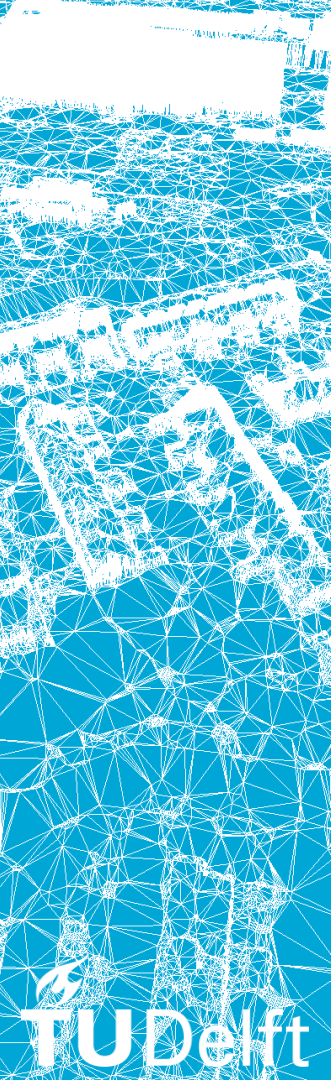


Simplication of Massive TINs with the Streaming Geometries Paradigm

Maarten de Jong

July 2021

1st Supervisor: Hugo Ledoux
2nd Supervisor: Ravi Peters
Co-reader: Balázs Dukai

Content

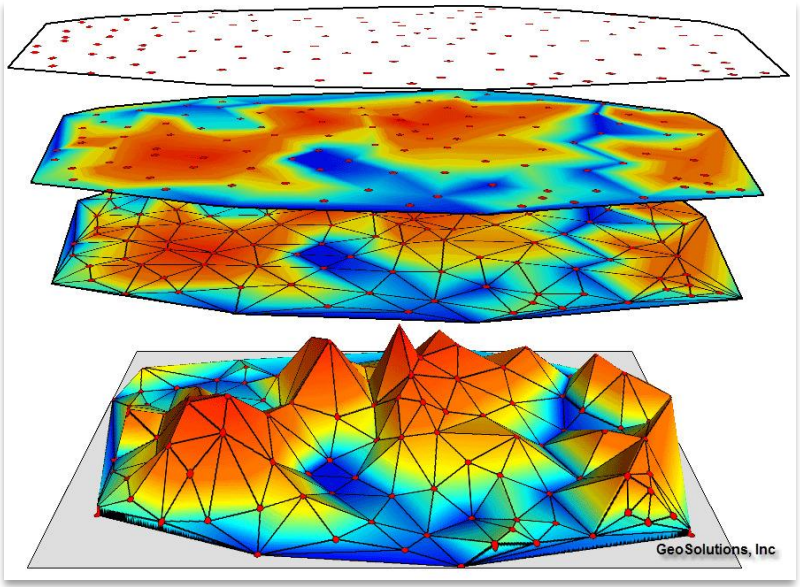
- Background
- Relevance
- Research questions
- Approach
- Results
- Conclusions

