Computers & Geosciences 59 (2013) 116–123

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

# A flooding algorithm for extracting drainage networks from unprocessed digital elevation models

Antonio Rueda *, José M. Noguera, Carmen Martínez-Cruz

*Departamento de Informática, Escuela Politécnica Superior, Universidad of Jaén, Paraje Las Lagunillas s/n, 23071 Jaén, Spain*

ARTICLE INFO

*Article history:*
Received 26 March 2013
Received in revised form
26 May 2013
Accepted 10 June 2013
Available online 20 June 2013

*Keywords:*
Drainage networks
Overland flow algorithms
Digital elevation models

ABSTRACT

A new method for extracting the drainage network from a digital elevation model (DEM) is presented. It is based on the well-known D8 approach that simulates the overland flow but uses a more elaborate water transfer model that is inspired by the natural behaviour of water. The proposed solution has several advantages: it works on unprocessed DEMs avoiding the problems caused by pits and flats, can generate watercourses with a width greater than one cell and detects fluvial landforms like lakes, marshes or river islands that are not directly handled by most previous solutions.



## 1. Introduction

Since the early 1980s, a vast amount of research has been devoted to the hydrological analysis of digital elevation models (DEMs) (see Barták, 2009 for an extensive review of existing literature). In particular most attention has been focused on the identification of the river network and catchment boundaries. This data is of extreme importance, since it is necessary for simulation of rainfall-runoff and river basin management among other applications.

Most methods are based on the modeling of the overland flow, as O'Callaghan and Mark (1984) originally proposed. This approach must always work in a hydrological correct DEM, i.e., those which meet the premise that starting out from any cell and following the greatest slope we can reach the edge of the DEM. In practice, diverse anthropogenic features, vegetation, measure and interpolation errors or lack of accuracy generate closed depressions (or pits) where the water flow stops and flat areas where flow directions are difficult to assign. The importance and complexity of these anomalies have motivated the design of many different solutions that share a common approach based on preprocessing the DEM for modifying its topography to resume the circulation of the water flow. However in many cases these modifications can be rather artificial and lead to incorrect drainage networks or even remove real terrain features like volcanic craters or lakes.

We propose a new approach for computing the drainage network of a DEM that avoids the effect of pits or flat areas in an elegant and straightforward way, and therefore works with non-hydrological correct DEMs. It can also generate wide watercourses and detect lakes, reservoirs or marshes, getting results that are closer to the actual shape of the river than other methods. The key of this new approach is extending the classic method of O'Callaghan and Mark by considering the level of water in each cell and enforcing the same level for neighbour cells.

The rest of the paper is organized as follows. Section 2 provides a necessary background and contextualizes our proposal. Section 3 presents our new algorithm and discusses how it handles problematic DEMs, such as those containing flat areas and closed depressions. Section 4 studies the influence of the parameters of our algorithm and its performance. Finally, Section 5 presents the conclusions and outlines some future work.

## 2. Previous work

The extraction of the river networks from a DEM can be done by different methods. Peucker and Douglas (1975), Douglas (1986), Tribe (1992) determine ridge and valley lines by topographic evaluation. Meisels et al. (1995) proposed an original solution based on performing a skeletonization process on the set of elevations of the DEM. But by far the most widely used approach is based on the work of O'Callaghan and Mark (1984) that uses a simulation of water flow over terrain to extract the drainage information.

The basic implementation of this approach has three stages. In the first stage, one or more drainage directions are assigned

* Corresponding author. Tel.: +34 953 212 893; fax: +34 953 212 472.
*E-mail addresses:* ajrueda@ujaen.es (A. Rueda), jnoguera@ujaen.es (J.M. Noguera), cmcruz@ujaen.es (C. Martínez-Cruz).

0098-3004/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2013.06.001

from each DEM cell to its eight neighbours following the line of steepest slope. In the second stage, a unit of flow is set to each cell, and the DEM is repeatedly scanned row by row. During each iteration, the cells containing a non-zero flow are determined. Next, the flow of these cells is transferred to their neighbours following the drainage directions computed before. In the final stage a threshold is chosen and all cells with an accumulated transferred flow greater than this threshold are included as part of the drainage network.

There are several strategies for assigning the flow direction from a given cell. The simplest one, used by O'Callaghan and Mark (1984) in their seminal paper is choosing the neighbour to which the slope is steepest. This is generally known in the literature as D8 although a more precise designation is SFD8 (Single Flow Direction chosen from 8 options). If the flow is divided among all the neighbours with lower elevation according to their local slope the method is called MFD8 (Freeman, 1991). Subsequently Tarboton (1997) and Seibert and Brian (2007) proposed a more general solution that considers the steepest slope in the $0^\circ$ to $360^\circ$ range around the cell, although the direction is finally assigned to one or two neighbours (SFD$\infty$), or more (MFD$\infty$).

The methods described before for assigning flow directions fail in situations where a cell or a group of cells is surrounded by cells of higher elevation (pits). The simplest method for removing a pit proposed by Jenson and Domingue (1988) involves finding the outflow point (i.e., the cell on the boundary of the pit through which the water will overflow when it is filled with water) and filling the pit by altering the height of the cells to the height of the outflow. This solution has two important drawbacks: it can significantly change the real topography of an area leading to incorrect results and also generates flat areas that have to be treated. The alternative approach followed by Rieger (1998) and Jones (2002) is based on decreasing the elevation of the cells in the pour point and its surroundings to ensure a natural outflow of the water. This method does not generate flats and is more adequate for processing large pits than the previous one although again the DEM can sometimes be altered in unrealistic ways.

