Geometric Optimization of Building Information Models in MEP Projects: Algorithms and Techniques for Improving Storage, Transmission and Display

Zhen-Zhong Hu[a,b,*], Shuang Yuan[b], Claudio Benghi[c], Jian-Ping Zhang[b], Xiao-Yang Zhang[d], Ding Li[d], Mohamad Kassem[c]

a Graduate School at Shenzhen, Tsinghua University, Shenzhen, Guangdong, 518055, China
b Department of Civil Engineering, Tsinghua University, Beijing, 100084, China
c Department of Mechanical and Construction Engineering, Northumbria University, Newcastle, United Kingdom
d CBIM Beijing Co., Ltd., Beijing, 100084, China

Abstract

Mechanical, Electrical and Plumbing (MEP) models are generally characterized by information redundancy and a high density of irregularly shaped components. Consequently, they require large storage spaces and are not conducive for interchange purposes. Geometric optimization of MEP models can play a significant role in facilitating model exchange and handover by increasing storage, transmission and display efficiency. To date, the body of knowledge on such geometric optimization, unfortunately, is still very narrow.

This paper presents and tests a solution for the optimization of storage, transmission, and display of MEP models. Storage optimization was achieved through a mapping-based model description method and a novel Quadric-Error-Metric (QEM) mesh simplification algorithm, reducing required storage while maintaining the contour of components. For transmission optimization, a normal vector compression algorithm and a fixed-dictionary specific compression algorithm were proposed to achieve efficient compression of data thus, fulfilling the need for cross-platform interchange. Display optimization was obtained through a normal vector regeneration algorithm for clustering triangle meshes to improve 3D display effects.

The evaluation of the solution, performed for each individual component separately as well as for an entire solution, proved successful. Volume of storage data was reduced by 40% to 50% through mesh simplification. Data transmission volume was reduced by more than 80% for components with complicated geometry without affecting the topology of the components. Finally, the display process was capable of decreasing the number of triangles and delivering very good quality of displayed models.

Keywords: BIM; MEP; geometric model; storage; transmission; display

1. INTRODUCTION

MEP (Mechanical, Electrical, and Plumbing) Engineering refers collectively to the engineering of water supply, drainage, heating, ventilation, air-conditioning, electricity, intelligence designs,

energy saving, etc. in buildings [1]. As an integral part of buildings, MEP systems can contribute up to 50% of the total investment in some complex large-scale building projects [2]. MEP systems have a significant influence on the safety, operational efficiency, energy consumption, and structural design flexibility of buildings, and are key to the creation of an agreeable indoor environment.

Traditionally, the time-consuming and labor-intense process of producing MEP information has been predominantly paper-based and as a result prone to human errors and delays [3][4]. MEP deliverables, when compared with other architectural and structural deliverables, are characterized by (i) an extremely large number of components, (ii) high complexity due to their strong interrelations with other engineering domains, and (iii) spatial constraints in general, especially when addressing design changes due to restrictions by other systems. To address these challenges, BIM (Building Information Model/Modelling) technologies have been increasingly adopted in MEP projects. BIM is a digital representation of physical and functional characteristics of a facility, a shared knowledge resource for facility information forming a reliable basis for decisions during its life-cycle [5], and the current expression of digital innovations within the construction sector [6]. The spatial distribution and logical relationships between different systems represented in a MEP building information model can be useful for information management purposes at the operation and maintenance stage [7].

BIM models of MEP systems can reach large sizes due to their complex irregular geometry, density of components, and redundancy of information. As a result, cross-platform exchange of such models during the project phases in general and handover to operation in particular is challenging. On the other hand, the exchanging and transmission of geometric information is of great importance during the life-cycle of a MEP system, as geometric models serve as an important auxiliary in the implementation and maintenance of MEP systems. Though in the MEP industry, semantics data are procured for accurate implementation and maintenance parameters, geometric models prove more directly perceivable and intuitive for facility managers, and are thus also often indispensable. Moreover, in typical BIM application scenarios, semantics data appear attached to individual components, and usually are procured upon user requests. Consequently, the transmission of semantics data typically features much smaller data volume and higher frequency compared to geometric data, which can be handled with relative ease. Geometric data, however, usually need be stored, transmitted and displayed in whole, lacking separability and divisibility compared to semantics data, thus causing pressure due to their immense volume and extremely high heterogeneity level. Therefore, effective measures to optimize the storage, transmission and display of geometric MEP models is of both great importance and value.

Techniques for geometric optimization of building information models in general and for MEP models in particular are sparse in literature. This paper presents and tests a solution for the geometric optimization of MEP building information model. The solution adopts an IFC-Based Model Description, and comprises of: (i) storage optimization algorithms employing a similarity mapping algorithm and a mesh simplification algorithm; (ii) transmission optimization using spherical compression algorithms, fixed-dictionary and GZIP compression, and (iii) transmission and display optimization through a normal vector regeneration algorithm. The paper is organized as follows: Section 2 provides a summary of the related studies from

the literature; Section 3 presents the IFC-based model description for efficient 3D-display; Section 4 describes the proposed solution whereby each subsection is focused on one individual component of the entire solution; Section 5 presents the results of the solution testing in real applications and discusses its limitations, and finally the research is concluded in Section 6.

2. LITERATURE REVIEW

BIM and MEP related research has mainly focused on modelling and design coordination, and facilities management. Khanzode et al. used a case study to analyze the challenges teams face in implementing BIM tools and processes for MEP coordination [2]. Leite et al. analyzed the modelling effort required at different Level of Details (LoD) and the impact of LoD on modelling effort and clash detection [8]. Tabesh and Staub-French used a case study to evaluate the 3D coordination process, and identify and classify building system coordination knowledge [9]. More in depth researches are available on the factors [10] and procedures [11] influencing design coordination, and on the development of coordination platforms [12]. BIM applications in relation to MEP were also explored in prefabrication during construction and modular applications [13].

Research into BIM for facilities management have started to attract significant attention. However, very few studies are solely focused on MEP. Becerik-Gerber et al. presented an overview of the application areas and value BIM brings to facilities management [14]. Kassem et al. demonstrated the benefits and challenges of implementing BIM in facilities management using a large university complex as a case study [15]. Schevers et al. reported on the technical aspects associated with developing a digital facility model for the Sydney Opera House using neutral standards (i.e. IFC) [16]. Motawa and Almarshad proposed a knowledge-based BIM system for assisting in building maintenance [17]. Xie et al. explored the applications of BIM across the project lifecycle and in particular the role of BIM in developing and managing MEP models and information [18]. Ko [19] and Motamedi and Hammad [20] explored the use of Radio Frequency Identification (RFID) technology to enhance building maintenance. Chen et al. have proposed BIM-based systems for MEP emergency management [21]. Chi et al. discussed the possibility and prospect of integrating augmented reality technologies with BIM to facilitate facility management [22]. Kalasapudi et al. reported a method of automated spatial change analysis for MEP components using 3D point clouds and BIM models [23].

Today, the most prominent standard for data exchange in the building industry is the Industry Foundation Classes (IFC) [24]. Another notable non-proprietary data format is the Construction Operations Building Information Exchange (COBie) [25] which serves the operation and management of existing buildings. Classification systems for the organization, sorting, and searching of information are also very important. Key classification systems currently include the OmniClass Construction Classification System (OCCS) [26], which is a full-life-cycle information classification standard for facilities, and Uniclass, which is a unified classification for the UK industry covering all construction sectors [27]. For the storage and expression of models, key BIM servers include IFC Model Server [28], EDM Model Server [29], BIMServer [30] etc., which support the storage and management of model data and browsing of the model.

The literature demonstrates BIM investigations in relation to MEP are on the rise. However,

there is still a dearth of studies addressing geometric compression and storage optimization. This gap is remarkable given the increasing complexity of construction projects generally and MEP work packages particularly. MEP are increasingly more complex, larger in size, and requiring cross platform interchanges. This challenge, according to [31], warrants attention as the requirements for efficient display and data interaction have exceeded the capabilities of existing technology. The lack of such methods may compromise the exploitation of valuable information - which has been already created during design and construction - in facilities management. It is within this context that this paper sets off to develop and test algorithms for optimizing storage, transmission and display of MEP models.

3. IFC-BASED MODEL DESCRIPTION FOR EFFICIENT 3D-DISPLAY

A complete MEP model includes information about a significant amount of building systems and elements; each is characterized by a kernel of identity and geometry data that can be extended with further datasets throughout the design, construction, and facility management to serve decision making across the project lifecycle. An efficient information storage is particularly important in the MEP domain due to the large number of instances found in models and the complexity and variability of geometric shapes. Spatial data must be organized and structured to optimize their further visualization and querying. Different strategies exist for the digital definition of geometric shapes. Data models such as IFC are not optimized for storage and their highly structured geometry is heavy in terms of computing cost, making it unsuitable for operations involving a large number of objects in a query [32]. The IFC schema, however, defines Boundary Representations (BREP), primitive geometry (e.g. profile sweeping), and Constructive Solid Geometry (CSG) approaches to achieve the task. Since each of these methods has unique indispensable characteristics, a multi-form storage mode was adopted in this research: primitive geometry and CSG hold essential abstract information such as lengths, surface areas and profile shapes that are required for collaboration. BREP, a richer form that uses restored topological faces that are reverse engineered from the triangulated, is necessary to obtain the geometries of arbitrary shapes. Since each digitalization strategy provides unique benefits, the developed system provides a data structure that enables the use of the most appropriate for each specific purpose. The MEP model's information storage and management architecture are~~is~~ shown in Figure 1.

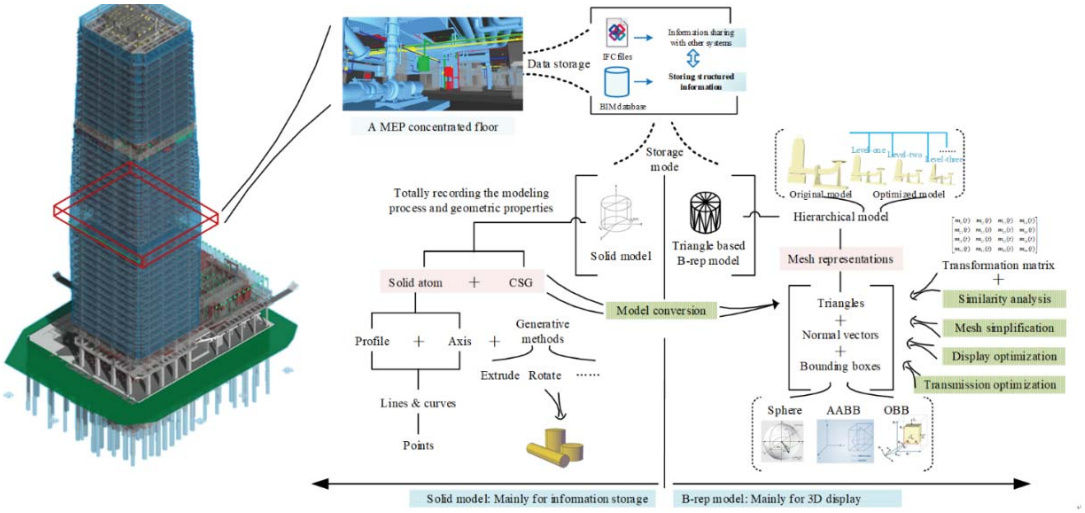

Figure 1. Architecture of Storage and Management of MEPIM

According to this architecture, model information is stored in two forms, namely IFC files and BIM databases, the former is intended for information sharing purposes with other relevant information systems and actors, while the latter is used to structure model information for storage purpose. When available, solid models based on primitives provide greater accuracy and precision for element shapes; however, their interpretation requires complex geometric algorithms that render its usage inappropriate for real-time applications, particularly on slower devices such as handheld systems. In these occurrences, the MEPIM databases also stores BREP triangulated models, which more directly suit 3D visualization libraries such as OpenGL[1] and Microsoft DirectX[2].

Boolean and meshing operations on solids are necessary for the CSG representation. Yet Boolean operations, namely union, difference, intersection, etc. involve complex calculations to yield even more complex results, and are inherently expensive and time-consuming. To address this challenge, the system is designed to perform these operations in advance and cache their results for real-time operations. Hence, the designed data structures are capable of retaining simplified models at different levels of optimization alongside the original geometry. This requirement however increases the demand on model storage which would become impracticable without optimization.

For instance, in the case of the Hedong Station of Guangzhou Metro – used as a testbed later on in the paper – the number of triangles of the original model reached 10 million and occupied nearly 2 GB of storage space, while Hedong Station is but an ordinary station amongst its many counterparts in Guangzhou. Due to the multi-layered storage mode and the need to manage numerous stations, efficiency of information storage is of paramount importance. To deal with

[1] OpenGL is a software interface used to produce high-quality, computer-generated images and interactive applications using 2D and 3D objects, bitmaps, etc.

[2] Microsoft DirectX a collection of application programming interfaces (APIs) for handling tasks related to multimedia, especially game programming and video, on Microsoft platforms.

substantial model geometric data, this research proposes a series of storage, transmission, and display optimization techniques for the triangle-based BREP storage mode of MEP project models. The objective is to achieve optimized storage and efficient transmission without compromising the display effects of such models.

4. STORAGE OPTIMIZATION ALGORITHMS FOR MEP MODELS

Storage optimization is a critical step in the process of MEP model optimization. First, a mapping-based model technique for storage optimization based on similarity analyses is proposed in sub-section 4.1. This technique aims to significantly reduce the number of similar components stored to a level where the overall data volume is shrunk by 20% to 40% compared with the original. Second, an improved QEM mesh simplification algorithm for complicated components is presented in sub-section 4.2 with the objective of reducing the number of triangles stored while preserving the basic outline of the mesh. Finally, to fulfill the requirement of efficient 3D model display especially in cross-platform transmissions, a series of model compression and display algorithms are introduced in sub-section 4.3.

