ShapeArchit: Shape-Inspired Architecture Design with Space Planning

Weidan Xiong a,*, Pengbo Zhang b, Pedro V. Sander c,*, Ajay Joneja c,*

a Nanyang Technological University, Singapore
b University of Macau, Macau Special Administrative Region, China
c Hong Kong University of Science and Technology, Hong Kong Special Administrative Region, China

A R T I C L E   I N F O

Article history:
Received 26 March 2021
Received in revised form 13 August 2021
Accepted 2 September 2021

Keywords:
Computer-aided design
Architecture design
Metaheuristics

A B S T R A C T

Various techniques have been proposed to improve the level of automation in stages of architectural design. We present an interactive interface, along with optimization algorithms focused on designing early stage symbolic architecture. The architect specifies some simple shape requirements by inputting a few binary images that resemble the shape of the building from different viewpoints. Our optimizer uses these to generate a conceptual 3D design that is guided by various aesthetic and structural requirements. We also develop an approach for planning the inner space of the given architecture model. The architect specifies idealized requirements on the key functional rooms by using instances of simple 3D primitive shapes. The pose and location of each instance are parameters of the optimization. By coupling this process together with the shape inspired exterior design process, we can construct good initial conceptual designs. We develop modified cuckoo search algorithms to efficiently obtain feasible solutions via offline optimization. Several examples are presented to illustrate the methodology and results. User studies based on a proposed dataset and interviews with domain experts have been carried out to demonstrate the usability and effectiveness of our system and algorithms.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

Architectural design involves many complex analytical problems. The quality of early conceptual design plays a critical role throughout the design life-cycle. Typically, the architect first converts his/her ideas into building sketches or drawings by considering the functional requirements, the surroundings, and space constraints (e.g. the footprint, height). This concept design is then converted into a preliminary design. Later, engineers and specialists are engaged to detail the plan: in this stage, the design is still fluid and often undergoes multiple rounds of changes and modifications. Ideally, however, the intent and outlook of the concept design is preserved through these changes. The detailed design must consider a variety of criteria (e.g. functionality, construction soundness, among others).

Several modern architects use non-architectural shapes as an inspiration for the design of landmark buildings (see Fig. 1(a)). The Bird’s Nest stadium in Beijing (architects: Jacques Herzog and Pierre de Meuron), the wood art museum in Harbin (architect: Ma Yansong) are notable examples. The process of converting a shape idea into an initial 3D design model is cumbersome. We introduce a methodology to provide good initial designs based on a small number of input images. In the system described here, the user provides three binary images as input.

Our architectural exterior design is guided by the idea that the generated model should faithfully mimic each input image from some viewpoints. One of the inputs is typically used as the plan view, and represents the footprint of building. The range of viewpoints can be constrained during the optimization in order to properly account for suitable viewing locations imposed by the city landscape. There are several other topological, physical or functional constraints that guide the design of buildings. Some examples of such constraints include structural stability (e.g. preventing long cantilever arms, middle-heavy arched shapes etc.), floor area, window area, etc. We encode several such constraints into the function evaluating quality of a given design. This function can then be optimized via an offline search process to generate low energy optimized designs.

Furthermore, during the process of early concept design, designers might intend to reserve several key internal spaces for specific functions, such as a multi-floor atrium, hall, etc. In general, the desired shape of internal spaces may be complex. However, in early stages of design, it is sufficient to use a few commonly used and regular shapes to allocate the volumetric spaces within...
the conceptual design, as shown in Fig. 1(b). A common primitive is the cuboid — many rooms and halls in buildings are essentially cuboids. Primitives such as cylinders, half-ellipsoids, and truncated cones are useful for special functionality.

We seek to automatically arrange internal space with user specified space entities, under the constraints on given exterior architectural shape. We provide several common 3D space shapes as options. The user is able to specify the size, orientation and spatial position of the corresponding units in our interface. An evaluation on fitness of the arrangement is performed. By employing a multi-objective search algorithm we can find a low energy optimized arrangement for a given set of space units. However, with the need of large-scale space elements, the units might not able to be perfectly placed into input architectural design. For users with preference on integrity of internal units, we provide the option on slightly modifying the overall shape of input building to accommodate this possibility. Moreover, architectural design often begins with specification of certain functional spaces. Hence, with user specified exterior shapes and interior elements, we explore the option on integrating interior design with exterior design. This global function can then be optimized via a search process to obtain low energy designs.

