

Intelligent Building System for 3D Construction of Complex Brick Models

HAO CAI¹, YANJIA CHEN¹, LINGLING XU¹, KHALID ELBAZ^{2,3}, AND CHENGDIAN ZHANG¹

¹Department of Computer Science, Shantou University, Shantou 515000, China

²Key Laboratory of Intelligence Manufacturing Technology, Ministry of Education, Shantou University, Shantou 515063, China

³College of Engineering, Shantou University, Shantou 515063, China

Corresponding author: Hao Cai (haocai@stu.edu.cn)

This work was supported by the China Guangdong Government through the Yangfan Project under Grant 14014600202.

ABSTRACT Brick elements are widely used for everything from designing toys to architectural designs. This article proposes an intelligent method to construct complex 3D brick models automatically. The proposed method is designed based on a set of core algorithms to generate a brick layout in accordance with a given 3D voxel model. In the system, each 3D model is sliced into tiers of flat vector polygons at first. Then, the vector polygons are converted to raster data for each tier. The bricks are built in a single tier until all tiers are completed. The proposed model is validated using two case studies: one is a series of solid sculpture models and the other is a building model. The results show that the proposed intelligent building system successfully builds visual models from brick assembly models. The proposed system can help engineers build large block models to improve the efficiency of constructing complex 3D building models.

INDEX TERMS Intelligent system, 3D building models, brick system, voxel model.

I. INTRODUCTION

Recently, intelligence algorithms become more and more popular to solve various engineering problems [1]–[3]. The demand for three-dimensional (3D) building models has increased in the field of urban design, which can be applied in architectural design, structural analysis, and other related areas with an intelligence algorithm [3]–[6]. The need for available models has prompted several researchers to develop an automated workflow for constructing 3D buildings [7], [8]. These toys often require constructing large 3D block models from a set of basic building block elements. The 3D block models are needed for several applications such as sculptures, landscape design, and automobile manufacturing.

Specifically, in engineering scenarios, EverBlock has been used to build durable wall dividers in homes and offices where blocks are available in 16 standard colors. EverBlocks are re-usable and can be transported to different locations as needed, and windows or doors can be adjusted as required. In addition, armies can use giant building blocks which replaces traditional wooden structures and provides flexible training options to build modular urban training facilities. In many cases, the tasks for these models must be completed within a strictly limited time. It is difficult and

time-consuming to construct the desired 3D shapes manually using basic building blocks.

Recently, many techniques have been proposed for complex 3D object modeling. Fukuda *et al.* [7] developed a dynamic physical model system to simulate the volume of buildings based on 3D digital models. This dynamic system uses operating rods in a network of mechanical engineering, electrical engineering, and architecture. Liu *et al.* [8] proposed an approach to assess the dimensional accuracy and structural performance of spatial structure elements using 3D laser scanning. The results indicated that the quality control of spatial structure elements is appropriate based on the 3D laser scanning. Furthermore, the detection of the elements in other systems, e.g., bridges, buildings, and culverts, can also be conducted using the suggested evaluation framework. Kim *et al.* [9] presented an automated registration process for 3D data with a 3D computer-aided design (CAD) model. Results indicated that the user can define the region of the input model data area and thus apply the developed algorithm for obtaining the stability feedback. Yang and Jian [10] developed 3D intelligent manufacturing technology that overcomes the low precision and molding inefficiency of the current 3D printing technology. However, fast output remains difficult to achieve. Hence, new technologies are needed that can more quickly generate prototypes of new designs.

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak¹.

As a different approach, Peysakhov and Regli [11] applied a so-called messy genetic algorithm optimization technique to the evolution of assemblies composed of Lego structures, which was also verified in other engineering applications [12]. In this way, each design system is a labeled assembly diagram of structural equations. Hegarty *et al.* [13] applied an assembly “Lego-scape” method: a procedural brick placement system that can rapidly create large objects. However, previous methods have poor accuracy and high computational complexity.

To improve the accuracy of such models, Kim *et al.* [14] developed an environment within which Lego models can be created from 3D mesh data. Users interactively select a region of the input model data and then apply optimization algorithms to obtain feedback regarding stability [15], [16]. Uhlmann and Peukert [17] presented an innovative approach to reconfiguring mechanical behavior based on smart building-block systems. However, none of the previous methods focuses on the balance problem while solving the construction problem. Moreover, building large-scale block models manually encounters difficulties calculating the explicit type and quantity of building blocks. It is nearly impossible to accomplish such tasks based solely on user experience. As such, it is difficult for ordinary users to assemble a desired model without instructions. Building a model usually takes one or two weeks, but in some situations a month or more may be required. In practice, excessive consumption of time increases labor costs, ultimately affecting the company’s competitive advantage.

The objective of this study is to propose a novel automated building model that assists engineers to build large and complex 3D assembly models within a limited time and to solve the balance problem during the construction process. The proposed intelligent building-block system follows automatic design principles and realizes full automation from imported model data—through model adjustment, layering, rasterization, intelligent selections of bricks, and schemes for each layer’s layout. Furthermore, the stability of two adjacent layers and the entire model is verified with a novel algorithm. To authenticate the validity of the proposed system, different experimental models were applied. The proposed model applies some engineers’ experience to the algorithm, greatly reducing the amount of calculation.

The remainder of the paper is organized as follows. Section II illustrates how the proposed system works, from inputting 3D geometric data to creating a building-block construction scheme, followed by the stabilization process in Section III; Two case studies are shown in Section IV; Extensions to the model are discussed in Section V. Finally, Section VI concludes with a summary and discussion of future work.

II. METHODOLOGY

Intelligent building-block systems automatically identify building blocks to establish a complete model. A flowchart for the proposed intelligent system is shown in Fig. 1.

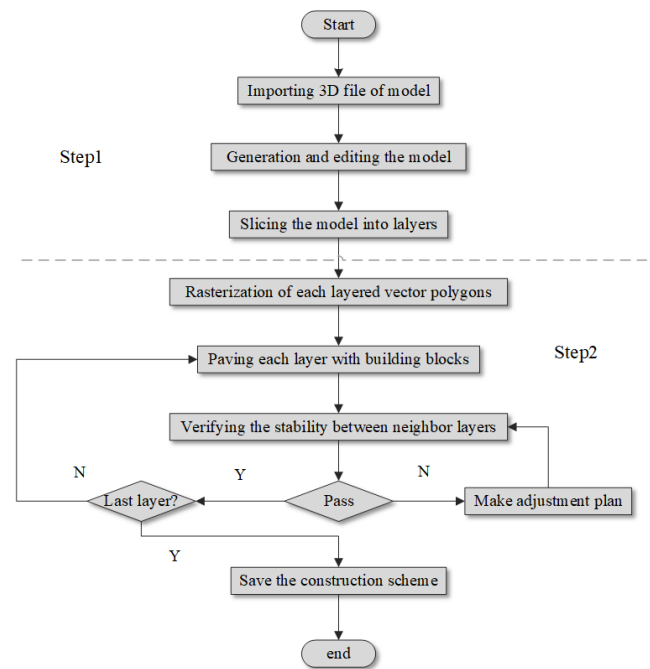


FIGURE 1. Steps of the proposed building blocks system.

