Efficient Data Structure and Highly Scalable Algorithm for Total-Viewshed Computation

S. Tabik, A. R. Cervilla, E. Zapata, L.F. Romero

***Abstract*—This work presents an efficient and highly scalable algorithm, designed from scratch, to calculate total-viewshed in large high resolution Digital Elevation Models without restrictions as to whether or not the observer is linked to the ground. The keys to the high efficiency of the proposed method are: i) the selection of a reliable sampling to represent the sub-areas of study, ii) the use of a compact and stable data structure to store the calculated data and, iii) the high reutilization of data and calculation between the large number of viewpoints. The obtained results demonstrate that the proposed algorithm is the fastest over the most commonly used GIS-software showing very similar numerical accuracy.**

***Index Terms*—Total-viewshed computation, Digital Elevation Model, Data-structure, Scalability.**

I. Introduction

An accurate knowledge of the visible parts of a terrain is of great interest in telecommunications, environment planning, ecology, tourism, archeology and so on. For instance, determining the least number of viewpoints that provide the maximum visual coverage in a given field would be substantially simplified if we have available the total-viewshed map of that field (Ben-Shimo et al. 2007, Franklin and Ray 1994).

The terrain is commonly represented by a regular grid of points called Digital Elevation Model (DEM), where the longitude and altitude of each point are known with exactitude. The visible areas from a specific observer situated at a high h from the ground is called viewshed. To calculate this viewshed, all the points of the DEM are possible obstacles.

Several algorithms in the literature calculate the viewshed at one single viewpoint or at most at a small set of observers (Atallah 1983, De Floriani and Magillo 1994, Hershberger 1989, De Floriani et al. 1989, Cabral and Springmeyer 1987, Miller 2001, Miller et al. 1995). Besides, the most used GIS-software, i.e., r.los under GRASS GIS 6.4 (GRASS 2011) and Viewshed tool under ArcGIS 10 (ESRI 2010) include tools that calculate the viewshed for small terrains. However, none of these methods calculate simultaneously the total-viewshed, i.e., viewshed of all the points of a terrain at once. This limitation is mainly due to the high complexity of the existing single-point-methods and also to the fact that the used data-structures were initially designed for one single observer.

There exist three strategies to make the calculation of total-viewshed more efficient or simply tractable. The first strategy consists of optimizing the basic single-point-viewshed method for specific chip architectures, for multicore (Ferreira et al. 2013), for GPUs (Zhao et al. 2013) or by including improved IO-operations (Fishman et al. 2009). The second strategy affords the total-viewshed computation by distributing the N-viewsheds calculation on multiprocessor systems without optimizing the viewshed kernel (Llobera et al. 2010, Mineter et al. 2003). While, the third strategy consists of globally rethinking and reformulating the total-viewshed problem in a way that it increases data and calculation reutilization between a large number of viewpoints. The algorithm proposed in this paper belongs to third category. It was designed from scratch to compute the total-viewshed for all the viewpoints simultaneously. Indeed, our algorithm is not suitable for single-point-viewshed calculation since the cost of the proposed data structure is compensated only when a large number of observers is involved.

As mentioned above an appropriate total-viewshed algorithm must consider the problem globally to allow an optimal data and computation reutilization. The first approximation in this context was introduced by Tabik et al. (Tabik et al. 2012) based on the following property: if a point P belongs to the frontier of a visible region of point P' then P' belongs to the frontier of the visible region of P. This algorithm reduces significantly the computational cost to $O(S.N.log(N))$, where S is the number of the considered regions (or sectors) of study, but it is applicable only to observers linked to the ground.

This paper presents the fastest and most efficient total-viewshed algorithm for observers located at a high, $h \geq 0$, from the ground. The key ideas of this algorithm are: i) the selection of reliable sampling points to represent the subareas of study, ii) the use of a compact and stable data structure to store the calculated profiles and, iii) the increase of data and computation reutilization.

The paper is organized as follows. The basis of the proposed model are provided in Section 2. The selection of the points that reliably represent the band of sight is analyzed in Section 3. The data structure used to store and calculate the profile is given in Section 4. Numerical and performance results and comparisons are given in Section 5 and finally conclusions.