To summarize, we introduce a new interactive approach: ShapeArchit, to assist architects in quickly generating new designs inspired by images of non-architectural forms as well as locating specified internal spaces within the design. We first develop a building shape that yields faithful silhouettes of the images from optimal locations by solving an energy minimization problem. Our model encodes physical as well as functional constraints. Furthermore, with the need on planning the interior space of the
building shape, we provide an approach to arrange the location and orientation of user-specified volumetric spaces. If the location of (optimized) internal spaces is fixed, our approach can run a subsequent optimization of the exterior shape. Moreover, by solving a global model, we develop the approach to generate both building shape and internal plan. In an earlier work, we had mainly addressed the exterior design problem and proposed a modified single-objective cuckoo search algorithm to find a feasible solution [2]. In this paper, we further address the space planning problem and integrate it with the exterior design process into a global design process and optimization. In achieving these objectives, we extend the modified cuckoo search algorithm for solving multiple objective functions and improve the algorithm efficiency. These modifications can easily be adopted in other domains. Furthermore, we have also performed a user study of our system regarding its utility and usability, engaging users from diverse backgrounds and domains.

2. Related work

In the past decades, many architectural interactive decision software and structure optimization techniques have been developed to assist architects and engineers designing (parts of) buildings, and refining existing designs. Early stage concept design plays an important role in determining the eventual project. Refinement and development of the preliminary design lead to many improvements [3]. The preliminary design process consists of several sub-problems, such as basic shape design, building orientation determination and space allocation [4].

Modeling. Generating 3D models from 2D shapes has been of interest for a long time. Several approaches have been proposed to recreate a 3D model from one or multiple 2D sketches [5], engineering drawings [6] or lines/contours/skeletons extracted from photos or images [7,8]. This has been extended even to models of assembled parts (each with its assigned sketch-lines in each view) [9].

Our approach borrows more from the visual hull introduced by Laurentini [10]. In general multiple images from a given (or any) view may not be compatible with a single solid object. This problem was studied by Trager et al. [11]. The shadow art technique allows minimal rigid deformation of the 2D silhouettes such that the generated visual hull would cast a set of shadows from given light sources, matching the deformed silhouettes as accurately as possible [12]. However, none of these approaches consider the structural integrity of generated models, nor other architectural needs, such as Total volume and Window ratio.

Our exterior design explores ranges of viewing positions based on constraints of both silhouettes and architectural requirements. By allowing the changes on image ratio of silhouettes instead of rigid deformation, our method attempts to preserve the input shape as accurately as possible. We use a voxel model for shape optimization in order to improve computational efficiency. Standard post-processing techniques are employed to create better rendered views of the final result. We use a higher resolution binary image to recompute a finer voxel model of the output; for an alternative model style, this is converted into a mesh model via a standard marching cubes algorithm of Lorensen and Cline [13]. To provide mesh with smoothness and features that are suitable for architectural shapes, we then employ a smoothing framework with roof flattening to create a plausible design [2]. Finally, some simple facade textures are used to provide more realistic rendered views of the model.

Architectural Designs. Automatic generation of architectural forms from image inputs, interactively drawn sketches or under functional requirements is a field that has been widely explored, with application in movies and virtual worlds [14]. One approach is through the use of procedural modeling [15,16]. Users can construct a valid description of building or city based on a set of grammatical rules with block units such as windows, walls, roof structures [17,18]. The model geometry may be inferred via images of the building with machine learning techniques [19]. Generation on specific style of architecture (ancient architectures, urban building , Chinese architectures, etc.) also attracted researchers attention [19-21].

Two key differences between our problem and the procedural modeling approaches are that we do not employ any shape grammars, nor begin with any images of buildings. Our optimization model is non-linear, non-convex and discontinuous. We modified an optimization algorithm called cuckoo search [22] because of the effectiveness of this technique in solving problems of a similar nature.

Structural integrity of a design is an important criterion in architecture. But it is difficult to explore design by detailed structural analysis in early stages of the design, because (i) the structural model has not been determined at the early stages of shape design and, as observed by Jiang et al. [23], (ii) even if a sufficiently detailed structural model may be inferred from a shape, solving for integrity requires very slow and complex numerical analysis which is not conducive to searching over a large space of shapes. A fast evaluation of the structural stability is preferable, even if it sacrifices some accuracy. One approach is by modeling the surface as a mesh and applying a simplified mechanics model, e.g. the thrust network method [24]. Efficient and robust techniques for statics-driven interactive mesh design have also been developed [25]. An alternate approach is to integrate structural stability into a procedural modeling framework [26]. Another approach is employing efficient physics engines.

Our implementation uses a voxel-based solver that employs a simplified finite difference model for dynamic simulation of soft objects [27]. This model uses several simplifying assumptions for computing dynamic distortions of a model.

Internal Space Planning. Space planning is one of the most challenging open problems during the architectural design process, which seeks to produce arrangements of a set of space elements satisfying specific design criteria. The general case of the problem is akin to bin packing and known to be NP-complete [28]. It is hard even to find satisfying solutions in a short time [29]. Many researchers employed evolutionary computing approaches [30] to generate feasible floor plans under requirements on design quality, aesthetics and usability [31,32].

