Scalable Algorithms for Large High-Resolution Terrain Data *

Thomas Mølhave
Computer Science
Duke University
thomasm@cs.duke.edu

Pankaj K. Agarwal
Computer Science
Duke University
pankaj@cs.duke.edu

Lars Arge
MADALGO
Aarhus University
large@madalgo.au.dk

Morten Revsbæk
MADALGO
Aarhus University
mrevs@madalgo.au.dk

ABSTRACT

In this paper we demonstrate that the technology required to perform typical GIS computations on very large high-resolution terrain models has matured enough to be ready for use by practitioners. We also demonstrate the impact that high-resolution data has on common problems. To our knowledge, some of the computations we present have never before been carried out by standard desktop computers on data sets of comparable size.

Categories and Subject Descriptors: D.2 [Software]: Software Engineering; F.2.2 [Nonnumerical Algorithms and Problems]: Geometrical problems and computations; H.2.8 [Database Management]: Database Applications—*Data Mining, Image Databases, Spatial Databases and GIS*

General Terms: Performance, Algorithms

Keywords: LIDAR, Massive data, GIS

1 Introduction

The revolution in sensing and mapping technologies is providing an unprecedented opportunity to characterize and understand the earth's surface, its dynamics, and its properties. For instance, second generation airborne LIDAR technology can map the earth's surface at a 15-20cm horizontal resolution, and the future generation of LIDAR scanners are expected to generate high-resolution maps of other planets. Capitalizing on these opportunities and transforming these massive amounts of topographic data into useful information for vastly different types of users requires solving several challenging algorithmic problems. GIS, geometric computing and other disciplines have made great strides, during the last few years, in providing theoretical insights, algorithmic tools, and software for meeting many of the challenges that arise when these large data sets have to be processed.

*Work in this paper was supported by ARO grant W911NF-04-1-0278. Pankaj K. Agarwal and Thomas Mølhave are also supported by NSF under grants CCR-00-86013, CCR-02-04118, and DEB-04-25465, and by a grant from the U.S.–Israel Binational Science Foundation. Lars Arge and Morten Revsbæk are also supported by a NABIIT grant from the Danish Strategic Research Council and by MADALGO - Center for Massive Data Algorithmics - a Center of the Danish National Research Foundation.

COM.Geo 2010 June 21-23, 2010, Washington, DC, USA.

Figure 1. The coast line of Denmark, the main study area for the experiments in this paper.

A major challenge is the sheer size of the gathered topographic data which is exposing serious scalability problems with existing GIS systems. The main reason for these problems is that algorithms in current systems often assume the data to fit in the main memory of the computer, which is not the case for the large data sets provided by modern LIDAR scanners. A main memory access is around 10^6 times faster than a disk access, and programs that do not use the disk efficiently are essentially slowed done by that factor. Thus it is key to use algorithms try to minimize disk accesses when handling massive data sets. Developing such Input/Output-efficient (or just *I/O-efficient*) algorithms have been a flourishing research area for the past many years.

There is a number of ways traditional GIS applications attempt to deal with the massive terrain elevation data sets. The simplest common technique is to *thin* the point cloud by more or less arbitrarily discarding a significant fraction of the points. A similar form of thinning is often applied to gridded terrain models where neighboring grid cells are averaged (or discarded) to produce a grid with a much larger cells size and correspondingly lower spatial fidelity. These methods, although obviously very effective at reducing the size of the point cloud or grid, are not ideal since important topographic features will be lost in such an operation. This can significantly reduce the accuracy and effectiveness of many topographic analysis techniques. Finding a way to drastically reduce the data set without invariably creating problems with non-trivial computations would in many cases likely not be much easier than solving the original problem directly.

Another practical and popular method is to bin the data into *tiles* of some fixed low size, transforming a very big data set into a list of

manageable units. This can be done directly on the original point cloud, but also on gridded terrain models. However this approach is not feasible for a lot of non-trivial computations where interactions across tile boundaries significantly increase the computational complexity of the method.

There has also been a growing amount of research into theoretically efficient algorithms for solving GIS-related problems for massive data sets [1, 2, 3, 10, 11, 19]. Many of the algorithmic developments have been mostly theoretical or their implementations have not been ready to be presented to a wider non-technical audience in a user-friendly manner. In 2007 some of the present authors presented a preliminary version of a tool, TerraSTREAM, that specializes in handling large terrain models [4] for solving a range of problems.

