan a divisar ningún punto que este por encima de su altura y por tanto, no tienen visibilidad, mientras que, puntos que se encuentran alrededor de la costa, divisan gran cantidad de superficie, ya que pueden ver todo el litoral costero hasta 10km de distancia.

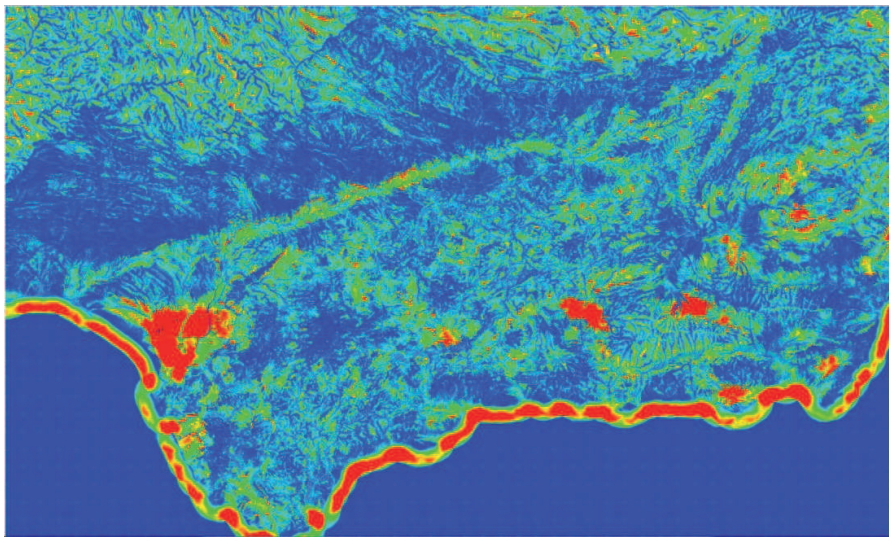


Figura 3.8: Mapa de Cuencas Visuales Totales de la comunidad autónoma de Andalucía.

La parte más importante de la computación realizada son los tiempos en los cuales se ha obtenido el Mapa de Cuencas Visuales totales, ya que el presente trabajo ha alcanzado la computación de todo el MDE de 1500 millones de puntos en unas $13,56$ horas de computo. Este tiempo de computo alcanzado es inferior al que cualquier otro trabajo anterior podría alcanzar e incluso ni pensar.

3.2. Cálculo de la Visibilidad Volumétrica

La Visibilidad Volumétrica se mostró en el apartado 1.5 del capítulo de introducción, mediante su definición practica y un ejemplo gráfico de la misma. El presente apartado se extiende la definición practica de la Visibilidad Volumétrica y muestra la definición matemática discreta, que es la que se implementa en definitiva bajo la metodología del algoritmo desarrollado.

La Visibilidad Volumétrica de un observador (POV), posicionado dentro de un terreno (T), se define como el volumen visible por POV en todas las direcciones, pero limitado en última instancia por su parte superior mediante las líneas o plano del horizonte del observador (HO_{POV}). De forma que si la posición del observador es (x_{POV}, y_{POV}) y la función de visibilidad para dicho

observador en todo el terreno es $Visibilidad_{POV}(P)$, siendo P cada uno de los puntos $(P(x, y))$ del terreno T; resulta que la Visibilidad Volumétrica de POV es la suma de los volúmenes o espacio visible limitados en su parte superior por los planos del horizonte HO_{POV} y en la parte inferior por la superficie visible que cubre todos los puntos $P(x, y) \in T$, tal que $Visibilidad_{POV}(P) = 1$.

En la figura 3.9 se muestra un ejemplo de Visibilidad Volumétrica sobre un hipotético perfil bidimensional de una superficie. Sobre este perfil se localiza casi en el centro un observador, al cual se estudia su Visibilidad Volumétrica o volumen visible para este perfil de partida, en una dirección y su opuesta. El volumen visible para el observador de la figura 3.9 se encuentra comprendido dentro de la zona coloreada en tonos con degradado de amarillo y rojo, siendo el amarillo el volumen visible cercano y el rojo para el volumen visible lejano. Además en la figura 3.9 se presentan en color verde los puntos visibles que forman la superficie visible por el observador y por otra parte en color azul se muestra los puntos no visibles por el observador, sobre los que se posiciona el volumen no visible. Por tanto, en la figura 3.9 se diferencia claramente el volumen visible y los volúmenes no visibles que están dentro de la zona de visibilidad o alcance del observador. Como ya se sabe si no hay limite de radio máximo de alcance, los limites de visibilidad de un observador vienen impuestos por el horizonte, ya que el observador nunca vera más lejos de su horizonte.

La situación de diferenciar los puntos visibles y no visibles, para limitar los segmentos visibles y no de un observador sobre un perfil en la obtención de la Visibilidad Volumétrica, es la misma situación que se da en las Cuencas Visuales para determinar la superficie visible. Es más, los segmentos visibles son los mismos el las Cuencas Visuales que para la Visibilidad Volumétrica. Esto permite reaprovechar todo el cálculo de los segmentos visibles sobre las Cuencas Visuales para el cálculo de la Visibilidad Volumétrica.

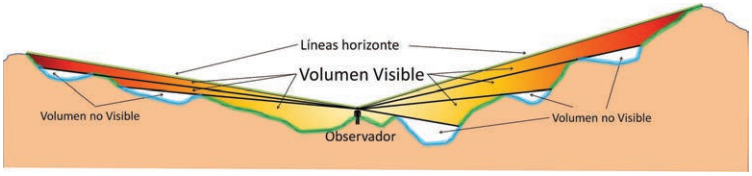


Figura 3.9: Volumen visible y no visible sobre un hipotético perfil.

En la figura 3.9 se puede observar como el volumen visible del observador en una dirección esta compuesto por la suma de cada uno de los volúmenes limitados por cada segmento visible. Es decir cada segmento visible va a limitar un volumen que va desde el observador hasta la superficie que limitada el segmento visible,

siendo el limite en la parte inferior del volumen el plano entre el observador y el comienzo de segmento visible y el limite superior el plano entre el observador y el fin del segmento visible. En el caso del ser el último segmento visible el limite superior lo impone la línea o plano del horizonte, es decir la línea entre el observador y el último punto visible en la dirección de estudio.

Para la obtención del Viewshed 3D, o Visibilidad Volumétrica, se ha empleado una metodología similar a la usada en el cálculo de las Cuencas Visuales. Por un lado se hace el análisis sectorial, del terreno representado por su MDE, para determinar los segmentos visibles que es uno de los primeros pasos para conseguir una buena aproximación de los volúmenes visibles por medio de los anillos sectoriales que conforman los segmentos visibles, ya que tras obtener los segmentos visibles se calcula el volumen visible de cada uno de los segmentos visibles y la suma de todos estos volúmenes es la Visibilidad Volumétrica.

En la figura 3.10 se muestra un ejemplo del volumen visible perteneciente a un segmento visible en una dirección arbitraria para un hipotético observador posicionado sobre la superficie de un terreno representado en la imagen. El segmento visible se encuentra delimitado por sus puntos de comienzo (SRS) y final (ERS) del segmento visible, de igual forma que sucedía en el cálculo de las Cuencas Visuales. El volumen visible correspondiente a dicho segmento visible se encuentra acotado por cinco planos, como se puede ver en la figura 3.10. Los planos son: (1) en la parte superior, por el plano formado por el observador y el final del segmento visible o anillo-sectorial (ERS), (2) y (3) en los laterales, por los planos verticales que delimitan el sector sobre el anillo-sectorial del segmento visible y parten desde el observador, (4) en la parte inferior por el plano que conecta el observador y el comienzo del sector del anillo-sectorial visible o punto de comienzo del mismo (SRS) y (5) en la parte frontal al observador por el plano del anillo-sectorial visible, limitado por el segmento visible con comienzo (SRS) y final (ERS).

El cálculo matemático del volumen visible de un observador correspondiente a un solo segmento visible, es más complejo que el cálculo del área del mismo segmento para la obtención de las Cuencas Visuales, ya que si se recuerda en el caso de las Cuencas Visuales bastaba con calcular el valor de los anillos sectoriales visibles para obtener la solución. Un ejemplo gráfico de como realizar el cálculo del volumen visible de un solo segmento visible se muestra en el de figuras 3.11. La primera de las figuras 3.11a, representa un hipotético segmento visible de un observador, obtenido de uno de los posibles perfiles generados a partir de uno de los sectores de estudio. Si a dicho segmento visible se le desea calcular el volumen visible, uno de los primeros pasos es la proyección 2D del volumen visible correspondiente, sobre el plano vertical que pasa por el vector direccional,

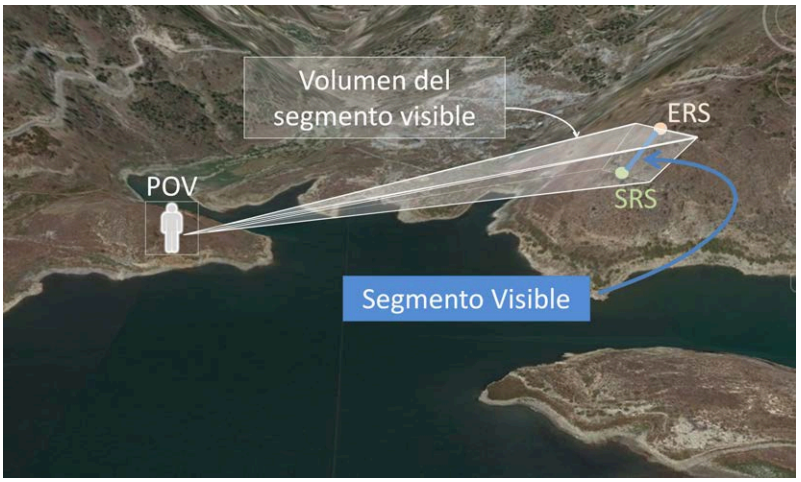
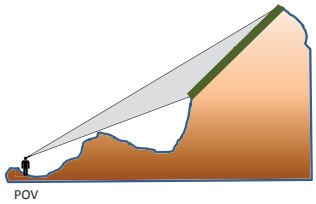


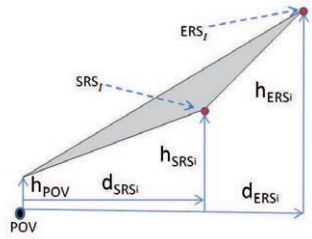
Figura 3.10: Volumen visible en 3D correspondiente a un segmento visible.

del sector de estudio, que podría ser$\vec{S}_i$. La proyección 2D del volumen visible se puede aproximar por un triángulo formado por tres puntos, SRS_i o inicio del segmento visible, ERS_i o final del segmento visible y POV observador, véase figura 3.11b. Si se recuerda, en el caso de las Cuencas Visuales la superficie de un segmento visible se obtiene mediante la rotación del segmento visible a los lados izquierdo y derecho, del vector direccional $\vec{S}_i$, y en torno a un eje vertical del observador POV. De esta forma queda limitado en anillo-sectorial visible correspondiente al segmento visible de estudio, que en el caso del segmento de la figura 3.11a el anillo-sectorial resultante es el que se muestra en la figura 3.11c, con color verde oscuro. De igual forma que la superficie del anillo-sectorial se puede obtener por la rotación del segmento visible, el volumen visible asociado a un segmento visible, se puede aproximar por el volumen construido en la rotación del triángulo visible mostrado en la figura 3.11b. Es decir se gira el triángulo $0,5^o$ de sólido de revolución a la izquierda y lo mismo a la derecha en torno a un eje vertical del observador POV obteniendo el volumen visible, como queda recogido en la figura 3.11d, donde la parte sombreada en verde corresponde al anillo-sectorial y todas las partes sombreadas en gris son los cuatro planos que acotan el volumen visible, junto al plano del anillo-sectorial visible.

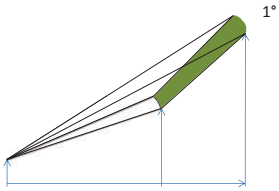
Una vez descrito el volumen de un segmento visible, y como obtenerlo de forma gráfica, lo siguiente es buscar una expresión matemática que permita calcularlo de forma eficiente sobre un equipo informático. Para dicha tarea se ha usado el segundo teorema del centroide de Pappus, que define el volumen de un sólido



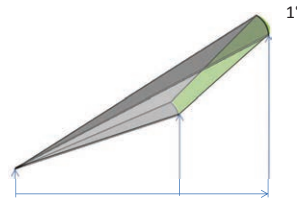
(a) Segmento visible desde un observador (El segmento en línea más gruesa)



(b) Triángulo Visible (El área de color gris)



(c) Anillo Sectorial visible (en área de color oscuro)



(d) Volumen Visible

Figura 3.11: Elementos básicos de la computación del volumen visible de un segmento visible.

de revolución generado por la rotación de la superficie que delimita dicho solido alrededor de un eje vertical sobre el que se rota.

El segundo teorema de Pappus, precisa más el cálculo del Volumen, ya que dice que el volumen V de un sólido de revolución generado por la rotación de una superficie plana A alrededor de un eje vertical externo, es igual al producto del área de la superficie (A), por la distancia que recorre (d_{Ce}) el centroide geométrico de la superficie, al rotar alrededor de un eje de giro. Este teorema se puede extrapolar directamente, al volumen visible de un observador correspondiente a un segmento visible, ya que resulta que el volumen visible del segmento visible es igual al barrido de la rotación sobre el eje del observador del área del triángulo formado entre el observador y el segmento visible. De forma que solo es necesario para la aplicación de este teorema el cálculo de área del triángulo visible y de la distancia que recorre el centroide del mismo.

