EUROGRAPHICS
CONFERENCE 2024

DISTRIBUTED SURFACE RECONSTRUCTION

Diana Marin, Patrick Komon, Stefan Ohrhallinger and Michael Wimmer

Research Unit of Computer Graphics
Institute of Visual Computing & Human-Centered Technology
TU Wien, Austria

INTRODUCTION

Recent advances in scanning abilities, together with the increase in popularity of digital twins usage, created the tools to create and, respectively, the need to process massive scans of real-life elements. Hence, we are now facing the question of how to quickly process amounts of data that easily exceed millions of points while still creating a faithful representation of the initial point cloud.

To answer this question, we extend a surface reconstruction algorithm [POEM24] based on the Delaunay triangulation to work distributedly on cluster, by taking advantage of the local connectivity required by the algorithm. We split the input into tiles and add a parameter-based padding to each tile to create an overlap. This ensures that no surface data is missing in the final reconstruction while the amount of duplicate information and the data transfer between nodes is minimized.

RELATED WORK

Numerous methods to parallelize the Delaunay triangulation [PMP14, CMYB19] have been developed recently, as it represents a first step in many surface reconstruction methods. The authors introduce various paddings to ensure that the node-computed triangulation agrees with the global triangulation. Our method, even though it uses the Delaunay triangulation, does not require a global triangulation of the input points as it imposes a maximum edge length of triangles in the output, condition we exploit in our work.

Recently, the state-of-the-art surface reconstruction - Poisson Reconstruction [KBH06], has been adapted for out-of-core usage [KH23]. However, our method does not require any preprocessing of the point cloud or additional information such as normals.

METHOD

We adapt the recent curve and surface reconstruction algorithm *Ballmerge* [POEM24] to work distributedly. Its surface reconstruction method has two variants: global and local, both starting from a Delaunay triangulation of the input. While the *Global Ballmerge* merges overlapping regions obtaining a manifold, water-tight mesh, the *Local Ballmerge* filters some of the triangles based on the following criteria: the intersection ratio between their respective Voronoi balls is less than a threshold δ, and the longest edge of the triangle is shorter than a predefined l. We will only concern ourselves with the *Local* version, as it is faster and easier to distribute. Moreover, point clouds with a magnitude of millions are usually scans of outdoor scenes for which, due to scan shadows or resolution, it is infeasible to enforce a manifold output.

SPLITTING

The input data needs to be split to parallelize the reconstruction algorithm. We choose to split the input along a three-dimensional regular grid due to its simplicity and ease of parallelization. Surface reconstruction can now be performed in each grid cell individually. However, edges and triangles might be shared between cells, hence the reconstruction might miss important parts of the surface. To mitigate this, we introduce a padding around each cell.

Since we filter triangles with edges longer than l, we pad the cells with l in the positive directions so no possibly valid triangles are missed in the Delaunay triangulation. Furthermore, we ensure that if points are in different cells, their Voronoi balls intersection ratio is larger than the predefined δ. This is done by padding with $\frac{2l}{\sqrt{(4\delta-\delta^2)}}$ in all directions. Following the original algorithm, we use $l = \frac{d}{200}$, where d is the diagonal length of the input bounding box, and change their default $\delta = 1.85$ value to $\delta = 1.35$ in our experiments, as this value improved the quality of our reconstruction.

DISTRIBUTED VERSION

The regularly subdivided tiles are assigned and distributed to nodes. Each node performs surface reconstruction on its assigned tiles by locally executing *Local Ballmerge*. As points obtained from real-world scans usually are not distributed uniformly, the number of points within tiles may vary a lot. To balance out the load, we assign tiles to nodes using longest-processing-time-first list scheduling, an efficient and good approximation of the optimal solution to the load-balancing problem. Merging is trivially done by taking the union of all local results. Because of the padding, the overall result is guaranteed to be the same as when running *Local Ballmerge* on the entire input. Using n points, p reconstruction nodes and c cores for splitting, our expected run time complexity is $\mathcal{O}(n/c + (n \log n)/p)$ in the best case of uniform point distribution, and $\mathcal{O}(n/c + n \log n)$ in the worst case, as the original method.

