Creating the Medial Axis Transform for billions of LiDAR points using a memory efficient method
Creating the **Medial Axis Transform** for billions of **LiDAR points** using a memory efficient method
LiDAR points
Medial Axis
Creating the Medial Axis Transform for billions of LiDAR points using a memory efficient method
Content

• Introduction
• Research question
• Methodology
  • Shrinking Ball Algorithm (MAT)
  • Scaling using Tiling
  • Scaling using Streaming
• Test results
• Conclusions / Discussion / Future Work
Introduction

Medial Axis
Introduction

Medial Axis
Introduction

Medial Axis

[Diagram of a rectangle with a medial axis drawn through its center]
Introduction

Medial Axis

Circle:
- It is completely inside the boundaries of the shape
- It is tangent to more than just one boundary point
Introduction

*Medial Axis*
Introduction

Medial Axis
Introduction

Medial Axis Transform
Introduction

Medial Axis Transform
Introduction

Medial Axis Transform
Introduction

Medial Axis Transform

Inner MAT
Introduction

Medial Axis Transform

Inner MAT
Introduction

Medial Axis Transform

Inner MAT

Outer MAT
Introduction

Medial Axis Transform - Properties

Complete
Introduction

Medial Axis Transform - Properties

Complete
Introduction

Medial Axis Transform - Properties

Complete Topology
Compact Hierarchy
Medial Sensitive
Introduction

Applications

Segmentation
M. Berger, Medial kernels

Point cloud simplification
J. Ma, 3D medial axis point approximation using nearest neighbors and the normal field

Surface reconstruction
N. Amenta, The power crust

Visibility Analysis
R. Peters, Visibility analysis in a point cloud based on the medial axis transform
Introduction

Problem Statement

Main memory: 500 MB
Introduction

Problem Statement

Main memory: 1500 MB
Introduction

Problem Statement

Main memory:

...
Introduction

Problem Statement

Internal Memory:
• Smaller
• Faster

VS

External Memory:
• Bigger
• Slower
Research question

How can the 3D medial axis point approximation using the shrinking ball algorithm be scaled in a memory efficient way for a large dataset which does not fit in the internal memory?
What are the challenges in scaling the 3D medial axis using the shrinking ball algorithm?

How to design and implement several methods for scaling the shrinking ball algorithm?

How do the methods compare to each other?
Methodology

Shrinking ball algorithm
Methodology

*shrinking ball algorithm*
Methodology

*Shrinking ball algorithm*
Methodology

Shrinking ball algorithm

\[ R_{\text{initial}} = 100 \text{m} \]
Methodology

*Shrinking ball algorithm*
Methodology

Shrinking ball algorithm
Methodology

Shrinking ball algorithm
Methodology

*Shrinking ball algorithm*
Methodology

Shrinking ball algorithm
Methodology

Shrinking ball algorithm
Methodology

*Shrinking ball algorithm*
Methodology

*Shrinking ball algorithm*
Methodology

*Challenges: Shrinking ball algorithm (part1)*

<table>
<thead>
<tr>
<th>X [m]</th>
<th>Y [m]</th>
<th>Z [m]</th>
<th>R [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4400.0</td>
<td>5000.0</td>
<td>10.1</td>
<td>56 (&lt; 100)</td>
</tr>
</tbody>
</table>
Methodology

*Shrinking ball algorithm*

\[ R_{\text{initial}} = 100 \text{m} \]

\[ R_{\text{initial}} \text{ should be the 0.5 the maximum detectable object} \]
Methodology

If a dataset it too large
Process smaller subsets
Methodology

If a dataset it too *large*
Process *smaller* subsets (tiles)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Grid image]
Methodology

If a dataset it too large
Process smaller subsets

Tiling
- Make extensive use of the external memory to store tiles temporary

Streaming
- Create tiles while reading the input file (Avoiding use of external memory)
Methodology

Challenges: Shrinking ball algorithm (part1)

What are the challenges in scaling the 3D medial axis using the shrinking ball algorithm?
Methodology

Challenges: Shrinking ball algorithm (part1)
Methodology

Challenges: Shrinking ball algorithm (part1)
Methodology

Challenges: Shrinking ball algorithm (part1)
Methodology

Challenges: Shrinking ball algorithm (part1)
Methodology

Challenges: Shrinking ball algorithm (part1)
Methodology

Tiling

How to design and implement several methods for scaling the shrinking ball algorithm?
Methodology

Tiling

How to design and implement several methods for scaling the shrinking ball algorithm?
Methodology

*Tiling*

- Dataset
- Tiling
- Segmentation
- Compute MAT
- Merge Dataset
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

Spread in x and y >>> z
Methodology

**Tiling**

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset
Methodology

**Tiling**

- Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

Storing Tiles on external memory
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset
Methodology

**Tiling**

1. Dataset
2. Tiling
3. Segmentation
4. Compute MAT
5. Merge Dataset

Diagram:

- 4 circles with a diameter of 100m, arranged in a square pattern.
- The circles are evenly spaced, with each circle touching the center of the diagram, and each other circle.