This flowchart including two steps illustrates the proposed system, from inputting 3D geometric data to creating a building-block construction scheme. The schematic process involves several automatic algorithms for rasterizing vector graphics, intelligently laying each tier with bricks, and verifying the stability of the final models. A step-by-step description is given as follows.

Step1: A 3D model is established with a building information modeling (BIM) application, and properties such as shape and color are obtained by importing the original 3D file of the object model using the “.3ds” file format. The model is evenly sliced into several layers to obtain vector polygon data for each sliced layer. The height of each layer is equal to the height of the building block.

Step2: Vector polygons are rasterized in all layers, followed by basic hollowing out and editing functions. To save material and reduce weight, large building blocks are usually hollowed out (i.e., solid units are not used). Subsequently, building blocks are assembled layer-by-layer from bottom to top using the proposed algorithm. Finally, the stability of neighboring layers is verified to ensure the stability of the established model. If stability of the construction is not achieved between neighbor layers, adjustment plan is provided to satisfy the stability requirements. The process will be carried out for all the layers until all layers with their adjacent layers met the stability requirements, then the construction scheme will be finished and saved.

A. VOXELIZATION AND RASTERIZATION

The proposed approach begins by dividing the model into several layers. These layers are numbered consecutively from 1

(the bottom) to n (the top). The height of each layer is equal to that of the bricks. This enables users to convert the 3D model input data into a data matrix by representing the 3D digital model as 1×1 brick. The value of each element is one for the occupied area and zero for the empty space. This process is based on the rasterization of each layer as a vector polygon [18], [19]. The minimum unit of the building block is fixed to be a 1×1 brick. The rasterization algorithm considers the size of the fixed network, and possibly, local high-resolution vector data. The slice data of each layer are composed of several polygons. Unfortunately, these polygons are often severely crossed or overlapped, especially in large and complex object models. The data for each edge include its color information and the coordinates of the two endpoints. Processing multiple crossed or overlapping polygons and the dyeing problem need to be considered, in addition to the rasterization of a single polygon.

Actually, the rasterization of vector data is easy to execute for points and lines. One of the most popular algorithms for rasterizing lines is the digital differential analyzer (DDA) [20]–[24]. An efficient shift invariant rasterization algorithm for all-angle polygons was also developed by Ding *et al.* [25]. However, previous rasterization strategies are inefficient when treating the rasterization of the layered vector polygons. For instance, when vector data are converted into raster data in the Geographic Information System (GIS) [26]–[29], the network size needs to be determined according to the data size and accuracy [30]. The method in this study was designed to avoid these limitations.

The proposed method employs a complete rasterization algorithm based on an existing edge rasterization algorithm. The complete rasterization algorithm addresses the cross-overlapping and dyeing problems with multiple polygons as follows. First, the edges of each polygon are rasterized and dyed according to the Bresenham algorithm. Edge rasterization can be viewed as filling and dyeing the grid where the edges pass. For each polygon edge, the grid is filled at the starting point (x_0, y_0) . Then, subsequent grid coordinates are calculated to be filled repeatedly. Second, the core of the Bresenham algorithm is adjusted to determine the increment of the next grid coordinate based on the sign of the error term formed by the slope of the line. The slope of the line is represented by the following equation. The slope value k is equal to the increment of y divided by the increment of x :

$$k = \Delta y / \Delta x. \tag{1}$$

The two-dimensional space is divided into eight areas from 1 to 8 according to the threshold of the slope of the line, as shown in Fig. 2. In the first area, the line segment is closer to the x -axis, where x is stepped once each time. Then, the error term sign is used to determine whether to increment the y value by 1 as well.

The initial error term (e) is assumed to be $(-1 / 2)$, after inferring the next coordinate, $e = e + \Delta y / \Delta x$. If $e \geq 0$, y is increased by 1; otherwise, y remains unchanged. Color information for the line segment is

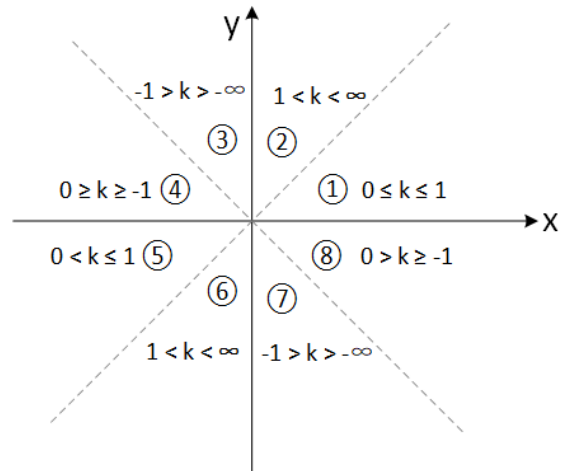


FIGURE 2. Two-dimensional space divided into eight areas.

simultaneously recorded when filling each grid where the line passes. After the rasterization of the edges is complete, the polygon is internally dyed using an intelligent system, called a flooding-fill algorithm [31], [32]. Unlike the typical seed-filling method [33], [34] and others [35]–[38], the flooding algorithm can fill the area outside the polygon and backfill the interior point. The flooding algorithm avoids the complex process of finding interior points for polygons. Each polygon is a simple closed polygon that avoids the situation depicted in Fig. 3.

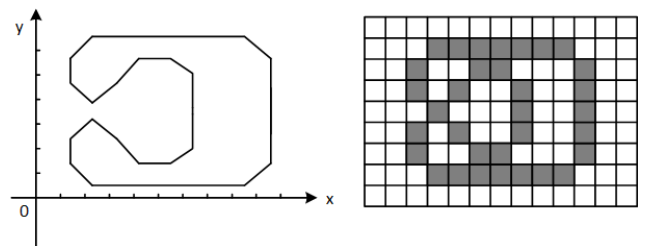


FIGURE 3. Case of a “hole” occurring after a polygon has been rasterized.

