Error detection during construction of buildings: A voxel-based technique

Andreas Gøricke
s153804@student.dtu.dk
Denmark, DTU Compute

Michael Guldborg
s174254@student.dtu.dk
Denmark, DTU Compute

Nicolai M. T. Lassen
s175407@student.dtu.dk
Denmark, DTU Compute

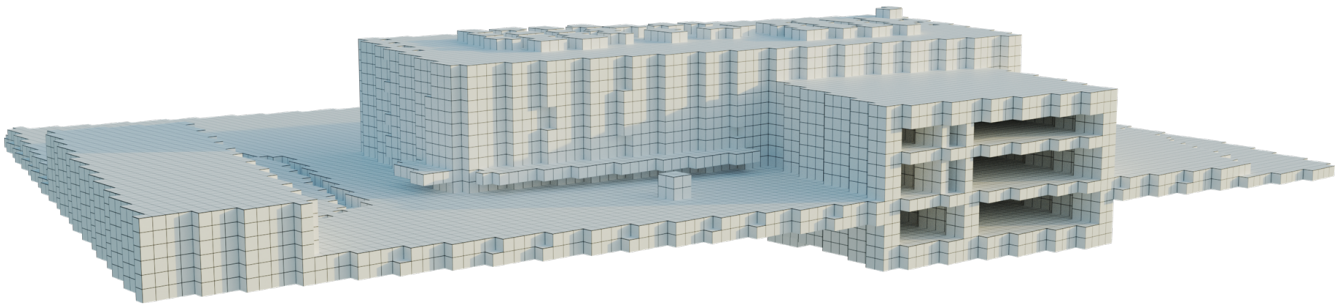


Figure 1. Surface voxel representation of a DTU campus building

Abstract

This paper presents a voxel-based technique for efficient error detection during the construction of buildings. Error detection in large scale building projects poses a set of problems for precision and performance due to the sheer amount of data in mesh topology and 3D scans. With the introduction of a voxel-based representation, this technique overcome the limitations posed by surface complexity for storage and computation. Expanding this representation with known methods of distance fields allows for constant-time lookup of the nearest surface voxel, which optimizes point deviation detection. Experiments on a real-world scenario demonstrate that the distance field search complexity follows a linear distribution $O(N)$, based on the number of points. In addition, the initial distance field voxelization is shown to handle large typologies in ideal resolution, following the results of 475ms for a triangle count of $2.6 \cdot 10^5$ and grid resolution of 1024^3. These results demonstrate the technique's feasibility in large-scale computations.[1]

Keywords: Error detection, distance fields, voxelization, GPU computing, LIDAR, overlap testing

1 Introduction

In the construction industry, technology presents new opportunities for on-time access to knowledge and information. These opportunities can facilitate useful tools that have the potential to make the construction process more efficient and therefore less expensive. One such opportunity involves the process of detecting errors in the building during the construction phase. The Get It Right Initiative estimated in a 2016 report, that building errors account for 5% of direct costs of construction and that the estimate might even be closer to 10%-25% when factoring in the indirect and unmeasured costs [7]. This translates to billions in lost revenue. The business case for developing technology to mitigate these errors is clearly evident.

A solution, depends on clever engineering and advanced technologies. Such technologies include Computer Aided Design (CAD) and Building Information Modelling (BIM) [8]. BIM is an intelligent 3D model-based process that allows project managers and architects to specify the exact requirements and specifications of the building before the construction process is begun. A virtual copy of the building is hereby made. In combination with technologies such as Light Detection and Ranging (Lidar) it is possible to reconstruct a digital version of the ongoing construction and compare it to the virtual model. This phase poses some difficulties. Most urgently the computing power required is significant for large buildings using traditional methods. This paper seeks to examine methods to improve the efficiency of this process. Specifically the use of voxel-based distance fields. Voxel representations are heavily employed in computer graphics. Offering a regular representation, independent of object and surface complexity, making it applicable to a variety of domains [10]. These voxel-based distance fields would theoretically allow a time complexity of $O(N)$ bounded by scan size.