However, despite the recent research, most popular GIS software packages remain unable to handle truly large elevation models. Thus users usually resort to crude approximations like the ones mentioned above, and do not routinely work with the full detail provided by their data sets. This decreases the value of the derived products and is unfortunate given the considerable investment a large-scale LIDAR survey represents.

The goal of our research is to make theoretical work practical and thereby making practical algorithms more grounded theoretically than the existing algorithms. This enables us to prove guarantees on the performance of the practical algorithms we propose. In this paper we demonstrate the value of these algorithms to the GIS community by demonstrating that running non-trivial computations on country-sized high-resolution terrain models is now feasible on standard desktop computers and without any complicated trickery required by the user. We also demonstrate why it is important to use the full high-resolution data. Our main examples in this paper will be two hydrology-related problems; flow modeling and flood risk mapping

The main data set used in this paper is a massive LIDAR point cloud of the country of Denmark (with 26 billion points). It is so big that most users are unable to perform non-trivial computations on the entire model at once. To our knowledge most of the computations in this paper have never before been performed on data sets of comparable size using standard desktop computers.

In Section 2 we will discuss the importance of the high-resolution data sets by showing examples of how flow modeling and flood mapping computations are affected by changes in topography. We will then, in Sections 3 and 4, give an overview the software and the process we can use to perform computations on very large data sets. Finally in Section 5 we will give more detail on the experiments and on how we visualize the results.

2 Importance of High-Resolution Data sets

In this section we illustrate our system by describing algorithms for two widely used hydrological problems. These problems illustrate how the high spatial fidelity of modern data sets improve our ability to reason about the terrain.

A very common operation on terrain models is modeling the flow of water on the surface of the terrain. Such modeling can reveal how water accumulates into creeks that converge and form streams, and later rivers, and can be used to extract the watersheds of the terrain. Intuitively, a *river network* is a collection of paths that indicate where large amounts of water, or rivers, are likely to flow on the terrain.

Another popular product of elevation models is the computation of flood risk information. Computing highly accurate flood risk information is a very complex task and involves many types of fine-grained information about the study area. However, with high-resolution terrain models that contain geometrically small but important features (suck as dikes), it is possible to get good initial flood risk estimates using only topography and ignoring the effect of e.g. sewers and the groundwater. In this type of computation, water rises from the oceans but is blocked by the terrain as represented by the model, an example can be seen in Figure 2.

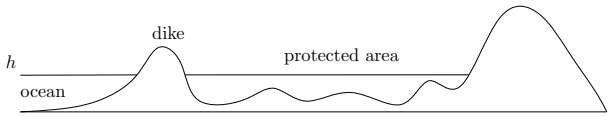


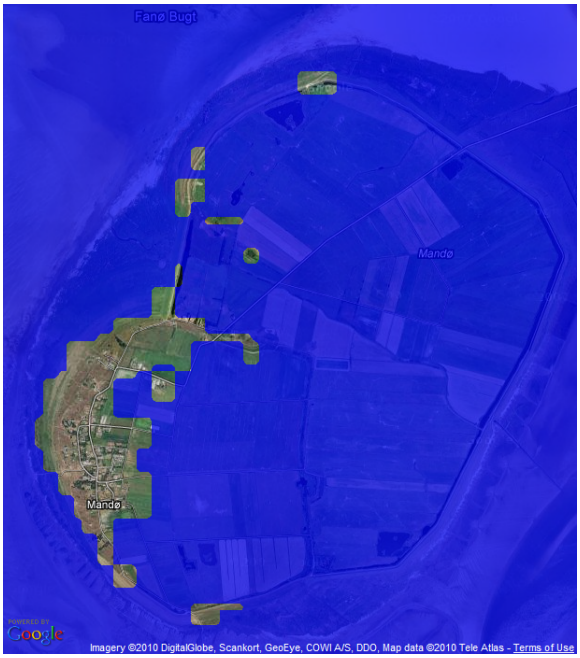
Figure 2. Flood map computation, the protected area behind the dike is not flooded when the ocean level reaches h.

It is important to note that both the flood risk estimation and the flow modeling problems are "non-local" in the sense that a small local change in topography can affect the result of the computation for the entire terrain model. It is this property that makes the problems hard to solve for large data sets, and is also why the quality and resolution of the elevation model is vital.

