A SERVICE-ORIENTED INDOOR POINT CLOUD PROCESSING PIPELINE

V. Stojanovic[1]*, M. Trapp[2], R. Richter[3] and J. Döllner[4]

Computer Graphics Systems Group, Hasso Plattner Institute, University of Potsdam
[1]vladeta.stojanovic@hpi.de, [2]matthias.trapp@hpi.de, [3]rico.richter@hpi.de, [4]juergen.doellner@hpi.de

Commission II

KEY WORDS: Indoor Point Clouds, Service Oriented, Processing and Visualization, Digital Twins, Deviation Analysis

ABSTRACT:

Visualization of point clouds plays an important role in understanding the context of the digital representation of the built environment. Modern commodity mobile devices (e.g., smartphones and tablets), are capable of capturing representations in the form of 3D point clouds, with their depth-sensing and photogrammetry capabilities. Points clouds enable the encoding of important spatial and physical features of the built environment they represent. However, once captured, point clouds need to be processed before they can be used for further semantic enrichment and decision making. An integrated pipeline for such processes is crucial for use in larger and more complex enterprise systems and data analysis platforms, especially within the realm of Facility Management (FM) and Real Estate 4.0. We present and discuss a prototypical implementation for a service-oriented point cloud processing pipeline. The presented processing features focus on detecting and visualizing spatial deviations between *as-is* versus *as-designed* representations. We discuss the design and implementation of these processing features, and present experimental results. The presented approach can be used as a lightweight software component for processing indoor point clouds captured using commodity mobile devices, as well as primary deviation analysis, and also provides a processing link for further semantic enrichment of *base-data* for Building Information Modeling (BIM) and Digital Twin (DT) applications.

1. INTRODUCTION

With recent adaptations of Industry 4.0 practices within Architecture, Engineering and Construction (AEC) sectors, the need for routine capture, processing and presentation of digital built environment data is becoming increasingly important. The ability to inspect, monitor, and forecast the current state of the built environment opens new dimensions of stakeholder engagement and enhancement of decision making. These developments particularly concern stakeholders involved with Facility Management (FM) operations, and are becoming part of what is known as *Real Estate 4.0* (RE 4.0). The ability to capture and process historic and current digital data pertaining to the physical and performance related aspects of a given building enables FM stakeholders to assess specific conditions and address any concerns more efficiently, while providing a greater return-of-investment for building management practices (Teicholz et al., 2013).

The most important aspect of these developments is the ability to visually analyze the results of digital *as-designed*, *as-built*, and *as-is* elements, together with related spatio-temporal data and building documentation. The problem of comparing spatial differences between geometric representations of *as-is* or *as-built* versus *as-designed* geometry representations is a common occurrence, especially when comparing point cloud and BIM geometry representations. This may include comparing the inclusion or exclusion of certain building elements when compared to the original *as-designed* built environment representation, as well as the their spatial alignment corresponding to existing documentation (e.g., floor plans).

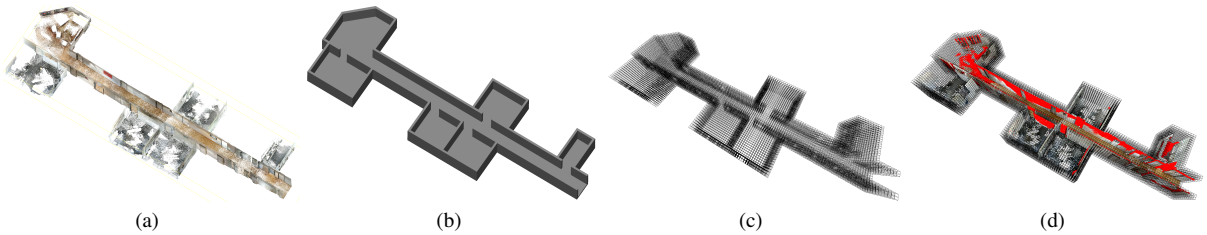
*Corresponding author

1.1 Problem Statement and Research Contributions

In order to carry out the comparison of spatial deviations between different geometry representations of the built environment, the *deviation analysis* process requires the compared geometry to be processed in order to contain specific attributes. This includes the generation of an additional finite element representation of the *as-designed* BIM geometry, as well as the post-processing of the compared point cloud geometry. Post-processing of captured point cloud geometry is required for their further use for semantic enrichment and geometry comparison operations (Qu et al., 2014).

Such point cloud post-processing tasks commonly include computation of normal vectors, segmentation, reconstruction and transformation operations. Once post-processed, the point cloud representation of a specific indoor environment can be compared against the finite element version of its *as-designed* BIM geometry counterpart for spatial deviations (Fig. 1). The generation of a finite element representation of BIM geometry can be accomplished with the use of voxelization-based geometry processing methods. While all of these processes can be implemented as separate software solutions and components, or using existing software, an integrated pipeline for such processes is crucial for use in larger and more complex enterprise systems and data analysis platforms.

We present a system design and prototypical implementation for a service-oriented point cloud processing pipeline, and demonstrate it's application for deviation analysis (Fig. 2). We discuss the design and implementation details, and present experimental results obtained using the prototypical system implementation. The presented results focus on *as-is* point cloud versus *as-designed* BIM geometry deviation analysis, and we discuss the potential for using the prototypical



(a) (b) (c) (d)

Figure 1. Example of point cloud and BIM geometry processing required for deviation analysis. (a) The captured indoor point cloud, (b) The extracted BIM geometry mesh, (c) The voxelized version of the BIM geometry mesh, and (d) Aligned point cloud with the voxelized BIM mesh, with spatial deviations highlighted in red.