Treatment of flats is also required as a preprocessing stage before applying a flow transfer approach. The first solution, again described by Jenson and Domingue (1988) is altering the elevations of the flat to construct an inclined plane towards the lower neighbours of the flat. As this approach can result in unrealistic parallel flow lines across the flat, Garbrecht and Martz (1997) and Barnes et al. (in press) subsequently proposed combining the gradient towards lower neighbours with the gradient from higher neighbours. In certain cases this may lead to new small depressions that have to be solved again by using the gradient towards lower neighbours approach. But eventually these two methods are based on assumptions and therefore it is not clear that they will produce a result that resembles the real river morphology in large and complex flats.

Alternatively, a few methods have addressed the problem directly without preprocessing the DEM. The best known is based on the search of the least-cost drainage paths (LCPs) by using the A* search algorithm (see Ehlschlaeger, 1989) and an improved version has been implemented in the *r.watershed* GRASS module by Metz et al. (2011). This method starts at the boundary of the DEM and visits all the cells using a search that follows the least steep uphill slope, or the steepest downhill slope when a depression is encountered. The result of the search is the flow directions for each cell and a standard flow accumulation can then be applied to compute the stream channels. More recently (Magalhaes et al., 2012) have proposed a simple and intuitive approach that starts by considering the DEM as an island and then raises the outside water level step by step until the entire DEM is submerged. As the water level increases, it gradually floods the cells of the DEM, filling the depressions and spreading on flat areas. The order in which cells are reached by water determines the flow directions (as water gets into a cell, the flow directions of its neighbours are set towards it). Again a flow accumulation stage finishes the computation of the drainage network.

## 3. Flooding algorithm for drainage network determination

The approach described in this work is based on the modeling of the outland flow but is novel in three main aspects:

- A cell is not initialized with a unit of flow but with a water layer of a given height that also contributes to the global height of the cell.
- A SFD8 strategy is used to transfer the water to a neighbour cell, but in contrast with a classic flow-based approach, the neighbour to which the slope is steepest changes from one iteration to the next due to the changes in the height of the water layer. A second difference is that not all the water is transferred to the neighbour; instead, the same water level in both cells is enforced when possible.
- No preprocessing of pits and flats is required, as the water layer fills the pits when required and also runs through the flats to continue its course.

### 3.1. Algorithm description

Our approach for extracting a drainage network from a DEM can be summarized in the following steps:

1. Simulate a homogeneous rainfall flooding the entire DEM by initializing all cells with a constant water depth value of $W$.
2. Simulate the water spilling over the DEM by iteratively draining water between adjacent cells. For each cell $c$ of the DEM, compute the neighbour $n$ to which the slope is steepest considering the sum ($ZW$) of the altitude ($Z$) of the cell and the height ($W$) of the water layer (see Fig. 1). Transfer water from the cell $c$ to $n$ to enforce the same level. Update the drainage accumulation ($DA$) of the cell $n$. Repeat this procedure while the overall amount of water transferred is greater than 0 (or a small predefined epsilon).
3. Mark as belonging to the drainage network the cells with a drainage accumulation greater than a predefined threshold.

The flow direction determination and water transfer in stage 2 is the key part of the algorithm. Given a cell $c$, the algorithm first checks whether the cell is in the border of the DEM. If this is the case, the water it contains is drained out of the DEM and the $W$ value of the cell becomes 0. Otherwise, the drainage direction for the cell is computed on the fly as the direction to the $3 \times 3$

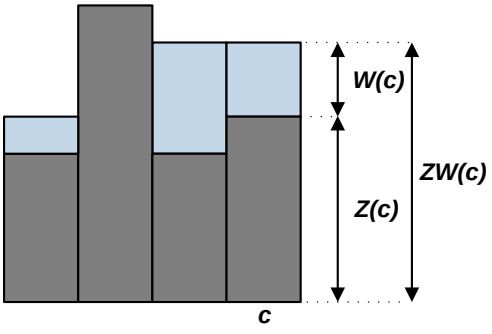


**Fig. 1.** Example illustrating some cells of a DEM and the notations used in the paper.

neighbour cell $n$ which has minimum elevation $ZW(n)$. Note that $ZW(n)$ values for neighbours in the diagonal directions of $c$ are adjusted by a $1/\sqrt{2}$ factor to compensate for the increased path length to the diagonal.

If the water level $ZW(n)$ of the selected neighbour cell $n$ happens to be higher than the level of the current cell $c$, then the cell $c$ is skipped and no water transfer occurs. Otherwise, the algorithm tries to level the values $ZW(c)$ and $ZW(n)$ of both cells by draining part (or all) of the water contained in $c$ to $n$ according to Eq. (1). Fig. 2 illustrates this process and provides some examples.

$$moving\_water = \min\left\{W(c), \frac{ZW(c)-ZW(n)}{2}\right\} \quad (1)$$

A minimum operation is needed in Eq. (1) to ensure that the cell does not drain more water than it contains, as depicted by Fig. 2b. The transferred water is added to the height level $W(n)$ of the neighbour to which it drained, and decreased to the height level $W(c)$ of the current cell. The drainage accumulation value $DA(c)$ for the current cell is also increased by the amount of water transferred (if any).