De las diferentes formas que existe para la obtención del área de un triángulo, hay una que realiza el cálculo del área mediante la información de las coordenadas bidimensionales de los vértices del triángulo. Esta forma dice que si tenemos

un triángulo cuyos vértices son A, B y C, entonces el área del triángulo comprendido entre dichos vértices, es igual a la mitad del producto escalar, en valor absoluto, del vector perpendicular($n_{\vec{AB}}$) al vector $\vec{AB}$ por el vector $\vec{AC}$, quedando matemáticamente la siguiente expresión:

$$Área = \frac{1}{2} \cdot |n_{\vec{AB}} \cdot \vec{AC}| \tag{3.8}$$

En el caso que nos ocupa que es el cálculo del área del triángulo formado por el observador POV, el comienzo del segmento visible SRS_I y el final del segmento visible ERS_I, resulta que cada uno de estos puntos es un vértice del triángulo a tratar (ver figura 3.11b). De forma que si el sistema de coordenadas de referencia se posiciona sobre la posición del observador POV, entonces las coordenadas de cada uno de los vértices del triángulo son POV=(0, h_{POV}), SRS_I=(d_{SRS_i},h_{SRS_i}) y ERS_I=(d_{ERS_i},h_{ERS_i}). Tras operar la formula matemática 3.8 con la información de los vértices del triángulo de un segmento visible se llega a la expresión 3.9, que permite el cálculo directo del área del triángulo de un segmento visible.

$$A = \frac{1}{2}|(d_{ERS_i}(h_{SRS_i} - h_{POV}) - d_{SRS_i}(h_{ERS_i} - h_{POV}))| \tag{3.9}$$

Una vez determinada el área del triángulo visible, el siguiente paso para poder aplicar el segundo teorema de Pappus es calcular la distancia que recorre el centroide del triángulo visible.

Si se llama C a la distancia entre el observador y el centroide, entonces la distancia que recorre el área del solido de revolución correspondiente al triángulo del volumen visible de un segmento visible es $d_C = 2\pi \cdot C/número\ de\ sectores$, que es el recorrido que hace el arco correspondiente al centroide al girar el espacio de un sector sobre el eje del observador.

La distancia al centroide de un triángulo (también llamado baricentro) desde uno de sus vértices, se puede calcular como 2/3 del segmento que une un vértice con el centro del lado opuesto. Por lo tanto la distancia al centroide del triángulo formado por un segmento visible queda como muestra la siguiente expresión:

$$C = \frac{2}{3}\sqrt{(\frac{d_{SRS_i} + d_{ERS_i}}{2})^2 + (\frac{(h_{SRS_i} - h_{POV}) + (h_{ERS_i} - h_{POV})}{2})^2} \tag{3.10}$$

Al ser prácticamente siempre $d_{SRS_i} + d_{ERS_i}$ mayor que $(h_{SRS_i} - h_{POV}) + (h_{ERS_i} - h_{POV})$ la expresión 3.10 se puede aproximar por:

$$C = \frac{2}{3}\sqrt{(\frac{d_{SRS_i} + d_{ERS_i}}{2})^2} = \frac{2}{3}\frac{d_{SRS_i} + d_{ERS_i}}{2} = \frac{1}{3}(d_{SRS_i} + d_{ERS_i}) \quad (3.11)$$

Por tanto, el volumen visible correspondiente a un segmento visible (V_{RS}) basándose en el teorema de Pappus y en el cálculo del centroide de un triángulo queda matemáticamente representado por la siguiente expresión:

$$V_{RS} = d_C \cdot A = \frac{2\pi \cdot C}{ns} \cdot \frac{1}{2}|(d_{ERS_i}(h_{SRS_i} - h_{POV}) - d_{SRS_i}(h_{ERS_i} - h_{POV}))|$$
$$C = (d_{SRSi} + d_{ERSi})/3$$
$$ns = número\ \ de\ \ sectores \quad (3.12)$$

La Visibilidad Volumétrica que corresponde a un observador localizado en un terreno, es la suma de todos los volúmenes visibles correspondientes a todos los segmentos visibles, en todos los sectores en los que se barre el MDE de partida. Por tanto, el volumen visible en cada sector es el sumatorio de los subvolúmenes visibles correspondientes a los segmentos visibles de ese sector. Esta forma de composición de la Visibilidad Volumétrica para un observador, es similar a la del cálculo de las Cuencas Visuales, donde se hacia el sumatorio de los anillos sectoriales visibles. La ecuación matemática discreta para el cálculo de la Visibilidad Volumétrica de un observador en un terreno(VV_{POV}), queda de la siguiente forma:

$$\boldsymbol{VV_{POV}} = \sum_{\boldsymbol{S_i=1}}^{\boldsymbol{S_i=nsec}} \sum_{\boldsymbol{rs_n=1}}^{\boldsymbol{rs_n=e_{rsS_i}}} \boldsymbol{V_{RS}}$$
$$VV_{POV} = Visibilidad\ \ Volumétrica\ \ del\ \ observador\ \ POV$$
$$S_i = Número\ \ del\ \ sector$$
$$nsec = Número\ \ de\ \ sectores\ \ totales$$
$$rs_n = Número\ \ del\ \ segmento\ \ visible\ \ dentro\ \ del\ \ sector$$
$$e_{rsS_i} = Número\ \ del\ \ último\ \ segmento\ \ visible\ \ dentro\ \ del\ \ sector$$
$$V_{RS} = Volumen\ \ de\ \ cada\ \ segmento\ \ visible \quad (3.13)$$

3.2.1. Implementación paralela de la Visibilidad Volumétrica Total para un MDE de tamaño medio

A la hora de implementar el cálculo de la Visibilidad Volumétrica sobre el algoritmo desarrollado, solo hay que modificar la función *ComputarVisibilidad()*. El resto de la implementación de algoritmo queda igual que la versión paralela para el cálculo de cualquier parámetro de visibilidad sobre un MDE de tamaño medio. En el pseudocódigo 3.2 se muestra en color rojo, cuales son las modificadas realizadas, en la función *ComputarVisibilidad()*, para la inclusión del segundo teorema de Pappus en la computación de la Visibilidad Volumétrica y de esta forma tras realizar los dos bucles externos del algoritmo desarrollado, se obtiene la Visibilidad Volumétrica para todos los puntos del MDE a estudiar.

Pseudocódigo 3.2: Kernel de Visibilidad Volumétrica.

```
Kernel(d_POV,h_POV,d_P,h_P,&maxanguloanterior,&visible)
  float anguloequivalente = (h_P-h_POV)/(d_P-d_POV);
  bool es_visible = anguloequivalente >= maxanguloanterior
  bool empieza_segmento = es_visible && !visible_segmento
  bool fin_segmento = !es_visible && visible_segmento

  if empieza_segmento then
     //Incremento del número de anillo-sectorial visibles
     Incremento(n_anillo_sectorial)
     //Almacenamiento del comienzo del segmento visible en un array
     Almacena(d_P);
     //Almacena la distancia de comienzo del anillo-sectorial visible
     d_SRS = d_P-d_POV;
   //Altura del primer punto del segmento visible respecto al POV
     h_SRS = h_P;
  endif

  if fin_segmento then
     //Almacenamiento del final del segmento visible en un array
     Almacena(d_P);
     //Almacena la distancia del fin del anillo-sectorial visible
     d_ERS = d_P-d_POV;
     //Altura del último punto del segmento visible respecto al POV
     h_ERS = h_P;
     //Volumen Visible en un segmento visible
   V_POV += (pi/nsec) * (d_ERS + d_SRS)/3) * |d_ERS*h_SRS - d_SRS*h_ERS|;

  endif

   //Guarda le estado visible del punto, para la siguiente iteración
   visible_segmento=es_visible;/
   //Actualización del máximo ángulo-equivalente para próxima iteración
   maxanguloanterior = max(anguloequivalente,maxanguloanterior)
end Kernel
```

3.2.2. Resultados Visibilidad Volumétrica Total para un MDE de tamaño medio

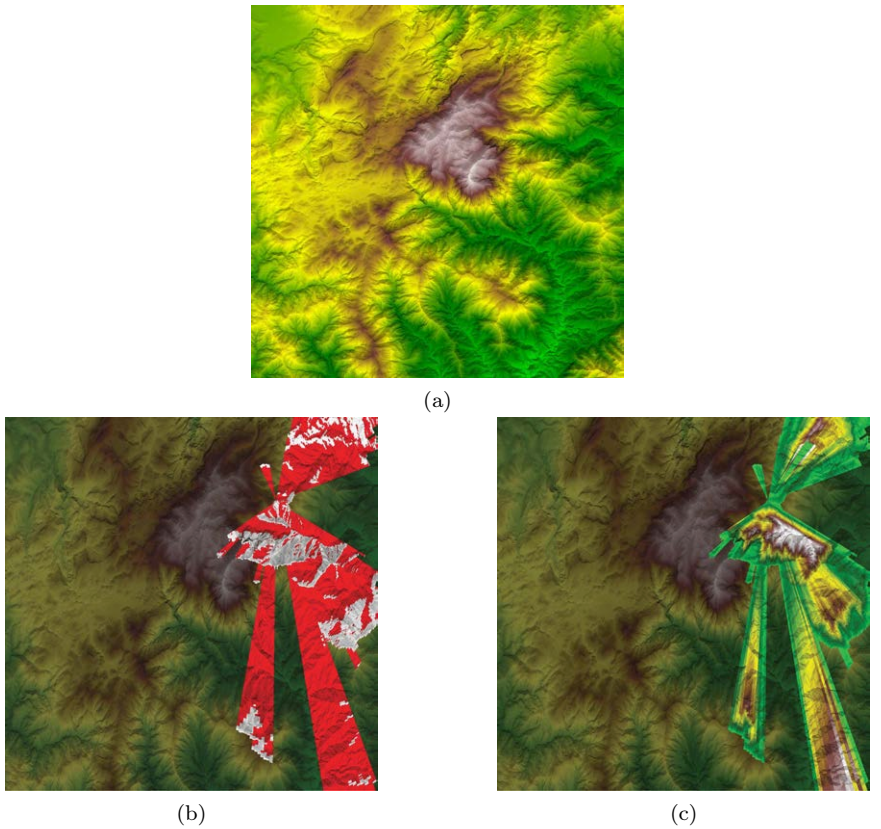
El cálculo de la Visibilidad Volumétrica Total se ha realizado tanto en ejecución paralela como en secuencial, obteniendo los mismos resultados en ambos casos, por lo que una vez comprobado este hecho, se han realizado todos los experimentos en ejecución paralela, por proporcionar esta los resultados en menor tiempo.

No se han podido comparar los resultados obtenidos para la Visibilidad Volumétrica Total con otros autores, ya que el presente trabajo es el primero que estudia este parámetro. Lo cual lo hace pionero, y aporta soluciones para varias aplicaciones, de las cuales ya se hablo en el capítulo de introducción de la memoria de la tesis.

Los resultados del cálculo de la Visibilidad Volumétrica Total se almacena simplemente en un fichero binario con extensión $.flt$, al igual que se realiza con el cálculo de las Cuencas Visuales Totales. Este almacenamiento permite análisis posteriores de los resultados. En el caso de Visibilidad Volumétrica la unidad empleada es de Hm^3, mientras que el caso de las Cuencas Visuales son Km^2. Con el archivo de resultados $.flt$ también se crean imágenes de mapas de Visibilidad Volumétrica Total usando la metodología que se describe en el apéndice B; solo que para la obtención de las imágenes de la Visibilidad Volumétrica es necesario un cambio en la escala de colores diferenciada de las Cuencas Visuales Totales.

En la figura 3.12a se presenta un MDE de tamaño medio sobre una zona con relieve altamente rugoso de la provincia de Málaga, conocida como Sierra de las Nieves, con coordenadas UTM zona 30S, N=4070000 y E=310000 para el primer punto superior izquierdo. A partir de ahora cuando volvamos a usar este MDE lo llamaremos por MDE de la Sierra de las Nieves. De este MDE se ha seleccionado un punto como observador y se le ha calculado las Cuencas Visuales y la Visibilidad Volumétrica obteniendo como resultados las imágenes de las figuras 3.12b y 3.12c respectivamente. En cuanto la imagen de Cuencas Visuales la zona gris muestra las áreas visibles por el observador mientras que las rojas las no visibles que están dentro de la zona visible. Por otro lado, la imagen que muestra la Visibilidad Volumétrica para el mismo observador se distingue una zona con pixel en color más blancos y verdes para indicar las columnas de aire visibles de los volúmenes visibles representados en 2D. Tras comparar las imágenes resultantes de calcular la Visibilidad Volumétrica y la Cuenca Visual para un punto, se ve claramente que ambas cubren la misma zona, ya que la Visibilidad Volumétrica cubre claramente la misma zona que la Cuenca Visual

zona roja más la zona no visible dentro de la Cuenca Visual en rojo.

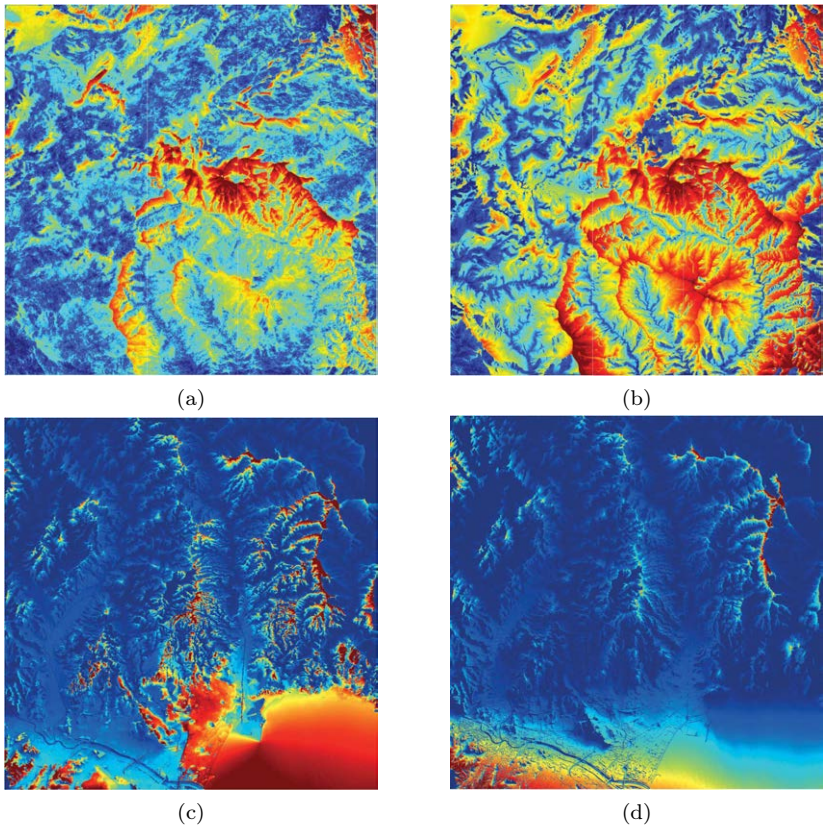


(a) (b) (c)

Figura 3.12: (a) MDE de una zona montañosa de la provincia de Málaga, España, con coordenadas UTM zona 30S, N=4070000 y E=310000 (b) Cuencas Visuales de un punto con coordenadas UTM zona 30S X=4063890N y Y=324530E (c) Visibilidad Volumétrica del mismo punto que para las Cuencas Visuales.