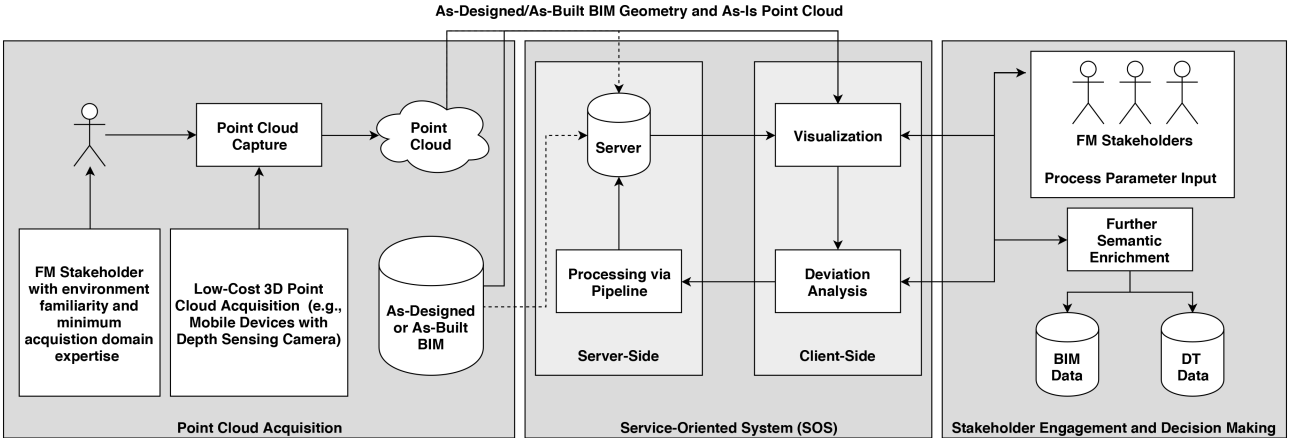


Figure 2. A high-level overview of the associated processes for capture, processing and visualization of indoor point clouds. These processes and components form the main parts of the proposed SOS-based processing pipeline for indoor point clouds representations, and further semantic-enrichment and stakeholder engagement operations.

implementation for further semantic-enrichment of *as-is* indoor point cloud representations within the realm of RE 4.0.

2. FOUNDATIONS AND RELATED WORK

Point Cloud Processing. The ability of point clouds to capture the physical state of the built environment is one of the key advantages of point cloud representations. Challenges arise when attempting to make use of point clouds as they do not feature any semantics by default, thus requiring semantic-enrichment (Poux et al., 2017). However, prior to semantic enrichment point clouds need to be processed. This processing includes at the most minimal levels the generation of point normals and segmentation of homogeneous point regions, followed by optional reconstruction as either simplified models reconstructed from bounding regions of segmented clusters, or fully triangular or boundary representation mesh reconstruction (Macher et al., 2017).

Grillit *et al.* provide an overview of common segmentation methods for laser scanned and photogrammetrically generated point clouds (Grilli et al., 2017). Such methods are suitable for segmentation of both indoor and outdoor point clouds, and rely on either the spatial coordinates, pre-computed point normals or RGB color values of a given point set in order to calculate distinguishable regions of homogeneous point clusters. A notable method for segmentation that we make use of is called *Region Growing*, and relies on iterative sampling of neighbouring points for determining the segmented point clusters, making use of either pre-computed point normals or colors properties, which are iteratively searched within a voxelized data structure (Bassier et al., 2017). The success of region-growing segmentation approaches is largely dependent on the initial point sampling size, thresholds for color variance, and the accuracy of the normals estimation required to compare the regions. Computation of point normals is most commonly computed using neighbourhood sampling methods, such as *k*-nearest neighbours (KNN) (Mitra, Nguyen, 2003).

Reconstruction of segmented point clouds can be accomplished using axis or object-oriented bounding boxes. Anagnostopoulos *et al.* describe a simple, but effective, approach for reconstruction of primary building objects (e.g., walls, floors and ceilings) (Anagnostopoulos et al., 2016). Since we are interested in reconstructing the key structural features of a given building, using the coordinates of the object-oriented bounding boxes of segmented indoor point cloud clusters provides a quick and efficient approximation for the generation of the primary boundaries of a given indoor space.

Deviation Analysis. Deviation analysis plays an important role in the approximation and visualization of spatial differences between *as-designed*, *as-built* and *as-is* digital built environment representations. This can include approximation of planar surfaces for measuring deviations between *as-built* and *as-is* BIM versus point cloud representations for core structural elements such as non-curved walls (Stojanovic et al., 2018). Anil *et al.* used an existing commercial software tool to import corresponding *as-is* BIM and point cloud representations, in order to perform comparative deviation analysis — using a computationally expensive method based on minimum Euclidean distance comparison (Anil et al., 2013).