RESULTS

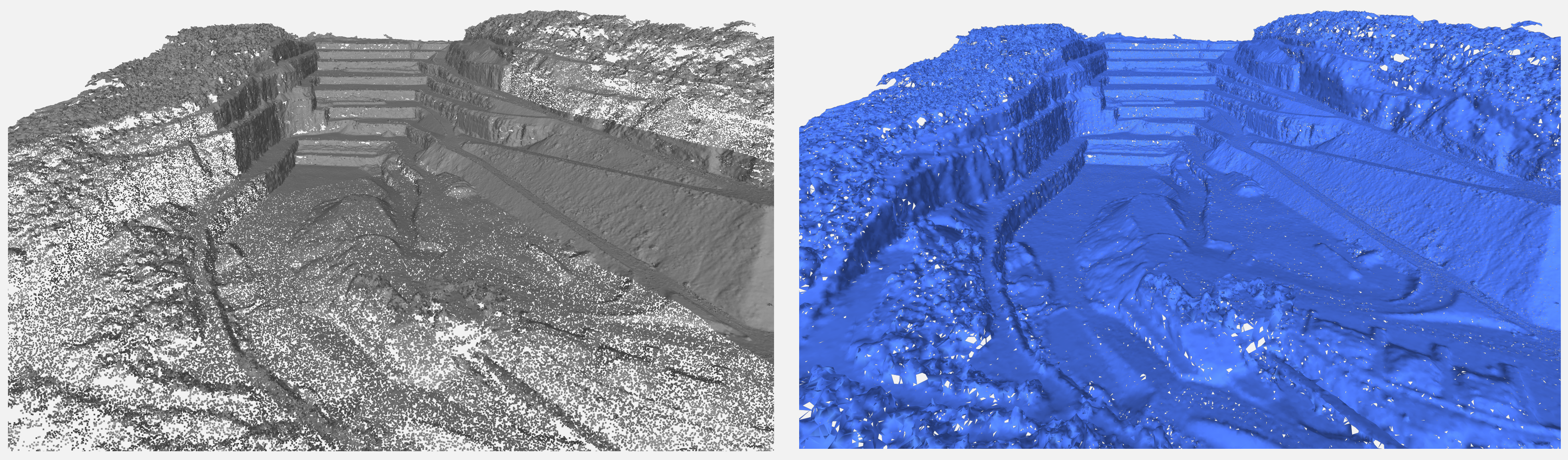

Figure 1: *Reconstructed surface of the eclepens dataset.*

Splitting is done on the GPU using CUDA on a single node while the reconstruction is distributed via MPI to up to 32 CPU nodes. We have evaluated our implementation on the VSC3+ HPC cluster, varying input sizes and numbers of nodes. We used truncated versions of a real-world aerial photogrammetry scan to simulate increasing input size - *Figure 1*.

Increasing the number of nodes with fixed input size and subdivision shows timings close to the linear speedup which is the theoretical maximum. By using 16 nodes and scaling up the input size, the run times of both original and distributed versions increase according to their expected linearithmic behavior.

The distributed version exhibits much slower growth, peaking at a speedup factor of 5 for 32 million points and 32 nodes due to data duplication - *Figure 2*. While varying the number of nodes for a fixed number of points, finer subdivisions allow for better work distribution, resulting in potentially lower run times. However, choosing finer subdivisions results in significantly more duplicated points, eventually canceling out any speed gains from adding more nodes.