Each circle represents a 100m segment, and the squares formed by the intersection of these circles represent the tiling process.
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

200m 200m 200m 200m 200m 200m 200m

200m 200m 200m 200m 200m 200m 200m
Methodology

**Tiling**

1. Dataset
2. Tiling
3. Segmentation
4. Compute MAT
5. Merge Dataset

Each step is 200m in size.
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

collection of tiles may fit in the memory as well…
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

space-driven segmentation

Process Tiles

Buffer Tiles
Methodology

**Tiling**

1. Dataset
2. Tiling
3. Segmentation
4. Compute MAT
5. Merge Dataset

**space-driven segmentation**

- Process Tiles
- Buffer Tiles
Methodology

*Tiling*

- Dataset
- Tiling
- Segmentation
- Compute MAT
- Merge Dataset

space-driven segmentation

Process Tiles

Buffer Tiles
Methodology

**Tiling**

1. Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

**data-driven segmentation**

- Process Tiles
- Buffer Tiles
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

data-driven segmentation

Process Tiles

Buffer Tiles
Methodology

*Tiling*

- Dataset
- Tiling
- Segmentation
- Compute MAT
- Merge Dataset

Data-driven segmentation

Process Tiles

Buffer Tiles
# Methodology

## Tiling

<table>
<thead>
<tr>
<th>Max Points per cluster</th>
<th>c.37en2</th>
<th>c.67hz1</th>
<th>c.11hz2</th>
<th>c.37gn1</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 mil</td>
<td>QT</td>
<td>KD</td>
<td>OKD</td>
<td>QT</td>
</tr>
<tr>
<td>23 21 23</td>
<td>12 11 12</td>
<td>23 25 25</td>
<td>10 9 9</td>
<td></td>
</tr>
<tr>
<td>31 33 32</td>
<td>14 16 16</td>
<td>35 32 31</td>
<td>14 15 15</td>
<td></td>
</tr>
<tr>
<td>48 50 47</td>
<td>26 24 25</td>
<td>51 53 50</td>
<td>24 21 21</td>
<td></td>
</tr>
<tr>
<td>90 89 91</td>
<td>44 44 44</td>
<td>95 103 100</td>
<td>41 41 41</td>
<td></td>
</tr>
<tr>
<td>111 114 109</td>
<td>56 50 48</td>
<td>122 121 117</td>
<td>50 50 48</td>
<td></td>
</tr>
<tr>
<td>154 - 147 74 74 -</td>
<td>- - - -</td>
<td>66 68 65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Methodology

**Tiling**

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

- Process Tiles
- Buffer Tiles
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

Process Tiles

Buffer Tiles
Methodology

Tiling

Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset

Process Tiles → Buffer Tiles
Methodology

**Tiling**

- Dataset → Tiling → Segmentation → Compute MAT → Merge Dataset
Methodology

**Streaming**

- Dataset
- Streaming Spatial Finalizer
- Segmentation
- Compute MAT
- Merge Dataset
Methodology

**Streaming**

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Streaming with spatial finalizer scans through the file and pipes tiles which are completed.
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Real-life datasets have good: Locality (spatial coherence)
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

Streaming

1. Dataset
2. Streaming Spatial Finalizer
3. Segmentation
4. Compute MAT
5. Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

**Streaming**

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Store read tiles (from the input) dataset in the main memory
Methodology

**Streaming**

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

**Streaming**

1. **Dataset**
2. **Streaming Spatial Finalizer**
3. **Segmentation**
4. **Compute MAT**
5. **Merge Dataset**

Store read tiles (from the input) dataset in the main memory
Methodology

*Streaming*

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Store read tiles (from the input) dataset in the main memory
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset
Methodology

**Streaming**

- Dataset
- Streaming Spatial Finalizer
- Segmentation
- Compute MAT
- Merge Dataset

![Diagram of methodology steps](image-url)
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Using regular buffers, the blue tile can only be deleted after all 24 surrounding tiles have arrived.
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Reduced Buffer

If medial ball is inside the boundary, the MAT is final

If medial ball is outside the boundary, the MAT might need more data
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Reduced Buffer

Hypothesis:
Points with a final MAT are not needed for the computation of MAT’s of outside the tile

You can remove these points after the final MAT is computed
Methodology

*Streaming*

- Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset

Thinned Reduced Buffer

Side view building
Thinning reduced buffer, based on MAT output does not always work.
Methodology

Streaming

Dataset → Streaming Spatial Finalizer → Segmentation → Compute MAT → Merge Dataset
Using “reduced buffer”, the points of the process tile can be removed after the MAT has been computed.
Using “reduced buffer”, the points of the process tile can be removed after the MAT has been computed.
Methodology

Streaming & Tiling merging
Methodology

Streaming & Tiling merging

Morton (Z-order) curve
Methodology

Summary

What are the challenges in scaling the 3D medial axis using the shrinking ball algorithm?

"Regular" Buffer
- Computes only the blue area

Reduced Buffer
- Computes the MAT from the yellow area as well
- Points of the blue area are not needed in any other computation

Thinned Reduced Buffer
- Deletes assumed unnecessary points in yellow area as well after computation
- Removing points based on their MAT not possible for pointclouds!
Methodology

Summary

How to design and implement several methods for scaling the shrinking ball algorithm?