Each time the object filled by the flooding algorithm is not the entire layer, but rather an individual polygon in a layer, it is mapped to a temporary two-dimensional array that is determined by the maximum cross-sectional area of that layer of the model. This ensures that no raster on the edge coincides with the array boundaries. Coloring begins at a boundary point in the array, which is selected randomly on the condition that the grid around the point has not already been dyed. The edge of the grid is dyed. When this area is fully dyed, the remaining unstained area inside of the polygon can be backfilled, as illustrated in Fig. 4.

After completing each filling process, the polygon raster data are rebuilt to the raster data table. Based on the aforementioned process, several polygons can be rasterized in each layer. For each polygon, a temporary table is created and a coordinate shifting operation is executed. These coordinates

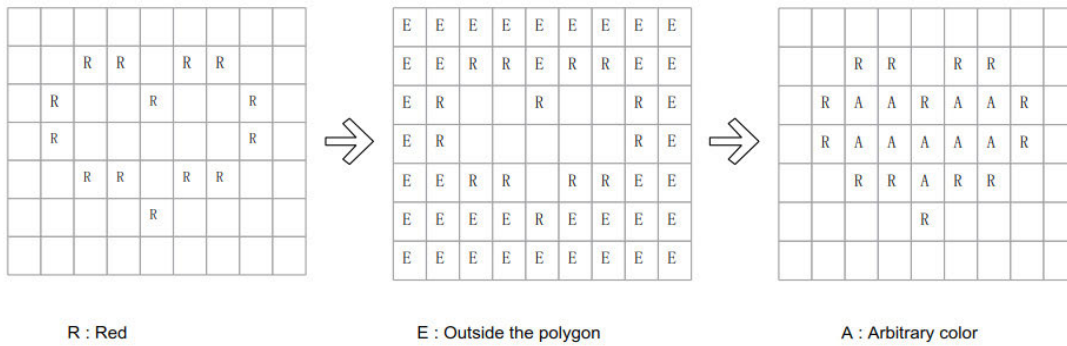


FIGURE 4. The dyeing process of the internal polygon.

are adjusted and calculated for the building process. Coloring and filling in the summary table proceed according to the following rules:

- 1) If a grid $s_1[x_1][y_1]$ in the temporary table is non-empty, with color information C_1 , it corresponds to the position $S[x][y]$ in the main table.
- 2) If the color of $S[x][y]$ is 'E' (outside the polygon) or 'A' (inside the polygon and not the edge), then C_1 is filled in at $S[x][y]$.
- 3) If the color information of $S[x][y]$ is the edge color, the value remains unchanged.

B. LAYOUT OF EACH LAYER WITH BRICKS

The diameter and height of the assembly model can reach 5.00 m. However, the length of a 1×1 brick may be less than 0.01 m. Thus, the flat layer can be equipped with up to 500×500 bricks. Building blocks can take dozens of different shapes, even when only basic pieces are considered, such as rectangular bricks and L-shaped corner bricks. Assuming that there are n bricks on a flat and there are m types of bricks, the complexity of the search is m to the power of n . The problem grows increasingly complicated if the user considers the direction of the bricks.

A randomization algorithm such as the genetic algorithm cannot improve the operational efficiency. To divide the polygon into multiple regions under the premise of dividing and conquering, dynamic programming was tested when laying the blocks in each polygon individually. However, the results indicated that this method was ineffective because its operation speed was significantly below the requirements. Since the coverage of each step in the model was too large, randomness and disorder undermined the structure of the area to be filled and thus required excessive computational time. An analysis of the results of the laying schemes given by these methods revealed that the building blocks were messy and irregular. The results differed substantially from those built by experienced engineers, who found that each layer had a certain pattern. In the central area of each polygon, the type of building block and the direction of placement are generally coherent. This rule becomes more apparent as the area of the polygon increases.

Thus, we established an algorithm based on formalized pattern rules. This algorithm consists of three stages: an initialization algorithm, brick adjustments, and merging.

1) INITIALIZATION ALGORITHM

The shapes of the rectangular bricks are selected with $(1 \times 1, 1 \times 2, 1 \times 3, 2 \times 2, 2 \times 3)$ standard bricks and (2×2) L-shaped corner bricks as basic laying components, as shown in Fig. 5. For instance, a (1×1) brick can be used to assemble any of the plane polygons using these six types of bricks. According to the geometric characteristics of the six types of building blocks, several priorities are obtained (Table 1).

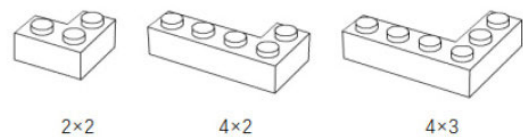











FIGURE 5. "L-turn" building block sample.

As shown in Table 1, a lower priority number in the table denotes a higher laying priority. The same type of building block can differ in priority due to differences in the placement direction and the parity of the layer number that is laid. These six types of building blocks are laid according to their different priority levels. For instance, if an odd-numbered layer is laid, polygons are first filled with 2×3 bricks, and then the 3×2 bricks are laid. In this way, the remaining space is filled with 2×2 blocks. Then, the L-shaped bricks are used until the remainder of the polygon is filled by 1×1 bricks to complete the process. It is noteworthy that three bricks— $2 \times 3, 1 \times 3,$ and 1×2 bricks—are distinguished by whether they are laid vertically or horizontally. Thus, a 2×3 brick and a 3×2 brick are considered to be two different types in the layout process. Likewise, the L-shaped corner brick varies in four directions. Because the built model is multi-colored and the building blocks are monochromatic, the color of a region covered by several building blocks will be the same with each brick type.

TABLE 1. Laying priority of building block.

Odd layer priority	Even layer priority	Building blocks (distinguished by laying direction)
1	2	2 × 3 brick 
2	1	3 × 2 brick 
3	3	2 × 2 brick 
4	4	2 × 2 L-shaped brick 
5	6	1 × 3 brick 
6	5	3 × 1 brick 
7	8	1 × 2 brick 
8	7	2 × 1 brick 
9	9	1 × 1 brick 

2) BRICK ADJUSTMENTS

To create a stable and effective model, the bricks must be assembled without any void space. To achieve this aim, 1 × 1 bricks are used to fill the layer, but this presents a risk to the stability of the overall model. To avoid this, all locations of 1 × 1 bricks are checked and adjusted according to the type and position of adjacent bricks and integrated into a variety of situations.

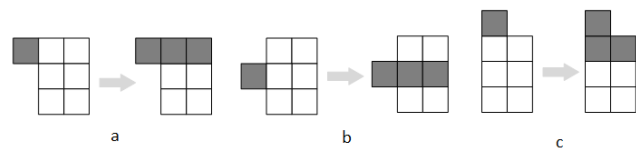


FIGURE 6. Three adjustment schemes for 1 × 1 brick adjacent to 2 × 3 brick.