Bosché and Guenet presented a set of methods for evaluating surface regularity, based primarily on matching points from a 3D point cloud to a *as-built* BIM component representation. The points of the 3D point cloud are matched to the corresponding BIM geometry using either surface proximity (measuring the orthogonal distance of a point to the BIM geometry surface), or using surface normal similarity (comparing the precomputed normal vectors of the each point to the normal vectors of the BIM geometry surface) (Bosché, Guenet, 2014).

Research by Turkan *et al.* looks at using deviation analysis methods for tracking of construction progress on building sites, specifically looking at detecting primary and secondary components from point clouds (Turkan et al., 2014). They present three techniques for deviation analysis, first two of which are based on earlier research by Bosché and Haas (2008) — based on aligning the *as-built* BIM geometry within the corresponding point cloud, where the range between the *as-is* and *as-designed* point/triangle intersection is measured using ray tracing, while their third method uses the bounding volumes of negative areas of each BIM geometry component to detect a number of points contained inside in post alignment (Bosche, Haas, 2008).

Bassier *et al.* evaluated the use of a Finite Element Analysis (FEA) method for structural analysis of heritage timber roof structures (Bassier et al., 2016). They compared two aligned models and checked for deviations in the post-registration phase - one was a simpler wireframe model and the other was a complex discretized FEA mesh. They noted the benefit of using a more complex geometry discretization scheme for increased deviation analysis accuracy, as opposed to using the least complex geometric representations often featured in BIM geometry models.

Wang *et al.* used deviation analysis for comparing an *as-is* point cloud captured using a remote UAV and the corresponding *as-designed* BIM model (Wang et al., 2015). They recommend the use of different registration methods for alignment of comparative data, particularly the zone fitting algorithms described by Choi and Kurfess - assuming no matching coordinate information is shared between the two data sets (Choi, Kurfess, 1999). The deviation analysis is then performed automatically using the selected Navisworks BIM software.

Bonduel *et al.* describe the use of *as-is* point cloud and *as-built* BIM comparisons for detecting micro and macro spatial changes (Bonduel et al., 2017). They make use of the CloudCompare software tool (Girardeau-Montaut, 2015), to carry out the primary deviation analysis for detecting macro spatial changes, and use a custom plugin for removing certain components (e.g., windows), from the IFC BIM model prior to detecting micro damage. The open-source CloudCompare software tool can compare point cloud versus point cloud, as well as point cloud versus mesh spatial deviations — generating both visual and numerical deviation analysis results using an octree-based approach with a linearly interpolated color map to visually highlight any spatial deviations.

It can be observed from the cited research that there is a paucity for flexible and computationally efficient deviation analysis solutions. The two biggest challenges are 1) Registration of the two data sets that are to be compared in the same coordinate system and transformation frame, and 2) Efficiently calculating the deviations for hundreds of millions of points to the nearest matching geometry surface. The first challenge can in most cases be solved using a selected point registration scheme, such as the Iterative Closest Point (ICP) (Men et al., 2011).

The second problem is more complex to solve, as it requires the use of a geometry discretization and point-to-polygon comparison scheme. Based on the literature review, we have opted for using a voxel-based deviation analysis approach, as it offers the best trade-off between geometry discretization complexity and deviation computation performance. A voxelized representation of an *as-built* or *as-designed* mesh can be generated by evaluating the shape and projection properties of the triangular mesh that is used to generate a 3D voxel grid (Eisert, 2005). This voxelized representation of the mesh can be computed, at varying resolutions, most commonly using octrees (Hornung et al., 2013) — and can preserve most of the geometric features of the original polygonal mesh, making it useful for approximating deviations of non-rectified or curved geometry. The use of a voxelized mesh representation is similar to the style of mesh representation used for FEA. Our general deviation analysis approach approach is influenced by earlier deviation analysis methods focusing on visualizing spatial deviations between *as-is* point cloud and *as-built* BIM models of indoor spaces (Stojanovic et al., 2018).

Service Oriented Processing and Visualization. Service Oriented Systems (SOS) architecture make a clear separation between the server and the client. This enables the design of software systems with a distinct advantage of being able decouple hardware requirements from client devices that may not have specific hardware capabilities (Klimke, 2019). This may include capabilities to process the analytics from the integrated data sources, generate complex visualizations, and to integrate non-monolithic software components for streaming to mobile devices up to enterprise-level applications (Discher et al., 2019).

The use of web-based graphics programming APIs and frameworks can further enhance the visualization outputs generated on or streamed to client devices. Notable Web3D technologies such as WebGL, and high-level frameworks such as Three.js (Cabello et al., 2010), can be used to visualize important scene elements that can enhance stakeholder engagement and decision making for RE 4.0 applications (e.g., deviation analysis results, or a BIM-based 3D scene with corresponding annotations) (Yan et al., 2011). Such systems for visualizing BIM models, IoT-enabled (Internet of Things) smart home visualization, and point cloud for robotics applications have been described by (Zhang et al., 2014), (Pouke et al., 2018) and (Toris et al., 2015). The combined use of Web3D-based graphics frameworks and SOS architecture allows for processing and visualizing of indoor point clouds and associated semantics, capable of running on different client hardware configurations (including mobile devices). Such system architecture and Web3D frameworks used in this research have been described and evaluated previously (Stojanovic et al., 2019).