The work presented here is a collaborative effort with Dalux. Dalux[2] provides software to construction companies

[1] Project code https://github.com/NicolaiLassen/dalux

[2] https://www.dalux.com

for continuous evaluation of construction sites, through products such as their Dalux BIM Viewer+ and Dalux field. Dalux plans to integrate error detection into BIM Viewer+ to facilitate further site evaluation.

The following section will address related work for the problem area of distance calculation and mesh approximation. The entire pipeline of error detection is then described along with details of implementation and results.

2 Related work

Research in 3D graphics has proposed many techniques to solve individual parts of this project problem area. This section examines the main areas of efficient 3D distance measurement and triangular mesh approximation.

2.1 3D distance fields

There has been substantial research on 3d distance measurement. Jones et al. [5] presents in the survey from 2006 an outline and comparison of techniques and applications for 3D distance fields. Distance fields are representations that, at each point of the field, tell the distance between that point and the closest point on any object within the domain [5]. In the survey, a method of generating a discrete signed 3D distance field from a triangle mesh was proposed by Bærentzen and Aanæs [3]. Specifically, this proposal seeks to simplify the computation of the sign when generating a signed distance field. In 2005 a more robust method was proposed by Bærentzen [2]. This approach uses graphics hardware to accelerate generation of layered depth images. Using this, binary volume and point representations are obtained that can be used to convert the binary volume into a distance field. This technique is found to be robust for handling holes, spurious triangles and ambiguities. As a means of handling mesh irregularities such as hollow information, robust distance fields are an ideal choice for error detection in construction. Likewise, the constant time lookup is suitable for later point evaluations using this distance grid approximation.

2.2 Voxelization

Voxelization refers to the process of converting a data structure storing geometric information in a continuous domain, such as a 3d triangular mesh, into a discrete approximation grid. A variety of voxelization algorithms have been devised. Fang and Chen [4] suggests constructing a surface voxelization slice-wise, rendering the geometry once per voxel slice while limiting the view volume to this slice. To reduce undesired results of voxelization for thin parts of a topology, Zhang et al. [11] employ an novel graphic accelerated voxelization method that handles these areas with extra processes in the voxelization passe. The presented algorithms allow voxelizers to handle geometries that occasionally lay close together. This technique could be deemed relevant for structures with thin features, though in construction this is seldom the case. So for the interest of this paper, precision need to be evaluated against performance.

Another paper issued in 2010 by Schwarz and Seidel [10] proposes a fast parallel method for surface and solid voxelization on the GPU. In comparison to previous GPU-based algorithms, their conservative voxelization method outperforms them by up to one order of magnitude. The technique employs among other a replacement for the current standard box triangle overlap test Separating Axis Theorem (SAT) [1], the technique envisioned in this paper merely evaluates the edge function checking the result's sign.

The work presented for this paper needs to show the feasibly of error detection in large-scale construction cases as a conceptual process. In this regard, it is deemed relevant for the devised method to allow control and implementation agility.

3 Method

This section details the method used for the error detection algorithm. In figure 2 a overview of this process is visualised. The method involves simplifying the structure of triangular meshes using reviewed voxelization methods. In addition, the section proposes an extension of the common voxel grid representation using distance fields to compute distance deviations for sets of points. This technique enables a fast comparison of the distance between large sets of 3d points and predefined meshes identifying deviation from the original topology. Data used for computation will first be examined.

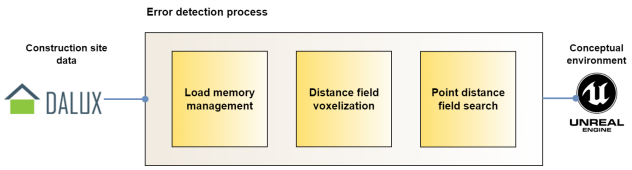


Figure 2. Paper process scope

3.1 Data

The data for this project is provided by Dalux, to fit their current construction software solution. This method uses two main data representations: polygon mesh and point cloud. Polygon mesh files are provided in the Standard Triangle Language (STL), which is a common format for Computer Aided Design (CAD) files. These typologies are provided from building projects to mimic the case.