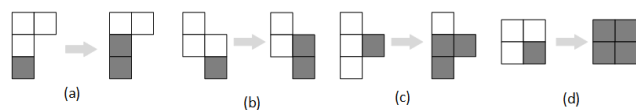


FIGURE 7. Four adjustment schemes for 1 × 1 building block adjacent to L-shaped blocks.

For example, Fig. 7 shows three situations of a 2 × 2 brick adjacent to a gray 1 × 1 bricks.

Fig. 7 depicts four different scenarios for adjusting a 1 × 1 brick in a 2 × 2 L-shaped turn. The four different scenarios are based on four different 2 × 2 L-shaped brick in the fourth line of Table 1. It can be seen that the edge or dangling point in Fig.7 (c) is superior to other conditions that need to be adjusted. When considering color, processing becomes more complex but follows the same principle. Owing to the demand for a stability test in consecutive layers, the information for each grid in the plane must include the number and type of paving blocks, as well as the color.

3) MERGING

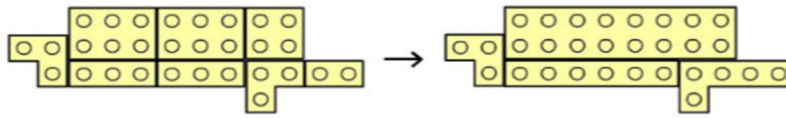
In this stage, the basic paving bricks are combined to generate large-scale bricks. The size and shape of the various bricks are stored in the system database. In Fig. 8, two 2 × 3 bricks and a 2 × 2 brick merge into 2 × 8 bricks. Two 1 × 3 bricks merge into a 1 × 6 brick. A 2 × 2 L-shaped brick and a 1 × 2 brick merge into 2 × 4 L-shaped bricks. In addition to the above six basic laying bricks, specifications for other bricks can be adjusted by the manufacturer according to demand. This process requires a search and enumeration algorithm for merging bricks. The process can be implemented when two adjacent blocks meet the assembling conditions and the assembled bricks are the same as the bricks in the database. Thus, the pieces are gradually reassembled from small to large, until no further merging can be achieved.

III. STABILIZATION PROCESS

In the 3D assembly model, the stability of each layer must be achieved to establish the blocks accurately. More broadly, individual layers in the model cannot fall off on account of being improperly connected to the other layers, as shown in Fig. 9. The structural stability of the proposed system can be achieved as follows.

First, the algorithm is adjusted to facilitate interlocking between each layer from lower to the upper blocks. Bricks of different lengths and widths represent two types of bricks. For example, 2 × 3 bricks may be regarded as two types: either 3 × 2 or 2 × 3 bricks, based on their layout direction. Moreover, these two types of bricks have different priorities in the adjacent layer. If the brick priority for a particular layer is horizontal, then the adjacent layer is vertical. The results from laying two priority orders are illustrated in Fig. 10. Selecting different bricks with different priorities in adjacent layers is beneficial when interlocking lower and upper building blocks, because doing so augments the adhesion of the adjacent layers, thus making the building block model more robust.

Second, the established model is verified layer-by-layer to check the interlocking robustness between the lower and upper layers. In the design system, to ensure the “whole” nature of the model, several data structures are tested. These tests are needed in order to record the relationship between adjacent bricks in the same layer and to consider which adjacent bricks are interlocked during the laying process. Because the single-layer laying algorithm depends on the



Two 2×3 bricks and a 2×2 brick merge into a 2×8 bricks. Two 1×3 bricks merge into a 1×6 brick. A 2×2 L-shaped brick and a 1×2 brick merge into a 2×4 L-shaped bricks.

FIGURE 8. Schemes of merging small bricks into large bricks.

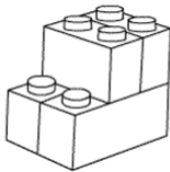


FIGURE 9. Upper and lower layer building blocks interlocked.

interlocking characteristics of the lower and upper layers, it is completely replaced by a strict standard to verify the stability of the model. Therefore, each building block is assigned to a node in the graph model, and the interlocked blocks are set by following the Union-Find algorithm. Furthermore, the algorithm transforms verification of the robustness of the building blocks into a graph theory problem. The algorithm is summarized in pseudo-code format in Algorithm 1. A more specific description follows the pseudo-code summary.

Algorithm 1 Verifying the Interlock Robustness of the Upper and Lower Layers

```

Input:  $\mathbb{N}^*$ : Each building block node;
           $M$ : number of layers of the model ( $M \in \mathbb{N}^+$ );
          setNum[k]: setNum[k] of each layer ( $k \in M$ );
Output: Whether a certain layer needs adjustment;
1: for  $i = 1; i \leq m$  do
2:   Compute connected components of each layer  $C_i \leftarrow$ 
   Establish the relationship among  $\mathbb{N}^*$ ;
3: end for
4: for  $j = 1; j \leq m$  do
5:   Compare the number of  $C_j$  with setNum[j];
6: end for
    
```

- A unique identifier is assigned to each building block, and a corresponding node in the graph is established. The identifier is composed of the number given by the single-layer building-block laying algorithm and the layer number of the model.
- The relationship between this building-block node and other nodes is then established. If the building block is interlocked with another building block in the lower layer, an edge is generated to connect the corresponding two nodes in the graph theory model. Consequently, a connected component is formed in the graph, and

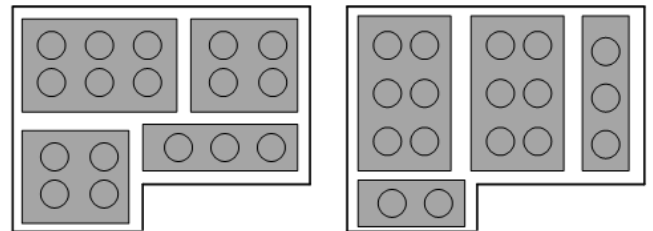


FIGURE 10. Comparison of two laying priority results.

each connected component is treated as a set. If two connected nodes belong to different sets, a merging operation is performed.

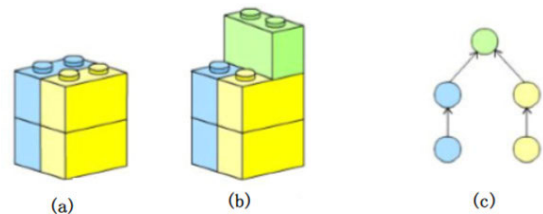


FIGURE 11. Establishing relationships among building block nodes.