II. Band of Sight

Commonly, GIS algorithms process DEMs by point, e.g., total-viewshed algorithms analyze all the N points of the terrain to identify the visible points from each point. More efficient algorithms such as the horizon algorithms described in (Stewart 1998, Tabik et al. 2011) divide the space into 360 sub-areas or sectors of one degree and find out the horizon of all the N points in each sector. It is worth noting that the way

Manuscript received April ??,??; revised December ??, ??.

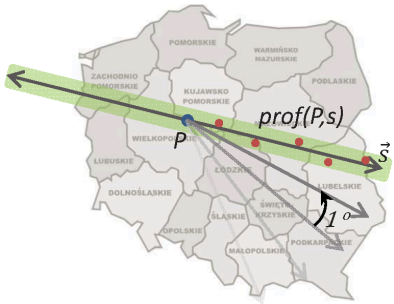


Fig. 1: The viewshed at point P is calculated in a discrete set of S directions. The viewshed is the set of visible points in the profile. Every point P has a profile, $prof(P, s)$, in direction s constituted by the closest points to this direction.

the sub-areas are defined is crucial for both the accuracy of the results and the performance of the algorithm. In this work, we utilize a new type of sub-area of study called band-of-sight which will be explained in this section.

The viewshed, $vs(P)$, of a point P of the terrain can be calculated by analyzing the points that belong to a discrete number, S, of lines of sight that start from P and radiate outward in S directions, see Figure 1 for illustration. The angle between successive lines of sight is usually fixed at one degree, 1°, as proposed in (Stewart 1998, Tabik et al. 2011). The total number of lines of sight will be $S = 360$.

A line of sight can be considered as a statistical sample of the points comprised in the angular sector of 1° crossed by $\vec{s}$. However, in most cases, the line of sight may contain very few or no points. To solve this problem, we incorporate the closest points to the line of sight s into the sample. This sample of points will be called from here on *band of sight*, $band(P, s)$. The points of $band(P, s)$ in the sense $\vec{s}$ together

Fig. 2: *The profile $prof(P, s)$ of a point P calculated in a line of sight s.*

with the points of $band(P, s')$ in the opposite sense $\vec{s'} = -\vec{s}$ form the profile of P, $prof(P, s)$, in direction $\vec{s}$.

$$prof(P, s) = band(P, s) \cup band(P, s')$$

Figure 2 shows the profile, which includes visible and nonvisible points, of a point from the city of Malaga, Spain. The vertical segment indicates the point of view P which separates the two bands of sight. Notice that the profile has all necessary information to calculate the viewshed in the band of sight.

Visible areas in sector s seen from a point P placed at a height h can be represented by segments or ring-sectors. As

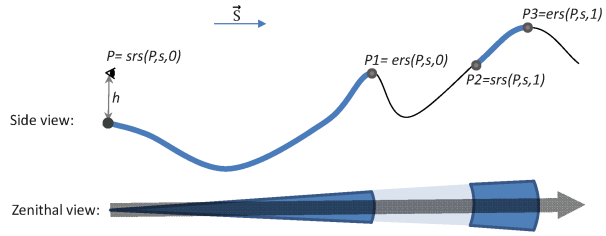


Fig. 3: Top: side view, a point P placed at a height h sees two visible segments (blue thick segments), $[P, P1]$ and $[P2, P3]$. Bottom: zenithal view, P has two visible ring-sectors delimited by $[P, P1]$ and $[P2, P3]$. The band of sight is represented by a thick dashed arrow.

shown in Figure 3, in the side view, P sees two segments, $[P, P1]$ and $[P2, P3]$, which correspond to two ring-sectors in the zenithal view. Each visible ring-sector is delimited by a lower radius or start of ring sector, $srs(P, s, i)$, and an upper radius or end of ring sector, $ers(P, s, i)$, where i is the identifier of the ring-sector. The viewshed at a point P can be calculated as the sum of all the visible ring-sectors using Algorithm 1. Where $RSsurface(srs(P, s, i), ers(P, s, i))$

Algorithm 1 Viewshed Calculation

```
vs(P) = 0
for s = 1 to S do
  for i = 1 to nrs(P, s) do
    /*Visible ring-sectors in s */
    vs(P)+ = RSsurface(srs(P, s, i), ers(P, s, i))
  end for
  for i = 1 to nrs(P, s') do
    /*Visible ring-sectors in s' = −s */
    vs(P)+ = RSsurface(srs(P, s, i), ers(P, s, i))
  end for
end for
```

calculates the surface of the ring-sector, i, in sector s, delimited by the inner and outer radius $srs(P, s, i)$ and $ers(P, s, i)$ respectively. $nrs(P, s)$ indicates the total number of visible ring-sectors from P in s. The $srs(P, s, i)$, $ers(P, s, i)$ and $nrs(P, s)$ are obtained from Algorithm 2 .