Para mostrar el potencial de la Visibilidad Volumétrica se ha realizado estudios sobre distintos MDEs de tamaño medio a las cuales se le ha calculado las Cuencas Visuales Totales y la Visibilidad Volumétrica Total. Ejemplo de este estudio se presenta en la figura 3.13 donde se pueden observar los resultados obtenidos en imágenes para dos MDEs de partida. Un MDE corresponde a de la Sierra de las Nieves, del cual se ha obtenido las imágenes 3.13a y 3.13b, siendo la primera imagen las Cuencas Visuales Totales y la segunda imagen la Visibilidad Volumétrica Total. El otro MDE de partida corresponde al de la Ciudad de

Málaga y se obtienen la imagen 3.13c para las Cuencas Visuales Totales y la imagen 3.13d para las Visibilidad Volumétrica total. De estas comparaciones se ve perfectamente que no siempre los puntos con máximas Cuencas Visuales presentan las máximas Visibilidades Volumétricas.



(a) (b)

(c) (d)

Figura 3.13: Mapas de Cuencas Visuales Totales (a) y (c). Mapas de Visibilidad Volumétrica Totales (b) y (d) dibujados en escala logarítmica. Para los MDE de área montañosa de las Sierra de las Nieves (a) y (b), y el área de la ciudad de Málaga(c) y (d).

A la hora de ver con más claridad las posibles diferencias entre la Visibilidad Volumétrica y las Cuencas Visuales se puede estudiar como son las distribuciones de los valores de cada uno de los parámetros respecto al valor máximo que alcanza cada uno do los parámetros sobre en análisis del mismo MDE. Nosotros hemos realizado este estudio sobre el MDE de tamaño medio de la ciudad de Málaga y

el resultado obtenido ha sido el que se muestra en la figura 3.14. En esta figura el eje X muestra los intervalos en tanto por ciento respecto del valor máximo del parámetro representado; por ejemplo, el intervalo $5\,\% - 10\,\%$ del eje X representa los valores de las Cuencas Visuales o Visibilidad Volumétrica que son mayores del 5 % del valor máximo y menores del 10 % del valor máximo. Por otra parte el eje Y indica la proporción de puntos totales cuyo valor pertenece a un especifico intervalo. En en la figura 3.14 se puede observar como alrededor del 60 % de los puntos del MDE presenta una Visibilidad Volumétrica con valores inferiores al 5 % del valor máximo de este parámetro, mientras que el porcentaje de puntos con Cuencas Visuales que su valor es inferior al 5 % del valor máximo de Cuencas Visuales es casi nulo. La conclusión más clara que se extrae de la diferencia de distribuciones de la figura 3.14 es que para el MDE de la ciudad de Málaga hay muchos más puntos con baja Visibilidad Volumétrica que para las Cuencas Visuales. Si esta información la relacionamos con las imágenes de los mapas de Cuencas Visuales Totales (figura 3.13c) y Visibilidad Volumétrica Total(3.13d) se ve claramente que los puntos pertenecientes a la superficie del mar presentan una baja Visibilidad Volumétrica mientras que un alto valor de Cuencas Visuales. Por otro lado, se observa como más del 80 % de los puntos del MDE de la ciudad de Málaga poseen menos del 25 % del valor máximo de los parámetros de visibilidad calculados.

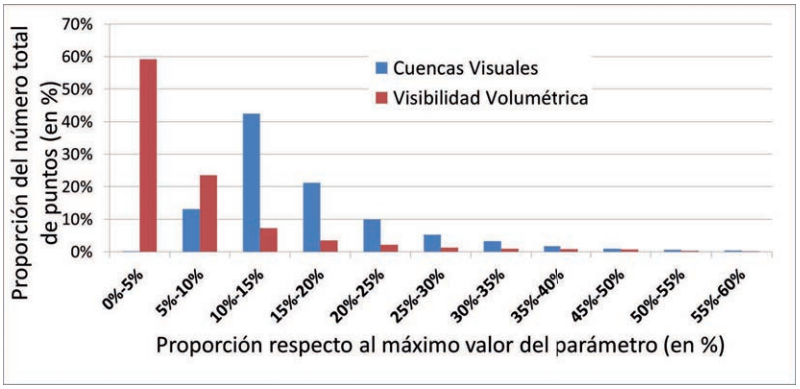


Figura 3.14: Distribución del valor de los puntos de las Cuencas Visuales y la Visibilidad Volumétrica respecto al valor máximo alcanzado para cada parámetro en la computación del MDE de la ciudad de Málaga.

3.2.3. Implementación paralela de la Visibilidad Volumétrica Total sobre un MDE de tamaño grande y resultados

Tras haber obtenido por un lado las Cuencas Visuales Totales para un MDE de tamaño grande como es el mapa completo de la comunidad autónoma de Andalucía con 1500 millones de puntos y por otro lado, haber calculado por primera vez un nuevo parámetro de visibilidad que se ha llamado Visibilidad Volumétrica en el presente trabajo, se ha considerado necesario la computación de este nuevo parámetro sobre el MDE de tamaño grande que abarca toda la superficie de la comunidad autónoma de Andalucía. De esta forma se puede hacer una comparación entre las Cuencas Visuales Totales y la Visibilidad Volumétrica Total de la comunidad de Andalucía y por otro lado, encontrar cuales son las zonas más interesantes para distintas aplicaciones según los parámetros calculados.

La implementación para el cálculo de la Visibilidad Volumétrica Total sobre MDE de tamaño grande consiste en usar el algoritmo desarrollado para el cálculo de las Cuencas Visuales Totales para un MDE de tamaño grande (mostrado en la sección 3.1.6) y cambiar la función *ComputarVisibilidad()* con las líneas de código que permiten la computación de la Visibilidad Volumétrica (mostrado en el pseudocódigo 3.2 de la sección 3.2.1). El resto del proceso para la computación de la Visibilidad Volumétrica Total es el mismo que para las Cuencas Visuales Totales. Es decir, se deben realizar los 4 pasos para la computación de un parámetro de visibilidad sobre un MDE de tamaño grande y adaptarlos al cálculo de la Visibilidad Volumétrica Total que se computa sobre el supercomputador Piccaso, por la necesidad de recursos necesarios. Los 4 paso quedan de la siguiente forma:

1. Paso 1. División del MDE de tamaño grande en 1500 MDEs de tamaño medio, desplazados.

2. Paso 2. La computación paralela, usando el algoritmo desarrollado con limitación de sectores de estudio de MDEs de tamaño medio en 4 nodos HP DL980G7, de forma que cada nodo computa 375 MDEs de tamaño medio.

3. Paso 3. Suma de los resultados de los 1500 MDEs de tamaño medio parcialmente solapados y con computación parcial de sectores, en 375 MDEs de tamaño medio que no se solapan y que tienen la información de la visibilidad para los todos puntos en todos los sectores.

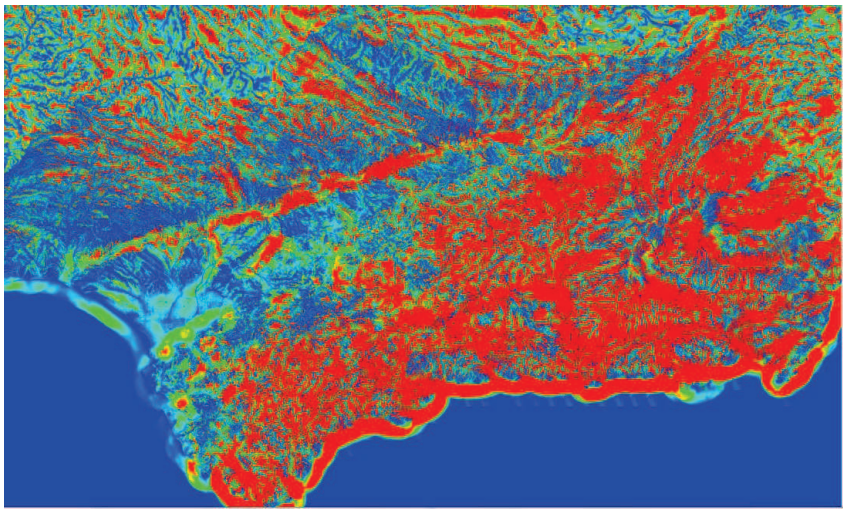
4. Paso 4. Obtención de las imágenes de Visibilidad Volumétrica Totales del MDE de tamaño grande de Andalucía, para el cual se modifica la paleta de colores, ya que si no se puede presentar los parámetros de visibilidad de forma correcta.

Tras la implementación y computación se ha obtenido el primer mapa, jamás creado, de Visibilidad Volumétrica Total del MDE que abarca la comunidad autónoma de Andalucía y que posee 1500 millones de puntos con una resolución de $10 \times 10\ m^2$. El resultado es el que se presenta la figura 3.15a. Si se compra el mapa de Visibilidad Volumétrica Total con respecto el de Cuencas Visuales Totales (ver figura 3.15b) se ve claramente las diferencias existentes. Los puntos con mayores valores Cuencas Visuales son los puntos más altos y los puntos cercanos al litoral costero. Mientras que los puntos con mayores valores de Visibilidad Volumétrica son los pertenecientes al litoral costero y los localizados en valles rodeados por montañas, como sucede con casi toda la cuenca del rio Guadalquivir.

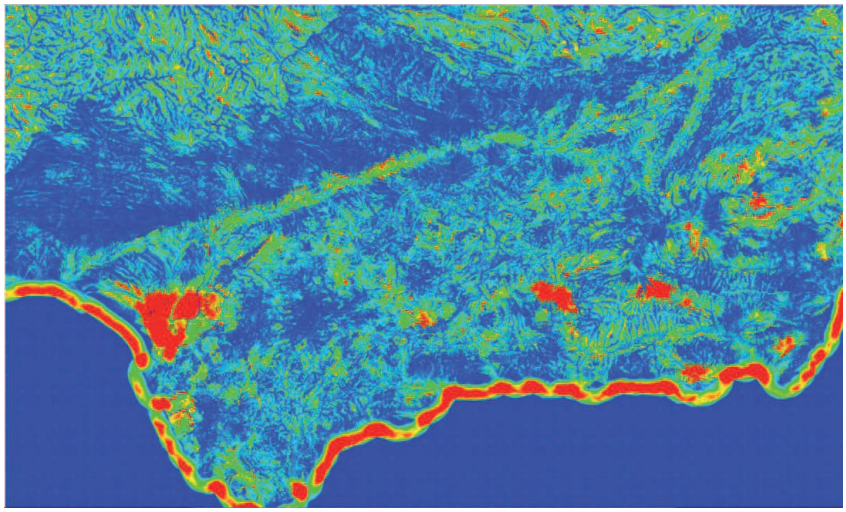
3.3. Posicionamiento del mínimo número de observadores que cubren la máxima superficie

La obtención de un número mínimo de observadores que garantiza la máxima cobertura visual sobre un área representada por un MDE de alturas, tiene gran interés en muchas aplicaciones o campos, como pueden ser en telecomunicaciones y en planificación ambiental, entre otros. Sin embargo, tras realizar una revisión bibliográfica no hemos encontrado herramientas o algoritmos que permitan determinar de forma rigurosa el número mínimo de observadores y sus posiciones, para que estos proporcionen la máxima cobertura. Uno de los problemas con los que se han encontrado los diseñadores de algoritmos que han intentado resolver este problema, es la complejidad de los mismos por el enorme volumen de datos a procesar, lo cual hace que sea un problema *Big data*, el cual está demostrado que se trata de un problema con complejidad NP [28].

Lo que si se ha encontrado en la bibliografía son métodos heurísticos que buscan el conjunto óptimo de localizaciones que permite alcanzar la mayor área posible con un número de observadores previamente establecido, este tipo de soluciones lo han llamado problemas de emplazamiento de múltiples-observadores. Dentro de este tipo de problemas se encuentran diferentes soluciones: i) métodos de búsqueda locales que reducen el área alrededor de los



(a)



(b)

Figura 3.15: Mapa de Visibilidad Volumétrica Total de Andalucía (a) y mapa de Cuencas Visuales Totales de Andalucía (b), ambos con radio máximo de 10 km y con una resolución de $10x10m^2$.

observadores en pequeños radios según las capacidades computacionales con las que se cuenta [15] ii) métodos de búsqueda *gridy* que inicialmente seleccionan un pequeño grupo de candidatos y sucesivamente incorporar nuevos puntos a ese conjunto hasta que cubran todo el área de interés [10] y iii) algoritmos genéticos [27]. Además, hay trabajos que utilizan estrategias de búsqueda aleatorias o exhaustivas [1].

Nuestro algoritmo, en comparación con todos los trabajos relacionados, se basa en la consideración de que todos los puntos pertenecientes al MDE que cubre una zona de interés son observadores potenciales. Por lo que se realiza una selección exhaustiva sobre todos los puntos del MDE de partida, para alcanzar un conjunto de observadores con las mayores coberturas posibles sin prefijar ningún conjunto de observadores previo. Por tanto, el conjunto de observadores resultante que proporciona nuestro algoritmo está compuesto por el mínimo número de observadores que garantizan la máxima cobertura. Las soluciones alcanzadas bajo nuestro método pueden usarse para mejorar la vigilancia en la lucha contra el fuego, en sistemas antirrobo, control ambiental, cálculo de ruta ocluida, cobertura de telefonía móvil y comunicaciones de radio o televisión, entre otros ejemplos.