The details of the Union-Find algorithm (e.g., path compression) are not discussed here. An example of the process by which relationships are established is offered in Fig. 11. In (a), blue and yellow building blocks form two independent blocks on the vertical plane. They're just put together, not connected with each other, so they are regarded as two sets of building blocks. (b) Blue and yellow building blocks form a connected block and become a set of building blocks through the knob of the third green building blocks. (c) The interlocked building blocks are then treated as an identical set in a graph.

- After scanning all the building blocks in the layer, the number of connected components in the graph is compared with a minimum set number, setNum[k], of the building block model. If the two are equal, the connection is robust. If the former is larger than the latter, a partial adjustment is made. The partial adjustment judges and matches the current situation, outputs some appropriate solutions. Then the output is later analyzed by an engineer during actual construction.

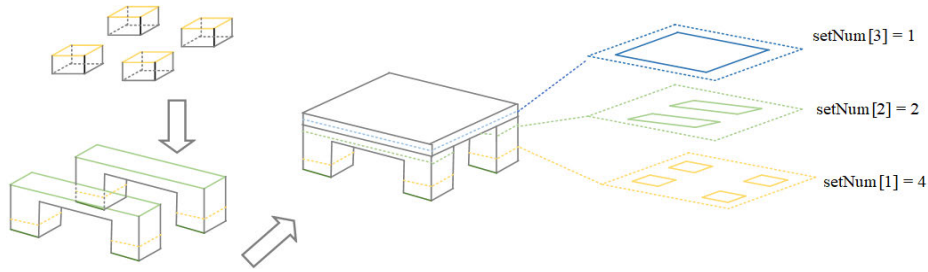


FIGURE 12. Schematic diagram of the minimum number of sets for each layer of the model.

Here, $setNum[k]$ denotes the minimum number of blocks for the k -th layer, also known as the minimum number of sets. That is, when building up to the k -th layer, if the part that is already built is “integral” (i.e., if a block can be formed), then $setNum[k]$ is 1. If it is not possible to form a block at the k -th layer, as in the case of a desk supported by four legs, then there are at least four independent blocks, and $setNum[k]$ is 4, as shown in Fig. 12. Therefore, the value of $setNum[k]$ depends in how many independent blocks there are. By verifying and adjusting the interlocked bricks between the upper and lower layers, the integrity of the entire 3D block model is established.

IV. CASE STUDIES

A. CASE 1: SERIES OF SOLID SCULPTURE MODELS

To demonstrate the performance of the proposed system, 3D scans of real tangible objects models were used, and the properties and voxels required to operate the dynamic physical model were measured. The system inputs 3D model data and outputs the corresponding construction scheme of the object model. To assemble 3D models automatically, C++ software with Open GL was used. Each model was repeated 100 times and adopted the average value.



FIGURE 13. Assembling a 3D Peppa pig model – 3D data model.

From Fig. 13 to Fig. 16 show an example of the result from a tested model of Peppa Pig. The Peppa Pig model was compared with the result of the proposed system. The 3D data file of the original model is shown in Fig. 13. These 3D data files are in “.3ds” format.

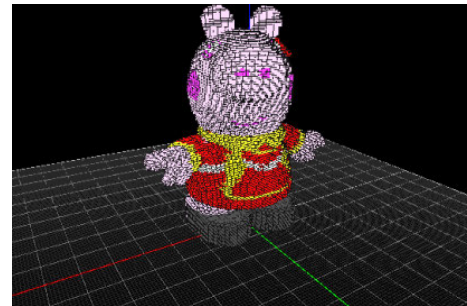


FIGURE 14. Assembling a 3D Peppa pig model - Voxelization results.

Fig. 14 shows the voxelization results of the Peppa Pig model, where the height, width, and depth are 0.99 m, 0.61 m, and 1 m, respectively, with 16,380 voxels.

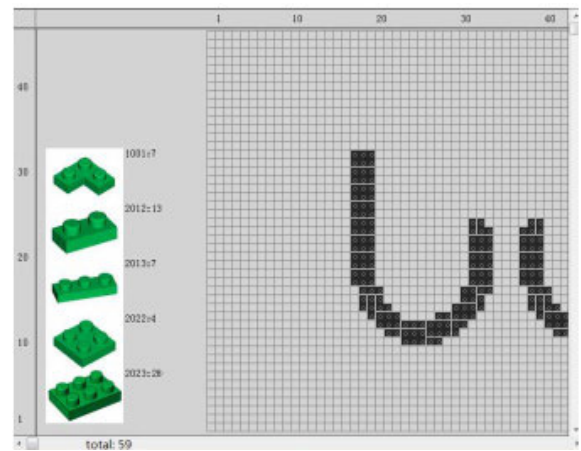


FIGURE 15. Assembling a 3D Peppa pig model – Layout of layer 1.

Fig. 15 illustrates the layouts of the model at the first layer.

Fig.16 shows the actual model assembled with bricks. The model was assembled according to the layout scheme of each layer output from the system. The weight of the bricks required for the Peppa model was 44,235.6 g, and the maximum, minimum, and average connected components of the model were 3, 1, and 2, respectively. Moreover, compared to constructing such a model manually with raw recruits, which would take two people six days to complete, our approach merely required one person over two days.

TABLE 2. 3D models constructed according to the system output scheme.

Name	3D Models		No. of Layers	Connected components			Bricks (people*day)	Weight (g)	STU-model	Lego-model	Manual methods
	height-width-depth(cm)	voxels		max	min	average					
Peppa pig	99 x 61 x 100	16380	44	3	1	2	8874	44235.6	1 x 2	1 x 2.4	11 x 2
Elephant	109 x 164 x 100	23425	44	8	1	5	19976	100349.3	2 x 2	2 x 2.3	17 x 2
Sheep	88 x 58 x 95	15183	42	4	1	3	4670	23668.36	1 x 2	1 x 2.4	9 x 2
Rhinoceros	188 x 63 x 100	22113	44	4	1	3	17976	89880.2	2 x 2	2 x 2.6	19 x 2
Teddy Bears	76 x 53 x 100	14427	44	3	1	2	8429	42387.34	1 x 2	1 x 2.5	10 x 2

**FIGURE 16.** Assembling a 3D Peppa pig model – Constructed model.

This result demonstrates the feasibility and efficiency of our algorithms and shows that a large-scale Peppa Pig model can be assembled using the proposed system.

To clarify the applicability of the proposed system to different shapes, five models were tested: Peppa Pig, an elephant, a sheep, a rhinoceros, and a teddy bear. In the simulated models, models of different shapes and various sizes were designed, as shown in Table 2, to clarify the accuracy of the proposal when automatically assembling 3D models. To adjust these models, the number of voxels and number of layers were selected. Furthermore, the statistical values for the connected components, number of bricks, block weight, and time were determined. The performance of three different schemes are compared in Table 2. The proposed intelligent building system for 3D construction of complex brick models (STU-model) and the Lego model methods [18] (Lego-model) and the traditional non-automatic analysis tools support methods (Manual methods).