Background

- TIN
  - Delaunay
- Streaming
- Previous work
- Spatial coherence

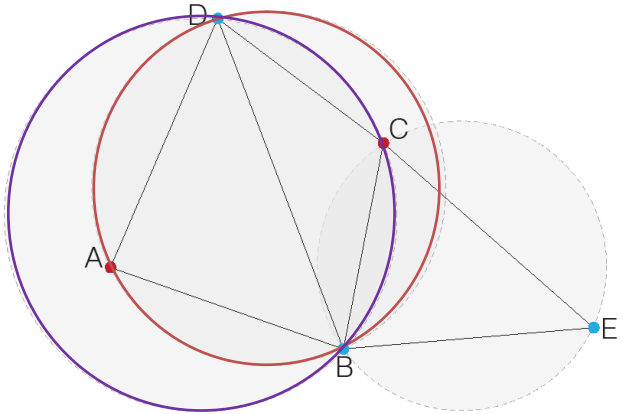
Background

- **TIN**
  - Delaunay
- Streaming
- Previous work
- Spatial coherence

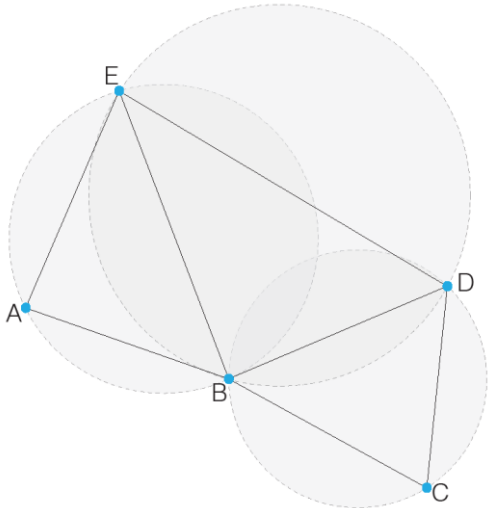


Background

- TIN
  - **Delaunay**
- Streaming
- Previous work
- Spatial coherence



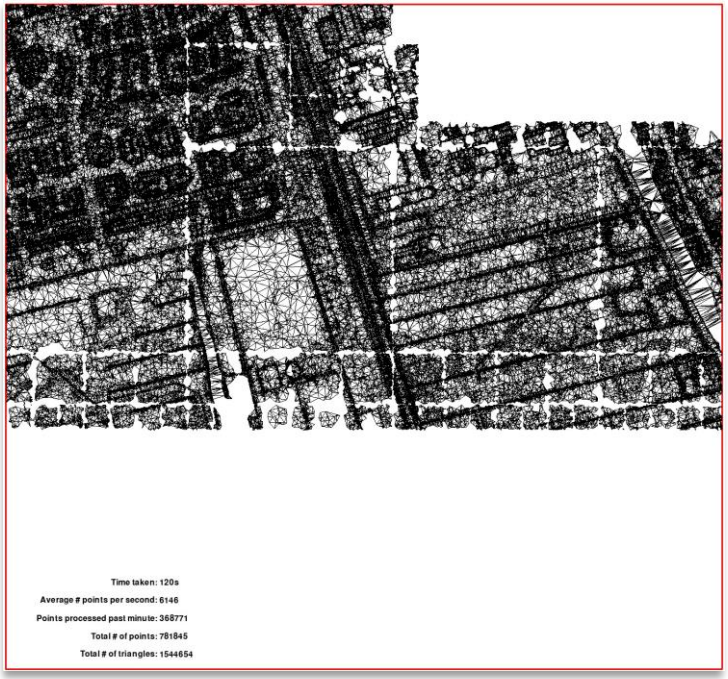
Invalid Delaunay

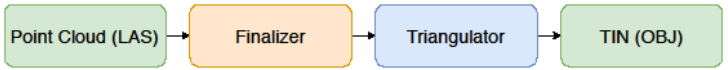


Valid Delaunay

Background

- TIN
  - Delaunay
- **Streaming**
- Previous work
- Spatial coherence

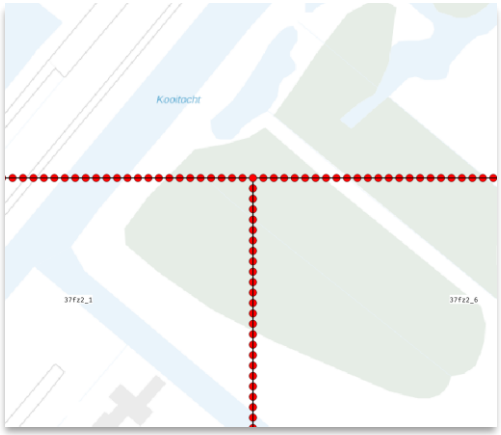




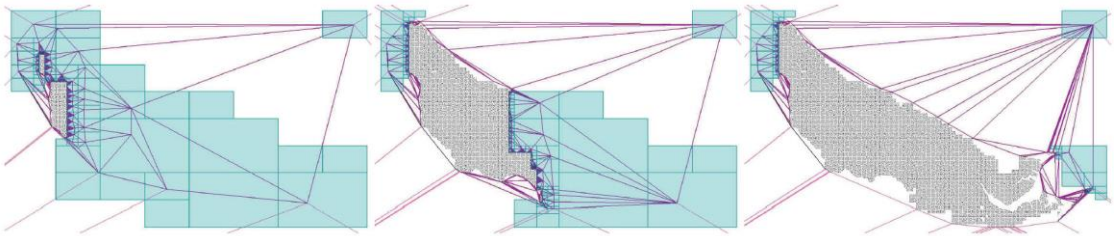
TU Delft

Background

- TIN
  - Delaunay
- Streaming
- **Previous work**
  - Spatial coherence



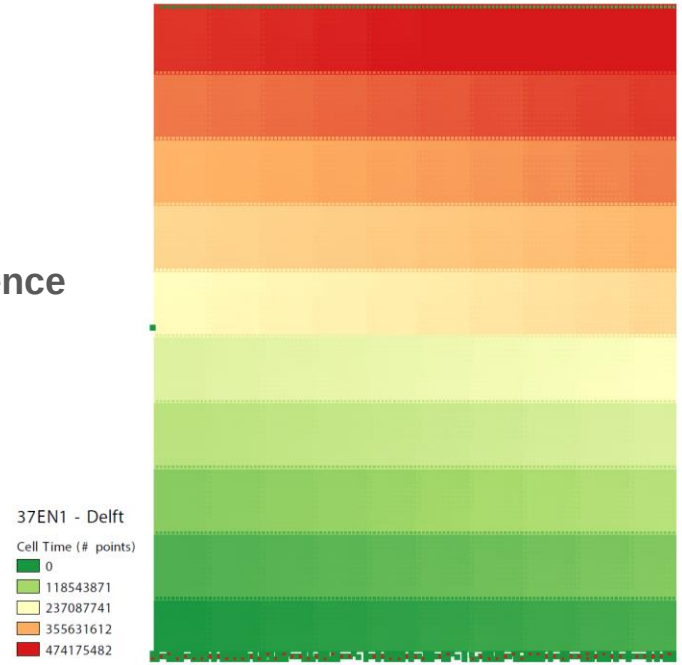
Dukai [2020]