Algorithm 2 Calculation of Ring-sectors

```
for s = 1 to S do
  for P = 1 to N do
    update_profile(band(P, s))
    /* Calculation of srs(P, s, i), ers(P, s, i) and
    nrs(P, s) in s */
    computeRS(band(P, s))
    /* Calculation of srs(P, s, i), ers(P, s, i) and
    nrs(P, s) in s' */
    computeRS(band(P, s'))
  end for
end for
```

Using the concept of the band of sight reduces the complexity of the viewshed computation from $O(N)$ to $O(S \cdot N^{1/2})$. The adequacy of this sampling can be demonstrated by the fact that i) the roughness of the terrain is random in most terrains, ii) the band of sight includes abundant information where it is most required since in practice higher accuracy is needed in the closer neighborhood around the point-of-view and iii) applications such as telecommunications require less accuracy in further distances. Recall that the attenuation of the waves is proportional to $1/d^2$ while the accuracy of the proposed viewshed algorithm is proportional to $1/d$, where d is the distance between the point of view and a different point from the DEM.

The selection of the points that constitute the band of sight can be based on two assumptions. The first considers that the band of sight has a fixed geometrical width while the second considers that the band of sight has a fixed number of points. Although both methods are very similar, the first option is more homogeneous statistically while the second option is more efficient and stable computationally. In this work we used the second option.

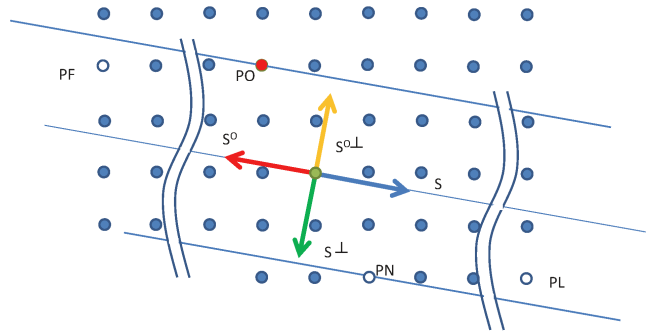


Fig. 4: The points of the band of sight are processed in direction $s^{\perp}$. PF, $P0$, PN and PL, are sentinel nodes. A video of the way of processing all the points of the DEM can be viewed in (http://bit.ly/totalvs).

III. Data Layout and Computation of the Profile

The data structure that implements the profile, $prof(P, s)$, must contain all the necessary information about the points it stores. The first characteristic of the profile is that there exists a central axis that crosses it in one unique point[1]. This point is called from here on point of view, PoV, i.e., the point to which the viewshed will be calculated. In the data structure, PoV will have HW points in each sector, in directions $\overrightarrow{s}$ and $-\overrightarrow{s}$. Therefore, the band of sight can be characterized by the total number of points it contains, $BW = 2 \cdot W + 1$, and the distance between each point of the band of sight and PoV. Actually, these two parameters are tightly linked. If the band of sight is very narrow, i.e., it contains a reduced number of points, then the distance between two points can be approximated by the difference between their coordinates in the s-axis. On the contrary, if the band of sight is too wide, the aforementioned approximation may be incorrect particularly for the points that are very close to each other[2]. This situation is analyzed in detail in Section IV.

The calculations shown along this paper use a grid of $N = 2000 \times 2000$ points, with $BW = 2001$ and $HW = 1000$ points in each side of P. Although this work considers regular DEMs, the results are also valid for non regular grids.