Notice that in contrast to D8 algorithms, in our proposal the flow direction of each cell cannot be computed beforehand as it can vary during the execution of the algorithm. Specifically, the drainage directions depend on two values: the altitude of the cells ($Z$) and their current water level ($W$). While the first value is constant, the second one typically varies from one iteration to the next. Two interesting unique properties of our algorithm derive from this:

- The water trapped in a pit can overflow once the pit is filled. In such situations, lower cells can actually drain into higher ones, as shown in Fig. 2c. This feature makes our algorithm very robust and capable of working with unprocessed DEMs containing pits or flats, as will be detailed in Section 3.2.
- The flow direction of any cell can vary between iterations, according to $W$. Therefore, our algorithm allows divergent flow direction for each cell, resulting in more realistic drainage networks that can be wider than one cell.

It also follows that if the cells are initialized with a very small value $W$, then $ZW(c) \approx Z(c)$ and the algorithm generates a result similar to the classic SFD8 of O'Callaghan and Mark (1984).

The water transfer stage must continue until every cell $c$ of the DEM fulfills at least one of the following conditions:

1. $W(c) = 0$.
2. $ZW(c) \leq ZW(n)$ for every neighbour cell $n$ of $c$.

That is, until every cell $c$ of the DEM has either drained out all its water (1) or its water level is equal to the water level of the eight adjacent cells (2). In this situation no more water transfers can occur in the DEM. In practice it suffices to check if the sum of the water drained by all the cells after a complete iteration is 0 or alternatively below a small predefined threshold. Algorithm 1 shows the detailed pseudocode of this stage.

Finally in the last stage, the drainage channels are identified using the common approach based on the flow accumulation proposed by O'Callaghan and Mark (1984): cells that have a drainage accumulation $DA$ above a user-specified threshold are considered to be river cells.

Fig. 3 shows two sample drainage networks computed by the algorithm described in this section. The DEMs were extracted from the SRTM3 v2 dataset. The cells have been initialized with a water layer of 50 mm and the simulation finished when the water transferred in a iteration fell below 1% of the total amount of water initially dropped on the DEM (314 iterations). Cells with a drainage accumulation above 2000 mm have been marked as belonging to a drainage channel. The drainage network of the area can be clearly seen, including reservoirs like *Pantano de Bermejales* (top of the left subfigure).

### 3.2. Processing of pits and flat areas

As explained in Section 2, D8 algorithms require the assignment of a proper flow direction to each DEM cell in a preprocessing stage. This flow direction is computed as the direction with

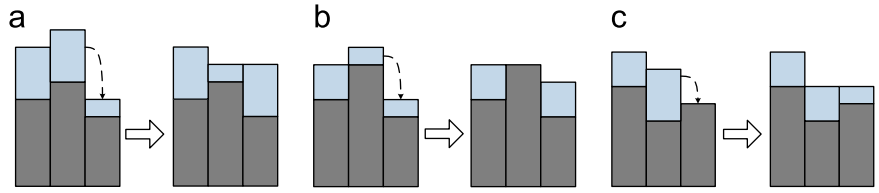


**Fig. 2.** Water levels before and after a water transfer operation between the midcell and a neighbour cell. (a) The midcell drains part if its water until the $ZW$ values of both cells are levelled. (b) The midcell does not contain enough water to level both cells, so it simply drains all its water to its neighbour. (c) Some water escapes from a pit by draining to a higher cell.

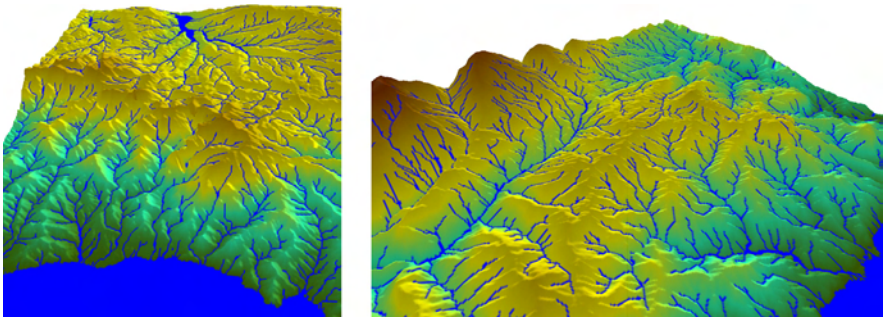

**Fig. 3.** Sample drainage channels computed by our method of the south of Granada (Spain), including the Mediterranean Sea and *Almijara*, *Contraviesa* and *Sierra Nevada* ranges.

the steepest slope away from the cell. However there are two situations in which drainage direction assignment becomes problematic: flat areas where neighbour cells have the same height and closed depressions of the DEM (pits). These cases must be treated before calculating the overland flow.

Our algorithm overcomes these problems in a straightforward way by just imitating the natural behaviour of water. Small depressions and pits are gradually filled-up by water accumulation until the water level reaches one or several pour points, resuming the course downhill. In order to do this, the flow directions of the cells included in such depressions and pits are adaptively re-assigned by our algorithm according to the current cell level *ZW* of the cells and their neighbours. Therefore, there is no need to treat any special case before calculating the overland flow.

Fig. 4 presents an example of a problematic DEM and how our algorithm treats it. The example consists of a squared flat surrounded on all sides by a wall of cells with higher elevation, which defines a closed depression or pit. On opposite sides of the wall there exist two pour points with lower height than the rest of the wall. The figure corresponds to an interior terrain piece of a larger DEM, but for the sake of simplicity only the flat and the surrounding wall are portrayed.