3. APPROACH

3.1 Point Cloud Capture

Modern commodity mobile devices with depth sensing capabilities and sufficient digital camera resolutions can capture

a point cloud using the depth-estimation provided by a time-of-flight (TOF) and stereo-vision capture and reconstruction methods. Notable research by (Angladon et al., 2018), (Senthilvel et al., 2017) and (Kalyan et al., 2016) all describe case studies where a Google Tango compatible mobile tablet or phone were used for capture of point clouds of outdoor and indoor building areas. The common census among the cited papers is that while the captured point clouds were not of high enough quality to be useful for accuracy-critical applications, they are more suitable for applications where the accuracy threshold error is tolerated. Furthermore, the authors also noted the low cost, ease of use, high portability of such devices as major benefits for routine point cloud capture for various FM-related applications.

The capture of indoor point clouds requires the operator of such a mobile scanning device to walk around a given indoor space and sequentially scan each of the surfaces and devices. The average recorded time taken to complete a scan of a $20m^2$ room is approximately 15 minutes. In most cases, the operator would typically start at the entry point to the room, and walk around the room in a clockwise or anti-clockwise manner while scanning items of interest. This would typically first involve scanning the walls, floor and ceiling, than all of the furniture items. Depending on the memory capacity of a given mobile device and the size of a given room/area, multiple scans may be required. A common limitation to this capture approach is that lighting conditions can affect the quality of the capture (e.g., environment is too dark, too reflective or has non-distinctive edge features). The application that we used for capture of the point cloud was the DotProduct Dot3D mobile app[1]. Using the Dot3D app on an ASUS Zenfone AR mobile device, the user has to point the mobile device camera in the direction of a given area and wait while the software processes the given frame.

The captured 3D point cloud also needs to be filtered for overlapping duplicate points (this is performed by the Dot3D app). As multiple scans are required for large areas, the final point cloud representation needs to be merged using partial post-capture scans that have been registered and aligned. We accomplished this with the CloudCompare software tool, where we parsed multiple partial scans of a given indoor area and aligned them manually using both floor plan images as references, as well as visual anchors within the point clouds themselves (e.g., an item that is distinguishable and featured in all related scans, which is used as a reference to transform and correctly align the related partial point cloud scans).

In most cases, the point cloud can also be sub-sampled to around 100 000 points per $20m^2$, in order to decrease processing time. We perform this using a random point selection scheme, where we set a given point removal sub-sampling value in the range from 0.0 to 1.0, in order to approximate the amount of original points that will be left remaining.

3.2 Service-Oriented System

Geometry Processing. The main processing components of the pipeline are implemented as command line tools that are executed on the server. The command line tools include segmentation and voxelization tools. The segmentation tool makes use of the Point Cloud Library framework (Cousins, Rusu, 2011), in order to perform segmentation operations on a given point cloud. The segmentation operations include computation of point normal vectors, segmentation using *region growing*, and generation of axis-aligned (AABB) and object-oriented (OOBB) bounding boxes generated from the segmented point clusters. The Binvox voxelization command line tool is used to generate a voxelized representation either of reconstructed *as-is*, or *as-built* or *as-designed* triangular mesh geometry (Min, 2004 - 2019). This voxelized mesh representation can then be aligned and compared against the *as-is* point cloud representation in order to highlight any spatial deviations.

The use of a SOS architecture enables scalable integration of key processing components, software frameworks and data sources for varying application requirements. The scalability of a SOS solution depends on the kind of client the end result is sent to, and/or further processed on - this includes commodity hardware personal computers and mobile devices. The server-side command line tools are invoked by an Express server, implemented using Node.js[2]. The server is able to process requests from the client web-application using real-time bi-directional communication enabled through the Socket.IO[3] JavaScript library. A custom command-line tool used for segmentation and bounding-box reconstruction based on PCL was implemented in C++, with the ability to directly pass user-defined parameters when invoked. Since natively compiled C++ runs faster than interpreted JavaScript, process-intensive tasks such as segmentation, can be accomplished faster and have better memory management options (thus being able to process larger point cloud scenes) (Smedberg, 2010).

Deviation Analysis. Deviation analysis is used to record both numerically and visually the spatial differences of overlapping geometric elements when comparing BIM and point cloud representations of the same built environment elements. In our case, we evaluate and visualize how close a cluster of points of a given *as-is* point cloud is to the overlapping voxel element of an voxlized *as-designed* BIM model in the same 3D space. In turn, the deviation threshold value is used to determine beyond what threshold (measured as distance in Euclidean space), we consider a 3D point to be deviating. This value can be adjusted by the user or based on the required use-case specific parameters. The deviation threshold value allows us to set an acceptable fault tolerance when comparing different geometric and primitive-type representations.

With voxelization we can approximate the general shape of the *as-designed* 3D geometry, including irregular boundaries, by fitting a number of $n \times m \times k$ voxel elements within the given polygonal shape boundaries. This then gives a voxelized representation of the *as-designed* 3D geometry that we can use to compare against the *as-built* 3D point cloud for deviation analysis. The voxel mesh generated by the Binvox tool is exported in the .MSH file format (a simple native file format used by the Gmsh FEA tool[4], where each center point of a voxel is recorded. The actual size of the voxels calculated by finding the common difference between XYZ coordinates of each current and next voxel element.