Our main study area for this paper is the country of Denmark, which is one of the growing list of countries that have performed a nationwide high-resolution LIDAR scanning. Through a company (COWI A/S) we have received a complete high-resolution LIDAR point cloud for Denmark. The point cloud is stored in 14063 LAS-files, each containing the points covering a four square kilometer area. The files themselves takes up about 1.7 terabytes. There is a total of 25,887,357,931 — nearly 26 billion bare earth points in the data set, this is a bit less than a one point square meter for the entire country. We constructed a triangulation of the entire point Denmark point cloud and used this triangulation to create a high-resolution country-wide grid model with a 2m horizontal resolution. We then algorithmically extracted river networks and computed detailed flood risk assessments. Before computing river networks we prepared the grid by removing spurious sinks in the terrain. We also used the data set generated by the Shuttle Radar Topography Mission (SRTM) [9]. The SRTM data set provides a 90m (at the equator) grid that covers most of the inhabited parts of the world. Using the same software tools we were able to compute flood risk mapping and river networks for the entire SRTM grid.

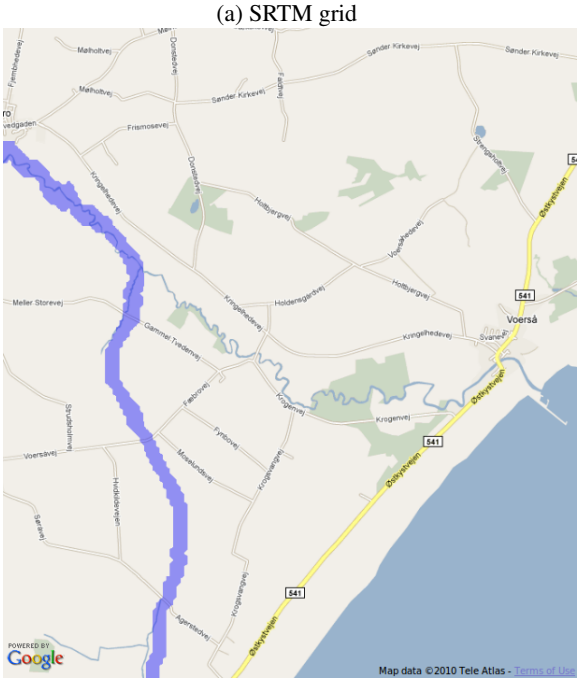
Figure 3(a,b) shows the result of the flood risk mapping for the island of "Mandø" in the Wadden-Sea of the west-coast of Denmark. This tiny island, its area is approximately seven square kilometers, has been hit by storm floods on multiple occasions. As a result of these incidents, the island has an approximately five meter tall perimeter dike which protects it from the sea. Due to the small horizontal extent of the perimeter dike, this feature is not present in the SRTM data, or in most "mid"-resolution data sets. Thus, when the flooding computation described above is performed for a water level of 2 meter, it looks as if most of the island will be under water. This is visualized in Figure 3(a). The same computation performed on the 2m-grid, shown in Figure 3(b) correctly finds that the dikes present in the terrain model block the water from entering the lower-lying areas inside the perimeter. This kind of example is easy to find, and there are many examples, including near major cities, where dikes have been built. The flooding simulations relevance as a good resource for doing an initial flood risk assessment is severely impeded if these dikes and similar natural features are not present in the elevation model.

Like the flood mapping example, the omission of a relatively small feature in a low-resolution data set can have a global impact