Through Fig. 4b–d, we simulate a constant inflow of water accessing the depression through the left gap. As the water flows inside, it begins to spill all over the flat until it reaches the opposite wall of the depression. From that moment, the water level *ZW* of the flat gradually increases. After several iterations of our algorithm, in Fig. 4c the water level *ZW* eventually reaches the height of the right pour point and consequently, the water starts to overflow. In successive iterations, more and more water escapes from the pit and resumes its downhill course as depicted in Fig. 4d.

The behaviour of our algorithm has several points in common with previous depression filling methods such as the proposals of Martz and Jong (1988), Jenson and Domingue (1988) and Wang and Liu (2006). The methods of Planchon and Darboux (2001) and Magalhaes et al. (2012) deserve a special mention as they also uses a water flooding technique, although the algorithms and its purpose are different from those of the method described in the present work. The first is a depression filling method for generating an hydrological correct DEM as preprocessing, and the second is a drainage computation method, but as we show in Section 2 it works in a different way by raising the water level from the border of the DEM to compute the flow directions. Also note that in contrast to our approach, all these techniques require additional logic and data structures for detecting depressions, finding pour points or guiding the filling process. Several of the above methods for depression filling introduce artificial flats in the DEM that must be subsequently treated. Our method does not generate flats: as shown in Fig. 4, as long as the water flows through a depression, there exists a gradient in the ZW values that forces the water to keep flowing to the pour point.

An inherent limitation associated to the D8 approaches is their incapability to estimate watercourses with a width greater than one cell. Therefore, it is not possible to detect wide segments of a river network, such as flooded areas or lakes. The flood-based approach of our algorithm allows us to address this limitation, and in contrast to other approaches such as the proposal of Turcotte et al. (2001) is capable of extracting river and lake morphology using only the information provided by the DEM.

Fig. 5 shows a comparison between an actual satellite image of a lake and the corresponding drainage network determined by our algorithm. By comparing both subfigures we clearly see that our method managed to delineate the shape of the lake and its main inlets/outlets in a realistic manner. The same figure also illustrates the ability of the algorithm to model bifurcations and divergent flows. It shows a good example of a divergent flow forming a natural loop around an island in the lake. This loop has been successfully detected by the algorithm, as well as another much smaller one around a tiny island south of the lake.

**Algorithm 1.** *Flooding_Algorithm*(*DEM*).

```
repeat
  accum ⇐0
  for each cell c of the DEM do
  if W(c) > 0 then
      if c is on the border then
        accum⇐accum + W(c)
        W(c)⇐0
      else
        n⇐GetLowerNeighbour(c)
        if ZW(c) > ZW(n) then
          moving_water⇐min(W(c), (ZW(c)−ZW(n))/2)
          accum⇐accum + moving_water
          DA(c)⇐DA(c) + moving_water
          W(c)⇐W(c)−moving_water
          W(n)⇐W(n) + moving_water
        end if
      end if
    end if
  end for
until accum < stop_threshold
result_cells⇐{c/DA(c) > threshold}
return result_cells
```

## 4. Results and discussion

In this section, we will first discuss the quality of the drainage networks provided by our method, as well as the influence of the parameters used by our flooding algorithm to control the

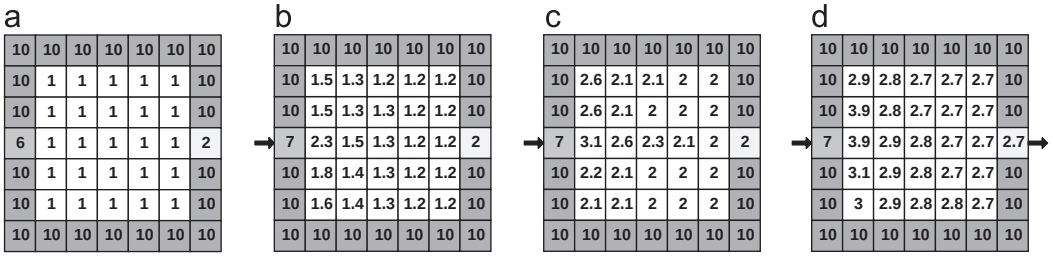


**Fig. 4.** An example of a flat. In (a), the numbers denote the elevation *Z* of the DEM. The flat is bordered by a thick wall illustrated with gray colours. This wall has two apertures, both of them higher than the flat, that is, the flat also constitutes a pit. In the example, we simulate a constant inflow of water through the left aperture, which results in the water spilling out from the flat through the right aperture. In pictures (b)–(d), the numbers denote the *ZW* value of each cell after several iterations of Algorithm 1.