4.1 Model Optimization through Similarity Mapping

Reusing internal libraries of geometric components multiple times is beneficial to the storage and visualization performance due to the instancing features available in most 3D graphics libraries [33]. The concept of representation mapping entails the storage of the geometric description of similar objects in a single copy that is then referenced multiple times through transformation matrices to determine their placement in the building. The IFC schemas provide ways to achieve this across multiple elements. However, not all design tools implement and use this feature sufficiently resulting in unnecessarily large model files.

An element similarity algorithm has been designed to resolve these occurrences and further increase the mapping-based storage optimizing technique. The algorithms perform similarity analyses on components using mesh similarity-matching methods, and merge sufficiently similar components in the representation.

Mapping-based model expression requires as a prerequisite that similarity analyses is conducted on existing models. There are plentiful of model similarity algorithms currently available. Based on the exploited shape characteristics of 3D models, they could be categorized as one of outline-based, topology-based, or vision-based. These algorithms achieve mutable results under different circumstances and are not flawless, as summarized in Table 1.

Table 1. Comparison of Model Similarity Algorithms

Category	Algorithm	Description	Drawback
Outline-Based	Statistical Histogram[34]	Performs statistics of feature information on vertices and meshes of 3D models to generate histograms, and then determines	Only the overall distribution of vertices is reflected. Fails to account for local outlines, resulting in relatively low

		similarity based on distances between histograms. Easy and fast.	matching accuracy.
	Extended Gaussian Image[35]	Maps each mesh surface of the 3D model to a vector on the extended Gaussian sphere, and then measures similarity based on the outputted extended Gaussian image.	Dependent on the coordinate system of the 3D model. Low robustness to model noises, tessellation, and mesh simplification.
	Functional Analysis[36]	First volume pixilates 3D models to meet the demands for functional analyses. Commonly used ones include Fourier analyses, spherical harmonic analysis, etc.	Relatively legion restrictions on models. Results largely dependent on volume pixilation methods and analyzing functions used.
Topology-Based[37]	Reep Graph Comparison	Calculates topological structure of 3D models from connected domains, and then compares to determine similarity.	Relatively strict demands on models, resulting in low computability. High computational complexity, unable to efficiently compare outlines of 3D models.
	Axes Comparison	Calculates topological structure of 3D models from skeletons, and then compares to determine similarity.	
Vision-Based	Projection Matching[38]	Calculates visual projection images on different directions, and then analyzes their shape characteristics to determine similarity.	Projection on a large number of directions have to be compared, resulting in relatively low efficiency.

These algorithms can deliver expected results under certain circumstances but rather they have weaknesses in terms of computability, operability or efficiency. Besides, they are not designed to be adapted to specific problems in the domain of construction industry. Construction projects, especially MEP projects, are usually characterized by: (i) Designed in orthogonal coordinate systems, with relatively rare rotations at random angles; (ii) The number of components to be matched is enormous, placing a significant emphasis on algorithm efficiency, and (iii) Scaling, rotation or affine projections usually cause rendering efficiency to decrease. Based on these considerations, only coincidences after translation are considered in the proposed process of similarity analysis, excluding coincidences after rotations, scaling or affine projections. The mesh similarity-matching algorithm proposed by this research is illustrated in Figure 2.

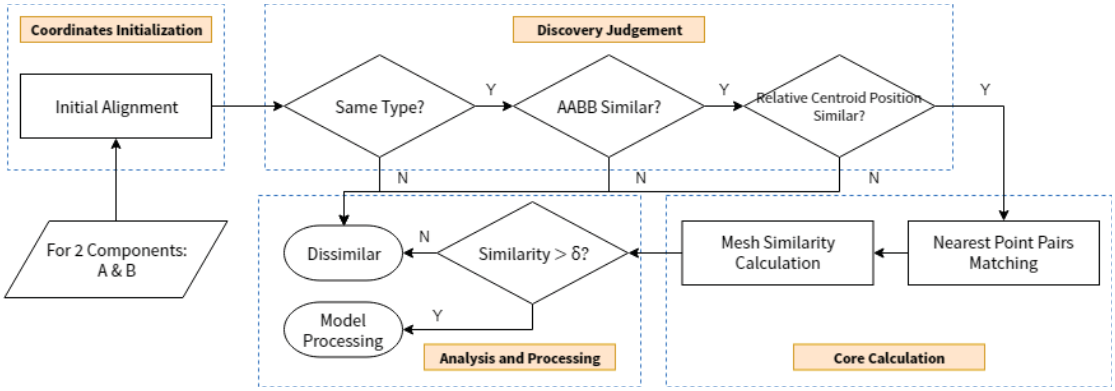

Figure 2. Mesh Similarity Matching Algorithm of Different Components

The inputs to the algorithm are two the components to be compared, denoted A and B respectively. The polygon meshes that constitute the two components are denoted accordingly as Mesh A and Mesh B. The algorithm's stages include:

- **Coordinates Initialization:** this stage finds the center point of the components, and then translates the center points to the origin, so that the two components are equally calibrated. Here the center point of a mesh refers to the center of its axis-aligned bounding box. All vertices coordinates would be expressed relative to their center point in this algorithm, excluding any scaling.
- **Discovery judgement**: This stage has a number of steps. The first step (i.e. component type comparison) compares two components. If they are of different types, the algorithm terminates and components are determined dissimilar. The second step (i.e. AABB comparison) compares the size difference between the ~~axis~~Axis-A~~a~~ligned B~~b~~ounding B~~b~~oxes of the components with the threshold. If the threshold is exceeded, the algorithm terminates and the components are determined dissimilar. The third and last step (i.e. centroid position comparison) compares the difference between the relative positions of the centroids in the bounding boxes and the threshold. If the difference exceeds the threshold, the algorithm terminates and the components are determined dissimilar.
- **Core calculation:** For each vertex in mesh A, the algorithm identifies its nearest neighbor in mesh B that minimizes the weighted error of distance and normal vectors, and thus generates a list of nearest neighbors. This operation is also repeated the other way around between mesh B and mesh A generating another list of nearest neighbors. –Then, the algorithm estimates the similarity between the two lists of nearest neighbors.

Assume there are two meshes, A and B, with P: $\{P_1, P_2 \dots P_m\}$ and Q: $\{Q_1, Q_2 \dots Q_n\}$ as their respective list of vertices. For each vertex in mesh A, P_i, find its nearest neighbor in mesh B, Q_j that minimizes $D(P_i, Q_j)$ to draw a list of nearest neighbors − ListA: $\{(P_i, Q_j), i=1, 2, \dots, m\}$. $D(P_i, Q_j)$ is the weighted ~~error~~ sum of the distance and the angle difference between the normal vectors of the two vertices and can be calculated using Equation 1.

$$D(P_i,Q_j) = \omega_1|P_i - Q_j| + \omega_2 \cdot arccos\left(\boldsymbol{N}_{P_i} \times \boldsymbol{N}_{Q_j}\right) \times l_{average} \quad (1)$$

The first term in Equation 1 is the Euclidean distance between the two vertices, and the second item measures the difference between the normal vectors at the two vertices, where $l_{average}$ refers

to the average size of the mesh, and could be calculated as the average of the length, width, and height of the bounding box of the component. ω_1 and ω_2 are the weights, which could be assigned 0.8 and 0.2 respectively. The selection of the weights affects the contribution of the Euclidean distance and the normal vector difference in the total difference measurement. There is no apparent logic for a reasonable contribution distribution between these 2 terms, and the choice of the actual value of the weights are subject to heuristics. The experiments carried out in this research show that the values 0.8 and 0.2 are an acceptable solution, but these weights can be tweaked according to actual performance and demand as an algorithmic parameter in implementation. The normal vector at a vertex is taken as the weighted average of the normal vectors of triangles containing the vertex. This calculation is repeated for each vertex in mesh B, Q_k, to find its nearest neighbor in mesh A, P_s, and produce the list of nearest neighbors − ListB: $\{(Q_k, P_s), k=1, 2, \ldots, n\}$.

Next, the weighted average squared distance between vertices of the two meshes (i.e. SD) is calculated. Vertices in the mesh vary in importance. For example, a vertex that describes a local detail has smaller neighboring triangles, and is of less importance. To better measure the similarity of the two components, it is prudent to use the sum of the areas of neighboring triangles, A_i and A_k, as weights when calculating SD, as defined in Equation 2.

$$SD = \frac{\Sigma_{i=1}^{m} A_i \times |P_i - Q_j|^2 + \Sigma_{k=1}^{n} A_k \times |Q_k - P_s|^2}{\Sigma_{i=1}^{n} A_i + \Sigma_{k=1}^{n} A_k} \tag{2}$$

When calculating the similarity, SD is compared with the average squared distance between the vertices to account for the sizes of the meshes. The similarity between two triangle meshes is defined as in Equation 3.

$$similarity = exp\left(\frac{-SD \times (m+n)}{\Sigma_{i=1}^{m} |P_i|^2 + \Sigma_{j=1}^{n} |Q_j|^2}\right) \tag{3}$$

If the similarity between two triangle meshes exceeds a preset threshold, δ, then it is determined the two components are similar, and the model data will therefore be mapped. The value of the preset threshold δ could be tweaked according to actual conditions, where a larger value stands for a lower tolerance for matching errors, and vice versa. The experiments assigned the value 0.95 as the threshold, which yielded quality results within the context of these specific experiments, and appeared acceptable to most practical usage. Still, the value should be adjusted according to actual demand and performance.

Mapping-based algorithms are designed to reduce the redundant storage of coinciding or similar geometric data in the same model. However, in MEP projects, due to the large numbers of components with irregular shapes and complex geometry, storage optimization could be challenging if it is based solely on similarity mapping. To address this challenge, in the next sub-section a set of algorithms and schemes are proposed for the geometric simplification of such components.

4.2 Mesh Simplification of Triangle-Based BREP Model

Mesh simplification refers to deleting or modifying geometric elements in a model that have relatively small influence on the overall shape, and aims to reduce the model size while keeping the perceived change from the original shape minimal [39]. A variety of mesh simplification algorithms exist in literature, with some of the influential ones categorized and described in Table 2.

Table 2. Main Mesh Simplification Algorithms

Category		Description
Geometric Element Deletion	Vertex Collapsing[40]	Delete geometric elements from the mesh, by collapsing vertices, edges or triangles, to simplify the mesh.
	Edge Collapsing[41]	
	Triangle Collapsing[42]	
Vertices Clustering[43]		Subdivide the mesh into cells. Replace all vertices in the same cell with one new vertex, and update edges and triangles accordingly.
Area Merging[44]		Take one triangle as a seed, and continually merge it with triangles around under a certain merging condition.
Wavelet Decomposition[45]		Decompose the mesh into a low-resolution approximation part using low-pass filtering, and a detailed part using high-pass filtering. Simplify the mesh by dropping the detailed part.

To complement the mapping-based storage optimization method of the whole model explained in the previous section, and further optimize the storage of complicated components, an improved Quadric Error Metrics (QEM)-based edge-collapsing algorithm was proposed. The original QEM-based algorithm by Garland and Heckbert [41] sorts the edges by sorting errors in an ascending order and collapsing the edge associated with the minimum error. However, feature vertices could possibly be deleted in this process, which would drastically change the topology of the model or even result in holes or overhangs. To preserve as much geometric features of MEP components as possible and prevent overly large simplification error, the proposed improved algorithm introduces: (i) Surface division and associated points recognition; (ii) Model feature point recognition and preprocessing, and (iii) Accumulated error measurement. The shortcoming of the original algorithm and the proposed features are explained and demonstrated in the following paragraphs.

–4.2.1 Surface Division and Associated Points Recognition

Points with the same coordinates but different normal vectors would be treated as two separate points in triangulation by this algorithm. Examples are ‘5’ and ‘12’, and points ‘6’ and ‘13’, etc. in Figure 3. With the original algorithm, the associated edge of a collapsed edge may not collapse along, resulting in holes or overhangs, as shown in the lower-left corner of Figure 3.

To prevent large errors like holes or overhangs and to have a better control on the topology of the simplified model, a per-surface edge-collapsing algorithm based on the original QEM-based algorithm was proposed. First, the proposed new algorithms preform surface division on the model through triangle indexing matching to determine whether triangles belong to the same surface by their vertices’ indices. For example, triangle (1, 2, 3) and triangle (1, 3, 4) share

common indices '1' and '3' and therefore, they belong to the same surface. However, they have no mutual indices with triangles (8, 9, 16) or (9, 10, 16) and hence, they are not in the same surface. According to this rule, the model in Figure 3 would be divided into three surfaces, namely Surface 1 (1, 2, 3, 4, 5, 6, 7), Surface 2 (8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21) and Surface 3 (22, 23, 24, 25, 26, 27, 28). Indexing association would be established between vertices such as '1' and '8', etc., as shown in the right of Figure 3. Then, simplifications would be performed per-surface. Whenever an edge in one surface is collapsed, its associated edges would be searched for in other surfaces to be simultaneously collapsed. For example, if edge 8-9 in Surface 2 were to be collapsed, its associated edge 1-2 in Surface 1 would be collapsed too. The benefits of this algorithm are twofold: i) the levels of simplification of surfaces can be controlled to prevent some surfaces from being over-simplified; and ii) holes and overhangs can be prevented, preserving the topology of the original model and improving the simplification quality.