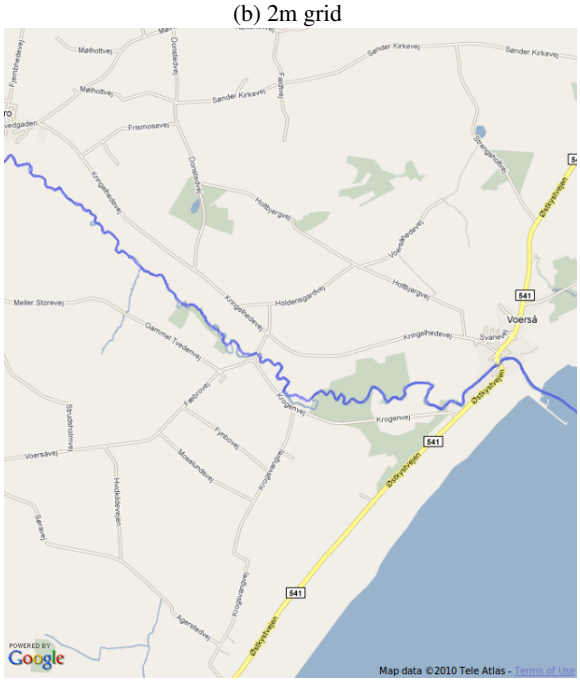


(a) SRTM grid

(b) 2m grid



(c) SRTM grid



(d) 2m grid

Figure 3. (a,b) A flood risk mapping of the island of Mandø in Denmark, using the SRTM (a) and high-resolution grid (b). (c,d) A river network showing cells with an upstream area greater than $50\mathrm{km}^2$ in blue. The first figure (c) shows the river network on the SRTM model, the other (d) shows the river network computed on the high-resolution grid. All the figures are screenshots of our custom map application building on Google Maps.

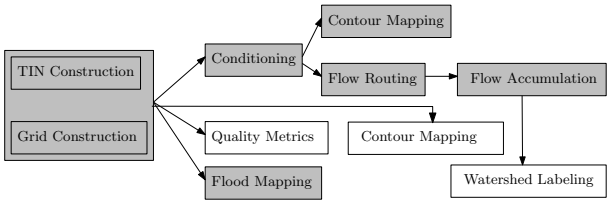


Figure 4. The modules of TerraSTREAM. The modules marked in gray are used in this paper to demonstrate the pipeline and are described in more detail in Section 4.

on the computed river network. An example of this is provided in Figure 3(c,d) which shows a section of the north-eastern part of the main Danish peninsula of Jutland. The networks derived from the SRTM grid and the high-resolution 2m terrain model diverge significantly at a point in the terrain. At the point of divergence a small drainage ditch is merging with the river. The ditch borders the fields to south-west of the divergence and ensures that the excess water from the fields make it to the ocean via the river. In the computation from the SRTM data set the small drainage ditch is marked as the main river and the remainder of the original river becomes a small tributary to this river. This means that the entire watershed upstream of the point of divergence is routed in a wrong direction and this makes the result misleading for most of the area downstream of this point.

3 TerraSTREAM

We have developed a scalable solution[1] that consists of a pipeline of components, where each component uses I/O-efficient algorithms, called TerraSTREAM. Thus, each algorithm in the pipeline scales to massive data sets. TerraSTREAM consists of a number of different modules which can be divided into two main components:

DEM construction: This component constructs triangular irregular network (TIN) and grid DEMs from a (LIDAR) point cloud. It checks the quality of a grid DEM by analyzing the distribution of the point cloud. Finally it also contains a module to construct contour DEMs.

Hydrology: The hydrology component computes flow routing and upstream area contributions from which river networks can be derived. It also partition the terrain into watersheds and perform a simple flooding simulation. Finally it also contains a module to "hydrologically" condition a terrain to make it more suitable for the flow computations, by removing spurious minima.

We will discuss in more detail (Section 4) some of the modules of these two components that are relevant for the results described here. Figure 4 illustrates the overall structure of the pipeline and the outputs of its several stages. The individual stages build upon new I/O-efficient algorithms as well as extensions to a number of previously developed algorithms. In addition, a considerable amount of engineering effort is devoted to making TerraSTREAM efficient and practical. Where it makes sense, the modules of TerraSTREAM works on both TIN and grid DEMs.

The most important feature of all the modules is their scalability. The user supplies an upper bound on the amount of memory that can be used during the computation and TerraSTREAM works with the disk in an efficient way to ensure this limit is not violated. TerraSTREAM can efficiently handle arbitrarily large data sets, regardless of the amount of memory available.

TerraSTREAM has now matured to a point where it is being used by a large number of users worldwide, many of them with little background in algorithms or programming. It includes extensions to some of the most popular commercial GIS software (ArcGIS and MapInfo), as well as a simple standalone application that does not require any other GIS software. We refer to [15, 4, 8] for more information from an application and algorithmic point of view. In this paper we will only briefly describe the actual stages of the pipeline used in the experiments.

[1]Project site: `http://madalgo.au.dk/Trac-TerraSTREAM`

4 Algorithms

In this section we begin by giving a brief overview of the DEM construction component and then discuss the hydrology component in more detail. High-resolution data and the scalability of our system are crucial for the hydrology component.