The accuracy of the deviation analysis based on comparing the point to voxels is correlated to the resolution of the the voxelised mesh. If the voxelized mesh is too coarse, certain

[1] https://www.dotproduct3d.com/dot3dedit.html

[2] https://nodejs.org/en/

[3] https://socket.io/

[4] http://gmsh.info/

geometry features may be omitted or too simplified to create an accurate enough deviation analysis comparison with the point cloud. However, the higher the resolution of the voxel mesh is, the longer it takes to compute the deviation analysis comparison between the intersecting points. The resolution for the computed voxel mesh is set by the user, and referred to as the *approximate resolution* - since the voxel fitting algorithm tries to adjust the voxel resolution along the width, height and depth based on the single parameter value. A good balance between accuracy and resolution is usually use-case dependent, but in most cases resolution should be high enough to feature all of the extruded, protruded and non-regular geometry features that are required for deviation analysis (Fig. 3).

The voxelization methods use by the Binvox tool are based on the *parity count* and *ray stabbing* methods (Nooruddin, Turk, 2003). The ray stabbing method is preferable for non-organic geometry that has intersecting components (e.g., double walls featured in building models), as the depth sampling only samples the initial and final ray sections along a given direction (that in turn is sampled multiple times in different directions for each polygon within a voxel grid). This allows for more accurate generation of complete voxel models (e.g., voxel models where the inside of the model is voxelised, rather than just the *shell* of the model).

Furthermore, we make use of a greedy algorithm for generation of voxelized geometry data used for visualization and deviation comparison, meaning that the entire set of the voxelized *as-designed* mesh is rendered. Since the captured point cloud representation is usually projected along a given plane (e.g., point clouds do not contain volume, but rather just surface representations), the use of a full-volume voxel models allows us to compare all points that might otherwise be missed if they were located in void space within the voxelized mesh. While such a greedy voxelization method is practical to implement for visualization of voxelized geometry, it is impractical for real-time 3D viewing of larger and more complex models, thus a more efficient polygon-based geometry rendering method can be used if rendering speed is a requirement. Such methods can represent each voxel as six-sided quad or cube, and check where intersecting quads or triangles are present in order to merge them — thus greatly improving rendering performance by removing redundant polygons that are not seen by the user.

For each voxel element, we check if it contains a point from the *as-is* 3D point cloud. If the point is contained in an voxel, we can mark the point as non-deviating, otherwise the point is marked as deviating. Since the voxel elements are represented as bounding boxes, we can test to see if they contain a point inside them or not. We first test to see what points from the complete point cloud are inside the bounding box, which we then copy to a temporary array — along with an additional integer key value to indicate which specific points in the point cloud array are copied. We then splice the complete point cloud array using these key values in order to obtain a new temporary array containing the deviating points. These deviating points are then added as a new point cluster to the scene and marked visually.

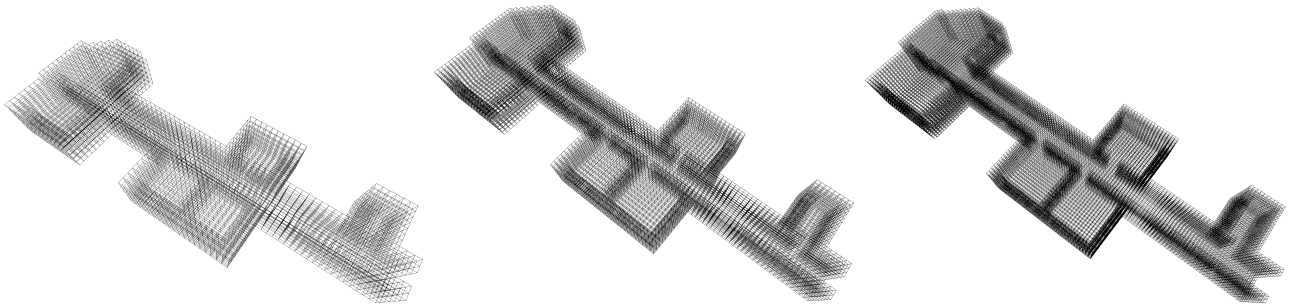
Apart from the binary deviation, we also need to take into account if the deviation is present as a surface damage or erosion element (e.g., damaged, missing or eroded elements). We call this the *point sparsity* deviation analysis. In order to assess point sparsity, we must first determine the average number of points contained in each voxel, and check to see if each voxel contains a number of points above this average threshold. If the case is that the voxel contains a number of points below the average threshold, we can assume that the missing points present a deviation. Voxels that are completely empty can also be marked as deviation, as these represent elements that are present in the *as-designed* geometry by not in the *as-is* 3D point cloud representation.

Web3D Visualization The visualization is implemented client-side, in the form of a prototypical web-based application programmed using HTML5 and JavaScript. We make use of the Three.js Web3D framework for the majority of the 3D visualization tasks. The use of the programmable graphics pipeline enables point cloud geometry parsing and generation methods for visualizing each of the processing results. In the 3D scene, the point clouds are parsed in the PLY file format, where the coordinates and vertex colors are parsed and assigned to each newly generated point material sub-object of the complete point cluster object. Using the default shader for point cloud materials, we can also change the color and opacity of any selected point group during run-time of the application. One limit of Three.js for visualizing point clouds is the lack of support for out-of-core rendering of massive amounts of point-cloud data, and therefore it can only be used to visualize point-cloud scenes in real-time with approximately 4.5 million points, without resorting to the use of more sophisticated scene and memory management methods (Discher et al., 2019).