To simplify the generation of the profile $prof(P, s)$ in direction $\overrightarrow{s}$, we calculate the viewshed in each point of the terrain following the increasing values of the coordinates in the $s^{\perp}$-axis. See Figure 4. In this way, the set of points that belong to $prof(P, s)$ will be almost identical to the profile of the point that has just been processed in the previous iteration. That is, in each iteration, $prof(P, s)$ inserts a new point from the $+s^{\perp}$ side and eliminates an other point in the $-s^{\perp}$ side. This strategy affects the order in which the iterations of the inner loop of Algorithm 2 are processed.

The initial setup of the calculation considers that $s = 0$ corresponds to an initial azimuth =90.001^o east, thus, the points of the band of sight located in the sides s and s^o are candidates to be seen in the east and west respectively by PoV. The difference 0.001 ensures that no two points or more would be along the same line since the points of the DEM are aligned with the cardinal directions. For this sector, the points are processed from north to south starting by the point situated in the northeastern corner. See video (http://bit.ly/totalvs). Processing in this specific order and for a fixed size of the profile will guarantee:

- All the points included in $band(P, s)$ are consecutive
- In each profile, there is only one different point with respect to the profile of the last processed point.

Among the *BW* points of the band of sight, there are five special points or sentinel nodes (see Figure 4):

- P or PoV: the current point-of-view
- PO: the oldest point inside the band of sight and with the smallest index or coordinate in $s^{\perp}$
- PN: the newest point with the largest index in $s^{\perp}$
- PL: the farthest point from PoV in direction s
- PF: the farthest point from PoV in direction s^o

Using this ordering, the viewshed computation at one point can be substantially simplified as shown in Algorithm 3.

Algorithm 3 is valid once the profile of the previous point is known. The first and the last *HW* iterations are treated differently. In the first *HW* iterations, function $update\_profile()$ adds two points per iteration without eliminating any. Whereas in the last HW iterations, it only eliminates and does not add any new point. The inner loop of Algorithm 2 can be rewritten as shown in Algorithm 4.

[1](Tabik et al. 2011) demonstrates that for S directions selected appropriately, there is no two points that belong to the same segment.

[2]Notice that the closest points to the considered point of view do not belong to the sector. Thus, a special processing is applied to them.

Algorithm 3 Profile Calculation

/ The profile of the previous point P must be known*/*
Require: $prof(P, s)$
/ eliminate the oldest point in the band-of-sight */*
delete PO
/ insert a new point into the band-of-sight */*
insert PO
/ Update PoV in order $s^{\perp}$*/*
$P = P + 1$
$identify(PL, PF)$

Algorithm 4 Calculation of the viewshed in a sector

for $P = 0$ to N **in the direction** $s^{\perp}$ **do**
  if $P < HW$ **then**
    $update\_profile(band(P, s), init\_phase)$
  else if $P >= N - HW - 1$ **then**
    $update\_profile(band(P, s), closing\_phase)$
  else
    $update\_profile(band(P, s), stationary)$
  end if
  $computeRS(band(P, s))$
  $computeRS(band(P, s^o))$
end for

A. The Profile Data Structure

The organization of the points inside the profile data structure is critical for the performance of our implementation. Recall that while the points are inserted and deleted in direction $s^{\perp}$, the calculation of the visible ring-sectors within the profile $prof(P, s)$ is performed in directions s and s^o for $band(P, s)$ and $band(P, s^o)$ respectively. Therefore, the data layout must take into account this double reordering in $(s, s^{\perp})$ coordinate system. An appropriate data structure that allows the analysis of the elements in a non-correlated reordering is a list doubly linked via an array of nodes, see Figure 5. These nodes can be ordered in memory either in direction s or $s^{\perp}$.

1) Ordering the Nodes in Direction $s^{\perp}$: In this case, the insertion and elimination of a point are performed in one step by converting the matrix of nodes into a circular list with a cyclic pointer that indicates the position of the head of the list, which is the tail at the same time. Otherwise, the calculation of the profile will need to walk through the links from P to PL in direction s, and from P to PF in the opposite direction, s^o. In this way, the kernel of the algorithm will access the memory in a disordered way using the links in the linked list. The nodes of the list are of type struct and contain six elements:

struct node{
  $node.id$ //identifier of the point
  $node.d$ // coordinate in direction s
  $node.h$ // elevation of the point
  $node.os$ //the order number in direction s
  $node.next$ // a pointer to the nearest node in direction s
  $node.prev$// a pointer to the nearest node in direction s^o
}