Point clouds follow the Lidar standard file of pts. These points are gathered with a device, such as a drone. Each point indicating the real-world snapshot of the current bounds of construction. The main concern for these point clouds is the sheer amount of data present for each file. To mitigate this aspect, densification offers performance and quality flexibility through partial file access.

The use of memory-mapped files allows for partial file views. A Memory-mapped file is a portion of virtual memory that is mapped directly byte-for-byte to some portion of the physical file system. The partial view p is calculated from the full view f using the density hyperparameter d.

$$p = f \cdot d \tag{1}$$

(2)

Through a fixed batch size bs the amount of batches can be found, thus

$$b = \left\lfloor \frac{p}{bs} \right\rceil \tag{3}$$

For the mapping of file views batches are uniformly distributed for the full block. Each batch is associated with a handle which can be accessed asynchronous through threading. Figure 3 illustrates this mapping and view handles for some given process. As a result, segments of a given pts file are loaded in place of the whole file, thereby reducing memory consumption.

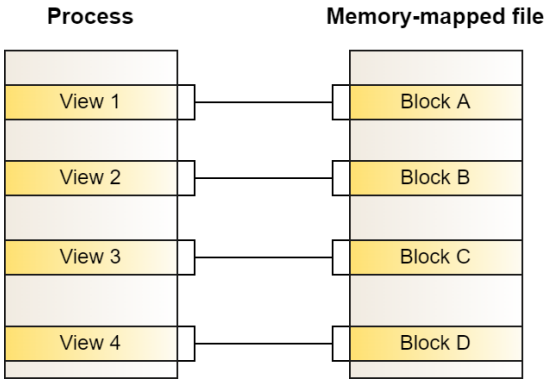
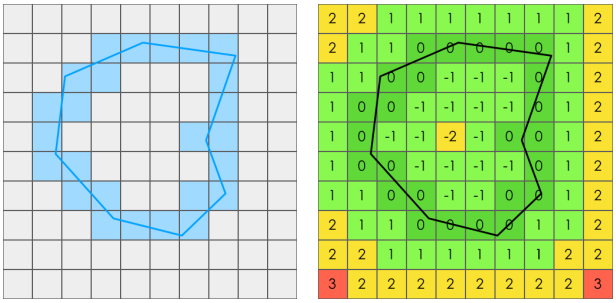


Figure 3. Views to a memory-mapped file

3.2 3D distance field voxelization

For the given scenario of construction site errors, voxelization can be used for the approximation of building CAD files. Voxelization not only allows for fast distance field calculations but can also be useful for visualizations. Some detail is lost when the triangular mesh is converted to voxels, but with high voxel count, this detail loss will be minimal. High voxel count could however also pose challenges to memory consumption.

The method combines the efforts of Schwarz and Seidel [10] for fast parallel voxelization with robust signed distance calculations [2] undergoing the voxelization process (See Figure 4). This allows the method to handle fast distance searches of points while holding the voxel representation for later visual use. For voxelization the aspects of parallelism is used, though this work implements the more simple ray-triangle intersection algorithm by Moller and Trumbore [9] for triangle intercept.



(a) Voxelization **(b)** Distance Field

Figure 4. Cross sections of 2D Conservative surface voxelization (4a) and 2D Signed Distance Field representation (4b)

Time complexity is determined by N triangles and the triangle bounds of $X \cdot Y \cdot Z$, thus the complexity is given $O(N \cdot M^3)$.

The following distance search uses each point for the memory views to lookup the corresponding grid position. For every position a distance to the approximation surface is found. The point is labelled according to a given distance precision threshold for the object class. Thus, the algorithm can detect if a point lays inside or outside of the volume. If this is the case a correspond set of errors is generated. The complexity of this process is a constant operation for each point, giving this a upper bound of $O(N)$, for N points.

4 Implementation

Implementation has been carried out with iterative prototyping. Different methods have been tested for solving subproblems of the overall error detection process. This iterative approach is used to explore the problem space before finalizing an optimized implementation. With this approach, Dalux has been able to provide continuous feedback for improving the implementation.