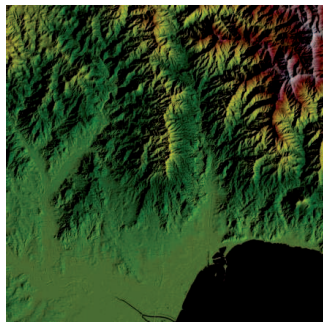
El algoritmo desarrollado para buscar el mínimo número de observadores que proporcionan la máxima cobertura, se basa en el cálculo de las Cuencas Visuales Totales, que se ha presentado en la sección 3.1 del presente capítulo, que permite extraer cuales son los puntos que poseen máxima visibilidad dentro de un MDE, siendo estos puntos los candidatos a observadores de máxima cobertura. En si el algoritmo desarrollado es un algoritmo iterativo que tras el cálculo de las Cuencas Visuales selecciona un observador con máxima cobertura y elimina la superficie cubierta para las próximas iteraciones.

La clave del algoritmo desarrollado es el uso de una máscara de puntos que permite diferenciar entre los puntos del MDE que no se tienen en cuenta en el cálculo de las Cuencas Visuales Totales de los que si. Los puntos que no se desean tener en cuenta en el cálculo de las Cuencas Visuales Totales por medio de la máscara atienden a dos motivos. El primero es porque puede ser necesario realizar un estudio selectivo del MDE de partida, donde se descarten zonas del MDE a las que no se desea dar cobertura por cualquier tipo de razón, como pueden ser zonas que exceden limites urbanos, o que exceden limites territoriales, limites de zonas de reserva, etc. Y el segundo motivo es porque la superficie ya esta siendo cubierta por otro observador y no es necesario que dos observadores o más tengan cobertura de los mismos puntos.

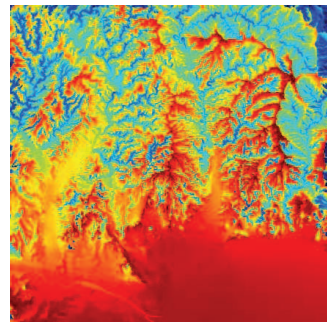
3.3.1. Máscara para MDE en el cálculo de las Cuencas Visuales

Un mapa de Cuencas Visuales Totales indica el número de puntos visibles por cada uno de los puntos de un terreno representado en un MDE de alturas de partida. Esto puede ser útil en muchos campos, sin embargo en muchos otros casos, hay parte del terreno el cual o bien tiene poco o ningún interés para los observadores. En el caso de la planificación del posicionamiento de celdas de telefonía móvil, por ejemplo, hay zonas sin población donde a una operadora puede no interesar prestar una excelente cobertura, si el sobre coste es muy alto para el beneficio que puede lograr. Esta es la típica situación de algunos zonas como el desierto de Tabernas en Almería, España. Donde hay espacios en dicho desierto que no hay cobertura y es el usuario es el que se mueve a espacios donde si la hay (normalmente partes altas de colinas). Otro ejemplo, donde no interesa tener cobertura de ciertas zonas, es en la vigilancia contra incendios, ya que, por ejemplo, no es necesario tener una cobertura visual del mar o de lagos, ya que sobre este tipo de superficie no se producirán incendios forestales. La no necesidad de estudiar todo el terreno para ciertas aplicaciones, es un hecho que también cubre el enmascaramiento de los MDE, ya que permite la discriminación de zonas de interés frente a otras que no lo son. En la figura 3.16a se presenta el MDE de la ciudad de Málaga y en la figura 3.16b se muestra una máscara, llamada MASCARA_0, que cubre un área de interés del MDE de la ciudad de Málaga sobre la cual, por ejemplo, se quiere dar cobertura visual mediante torres de vigilancia para el control de incendios. En la figura 3.16b se muestra el mapa de las Cuencas Visuales Totales para el MDE de partida, mientras que en la figura 3.16d se muestra el mapa de como quedan las Cuencas Visuales Totales para la zona de interés o enmascarada bajo el algoritmo del presente apartado.

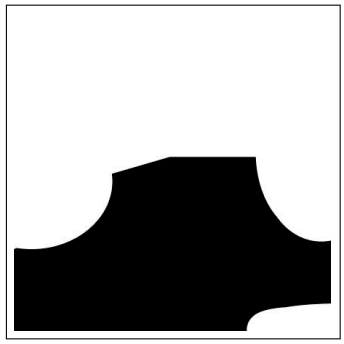
De la figura 3.16d, mapa de Cuencas Visuales Totales Enmascaradas, se puede extraer dos observaciones. La primera observación es que tras compararla con el mapa de Cuencas Visuales Totales normales (figura 3.16b), se aprecia claramente la diferencia existente entre ambos. Donde mayor diferencia se da es en los puntos con menor altitud no pertenecientes al área de interés, ya que, por ejemplo, los puntos de los barrancos exteriores a la zona de interés no tienen prácticamente visibilidad de puntos de la zona de interés. Y por otro lado, la segunda observación es que el algoritmo utilizado calcula las Cuencas Visuales para todos los puntos del MDE de partida, pertenezcan o no a la zona de interés, aunque solo se computa la visibilidad sobre la superficie de la zona interés. Es decir, cada punto del MDE es considerado como observador al cual se limita el estudio de sus Cuencas Visuales a la zona enmascarada. Esta acción de calcular las Cuencas Visuales a todos los



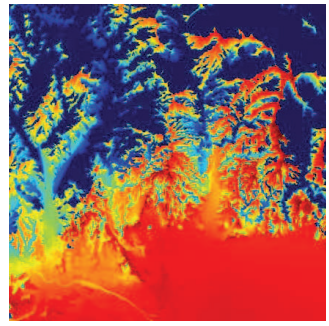
(a) MDE de la ciudad de Málaga



(b) Mapa Cuencas Visuales Totales



(c) Mascara (MASCARA_0) para un área de interés que cubre sobre el del MDE de origen



(d) Mapa de Cuencas Visuales Totales Enmascarado (por MASCARA_0)

Figura 3.16

puntos del MDE de partida, permite no descartar puntos que pueden presentar una mayor cobertura sobre la zona de interés aunque no estén dentro de la misma. Esta consideración permite no descartar observadores con mayores coberturas que se puedan ahorrar costes de instalación y mantenimiento que si solo se tienen en cuenta posibles observadores dentro de la zona de interés.

La máscara propuesta consiste en una máscara de booleanos que cubre el MDE e indica que puntos se consideran visibles y cuales no para el cálculo de las Cuencas Visuales Totales, dando lugar a una nueva cartografía llamada MDEs Enmascarados. Al aplicar el algoritmo de las Cuencas Visuales Totales sobre estos nuevos MDEs Enmascarados, se obtienen las Cuencas Visuales Totales Enmascaradas.

Respecto la máscara, aunque se ha usado en el presente trabajo una máscara de tipo booleano, esta puede ser transformada o reemplazada por una de enteros de tamaño byte. Si la máscara es de tipo byte permite dar más valores que solo el *verdadero* o *falso* del tipo booleano, haciendo que se puedan presentar más estados de los puntos de un MDE para el cálculo de los parámetros enmascarados y haciendo que según el valor del byte pueda representarse distinta información. De hecho, un valor negativo para de la máscara puede indicar que el punto tiene cierto valor significativo para el observador. De esta manera se puede hacer una segunda discriminación, donde se puede resaltar una zona con un interés mayor dentro de la propia zona de interés de la máscara normal y así calcular un mapa de Cuencas Visuales Enmascarado y ponderado a las necesidades del problema.

3.3.2. Algoritmo desarrollado

La principal característica del algoritmo desarrollado para el posicionamiento optimo del del minio número de observadores es usar una metodología iterativa con un coste computacional bajo, en el desarrollo del mismo, gracias al algoritmo de Cuencas Visuales Totales de partida.

En la primera iteración del algoritmo se calcula el mapa de Cuencas Visuales Totales Enmascarado para el MDE de partida y la primera máscara, llamada Máscara_0. A continuación de este primer mapa de Cuencas Visuales Totales Enmascarado se selecciona, normalmente, el punto o observador con mayor valor de Cuencas Visuales, ya que este punto es el observador que más cobertura abarca. Hemos dicho que este punto es el que normalmente se selecciona porque puede haber limitaciones por las que no se pueda seleccionar ese punto (dificultad de acceso por equipos de instalación, imposibilidad de instalación de torres por requisitos de protección de zona, encontrarse la localización en otro termino municipal, etc...) y sea necesario seleccionar el siguiente punto o cualquier otro punto con máximas Cuencas Visuales, que cumpla con los requisitos de posicionamiento. Tras la selección del primer observador, si la cobertura de este no alcanza toda la zona deseada, se pasa a la generación de una nueva máscara. Esta nueva máscara parte de la máscara original, a la cual se le extrae el área que cubre el observador seleccionado. De forma que la nueva máscara, llamada Máscara_1, posee la información de que puntos se quieren cubrir con el siguiente observador que se pasa a buscar tras la realización de una nueva iteración del algoritmo, donde se repite el proceso de cálculo de Cuencas Visuales Totales Enmascaradas, para volver a seleccionar un nuevo candidato a observador de máxima cobertura. El proceso se repite tantas veces como sea necesario, asta alcanzar la cobertura deseada o se alcance un número de observadores prefijado,

que no se desea superar.

En el pseudocódigo 3.3 tenemos un resumen del algoritmo desarrollado, donde se parte de una máscara (MASCARA_0) representativa de una zona de interés y mientras no se alcanza algunas de las condiciones, de alcance de la cobertura deseada o el máximo número de observadores (max_n_obs) dispuestos a posicionar, se va realizando iteraciones del algoritmo. En el pseudocódigo 3.3 lo primero que se realiza dentro del bucle iterativo (*while*) es obtener las Cuencas Visuales Enmascaradas (MAP_i), para a continuación buscar el punto con máxima cobertura (OBS_i), que sera el observador seleccionado en la presente iteración. Tras la selección del observador se analiza cuales son sus Cuencas Visuales y de esta forma comprobar si el observador cubre el área deseada o sería necesaria otra iteración. En el caso de no cubrir el área deseada se resta el área que cubre el observador a la máscara anterior, generando una nueva máscara y se vuelve a iterar el algoritmo si no se ha alcanzado en número máximo de observadores.

Pseudocódigo 3.3: Algoritmo de posicionamiento de observadores.

```
input mask MASCARA_0
while(condición) do//Si i_obs<=max_n_obs or cobertura<coberturaDeseada
  i++;
  MAP_i= computarCuencasVisualesEnmascaradas(MDE, MASK_i-1);
  OBS_i= max(MAP_i);
 CuencasVisualesOBS_i= CuencasVisualesTotales(OBS_i, MDE);
 if (!cobertura) then
   MASK_i= MASK_i-1 - CuencasVisualesOBS_i;
 endif
endwhile
```

3.3.3. Implementación del algoritmo para el posicionamiento óptimo de observadores

Lo que hemos hecho para implementar el cálculo de las Cuencas Visuales Totales Enmascaradas sobre nuestro algoritmo que calcula las Cuencas Visuales Totales, es que dentro de la función *Kernel()* del algoritmo de partida para las Cuencas Visuales, se ha añadido una nueva línea casi final de la función, como se puede ver en el pseudocódigo 3.4, donde aparece una línea en de código en rojo correspondiente a la modificación realizada. En esta nueva línea ($if(!Puntoenmascarado(p)\&\&visible)CV_{POV} -= superficie(p)$) se comprueba que si el punto es visible y no pertenece a la máscara de la zona de interés. Si el punto es visible pero no pertenece a la máscara se realiza la resta del área o superficie equivalente correspondiente a dicho punto sobre las Cuencas Visuales.

Con esta sencilla operación el cálculo Cuencas Visuales de un observador no tiene en cuenta las superficie de la zona exterior a la máscara, ya que esa área se ha restado previamente .

Pseudocódigo 3.4: Kernel de Cuencas Visuales Enmascaradas.

```
Kernel(d_POV,h_POV,d_P,h_P,&maxanguloanterior,&visible,P)
  float anguloequivalente = (h_P-h_POV)/(d_P-d_POV);
  bool es_visible = anguloequivalente >= maxanguloanterior
  bool empieza_segmento = es_visible && !visible_segmento
  bool fin_segmento = !es_visible && visible_segmento

  if empieza_segmento then
     //Incremento del número de anillo-sectorial visibles
     Incremento(n_anillo_sectorial)
     //Almacenamiento del comienzo del segmento visible en un array
     Almacena(d_P);
   //Almacena la distancia de comienzo del anillo-sectorial visible
     d_SRS = d_P-d_POV;
  endif

  if fin_segmento then
     //Almacenamiento del final del segmento visible en un array
     Almacena(d_P);
     //Almacena la distancia del fin del anillo-sectorial visible
     d_ERS = d_P-d_POV;
     //Cálculo y incremento de la Cuenca Visual del  observador POV
     CV_POV += (d_ERS^2 - d_SRS^2) * (pi/nsec);

  endif

  /*Comprueba que el punto P no pertenece a la marcara y si es visible
      para restar su superficie*/
   if(!Puntoenmascarado(p) && visible) CV_POV -= superficie(p)

   //Guarda le estado visible del punto, para la siguiente iteración
   visible_segmento=es_visible;/
  //Actualización del máximo ángulo-equivalente para próxima iteración
   maxanguloanterior = max(anguloequivalente,maxanguloanterior)
end Kernel


}
```

La función llamada $superficie()$ que representa la operación de cálculo de la superficie correspondiente al punto P consiste en la ejecución de la ecuación 3.15. Esta ecuación lo que calcula en realidad es la superficie equivalente del anillo-sectorial entre dos puntos consecutivos, ya que en el análisis sectorial las superficie equivalente de un punto sobre la línea central del sector es este; donde d_P es la distancia entre el punto del perfil y el observador.

$$superficie(p) = (d_p^2 - d_{p-1}^2) \cdot (pi/360) \tag{3.14}$$

$$\tag{3.15}$$

Una nota sobre la inclusión de la línea de código que comprueba el punto y realiza la resta de la superficie en la función $Kernel()$ es que esta incrementa como es normal el coste computacional del algoritmo. Pero resulta interesante que ante la existencia de otra operación (división), que en cada llamada a la función $Kernel()$ se realiza y presenta un coste computacional mucho más alto en ciclos de CPU, hace que en la ejecución en paralelo junto con las nuevas operaciones, las cuales se realizan en otras unidades funcionales de la CPU y en ejecución fuera de orden en el core de la microarquitectura empleada en nuestro experimento. Todos los experimentos hechos han dado que no se realiza ningún incremento de tiempo adicional significativo, ya que en media dan el mismo tiempo.