3.3 Stakeholder Engagement and Decision Making

Process Parameter Input. Using the web-based client tool, users are able to adjust each of the segmentation, reconstruction and deviation analysis parameters, and interactively view and inspect the generated results sent by the server. This includes setting parameter such as the sub-sampling and scaling factors for the point cloud, region growing segmentation parameters (min/max point sampling, surface curvature threshold, k-nearest neighbour sampling size), and the desired voxel resolution size for the voxelized version of the *as-is* OOBB mesh used for deviation analysis comparison. These parameters are sent as socket data packets to the server, which in turn parses them either as string or numerical data-type parameters for the command-line tools. Certain domain expertise concerning point cloud representations and processing is expected among the users who input the parameters, but the ability to tie such parameters to a GUI potentially allows for instructive and educational use as well. We make use of common file formats for exchanging point clouds and triangular geometry data (PLY and XYZRGB file format for the point cloud, and OBJ for the triangulated geometry models such as AABB and OOBB reconstructions).

Further Semantic Enrichment. The segmented point cloud and reconstructed AABB/OOBB geometry representations can further be used for generating base-data for *as-is* BIM and DT representations. The generation of such data requires further semantic enrichment of either the processed point cloud, or the reconstructed AABB/OOBB geometry representations. Semantics generation are a key requirement for the generation of usable data for further analysis. The process of semantic enrichment is used to add *context* to the processed base-data, by introducing understandable labels for each of the segmented and/or reconstructed elements of a given indoor environment representation. Semantics can either be added by the stakeholders — by directly annotating a given point cloud



(a) (b) (c)

Figure 3. Example of progressively increasing voxelized mesh, based on the corresponding BIM geometry mesh. (a) 3608 voxels (approximate resolution of 64^3), (b) 8345 voxels (approximate resolution of 96^3), (c) 14604 voxels (approximate resolution of 128^3).

or reconstructed object, or it can be automatic with the use of deep learning classification methods. A specific version of our prototype application used for semantic enrichment of indoor point clouds is presented and described in (Stojanovic et al., 2019).

4. CASE STUDY

For the testing of the pipeline we make use of point cloud representations of a typical office environment consisting of a hallway with five connected rooms and a communal sitting area extension. The point clouds used for testing were captured using a Google Tango specification compatible mobile phone (ASUS ZenPhone AR). The capture was completed during day time under natural lighting conditions. Windows and highly-reflective surfaces were not fully captured, and were thus excluded from the scan (though for the case study the part of the room with large windows is treated as a solid wall). A 3D mesh representation of the given room was manually modeled based on the original floorplan layout, and this served as an synthetic *as-designed* version of the room that is used for the deviation analysis comparison.

The initially captured point cloud of the office area was further manually edited where noisy and partially scanned clusters without significance were manually removed. We also segmented out the ceiling, furniture objects and any partially open office doors. The point cloud originally contained 2 506 858 points, but was sub-sampled to 501 372 points to decrease processing time. The voxelized BIM geometry mesh used for comparison contains 14604 voxel elements (approximate resolution of 128^3).

4.1 Empirical Deviation Analysis Results

We evaluated the implement deviation analysis approach, using binary and point sparsity outputs to highlight spatial differences between the compared *as-is* and *as-designed* geometry. We compared voxlized BIM geometry against the corresponding point cloud for obvious deviations (Fig. 4), where deviating points are highlighted in red. Point sparsity deviation analysis is also performed - highlighting voxels with eroded or missing deviations in red, while *healthy* voxels are highlights in blue (Fig.5). Additionally we also present preliminary reconstruction results for a smaller office area, the point cloud of which was captured separately and used the test region growing segmentation and AABB/OOBB reconstruction capabilities of the prototypical pipeline application (Fig. 6). This office area consists of 222 084 points. Finally, we present the preliminary performance results for the deviation analysis approach. We measured the average computation taken (in milliseconds), for the two different deviation analysis methods to be performed client-side. The average time taken to perform the binary deviation analysis was 645368 milliseconds, while the average time taken to perform the point sparsity deviation analysis was 452215 milliseconds. For generating the final visualization results, we used a commodity laptop with an Intel i5 1.8 GHz CPU, 8 GB RAM, and NVidia GeForce MX150 GPU with 2 GB video memory, running the FireFox 67.0 web browser.

5. DISCUSSION AND CONCLUSIONS

The presented approach enables the processing and spatial analysis of captured indoor point clouds for manifold applications in BIM, FM — specifically O&M operations. The use of a service-oriented paradigm, focusing on modular, light-weight software components, enables the processing indoor point clouds captured using commodity mobile devices. This also removes the dependency on using third party monolithic software tools for important tasks such as segmentation, voxelization and deviation analysis. Using our approach, we can quickly approximate spatial deviations between voxelized *as-designed* BIM geometry and corresponding *as-is* point clouds that are aligned in the same 3D space. Server-side processing has also been implemented for more computationally expensive tasks, such as region-growing segmentation and simple reconstructions using bounding-box approximations. The use of a front-end web-based GUI enables user adjustment of processing parameters, thus making the our approach adaptive to various indoor point cloud representations. This can also encourage users to experiment and learn what the best parameter configurations are for their specific needs. Additionally, with Web3D-based visualization, the users can interactively inspect and annotate the point cloud results for various processing and analysis stages. We did not focus on performance optimizations of our approach, instead focusing on demonstrating the feasibility of our prototypical pipeline implementation. For future work, we plan to investigate the the use of a multi-directional raycasting method for more accurate deviation analysis, as well as interpolated color value mapping to the visualized deviating points — in order to visualize the distance the points are deviating from the nearest surface. Finally we plan to involve FM stakeholders for further user-centered testing of the client-side web application.