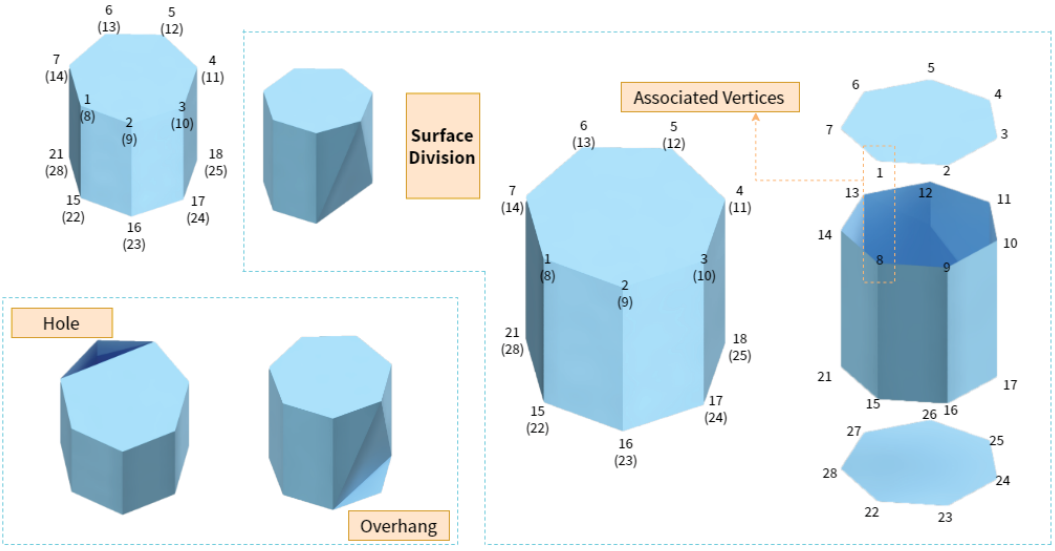

Figure 3. Preprocessing of Surface Division

4.2.2 Model Feature Point Recognition and Preprocessing

Model feature point recognition and preprocessing refers to the classification of model vertices into ~~*simple*~~ *enclosed* vertices and ~~*complex*~~*open* vertices according to their geometric and topologic properties, and further into *reducible* vertices, *lagging reducible* vertices, and *irreducible* vertices (Figure 4).

If the opposite edges of a vertex in all the triangles containing it form a closed 3D polyline, and all the edges linked to the vertex are shared by exactly two triangles, then the vertex is classified as an *enclosed* ~~a *simple*~~ vertex. ~~*Simple*~~ *Enclosed* vertices are further classified into *ordinary* vertices, *inner* ~~*edge*~~ vertices, and *poignant* ~~*angular*~~ vertices according to the geometric properties of the mesh in their neighborhood. The common edge between two triangles is classified as a *feature edge* of the model (See bold lines in Figure 4) if the dihedral angle exceeds a designated threshold. If no feature edge is identified in the set of edges containing an ~~*simple*~~ *enclosed* vertex, then the vertex is classified as an *ordinary* vertex. If exactly two feature edges are found, the vertex is classified as an *inner* ~~*edge*~~ vertex. Finally, if only one or more

than two feature edges are identified, the vertex is classified as a *poignantn* ~~angular~~ vertex (Figure 4). A non-~~*simple*~~ *enclosed* vertex is labelled an *open* ~~complex~~ vertex, which either belongs to an edge that is not a common edge of exactly two triangles, in which case labelled a *complex* vertex, or its opposite edges cannot form a closed polyline, categorized as a *boundary* vertex.

Vertices dictate the choice of the edge to collapse in the process of edge collapsing. ~~These~~To fulfill such role, the types of vertices are further categorized into *reducible* vertices, *lagging reducible* vertices, and *irreducible* vertices (Figure 4). This geometric and topologic classification of vertices will guide the edge collapsing process in deciding the order and the direction of collapsing, ensuring correct treatment of feature points and preventing large distortion of model topology.

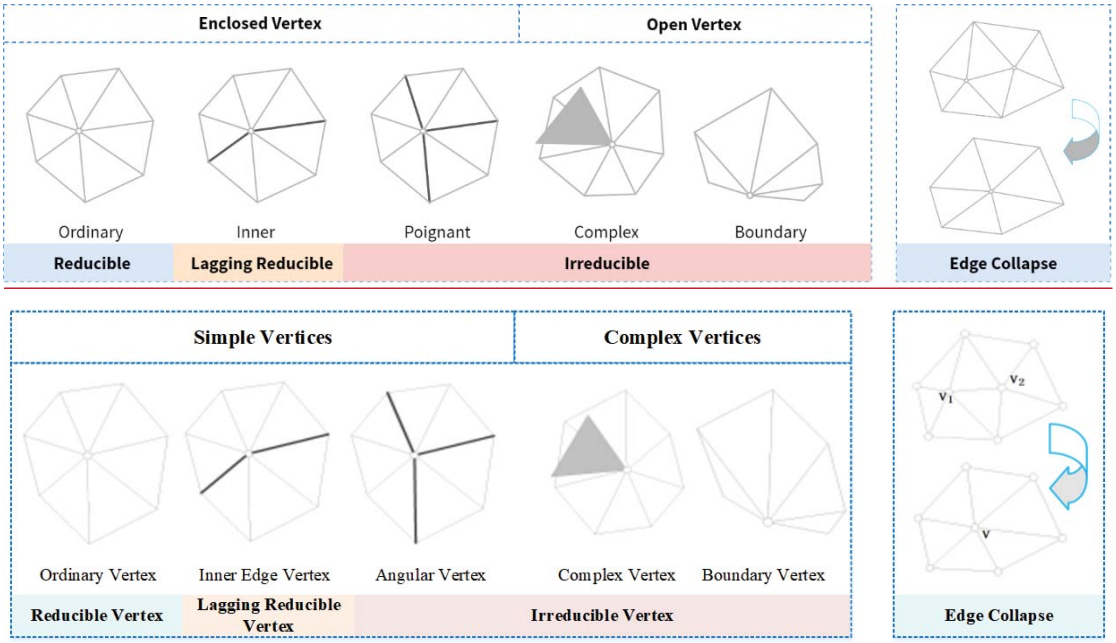

Figure 4. Sketch Map of Vertex Classification

4.2.3 Accumulated Error Measurement

It is generally sufficient to perform mesh simplification with a combination of the original QEM-based algorithm and the two optimization methods explained earlier. However, to better control the level of simplification and avoid severe distortion of the original model, this research proposes to consider the overall accumulated error.

Edge Collapsing Error Measurement

The original QEM-based algorithm measures the error as the sum of squared distances between the vertices and planes, which is a heuristic that characterized the cost of collapsing a selected edge. While geometrically intuitive, this measurement does not reflect the loss of local detail properly, as the distances between a certain vertex and all planes in the mesh are accounted. This research, contrarily, measures the collapsing error using the sum of squared volume error, which is more representative of local geometry changes, to better reflect the topology change of the simplified mesh. An edge collapse operation is shown in the right of Figure 4, as edge

v_1v_2 collapses to vertex v. The collapsing error is defined as the sum of squared volumes of all triangular pyramids with v as apex and a triangle face in $T(v_1)$ or $T(v_2)$ as base, where $T(v)$ stands for a set of triangles in the mesh that contains vertex v.

If a triangle belongs to the plane $ax + by + cz + d = 0$ and the normalized coefficients satisfy $a^2+b^2+c^2=1$, then the triangle is represented with $\mathbf{p} = (a, b, c, d)^T$. The collapsing error, which is the sum of squared volumes of triangular pyramids with v as the apex and a triangle in $T(v_1)$, can be calculated using Equation 4.

$$\Delta(\boldsymbol{v},\boldsymbol{v}_1) = \Sigma_{\boldsymbol{p} \in T(v_1)} \boldsymbol{v}^T \left(\frac{S_{\boldsymbol{p}}^2}{9} \boldsymbol{p}\boldsymbol{p}^T \right) \boldsymbol{v} = \boldsymbol{v}^T \left(\Sigma_{\boldsymbol{p} \in T(v_1)} \frac{S_{\boldsymbol{p}}^2}{9} \boldsymbol{p}\boldsymbol{p}^T \right) \boldsymbol{v} = \boldsymbol{v}^T \boldsymbol{Q}_1 \boldsymbol{v} \tag{4}$$

Where $S_{\mathbf{p}}$ is the area of the triangle represented by $\boldsymbol{p}$, and $\boldsymbol{v}$ is a 4-dimensional vector composed by appending a '1' to the geometric coordinates of the vertex v, so that $\boldsymbol{p}^T\boldsymbol{v}$ is the height of the triangular pyramid. $\mathbf{Q}_1$ is a symmetrical matrix.

Similarly, the sum of squared volumes of triangular pyramids with v as the apex and a triangle in $T(v_2)$ could be calculated as $\Delta(\boldsymbol{v},\boldsymbol{v}_2) = \boldsymbol{v}^T \boldsymbol{Q}_2 \boldsymbol{v}$. At this stage, the collapsing process works according to the following conditions:

a) If both vertices of an edge are *irreducible*, then the edge should not be collapsed;
b) If one of the vertices of the edge to be collapsed is *irreducible*, then collapse the edge to this vertex. For example, if vertex v_1 of edge v_1v_2 is irreducible, then collapse the edge v_1v_2 to vertex v_1, and the collapsing error will be shown in Equation 5.

$$\Delta(v) = \Delta(v_1,v_2) = \boldsymbol{v}_1^T \boldsymbol{Q}_2 \boldsymbol{v}_1 \tag{5}$$

c) If none of the vertices of the edge to be collapsed is *irreducible*, then the error of collapsing the edge v_1v_2 to vertex v is calculated using Equation 6.

$$\Delta(v) = k_1\Delta(v,v_1) + k_2\Delta(v,v_2) = \boldsymbol{v}^T(k_1\boldsymbol{Q}_1 + k_2\boldsymbol{Q}_2)\boldsymbol{v} = \boldsymbol{v}^T\boldsymbol{Q}\boldsymbol{v} \tag{6}$$

Where, k is '1' if its corresponding vertex is *ordinary*, and 2 if its corresponding vertex is *lagging reducible*. The position of the vertex v in this case should be carefully selected to ensure the topology remains as closed to the original mesh as possible. The position of v that optimizes the simplification is identified using Equation 7, when the third-order leading principal submatrix of the symmetric matrix $\mathbf{Q}$ is *invertible*.

$$\boldsymbol{v} = -\begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix}^{-1} \begin{bmatrix} q_{14} \\ q_{24} \\ q_{34} \end{bmatrix} \tag{7}$$

If the third-order leading principal submatrix of $\mathbf{Q}$ is not *invertible*, then v could be searched along edge v_1v_2, or $(v_1+ v_2)/2$ is selected as the collapsed vertex.

Whenever an edge is collapsed, the mesh stored is updated, and the geometric parameters including normal vectors of the newly formed triangles are calculated.

Edge Collapsing Cost Considering Historical Errors and Overall Accumulated Error

The edge collapsing error described above, $\Delta(v)$, represents changes in the local mesh neighborhood after one collapsing operation only and does not consider the simplification history. Therefore, the mesh is vulnerable to successive collapses in certain areas where it could be subjected to rapid loss of details. To address this challenge, historical errors are considered by defining the collapsing cost of an edge as in Equation 8.

$$C(v_1,v_2) = \Delta(v) + h(v_1) + h(v_2) \tag{8}$$

Where 7 and 7 stand for the historical collapsing errors at vertices v_1 and v_2. The historical collapsing error at a vertex v formed as a result of the collapse of edge v_iv_j is defined as in Equation 9.

7 6 (9)

In the process of simplification, the edge with the least collapsing cost is collapsed first. The initial historical collapsing error at every vertex is 0. As the simplification process advances, the historical collapsing error accumulates at the collapsed vertex, and the collapsing costs of edges containing collapsed vertices rise. Consequently, their positions in the collapsing priority list drop back, ensuring local details do not vanish too quickly. Before each collapsing operation, the similarity between the simplified model and the original model is calculated, and if it is below the threshold, the simplification process is triggered to terminate ensuring the accuracy of the simplified outcome.

4.2.4 The Mesh Simplification Algorithm Procedure

The full procedure of the mesh simplification algorithm is illustrated in Figure 5.

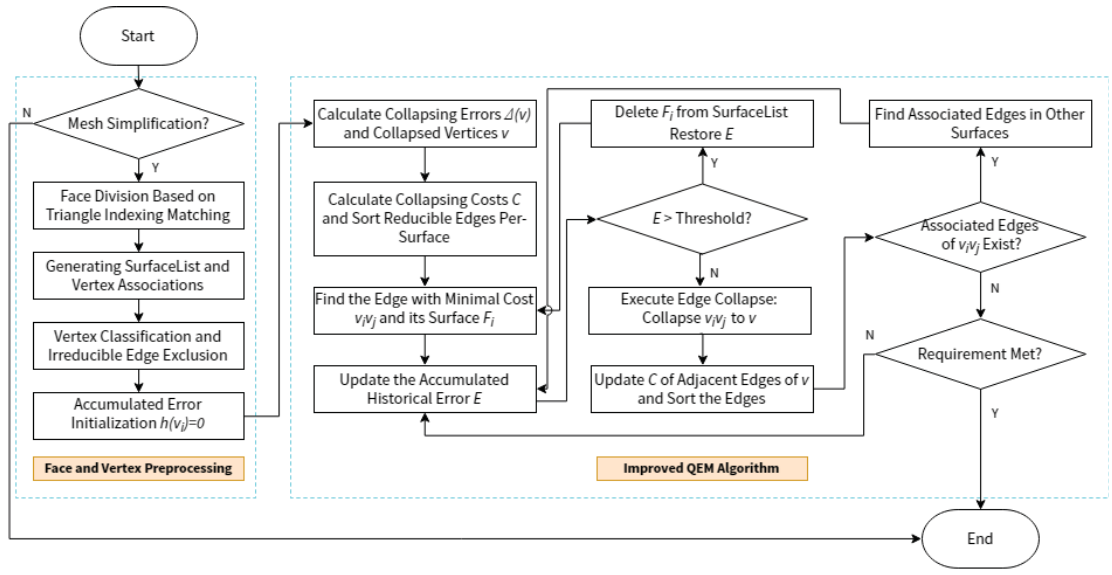

Figure 5. Workflow of Mesh Simplification

The algorithm unfolds according to these steps:

a) Perform surface division using triangle indexing matching, and generate the list of surfaces to be simplified (i.e. SurfaceList);

b) Establish the association between vertices in different surfaces;
c) Classify vertices in the surfaces according to geometric and topologic properties, and exclude irreducible edges;
d) Initialize the historical simplification error $h(v_i) = 0$ at all vertices;
e) Calculate the collapsing errors $\Delta(v)$ of all reducible edges in each surface, positions of the collapsed vertices v, and the collapsing costs C. Sort all the reducible edges by surface using C as the key in ascending order;
f) Find the edge with the minimal collapsing cost (i.e. $v_i v_j$) from the SurfaceList and the surface it belongs to (i.e. F_i);
g) Update the accumulated historical error of the surface, $E \mathrel{+}= \Delta(v)$. If E exceeds the threshold of surface F_i, then delete it from SurfaceList and return to step f), otherwise proceed to step h);
h) Collapse edge $v_i v_j$, and update the historical error at the collapsed vertex v, $h(v) = C(v_i, v_j)$, and delete the vanishing edge;
i) Calculate the collapsing errors, positions of collapsed vertices, and collapsing costs of adjacent edges of vertex v, and organize them within the list in an ascending order according to their collapsing costs;
j) Search for associated edges of $v_i v_j$ in other surfaces, and repeat steps g) to i) for each associated edge found. Then, proceed to step k), and
k) If the model has met the simplification requirements, then terminate; otherwise jump to step f) to keep simplifying.