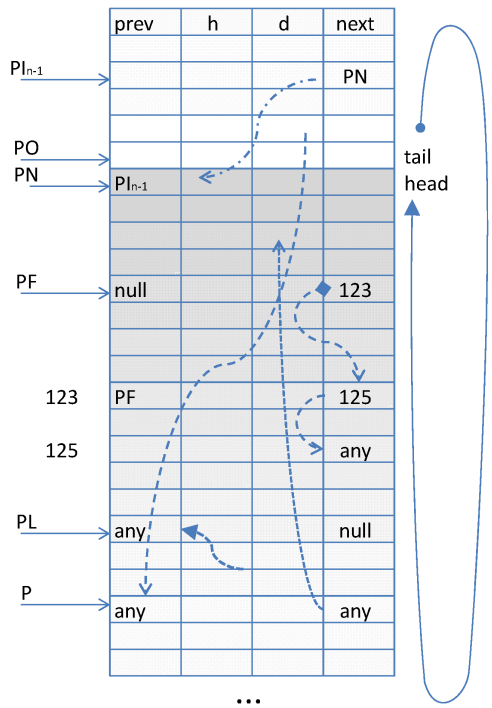


Fig. 5: The data structure that implements the band of sight. The points are ordered according to their position in memory from the oldest point, with smaller $s^{\perp}$ coordinate, i.e., PO (rows shown in lighter color background) to the most recent point, i.e., PN (rows shown in darker color background). The point of view P is situated at the center of the circular data structure. The pointers $prev$ and $next$ maintain the ordering of the nodes from PF to PL. The points from PF to P represent the band of sight in $s^{\perp}$, $band(P, s^{\perp})$ (dashed-line arrow). The points from P to PL represent the band of sight in direction s, $band(P, s)$ (dotted-line arrow). A priori, PO, PN, and the point of insertion, PI, can be in either side, s or $s^{\perp}$, of point P.

- **Optimizations**
  Ordering the nodes in memory in direction $s^{\perp}$ has the following disadvantage. Inserting new nodes in the list by function $update\_profile(prof)$ needs to know a priori where the insertion will be produced. Therefore, it is necessary to walk along the list to find the exact location using parameter $node.os$. This search can be carried out in the inner loop in Algorithm 2. Alternatively, we introduced an important optimization in this context by recalculating once the location of the inserted nodes in

the list. Then, the location of the insertion, PI, or point of insertion is declared as a new sentinel node in addition to the five existing ones. The N precalculated points of insertion are stored in a file and can be reused by any terrain with identical dimensions.

Indeed, in this work, we calculated the total-viewshed of the DEM of Andalusia, Spain, composed of 416 square blocks of dimension 2000×2000 points. This means that $4.000.000$ indices of the points of insertion PI are identical among all the blocks and can be precalculated once for the first block. In addition, once the point of insertion is known, it will be unnecessary to store the parameters $node.os$ and $node.id$. This will further simplify the data structure by exclusively storing the pointers of the linked list of type short integer, the height h of type unsigned short and, the coordinates $node.d$ of type float for DEMs of precision 0.1 meters or higher. This information, of about 20KB, can fit in most L1 caches of most processors.

Finally, it is essential to notice that the efficiency of this algorithm is based on the stability of the data structure, namely the size of the band of sight is constant independently of the analyzed point and sector. Using a band of sight of fixed width would introduce a slight irregularity of negligible impact on the numerical result but with a significant reduction of the computational cost.

2) Ordering the Nodes in Direction s: In this case, updating the profile using function $computeRS()$ in Algorithm 2, which represent the most costly and time consuming part of the code, will access the memory linearly but for inserting and eliminating new nodes, they will perform a complete realocation of the information in memory. The experiments demonstrate that this organization is less efficient, including on architectures with L1 data cache of size smaller than the profile. This is due to the fact that all the benefit obtained from the lineal access to data memory in the inner loop is completely hidden by floating point operations.

B. Calculation of Visible Ring-Sectors

The calculation of the visible ring-sectors between P and PL can be summarized in Algorithm 5. The calculation of the profile in the opposite direction, s^o, is performed using the same function but traversing the linked list in the opposite direction, from P to PF using $node.prev$ links.