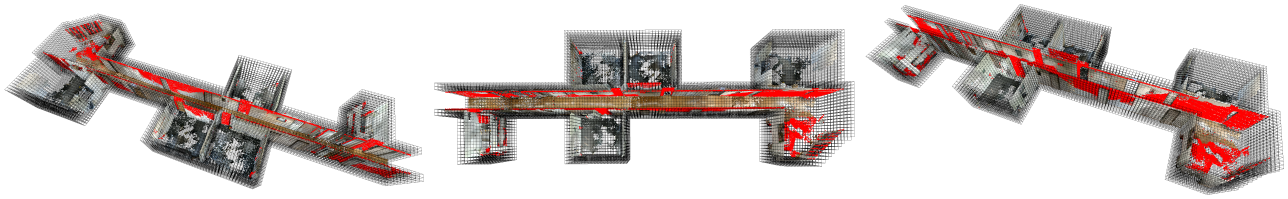


Figure 4. Experimental visualization results for the binary deviation analysis, with detected deviating points colored in red.

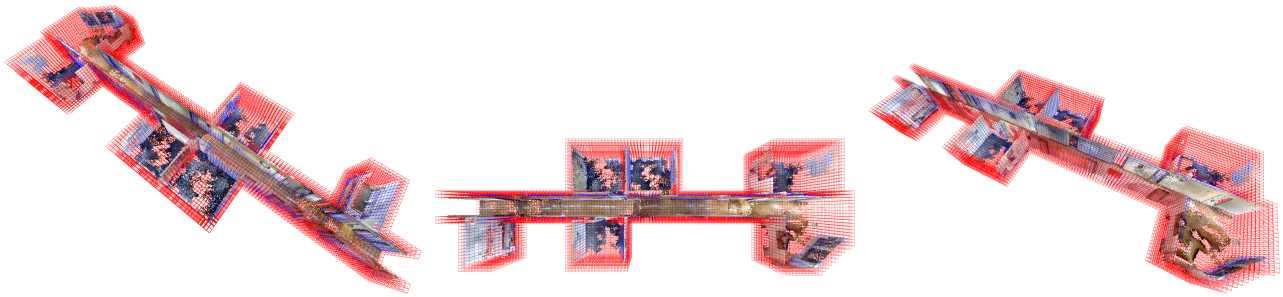


Figure 5. Experimental visualization results for point sparsity deviation analysis, with voxels containing below average number of points highlighted in red, and those that are *healthy* highlighted in blue.

ACKNOWLEDGEMENTS

This work has been partially funded by the Research School on *Service-Oriented Systems Engineering* of the Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany.

REFERENCES

Anagnostopoulos, I., Belsky, M., Brilakis, I., 2016. Object boundaries and room detection in as-is bim models from point cloud data. *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering, Osaka, Japan*, 6–8.

Angladon, V., Gasparini, S., Charvillat, V., 2018. Room floor plan generation on a project tango device. *International Conference on Multimedia Modeling*, Springer, 226–238.

Anil, E. B., Tang, P., Akinci, B., Huber, D., 2013. Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data. *Automation in Construction*, 35, 507–516.

Bassier, M., Bonduel, M., Van Genechten, B., Vergauwen, M., 2017. Segmentation of Large Unstructured Point Clouds Using Octree-Based Region Growing and Conditional Random Fields. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42(2W8), 25–30.

Bassier, M., Hadjidemetriou, G., Vergauwen, M., Van Roy, N., Verstrynge, E., 2016. Implementation of scan-to-bim and fem for the documentation and analysis of heritage timber roof structures. *Euro-mediterranean conference*, Springer, 79–90.

Bonduel, M., Bassier, M., Vergauwen, M., Pauwels, P., Klein, R., 2017. Scan-to-bim output validation: Towards a standardized geometric quality assessment of building information models based on point clouds. *5th International Workshop LowCost 3D-Sensors, Algorithms, Applications*, 42, Copernicus GmbH, 45–52.

Bosché, F., Guenet, E., 2014. Automating surface flatness control using terrestrial laser scanning and building information models. *Automation in construction*, 44, 212–226.

Bosche, F., Haas, C. T., 2008. Automated retrieval of 3D CAD model objects in construction range images. *Automation in Construction*, 17(4), 499–512.

Cabello, R. et al., 2010. Three. js. *URL: https://github. com/mrdoob/three. js*.

Choi, W., Kurfess, T. R., 1999. Dimensional measurement data analysis, part 1: a zone fitting algorithm. *Journal of manufacturing science and engineering*, 121(2), 238–245.

Cousins, S., Rusu, R. B., 2011. 3d is here: point cloud library (pcl). *IEEE International Conference on Robotics and Automation, Shanghai (China)*.

Discher, S., Richter, R., Döllner, J., 2019. Concepts and Techniques for Web-based Visualization and Processing of Massive 3D Point Clouds with Semantics. *Graphical Models*, 101036.