4.1 DEM construction

A common approach for constructing a grid DEM, of a user-specified cell size from a point cloud is to compute the height value at each grid point by using an interpolation or approximation scheme (refer to e.g., [13] and the references therein). This method is, however, impractical even for a point cloud of a few thousand points because of the computational complexity of solving large systems of linear equations. Using I/O-efficient techniques [1], TerraSTREAM breaks the large systems into smaller pieces and extends many different interpolation routines to these smaller pieces.

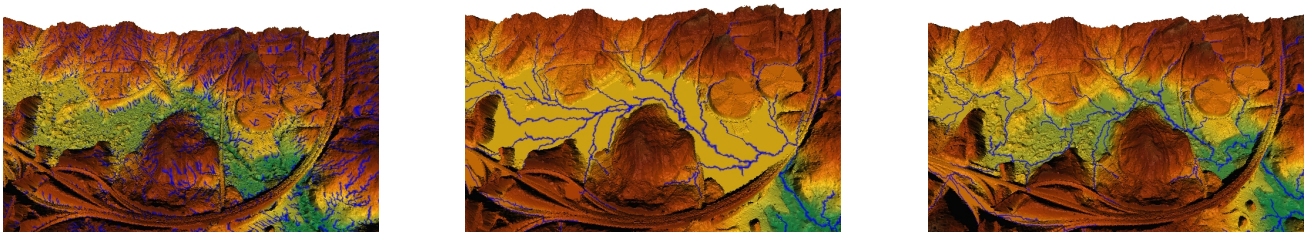
The benefits of sophisticated interpolation schemes (e.g. [14]) for high-resolution data sets are often out-weighted by their high time-complexity. TerraSTREAM therefore also provides a triangulation based linear interpolation scheme [2].

4.2 Hydrological conditioning

Most flow modeling algorithms assumes water flows downhill until it reaches a local minimum or *sink*. In practice however, local minima in DEMs fall into two primary categories; *significant* and *insignificant*, or spurious, sinks. Significant sinks correspond to large real geographic features such as quarries, sinkholes or large natural *closed* basins with no drainage outlet. The insignificant sinks may be due to noise in the input data or correspond to small natural features that flood easily. When modeling water flow these insignificant sinks impede flow and result in artificially disconnected hydrological networks. We "hydrologically condition" the DEM to solve this problem, by removing insignificant sinks, while preserving significant sinks.

Many popular hydrological conditioning algorithms remove *all* sinks using a so-called *flooding* approach, which simulates the process of uniformly pouring water on the terrain until a steady-state is reached. A weakness of this approach is that it removes even significant sinks. See Figures 5(a) and (b). We use a *partial flooding* algorithm, based on *topological persistence* [7, 6], that detects and removes only insignificant sinks, as indicated in Figure 5(c). This leads to a more realistic flow network. We briefly describe topological persistence and then present our algorithm.

Topological persistence. In the context of a terrain T represented by a grid, topological persistence [6, 7] matches each local minimum (sink) cell v of T to a higher "saddle" cell w (see [5] for the precise definition of a saddle) and assigns a *persistence* value, denoted by $\pi(v)$, to v; $\pi(v)$ is defined to be the difference in the



(a) (b) (c)

Figure 5. (a) Original terrain. (b) Terrain flooded with $\tau = \infty$. (c) Terrain partially flooded with persistence threshold $\tau = 10$.

heights of v and w, i.e., $\pi(v) = h(w) - h(v)$ [6]. The persistence $\pi(v)$ denotes the *significance* of the sink v. Intuitively, the saddle cell w is a cell at which two distinct connected components of the portion of T lying strictly below w merge. Each connected component is represented by the lowest vertex in the component. Suppose v is the highest representative of the two connected components merged by w and let u denote the representative of the other component.

Partial flooding. We use topological persistence as a measure of the significance of a sink. Given a user-specified threshold τ, we declare all sinks with persistence less than τ to be the insignificant sinks and remove all such sinks using a partial flooding method described below. The user can change the threshold to control the smallest feature size to be preserved.

Let $\zeta_1, \ldots, \zeta_k$ be the significant sinks in T. We construct a graph on T by connecting each grid vertex to its neighbors. Let the *height of a path* in this graph be the height of the highest vertex on the path, and let the *raise elevation* of a vertex v be the minimum height of all paths from v to ζ_i for any $1 \leq i \leq k$. In *partial flooding*, we change the height of each vertex to its raise elevation. Partial flooding produces a terrain containing only significant sinks whose persistence value is greater than τ. Note that if $\tau = \infty$, partial flooding is the same as the original definition of flooding. Thus, partial flooding is a tunable way to condition the terrain for the purpose of flow modeling.