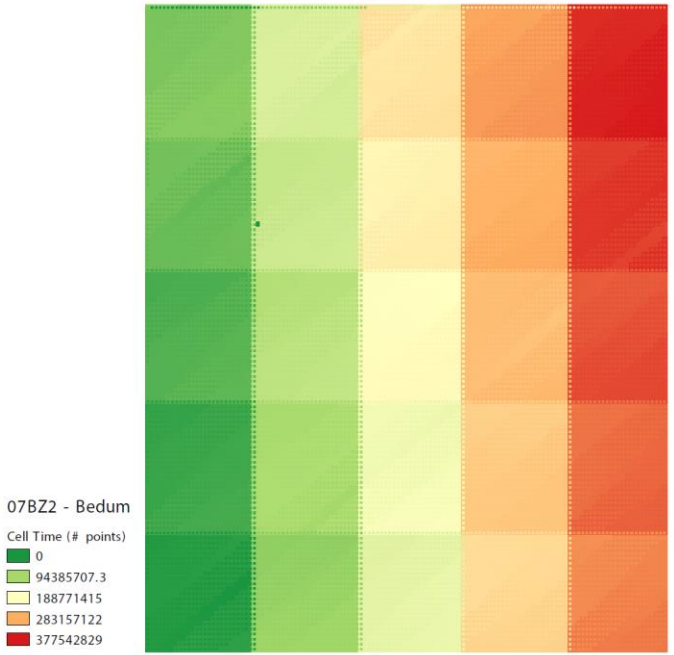


Isenburg et al. [2006]

Background

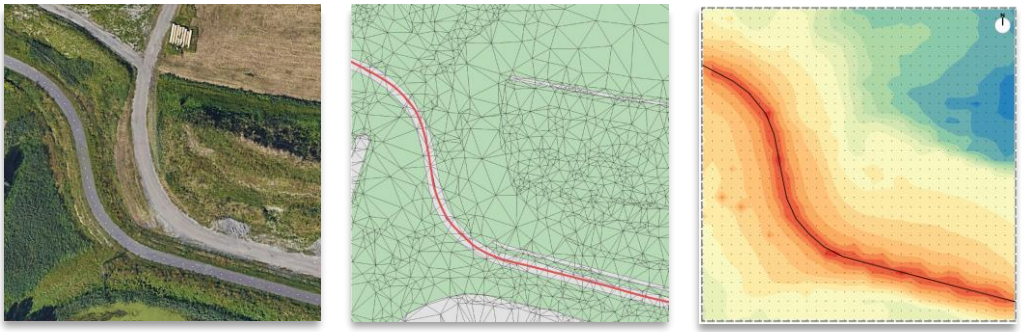
- TIN
  - Delaunay
- Streaming
- Previous work
  - **Spatial coherence**



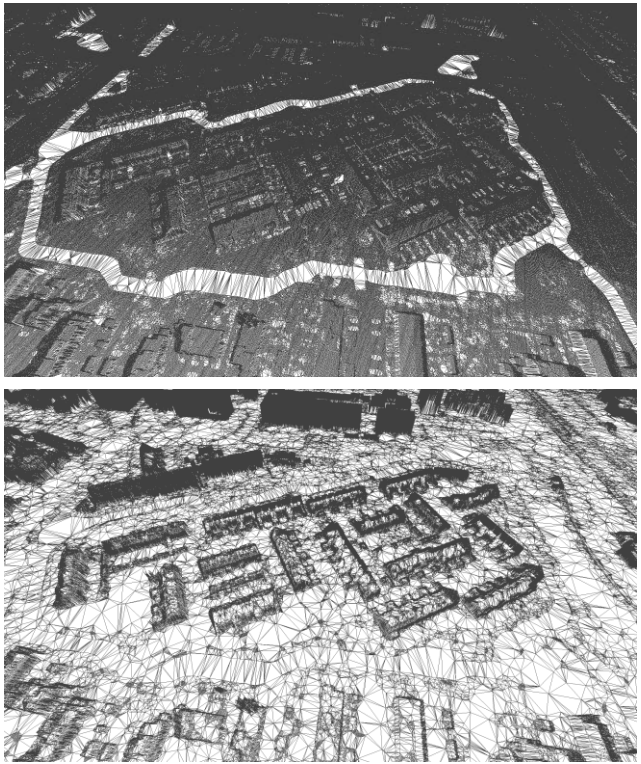


Relevance

- Usability of massive dataset
- Noise modelling
- 3D reconstruction
- Accurate surface representation



van Rijssel et al. [2020]