Otro punto importante de la implementación del algoritmo es la obtención de las nuevas máscara que tienen la información de la zona de interés a cubrir más las coberturas de los observadores seleccionados. Esta tarea no es compleja de realizar gracias a que en la ejecución de las Cuencas Visuales Totales a la vez que se calcula el valor de las Cuencas Visuales se guarda la información de los anillos sectoriales visibles por medio de la función *almacena(posición)* que se localiza dentro de la función *Kernel()*, como se puede ver en el ejemplo del pseudocódigo 3.4. En concreto dentro de esta función *almacena(posición)* para cada observador (POV) almacena las posiciones de los comienzos y finales de los anillos sectoriales que contienen la zona visible. De forma que para generar una nueva máscara con la información de la zona de interés más la cobertura de un observador ya obtenido, basta con recorrer los puntos del MDE de partida y comprobar si ese punto pertenece a la zona de interés y si no ha sido visible por el observador seleccionado para que el punto pertenezca a la nueva mascara, con las que seguir haciendo las iteraciones del algoritmo de posicionamiento de múltiples observadores.

3.3.4. Resultados

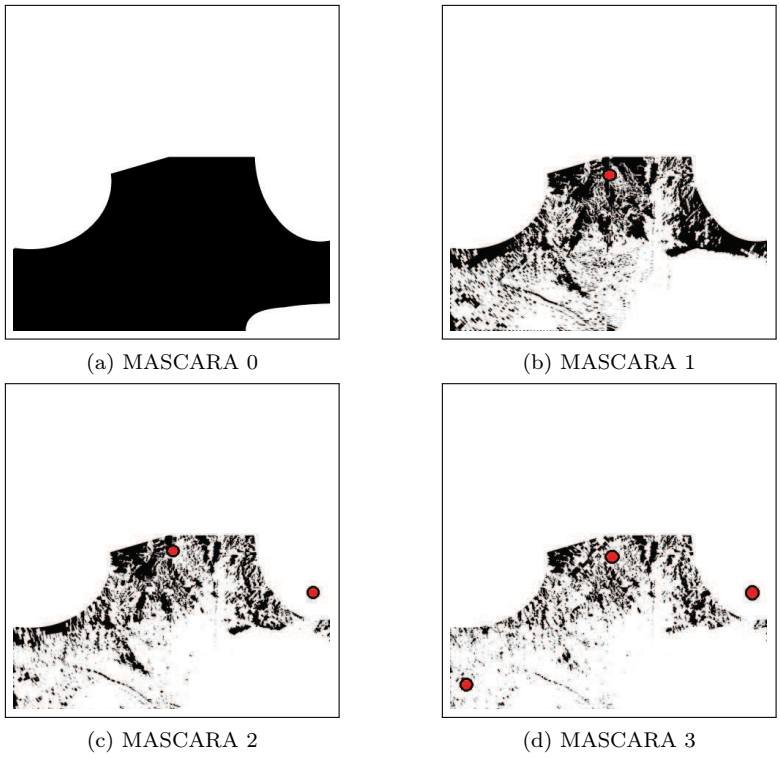
En la evaluación de los resultados hemos partido del MDE de tamaño medio de la ciudad de Málaga. Este MDE presenta tres tipos de terreno bien definidos que son mar, montaña y una parte plana correspondiente a un valle. Sobre este MDE, se quiere dar cobertura visual a la zona interés correspondiente a la máscara 3.17a, que es la zona donde se concentra la mayoría de la población.

Para buscar las posiciones de los observadores que cubren el mayor área de la zona de interés, se ha considerado realizar la computación de las Cuencas Visuales Totales suponiendo que los observadores está situados a 10 metros sobre el suelo, como si se colocasen torres de vigilancia con dicha altura. En la figura 3.18a se muestra el mapa de Cuencas Visuales Totales Enmascarado con observadores situados a 10 metros sobre el suelo.

En un primer experimento se ha limitado el número de torres de vigilancia a tres. Resultando de cada iteración un mapa de Cuencas Visuales Totales Enmascarado, una posición para una torre y una máscara nueva de cobertura para la siguiente iteración. Las máscaras resultantes de cada iteración son las presentadas en las figuras 3.17b, 3.17c y 3.17d para las iteraciones 1, 2 y 3. Al igual sucede con los mapas de Cuencas Visuales Totales Enmascarados que son los mostrados en las figuras 3.18b, 3.18c y 3.18d. Dentro de las figuras de las máscaras se distinguen unos puntos (en color rojo). Estos puntos es donde están posicionadas las torres. La primera torre que se puede aprecias sobre la figura 3.17b, esta dentro de la zona de interés, mientras que la segunda torre que se puede apreciar en la figura 3.17c se localiza fuera de la zona de interés. Por otro lado, estas posiciones también se han añadido a los mapas de Cuencas Totales Enmascarados de las figuras 3.18b, 3.18c y 3.18d. presentándose mediante una estrella en color rosa.

Tras la ejecución de algoritmo que proporciona las posiciones optimas que dan máxima cobertura de la zona de interés seleccionada, nosotros hemos obtenido los resultados de la tabla 3.3. Donde en la columna 1 se indica el número de la máscara y el número de puntos que cubre la mascara respecto del MDE original. La columna 2, indica que porcentaje del MDE esta ocupado por la mascara. La columna 3, indica las coordenadas donde se posiciona la torre o observador que dan la máxima cobertura para el área libre de la máscara. La columna 4, indica el número de puntos que cubre cada una de las torres dentro de la zona de interés o máscara de partida. Y la columna 5 indica el porcentaje de superficie de que cubre cada torre junto con las anteriores.

En la tabla 3.3, la primera columna corresponde a la búsqueda de la posición del observador con máxima cobertura cuando no hay ninguna máscara. Para esta tarea se ha calculado previamente el mapa de Cuencas Visuales Totales, resultando que la posición del observador con mayor visibilidad es 958/1028. Como estamos en un MDE de 2000×2000 y 10 metros de resolución, resulta que el observador esta a 9580 metros hacia al Sur y 10280 metros hacia el Este respecto al primer punto superior izquierdo del MDE. La segunda fila de la primera columna de la tabla es correspondiente a la máscara 0, la cual se indica por el número 0 y con el número de puntos que no se desea estudiar en las próximas iteraciones.



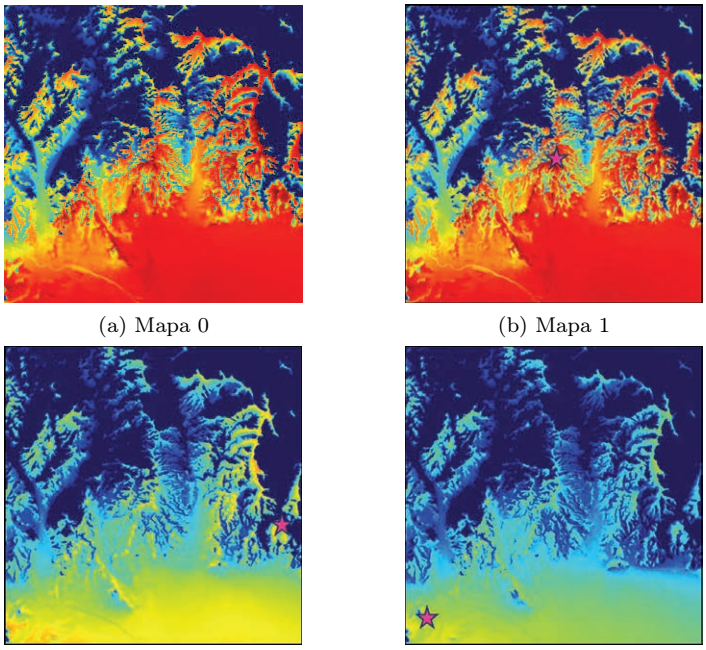
(a) MASCARA 0 (b) MASCARA 1

(c) MASCARA 2 (d) MASCARA 3

Figura 3.17: Máscara de la zona de interés y máscaras resultantes de cubrir el área de interés con torres en las posiciones optimas #1, #2 y #3 (puntos en color rojo).

Este número es de 2416436 puntos que forman la parte que no pertenecen a la zona de interés y que conforma aproximadamente el 60,41 % del total de puntos del MDE que se esta analizando. Tras calcular el mapa de las Cuencas Visuales Totales enmascaradas con esta máscara 0, resulta que la posición del observador o primera torre con máxima cobertura es 1016/1006 y el número de puntos que cubre dentro de la zona de interés es de 1085474. Este número de puntos es el que se resta a la máscara 0 para la constitución de la siguiente máscara, la 1 con un total de 35019010 puntos sin interés a la que se le vuelve ha hacer el mismo proceso para la obtención del segundo mapa de Cuencas visuales Totales enmascarado.

Tras el posicionamiento de las tres torres se alcanza a cubrir más del 94 % de la superficie total.



(a) Mapa 0 (b) Mapa 1

(c) Mapa 2 (d) Mapa 3

Figura 3.18: Mapas de las Cuencas Visuales Totales Enmascaradas usando las máscaras #0, #1 y #2 respectivamente. La estrella de la imagen (en color rosa) son las posiciones de las torres o observadores obtenidas de cada máscara.

Tabla 3.3: Localización de torres de observación y su cobertura.

Máscara	Cobertura máscara	Localización Máximo (fila/columna)	Puntos Cubiertos	Cobertura Torres
none	0 %	958 / 1028		
0 (2416436)	60,41 %	1016 / 1006 (torre 1)	+1085474	87,54 % (#1)
1 (3501910)	87,54 %	1240 /1839 (torre 2)	+166092	91,70 % (#1,2)
2 (3668002)	91,70 %	1696/21 (torre 3)	+102416	94,26 % (#1-3)

Capítulo 4

Conclusiones y trabajos futuros

Este trabajo presenta una importante contribución para los Sistemas de Información Geográfica, consistente en la elaboración de un conjunto de algoritmos para la computación de los principales parámetros de visibilidad total sobre Modelos Digitales de Elevaciones.

El algoritmo sectorial descrito en la sección 2.2 es el único, hasta la fecha, que proporciona los resultados de los parámetros de visibilidad total en unos tiempos razonables. Su eficiencia reside en la reducción a unos tamaños ínfimos de los fallos de caché en la L1. Esta reducción es independiente del tamaño del MDE de partida, ya que si el mismo supera un determinado valor, entonces se descompone el modelo de partida en MDEs más pequeños, que se procesan en paralelo.

El elemento mas importante del algoritmo es una estructura de datos compacta que consiste en una lista doblemente 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.

Por otra parte el presente trabajo ha obtenido los primeros mapas de visibilidad total para observadores no ligados al suelo, que proporcionan información tanto de las Cuencas Visuales como de un nuevo parámetro de gran valor paisajístico: la Visibilidad Volumétrica.

Gracias a la computación de las Cuencas Visuales, se ha desarrollado una aplicación basada en máscaras, que permite el posicionamiento del mínimo número de observadores que pueden cubrir de forma visual una superficie, garantizando una máxima cobertura. Es evidente que esta herramienta supone un importante avance en el campo de las Telecomunicaciones, pero, además, abre la puerta a numerosas aplicaciones en otros ámbitos, como la vigilancia forestal.

Precisamente, entre los trabajos futuros que pueden derivarse de los realizados en esta tesis se encuentra un algoritmo para el cálculo de la trayectoria óptima de drones de vigilancia y del posicionamiento de las correspondientes cámaras para la detección de conatos de incendio, mediante técnicas de inteligencia artificial (*Deep-Learning*). En el momento de la elaboración de esta memoria ya se han obtenido interesantes resultados, que se presentan en el apéndice C, al no pertenecer estrictamente a la investigación desarrollada en esta tesis.

Capítulo 5

Producción científica relacionada con esta tesis.

Las contribuciones y las conclusiones de esta tesis han sido publicadas en varios foros:

5.1. Revistas Internacionales indexadas

Efficient data structure and highly scalable algorithm for total-viewshed computation. S Tabik, **AR Cervilla**, E Zapata, LF Romero. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 8(1):304-310, Jan 2015.
Factor de Impacto: 2,145 (Q1).

Total 3D-viewshed Map: Quantifying the Visible Volume in Digital Elevation Models. AR Cervilla, S Tabik, J Vías, M Mérida, LF Romero. Transactions in GIS, 21(3):591-607, Jun 2017.
Factor de Impacto: 2.252 (Q2).

5.2. Publicaciones en *Proceedings* de conferencias internacionales con presentación oral

Siting multiple observers for maximum coverage: An accurate approach. AR Cervilla, S Tabik, LF Romero. International Conference On Computational Science, (ICCS'15), Reykjavík (Iceland), Jun 2015.
Ranking conferencia : A (Core 2014), A(Core 2018).

3D-viewshed map: a measure of landscape deepness. AR Cervilla, S Tabik, LF Romero. International Conference on Advances in Geographic Information Systems, (SIGSPATIAL'15), Seattle, Washington, Nov 2015.
Ranking conferencia: B (Core 2014), B(Core 2018).