We also have the ability to compute the *volume* and *area* of the sinks in the terrain [18]. This allows for more flexibility in defining the significance of a sink so that e.g. sinks with high volume but low persistence ("height") are not marked as insignificant.

4.3 Flow modeling

In the third stage of our experiment we model the flow of water on the hydrologically conditioned DEM. It consists of two phases. The first *flow-routing* phase, we compute a *flow direction* for each cell in the grid that intuitively indicates the direction water will flow on that particular cell. In the second *flow-accumulation* phase, we intuitively compute the area of all the cells upstream of each cell. These flow-accumulation can then be used to derive a river network.

Flow routing. The flow-routing phase computes for each cell, c, a list of those of its neighbors that water reaching c can flow to. These lists are constructed by looking at c and its neighbors in the grid and then applying a *flow-direction* model. We have implemented two popular flow-direction models:

- *Single-flow-direction* (SFD) model: The water for each cell flows to the steepest-descent down-slope neighbor.
- *Multi-flow-directions* (MFD) model: The water for each cell flows to all down-slope neighbors.

Several other flow-direction models have also been proposed (e.g., [21]), and most of them can be incorporated in TerraSTREAM.

Flat areas. The traditional flow-direction models are only defined for cells with at least one down-slope neighbor. However, realistic terrains can have large *flat areas* of same-height cells. Flat areas can be natural plateaus in the terrain, or they can appear as by-products of the hydrological conditioning stage. If a cell on a flat area has a down-slope neighbor it is called a *spill point*. When routing flow on flat areas we distinguish between flat areas that have at least one spill point and those that do not; in the first case water should be able to flow out of the flat area through the spill points, while in the second case water is simply absorbed into the extended sink.

In many early flat area routing approaches flow directions were assigned in a simple way such that each cell v was assigned a flow direction to the adjacent neighbor that was along the shortest Euclidean path along grid edges from v to the closest spill point. However, these approaches are not hydrologically realistic and tend to create many parallel flow lines [22]. TerraSTREAM circumvents this problem by using a more sophisticated method that uses geodesic distance on T, based on the approach described in [20].

Flow accumulation. Given a grid in which individual cells have been assigned flow directions, the flow accumulation [17] phase computes the amount of water that reaches each grid cell of T. More precisely, each cell c is assigned some initial flow; c then receives incoming flow from up-slope neighbors and distributes all incoming and initial flow to one or more of its down-slope neighbors. The flow accumulation of c is the sum of its initial flow and incoming flow from up-slope neighbors. Given the flow accumulations for all vertices, we can extract *river networks* [17] simply by extracting edges incident to vertices whose flow accumulation exceeds a given threshold.

4.4 Flood mapping

As discussed in Section 2 computing highly accurate flood risk information is a very complex task and involves many types of fine-grained information about the study area. However, with modern high-resolution DEMs that contain small but important features (e.g. dikes), it is possible to get good initial flood risk estimates using only the elevation information, and ignoring the effect of sewers and the groundwater.

Given a flood height h, a simple way of computing the flooded area is to mark all vertices of elevation less than h as flooded. This naive approach is problematic because it ignores the effect of dikes

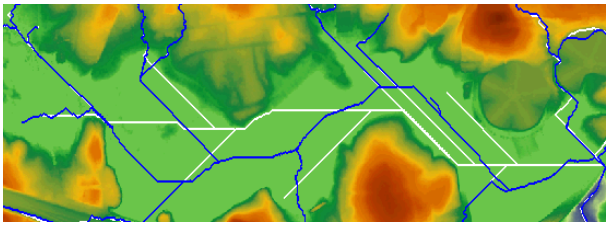


Figure 6. Comparison of routing methods on a flat area with a single spill point on the right. Rivers indicated in white were extracted by using the smallest Euclidean distance. Blue (black) river lines were computed using the Soille et al. approach

and wrongly marks many depressions as being flooded; see Figure 2 for an example. TerraSTREAM uses a more sophisticated approach in which a cell is only marked as flooded if there is a path from that cell to the flooding source (the ocean in most cases) that never passes through a point that is higher than the new water level. This ensures that cells behind dikes or other terrain features are not flooded until the water passes over the feature. Figure 3(a,b) was generated based on flood maps computed in this way.