The statistic in Table 2 indicates that STU-model method greatly improves the construction efficiency compared with

manual methods. Compared with the Lego-model method, the efficiency is slightly improved, STU-model greatly reduces the amount of calculation.

B. CASE 2: BUILDING

A residential building was modeled using the proposed system. The execution of the system for construction planning required approximately 300 milliseconds on a computer with an Intel^(R) Core^(TM) i7-4790 CPU and 32 GB of RAM. Construction activities followed the instructions. Fig. 14 shows this building model for urban construction analysis. The generated models were balanced, and required less material and processing time. It includes 7 floors and 17,693 building blocks.

Experiment results show that the proposed system is feasible and efficient when automatically generating constructing plans. Our method significantly decreases the time needed to generate such plans, compared to methods that do not utilize layout schemes. The system automatically generates a brick layout for each layer in less than a second. When constructing a real 3D object, an engineer simply assembles the bricks according to the scheme of each layer of the system output. Therefore, the voxelized models in this study can be used as a real brick-building guide with high reliability and speed.

V. DISCUSSION

The designs depended on a set of core algorithms to generate feasible brick layouts in accordance with a given 3D voxel model. The proposed approach consists of three main stages: an initialization algorithm, brick adjustments, and a merging and stabilization process. To obtain accurate registration results, different experimental models were applied and evaluated.

A. REAL SCULPTURES OF THE INTELLIGENT BUILDING SYSTEM

We conducted different case studies (Table 2), and the results indicate that the proposed system is feasible for establishing dynamic physical models in accordance with a given 3D



FIGURE 17. Final built building.

voxel model. Real Lego shapes were established according to the automatically generated results, demonstrating that our proposed system fully exploits the characteristics of bricks and expands the possibilities for automatically generating brick sculptures (see Table 2). By importing the 3D voxel model into the proposed system, a physical model of appropriate scale can be rapidly generated in a few minutes, with an acceptable tolerance level. Thus, users such as architects, designers, and design stakeholders can quickly and accurately produce and present physical models through 3D voxel models.

B. VISUALIZATION OF THE INTELLIGENT SYSTEM FOR PRACTICAL CIVIL APPLICATIONS

The proposed intelligent building system has many practical applications. It can convert architectural models into brick sculptures while preserving the visual features of the original models. To extend the practical applications of the intelligent system to other design scenarios, and to simulate the volume and the shape of a building in a city, the user can design other models in the city based on population, temperature, and traffic. These models can then be compared with alternatives.

For other applications, design simulations can be enriched using projection mapping technology. The user can thus simulate building facades and their design. Additionally, the proposed system can be used to simulate the tops of buildings at the microscale level or for land use at the mesoscale level by projecting design proposals for top surfaces.

The proposed system has many areas to be developed, such as the internal hollowing out of the entire building block model, local and overall force analysis. Due to lacking of professional knowledge in those areas, and the actual construction often needs to build brackets in large hollow models, this system only gives a rough reference in these aspects. In this regard, we can preliminarily imagine that if each building block can be given specific physical characteristics, combined with knowledge of mechanics, architecture, etc., it is expected to introduce a more perfect system that can be applied to other engineering projects.

VI. CONCLUSION

This article proposed a novel automated building system to assist engineers in building large, complex 3D assembly models. The following conclusions are drawn:

- 1) The proposed intelligent system provides a reasonable tool to establish models quickly with different shapes. The proposed system is particularly effective for large and complex models. In practice, the system can be used to reduce labor costs. It allows engineers to focus their expertise and effort on internal support, aesthetics, and other aspects.
- 2) To evaluate the validity of the proposed system, different case studies were tested. Case studies were conducted to consider different models, e.g. Peppa Pig. The results demonstrate that the proposed system can effectively establish vivid models.
- 3) The experimental results further demonstrated that the proposed system and algorithms offer a fully automated method of building a 3D brick assembly model. The proposed system will be beneficial for various applications that entail designing models.

REFERENCES

- [1] Y.-F. Jin and Z.-Y. Yin, "ErosLab: A modelling tool for soil tests," *Adv. Eng. Softw.*, vol. 121, pp. 84–97, Jul. 2018, doi: [10.1016/j.advengsoft.2018.04.003](https://doi.org/10.1016/j.advengsoft.2018.04.003).
- [2] H. Xiong, Z.-Y. Yin, and F. Nicot, "Programming a micro-mechanical model of granular materials in Julia," *Adv. Eng. Softw.*, vol. 145, Jul. 2020, Art. no. 102816, doi: [10.1016/j.advengsoft.2020.102816](https://doi.org/10.1016/j.advengsoft.2020.102816).
- [3] H. M. Lyu, S. L. Shen, J. Yang, and A. Zhou, "Risk assessment of earthquake-triggered Geohazards surrounding Wenchuan, China," *Natural Hazards Rev.*, vol. 21, no. 3, 2020, Art. no. 05020007, doi: [10.1061/\(ASCE\)NH.1527-6996.0000375](https://doi.org/10.1061/(ASCE)NH.1527-6996.0000375).
- [4] M.-Y. Gao, N. Zhang, S.-L. Shen, and A. Zhou, "Real-time dynamic Earth-pressure regulation model for shield tunneling by integrating GRU deep learning method with GA optimization," *IEEE Access*, vol. 8, pp. 64310–64323, 2020, doi: [10.1109/ACCESS.2020.2984515](https://doi.org/10.1109/ACCESS.2020.2984515).
- [5] P. G. Atangana Njock, S.-L. Shen, A. Zhou, and H.-M. Lyu, "Evaluation of soil liquefaction using AI technology incorporating a coupled ENN / t-SNE model," *Soil Dyn. Earthq. Eng.*, vol. 130, Mar. 2020, Art. no. 105988, doi: [10.1016/j.soildyn.2019.105988](https://doi.org/10.1016/j.soildyn.2019.105988).
- [6] S.-L. Lu, N. Zhang, S.-L. Shen, A. Zhou, and H.-Z. Li, "A deep-learning method for evaluating shaft resistance of the cast-in-site pile on reclaimed ground using field data," *J. Zhejiang Univ.*, vol. 21, no. 6, pp. 496–508, Jun. 2020, doi: [10.1631/jzus.A1900544](https://doi.org/10.1631/jzus.A1900544).
- [7] T. Fukuda, T. Tokuhara, and N. Yabuki, "A dynamic physical model based on a 3D digital model for architectural rapid prototyping," *Autom. Construct.*, vol. 72, pp. 9–17, Dec. 2016, doi: [10.1016/j.autcon.2016.07.002](https://doi.org/10.1016/j.autcon.2016.07.002).
- [8] J. Liu, Q. Zhang, J. Wu, and Y. Zhao, "Dimensional accuracy and structural performance assessment of spatial structure components using 3D laser scanning," *Autom. Construct.*, vol. 96, pp. 324–336, Dec. 2018, doi: [10.1016/j.autcon.2018.09.026](https://doi.org/10.1016/j.autcon.2018.09.026).
- [9] C. Kim, H. Son, and C. Kim, "Fully automated registration of 3D data to a 3D CAD model for project progress monitoring," *Autom. Construct.*, vol. 35, pp. 587–594, Nov. 2013, doi: [10.1016/j.autcon.2013.01.005](https://doi.org/10.1016/j.autcon.2013.01.005).
- [10] W. Yang and R. Jian, "Research on intelligent manufacturing of 3D printing/copying of polymer," *Adv. Ind. Eng. Polym. Res.*, vol. 2, no. 2, pp. 88–90, Apr. 2019, doi: [10.1016/j.aiepr.2019.03.001](https://doi.org/10.1016/j.aiepr.2019.03.001).
- [11] M. Peysakhov and W. C. Regli, "Using assembly representations to enable evolutionary design of lego structures," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 17, no. 2, pp. 155–168, May 2003, doi: [10.1017/S0890060403172046](https://doi.org/10.1017/S0890060403172046).
- [12] Z.-Y. Yin, Y.-F. Jin, S.-L. Shen, and H.-W. Huang, "An efficient optimization method for identifying parameters of soft structured clay by an enhanced genetic algorithm and elastic-viscoplastic model," *Acta Geotechnica*, vol. 12, no. 4, pp. 849–867, Aug. 2017, doi: [10.1007/s11440-016-0486-0](https://doi.org/10.1007/s11440-016-0486-0).