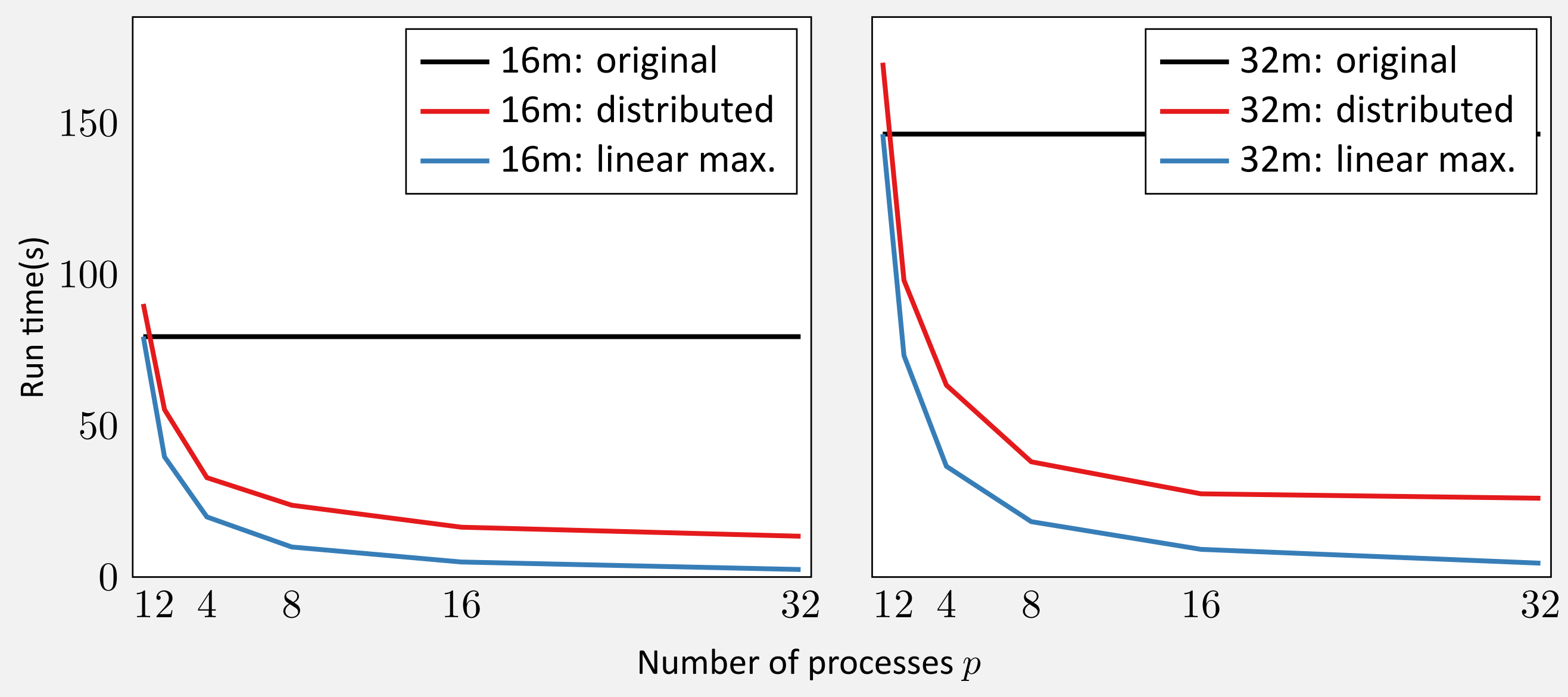

Figure 2: *Run time comparison between original and distributed version using datasets with 16 and 32 million points on 1-32 nodes.*

CONCLUSION

Our method adapts a surface reconstruction algorithm to work distributedly, with minimal data duplication, achieving speedups close to linear for various configurations. We plan to investigate other data-splitting approaches, as well as the possibility of parallelizing the reconstruction on a node level. Moreover, we aim to evaluate our method on more configurations and compare it to recent work.

REFERENCES

[CMYB19] CARAFFA L., MEMARI P., YIRCI M., BRÉDIF M.: Tile & merge: Distributed delaunay triangulations for cloud computing. In 2019 IEEE International Conference on Big Data (2019), pp. 1613–1618.
[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In Proceedings of SGP (2006), SGP '06, p. 61–70.
[KH23] KAZHDAN M., HOPPE H.: Distributed poisson surface reconstruction. In Computer Graphics Forum (2023), vol. 42, p. e14925.
[PMP14] PETERKA T., MOROZOV D., PHILLIPS C.: High-performance computation of distributed-memory parallel 3d voronoi and delaunay tessellation. In SC '14 (2014), pp. 997–1007.
[POEM24] PARAKKAT A. D., OHRHALLINGER S., EISEMANN E., MEMARI P.: Ballmerge: High-quality fast surface reconstruction via voronoi balls. In Proc. of Eurographics, to appear (2024).

TU WIEN