Major study of this problem started from the 1960s, mostly focused on generating 2D floor-wise plans with rectangular elements [33]. Harada et al. [34] developed interactive architectural layout design system based on shape grammars. Approaches for extruding feasible floor plans into a 3D building model were introduced by Yin et al. [35]. Moreover, Kurmann et al. [36] tackled the problem on city planning or single object with elements into a volumetric form. With large-scale image data on residential buildings, Merrell et al. [37] presented a method for constructing three-dimensional building with algorithm-generated floor plans. Recently, Fan and Wonka [19] developed a framework that can generate multiple 3D completed designs according to partially observed building models from photographs.

Our space planning problem seeks to pack objects of different sizes into containers with finite volume. We handle extremely irregular shape of containers and items of arbitrary shapes for packing.

Moreover, like virtual machines packing, intersection between items are allowed, which make our problem harder to approximate [38]. We explore ranges of possible space arrangements...
3. Interactive architectural design process

This section introduces the exterior design process in our interactive architecture design system. We start with the three shape templates $I_1$, $I_2$, $I_3$ (e.g., "A", "M", and "C" in Fig. 2(b)). We seek to design a building such that when viewed from three different locations, or cameras, each building silhouette matches a corresponding input shape template. Initially the three cameras are placed in orthogonal positions, with two at ground level and one as the top view. The user is then able to manipulate several camera and image parameters interactively to fine-tune this shape. At any point of the design session, the user can invoke the optimization algorithm (Section 3.4) using the current configuration as the input.

3.1. Discrete visual hull

By projecting the three shape templates from their respective camera positions, a visual hull can be constructed from the intersection of their silhouette cones (see Fig. 2(a)). For simplicity and efficiency, we use a voxel representation for the visual hull.

**Optimizations** To improve efficiency during active user interaction and update the voxel grid in real-time, the system downsamples the voxel grid by $2 \times 2 \times 2$ for all computations except for the more expensive structural integrity computation which is downsampled by $4 \times 4 \times 4$. The results are upscaled based on nearest neighbors for the visualization. Table 1 lists all the resolutions used in our examples.

3.2. Objectives

When designing a building, there are several aesthetic, structural, and practical considerations that must be taken into account. We categorize some of the important considerations. These are described below, with a description of how we measure the quality, or suitability, of the shape design.

The first two objectives seek to measure the aesthetic and structural properties of the design:

- **Shape template integrity** measures how faithfully the resulting building silhouettes match corresponding shape templates. We accomplish this by tracing a ray for each image pixel and setting it to 1 if it intersects an active voxel. For each shape template $j$, we compute a new binary image $I^*_j$ representing projection of the current voxel grid onto the image plane of camera $j$. We define the error as the Frobenius-norm between the desired shape template $I_j$ and the current shape $I^*_j$ integrated over all three images:

$$E_i = \sum_{j=1}^{3} \frac{||I^*_j - I_j||^2}{w_j}. \quad (1)$$

where $w_j$ is the number of pixels in binary image $I_j$.

- **Structural integrity** measures the ability of the structure to be able to support itself without breaking or collapsing. For efficiency and simplicity, we use a simple physical simulation in the form of voxel-based mass–spring lattice. We assign each voxel a fixed weight $w$, connect it to its six direct neighbors and remove corresponding degrees of freedom, constrain the position of the ground voxels, and simulate the system under the force of gravity until convergence. Our voxel-based solver uses a simplified finite difference model for dynamic simulation of soft objects [27]. The only external force that we consider is gravity (we neglect effects of wind loads), and the dynamic model is run long enough until each voxel has nearly zero velocity and acceleration. The magnitude of a voxel’s displacement is used as a proxy for structural instability. Voxel displacements larger than $\epsilon_m$ voxel units denote moderate stress contributors, color coded in yellow (light) to red (heavy) in Fig. 3. We measure structural integrity as the fraction of active voxels that contribute to severe stress (color coded in red):

$$E_s = g_{red} / g_a$$

where $g_{red}$ is the number of “red” voxels and $g_a$ is the number of active voxels (one voxel thick surface layer of downsampled visual hull solid). In our experiments, we use $w = 50$ kg, $\epsilon_m = 0.5$. These parameters can be adjusted by the architect to produce a more (or less) conservative design.
In addition to the above criteria, we also provide and consider the following practical considerations in our interactive system:

- **Total volume** $V_t = \frac{g_a}{D^3}$ measures the overall size of the building. It is the number of active voxels $g_a$ normalized by total cubic dimensions of the voxel space $D^3$ in number of voxels.