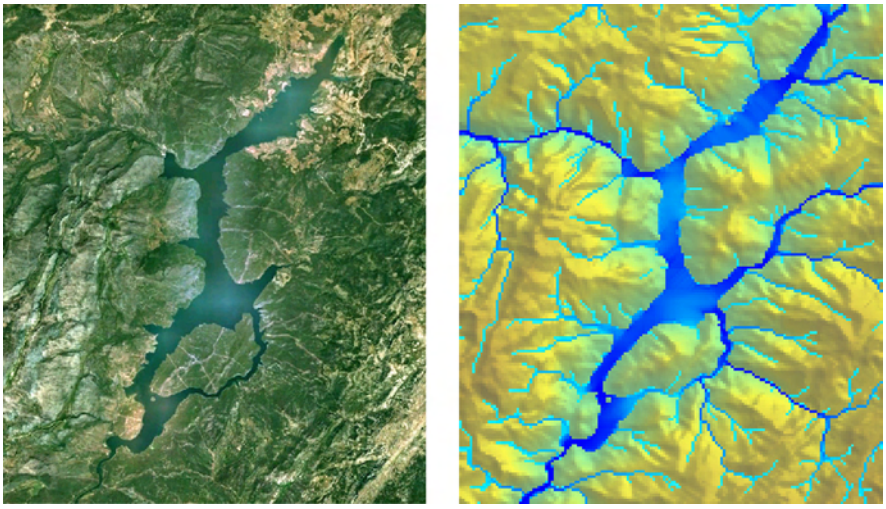

**Fig. 5.** *El Tranco de Beas* reservoir in the *Cazorla, Segura y Las Villas* National Park (Jaén, Spain). Blue shades in the right figure denote the drainage. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

computation. Next, some details about the general performance of our method are also reported and discussed.

### 4.1. Influence of the parameters

The source of elevation data used in this study was the widespread 3 arcsecond resolution (90 m) SRTM3 DEM v2.1 provided by the NASA Jet Propulsion Laboratory. We selected four regions from Spain with distinct terrain features, and the DEMs were minimally processed to fix small gaps in the elevation data by using the average height of the neighbour cells.

Figs. 6a and 7a show the networks obtained by our algorithm using the selected DEMs as provided, that is, without any further preprocessing besides the elimination of voids. Our goal was to evaluate the capacity of our solution to handle challenging DEMs with a high number of artifacts and pits. Figs. 6b and 7b, on the other hand, show the quality of the drainage networks computed by our algorithm after preprocessing the DEMs to fill pits and small depressions. The preprocessing technique we used was a simplification of the method of Planchon and Darboux (2001) that guarantees that for each DEM cell there is a path that leads to the boundary where every cell is higher or the same level as the following. Note that this pit removal technique does not treat flats and therefore, it relies in the ability of our algorithm to handle these areas. Finally, Figs. 6c and 7c show the networks computed by the *r.watershed* module included in GRASS GIS (Neteler et al., 2011) using the MFD8 dispersal algorithm.

We experimentally selected a set of values for the parameters $W$ and $DA$ of our algorithm. Namely, the drainage networks from the unprocessed DEMs shown in the figures were obtained using $W=100$ mm, $DA=20\,000$ mm. For the preprocessed DEMs, we selected $W=10$ mm, $DA=2000$ mm. Finally, *r.watershed* was configured to use a threshold value $DA=200$. We considered that our algorithm converged to a solution when the accumulated height of the water transferred by all the cells in one iteration fell below 1% of the total water introduced in the DEM during the initialization.

The corresponding networks obtained by the three methods are very similar, especially in the upper courses of the rivers. However, the differences between our new algorithm and *r.watershed* become more obvious in the width of the main channels and the lakes. The main channel and meanders of the Ebro river were clearly delineated by our algorithm in Fig. 6 (left) as well as the lakes and insular islands in Fig. 6 (right). These lakes are represented in the DEM as large flat areas, which resulted in linear, artificial channels in the network delineated by *r.watershed*.

Regarding the influence of the parameters of our algorithm, from our experimentation we observed that there exists a linear relation between $W$ and $DA$. Increasing one requires increasing the other in the same proportion in order to achieve similar results. In all tests we kept a relation of 1:200, which means that all DEM cells that receive an amount of flowing water equivalent to the initial water of 200 cells were included in the drainage network. We found that this relation is also consistent with *r.watershed*, which yielded similar networks with a threshold parameter of 200. Nevertheless, the relation between $W$ and $DA$ can be adjusted to control the resulting drainage network. As was pointed out by O'Callaghan and Mark (1984), low threshold $DA$ values can cause wide runoffs of parallel channels with no intervening cells, especially on steep mountainsides. On the contrary, increasing the threshold reduces the amount of tributaries which results in a more outlined network.

We did not find significant differences between the resulting networks obtained with several $W$ values ranging from 1 mm to 100 mm when using depression-less DEMs. The reason for this is that all the water flows along its course downhill without getting trapped in any depression. However terrains with an important part of the surface covered by lakes typically required a higher value of $W$ to prevent a small amount of water from spilling over a large area, which can cause disconnections. In general $W=10$ yielded good results on most of the SRTM DEMs we tested. In contrast the election of the initial $W$ value proved to play an important role in the drainage networks obtained from unprocessed DEMs. This stems from the fact that a considerable volume of water is required for filling the pits and small depressions found along the course of the rivers. This can reduce the network connectivity if the amount of water introduced in the DEM is small. For example, in Fig. 6a (left) it was not possible to completely fill the lake with the available water. The bottom left corner of Fig. 7a (left) depicts another example of disconnection caused by a large pit. In our experiments $W$ was kept approximately one order of magnitude greater in comparison when working with preprocessed SRTM DEMs. When comparing the drainage networks extracted from both cases in Figs. 6 and 7 we observe that, in general, our algorithm managed to achieve a very similar network connectivity in all tested scenarios.