5.3. Publicaciones en Congresos nacionales

Visibilidad total para observadores desligados del terreno. AR Cervilla, S Tabik, EL Zapata, LF Romero.
Jornadas Sarteco 2013.

5.4. Estancia de investigación en el extranjero

Centro: Universidad Técnica de Eindhoven, Paises Bajos.
Supervisor: Mark de Berg.
Fecha: De Septiembre a Diciembre de 2015.
Financiación: HPC Europa.

Apéndice A

Sistemas multi-core utilizados

El trabajo se ha desarrollado en un PC con CPU Intel E7500 dual-core y probado sobre distintos sistemas multi-core. El más relevante de los distintos sistemas multi-core probados, y del que mostramos sus características, es un nodo HP DL980G7 del Centro se Supercomputación de la Universidad de Málaga. Además indicamos las principales características de PC de desarrollo ya que en este se han hecho las pruebas previas y verificación del correcto funcionamiento de nuestro código.

A.1. PC con CPU Intel E7500

El PC de desarrollo se trata de un ordenador de la marca Dell de sobremesa cuyas principales características se muestran en la tabla A.1

A.2. Nodo HP DL980G7 del sistema Picasso de la Universidad de Málaga

El equipo HP DL980G7 utilizado en nuestro trabajo es un nodo perteneciente a un cluster del Centro de Supercomputación y Bioinnovación de la Universidad de Málaga (SCBI) es el sistema o equipo sobre el que más se ha estudiado la paralización del algoritmo presentado en este trabajo.

Sistema Operativo	Ubuntu 12.04
Memoria RAM	6GB DDR3
Procesador	Intel E7500
Conjunto de Instrucciones	64 bits
Número de cores	2
Número de subprocesos	2
Tamaño cache nivel 2	3MB
Tamaño cache nivel 1	2x64KB
Frecuencia	2933MHz.

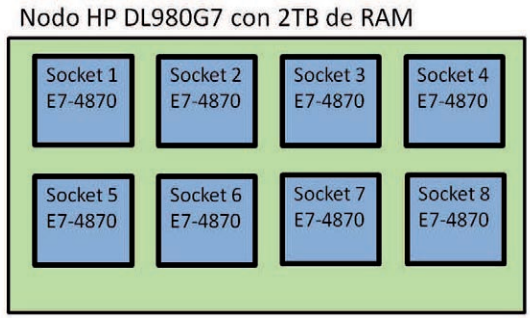
Tabla A.1: Características del PC de desarrollo.

El equipo se trata de un sistema multi-socket que en total presenta 80 cores con capacidad de 160 hilos pero, desactivada la opción Hyper-threading por el SCBI, de forma que solo puede ejecutar en paralelo 80 hilos. La organización del nodo la podemos ver en la figura A.1 y las principales características del socket E7-4870 se muestra en la tabla A.2.

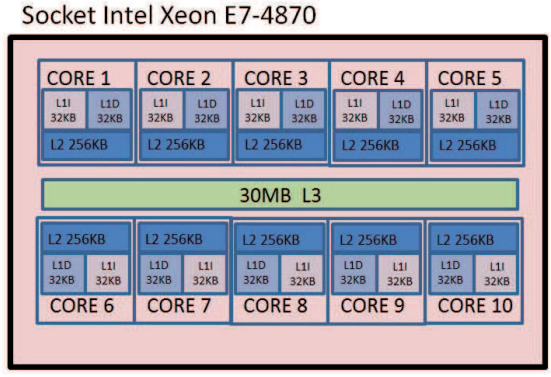
Respecto a las principales características del nodo mencionar que posee 2TB de memoria RAM y que dispone de chipset Intel 7500 con seis interconexiones de hasta6.4GT/s.

Microarquitectura	Nehalem
Plataforma	Boxboro-EX
Conjunto de Instrucciones	64 bits
Número de cores	10
Número de subprocesos	20
Tamaño cache nivel 3	30 MB
Tamaño cache nivel 2	10x256kB
Tamaño cache nivel 1	10x64KB, (32Kb L1 datos, 32kb L1 Inttrucciones)
Frecuencia	2400MHz. Turbo frecuencia2.8GHz(2 cores), 2553MHz(10 cores)
Tecnologías avanzadas incorporadas	Hyper-Threading, Turbo Boost, Virtualización E/S ...

Tabla A.2: Características de los sockets multi-core E7-4870.



(a) Nodo



d

(b) Socket

Figura A.1: Estructura de cores y cache del Nodo HP DL980G7 del Centro de Supercomputación de la Universdad de Málaga.

Apéndice B

Visualización de datos

El paso de ficheros binarios de resultados a imágenes es relativamente sencillo, basta con usar una función que convierta de valor cada dato almacenado pare parámetro de visibilidad calculado a un pixel de color para generar una imagen de salida.

En nuestro caso se ha usado una función de la librería en C desarrollada por [39]. El nombre de la librería es LodePNG y la función usada ha sido:

unsigned **encode**(const std::string **filename**, const unsigned char **in**, unsigned **w**, unsigned **h**, LodePNGColorType **colortype** = LCT_RGBA, unsigned **bitdepth** = 8)

Los parámetros entrada-salida de dicha función son:

- *filename* es el nombre de salida de la imagen, en nuestro caso con extensión .png .
- *in* el puntero o vector con los datos de entrada a convertir en imagen. Este vector ha de estar en un formato determinado para que la función pueda hacer su trabajo. Por lo que hay que hacer una previa conversión a partir del fichero binario (.dat).
- *w* la anchura de la imagen.
- *h* la altura de la imagen.

- *colortype* es el formato del vector de entrada que reconoce la función para poder ser convertido. En este caso se trata del estándar RGBA (Red Green Blue Alpha).
- *bitdepth* es la profundidad en bit de cada una de los parámetros de color del formato de entrada. En este caso se trata de 8 bit de profundidad que por 4 parámetros de color hacen 32 bits por pixel.

Lo más complicado de la función *encode* es la conversión previa para el vector *in*. Esta conversión se divide en dos partes, la primera parte generación de la paleta de conversión y la segunda la generación del vector *in*.

B.0.1. Generación de la paleta de conversión

La paleta de colores indica cual es el valor de un punto, en función del color representado sobre dicho punto.

En el presente trabajo la paleta va a ir del color azul intenso para puntos de mínimo valor, al color rojo intenso para los puntos de máximo valor, pasando por el verde y el amarillo. De forma gráfica la paleta de colores usada es como se muestra en la figura B.1.

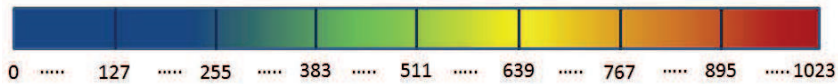


Figura B.1: Paleta de colores desarrollada para mostrar los mapas de visibilidad.

La paleta de colores se ha cuantificado en 1024 escalones de color, que van del 0 al 1023, como se puede ver en la figura B.1. El número de escalones elegido se basa en diferentes pruebas visuales para el estudio y visualización de un mapa de visibilidad.

Una vez definida la paleta por sus colores y valores, el siguiente paso es generar la paleta de conversión, que asigna a cada valor un color de la paleta en formato RGBA. La paleta de conversión consiste en un array de tamaño igual al número de escalones cuantificados, donde cada elemento del array guarda el valor correspondiente en formato RGBA de ese elemento. La función en código C implementada para la generación de la paleta de conversión se muestra en el código B.1.

Código B.1: Código en C para la generación de la Paleta de Conversión.

```
color * build_palette()
{
 color* paletacon = new color[1024]; // Paleta de conversión vacía
 float dv=1024; // Nùmero de escalones
 color c={255,255,255};//Variable auxiliar de tipo color inicializada al máximo del valor de color.

 for (int i = 0; i < 1024; i++)//Generación paleta conversión
 {
  if (i < (0.25 * dv))
  {//Ausencia de color rojo, presencia total de azul y variación del verde
   c.R = 0;
   c.G = 256*(4 * i / dv);
  } else if (i < (0.5 * dv))
  {//Ausencia de color rojo, presencia total de verde y variación del azul
   c.R = 0;
   c.B = 256*(1 + 4 * (0.25 * dv - i) / dv);
  } else if (i < (0.75 * dv))
  {//Ausencia de color azul, presencia total de verde y variación del rojo
   c.R = 256*(4 * (i - 0.5 * dv) / dv);
   c.B = 0;
  } else
  {//Ausencia de color azul, presencia total de rojo y variación del verde
   c.G = 256*(1 + 4 * (0.75 * dv - i) / dv);
   c.B = 0;
  }

 paletacon[i]=c;
 }

return paleta;
}
```

Para poder entender un poco mejor el código B.1, decir que el tipo *color* es una estructura con tres parámetros *unsigned char* de 8 bit (R,G,B), que junto con alpha constante a su máximo valor en nuestro caso conforman el formato RGBA necesario en la generación de nuestros mapas. Cada uno de los elementos RGB solo puede tomar valores de 0 a 255.

B.0.2. Generación del vector de entrada a convertir

Una vez realizada la paleta de conversión colores realizada, el siguiente paso es generar el vector de entrada **in**.

El tamaño de elementos de **in** es 4 veces el del fichero de entrada o vector de datos. El tipo de datos del vector **in** es *unsigned char* de 8 bit. Por lo tanto si los datos a convertir son flotantes de 32 bits el tamaño en memoria del vector de datos y del vector **in** es el mismo ya que en ambos casos hacen falta 32bits para cada elemento del mapa de visibilidad.

Para poder meter todos los valores entre los 1024 colores prefijados, lo que se hace es un escalado de los datos de entrada, de forma que el máximo valor del mapa de visibilidad sea menor o igual a 1023. Esto es simplemente dividir los datos de entrada a convertir por un valor que permita estudiar en detalle la imagen generada. En el caso de solo estudiar un mapa, el valor adecuado de escalado es 1024 dividido entre el máximo del valor del mapa; si se estudiaran varios el adecuado es 1024 dividido entre el máximo de todos los mapas.

La forma de generar el vector **in** queda de la forma descrita por el código B.2

Código B.2: Código en C para la generación de la Paleta de Conversión.

```
int v,x;
std::vector<unsigned char> in;
in.resize(xdim * ydim * 4); // Asignación del tamaño del vector in
for(unsigned i = 0; i < xdim*ydim; i++)
{
 v=1024.0*(data[i]/maxval);//escalado del dato de conversión
 if(v<0)v=0;
 if(v>1023)v=1023;
 x=(i\%ydim)*xdim+i/ydim;//Transformación de filas a columnas los datos
 in[4 * x + 0] = paletacon[v].R;
 in[4 * x + 1] = paletacon[v].G;
 in[4 * x + 2] = paletacon[v].B;
 in[4 * x + 3] = 255;//Valor alpha del formato RGBA
}
```

Una vez generado el vector ***in*** se ejecuta la función *encode* y en la dirección indicada nos guarda directamente la imagen del mapa de visibilidad, con el nombre especificado

Apéndice C

Trabajos derivados

C.1. Horizontes y Detección de zonas aisladas.

Horizontes y detección de zonas aisladas, se han agrupado en la misma sección, por que aunque las zonas aisladas se pueden obtener mediante la computación de diferente parámetros de visibilidad. Ha sido bajo el cálculo de los horizontes, con los que el presente trabajo ha encontrado los mejores estadísticos que permiten la obtención de las zonas aisladas de una forma sencilla y fiable.

C.1.1. Horizontes

En puntos anteriores se ha mencionado el parámetro de visibilidad llamado horizonte, pero no se ha hecho mucho hincapié en el, ya que las Cuencas Visuales y la Visibilidad Volumétrica aportan más información y son más usadas en aplicaciones que requieren un conocimiento del espacio visible de un terreno. Ahora bien, el presente trabajo ha buscado que el algoritmo de visibilidad desarrollado sirva para el estudio de cualquier parámetro de visibilidad total, que se pueda calcular a partir de los datos de un MDE de alturas. Como el estudio de los horizontes en SIG ha sido ampliamente tratado en la bibliografía [9, 34, 8, 3, 36], se ha pensado es este parámetro para el cálculo de zonas aisladas, que como veremos más adelante arroja mejores resultados que otros parámetros.

Es importante saber que el presente trabajo se ha centrado en el horizonte

visible de un observador, ya que existe distintos tipos de horizonte y por tanto, de definiciones que deben conocerse para no dar lugar a confusiones. Los tipos de horizonte son:

- Horizonte visible: se define como la línea que aparentemente separa el cielo y la superficie de la tierra. Esta línea siempre aparece a la altura de visión del observador. Por ejemplo, para un espectador humano, la línea esta a la altura de sus ojos.

- Horizonte astronómico o racional: se define como el plano que corta la esfera celeste y pasa por el observador, siendo este plano perpendicular a la la línea cenit-nadir.

- Horizonte aparente: se define como el plano ideal tangente a la superficie de la Tierra en el punto de observación. Este tipo de horizonte se basa en la forma terrestre y su aproximación a una esfera perfecta con la que calcular el horizonte.

- Horizonte geométrico: se define como la superficie cónica donde el vértice se localiza en el observador y la superficie del cono lo limita la tangente a la superficie terrestre.

- Horizonte físico o óptico: su definición es la misma que el horizonte visible o real, pero tiene en cuenta la refracción atmosférica, que permite ver por debajo del horizonte real o no, dependiendo de las condiciones atmosféricas.

Ante la variedad de tipos de horizonte remarcamos que el presente trabajo se centra en el horizonte visible, que se basa en la visibilidad real de un observador. Como desde cada localización sobre la superficie de la tierra, lo que se ve cambia (tamaño, forma), el horizonte de cada observador es único y por tanto, dos observadores que presenten distinta localización o altura nunca podrán tener el mismo horizonte. Tras la definición del **horizonte** visible de un observador(POV) en una dirección ($\vec{S_i}$) como el limite entre el cielo y la tierra visto por dicho observador. Resulta que, que a más alto y distancia del horizonte el observador solo ve cielo y antes de este el observador ve parte de la superficie terrestre.