The final algorithm processes the four nearest points to PoV in each band of sight differently due to their proximity and because the relative deviation with respect to the axis of the band can affect the quality of the final results. This particularly affects the initial values such a $currN$ and the maximum angle max in Algorithm 5.

Algorithm 5 Calculation of the visible ring-sectors of P in sector s

Require: computeRS(band(P,s))
  $d0 = P.d$;
  $h0 = P.h$;
  bool $visible = false$
  float max = -∞ // Max angle
  $currN = P$
  while $currN! = PL$ **do**
    $currN = currN.next$;
    kernel$(d_0, h_0, currN.d, currN.h, \&max, \&visible)$
  end while

Require: kernel$(d_0, h_0, d, h, \&max, \&visible)$
  float $angle = (h - h_0))/(d - d_0)$;
  bool $this\_visible = angle > max$
  bool $opening = this\_visible$ && $!visible\_area$
  bool $closing =$ $!this\_visible$ && $visible\_area$
  if opening **then**
    $store\_srs(d)$
    $nrs[P, s] + +$
  end if
  if closing **then**
    $store\_ers(d)$
  end if
  $visible\_area = this\_visible$
  $max = max(angle, max)$

IV. Results

Due to the absence of authentic total-viewshed algorithms and also to the difficulties of comparing algorithms of different purposes, we provide in this section a comparison of all the comparable aspects between our total-viewshed algorithm and the most commonly used single-point viewshed software. For the numerical comparisons we analyze the relative difference values and graphically illustrate one-point-viewshed maps calculated using all the compared software.

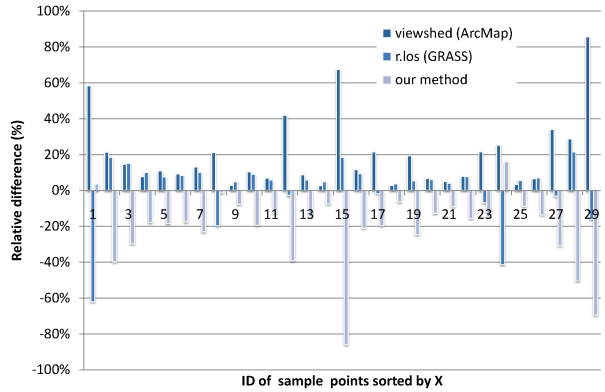
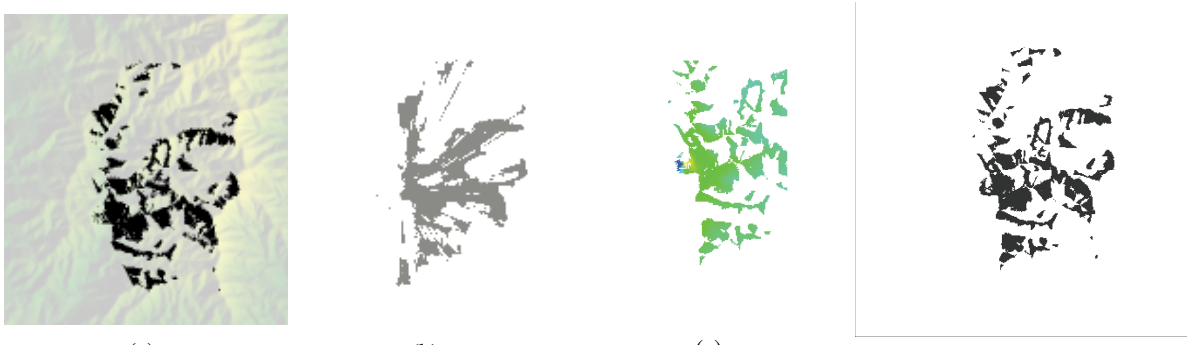


Fig. 6: *Comparison of relative difference, in %, between viewshed tool (ArcMap), r.los (GRASS), and our model with respect to the mean value of the three models.*

A. Comparisons of Relative Differences

In the first analysis we compare the viewshed of 29 different points calculated using r.los under GRASS, viewshed tool un-



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

Fig. 7: Viewshed at point UTM (zone 30S) of coordenates, X=368.800 and Y=4.070.140, in a 2000×2000-points DEM of resolution $10 * 10m^2$, using (a) the algorithm proposed in this work, (b) the model proposed in (Tabik et al. 2012) (c) r.los under GRASS and, (d) Viewshed tool under ArcGIS.

der ArcGIS and our method. Notice that 29 is a representative sample statistically.