5 Experiments & Visualization

This section describes different computations that we have performed on the data sets. No manual setup or tweaking to the data set, hardware or software were performed, showing that the whole sequence of operations were performed without any manual intervention. We used a computer with an Intel Core 2 Duo processor, 4 gigabytes of memory and 3TB in 3 standard 7.200RPM disk drives. An external low-cost E-SATA raid enclosure holding 5 1.5TB 7.2000RPM drives was used to store the temporary files required by our tools. The total computation times are shown in Table 1. The running times can be improved significantly by tuning the disk system.

The main data set we use is the large point cloud of Denmark, but to illustrate the scalability of the system we also present the running times of our tools on two different data sets, the SRTM grid [9] and a portion of the the new so-called ASTER GDEM grid [16].

For the Denmark data set, the first step is to compute a grid from the nearly 26 billion bare earth points in the given point cloud. We used the module of TerraSTREAM that first computes one big Delaunay triangulation of the entire point cloud and then interpolates linearly across individual triangles to get the elevation value for each grid cell. The resulting grid has a horizontal resolution of 2m and the raw uncompressed raster file in single precision floating point takes up about 10^{11} bytes. All this was accomplished by one invocation of the grid-construction module of TerraSTREAM.

The SRTM grid is a near-worldwide data set from 60° North to 56° South that was gathered during an 11-day mission by the Space Shuttle Endeavour in 2000. The spatial resolution of the SRTM data is 3 arc-seconds or about 90 meters at the equator. The Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Global Digital Elevation Model (GDEM) [16] was released in the summer of 2009 by NASA and Japan's Ministry of Economy, Trade and industry (METI). The ASTER GDEM is a massive grid that covers 99 percent of the land mass on earth from 83° North to 83° South latitude with elevation points measured every 1 arc-second (30 meters at the equator). We downloaded a part of the US East Coast (from 30° to 44°N and 70°-89°W) to get a grid with roughly $3.9\,10^9$ cells and ran this through the same set of modules to illustrate the performance of the tools on a smaller data set.

For the purpose of computing the flow accumulation grids we used the hydrological condition module to remove all sinks. This gives the best idea of the worst-case run-time since this is the most time-consuming option. Following this we computed the flow routing using the SFD routing model and finally computed the flow accumulation values. As mentioned previously, the running times can be seen in Table 1.

To better visualize and communicate the results of large scale computations we have developed a Google Maps application combining a simple front end with custom-developed optimized server for serving the data for the map itself. Simple sliders allows the user to select the water height and the flow accumulation thresholds used to define the river network dynamically. The data is stored in a generic format by the tile server and based on the value of the sliders, the information is transformed on the fly to create the overlay tiles. Figure 3 was generated by taking screen shots of the map application (it works better "live" than on text[2]).

6 Conclusion

We have demonstrated the significance of high-resolution data sets by giving a few simple examples, and argued why they should be used as they become available. Additionally, we demonstrated on concrete data sets how sophisticated algorithms, such as those implemented in TerraSTREAM, make it possible to work with these large terrains without significant deterioration in performance as data sizes increase.

There are several interesting directions that we are currently pursuing. One line of work focuses on computing a hierarchical DEM that preserves derived features, e.g. river networks. This will enable us to compute desired features at desired resolution by analyzing the terrain at an appropriate level of hierarchy. Second, we are adding more sophisticated modules to TerraSTREAM and make them scalable, e.g. some of the hydrological analysis modules available in the the HydroSHEDS project [12]. This will enable TerraSTREAM to perform computations similar to HydroSHEDS but on much larger and higher resolution data sets, e.g. ASTER data set.

Acknowledgments

We thank Andrew Danner, Peter Hachenberger, Henrik Blunck, Kasper Dalgaard Larsen, Thor Prentow, & Adam Ehlers Nyholm Thomsen for their contributions to TerraSTREAM and Jakob Truelsen for contributions to the tile server. We thank COWI A/S for giving accesss to their Denmark dataset NASA for making the SRTM data set available, and NASA and Japan's Ministry of Economy, Trade and industry for making the ASTER GDEM available.

References

[1] P. K. Agarwal, L. Arge, and A. Danner. From point cloud to grid DEM: A scalable approach. In A. Riedl, W. Kainz, and G. Elmes, editors, *Progress in Spatial Data Handling. 12th International Symposium on Spatial Data Handling*, pages 771–788. Springer-Verlag, 2006.