Tiling

- Extensive use of external memory
- Making large collections of tiles possible
- Can make use of “regular” buffers

Streaming
(using spatial finalizer)

- Avoids use of external memory
- make small collections of tiles
- To be efficient, should make use of “reduced” buffer
Test Results

How do the methods compare to each other?
Test Results

Rotterdam Dataset
(88000, 432000)(89600, 433600)

Size: 1600 x 1600 m²
Points: 76 Million
Subdivided: 64 tiles
### Test Results

Rotterdam Dataset
(88000, 432000)(89600, 433600)

<table>
<thead>
<tr>
<th># collections</th>
<th>Comp. Time</th>
<th>Main memory</th>
<th>Ext. memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 collections</td>
<td>1686 sec</td>
<td>1569 MB</td>
<td>1828 MB</td>
</tr>
<tr>
<td>16 collections</td>
<td>1626 sec</td>
<td>1059 MB</td>
<td>1828 MB</td>
</tr>
<tr>
<td>64 collections</td>
<td>1755 sec</td>
<td>927 MB</td>
<td>1828 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># collections</th>
<th>Comp. Time</th>
<th>Main memory</th>
<th>Ext. memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 collections</td>
<td>2886 sec</td>
<td>1250 MB + 152 MB</td>
<td>342 MB</td>
</tr>
</tbody>
</table>

### Test Results Details

- **Size:** 1600 x 1600 m²
- **Points:** 76 Million
- **Subdivided:** 64 tiles

**Tiling (tiling 20.6 s, 0.1 s)**

External memory usage is irrelevant as the MAT takes the about same amount of space.

**Streaming (14 s)**
Test Results

Rotterdam Dataset
(88000, 432000)(89600, 433600)

Size: 1600 x 1600 m²
Points: 76 Million
Subdivided: 64 tiles

Because the streaming algorithm cannot make use of “thinned” reduced buffer, it performs much worse.
Test Results

AHN Dataset
c_37gn1

Size: 5000 x 3000 m²
Points: 237 Million
Subdivided: 375 tiles
Test Results

AHN Dataset
- Size: 5000 x 3000 m²
- Points: 237 Million
- Subdivided: 375 tiles

![Graph of Memory Usage](image1.png)

Memory Usage

- Memory Usage [MB] vs. collections

![Graph of Computation Time](image2.png)

Computation Time

- Computation Time [s] vs. collections
Test Results

With the increase of amounts of collections:
- Memory usage decreases exponentially
- Computation time increases linearly

Points in dataset are not spread homogeneous
Test Results

A good method to compute the MAT using the tiling method:
- Determine what the maximum amount of points per individual tile is (including buffers)
- Then create collections using that maximum amount of points found in the previous step

![Memory Usage Graph](image1)

![Computation Time Graph](image2)
Conclusion

How can the 3D medial axis point approximation using the shrinking ball algorithm be scaled in a memory efficient way for a large dataset which does not fit in the internal memory?

The MAT can be created memory efficiently using the shrinking ball algorithm in combination with the tiling method using the segmentation method based on the optimized KD-tree by making many similar small sized collections with buffers.
Future work

Streaming could perform much better if a thinned reduced buffer method is found

Normal calculations could be improved for streaming method

Other segmentation methods could be tried for the tiling method
Test Results

Rotterdam Dataset
(88800, 432800)(89600, 433600)

Size: 800 x 800 m²
Points: 22 Milion
Subdivided: 16 tiles
# Test Results

Rotterdam Dataset
(88800, 432800)(89600, 433600)

<table>
<thead>
<tr>
<th>Size:</th>
<th>800 x 800 m²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>22 Million</td>
</tr>
<tr>
<td>Subdivided:</td>
<td>16 tiles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tiling (tiling 2.9 s, segmentation 0.1 s)</th>
<th>Streaming (4.6 s)</th>
</tr>
</thead>
<tbody>
<tr>
<td># collections</td>
<td>1 collections</td>
</tr>
<tr>
<td>Comp. Time</td>
<td>592 sec</td>
</tr>
<tr>
<td>Main memory</td>
<td>1342 MB</td>
</tr>
<tr>
<td>Ext. memory</td>
<td>494 MB</td>
</tr>
</tbody>
</table>

![Grid Diagram](image)
Methodology

Streaming

1\textsuperscript{st} pass:
Boundary of the dataset
And create cells
Methodology

**Streaming**

1. **Dataset**
2. **Streaming Spatial Finalizer**
3. **segmentation**
4. **Compute MAT**
5. **Merge Dataset**

**1st pass:**
Boundary of the dataset
And create cells

**2nd pass:**
Count points per cell
Methodology

Streaming

1\textsuperscript{st} pass:
Boundary of the dataset
And create cells

2\textsuperscript{nd} pass:
Count points per cell

3\textsuperscript{rd} pass:
Count points per tile till all have arrived then output.