Point distance field search

The voxel grid being a unsigned distance field indicating the nearest distance to the 3D model. This allows $O(N)$ time complexity for the subsequent distance calculations, where N denotes the amount of points in the point cloud. However, the memory required for the voxel grid is quite substantial, therefore it is only feasible to use this algorithm with small models. The results are shown in table 1 and follows a linear distribution based on the amount of points and the time it takes to calculate the distance to the geometry, which is consistent with the time complexity. A variety of measures can be implemented to improve the accuracy and speed of the algorithm, such as running the code in parallel and on the GPU rather than the CPU. Another step to improve accuracy is to change the structure of the distance field from a uniform to an octree based one, which allows for smaller voxels in

areas with more details while simultaneously reducing the overall memory consumption.

Points	Density	Runtime (ms)
$1.4 \cdot 10^8$	$1 \cdot 0.1^4$	373
$1.4 \cdot 10^8$	$1 \cdot 0.1^3$	3722
$1.4 \cdot 10^8$	$1 \cdot 0.1^2$	35615

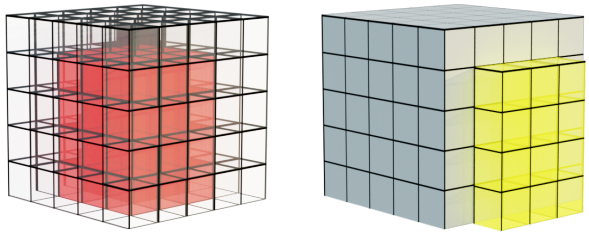
Table 1. Running time (ms) for the error detection in the signed distance field read from dummy array

Dalux BIM viewer+ environment

To comply with the environment presented by the Dalux BIM viewer+ product, the finalist implementation uses the C# language with the .Net core 3.1 framework. Programming interfaces for the OpenCL library have been utilised for GPU acceleration. Subsequently packages such as the OpenGL wrapper OpenTk is introduces for common 3D operations and representations such as multidimensional vector math. For polygon format imports a binary STL parser has been implemented. Memory management of PTS point cloud files is handled asynchronous using the build in library Memory-mapped files [6] as the handle for each file batch.

Visualization

As a conceptual environment, the visualization in this work was carried out in the 3D engine Unreal Engine 5. For visualising errors two types are proposed: inside and outside volume errors, these are respectively colored red and yellow. The simple color scheme helps the user understand what's wrong with the construction. To find errors efficiently transparency for geometry rendering is utilised. This feature is also necessary for finding errors inside geometry. Below a conceptual illustration is given for the two error types, to give a illustrative example a voxel cube is used to depict solid topology.



(a) Inside volume error **(b)** Outside volume error

Figure 5. Visualization of error types against a voxel grid cube

5 Results

The runtime of the implementation is calculated. The results are obtained by running the algorithms multiple times and taking the average score. The GPU instance used for testing is a NVIDIA GeForce GTx 1650 and the CPU is a Intel Core i7-9750h. The results for the voxelization and distance field generation are shown in figure 2. All these calculations are done on GPU instance. The results are good, as is evident by the runtime scores. These calculations are not as time sensitive as the point distance calculations, so the scores are deemed acceptable. The results for the point distance field search are shown in Table 3. The results are decent, albeit they could be improved upon. The Table shows that it takes 416 ms (0,4 seconds) to calculate the distance for 1,4 million points and 9824 ms (9,8 seconds) to calculate the distance for 140 million points.

Voxelization

Triangles	Grid size	Runtime (ms)
$2.6 \cdot 10^5$	256^3	20
$2.6 \cdot 10^5$	512^3	107
$2.6 \cdot 10^5$	1024^3	475

Table 2. Running time (ms) for the voxelization implementation

Point distance field search

Points	Density	Runtime (ms)
$1.4 \cdot 10^8$	$1 \cdot 0.1^4$	416
$1.4 \cdot 10^8$	$1 \cdot 0.1^3$	3104
$1.4 \cdot 10^8$	$1 \cdot 0.1^2$	9824

Table 3. Running time (ms) Point distance field search read from batched memory-mapped file

6 Conclusion and future Work

This paper proposes a voxel based approach for efficiently detecting errors during the construction of buildings. This approach allows for optimizations in distance calculations between 3d geometry and larges sets of points. A prototype was developed before the final approach of developing a model that could integrate with the Dalux BIM viewer+ environment.

