Storing and analysing massive aerial LiDAR datasets in a DBMS

Hugo Ledoux

TU Delft
Technische Universiteit Delft

November 30 2010
ELMF, Den Haag
Computers have many problems dealing with billions of points:

- Storing is OK
- Visualisation is still a challenge
- **Processing** and **analysis** are very problematic

Processing operations:

- derivation of slope/aspect,
- conversion to grid format,
- calculations of area/volumes,
- viewshed analysis,
- creation of simplified DTM,
- extraction of bassins,
- etc.

Advances in technologies to collect data are far superior to our ability to process data.
Computers have many problems dealing with billions of points:
- Storing is OK
- Visualisation is still a challenge
- **Processing** and **analysis** are very problematic

Processing operations:
- derivation of slope/aspect,
- conversion to grid format,
- calculations of area/volumes,
- viewshed analysis,
- creation of simplified DTM,
- extraction of bassins,
- etc.

Advances in technologies to collect data are far superior to our ability to process data.
LiDAR datasets are formed by scattered points in 3D space, which are the samples of a surface that can be projected on the horizontal plan.
Reconstruction of the surface

Original LiDAR points
Reconstruction of the surface

Raster representation
Reconstruction of the surface

Raster representation
Reconstruction of the surface

TIN (with Delaunay triangles)
Reconstruction of the surface

TIN (with Delaunay triangles)
Reconstruction of the surface

TIN (with Delaunay triangles)
Related work

- Terrasolid: max is main memory
- ArcGIS: Terrain type (hierarchical structure)
- Oracle Spatial 11g: Point Cloud & TIN types
- External memory algorithms [AAD06, ADHZ06]
- Streaming of geometries of Isenburg et al. [ILSS06, ILS+06]
1. Storing independently triangles (∼ OGC)
2. Triangle-based data structure used by triangulation libraries [BDP⁺02]
3. Edge-based data structure (e.g. half-edge [M⁸⁸])
A star-based data structure

- Goes beyond the usual “store points and edges/triangles”
- Ideas come from data structures for compression of graphs [BBCK05]
A star-based data structure

- Goes beyond the usual “store points and edges/triangles”
- Ideas come from data structures for compression of graphs [BBCK05]
A star-based data structure

- Goes beyond the usual “store points and edges/triangles”
- Ideas come from data structures for compression of graphs [BBCK05]
A star-based data structure

- Goes beyond the usual “store points and edges/triangles”
- Ideas come from data structures for compression of graphs [BBCK05]

\[ \text{star}(p) = abcdef \]
Every star($v$) is stored $\rightarrow$ implicit triangles
Every \( \text{star}(v) \) is stored \( \rightarrow \) implicit triangles.
Every star($v$) is stored $\rightarrow$ implicit triangles
Every star($v$) is stored $\rightarrow$ implicit triangles
### Stars in a DBMS

<table>
<thead>
<tr>
<th>ID</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>star[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.21</td>
<td>5.23</td>
<td>2.11</td>
<td>[2,44,55,61,23]</td>
</tr>
<tr>
<td>2</td>
<td>5.19</td>
<td>29.01</td>
<td>4.55</td>
<td>[7,98,111,233,222]</td>
</tr>
<tr>
<td>3</td>
<td>22.43</td>
<td>15.99</td>
<td>8.19</td>
<td>[99,101,73,23]</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5674</td>
<td>221.19</td>
<td>15.23</td>
<td>37.81</td>
<td>[309,802,793,1111]</td>
</tr>
</tbody>
</table>

#### Advantages:

1. **Only one** table with `id − x − y − z − star`
2. No spatial index needed: fetching of triangles based on “walking”
3. Star column need not be filled (≈ Simple Features)
4. Local updates are possible (insertion and removals)
Point Location = “Walking” in the triangulation

(Can be made efficient with some tricks [MSZ99])
Range Queries: also uses the triangulation

(a) 

(b)
One problem: how to create that DT in the first place?

Streaming Delaunay Pipeline

```
spfinalize -i points.raw -ospb | spdelaunay -ispb -o tin.smb
```

- **Bounding box**
  - Enhances points with spatial finalization
- **Counts and sprinkles**
- **Sprinkler**
- **Chunker**
  - Uses spatial finalization to certify triangles as final

Figure from Martin Isenburg’s presentation at GIScience 2006 [ILSS06]
Streaming of geometries to construct massive TINs

\[ x, y, z \]
\[ x, y, z \]
\[ \ldots \]
\[ x, y, z \]

- \text{input.txt}
- \text{spfinalize}
- \text{spdelaunay2d}
- \text{smb2star.py}
- \text{PostgreSQL}

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]

\[
\begin{array}{c}
\text{points} \\
\text{points + triangles} \\
\text{ID, x, y, z, link}
\end{array}
\]
### Experiments with AHN2 datasets