This set of simplification algorithms and rules, combined with the mapping-based optimization scheme explained earlier, geometric data storage can be significantly reduced. However, there remains the challenge of optimizing model transmission and display which is a necessity in practical projects. An approach to address this challenge is proposed in the next section.

4.3 TRANSMISSION AND DISPLAY OPTIMIZATION

To complement the storage optimization capabilities and address the complete process of MEP model geometric optimization and display, this section presents algorithms and workflows for both model compression and model display optimization.

4.3.1 Compression Algorithms of MEP Models

The design, construction and operation of MEP systems require collaboration between multiple project's actors, and results in large-sized models that involve interchanges across different platforms. The efficient storage and transmission of these large models is a key element to enable this process. Reducing the size of the data to decrease the storage space and transmission time is essential in information and communication systems. The process of storage optimization explained in the previous section partly contributes towards this objective by addressing the storage aspect. To facilitate the efficient transmission, compression algorithms applicable to triangulation data and model properties, which are the most important data in the transmission process, are proposed in this section.

JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data interchange format [46]. It is suitable for the front-end development of the proposed system and is used as the format of data interface to transform all necessary information from server to client side. Nowadays, there are many data compression algorithms including human coding, arithmetic coding, LZ series algorithms and so on [47]. GZIP (GNU Zip) is a compression utility designed to be a replacement for *compress*, which is a primitive Unix utility. GZIP was selected as the compression utility due to its advantages over *compress* such as improved compression and freedom from patented algorithms. To further improve the efficiency of the transmission process, two algorithms are proposed to be executed prior to the GZIP compression: a normal vector compression algorithm for triangulation data, and a fixed-dictionary compression algorithm for model properties.

First, a data storage format (i.e. Mesh) has been defined for triangulation data (see left side of Figure 6). It defines the storage format for data of relevant vertices, faces, triangles, and normal vectors, and employs a normal vector compression algorithm to optimize the transmission. A normal vector would usually be stored as three floating-point numbers. However, transmitting normal vectors in the raw floating-precision format is often unnecessary and can adversely affect the transmission speed.

Numerous algorithms have been proposed for the compression of vectors. Among these, spherical compression was identified as providing optimal balance between compression rate and error [48] and therefore, was employed in the transmission optimization architecture.

This algorithm is capable of compressing the storage of normal vectors into two bytes (see right side Figure 6). The three coordinates of a unit normal vector are first transformed into two spherical angles (i.e., $lon = \arctan z/x$, and $lat = \cos y$) which are then normalized. Then, PackSize is set to 255, and the normalized angles are mapped to integers u and v in the range 0-255. In doing so, normal vectors data are compressed and the model will later be compressed by GZIP. When the data is decompressed at the client side, it is first decompressed using GZIP, then u and v are reversely mapped into angles, then into coordinates, and finally normalized. This normal vector algorithm is not lossless, but the loss is limited to minor angle errors. However, consecutive model optimization algorithms will ensure that the overall geometric expression of models is not affected as evidenced later in the paper.

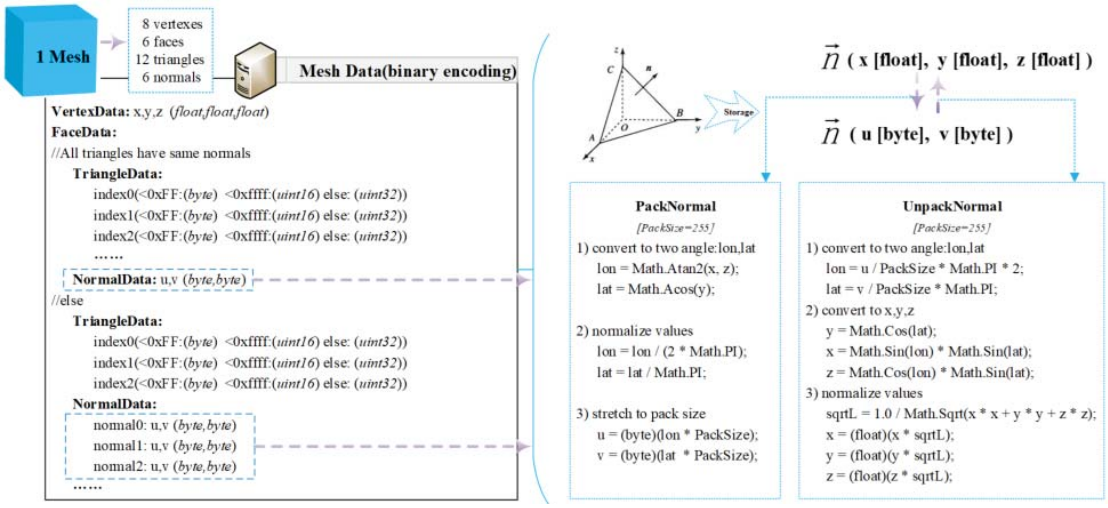

Figure 6. Definition of Mesh Data and Optimized Storage of Normal Vector Data

Large quantities of model properties data are also transmitted during project development and asset handover. To achieve efficient data transmission, a JSON data transmission interface [49] and a dictionary versioning metadata are proposed. As all of the texts to be compressed are self-defined, the adoption of a fixed-dictionary compression is possible. A fixed-dictionary can be established at the early design stages in a way that both the system server and the client side could load it in advance. It is possible to establish the MEP property names as the dictionary contents and address the issue of recurrence and duplication that would consume storage space. This challenge can be addressed by retrieving the list of all property names in the interface file and removing duplicate items and relative short names to form a fixed-dictionary that can be embedded into front-end script files. Consequently, the content of the dictionary would be transmitted only once instead of multiple times, and together with the suppression of redundant data, will result in an efficient transmission. The fixed-dictionary compression is the step before GZIP compression and aim to improve the level of relevance of the compression. In the process of transmission, the server first encodes the texts using the fixed-dictionary. This is followed by GZIP compression, transmission and GZIP decompression at the client side, which are configured and controlled by the server. Finally, the front-end script files use the preloaded dictionary - according to the dictionary versioning metadata - to decode the data into original texts, concluding the compression process.

The data transmission process is illustrated in Figure 7. MEP data (i.e. mesh data and property data) for information transmission is first retrieved from the database and stored in their respective self-defined data formats. Mesh data are then preprocessed using the normal vector compression algorithm, while the property data are preprocessed with the fixed-dictionary compression algorithm. Then, they are both compressed with GZIP in preparation for the subsequent transmission. The client side, upon receiving the transmitted data, would first unzip the data packets with GZIP to retrieve the compressed geometric and semantics data, which are then decompressed by reversely mapping spherical angles to coordinates, and by using a preloaded fixed dictionary, respectively. The decompression algorithms require minimal calculation, and does not burden the performance to the extent as to be worthy consideration.

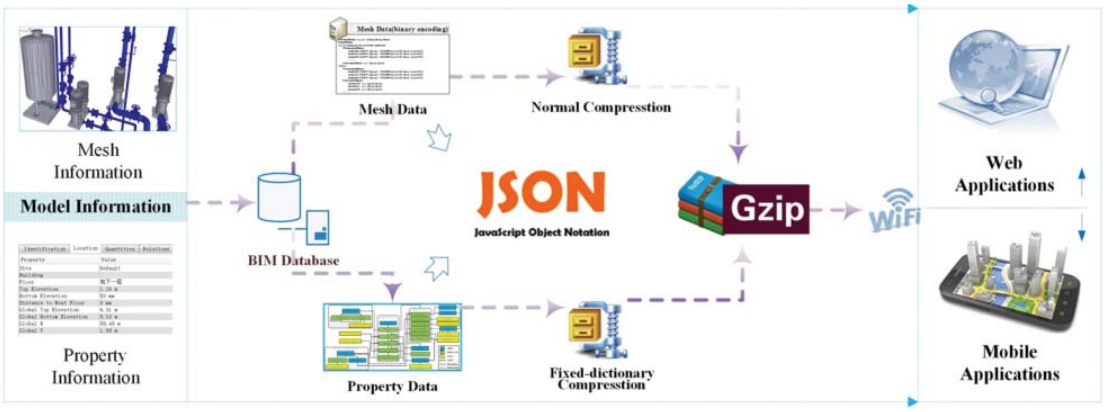

Figure 7. Workflow of Data Compression for Information Transmission

4.3.2 Display Optimization: Normal Vector Regeneration

To provide a holistic solution for the exchange of large MEP models, the optimization of model storage and transmission - through the optimization algorithms explained in the previous subsection - needs to be accompanied with improvement to the frontend display. To this end, this research presents a clustering-based *normal vector regeneration algorithm* for triangle meshes.

Vertices in a 3D mesh are categorized according to the directions of the normal vectors of their adjacent triangle faces. Relatively concentrated directions suggest a vertex in a smooth curved surface whose normal vector can be obtained as the weighted average of those of neighboring triangles. Otherwise, the vertex can be considered as the intersection of a few surfaces. In this case, neighboring triangles are divided into groups with concentrated normal vector directions, and the normal vector of each group is separately estimated. In the process of generating normal vectors for vertices, *k-means* clustering algorithm is adopted to divide the neighboring triangles into clusters, so that a normal vector for the vertex can be calculated from each cluster [50]. The algorithm divides n data items into k clusters in such a way the sum of squared distances between clusters is minimized. The accuracy of normal vector generation using different indicators as weights were compared for various types of curved surfaces, and angular weights were identified as having the highest accuracy in most cases [51]. Hence, this research adopts angular weights in the normal vector regeneration algorithms for triangle meshes.

Unlike ordinary k-means algorithms, the number of clusters, k, could not be predetermined. The optimal value for k depends on whether the vertex is inside one smooth surface or is on the intersection of multiple planes of curved surfaces. If the vertex is inside a smooth surface, then its normal vector can be decided by all its neighboring triangles, and clustering in this case is not necessary. A threshold, β, was inserted into the algorithm. When clustering, a value for k within the interval $[1, n]$ is selected starting from 1, the maximum angle, α_{max}, between any two normal vectors in one cluster is identified. If $\alpha_{max} < \beta$, then the clustering results are satisfactory, and the algorithm terminates; otherwise, k is incremented and a new round of clustering calculation is performed. The complete process is shown in Figure 8.