El horizonte de un observador en una dirección, $\vec{S_i}$, viene determinado por dos parámetros, el primero es la distancia del observador al punto visible de la superficie de la tierra más alejado o punto horizonte, que hace de separación entre en cielo y la tierra, y el segundo parámetro es el ángulo que forma el plano del observador, con el plano que contiene la línea del horizonte.

Para obtener el horizonte de un observador en todas las direcciones o en un conjunto de direcciones (sectores) se puede utilizar el algoritmo desarrollado en la sección 2.2.2. Simplemente basta con añadir las lineas de código que permitan buscar cual es el último punto visible en cada uno de los sectores de análisis y guardar cual es su distancia hasta el observador y almacenar el ángulo de elevación del mismo, ya que estos puntos son los puntos de horizonte visible.

En el pseudocódigo C.1 se muestra como quedaría la implementación para la computación del horizonte sobre el algoritmo desarrollado. Este pseudocódigo corresponde a la versión secuencial, aunque, realizando las mismas modificaciones se podría ejecutar en la versión paralela. Se puede ver en lineas rojas las modificaciones echas, que son la generación de los arrays para guardar las informaciones de las distancias de horizonte y del ángulo del mismo para cada punto y cada sector. Por otro lado tras ejecutar la función *ComputarVisibilidad()* se comprueba que para el sector seleccionado se ha llegado a contabilizar al menos un segmento visible, ya que si no hay ningún comienzo de segmento visible entonces, el observador en ese sector no tiene ninguna visibilidad. Una vez comprobado que al menos existe el comienzo de un segmento visible (n_a_sec>0), se guarda la información de la distancia y en ángulo del último punto visible en un sector y que se ha obtenido dentro de la función *Kernel()*. Esta función también se ha modificado como se puede observar en el pseudocódigo C.1 con la eliminación de la búsqueda de los finales de los segmentos visibles y la incorporación del retorno de la información hacia el código principal de distancia y el ángulo-equivalente del último punto visible.

C.1.2. Detección de zonas aisladas

Uno de los principales problemas que plantea la vigilancia contra incendios y la cobertura de sistemas de telecomunicaciones en zonas de orografía escarpada, es que existen zonas especialmente aisladas (visualmente). En su mayoría, estas zonas coinciden con las cañadas de los arroyos que, tanto por su situación geomorfológica (valles muy estrechos y serpenteantes, rodeados por paredes altas de rocas), como por su abundante vegetación (casi siempre muy seca en verano), resultan especialmente sensibles a ser localizaciones para el comienzo de grandes incendios, tanto provocados como accidentales. Es habitual en dichas zonas el paso senderistas descuidados, que le apasiona lugares recónditos y delincuentes que le gusta provocar incendios en parajes de poca vigilancia. Los incendios de zonas aisladas visualmente son detectados una vez que alcanzan zonas vigiladas, momento en el cual los incendios suelen tener unas dimensiones difíciles de controlar.

Código C.1: Pseudocódigo de Horizontes.

```
LeerMDE(higth);//Lectura del MDE, datos alturas
Crear_BOS_va(bw);//Generar la estructura BOS y variables auxiliares

int dishorizonte[N][360];// Distancias horizontes
int anghorizonte[N][360];// Ángulos horizontes

int n_a_sec;//Número de anillos sectoriales
for s=0 to s<=minsec do // Recorrido de sectores
 Ordenar_Puntos(s);
 for POV=0 to POV<N do
 Actualizar(BOS);
 so=s+180;//sector opuesto al normal
 ComputarVisibilidad(POV,s,n_a_sec,distancia,anguloequivalente_h);
 if(n_a_sec>0 ) then
   dishorizonte[POV][s]=distancia;
   anghorizonte[POV][s]=arctag(anguloequivalente_h);
 endif
 ComputarVisibilidad(POV,so,n_a_sec,distancia,anguloequivalente_h);
 if(n_a_sec>0 )the
   dishorizonte[POV][so]=distancia;
   anghorizonte[POV][so]=arctag(anguloequivalente_h);
 endif
 endfor
endfor

Almacenar(parámetros calculados);//Almacenar Datos
// Eliminar datos intermedios y estructura de datos
deleteBOSandaux();
```

```
Kernel(d_POV,h_POV,d_P,h_P,&maxanguloanterior,&visible,
     &n_a_sec,&distancia,&anguloequivalente_h)
  float anguloequivalente = (h_P-h_POV)/(d_P-d_POV);
  bool es_visible = anguloequivalente >= maxanguloanterior;
  bool empieza_segmento = es_visible && !visible_segmento;

 if empieza_segmento then
    //Incremento del número de anillo-sectorial visibles
    Incremento(n_a_sec)
  endif

 if es_visible then
    distancia= d_P-d_POV;
    anguloequivalente_h=anguloequivalente;
  endif

  visible_segmento=es_visible;
  maxanguloanterior = max(anguloequivalente,maxanguloanterior)
end Kernel
```

Las zonas aisladas presentan problemas para los sistemas de Telecomunicaciones, donde al no haber una visión directa de las antenas de telecomunicación, las señales de dichos equipos llegan con menor intensidad que en zonas no aisladas donde la visión es directa. En concreto cuando una zona es visible por una antena de Telecomunicación la propagación de las ondas

electromagnéticas es directa, mientras que cuando no es visible la propagación, las ondas electromagnéticas pueden alcanzar a un observador en una zona aislada bien por rebote en capas atmosféricas, por reflexión en la superficie terrestre o difracción sobre la misma. Pero siempre que no sea una propagación directa las señales decrementan significativamente su intensidad, haciendo que el alcance sea mucho menor. Por tanto, el poder limitar de forma precisa que zonas están aisladas visualmente tiene un gran interés en un gran número de aplicaciones para vigilancia y telecomunicaciones.

En este trabajo se propone una solución, que permite identificar las zonas visualmente aisladas del resto utilizando la información que proporciona los MDE's.

Una de las primeras ideas que surge para el cálculo de zonas aisladas es usar las Cuencas Visuales por ser el parámetro de visibilidad más estudiado en la bibliografía. Para llevar dicha tarea a cabo se calcularían las Cuencas Visuales de todos los puntos, se almacenarían los anillos sectoriales y se aplicaría una estrategia de búsqueda de zonas menos visibles por todos los puntos. Esta estrategia es altamente ineficiente, ya que es altamente costoso el comparar todas las zonas no visibles de todos los puntos de un MDE.

En la búsqueda de soluciones a este problema, se piensa en la posibilidad de la reutilización del cálculo de múltiples observadores que cubran la mayor superficie, usando la estrategia de las mascara. Pero esta idea sigue siendo costosa computacionalmente y no limita una serie de puntos aislados que es lo que define una zona aislada.

Si el problema se enfoca desde otra perspectiva y se parte de un observador localizado en una zona aislada, resulta que cuando este mira a su alrededor, todo lo que ve se encuentra situado a unos pocas decenas o cientos de metros, sea cual sea la dirección que mire. Más lejos de estas distancias el observador solo ve el cielo. Por tanto, en vez de buscar que puntos son menos visibles, la idea es buscar que puntos tienen menos visibilidad, ya que la visibilidad es reciproca. Es decir un observador que es capaz de divisar solo a unos pocos puntos de un terreno, es a su vez solo visible desde unas pocas localizaciones del terreno. Pero incluso se puede limitar más la idea de observador aislado, ya que un observador que alcanza ver solo unos pocos puntos, normalmente los que ve no suelen estar a grandes distancias, sino que la mayoría de las veces son decenas de metros. Por ello, no es tanto el tamaño de la Cuenca Visual el que determina su situación de aislamiento, sino la distancia a la que se encuentran los límites de la Cuenca Visual, es decir, la distancia del **horizonte**. Los puntos aislados presentaran normalmente distancias de horizonte pequeñas.

En el presente trabajo, se piensa que cualquier métrica utilizada para detectar las zonas aisladas, debería utilizar un estadístico basado en la distancia del horizonte, ya que los demás parámetros de visibilidad son más costosos de computar. Además la probabilidad de producir un falso punto aislado es mayor en el uso de otros parámetros que no sea el horizonte, al no ser el parámetro de orden lineal, sino cuadrático como las Cuencas Visuales o cubico como la Visibilidad Volumétrica. Por ejemplo, si se localizase un observador en un profundo arroyo o barranco cerca del mar, de manera que el observador se encuentre aislado por las paredes del barranco, pero en una de las direcciones puede divisar el mar. Entonces resulta que el valor medio de las Cuencas Visuales de dicho observador se ve incrementado sensiblemente, por que en la dirección del mar, la Cuenca Visual que puede captar es enorme, por ser el cálculo de las Cuencas Visuales dependiente del cuadrado de la distancia. Si se recuerda de la sección 2.2.2, el área para una anillo-sectorial es: $A_{RS} = \frac{pi \cdot ERS^2 - SRS^2}{360}$, donde si $SRS = 0$ y $ERS = horizonte$, entonces el valor de la cuenca Visual en dicha dirección es $A_{RS} = \frac{pi \cdot horizonte^2}{360}$. De forma, que queda bien demostrado que el incremento cuadrático de las Cuencas Visuales mientras que el valor del horizonte no crece en la misma proporción al solo tener en cuenta cual es la distancia máxima visible en cada sector.

Por todo lo anteriormente citado se ha usado una distancia máxima a la que se sitúa el horizonte de un observador, calculado con la mayor de las distancias a la que se encuentra el horizonte en cualquier dirección. Por darle un nombre a la métrica, se ha llamado MHD (distancia al horizonte más lejano). En la Figura C.1 se muestra el resultado del cálculo del MHD para el MDE de las Sierras de las Nieves, dando lugar a una cartografía basada en esta métrica (se podría decir, que es la primera cartografía de estas características jamás realizada). Los tonos verdes indican las regiones con un valor bajo de MHD (horizontes muy próximos) y los tonos azulados y morados son de MHD alto.

A este mapa de horizontes generado, al que se ha llamado cartografiá de MHD, se le ha realizado un estudio con una herramienta GIS bastante conocida en el ámbito geográfico, la función *r.contour* de GRASS, para determinar las curvas de nivel en esta cartografía. Estas curvas de nivel, a las que se denominan el presente trabajo iso-MHD, son una herramienta de gran utilidad para la detección de una región aislada. Si por ejemplo, se eligen las curvas iso-MHD correspondiente a 1 kilómetro. Cualquier observador situado en dicha curva sólo puede ver, mirando a su alrededor, objetos situados como mucho a 1 kilómetro, por lo cual ningún punto de dicha línea puede ser observado por un punto que esta a más de 1 kilómetro de distancia. El resultado obtenido tras aplicar la función *r.contour* se puede ver en la figura C.2. En esta figura se pueden distinguir zonas o recintos

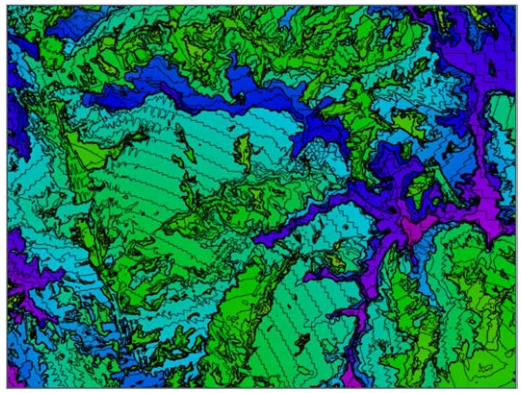


Figura C.1: Mapa de MHD total del MDE de Sierras de las Nieves.

cerrados sobre los demás tonos de la figura. Estos recintos son zonas cuyos puntos presentan un horizonte inferior a un Kilómetro; por tanto son zonas aisladas.

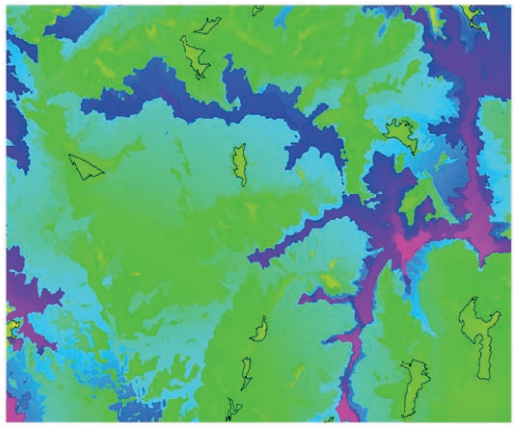


Figura C.2: Curvas iso-MHD de un Kilómetro.

Estas curvas pueden limitar zonas en las que cualquier otra iso-MHD es de menor distancia, o de mayor distancia. En el primer caso, es evidente que la curva encierra puntos dónde el parámetro MHD es incluso inferior, por lo que evidentemente es una delimitación para una región visualmente aislada. Si se quiere posicionar de una forma más clara las zonas aisladas, obtenidas mediante la función *r.contour* de GRASS, se puede proyectar sobre la herramienta Google Earth y obtener la figura C.3.

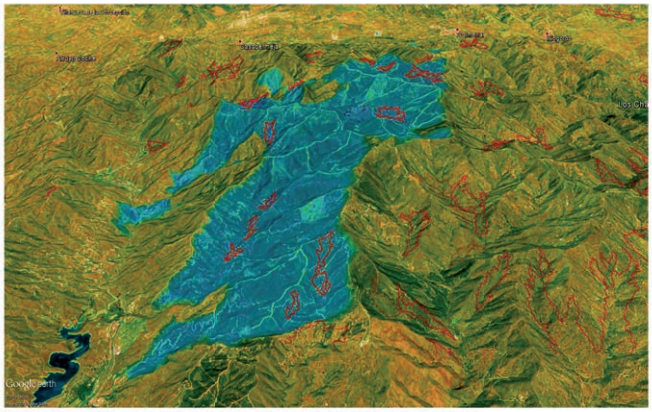


Figura C.3: Proyección sobre Google Earth de las zonas aisladas (delimitadas por líneas rojas).