- [13] J. Hegarty, B. Smith, J. Jebens, and J. P. Molloy, "Assembling environments with LEGOscape," in *Proc. ACM SIGGRAPH*, New York, NY, USA, 2014, pp. 1–5, doi: [doi:10.1145/2614106.2614180](https://doi.org/10.1145/2614106.2614180).
- [14] J. W. Kim, K.-K. Kang, and J. Lee, "Automated LEGO assembly construction by interactive selection from multiple optimization techniques," in *Proc. 17th Int. Conf. Adv. Commun. Technol. (ICACT)*, Jul. 2015, pp. 182–185, doi: [doi:10.1109/ICACT.2015.7224780](https://doi.org/10.1109/ICACT.2015.7224780).
- [15] B. Bortoluzzi, I. Eftremov, C. Medina, D. Sobieraj, and J. J. McArthur, "Automating the creation of building information models for existing buildings," *Autom. Construct.*, vol. 105, Sep. 2019, Art. no. 102838, doi: [doi:10.1016/j.autcon.2019.102838](https://doi.org/10.1016/j.autcon.2019.102838).
- [16] R. Volk, J. Stengel, and F. Schultmann, "Building information modeling (BIM) for existing buildings — Literature review and future needs," *Autom. Construct.*, vol. 38, pp. 109–127, Mar. 2014, doi: [doi:10.1016/j.autcon.2013.10.023](https://doi.org/10.1016/j.autcon.2013.10.023).
- [17] E. Uhlmann and B. Peukert, "Reconfiguring machine tool behavior via smart building block systems," *Procedia Manuf.*, vol. 28, pp. 127–134, Oct. 2019, doi: [doi:10.1016/j.promfg.2018.12.021](https://doi.org/10.1016/j.promfg.2018.12.021).
- [18] R. Gower, A. Heydtmann, and H. Petersen, "Lego: Automated model construction, 32nd European study group with industry," Tech. Univ. Denmark, Lyngby, Denmark, Final Rep., Aug./Sep. 1998, pp. 81–94. [Online]. Available: <http://www.maths-industry.org/past/ESGI/32/>
- [19] C. Bridgewater, "Principles of design for automation applied to construction tasks," *Autom. Construct.*, vol. 2, pp. 57–64, Dec. 1993, doi: [doi:10.1016/0926-5805\(93\)90035-V](https://doi.org/10.1016/0926-5805(93)90035-V).
- [20] P. G. McCrea and P. W. Baker, "On digital differential analyzer (DDA) circle generation for computer graphics," *IEEE Trans. Comput.*, vols. C-24, no. 11, pp. 1109–1110, Nov. 1975, doi: [doi:10.1109/T-C.1975.224140](https://doi.org/10.1109/T-C.1975.224140).
- [21] J. Bresenham, "Pixel-processing fundamentals," *IEEE Comput. Graph. Appl.*, vol. 16, no. 1, pp. 74–82, 1996, doi: [doi:10.1109/38.481626](https://doi.org/10.1109/38.481626).
- [22] D. Hearn and M. P. Baker, *Computer Graphics With OpenGL*, 2004.
- [23] D. E. Knuth, "Digital halftones by dot discussion," *ACM Trans. Graph.*, vol. 6, pp. 245–273, Oct. 1987, doi: [doi:10.1145/35039.35040](https://doi.org/10.1145/35039.35040).
- [24] W. Guangchen, Z. Jianzhang, and L. Yan, "A high-effective algorithm for rasterization of vector data: Winding number algorithm," *Sci. Surv. Mapping*, vol. 34, pp. 50–52, Feb. 2009.
- [25] Y. Ding, C. Chu, and X. Zhou, "An efficient shift invariant rasterization algorithm for all-angle mask patterns in ILT," in *Proc. 52nd Annu. Des. Autom. Conf.*, 2015, p. 72.
- [26] Y. Wang, Z. Chen, L. Cheng, M. Li, and J. Wang, "Parallel scanline algorithm for rapid rasterization of vector geographic data," *Comput. Geosci.*, vol. 59, pp. 31–40, Sep. 2013, doi: [doi:10.1016/j.cageo.2013.05.005](https://doi.org/10.1016/j.cageo.2013.05.005).
- [27] H.-M. Lyu, S.-L. Shen, A.-N. Zhou, and W.-H. Zhou, "Flood risk assessment of metro systems in a subsiding environment using the interval FAHP-FCA approach," *Sustain. Cities Soc.*, vol. 50, Oct. 2019, Art. no. 101682, doi: [doi:10.1016/j.scs.2019.101682](https://doi.org/10.1016/j.scs.2019.101682).
- [28] H. M. Lyu, W. H. Zhou, S. L. Shen, and A. N. Zhou, "Inundation risk assessment of metro system using AHP and TFN-AHP in Shenzhen," *Sustain. Cities Soc.*, vol. 56, Oct. 2020, Art. no. 102103, doi: [doi:10.1016/j.scs.2020.102103](https://doi.org/10.1016/j.scs.2020.102103).
- [29] H.-M. Lyu, S.-L. Shen, A. Zhou, and J. Yang, "Risk assessment of mega-city infrastructures related to land subsidence using improved trapezoidal FAHP," *Sci. Total Environ.*, vol. 717, May 2020, Art. no. 135310, doi: [doi:10.1016/j.scitotenv.2019.135310](https://doi.org/10.1016/j.scitotenv.2019.135310).
- [30] R. C. Gonzalez and R. E. Woods, *Digital Images Processing*, 2008.
- [31] S. Burtsev and Y. Kuzmin, "An efficient flood-filling algorithm," *Comput. Graph.*, vol. 17, pp. 549–561, Dec. 1993, doi: [doi:10.1016/0097-8493\(93\)90006-U](https://doi.org/10.1016/0097-8493(93)90006-U).
- [32] H.-M. Lyu, S.-L. Shen, J. Yang, and Z.-Y. Yin, "Inundation analysis of metro systems with the storm water management model incorporated into a geographical information system: A case study in shanghai," *Hydrol. Earth Syst. Sci.*, vol. 23, no. 10, pp. 4293–4307, Oct. 2019, doi: [doi:10.5194/hess-23-4293-2019](https://doi.org/10.5194/hess-23-4293-2019).
- [33] M. A. Sid-Ahmed, *Image Processing: Theory, Algorithms, and Architectures*, Int. ed. New York, NY, USA: McGraw-Hill, 1995. [Online]. Available: <https://lib.ugent.be/en/catalog/rug01:001044647>
- [34] L. Feng and S. H. Soon, "An effective 3D seed fill algorithm," *Comput. Graph.*, vol. 22, pp. 641–644, Dec. 1998, doi: [doi:10.1016/S00978493\(98\)00073-9](https://doi.org/10.1016/S00978493(98)00073-9).
- [35] F. Xue, W. Lu, K. Chen, and A. Zetkovic, "From semantic segmentation to semantic registration: Derivative-free Optimization–Based approach for automatic generation of semantically rich as-built building information models from 3D point clouds," *J. Comput. Civil Eng.*, vol. 33, no. 4, Jul. 2019, Art. no. 04019024.
- [36] H. M. Lyu, W. J. Sun, S. L. Shen, and A. N. Zhou, "Risk assessment using a new consulting process in fuzzy AHP," *J. Construct. Eng. Manage.*, vol. 146, no. 3, 2020, Art. no. 04019112, doi: [doi:10.1061/\(ASCE\)CO.1943-7862.0001757](https://doi.org/10.1061/(ASCE)CO.1943-7862.0001757).
- [37] N. Zhang, S.-L. Shen, A. Zhou, and Y.-S. Xu, "Investigation on performance of neural networks using quadratic relative error cost function," *IEEE Access*, vol. 7, pp. 106642–106652, 2019, doi: [doi:10.1109/ACCESS.2019.2930520](https://doi.org/10.1109/ACCESS.2019.2930520).
- [38] K. Elbaz, S.-L. Shen, W.-J. Sun, Z.-Y. Yin, and A. Zhou, "Prediction model of shield performance during tunneling via incorporating improved particle swarm optimization into ANFIS," *IEEE Access*, vol. 8, pp. 39659–39671, 2020, doi: [doi:10.1109/ACCESS.2020.2974058](https://doi.org/10.1109/ACCESS.2020.2974058).