Figure 6 compares the relative differences of the three tools with respect to the mean value calculated as:

$$Relative\ difference(model_i) = \frac{\#visible\ km^2(model_i) - meanValue(viewshed, r.los, ours)}{meanValue(viewshed, r.los, ours)}$$

As can be observed in this Figure, the three models provide similar values for about 69% of the total number of points with a relative difference $\in [-20\%, +20\%]$, which can be considered more than acceptable due to the fact that the viewshed problem is numerically instable since a slight change in the position of the observer PoV may produce a huge change in the size of the viewshed. Usually, in this kind of problem, differences in the order of 25% are acceptable (Maloy and Dean 2001). Besides, our algorithm gives the nearest value to the mean value for point 1. However, the three models obtain dissimilar values for points 12, 15 and 29. This can be due to the fact that for very instable viewpoints it may happen that the nearest elevations outside the considered sector maybe neglected (Stewart 1998). Due to the instability of the problem, the calculated viewsheds of these instable points, using the three tools, correspond in fact to viewsheds of real points in the closer surrounding area. In consequence, this is translated into a slight shift, of pixel scale, of these viewpoints in the final total-viewshed map.

In addition, the impact of different sizes of the band of sight, BW, on the quality of the numerical results has been also analyzed. We found that sizes from 50% to 200% with respect to $\sqrt{N}$, hardly affect the quality of the results. However, for values outside this range, the quality of the results degrades substantially. A deeper study could determine the optimal size of the band of sight in terms of the angle. To make this possible, one must contrast real viewshed data with the calculated ones, which is not feasible (Wechsler and Kroll 2006). The same problem happens with the number of sectors S. Therefore, values in $[90^o\ 360^o]$ are enough to ensure reliability of the results without excessively affecting the performance. Notice that although the complexity of the calculation is directly proportional to S, the scalability is not at all affected by S.

B. Comparisons of the Shape of Viewshed Maps

A comparison of the viewshed calculated at a specific point in the DEM of Malaga, Spain, using our model, algorithm (Tabik et al. 2012), r.los under GRASS and, Viewshed tool under ArcGIS is shown in Figure 7. Where, the visible surface calculated by our total-viewshed algorithm, algorithm (Tabik et al. 2012), r.los and, Viewshed is, 8411, 8241, 8583, and 8684 dam^2 respectively. In general, the four models show very similar viewshed shapes and numbers.

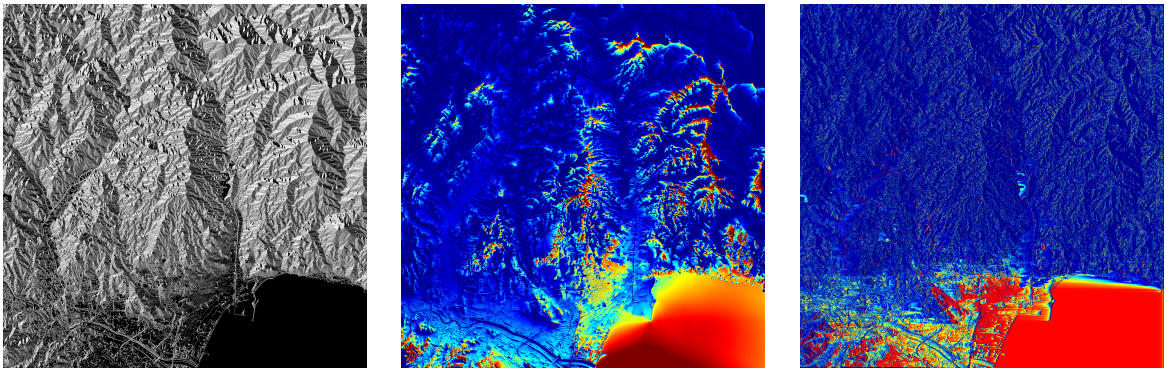
Figure 8(b) shows the viewshed maps of a block of the DEM of Andalusia for an observer situated at $2m$ and Figure 8(c) the difference between the map shown in Figure 8(b) and a viewshed map of an observer located at 0 meters from the ground. These differences are in the order of tens of km^2 with a percentage larger then 1000% in smooth regions. This demonstrates the limitations of the models where the observer cannot be situated above the terrain.