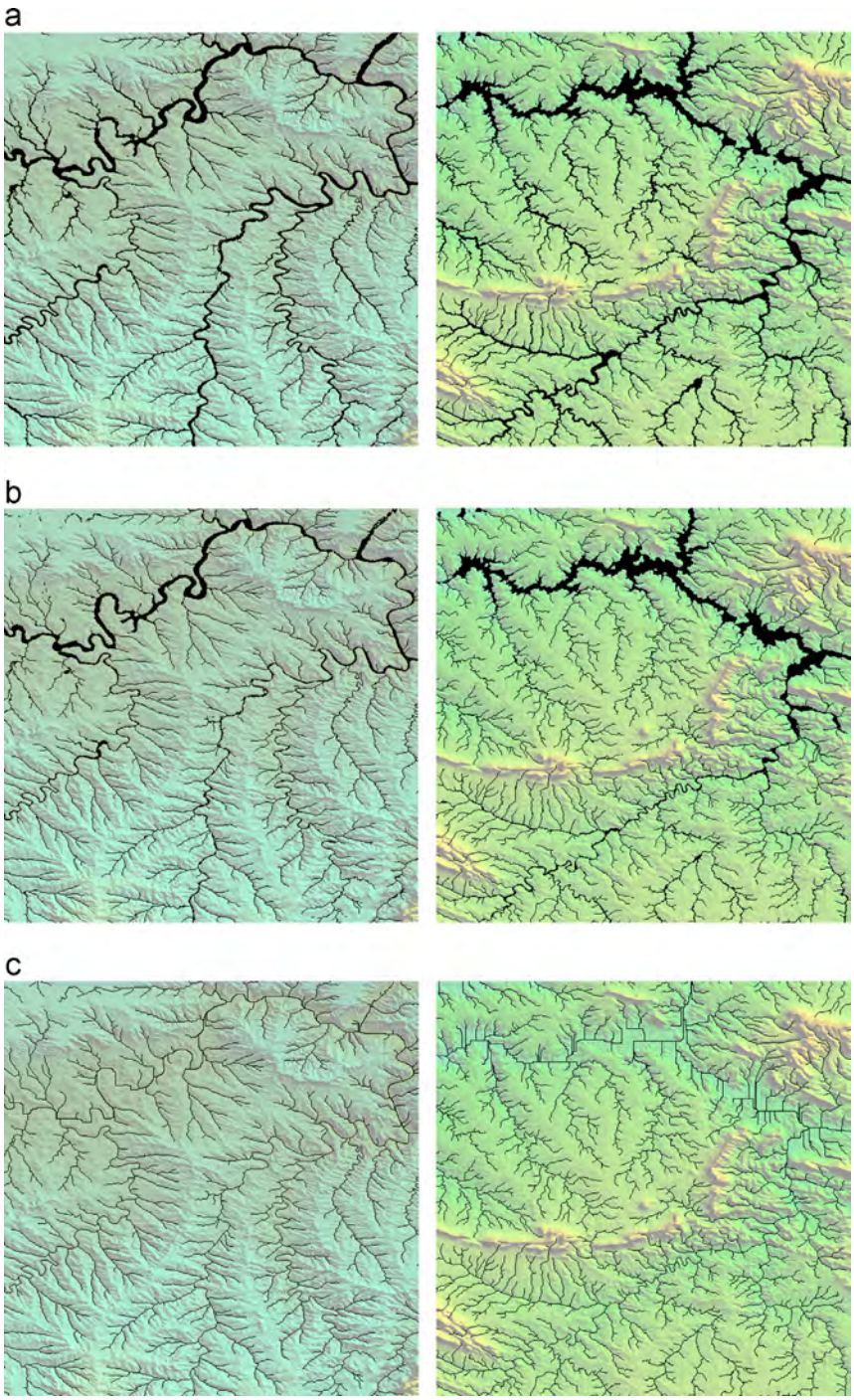


**Fig. 6.** Details of networks extracted in a terrain from: Ebro river in Zaragoza (left) and Zújar river in Badajoz (right). (a) Without preprocessing. $W = 100, DA = 20\,000$. (b) with preprocessing. $W = 10, DA = 2000$ and (c) GRASS (*r.watershed*). $DA = 200$.

### 4.2. Performance

The proposed algorithm was tested on several real life datasets to evaluate its efficiency depending on the DEM size. Our experimentation was carried out on a MacBook Pro equipped with an Intel Core i7 Quad-core processor running at 2.2 GHz with 4 GB of RAM and a Mac OS X Lion 10.7.5. The algorithm was directly implemented in C++ and compiled with the clang $86 \times 64$ bit compiler. The tested implementation is an optimized version of Algorithm 1 that uses a FIFO data structure to avoid unnecessary processing of cells without water. This FIFO is initialized with all the DEM cells, and Algorithm 1 iterates over it instead of the full DEM. Empty cells are removed from this FIFO, and empty cells that receive water are pushed again in the FIFO.

All the DEMs used in this experiment were also extracted from the NASA's SRTM3 v2.1 dataset. In all our experiments, the DEM cells have been initialized with a water layer $W$ of 100 mm. We considered that the algorithm converged to a solution when the accumulated height of the water transferred by all the cells in one iteration fell below 1% of the total water introduced in the DEM during the initialization.