HAO CAI received the B.S. degree in civil engineering from Central South University, Changsha, China, in 1997, and the M.S. degree in information processing and the Ph.D. degree in computer science from the University of York, York, U.K., in 2001 and 2005, respectively.

From 2005 to 2008, he was a Research Associate with the Safety Critical System Laboratory, London Metropolitan University, London, U.K. From 2008 to 2012, he was a Technical Lead at Altran Praxis High Integrity Systems, Bath, U.K. He is currently the Technical and Scientific Director for multiple complex safety critical systems, such as the NATS Air Traffic Control System, U.K. and the Engine Monitoring Unit (EMU) System for Trent XWB and Trent 900 Jet Engine. Since 2012, he has been a Professor (formerly the Department Chair) with the Department of Computer Science, Shantou University, China. He is the author of three books and more than 50 articles. He holds more than ten patents. His research interests include safety critical systems, hardware and software architectures for high-integrity real-time systems, and dependable and distributed computing systems for industrial robotics.



YANJIA CHEN received the B.S. degree from Guangdong Pharmaceutical University, Guangzhou, China, in 2018. She is currently pursuing the master's degree with the Department of Computer Science and Technology, College of Engineering, Shantou University, China.

Her research interests include real-time big data systems and real-time systems.



LINGLING XU received the B.S. degree in software engineering from Central China Normal University, Wuhan, China, in 2019. She is currently pursuing the master's degree with the Department of Computer Science and Technology, College of Engineering, Shantou University, China.

Her research interests include real-time distributed systems and real-time big data systems.



KHALID ELBAZ received the Ph.D. degree from the Department of Civil Engineering, School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2019. He is currently a Postdoctoral Fellow with the School of Civil and Environmental Engineering, Shantou University, China. At fine levels, he has developed some computational models for solving engineering problems to support sustainable and resilient infrastructures. Finally, he has published several articles in top-tier technical journals and international conferences. His research interests include developing and applying machine and deep learning techniques as well heuristic optimization algorithms to solve and adjust issues related to intelligent building systems.



CHENGDIAN ZHANG received the B.S. and M.S. degrees from the Department of Mathematics, Xi'an Jiaotong University, China, in 1982 and 1984, respectively, and the Ph.D. degree from the Institute for Applied Mathematics, University of Bonn, Germany, in 1989.

From 1984 to 1985, he was a Teaching Assistant with Xi'an Jiaotong University. From 1985 to 1989, he was a Research Assistant with the Institute for Applied Mathematics, University of Bonn. From 1989 to 1990, he was a Research Assistant with the OST-Asian Institute, University of Bonn. From 1990 to 1995, he was a Software Engineer at EPIS GmbH, Duisburg, Germany. From 1997 to 1998, he was the Manager of Software Development at Allkauf Haus GmbH, Mönchengladbach, Germany. From 1998 to 2010, he was a Senior Engineer at Metris GmbH, Hilden, Germany. Since 2010, he has been an Associate Professor with the Department of Computer Science and Technology, Shantou University, China. He is the author of two books and more than 20 articles.

...