Future work could include a stronger focus on visualisation. This is to better encapsulate the user experience of viewing and understating errors for the BIM viewer+. Introducing this through VR or AR would allow construction workers to view and inspect errors in real-time. However, this would impose strict requirements on the algorithms' speed and accuracy to provide a smooth experience.

Acknowledgments

We thank the company Dalux, for the development of this case project and to their contribution with insights and data. A special thanks to Ann-Sofie Fisker and Bent D. Larsen of the Dalux team for their valuable guidance throughout this project.

References

[1] Tomas Akenine-Möller. 2005. Fast 3D Triangle-Box Overlap Testing. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 8–es. https://doi.org/10.1145/1198555.1198747

[2] J. Andreas Bærentzen. 2005. Robust Generation of Signed Distance Fields from Triangle Meshes. *Volume Graphics* (2005).

[3] J. Andreas Bærentzen and Henrik Aanæs. 2002. Generating Signed Distance Fields From Triangle Meshes. *IMM-TECHNICAL-REPORT* (2002).

[4] Shiaofen Fang and Hongsheng Chen. 2000. Hardware Accelerated Voxelization. *Computers Graphics* 24 (02 2000), 433–442. https://doi.org/10.1016/S0097-8493(00)00038-8

[5] Mark Jones, Andreas Bærentzen, and Milos Sramek. 2006. 3D distance fields: A survey of techniques and applications. *IEEE transactions on visualization and computer graphics* 12 (08 2006), 581–99. https://doi.org/10.1109/TVCG.2006.56

[6] Randy Kath. 1993. Managing memory-mapped files. https://docs.microsoft.com/en-us/previous-versions/ms810613(v=msdn.10)

[7] Ed McCann and Tom Barton. 2016. *Get It Right Initiative Improving value by eliminating error.* Technical Report. 1–74 pages.

[8] P. Mesároš, T. Mandičák, M. Behún, and J. Smetanková. 2018. Applications of Knowledge Technology in Construction Industry. In *2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. 367–372. https://doi.org/10.1109/ICETA.2018.8572231

[9] Tomas Moller and Ben Trumbore. 1998. *Fast, minimum storage ray-triangle intersection.* Technical Report 1425. 109–115 pages. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0031624041&partnerID=40&md5=2ad3c15f3085a32fd46e90d0602b35fb cited By 4.

[10] Michael Schwarz and Hans-Peter Seidel. 2010. Fast Parallel Surface and Solid Voxelization on GPUs. *ACM Transactions on Graphics - TOG* 29. https://doi.org/10.1145/1882261.1866201

[11] Zhuopeng Zhang, Shigeo Morishima, and Changbo Wang. 2018. Thickness-aware voxelization. *Computer Animation and Virtual Worlds* 29, 3-4 (2018), e1832. https://doi.org/10.1002/cav.1832 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.1832 e1832 cav.1832.

Appendix

A CONTRIBUTIONS

- Andreas Gøricke, s153804
- Michael Guldborg, s17425
- Nicolai M. T. Lassen, s175407

Article	0	1	2	3	4	5	6
Gøricke	0%	**40**%	20%	**40**%	**33**%	30%	**50%**
Guldborg	0%	**40**%	20%	30%	**33**%	**40%**	0%
Lassen	**100%**	20%	**60%**	30%	**33**%	30%	**50%**

Prototypes	Cloud reduction tests	Voxelization tests
Gøricke	10%	**50%**
Guldborg	**80**%	20%
Lassen	10%	30%

C# program	PTS	STL	Dist Voxelizer	Dist search	voxel parser
Gøricke	20%	10%	25%	**40%**	0%
Guldborg	**60**%	10%	25%	30%	0%
Lassen	20%	**80%**	**50%**	30%	**100%**

Visualization	Voxel & geometry representation	Error visulization
Gøricke	**33%**	0%
Guldborg	**33%**	**80%**
Lassen	**33%**	20%