Objective

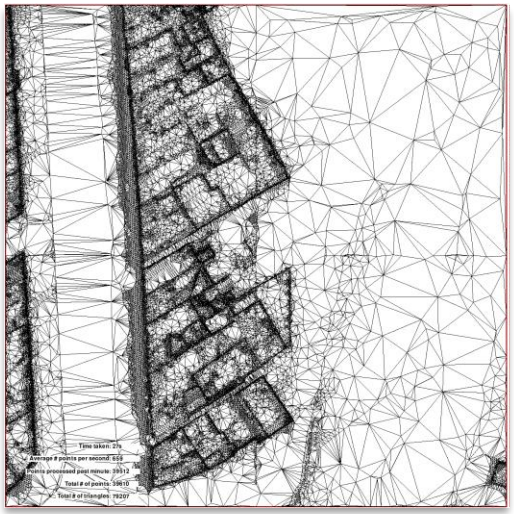
Seamless, simplified TIN of 8 AHN3 tiles in Zuid-Holland

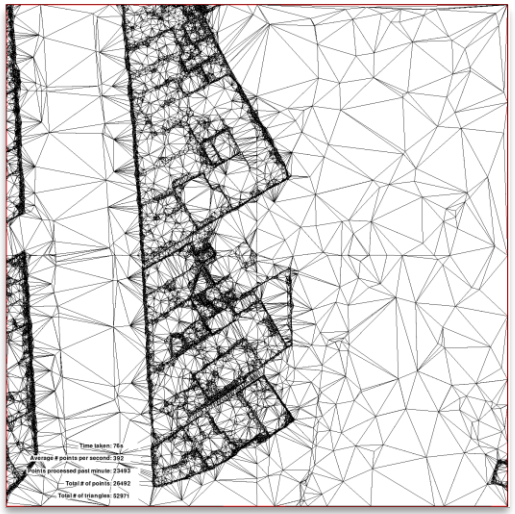
Research Questions

- How can a seamless, simplified, Delaunay TIN for all AHN3 points be constructed using the streaming geometries paradigm?
  - How can simplification be integrated into the streaming creation of a Delaunay TIN?
  - Which TIN simplification method produces the best results when used in a streaming pipeline?
  - How does the streaming creation and simplification of a Delaunay TIN perform in comparison to existing methods in terms of execution time, memory usage, and accuracy?

Approach

- Integrate simplification into existing streaming pipeline
- Write algorithms from the ground up
- Iterate on most promising options

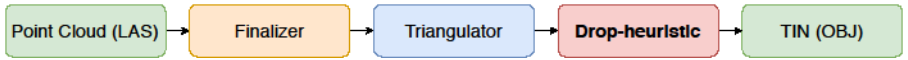




Streaming Simplification

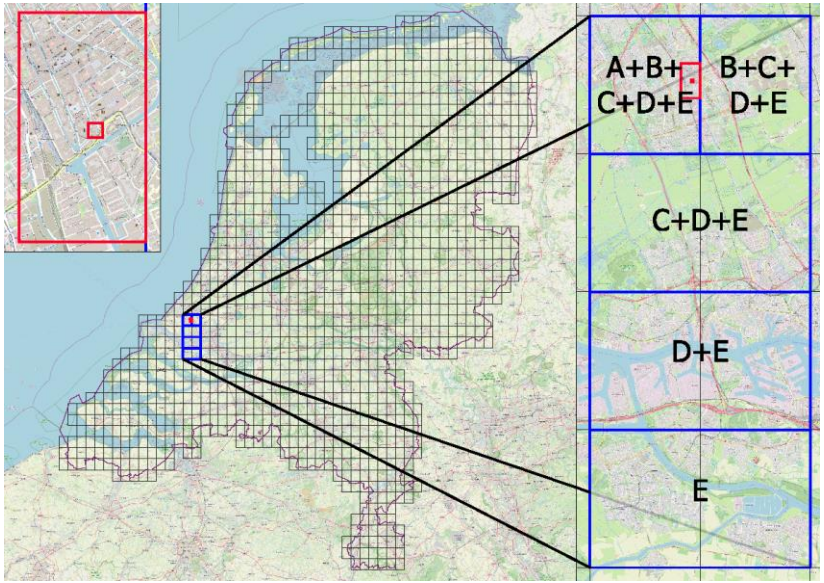
Streaming Simplification

- Random thinning
  - Control method
- Drop-heuristic
  - Decimation approach
- MAT
  - Simplifies based on points
- Refinement
  - Constructive simplification



Datasets

	area	number of points	average density
Delft-Tiny	0.01km^2	212,550	21.3 points/m^2
Delft-Small	1.40km^2	20,310,070	14.5 points/m^2
One Tile	31.25km^2	343,044,606	11.0 points/m^2
Two Tiles	62.50km^2	642,167,999	10.3 points/m^2
Four Tiles	125.00km^2	1,354,295,996	10.0 points/m^2
Six Tiles	197.50km^2	1,902,652,509	9.6 points/m^2
Eight Tiles	250.00km^2	2,424,250,813	9.7 points/m^2