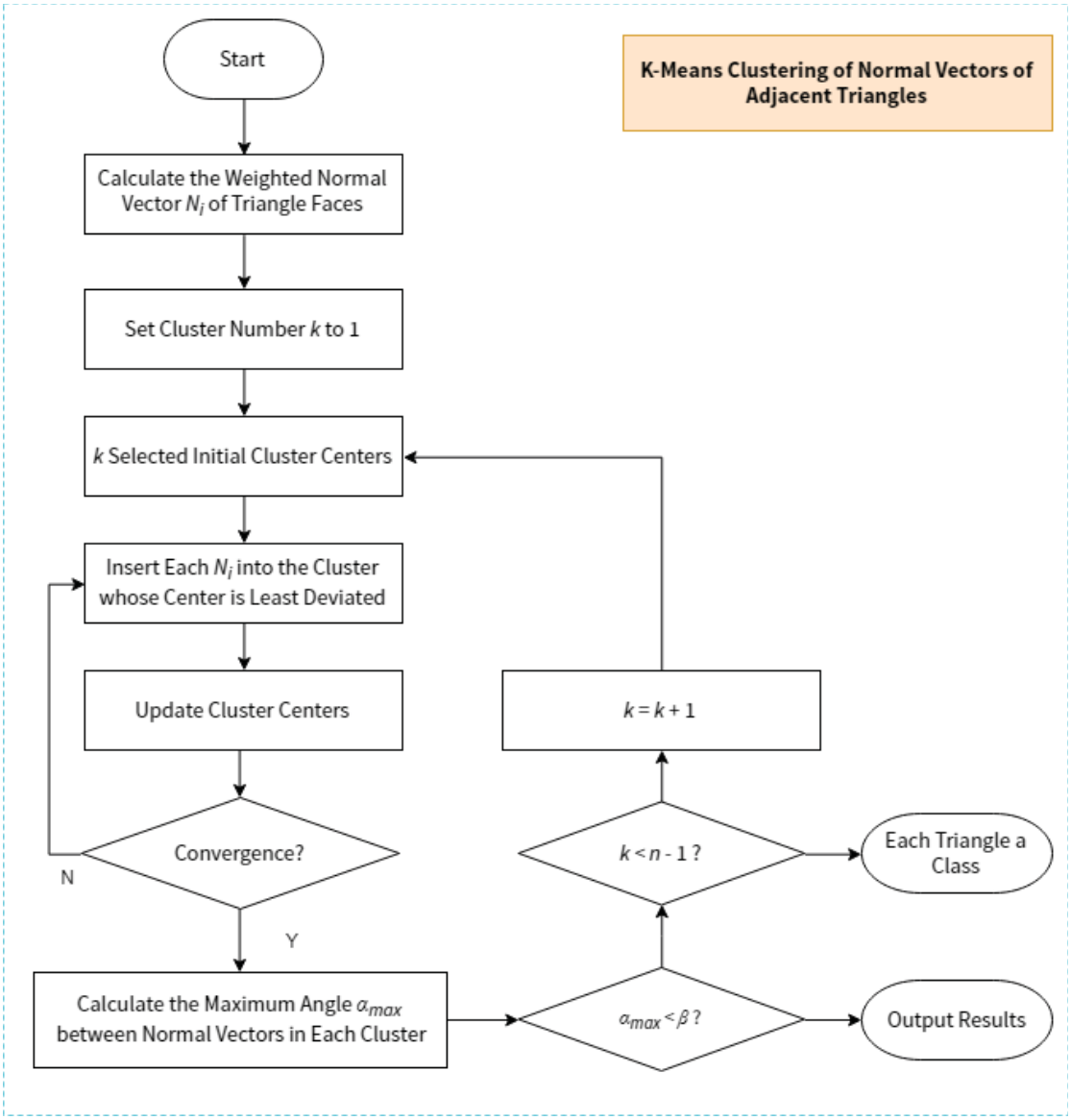

Figure 8. Workflow of K-Means Clustering of Adjacent Triangles with Common Vertex

The complete process of normal vector regeneration is as follows:

a) Checks whether the mesh description of the component is already accurate enough according to the component type and number of triangles (i.e. if the component comprises of predominantly plain surfaces, or has enough triangles to represent curved surfaces, the algorithms adopts face normal vectors and terminates);
b) Computes the normal vectors of triangle faces, and generates the list of vertices of the mesh;
c) For each vertex in the list of vertices, the algorithm identifies all triangles in the mesh that contain it;
d) Performs k-means clustering on the triangles as shown in Figure 9 to yield k groups of triangles, and
e) For each triangle mesh, C_j, the algorithm calculates the weighted average normal vector, N_j, as the eventual normal vector, and set it as the normal vector of the vertex corresponding to the cluster.

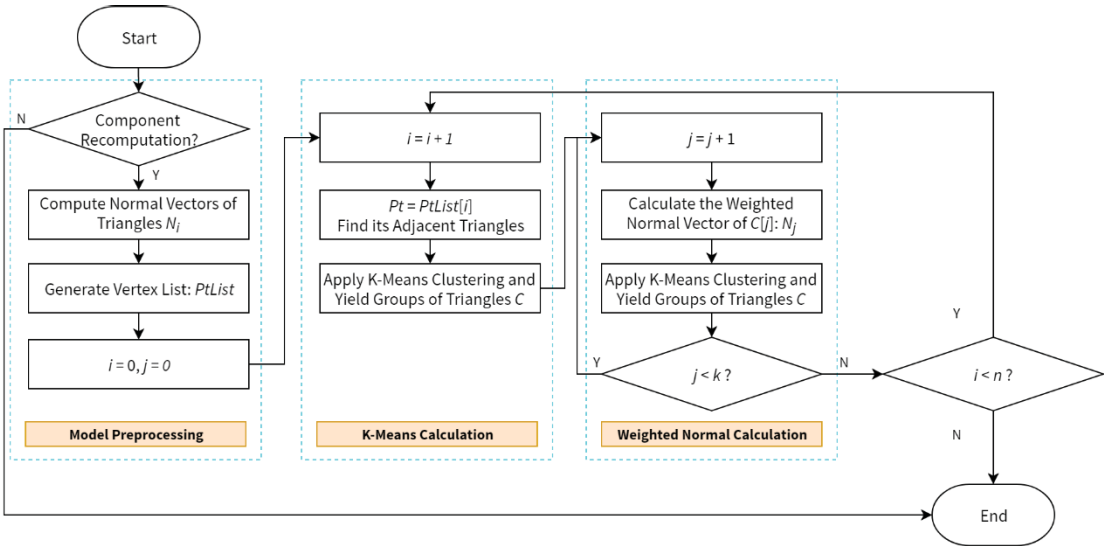

Figure 9. Workflow of K-Means-Based Normal Vector Calculation

5. CASE STUDIES AND DISCUSSIONS

This study proposed algorithms and techniques for the optimization of storage, transmission and display of MEP models. This section implements and tests such algorithms and techniques using real-world case studies. The demonstration is first performed for individual components separately using relatively small-scale models, and then holistically with the application the proposed solution to a large-scale MEP project, the Hedong Subway Station project in Guangzhou Metro. In such a way, the demonstration methodology merges together unit evaluation (i.e. testing individual methods and functions of the solution's modules implemented as part of the solution) with summative or end-to-end evaluation (i.e. evaluating the utility of the software for its intended purpose).

To reduce the redundant storage of geometric information, a similarity-matching algorithm was devised and describe in Section 3.1. The algorithm considers coincidences and similarities between meshes after translation, but not after any other geometric transformations such as rotation or scaling. Similarity between two meshes is measured based on the weighted mean squared distance between vertices of the two meshes. Meshes whose similarity exceeds a preset threshold will be identified as similar and their geometric data will be mapped to reduce redundant storage.

In Figure 10(a), a similarity matching was performed between models of the same elbow component with different numbers of vertices resulting from different levels of geometric simplification. The original model has 456 vertices and its similarity with itself is obviously 1.0. The moderately simplified mesh in the middle has 142 vertices (reduction by ~~31~~69%) with a calculated similarity of 0.984. The substantially simplified mesh on the right remained with 47 vertices only, and preserved a similarity of 0.905 compared with the original model. In Figure 10(b), similar-sized fine models of an equal tee, a concentric reducer, and a tube segment are matched with the model of the elbow, and the highest similarity was 0.833 and was obtained

with the equal tee. The successful identification of higher similarity between meshes of the same component even after significant simplification, than between meshes of different types of components, is a successful demonstration of similarity--matching capabilities of the proposed algorithms.

In Figure 10(c), the proposed similarity-matching algorithm was integrated into the mapping-based optimization schemes, and was tested with a model of an airport check-in island. The original model consists of 1,367 components and 211,000 triangles, while the optimized model, using a similarity threshold of 0.99, contains only 374 components and 50,600 triangles. The data volume has been reduced by 76% the original size, saving storage and graphics memory consumption, and enhancing rendering efficiency.

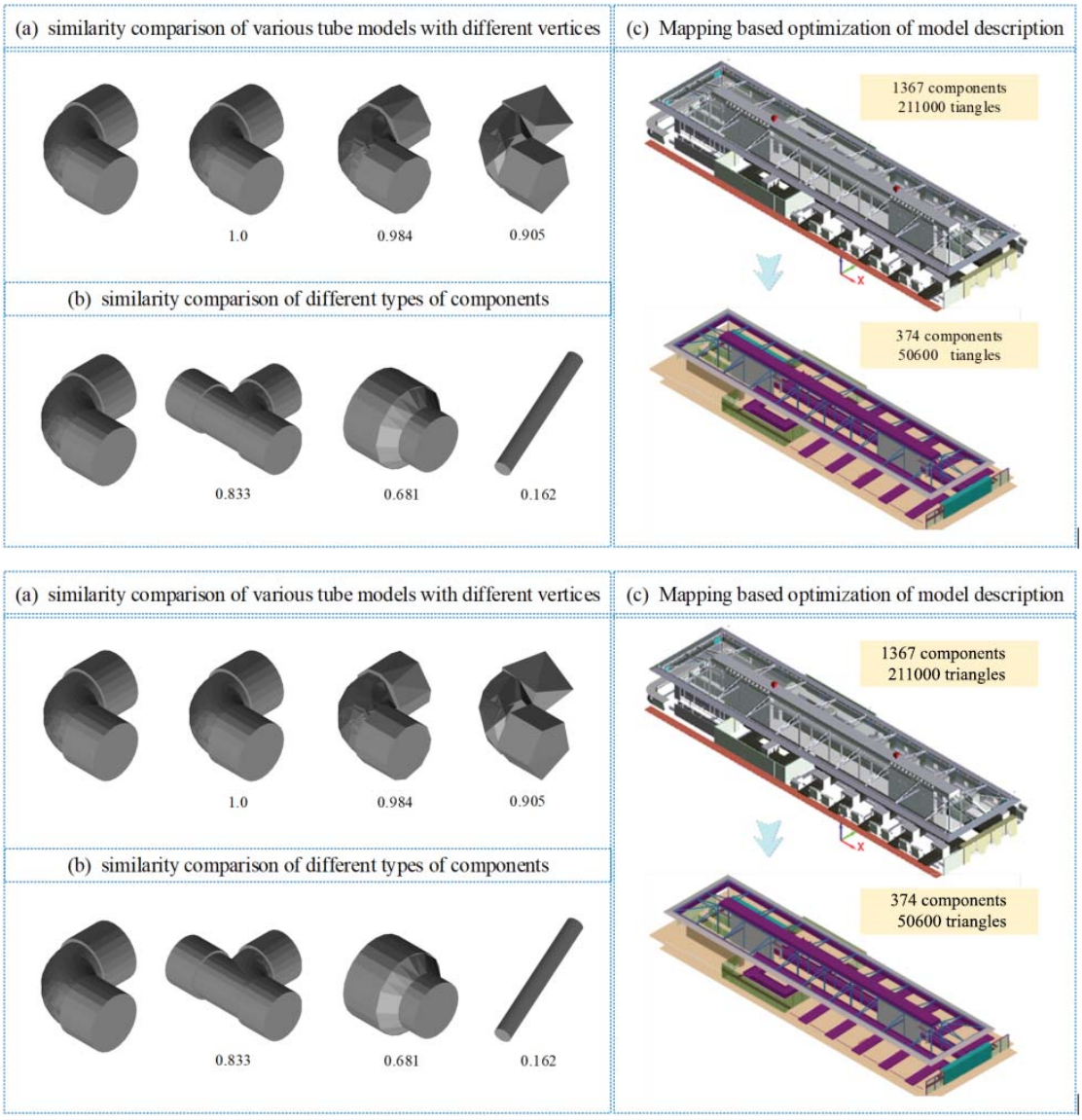

Figure 10. Similarity Calculation and Mapping-Based Storage Optimization

Despite the successful testing of the algorithm, there are still opportunities for further improvement. For example, in the current algorithm, meshes are aligned only for translation before matching which excludes coincidences after any other geometric transformations

including rotation. This approach assumed that MEP systems are mostly designed based on orthogonal coordinate systems, making coincidences after random rotations very rare. However, this assumption is not valid at all times as the case with the airport check-in island model. The level of data volume reduction would be negatively influenced in such cases, since components that would have coincided after rotation are considered as dissimilar by the algorithm. Moreover, the algorithm depends on properties information for branch-pruning in order to compare the type of meshes before similarity calculation. In the absence of adequate properties data, the efficiency of the algorithm would decrease and concurrently the result would be more susceptible to mismatching of different types of components.

In Section 3.2, a set of approaches and algorithms for simplifying geometrically complicated components were presented. First, a surface division and associated vertex recognition approach was proposed as a preprocessing procedure to prevent holes and overhangs during simplification. Second, a classification scheme for model vertices according to their topological context was devised to determine the collapsing priorities and guide the collapsing process. Finally, a mesh simplification algorithm employing an improved QEM metric, supplemented with an accumulated historical error metric, was developed to perform mesh simplification.

Figure 11 shows the result of mesh simplification for four different MEP components using the proposed algorithm. The simplification rate for relatively complicated components, such as the camera, the elbow tube fitting, and the tuyère, have reached as high as 70% - 80%. For geometrically simpler components, such as the pipeline segment, the simplification rate is around 40% - 50%. The results suggest the proposed algorithm could effectively reduce the number of triangles in complicated meshes, while preserving their topologies. The automatic termination of the algorithm according to accumulated historical errors, combined with the surface-wise simplification strategy, prevented local oversimplification in the result outputs. When combined with the mapping-based optimization technique, the overall rate of reduction in size for complicated components can exceed the achieved 80% and can possibly reach as high as 90%. The proposed algorithm is most suitable for simplifying complicated components as its performance generally improve with mesh complexity. Hence, a tradeoff problem between the storage saving targets and the extra processing power required should be considered when selecting meshes for simplification.

To enhance the display effect, a normal vector regeneration algorithm. The core of the algorithm is the clustering of the normal vectors at a vertex into suitable groups. Since the number of surfaces the vertex borders is not known in advance, the algorithm clusters normal vectors using *k-means* algorithm with an iteratively increasing value of k, and terminates once an acceptable result is obtained within a preset threshold.