[2] P. K. Agarwal, L. Arge, and K. Yi. I/O-efficient construction of constrained Delaunay triangulations. In *Proc. European Symposium on Algorithms*, pages 355–366, 2005.

[2]Try it here: `http://madalgo.au.dk/~thomasm/floodmaps`

Data set	Cells (billions)	Model Construction	Hydrological Conditioning	Flow Routing	Flow Accumulation
Denmark	26.8 (19.3)	3 days 16 hours	1 day 17 hours	12 hours	23 hours
SRTM	60.6 (20.8)	—	2 days 3 hours	1 day 4 hours	1 day 8 hours
ASTER	3.9 (2.9)	—	3 hours 22 min	2 hours 7 min	2 hours 41 min

Table 1. Running times for the data sets. The number of cells is given as the total number of cells in the grid, and the number of cells that contain real data (not NODATA). The SRTM and ASTER data sets are provided as grids, so we did not have to construct the model.

[3] A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, and E. Ramos. Randomized external-memory algorithms for some geometric problems. *International Journal of Computational Geometry & Applications*, 11(3):305–337, June 2001.

[4] A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitasova. TerraStream: from elevation data to watershed hierarchies. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[5] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, England, 2001.

[6] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. ACM Sympos. Comput. Geom.*, pages 70–79, 2001.

[7] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proc. IEEE Sympos. Found. Comput. Sci.*, pages 454–463, 2000.

[8] J. M. Eshøj, P. K. Bøcher, J.-C. Svenning, T. Mølhave, and L. Arge. Impacts of 21st century sea-level rise on a major city (Aarhus, Denmark) âĂŞ an assessment based on fine-resolution digital topography and a new flooding algorithm. *IOP Conf. Series: Earth and Environmental Science*, 8:12—22, 2009.

[9] T. G. Farr, P. A. Rosen, E. Caro, R. Crippen, R. Duren, S. Hensley, M. Kobrick, M. Paller, E. Rodriguez, L. Roth, D. Seal, S. Shaffer, J. Shimada, J. Umland, M. Werner, M. Oskin, D. Burbank, and D. Alsdorf. The shuttle radar topography mission. *Rev. Geophys.*, 45, 5 2007.

[10] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink. Streaming computation of Delaunay triangulations. In *Proceedings of SIGGRAPH*, 2006.

[11] M. Isenburg, Y. Liu, J. Shewchuk, J. Snoeyink, and T. Thirion. Generating raster DEM from mass points via TIN streaming. In M. Raubal, H. Miller, A. Frank, and M. Goodchild, editors, *Geographic Information Science - Fourth International Conference, GIScience 2006, Münster, Germany, September 2006.*, 2006.

[12] B. Lehner, K. Verdin, and A. Jarvis. New global hydrography derived from spaceborne elevation data. *Eos, Transactions*, 89(10):93–94, 2008.

[13] L. Mitas and H. Mitasova. Spatial interpolation. In P. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, editors, *Geographic Information Systems - Principles, Techniques, Management, and Applications*. Wiley, 1999.

[14] H. Mitasova, L. Mitas, and R. S. Harmon. Simultaneous spline interpolation and topographic analysis for lidar elevation data: methods for open source gis. *IEEE Geoscience and Remote Sensing Letters*, 2(4):375–379, 2005.

[15] T. Mølhave. *Handling Massive Terrains and Unreliable Memory*. PhD thesis, Aarhus University, Department of Computer Science, 8 2009.

[16] NASA and Japan (METI). NASA, Japan release most complete topographic map of earth, June 2009.

[17] J. F. O'Callaghan and D. M. Mark. The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics and Image Processing*, 28, 1984.

[18] M. Revsbæk. I/O efficient algorithms for batched union-find with dynamic set properties and its applications to hydrological conditioning. Master's thesis, Aarhus University, Denmark, 2007.

[19] J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Comput. Graph.*, 31(2):157–174, 2007.

[20] P. Soille, J. Vogt, and R. Colombo. Carving and adaptive drainage enforcement of grid digital elevation models. *Water Resources Research*, 39(12):1366–1375, 2003.

[21] D. Tarboton. A new method for the determination of flow directions and contributing areas in grid digital elevation models. *Water Resources Research*, 33:309–319, 1997.

[22] A. Tribe. 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, 1992.