Aunque se ha detallado que la distancia más lejana de la curva del horizonte de un observador es un buen estadístico para identificar que un punto esté visualmente aislado, en realidad no nos hemos basado en un criterio objetivo. Son muchos los parámetros que habría que tener en cuenta, y muchos de ellos subjetivos, para identificarlas con plena seguridad. Si se repite la situación de un observador en un profundo barranco o en un cráter con una pequeña fractura en una de las paredes, por la que se puede divisar el mar, y el horizonte esta situado a 20 kilómetros (MHD=20km). Evidentemente, este observador, así como todo el interior del cráter, quedaría excluido de las denominadas zonas aisladas, a pesar de que claramente lo es.

Precisamente ha sido el hipotético vuelo[1] que visite las zonas visualmente aisladas calculadas mediante las curvas iso-MHD la que nos ha hecho darnos cuenta de que hay zonas que dejarían de ser visualizadas. Un análisis de dichos lugares muestra que, en su mayor parte, la curva del horizonte para observadores allí localizados estaría situada a baja distancia. Pensamos por este motivo que el estadístico no debería ser el valor máximo de la distancia del horizonte sino la Media Armónica. La media armónica (designada usualmente mediante H) de una cantidad finita de números es igual al recíproco de la media aritmética de los recíprocos de dichos valores y es usualmente recomendada para promediar velocidades. Su ventaja frente a otros estadísticos es que resulta poco influida por la existencia de determinados valores mucho más grandes que el conjunto

[1] Vuelo de un drone a 20 metros de altura

de los otros, siendo en cambio sensible a valores mucho más pequeños que el conjunto. Aunque la media armónica no está definida en el caso de que exista algún valor nulo, eso no supone ningún problema ya que la menor distancia del horizonte en un MDE siempre es como mínimo, la propia resolución del modelo.

Apéndice D

Análisis del modelo de rendimiento *Roofline* del algoritmo de visibilidad

En esta sección se ofrece un breve análisis teórico del rendimiento que ofrece el algoritmo desarrollado para los MDE's de tamaño medio, ya que el MDE puede abarcar muchos Kilómetros y sin embargo el número de puntos no es extremadamente grande. El estudio de rendimiento se limita a lo que podría ser el número de accesos a memoria y la cantidad de operaciones en punto flotante en caso del cálculo del volumen total visible, utilizando el análisis teórico del modelo de techo descrito en [41]. De esta forma podemos averiguar si el algoritmo propuesto es *"memory bound"* o *"compute bound"* (Estos términos se utilizan para determinar si un código está limitado por un exesivo uso de memoria o CPU).

D.0.3. Accesos a memoria

Consideremos MDE en formato raster con n puntos. Asimismo consideremos, por simplicidad, que la rasterización consiste en una malla cuadrada de dimensiones $\sqrt{n}\times\sqrt{n}$, con distribución uniforme en las dimensiones X (oeste-este) e Y (norte-sur). El modelo simplificado descrito no sólo es el más frecuente en sistemas SIG sino que, además no supone ninguna limitación, ya que cualquier otro formato se puede reconvertir al descrito.

El algoritmo utilizado en la subsección 3.2, aplicado sobre un número de sectores s, se caracteriza por: n operaciones de almacenamiento de datos de elevación (de tipo short) en memoria ($2{\cdot}n$ $bytes$). Recordamos que por cada punto se crea una estructura (nodo) que incluye 2 enteros (números de ordenamiento original, y ordenamiento según el sector traspuesto) y dos flotantes (elevación y coordenada en la dirección del sector) ($16 \cdot nbytes$). Un MDE típico en memoria ocupa $64Mbytes$, por lo que probablemente no quepa en el sistema de caches. La superestructura de datos BOS almacena típicamente $\sqrt{n}/2$ nodos, que para un valor típico de n (4000000), suponen $16Kbytes$ que pueden almacenarse en una cache L1.

El total de acceso en lectura de datos del algoritmo es:

$$operaciones\ load = \frac{s}{2} \cdot n \cdot 16 \cdot \left[1 + \frac{\sqrt{n}}{2}\right] bytes, \tag{D.1}$$

donde el primer término del paréntesis se refiere a la actualización de nodos en la estructura, e implica probables accesos a memoria DRAM (a menos que la cache tenga capacidad para toda el MDE). El resto de accesos (consulta de datos en la BOS) suponen aciertos en L1.

En escritura, los accesos son:

$$operaciones\ store = \frac{s}{2} \cdot n \cdot 4 \cdot 4bytes, \tag{D.2}$$

ya que son cuatro datos de simple precisión los que se actualizan por punto y sector: su coordenada en la dirección del sector, su nuevo índice, y los parámetros de visibilidad calculados. Todos estos accesos se convierten probablemente en aciertos cache.

En estos cálculos no se han contabilizado la creación de la estructura de nodos ni la actualización final de resultados, ya que son insignificantes, al no estar escalados por el factor s.

D.0.4. Operaciones en punto flotante

En cada punto y sector ($s \cdot n/2$), el barrido a lo largo de la banda de visión ($\sqrt{n}/2$) incluye 4 operaciones en punto flotante por cada punto procesado:

$$minflops = \frac{s}{2} \cdots n \cdots \frac{\sqrt{n}}{2} 2 \cdots 4 \tag{D.3}$$

Adicionalmente, en caso de que se detecte un anillo sectorial de visibilidad (siendo r el número de anillos por sector y punto), se realizan nuevas operaciones con flotante: 4 en caso del cálculo de la superficie visible o Viewshed [37], y 10 operaciones si calculamos también la Visibilidad Volumétrica.

$$flops = \frac{s}{2} \cdot n \cdot \left(\frac{\sqrt{n}}{2} \cdot 4 + 10 \cdot r \right) = s \cdot n \cdot \left(\sqrt{n} + 5 \cdot r \right) \tag{D.4}$$

$$accesosdram = 8 \cdot n \cdot sbytes) \tag{D.5}$$

De las ecuaciones D.5 podemos decir que nuestro algoritmo es claramente *"compute bound"*.

La intensidad operacional (IO) resultante, para valores típicos (n=4000000, s=360 y r=12) es

$$OI = 257{,}5 flops/byte \tag{D.6}$$

Haciendo una prueba experimental de la Visibilidad Volumétrica en un mapa de 2000×2000 puntos, y usando una ventana BOS de tamaño 1001 puntos, sobre un equipo con una cpu de arquitectura Sandy Bridge modelo E5-2620, el cálculo se ha ejecutado en unos 23 segundos por sector, lo que representa un 10 % del teórico pico de ejecución publicado por intel. Haciendo otra prueba, bajo las mismas condiciones pero esta vez usando un equipo con un dual Socket Xeon E5-2698, requiere solo 1.5 minutos, es decir, unos 15 segundos por sector. Este resultado se puede considerar muy satisfactorio, ya que es un 25 % del pico.

Se ha evaluado la vectorización de los bucles que analizan los puntos de BOS y encontramos que esta estrategia, a pesar de mejorar el rendimiento de las operaciones punto flotante, penaliza drásticamente la gestión de la caché L1, debido a la utilización de una nueva matriz auxiliar. En este experimento se utilizo la computación paralela a nivel sectorial con dos hilos por núcleo, 15 núcleos, lo que se traduce en 12 sectores por hilo.

Bibliografía

[1] Akella, M. R., Delmelle, E., Batta, R., Rogerson, P., and Blatt, A. Adaptive cell tower location using geostatistics. *Geographical Analysis 42*, 3 (2010), 227–244.

[2] Atallah, M. J. Dynamic computational geometry. In *Foundations of Computer Science, 1983., 24th Annual Symposium on* (1983), IEEE, pp. 92–99.

[3] Ben-Moshe, B., Carmi, P., and Katz, M. J. Approximating the visible region of a point on a terrain. In *In Proc. Algorithm Engineering and Experiments (ALENEX'04), accepted* (2004), Citeseer.

[4] Benedikt, M. L. To take hold of space: isovists and isovist fields. *Environment and Planning B 6*, 1 (1979), 47–65.

[5] Cabral, B., Max, N., and Springmeyer, R. Bidirectional reflection functions from surface bump maps. In *ACM SIGGRAPH Computer Graphics* (1987), ACM, pp. 273–281.

[6] De Floriani, L., Falcidieno, B., Nagy, G., and Pienovi, C. Polyhedral terrain description using visibility criteria. *Unpublished, October* (1989).

[7] De Floriani, L., and Magillo, P. Algorithms for visibility computation on digital terrain models. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice* (1993), ACM, pp. 380–387.

[8] De Floriani, L., and Magillo, P. Computing visibility maps on a digital terrain model. In *European Conference on Spatial Information Theory* (1993), Springer, pp. 248–269.

[9] De Floriani, L., and Magillo, P. Horizon computation on a hierarchical triangulated terrain model. *The Visual Computer 11*, 3 (1995), 134–149.

[10] de Magalhaes, S. V., Andrade, M. V., and Ferreira, C. Heuristics to site observers in a terrain represented by a digital elevation matrix. *on Geoinformatics* (2010), 110.

[11] De Reu, J., Bourgeois, J., Bats, M., Zwertvaegher, A., Gelorini, V., De Smedt, P., Chu, W., Antrop, M., De Maeyer, P., Finke, P., et al. Application of the topographic position index to heterogeneous landscapes. *Geomorphology 186* (2013), 39–49.

[12] Fisher, P. F., and Tate, N. J. Causes and consequences of error in digital elevation models. *Progress in physical Geography 30*, 4 (2006), 467–489.

[13] Fisher-Gewirtzman, D., Shashkov, A., and Doytsher, Y. Voxel based volumetric visibility analysis of urban environments. *Survey Review 45*, 333 (2013), 451–461.

[14] Floriani, L. D., and Magillo, P. Visibility algorithms on triangulated digital terrain models. *International Journal of Geographical Information Systems 8*, 1 (1994), 13–41.

[15] Franklin, W. R., and Ray, C. Higher isn't necessarily better: Visibility algorithms and experiments. In *Advances in GIS research: sixth international symposium on spatial data handling* (1994), vol. 2, Taylor & Francis Edinburgh, pp. 751–770.

[16] Grohmann, C. H., Smith, M. J., and Riccomini, C. Multiscale analysis of topographic surface roughness in the midland valley, scotland. *Geoscience and Remote Sensing, IEEE Transactions on 49*, 4 (2011), 1200–1213.

[17] Haldemann, A., Bridges, N., Anderson, R., and Golombek, M. Rock statistics at the mars pathfinder landing site, roughness and roving on mars. In *Workshop on Mars 2001: Integrated Science in Preparation for Sample Return and Human Exploration* (1999), vol. 1, p. 45.

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

[19] Hook, J. The quantification of landform characteristics—a roughness index. In *Proceedings of the Indiana Academy of Science* (1958), vol. 68, pp. 277–282.

[20] Katz, M. J., Overmars, M. H., and Shairr, M. Efficient hidden surface removal for objects with small union size. In *Proceedings of the seventh annual symposium on Computational geometry* (1991), ACM, pp. 31–40.

[21] Kidner, D. B., Rallings, P. J., and Ware, J. A. Parallel processing for terrain analysis in gis: visibility as a case study. *GeoInformatica 1*, 2 (1997), 183–207.

[22] Lebeck, N., Mølhave, T., and Agarwal, P. K. Computing highly occluded paths on a terrain. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2013), ACM, pp. 14–23.

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

[24] Max, N. L. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer 4*, 2 (1988), 109–117.

[25] Miller, D. A method for estimating changes in the visibility of land cover. *Landscape and Urban Planning 54*, 1 (2001), 93–106.

[26] Miller, D. R., Brooker, N. A., and Law, A. N. The calculation of a visibility census for scotland. In *Proceedings of the ESRI annual conference* (1995).

[27] Mitchell, M. *An introduction to genetic algorithms*. MIT press, 1998.

[28] Nagy, G. Terrain visibility. *Computers & graphics 18*, 6 (1994), 763–773.

[29] Ogburn, D. E. Assessing the level of visibility of cultural objects in past landscapes. *Journal of Archaeological Science 33*, 3 (2006), 405 – 413.

[30] Perez, O., Telfer, T., and Ross, L. Use of gis-based models for integrating and developing marine fish cages within the tourism industry in tenerife (canary islands). *Coastal Management 31*, 4 (2003), 355–366.

[31] Pfister-Altschul, E. *Comparison of Ground-to-Air Visibility Analysis Methods*. PhD thesis, Master's thesis, 2014.

[32] Riggs, P. D., and Dean, D. J. An investigation into the causes of errors and inconsistencies in predicted viewsheds. *Transactions in GIS 11*, 2 (2007), 175–196.

[33] Schaufler, G., and Jensen, H. W. Ray tracing point sampled geometry. *Rendering Techniques 2000* (2000), 11th.

[34] Stewart, A. J. Fast horizon computation at all points of a terrain with visibility and shading applications. *Visualization and Computer Graphics, IEEE Transactions on 4*, 1 (1998), 82–93.

[35] Suleiman, W., Joliveau, T., and Favier, E. 3d urban visibility analysis with vector gis data. *Proceedings of Geographical Information Systems Research UK (GISRUK). Portsmouth, UK* (2011).

[36] Tabik, S., Romero, L. F., and Zapata, E. L. High-performance three-horizon composition algorithm for large-scale terrains. *International Journal of Geographical Information Science 25*, 4 (2011), 541–555.

[37] Tabik, S., Zapata, E., and Romero, L. Simultaneous computation of total viewshed on large high resolution grids. *International Journal of Geographical Information Science 27*, 4 (2013), 804–814.

[38] Tomlinson, R. F. *Thinking about GIS: geographic information system planning for managers*. ESRI, Inc., 2007.

[39] Vandevenne, L. Lodepng for c (iso c90) and c++, 2013.

[40] Wechsler, S. P., and Kroll, C. N. Quantifying dem uncertainty and its effect on topographic parameters. *Photogrammetric Engineering & Remote Sensing 72*, 9 (2006), 1081–1090.

[41] Williams, S., Waterman, A., and Patterson, D. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM 52*, 4 (2009), 65–76.