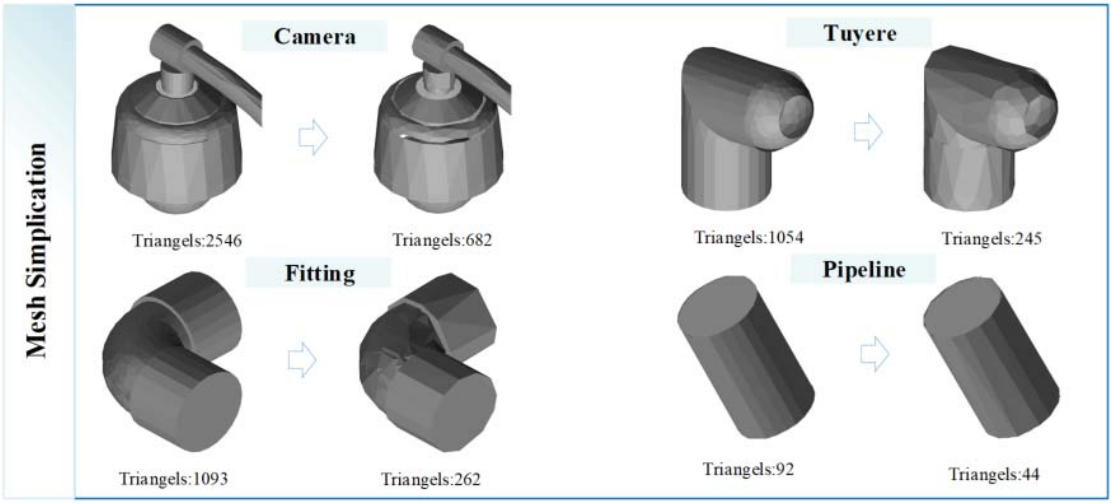

Figure 11. Demonstration of Mesh Simplification Results

Figure 12 shows the result of applying the normal vector regeneration algorithm to the simplified model of a camera component. The original model is shown on the left, and the simplified model in the middle. It can be observed that before applying the normal vector regeneration, the edges of the triangle faces are visible, making each of them appear as a separate surface. The model after normal vector regeneration is shown on the right of Figure 12. The output is satisfactory, as the curved exterior surfaces appear smooth without edges, and with visually realistic specular highlights.

Figure 12. Model Display Optimization

However, this algorithm has some issues. Firstly, the selection of clustering algorithms could be disputed. The *K-means* clustering algorithm requires as input the number of clusters to produce which cannot be known in advance. For the proposed algorithm, an iteration of possible values of k has to be made which may decrease the efficiency. Secondly, closely aligned normal vectors, especially those with an angle that is smaller than the threshold, are very likely to be assigned to the same cluster. This produces desirable results in most cases but there are some exceptions. Finally, distinguishing closely aligned planes from smooth curved surfaces from pure geometric information is very challenging, and support of additional information would be required.

The solution proposed in this study was also applied to a large-scale MEP project, the Hedong

Station of Guangzhou Metro in Liwan District, Guangzhou, China, as shown in Figure 13. The station is located at a saddle-shaped terrain, and is a two-story underground island-platform station. The flat section measures around 300 meters in length, and the total area is 9,034.45 m^2, which includes 7,384.00 m^2 for the main body, and 1,650.45 m^2 for subsidiary bodies. The components in the MEP systems are in large numbers and complex shapes, with the number of triangles reaching up to about 10 million and occupying a storage space of around 2 GB of space. As a result, this model poses significant demand on the storage, cross-platform transmission and efficient display.

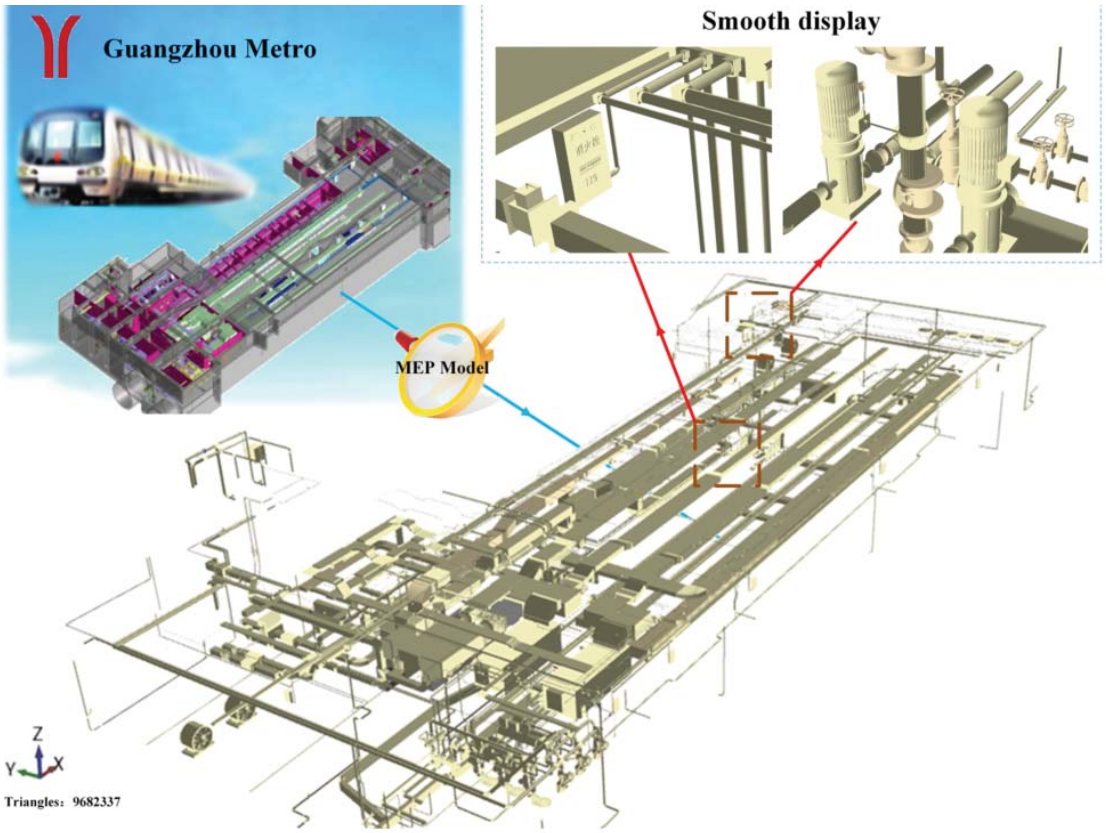

Figure 13. Application to Hedong Station of Guangzhou Metro

Figure 14 shows that with the proposed simplification algorithm, the topology of a complicated fire pump remained largely intact visually, even with a simplification rate of 80%. This demonstrates the simplification quality attainable with the proposed algorithm.

Cross-platform data transmission and model displaying are also essential in large construction and engineering with major MEP systems. The transmission of large amounts of data can create capacity issues as it is resource demanding and time consuming. The proposed approach enables significant reduction in the volume of the data to be transmitted. Figure 15 shows that the proposed techniques reduced the transmission volume of a subsystem by more than 80% which is considered a significant contribution in terms of time and costs associated with

transmission and display of models.

To enhance the 3D display effect and compensate for the side effect caused by the storage and transmission optimization, this solution proposes a clustering-based normal vector generation algorithm for triangle meshes. The results from testing these components of the solution are shown in Figure 14 (refer to upper right part). The displayed 3D model exhibits smoothness and clarity hence, it can be concluded that quality and efficient display can be achieved in addition to optimized and efficient storage.

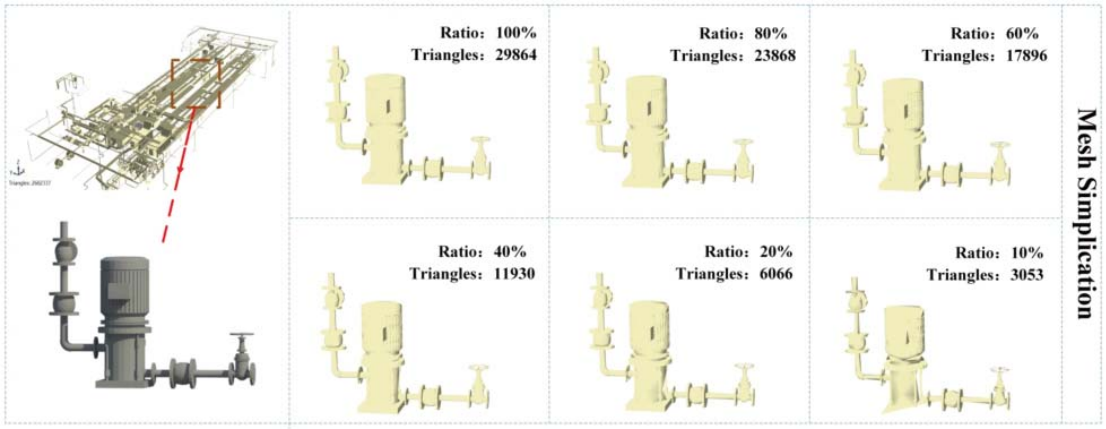

Figure 14. Different Levels of Mesh Simplification for a Fire Pump

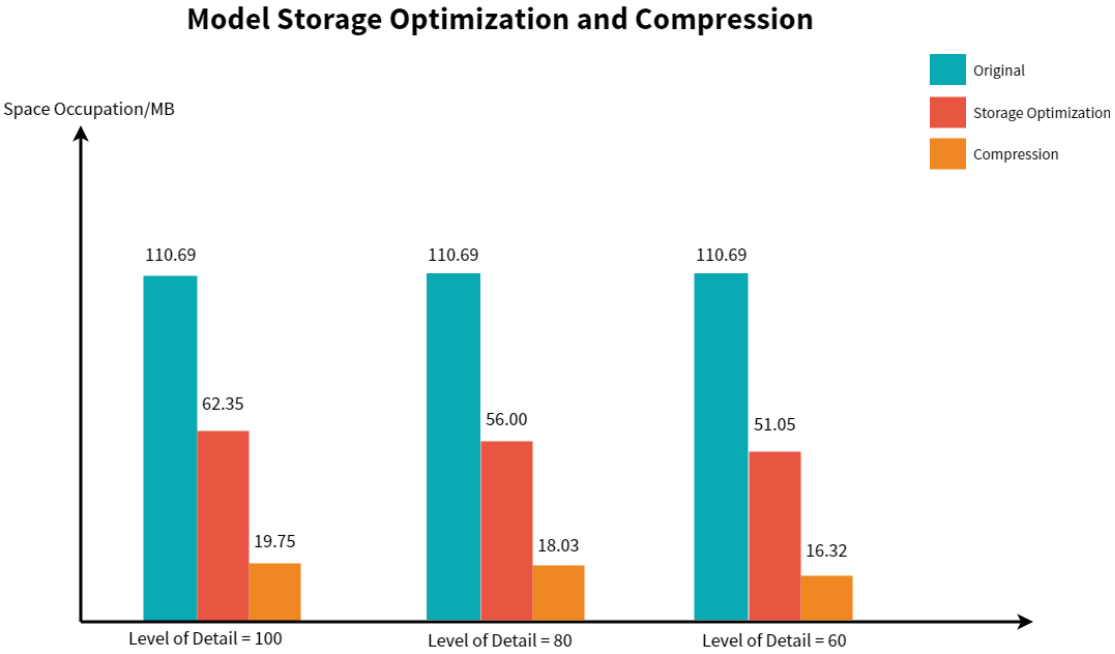

Figure 15. Results of Storage Optimization and Compression of MEP Model

Overall, this research proposed a solution consisting of set of algorithms and technique for the geometric optimization of MEP models. The solution addresses all phases involved in this process from compression, through storage, to transmission and display. The testing results were generally found to be positive and proved that the proposed solution has capabilities that are commensurate with the complexity and difficulty for management of MEP projects.

Some limitations to specific algorithms (i.e. the similarity-matching algorithm considers coincidences and similarities between meshes after translation, but not after any other geometric transformations such as rotation or scaling) were identified and some issues regarding the selection of certain algorithms (i.e. *K-means* clustering algorithm requiring as input the number of clusters which is not known in advance) are still open for debate. Moreover, the algorithms proposed are also not easily parallelized. The similarity-matching algorithm involves iterating through all the meshes in the model and test the similarity between each pair, and the mesh simplification algorithm needs to find the vertex with the least collapsing error globally, thus both are denied the potential of being greatly boosted by parallelization. However, such limitations are inherent to the problems themselves, instead of the proposed implementations. The rendering of the models, obviously, are parallelized and calculated on the GPU, which is implicitly implemented by the graphics facility.

On the one hand, the storage optimization algorithms assume memory to be one of the main limiting factors of MEP applications due to the large size of MEP models [49], and the selected algorithms require more processing power to produce lightweight models for enhanced transmission and rendering speed. Although this assumption is generally valid, there exist exceptions where some graphical workstations are specially configured to be equipped with extraordinarily large RAM sizes. These workstations can manipulate even entire MEP models of large construction and engineering projects. In such circumstances, the additional processing time required by the proposed algorithms to reduce memory usage may become counterproductive. However, with the exemption of large and ~~capital intensive~~capital-intensive projects, most engineering consultants responsible for MEP systems are either small or micro enterprises and cannot generally afford expensive workstations. The proposed solution can be very beneficial with this cluster.

Moreover, geometric optimization may be consisted only a small and specialized contribution within the holistic challenge of improving productivity performance of MEP projects. However, they are other important process management and technology challenges affecting MEP systems. Key challenges include: the limited cooperation between actors and the high number of collisions affecting MEP systems; and the challenges of handing over MEP data to facilities managers [52]. The proposed solutions, by adopting open standards (i.e. IFC) and lightening heaving MEP models, contribute to the two challenges and democratize the use of BIM models across both the supply chain and the project phases including the facilities management phase.

6. LIMITATION AND FUTURE WORKS

Certainly, this work is not without limitations, and there exists room for further improvements. We've identified some major limitations of this research work and discussed the possibility of future works for further improvements.