<table>
<thead>
<tr>
<th></th>
<th># pts</th>
<th># triangle</th>
<th>degree\textsubscript{avg}</th>
<th>degree\textsubscript{max}</th>
</tr>
</thead>
<tbody>
<tr>
<td>20tiles</td>
<td>281 884 687</td>
<td>563 768 199</td>
<td>6.00</td>
<td>63</td>
</tr>
<tr>
<td>g37en1_15</td>
<td>8 605 090</td>
<td>17 201 289</td>
<td>6.00</td>
<td>39</td>
</tr>
</tbody>
</table>
Experiments with AHN2 datasets
Examples of queries: statistics about convex hull

20tiles=# select count(id) from points where is_convexhull(star) is true

count
-------
1173
(1 row)

Time: 333050.861 ms
Examples of queries: statistics about convex hull

```
20tiles=# select count(id) from points where is_convexhull(star) is true
count
-------
1173
(1 row)
Time: 333050.861 ms
```
Examples of queries: statistics about convex hull

20tiles=# select count(id) from points where is_convexhull(star) is true
    count
-------
    1173
(1 row)
Time: 333050.861 ms
Examples of queries: statistics about degree of vertices

```sql
20tiles=# select avg(degree(star)) from points;
   avg
---------------------
  5.9999958142601850
(1 row)
Time: 332265.041 ms
```
Examples of queries: statistics about degree of vertices

```
20tiles=# select avg(degree(star)) from points;
    avg
-------------------
5.9999958142601850
(1 row)
Time: 332265.041 ms
```
20tiles=# select avg(degree(star)) from points;

avg
-------------------
5.9999958142601850
(1 row)

Time: 332265.041 ms
Examples of queries: statistics about degree of vertices

\[
g37en1_15=# \text{select degree(star) as degree, count(id) from points group by degree order by degree;}
\]

<table>
<thead>
<tr>
<th>degree</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>65620</td>
</tr>
<tr>
<td>4</td>
<td>844625</td>
</tr>
<tr>
<td>5</td>
<td>2277911</td>
</tr>
<tr>
<td>6</td>
<td>2484212</td>
</tr>
<tr>
<td>7</td>
<td>2005407</td>
</tr>
<tr>
<td>8</td>
<td>698540</td>
</tr>
<tr>
<td>9</td>
<td>170214</td>
</tr>
<tr>
<td>10</td>
<td>37534</td>
</tr>
<tr>
<td>11</td>
<td>9587</td>
</tr>
<tr>
<td>12</td>
<td>3395</td>
</tr>
<tr>
<td>13</td>
<td>1552</td>
</tr>
<tr>
<td>14</td>
<td>772</td>
</tr>
<tr>
<td>15</td>
<td>456</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>3</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
</tr>
</tbody>
</table>

(35 rows)

Time: 39722.017 ms
Examples of queries: point location

g37en1_15=# select point_location(84111, 446666, 0, 0.2);
WARNING: # of samples checked: 24
WARNING: start distance is 49.843357
WARNING: # of triangles visited is 222
point_location
---------------------------
(3672278,3695197,3695256)
(1 row)
Time: 191.903 ms
Visualisation with a GIS
Thanks for your attention

Hugo Ledoux
h.ledoux@tudelft.nl
P. K. Agarwal, L. Arge, and A. Danner.
From point cloud to grid DEM: A scalable approach.

L. Arge, A. Danner, H. Haverkort, and N. Zeh.
I/O-efficient hierarchical watershed decomposition of grid terrain models.

Daniel K. Blandford, Guy E. Blelloch, David E. Cardoze, and Clemens Kadow.
Compact representations of simplicial meshes in two and three dimensions.

Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec.
Triangulations in CGAL.

Generating raster DEM from mass points via TIN streaming.

Martin Isenburg, Yuanxin Liu, Jonathan Richard Shewchuk, and Jack Snoeyink.
Streaming computation of Delaunay triangulations.

Martti Mäntylä.
An introduction to solid modeling.

Ernst P. Mücke, Isaac Saias, and Binhai Zhu.
Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations.
<table>
<thead>
<tr>
<th>tiles</th>
<th>time pipeline</th>
<th>time B-tree</th>
<th>time tr</th>
<th>time lasblock</th>
<th>width max</th>
</tr>
</thead>
<tbody>
<tr>
<td>20tiles</td>
<td>178.3</td>
<td>24.65</td>
<td>36.8</td>
<td>crashed</td>
<td>53 003</td>
</tr>
<tr>
<td>g37en1_15</td>
<td>4.7</td>
<td>0.3</td>
<td>1.1</td>
<td>4.3</td>
<td>7 601</td>
</tr>
</tbody>
</table>