C. Performance Comparisons

TABLE I: Performance comparison between our total-viewshed algorithm, the algorithm described in (Tabik et al. 2012), r.viewshed (GRASS) and Viewshed(ArcMap).

Viewshed tool	Runtime (seconds)
Our total-viewshed algorithm	0.0032
(Tabik et al. 2012)	0.0063
r.viewshed (GRASS)	18
Viewshed (ArcMap)	10

This section provides a comparison between our total-viewshed implementation, r.viewshed (an optimized viewshed tool under GRASS) and Viewshed (ArcMap) on an Intel(R) Core(TM)2 Duo Processor E8500. We compare the runtimes of r.viewshed and Viewshed on one viewpoint versus the runtime of our algorithm divided by $4.000.000$, the total number of points of the DEM used in the experiments. The average runtimes of r.viewshed and Viewshed shown in Table 1



(a) (b) (c)

Fig. 8: (a) A DEM of the city of Malaga, Spain, of resolution $10 * 10m^2$. (b) Viewshed calculated at an observer situated at 2 meters from the ground, where the red and blue colors correspond to high and low visibility respectively. (c) Map of the difference between the viewshed maps of an observer situated at 2 and 0 meters from the ground.

were measured using 30 viewpoints selected randomly. From this Table, one can observe that our algorithm is more efficient than the cited related works. This result corroborates the fact that total-viewshed algorithms must be redesigned from scratch and using single-point viewshed algorithms to compute the total-viewshed is an inefficient strategy.

V. Conclusions

This work presents an important contribution to the GIS community, the fastest and most efficient total-viewshed algorithm without restrictions whether or not the observers are linked to the ground. The key ideas are: i) selecting reliable sampling points to represent the subareas of study, ii) using a compact and stable data structure to store the calculated profile and, iii) increasing data and computation reutilization. These contributions are not only valid for total-viewshed algorithm but also for most algorithms that need to sweep a regular map from a specific angle, such as wavefront methods used in video codification and axial tomographic reconstruction. The total-viewshed implementation described in this paper is available at (www.ac.uma.es/∼siham/tvs.tar.gz).

The total-viewshed maps obtained by our algorithm has been used in a large number of applications. An example of the total-viewshed from a specific area can be visualized in (http://bit.ly/totalvs).

Acknowledgment

This work was supported by the Spanish Ministry of Education and Science throughout project TIN2010-16144, Junta de Andalusia through Project TIC-8260.

References

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

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

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

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

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

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

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

[8] R. Franklin. (2002) Sitting observers on terrain. Symposium on Spatial Data Handling.

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

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

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

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

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

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

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

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

[17] M. Mineter, S. Dowers, D. Caldwell,and B. Gittings. (2003) High-Throughput Computing to Enhance Intervisibility Analysis. 7th International Conference on GeoComputation, 1–10.

[18] A.J. Stewart. (1998) Fast Horizon Computation at All Points of a Terrain With Visibility and Shading Applications. IEEE Transactions on Visualization and Computer Graphics, 4(1), 82–93.

[19] L. F. Romero, S. Tabik, http://bit.ly/totalvs.

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

[21] S. Tabik, L. F. Romero and E. L. Zapata. (2012) Simultaneous Computation of Total-Viewshed on Large High Resolution Grids. International Journal of Geographical Information Science, In press.

[22] J. Fishman, H. Haverkort, L. Toma. (2009) Improved visibility computa-

tion on massive grid terrains, In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, 121-130.

[23] Y. Zhao, A. Padmanabhan, S. Wang. (2013) A parallel computing approach to viewshed analysis of large terrain data using graphics processing units, International Journal of Geographical Information Science, 27(2), 363-384.

[24] C.R. Ferreira, M.V.A. Andrade, S.V.G. Magalhes, W.R. Franklin and G.C. Pena. (2013) A Parallel Sweep Line Algorithm for Visibility Computation, Proceeding of the XIV Brazilian Symposium on Geoinformatics.