Firstly, as has previously been discussed, the proposed algorithms had limitations which can be resolved in the future. The similarity-matching algorithm considers coincidences and similarities between meshes after translation, but not after any other geometric transformations based on the postulation that MEP systems are mostly designed in orthogonal coordinate systems. However, coincidences after a combination of translation and other geometric

transformations also occasionally occur, and support for such similarities should be implemented in the future. The k-means clustering algorithm used to cluster adjacent triangles into surfaces requires as input the number of surfaces to be clustered into, which cannot be determined in advance, forcing the algorithm to iterate over all possible values, resulting in a multiplication of time consumption. In the future, hierarchical or density-based clustering algorithms could be used and tested to avoid the aforementioned overhead.

Secondly, there lacks means to fully quantitatively evaluate and benchmark the proposed algorithms. Although some meshes are conventionally used by algorithm developers to test and benchmark the results, e.g. the Stanford Bunny[53], the applicability to the results of this research is poor, as the proposed algorithms seek to exploit characteristics specific to MEP systems. Currently, an open access, publicly accepted, and conventionally used benchmarking model is yet to be established, which is an agenda requiring contributions and efforts from the entire community. Consequently, the test-case used in this research is the MEP system from Hedong Station. There stands a chance to reassess and benchmark the proposed algorithms against the mentioned standard model in the future.

Finally, the value and necessity of geometric optimization techniques in the industry is also open to debate. The research assumed memory-saving as the major objective, which in real-life projects are not short of exceptions. Specialized graphical workstations are able to consume and manipulate entire MEP projects of even large-scale projects, in which case the proposed methods are futile. Geometric optimizations are also but a specialized aspect in the performance of MEP projects, the value of the research is limited within the extent to which geometric models could benefit MEP projects. Many other key challenges await to be countered to revolutionarily change the MEP industry in the future.

7. CONCLUSION

This research addressed challenges affecting the storage, transmission, and display of geometric data in the application of BIM to large-scale MEP projects. The foundation of the solution for optimizing model storage and display consisted of a novel information storage and management architecture. This architecture included a mapping-based model description method for storage optimization based on similarity analysis; and a novel Quadric-Error-Metric (QEM) mesh simplification algorithm for complicated components. The testing of these methods with real industry data from major infrastructure projects in China were successful, and showed that the volume of storage data can be reduced by 40% to 50%, without affecting the display visual quality. The key contribution to achieve this level of performance was the improvement introduced to the original QEM-based mesh simplification algorithm with: the introduction of surface division, model feature point recognition, and accumulated error measurement. These improvements helped to substantially reduce the number of triangles in a mesh while maintaining topology.

Novel algorithms were proposed for the optimization of transmission and display of MEP models. These included: the self-defined Mesh data format using the JSON data exchange

interface; a normal vector compression algorithm, and a fixed-dictionary compression algorithm to compress key geometric and properties data. These algorithms improve the transmission efficiency by addressing the challenges in the coordination between the front-end and back-end, and in cross-platform interactions. The testing of these algorithms with real case studies showed that the volume of transmitted data can be reduced by over 80% without affecting the topology of the components.

Finally, to improve the efficiency of the display of MEP models without affecting its quality, this research proposed: a clustering-based normal vector generation algorithm for triangle meshes, and a dynamic lighting calculation technique to compensate for the potential side effects of storage and transmission optimization on the quality of displayed model. Both the algorithms and the techniques were successfully tested and respectively showed a significant reduction in the number of triangles, and a very good quality of displayed models.

An additional testing of the entire solution was performed within the Guangzhou Metro project and the positive results obtained from the individual testing of each algorithm and technique were confirmed. Assumptions and key limitations in the selection and mode of operation of certain algorithms were identified and discussed. In additional to the theoretical contribution identified at both the entire solution and its individual algorithms, this research provides practical implications. Its practical uses lie in all applications within the architectural, engineering and construction sector requiring the storage, transmission and display of large volume of model data, or in the operation stage where the adoption of 3D models is still embryonic.

APPENDIX

Flowcharts in this paper adhere to ISO-5807:1985 for standardized representation of symbols. The symbolisms of the shapes appeared in this paper are as suggested by Figure 16.

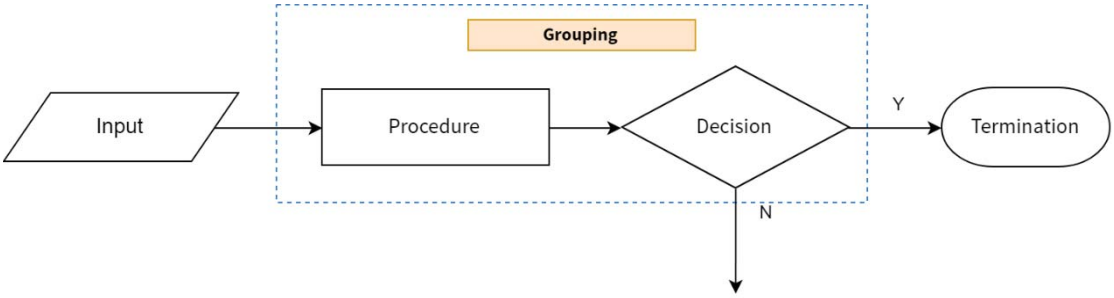

Figure 16. Representations of Shapes in Flowcharts

ACKNOWLEDGEMENTS

This research was supported by the National Key R&D Program of China (Grant No. 2017YFC0704200), the National Natural Science Foundation (No. 51478249), and the Tsinghua University – Glodon Joint Research Center for Building Information Modelling.

REFERENCES

[1] Korman T. M., Fischer M. A., Tatum C. B. (2003) Knowledge and Reasoning for MEP Coordination. *Journal of Construction Engineering and Management*, 129(6):627-634, doi:10.1061/(asce)0733-9364(2003)129:6(627)

[2] Khanzode A., Fischer M., Reed D. (2007) Benefits and Lessons Learned of Implementing Building Virtual Design and Construction (VDC) Technologies for Coordination of Mechanical, Electrical and Plumbing (MEP) Systems of a Large Healthcare Project, *Electronic Journal of Information Technology in Construction*, 13:324-342, https://www.itcon.org/papers/2008_22.content.04920.pdf, accessed July 8th, 2019

[3] Nicolle C., Cruz C. (2011) Semantic Building Information Model and Multimedia for Facility Management, *International Conference on Web Information Systems and Technologies*. vol 75. Springer, Berlin, Heidelberg, doi:10.1007/978-3-642-22810-0_2

[4] Vanlande R., Nicolle C., Cruz C. (2008) IFC and Building Lifecycle Management, *Automation in Construction*, 18(1):70-78, doi:10.1016/j.autcon.2008.05.001

[5] buildingSMART (2015) National BIM Standard-United States® (NBIMS-US™) Version 3, retrieved from https://www.nationalbimstandard.org/, accessed July 8th, 2019

[6] Kassem M., Succar B. (2017) Macro BIM Adoption: Comparative Market Analysis, *Automation in Construction*, 81:286-299, doi:10.1016/j.autcon.2017.04.005

[7] Yalcinkaya M., Singh V. (2014) Building Information Modeling (BIM) for Facilities Management – Literature Review and Future Needs, *International Federation for Information Processing International Conference on Product Lifecycle Management*, vol 442. Springer, Berlin, Heidelberg, doi:10.1007/978-3-662-45937-9_1

[8] Leite F., Akcamete A., Akinci B., et al. (2011) Analysis of Modeling Effort and Impact of Different Levels of Detail in Building Information Models, *Automation in Construction*, 20(5):601-609, doi:10.1007/978-3-662-45937-9_1

[9] Tabesh A. R., Staub-French S. (2006) Modeling and Coordinating Building Systems in Three Dimensions: A Case Study, *Canadian Journal of Civil Engineering*, 33(12):1490-1504, doi:10.1139/l06-124

[10] Riley D. R., Varadan P., James J. S., et al. (2005) Benefit-Cost Metrics for Design Coordination of Mechanical, Electrical, and Plumbing Systems in Multistory Buildings, *Journal of Construction Engineering and Management*, 131(8):877-889, doi:10.1061/(asce)0733-9364(2005)131:8(877)

[11] Yang K., Kang D., Xu P., Chen C. (2014) BIM-Based MEP Design Technology, *Construction Technology*, (03):88-90, doi:10.7672/sgjs2014030088

[12] Korman T. M., Tatum C. B. (2006) Prototype Tool for Mechanical, Electrical and Plumbing Coordination, *Journal of Computing in Civil Engineering*, 20(1):38-48, doi:10.1061/(asce)0887-3801(2006)20:1(38)

[13] Korman T. M., Lu N. (2011) Innovation and Improvements of Mechanical, Electrical, and Plumbing Systems for Modular Construction Using Building Information Modeling, *Architectural Engineering Conference 2011*, *Building Integration Solutions*, 448-455, doi:10.1061/41168(399)52

[14] Becerik-Gerber B., Jazizadeh F., Li N., et al. (2012) Application Areas and Data Requirements for BIM-Enabled Facilities Management, *Journal of Construction Engineering and Management*, 138(3):431-442, doi:10.1061/(asce)co.1943-7862.0000433

[15] Kassem M., Kelly G., Dawood N, et al. (2015) BIM in Facilities Management Applications: A Case Study of a Large University Complex, *Built Environment Project and Asset Management*, 5(3):261-277, doi:10.1108/bepam-02-2014-0011

[16] Schevers H., Mitchell J., Akhurst P., et al. (2007) Towards Digital Facility Modeling for Sydney Opera House Using IFC and Semantic Web Technology, *Electronic Journal of Information Technology in Construction*, 12:347-362, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.1510&rep=rep1&type=pdf, accessed on July 8th, 2019

[17] Motawa I., Almarshad A. (2013) A Knowledge-Based BIM System for Building Maintenance, *Automation in Construction*, 29:173-182, doi:10.1016/j.autcon.2012.09.008

[18] Xie H., Tramel J. M., Shi W. (2011) Building Information Modeling and Simulation for the Mechanical, Electrical and Plumbing Systems, *2011 Institute of Electrical and Electronical Engineers International Conference*, vol 3. IEEE， doi:10.1109/csae.2011.5952637

[19] Ko C. H. (2009) RFID-Based Building Maintenance System, *Automation in Construction*, 18(3):275-284, doi:10.1016/j.autcon.2008.09.001

[20] Motamedi A., Hammad A. (2009) Lifecycle Management of Facilities Components Using Radio Frequency Identification and Building Information Model, *Electronic Journal of Information Technology in Construction*, 14:238-262, doi:10.22260/isarc2011/0088

[21] Chen Q., Deng Z., He B., Zhang J. (2014) Key Technology and Software Implementation Method of BIM-Based MEP Emergency Management System, *Journal of Information Technology in Civil Engineering and Architecture*, (02):15-19, doi:10.3901/cjme.2010.02.217

[22] Chi H., Kang S., Wang X. (2013) Research Trends and Opportunities of Augmented Reality Applications in Architecture, Engineering, and Construction, *Automation in Construction*, 33:116-122, doi:10.1016/j.autcon.2012.12.017

[23] Kalasapudi V. S., Turkan Y., Tang P. (2014) Toward Automated Spatial Change Analysis of MEP Components Using 3D Point Clouds and As-Designed BIM Models, *2nd International Conference on 3D Vision*, 2:145-152, doi:10.1109/3dv.2014.105

[24] buildingSMART (2018) Home – Welcome to buildingSMART-Tech.org, retrieved from http://www.buildingsmart-tech.org/, accessed Aug. 13th, 2018

[25] Leon M., Laing R., Salman H., et al. (2014) Development and Testing of a Design Protocol for Computer Mediated Multidisciplinary Collaboration during the Concept Stages with Application to the Built Environment, Procedia Environmental Sciences, 22:108-119

[26] OCCS Development Committee Secretariat (2018) OmniClass, a Strategy for Classifying the Built Environment, retrieved from http://www.omniclass.org/, accessed July 8th, 2019

[27] NBS (2015) Introducing Uniclass 2015, retrieved from https://www.thenbs.com/, accessed on July 8th, 2019

[28] Jorgensen K. A., Skauge J., Christiansson P., et al. (2008) Use of IFC Model Servers Modeling Collaboration Possibilities in Practice, Department of Production, Aalborg University, Aalborg, Denmark, https://adk.elsevierpure.com/ws/files/74831/ReportIfcModelServer-Final.pdf, accessed July 8th, 2019

[29] Jotne IT (2018) EDMmodelServerTM(ifc), retrieved from http://www.jotneit.no/products/edm-model-server-ifc, accessed July 8th, 2019

[30] Beetz J., de Laat R., van Berlo L., van den Helm P. (2010) Towards an Open Building Information Model Server, *Proc. of the 10th International Conference on Design and Decision Support Systems*

in Architecture and Urban Planning, The Netherlands, https://pure.tue.nl/ws/portalfiles/portal/107876828/ddss2010_18.pdf, accessed July 8th, 2019

[31] Eastman C., Teicholz P., Sacks R., Liston K. (2011) BIM Handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors (2nd ed.), John Wiley and Sons, Hoboken, New Jersey, USA, isbn-13: 9781119287537

[32] Solihin W., Eastman C., Lee Y. C. (2017) Multiple Representation Approach to Achieve High-Performance Spatial Queries of 3D BIM Data Using a Relational Database, *Automation in Construction*, 81:369-388, doi:10.1016/j.autcon.2017.03.014

[33] Wright R. S., Haemel N., Sellers G., Lipchak B. (2013) OpenGL SuperBible: Comprehensive Tutorial and Reference, Pearson Education, isbn-13: 9780672337475

[34] Ohbuchi R., Otagiri T., Ibato M., Takei T. (2002) Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics, *Computer Graphics and Applications*, *10th Pacific Conference on Institute of Electrical and Electronical Engineers*, 265-274

[35] Sun C., Sherrah J. (1997) 3D Symmetry Detection Using the Extended Gaussian Image, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):164-168, doi:10.1109/34.574800

[36] Erturk S., Dennis T. J. (1997) 3D Model Representation Using Spherical Harmonics, *Electronics Letters*, 33(11):951-952, doi:10.1049/el:19970659