Table 1 shows the performance of our algorithm when applied to DEMs with increasing sizes. From left to right, the table lists the name of the DEM, its size, the number of $1 \times 1$ pits (local minima)

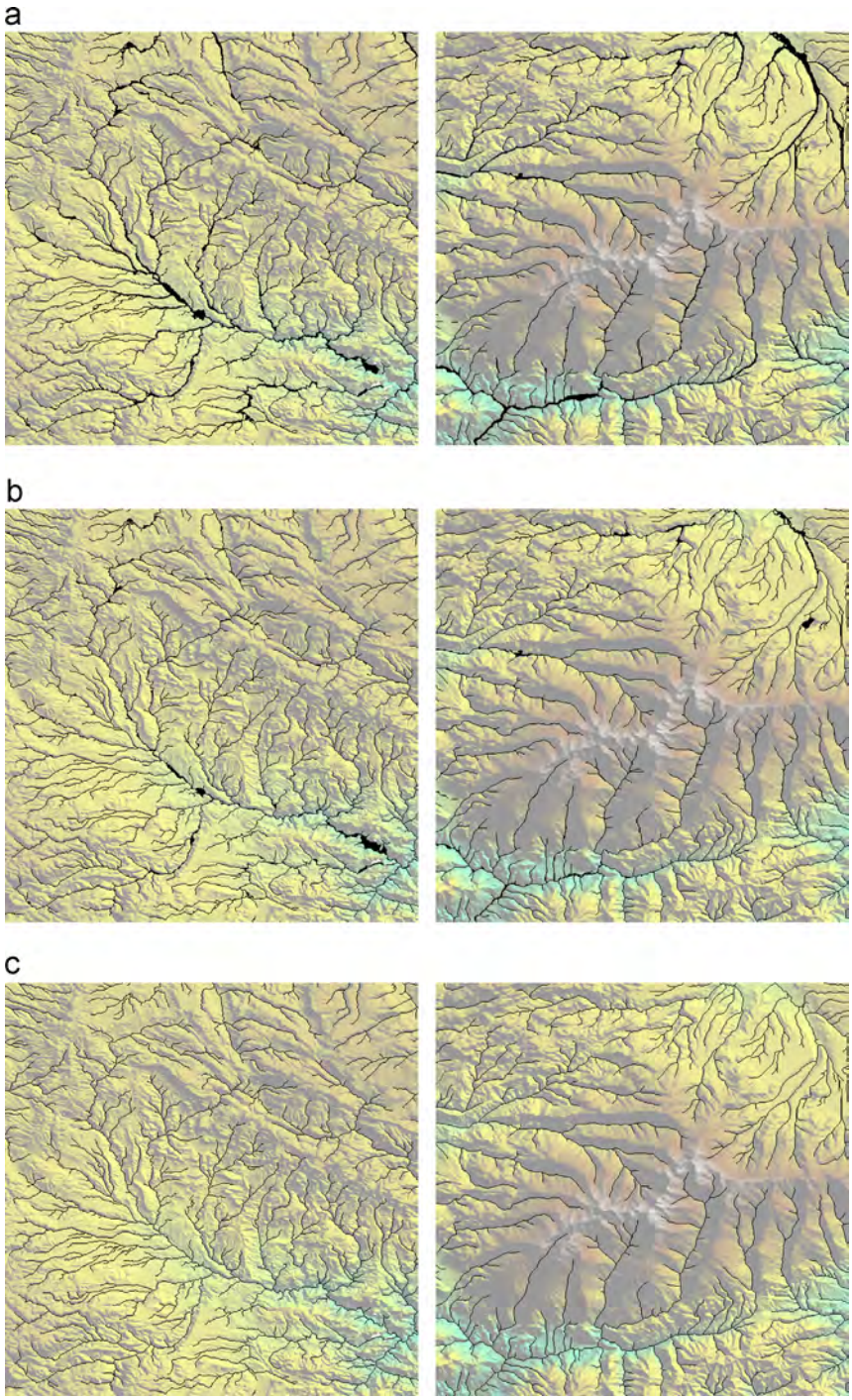


**Fig. 7.** Details of networks extracted in a terrain from: Guadalope river in Teruel (left) and Sierra Nevada in Granada (right). (a) Without preprocessing. $W = 100, DA = 20\,000$, (b) with preprocessing. $W = 10, DA = 2000$ and (c) GRASS (*r.watershed*). $DA = 200$.

**Table 1**
Characteristics of the DEMs used in our experiments and performance of our algorithm.

| Dataset | Size | Number of pits | Number of iterations | Execution time (s) |
|---|---|---|---|---|
| Dataset 1 | $301 \times 301$ | 152 | 613 | 0.056 |
| Dataset 2 | $1201 \times 401$ | 479 | 2105 | 0.768 |
| Dataset 3 | $1201 \times 1201$ | 7092 | 7651 | 20.800 |
| Dataset 4 | $2401 \times 2401$ | 34 014 | 7595 | 118.037 |

it contains, the number of iterations required by the repeat-until loop of Algorithm 1 to converge, and the computation time in seconds.

During our experimentation we observed that most water quickly flows downhill and either abandons the DEM or achieves a stable state within the first few iterations. However, a small

fraction of the water gets accumulated in the lower parts of the terrain and requires a relatively longer time to settle down. Nevertheless, through the analysis of Table 1 we see that our algorithm manages to converge to a solution in a reasonable number of iterations on all the experiments. As expected, the number of cells to process during each iteration directly depends on the size of the DEM. Therefore, execution times increased significantly with the size of the DEMs, ranging from some milliseconds for the smallest dataset to 118 s for the largest one. Nonetheless, we consider that these numbers are satisfactory considering that our method did not require any previous step to process the DEM. Note that this preprocessing part is, in most cases, the most time-consuming step in extracting a drainage network, as was pointed out by Wang and Liu (2006).