Results

Results

Delft-Tiny

	time taken	vertices	triangles	RMSE	max error (m)	max memory (MB)
No Simplification	2s	212,550	425,046	-	-	-
Rand5	1s	35,318	70,600	0.94	10.3	12
Decim100	27m25s	53,345	104,999	0.23	8.4	675
Greedy025	1m25s	37,651	75,270	0.07	5.2	357
Greedy1	44s	39,994	39,957	0.09	5.3	362
Greedy5	23s	44,570	89,109	0.06	1.4	358
FCFS	1s	34,704	69,395	0.13	6.2	337
FCFS + Decim10	8m30s	26,419	52,825	0.31	7.8	400
MAT02	15s	14,551	29,057	1.06	11.0	142

Delft-Small

	time taken	vertices	triangles	RMSE	max error (m)	max memory (MB)
No Simplification	3m30s	20,319,438	40,638,770	-	-	-
Rand3	1m37s	5,079,401	10,158,749	1.38	30.0	12
Decim100	>18h	-	-	-	-	-
Greedy025	29m26s	5,021,178	10,042,286	0.12	12.3	684
Greedy1	17m15s	5,333,951	10,667,831	0.10	9.0	630
Greedy5	9m12s	5,970,399	11,940,721	0.11	10.9	773
FCFS	4m4s	4,916,929	9,833,779	0.20	15.0	491
FCFS + Decim10	8h4m	3,614,934	7,229,797	0.46	24.0	385
MAT02	14m5s	2,236,205	4,472,145	1.53	55.7	345
Full Refinement	17h53m	4,210,394	8,420,723	0.16	20.0	673

TU Delft

Results

- Greedy1
  - Low RMSE
  - Long runtime
- Greedy5
  - Same RMSE as Greedy1
  - 2h less runtime vs. Greedy1
- FCFS
  - Relatively high RMSE
  - Shortest runtime

Eight Tiles

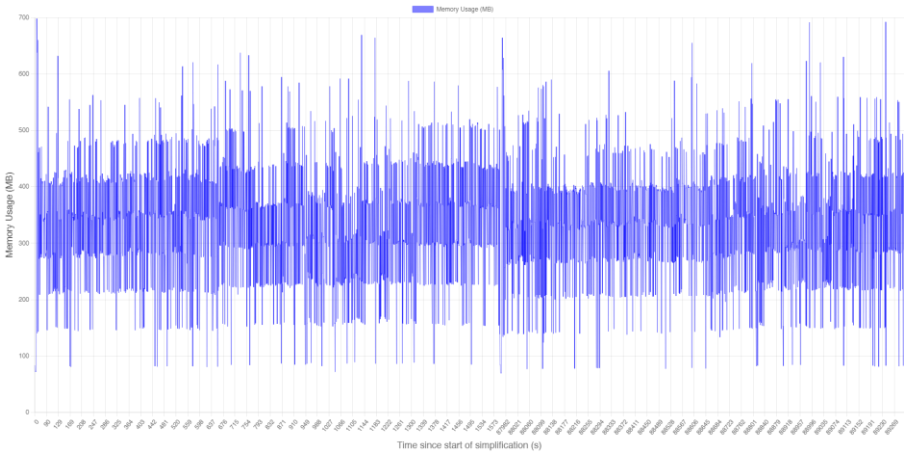
	time taken	vertices	triangles	RMSE	max error (m)	max memory (MB)
No Simplification	12h55m[1]	2,424,250,813	-	-	-	-
Rand10	2h23m	225,004,204	450,008,309	1.76	44.2	12
Greedy1	17h11m	228,154,328	456,308,291	0.16	31.5	729
Greedy5	15h23m	279,231,753	558,463,139	0.16	38.4	828
FCFS	10h50m	199,632,116	399,263,875	0.21	39.1	776