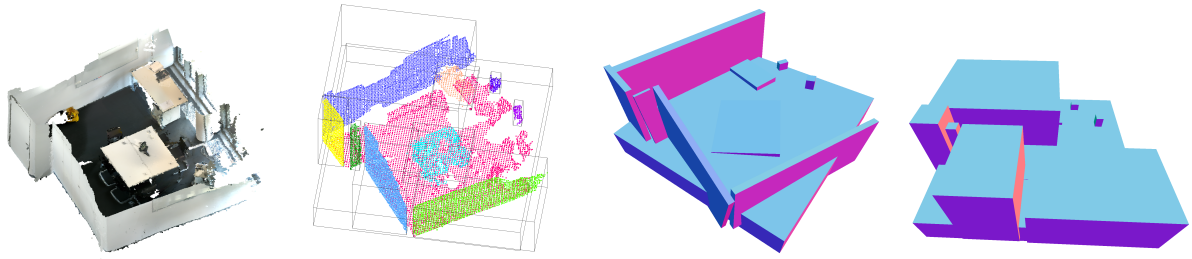
Eisert, P., 2005. Reconstruction of volumetric 3d models. *3D Videocommunication: Algorithms, Concepts and Real-Time Systems in Human Centred Communication*, 133.

Girardeau-Montaut, D., 2015. Cloud compare3d point cloud and mesh processing software. *Open Source Project*.

Grilli, E., Menna, F., Remondino, F., 2017. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42, 339.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., Burgard, W., 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34(3), 189–206.

Kalyan, T. S., Zadeh, P. A., Staub-French, S., Froese, T. M., 2016. Construction quality assessment using 3D as-built



(a) (b) (c) (d)

Figure 6. Examples of point cloud processing operations, including (a) The original point cloud, (b) Segmentation using region growing, (c) Generation of OOBBs, and (d) Generation of AABBs.

models generated with Project Tango. *Procedia Engineering*, 145, 1416–1423.

Klimke, J., 2019. Web-based provisioning and application of large-scale virtual 3D city models.

Macher, H., Landes, T., Grussenmeyer, P., 2017. From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences*, 7(10), 1030.

Men, H., Gebre, B., Pochiraju, K., 2011. Color point cloud registration with 4d icp algorithm. *2011 IEEE International Conference on Robotics and Automation*, IEEE, 1511–1516.

Min, P., 2004 - 2019. binvox. `http://www.patrickmin.com/binvox` or `https://www.google.com/search?q=binvox`. Accessed: 2019-10-07.

Mitra, N. J., Nguyen, A., 2003. Estimating surface normals in noisy point cloud data. *Proceedings of the nineteenth annual symposium on Computational geometry*, ACM, 322–328.

Nooruddin, F. S., Turk, G., 2003. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2), 191–205.

Pouke, M., Virtanen, J.-P., Badri, M., Ojala, T., 2018. Comparison of two workflows for web-based 3d smart home visualizations. *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, IEEE, 1–8.

Poux, F., Neuville, R., Hallot, P., Billen, R., 2017. Model for reasoning from semantically rich point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, 107–115.

Qu, T., Coco, J., Rönnäng, M., Sun, W., 2014. Challenges and trends of implementation of 3d point cloud technologies in building information modeling (bim): case studies. *Computing in Civil and Building Engineering (2014)*, 809–816.

Senthilvel, M., Soman, R. K., Varghese, K., 2017. Comparison of handheld devices for 3d reconstruction in construction. *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, 34.

Smedberg, F., 2010. Performance analysis of javascript.

Stojanovic, V., Richter, R., Döllner, J., Trapp, M., 2018. Comparative Visualization of BIM Geometry and Corresponding Point Clouds. *International Journal of Sustainable Development and Planning*, 13(1), 12–23.

Stojanovic, V., Trapp, M., Richter, R., Döllner, J., 2019. Service-Oriented Semantic Enrichment of Indoor Point Clouds using Octree-Based Multiview Classification. *Graphical Models*, 101039.

Teicholz, P. et al., 2013. *BIM for facility managers*. John Wiley & Sons.

Toris, R., Kammerl, J., Lu, D. V., Lee, J., Jenkins, O. C., Osentoski, S., Wills, M., Chernova, S., 2015. Robot web tools: Efficient messaging for cloud robotics. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 4530–4537.

Turkan, Y., Bosché, F., T. Haas, C., Haas, R., 2014. Tracking of secondary and temporary objects in structural concrete work. *Construction Innovation*, 14(2), 145–167.

Wang, J., Sun, W., Shou, W., Wang, X., Wu, C., Chong, H.-Y., Liu, Y., Sun, C., 2015. Integrating BIM and LiDAR for real-time construction quality control. *Journal of Intelligent & Robotic Systems*, 79(3-4), 417–432.

Yan, W., Culp, C., Graf, R., 2011. Integrating BIM and gaming for real-time interactive architectural visualization. *Automation in Construction*, 20(4), 446–458.

Zhang, X.-Y., Hu, Z.-Z., Wang, H.-W., Kassem, M., 2014. An industry foundation classes (ifc) web-based approach and platform for bi-drectional conversion of structural analysis models. *Computing in Civil and Building Engineering (2014)*, 390–397.