[37] Zhang Z. (2005) The Research for 3D Model Geometry Shape Similarity Matching, Zhejiang University, Hangzhou, China, http://kreader.cnki.net/Kreader/CatalogViewPage.aspx?dbCode=cdmd&filename=2006015253.nh&tablename=CDFD9908&compose=&first=1&uid=, accessed July 8th, 2019

[38] Kazhdan M., Funkhouser T., Rusinkiewicz S. (2004) Shape Matching and Anisotropy, in *Associate of Computing Machinery Transactions on Graphics*, *Associate of Computing Machinery*, vol 23(3):623-629, doi:10.1145/1186562.1015770

[39] Clark J. H. (1976) Hierarchical Geometric Models for Visible-Surface Algorithms, *Communications of the Associate of Computing Machinery*, 19(10):547-554, doi:10.1145/965143.563323

[40] Cohen J., Varshney A., Manocha D., et al. (1996) Simplification Envelopes, *the 23rd Annual Conference on Computer Graphics and Interactive Techniques, Associate of Computing Machinery*, 119-128, doi:10.1145/237170.237220

[41] Garland M., Heckbert P. S. (1997) Surface Simplification Using Quadric Error Metrics, *the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press / Addison-Wesley Publishing Co., 209-216, doi:10.1145/258734.258849

[42] Hamann B. (1994) A Data Reduction Scheme for Triangulated Surfaces, *Computer Aided Geometric Design*, 11(2):197-214，doi:10.1016/0167-8396(94)90032-9

[43] Low K. L., Tan T. S. (1997) Model Simplification Using Vertex Clustering, *the 1997 Symposium on Interactive 3D Graphics, Associate of Computing Machinery*, 75-ff, doi:10.1145/253284.253310

[44] Kalvin A. D., Taylor R. H. (1996) Superfaces: Polygonal Mesh Simplification with Bounded Error, *Institute of Electrical and Electronical Engineers Computer Graphics and Applications*, 16(3):64-77, doi:10.1109/38.491187

[45] Lounsbery M., DeRose T. D., Warren J. (1997) Multiresolution Analysis for Surfaces of Arbitrary Topological Type, *Associate of Computing Machinery Transactions on Graphics*, 16(1):34-73, doi:10.1145/237748.237750

[46] Crockford D. (2006) The Application/JSON Media Type for JavaScript Object Notation (JSON),

https://tools.ietf.org/html/rfc4627, accessed on July 8th, 2019

[47] Fang S., Yuan L. I., Gang H. U. (2002) The Summary of Text Compression Technology, *Industrial Engineering Journal*, 5(2):15-18, doi:10.3969/j.issn.1007-7375.2002.02.004

[48] Cignoni P., Ertl T. (2012) Spherical Compression of Normal Vectors, *Eurographics*, 31(2):2-7, https://pdfs.semanticscholar.org/3956/3ef791a826713b5e3ddcc837eb6dc362763c.pdf, accessed July 8th, 2019

[49] Hu Z., Zhang X., Wang H., Kassem M. (2016) Improving Interoperability between Architectural and Structural Design Models: An Industry Foundation Classes-Based Approach with Web-Based Tools, *Automation in Construction*, 66:29-42, doi:10.1016/j.autcon.2016.02.001

[50] He Z. (2016) Evolutionary K-Means with Pair-Wise Constraints, *Soft Computing*, 20(1):287-301, doi:10.1007/s00500-014-1503-6

[51] Jin S., Lewis R. R., West D. (2005) A Comparison of Algorithms for Vertex Normal Computation, *The Visual Computer*, 21(1-2):71-82, doi:10.1007/s00371-004-0271-1

[52] Hu Z., Tian P., Li S., Zhang J. (2018) BIM-Based Integrated Delivery Technologies for Intelligent MEP Management in the Operation and Maintenance Phase, *Advances in Engineering Software*, 115:1-16, doi:10.1016/j.advengsoft.2017.08.007

[53] Turk G., Levoy M. (2005) The Stanford Bunny, doi:10.24097/wolfram.91305.data

[1] Korman T. M., Fischer M. A., Tatum C. B. (2003) Knowledge and Reasoning for MEP Coordination. Journal of Construction Engineering and Management-ASCE, 129(6):627-634

[2] Khanzode A., Fischer M., Reed D. (2008) Benefits and Lessons Learned of Implementing Building Virtual Design and Construction (VDC) Technologies for Coordination of Mechanical, Electrical and Plumbing (MEP) Systems of a Large Healthcare Project, Electronic Journal of Information Technology in Construction, 13:324-342

[3] Nicolle C., Cruz C. (2011) Semantic Building Information Model and Multimedia for Facility Management, In: Filipe J., Cordeiro J. (eds) Web Information Systems and Technologies. WEBIST 2010. Lecture Notes in Business Information Processing, vol 75. Springer, Berlin, Heidelberg

[4] Vanlande R., Nicolle C., Cruz C. (2008) IFC and Building Lifecycle Management, Automation in Construction, 18(1):70-78

[5] buildingSMART (2015) National BIM Standard-United States® (NBIMS-US™) Version 3, retrieved from https://www.nationalbimstandard.org/, accessed Aug. 13th, 2018

[6] Kassem M., Succar B. (2017) Macro BIM Adoption: Comparative Market Analysis, Automation in Construction, 81:286-299

[7] Yalcinkaya M., Singh V. (2014) Building Information Modeling (BIM) for Facilities Management – Literature Review and Future Needs, In: Fukuda S., Bernard A., Gurumoorthy B., Bouras A. (eds) Product Lifecycle Management for a Global Market. PLM 2014. IFIP Advances in Information and Communication Technology, vol 442. Springer, Berlin, Heidelberg

[8] Leite F., Akcamete A., Akinci B., et al. (2011) Analysis of Modeling Effort and Impact of Different Levels of Detail in Building Information Models, Automation in Construction, 20(5):601-609

[9] Tabesh A. R., Staub-French S. (2006) Modeling and Coordinating Building Systems in Three Dimensions: A Case Study, Canadian Journal of Civil Engineering, 33(12):1490-1504

[10] Riley D. R., Varadan P., James J. S., et al. (2005) Benefit-Cost Metrics for Design Coordination of Mechanical, Electrical, and Plumbing Systems in Multistory Buildings, Journal of Construction Engineering and Management-ASCE, 131(8):877-889

[11] Yang K., Kang D., Xu P., Chen C. (2014) BIM-Based MEP Design Technology, Construction

Technology, (03):88-90
[12] Korman T. M., Tatum C. B. (2006) Prototype Tool for Mechanical, Electrical and Plumbing Coordination, Journal of Computing in Civil Engineering, 20(1):38-48
[13] Korman T. M., Lu N. (2011) Innovation and Improvements of Mechanical, Electrical, and Plumbing Systems for Modular Construction Using Building Information Modeling, AEI 2011, Building Integration Solutions, 448-455
[14] Becerik-Gerber B., Jazizadeh F., Li N., et al. (2012) Application Areas and Data Requirements for BIM-Enabled Facilities Management, Journal of Construction Engineering and Management-ASCE, 138(3):431-442
[15] Kassem M., Kelly G., Dawood N, et al. (2015) BIM in Facilities Management Applications: A Case Study of a Large University Complex, Built Environment Project and Asset Management, 5(3):261-277
[16] Schevers H., Mitchell J., Akhurst P., et al. (2007) Towards Digital Facility Modeling for Sydney Opera House Using IFC and Semantic Web Technology, Electronic Journal of Information Technology in Construction, 12:347-362
[17] Motawa I., Almarshad A. (2013) A Knowledge-Based BIM System for Building Maintenance, Automation in Construction, 29:173-182
[18] Xie H., Tramel J. M., Shi W. (2011) Building Information Modeling and Simulation for the Mechanical, Electrical and Plumbing Systems, 2011 IEEE International Conference, vol 3. IEEE
[19] Ko C. H. (2009) RFID-Based Building Maintenance System, Automation in Construction, 18(3):275-284
[20] Motamedi A., Hammad A. (2009) Lifecycle Management of Facilities Components Using Radio Frequency Identification and Building Information Model, Electronic Journal of Information Technology in Construction, 14:238-262
[21] Chen Q., Deng Z., He B., Zhang J. (2014) Key Technology and Software Implementation Method of BIM-Based MEP Emergency Management System, Journal of Information Technology in Civil Engineering and Architecture, (02):15-19
[22] Chi H., Kang S., Wang X. (2013) Research Trends and Opportunities of Augmented Reality Applications in Architecture, Engineering, and Construction, Automation in Construction, 33:116-122
[23] Kalasapudi V. S., Turkan Y., Tang P. (2014) Toward Automated Spatial Change Analysis of MEP Components Using 3D Point Clouds and As-Designed BIM Models, 2nd International Conference on 3D Vision, 2:145-152
[24] buildingSMART (2018) Home — Welcome to buildingSMART-Tech.org, retrieved from http://www.buildingsmart-tech.org/, accessed Aug. 13th, 2018
[25] Leon M., Laing R., Salman H., et al. (2014) Development and Testing of a Design Protocol for Computer Mediated Multidisciplinary Collaboration during the Concept Stages with Application to the Built Environment, Procedia Environmental Sciences, 22:108-119
[26] OCCS Development Committee Secretariat (2018) OmniClass, a Strategy for Classifying the Built Environment, retrieved from http://www.omniclass.org/, accessed Jan. 27th, 2018
[27] NBS (2015) Introducing Uniclass 2015, retrieved from https://www.thenbs.com/, accessed on Sept. 10th, 2018
[28] Jorgensen K. A., Skauge J., Christiansson P., et al. (2008) Use of IFC Model Servers Modeling Collaboration Possibilities in Practice, Department of Production, Aalborg University, Aalborg,

Denmark
[29] Jotne IT (2018) EDMmodelServerTM(ifc), retrieved from http://www.jotneit.no/products/edm-model-server-ifc, accessed Jan. 27th, 2018
[30] Beetz J., de Laat R., van Berlo L., van den Helm P. (2010) Towards an Open Building Information Model Server, Proc. of the 10th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, The Netherlands
[31] Eastman C., Teicholz P., Sacks R., Liston K. (2011) BIM Handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors (2nd ed.), John Wiley and Sons, Hoboken, New Jersey, USA
[32] Solihin W., Eastman C., Lee Y. C. (2017) Multiple Representation Approach to Achieve High-Performance Spatial Queries of 3D BIM Data Using a Relational Database, Automation in Construction, 81:369-388
[33] Wright R. S., Haemel N., Sellers G., Lipchak B. (2013) OpenGL SuperBible: Comprehensive Tutorial and Reference, Pearson Education
[34] Ohbuchi R., Otagiri T., Ibato M., Takei T. (2002) Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics, Computer Graphics and Applications, Proceedings. 10th Pacific Conference on IEEE, 265-274
[35] Sun C., Sherrah J. (1997) 3D Symmetry Detection Using the Extended Gaussian Image, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2):164-168
[36] Erturk S., Dennis T. J. (1997) 3D Model Representation Using Spherical Harmonics, Electronics Letters, 33(11):951-952
[37] Zhang Z. (2005) The Research for 3D Model Geometry Shape Similarity Matching, Zhejiang University, Hangzhou, China
[38] Kazhdan M., Funkhouser T., Rusinkiewicz S. (2004) Shape Matching and Anisotropy, in ACM Transactions on Graphics, ACM, vol 23(3):623-629
[39] Clark J. H. (1976) Hierarchical Geometric Models for Visible-Surface Algorithms, Communications of the ACM, 19(10):547-554
[40] Cohen J., Varshney A., Manocha D., et al. (1996) Simplification Envelopes, Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, ACM, 119-128
[41] Garland M., Heckbert P. S. (1997) Surface Simplification Using Quadric Error Metrics, Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press / Addison-Wesley Publishing Co., 209-216
[42] Hamann B. (1994) A Data Reduction Scheme for Triangulated Surfaces, Computer Aided Geometric Design, 11(2):197-214
[43] Low K. L., Tan T. S. (1997) Model Simplification Using Vertex Clustering, Proceedings of the 1997 Symposium on Interactive 3D Graphics, ACM, 75-ff
[44] Kalvin A. D., Taylor R. H. (1996) Superfaces: Polygonal Mesh Simplification with Bounded Error, IEEE Computer Graphics and Applications, 16(3):64-77
[45] Lounsbery M., DeRose T. D., Warren J. (1997) Multiresolution Analysis for Surfaces of Arbitrary Topological Type, ACM Transactions on Graphics (TOG), 16(1):34-73
[46] Crockford D. (2006) The Application/JSON Media Type for JavaScript Object Notation (JSON), https://tools.ietf.org/html/rfc4627
[47] Fang S., Yuan L. I., Gang H. U. (2002) The Summary of Text Compression Technology, Industrial Engineering Journal, 5(2):15-18

[48] Cignoni P., Ertl T. (2012) Spherical Compression of Normal Vectors, Eurographics, 31(2):2-7
[49] Hu Z., Zhang X., Wang H., Kassem M. (2016) Improving Interoperability between Architectural and Structural Design Models: An Industry Foundation Classes-Based Approach with Web-Based Tools, Automation in Construction, 66:29-42
[50] He Z. (2016) Evolutionary K-Means with Pair-Wise Constraints, Soft Computing, 20(1):287-301
[51] Jin S., Lewis R. R., West D. (2005) A Comparison of Algorithms for Vertex Normal Computation, The Visual Computer, 21(1-2):71-82
[52] Hu Z., Tian P., Li S., Zhang J. (2018) BIM-Based Integrated Delivery Technologies for Intelligent MEP Management in the Operation and Maintenance Phase, Advances in Engineering Software, 115:1-16
[53] Turk G., Levoy M. (2005) The Stanford Bunny.