[1] data piped to /dev/null instead of file

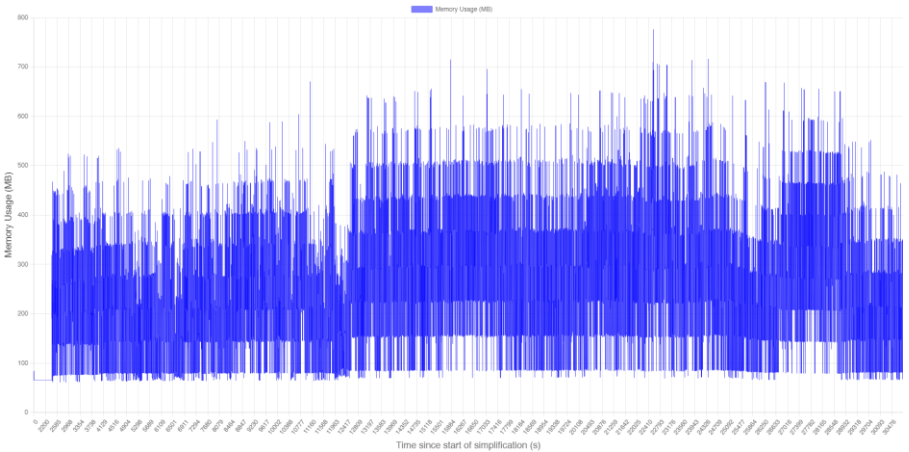
TU Delft

Results

FCFS – Two Tiles

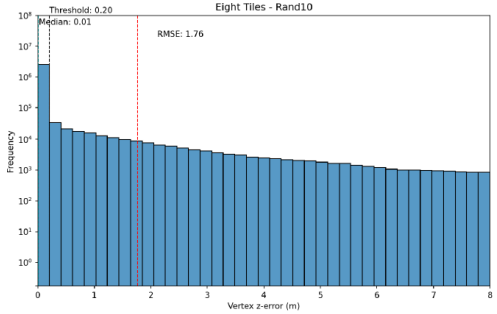


FCFS – Eight Tiles

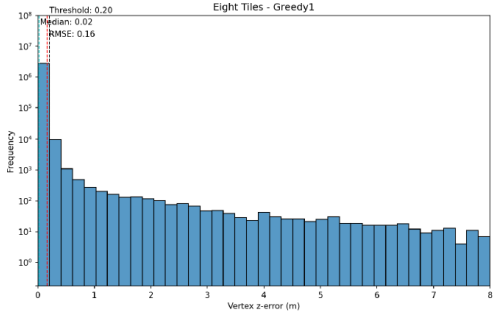


TU Delft

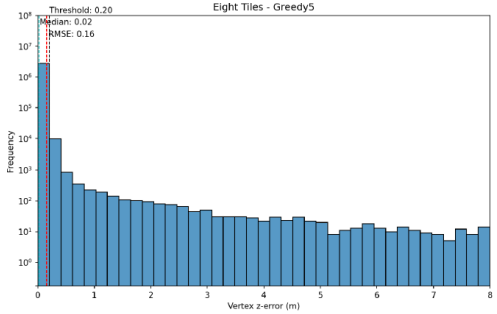
Results



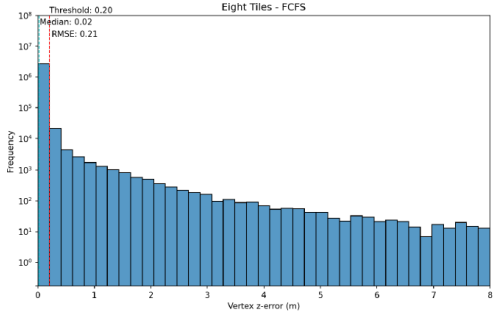
(a) Rand10



(b) Greedy1



(c) Greedy5



(d) FCFS

TU Delft

Results

- Random simplification

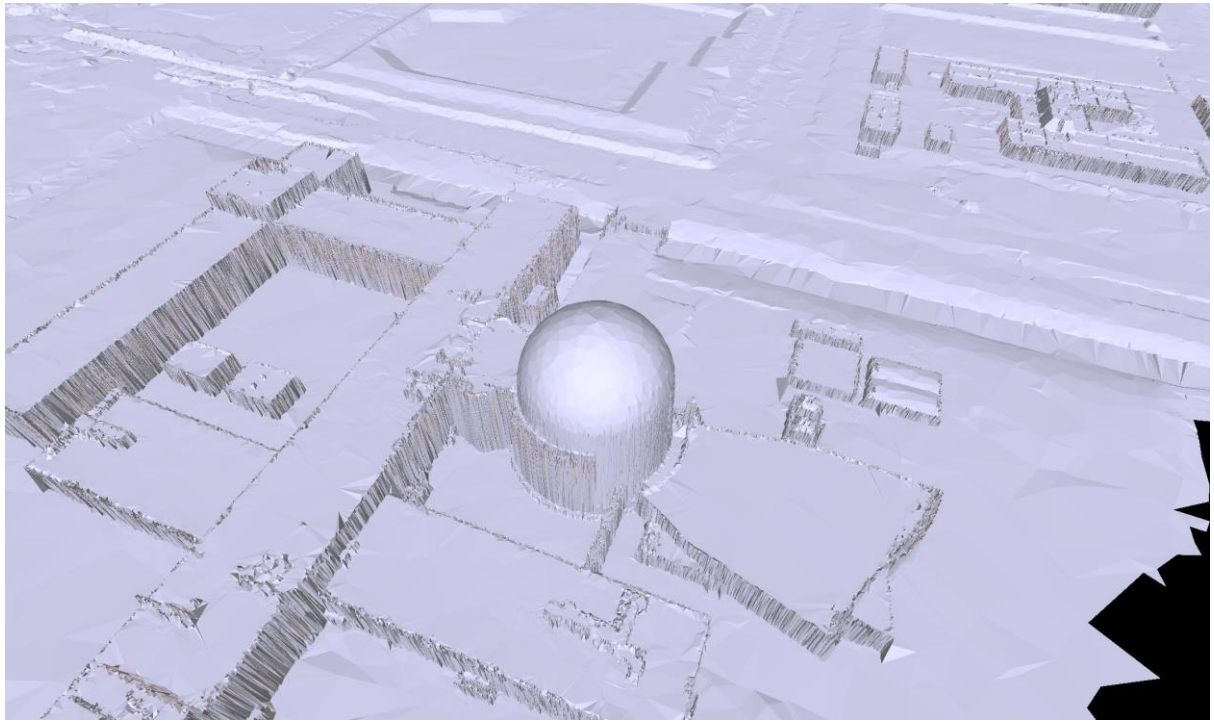
Results

- Greedy1

Results

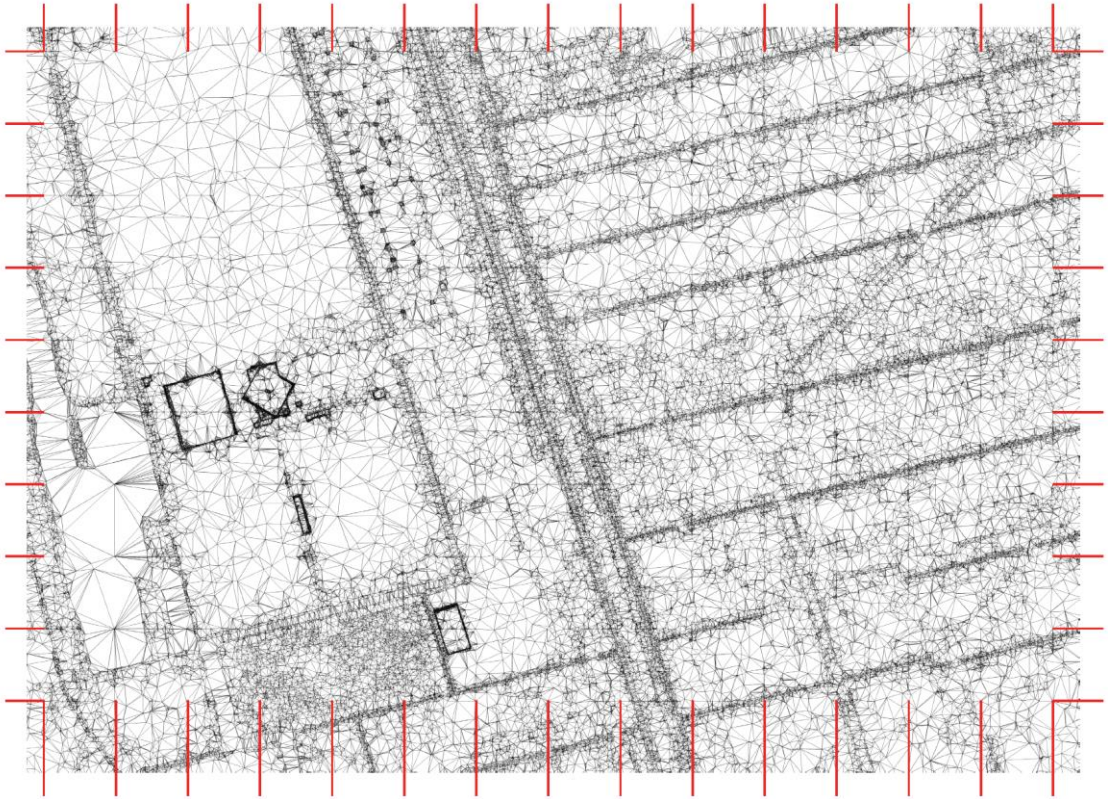
- FCFS

Results



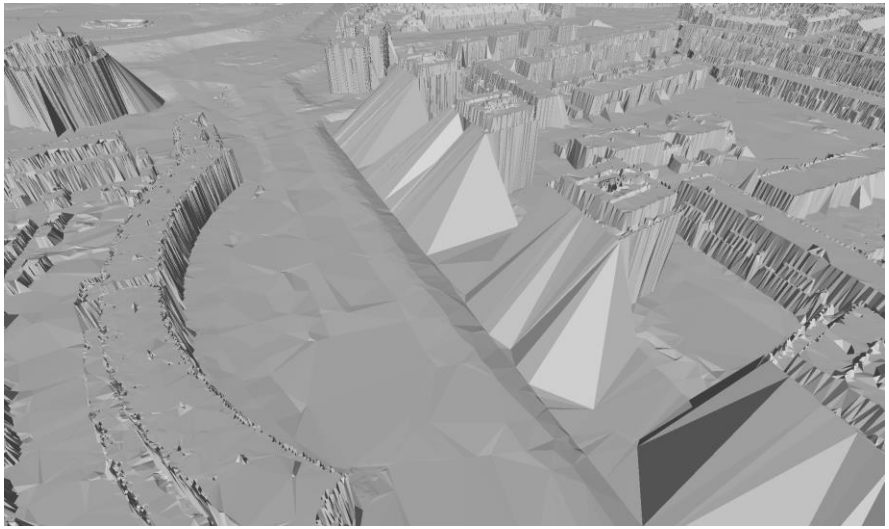
Artefacts

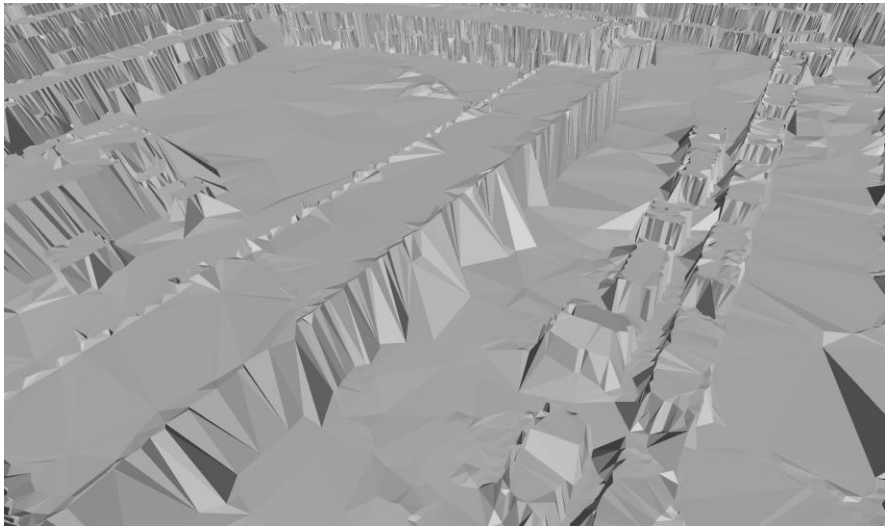
- Cell borders



Artefacts

- Roof edges





Comparison

- *Isenburg*
  - Similar speed, less simplification
  - Unknown RMSE
- *Hegeman*
  - Slower
  - Better RMSE
- *Dukai*
  - Slightly better RMSE
  - Unknown speed comparison

	triangles per second	simplification
Isenburg et al. [2006d]	100,000	90%
Greedy1	74,251	87%
Greedy5	104,905	84%
FCFS	127,051	88%

	vertices per second	simplification	RMSE (m)
Hegeman et al. [2014]	1,600,000	32%	0.07
Hegeman et al. [2014]	4,300,000	81%	1.90
Rand6	184,430	86%	1.44
Greedy1	37,125	87%	0.14
Greedy5	52,452	84%	0.14
FCFS	63,526	88%	0.18

	simplification	RMSE (m)	max error (m)
Dukai [2020]	99.7%	0.10	8.9
Greedy1	98.5%	0.09	6.7
Greedy5	97.6%	0.09	4.4