## 5. Conclusion and future work

In this paper we have presented a new approach for extracting the drainage network from a digital elevation model based on a novel flooding strategy that is inspired by the natural behaviour of water (e.g. the law of communicating vessels). The resulting algorithm is elegant, simple and capable of overcoming most DEM artifacts that affect other methods based on the modeling of the water flow over terrain. The proposed solution can be categorized as a multiple flow method, and is also capable of identifying wide watercourses, lakes and divergent flows in a realistic way.

Several aspects of the flooding algorithm remain as open problems or future work. When working with unprocessed DEMS it would be desirable to estimate the depth of the initial water layer by analyzing the number and size of pits and flats in the DEM, together with other relevant morphologic features. Alternatively, an adaptive flooding depending on the area of the DEM could be considered. For instance if it is detected that the water flow is interrupted in a pit, in subsequent iterations additional water could be added to the pit cells until the water flow is resumed. We also believe that a multicore or GPU-based implementation of the algorithm could reduce computation times dramatically. Compared to a classic D8 algorithm, the relatively higher complexity of the water transfer simulation and the fact that the number of cells that have to be processed after each iteration (i.e., with $W > 0$) decreases at a slower rate suggest a much higher benefit of a parallel implementation (see Ortega and Rueda, 2010).

The source code of the drainage extraction algorithm described in this paper can be downloaded from: https://github.com/cageo/Rueda-2013.

## References


Barnes, R., Lehman, C., Mulla, D. An efficient assignment of drainage direction over at surfaces in raster digital elevation models. Computers & Geosciences, in press.

Barták, V., 2009. How to extract river networks and catchment boundaries from DEM: a review of digital terrain analysis techniques. Journal of Landscape Studies 2, 57–68.

Douglas, D.H., 1986. Experiments to locate ridges and channels to create a new type of digital elevation model. Cartographica 23, 29–61.

Ehlschlaeger, C., 1989. Using the A* search algorithm to develop hydrologic models from digital elevation data, in: Proceedings of International Geographic Information Systems (IGIS) Symposium, pp. 275–281.

Freeman, T.G., 1991. Calculating catchment area with divergent flow based on a regular grid. Computers & Geosciences 17, 413–422.

Garbrecht, J., Martz, L.W., 1997. The assignment of drainage direction over flat surfaces in raster digital elevation models. Journal of Hydrology 193, 1–4.

Jenson, S.K., Domingue, J.O., 1988. Extracting topographic structure from digital elevation data for geographic information system analysis. Photogrammetric Engineering and Remote Sensing 54, 1593–1600.

Jones, R., 2002. Algorithms for using a DEM for mapping catchment areas of stream sediment samples. Computers & Geosciences 28, 1051–1060.

Magalhaes, S.V.G., Andrade, M.V.A., Franklin, W.R., Pena, G.C., 2012. A new method for computing the drainage network based on raising the level of an ocean surrounding the terrain, in: Proceedings of 15th AGILE International Conference on Geographic Information Science, Avignon (France), pp. 391–407.

Martz, L.W., Jong, E.D., 1988. Catch: a fortran program for measuring catchment area from digital elevation models. Computers & Geosciences 14, 627–640.

Meisels, A., Raizman, S., Karnieli, A., 1995. Skeletonizing a DEM into a drainage network. Computers & Geosciences 21, 187–196.

Metz, M., Mitasova, H., Harmon, R.S., 2011. Efficient extraction of drainage networks from massive, radar-based elevation models with least cost path search. Hydrology and Earth System Sciences 15, 667–678.

Neteler, M., Bowman, M., Landa, M., Metz, M., 2011. GRASS GIS: a multi-purpose Open Source GIS. Environmental Modelling & Software 31, 124–130.

O'Callaghan, J.F., Mark, D.M., 1984. The extraction of drainage networks from digital elevation data. Computer Vision, Graphics, and Image Processing, 323–344.

Ortega, L., Rueda, A., 2010. Parallel drainage network computation on CUDA. Computers & Geosciences 36, 171–178.

Peucker, T.K., Douglas, D.H., 1975. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. Computer Graphics and Image Processing 4, 375–387.

Planchon, O., Darboux, F., 2001. A fast, simple and versatile algorithm to fill the depressions of digital elevation models. Catena 46, 159–176.

Rieger, W., 1998. A phenomenon-based approach to upslope contributing area and depressions in DEMs. Hydrological Processes 12, 857–872.

Seibert, J., Brian, M., 2007. A new triangular multiple flow-direction algorithm for computing upslope areas from gridded digital elevation models. Water Resources Research 43, 1–8.

Tarboton, D.G., 1997. A new method for the determination of flow directions and upslope areas in grid digital elevation models. Water Resources Research 33, 309–319.

Tribe, A., 1992. Automated recognition of valley lines and drainage networks from grid digital elevation models: a review and a new method. Journal of Hydrology 139, 263–293.

Turcotte, R., Fortin, J.P., Rousseau, A., Massicotte, S., Villeneuve, J.P., 2001. Determination of the drainage structure of a watershed using a digital elevation model and a digital river and lake network. Journal of Hydrology 240, 225–242.

Wang, L., Liu, H., 2006. An efficient method for identifying and filling surface depressions in digital elevation models for hydrologic analysis and modelling. International Journal of Geographical Information Science 20, 193–213.