FCFS	99.3%	0.09	6.3

TU Delft

Conclusions

- **How can a seamless, simplified, Delaunay TIN for all AHN3 points be constructed using the streaming geometries paradigm?**

- Using sst
- Can be done using either Greedy or FCFS refinement
- 86 days for FCFS for entire AHN3
- 107 days when using Greedy5

TU Delft

Conclusions

- **How can TIN simplification be integrated into the streaming creation of a Delaunay triangulation?**

- At one of three positions
- Preferably at B

TU Delft

Conclusions

- **Which TIN simplification method produces the best results when used in a streaming pipeline?**

- Greedy methods have the lowest RMSE
- FCFS best overall for speed and relatively low RMSE

Conclusions

- **How does the streaming creation and simplification of a Delaunay TIN perform in comparison to existing methods in terms of execution time, memory usage, and accuracy?**

- Marginally outperforms Dukai
- Is outperformed by Hegeman on speed
- Provides best RMSE of all comparisons

	triangles per second	simplification
Isenburg et al. [2006d]	100,000	90%
Greedy1	74,251	87%
Greedy5	104,905	84%
FCFS	127,051	88%

	vertices per second	simplification	RMSE (m)
Hegeman et al. [2014]	1,600,000	32%	0.07
Hegeman et al. [2014]	4,300,000	81%	1.90
Rand6	184,430	86%	1.44
Greedy1	37,125	87%	0.14
Greedy5	52,452	84%	0.14
FCFS	63,526	88%	0.18

	simplification	RMSE (m)	max error (m)
Dukai [2020]	99.7%	0.10	8.9
Greedy1	98.5%	0.09	6.7
Greedy5	97.6%	0.09	4.4
FCFS	99.3%	0.09	6.3

TU Delft

Contributions

- Statistics on simplification
- Open-source method for simplifying massive TINs
- FCFS streaming simplification method

Discussion

- Practicality of massive TINs
- Performance of simplification algorithms
- Memory usage

Future Work

- Improving finalizer initialization
- Expanding FCFS
- Quadric error as simplification
- Further modules

TU Delft

References

- Dukai, B. (2020). Full AHN3 TIN using Steiner Points. http://godzilla.bk.tudelft.nl/tin/gpkg/.
- Garland, M. and Heckbert, P. (1997a). Fast triangular approximation of terrains and height fields. Submitted for publication, (October 1999):1–19.
- Garland, M. and Heckbert, P. S. (1997b). Surface simplification using quadric error metrics. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997, pages 209–216.
- Isenburg, M., Liu, Y., and Snoeyink, J. (2006d). Streaming Extraction of Elevation Contours from LIDAR Points. Citeseer, (1):6.
- Peters, R. (2018a). Geographical point cloud modelling with the 3D medial axis transform.
- van Rijssel, L., Dinklo, C., Prusti, M., Giannelli, D., and Hobeika, N. (2020). 3D noise simulation. http://resolver.tudelft.nl/uuid:9e83e3c1-0d7b-4026-a34c-2fbb61aaec2c.

TU Delft