- **Floater volume** $V_f = \frac{g_f}{g_a}$ measures the volume of floaters, where $g_f$ is the total number of voxels in the building components that are disconnected from the ground, and $g_a$ is the total number of external voxels. Such disconnected regions are undesirable as they would need to be completely removed or require additional support. Our implementation uses a flood fill algorithm starting at the ground voxels to identify floater voxels $g_f$. Such floaters are visualized in black (see Fig. 3).

We also considered criteria for building footprint (Ground area), total Floor area, and the ratio between window voxels and interior voxels (Window ratio). However, based on our user study with professional architects, these measurements were deemed less important during the conceptual design stage, but potentially important during detailed design. We provide a framework for incorporating these criteria in the supplemental material.

### 3.3. Parameters

To optimize the building shape, the system interface provides the user with a number of adjustable camera and image parameters. Both orthographic $x_o$ and perspective $x_p$ projection are supported. The parameter vector for orthographic projection is:

$$x_o = (t_1, s_1, \theta_1, t_2, s_2, \theta_2, t_3, s_3) \tag{3}$$

where $t_j$, $s_j$, $\theta_j$ are the 2D image space translation, scale, and camera rotation about the $y$-axis of shape template $j$. Note that the third camera is overheard and does not include a $y$-axis rotation parameter. Moreover, user could set constraints on the range of feasible viewing angles of two cameras at the ground level, which are $\theta_1$ and $\theta_2$, as Fig. 2(b).

In perspective projection mode, we also incorporate a distance parameter $d_j$ that allows camera $j$ to be moved toward or away from the building:

$$x_p = x_o + (d_1, d_2, d_3) \tag{4}$$

Given the distance, the field of view of each camera is then computed so as to fit the entire structure in the image plane.

### 3.4. Modified Cuckoo search optimization

In this section, we describe our optimization algorithm to automatically adjust the parameters in order to achieve a result that better addresses the desired objectives in shape driven architectural design problem.

**Parameters.** If an orthographic projection is desired our optimization has $\text{dim}(x_o) = 11$ parameters, whereas for a perspective projection there are $\text{dim}(x_p) = 14$ parameters.

**Objective function.** We seek to minimize a function based on the objectives of Section 3.2. The relative importance of these objectives is subjective. For the results we considered a weighted combination of template integrity, structural integrity, total volume, and floater volume. Our energy function is:

$$f_{\text{shape}}(x) = w_1 E_t + w_2 E_s + w_3 (1 - V_t) + w_4 V_f \tag{5}$$

where $E_t, E_s, V_t$, and $V_f$ are the energy terms described above and computed based on parameters $x$. In our results, we use the weights $w_1 = 0.1$, $w_2 = 5$, $w_3 = 0.75$, and $w_4 = 5$, which we found to be a good compromise between the objectives. The weights can be easily adjusted before or during the design session based on the needs of the designer. Users may easily form an alternate energy function with any combination of these criteria as the objective function.

Since the energy function $f_{\text{shape}}(x)$ in Eq. (5) is highly non-linear, we investigated different probabilistic techniques to achieve an approximate global minimum. We have found a novel adaptation of the cuckoo search metaheuristic to be a suitable choice due to its simplicity and flexibility in exploring different candidate solutions.

A standard cuckoo search strategy (referred here as ORIG) maintains a set of $n$ nests, each with a potential solution, or egg. In each iteration, a new solution, or cuckoo egg, is generated from a randomly selected egg via a Lévy flight. The cuckoo egg replaces the egg in a randomly selected nest if it improves upon the latter. At the end of each iteration, the algorithm stores the current best solution and drops a fraction $p_o$ of the nests, replacing them by new random solutions. In our modified cuckoo search, instead of randomizing all of the parameters in a candidate solution, we only modify a selected fraction of the parameters. This modified strategy is called fixed likelihood parameter replacement (FLPR). Furthermore, we improve the result of FLPR by fine-tuning the values of $p_o$ for different parameters (adaptive likelihood parameter replacement, or ALPR). These modifications, which are described in Section 3.4.3, significantly improve the outcome.

### Periodic structural simulation

$E_s$ is the most time-consuming part of computing $f_{\text{shape}}$. We explored the strategy on computing $E_s$ once every $p$ iterations to improve the algorithm efficiency (especially for parameter tuning stage). To compensate for the reduction in this term, in iteration $k$: $f'(x) = f_{\text{shape}}(x)$, while in the iterations $k+1$ to $k+p-1$: $f(x) = f_{\text{shape}}(x)^{w_3} w_4 E_s + w_4 (1 - V_f) + w_4 V_f$. The energy comparison is based on the principle of equivalence, where $f_{\text{shape}}(x'^*)$ would be compared to $f_{\text{shape}}(x')$. For iteration $k$ with $E_s$, the solution in nest $j$ rolls back to the solution in iteration $k - p$ if it deteriorates.

Our modified cuckoo search is summarized in Algorithm 1. In our experiments we let the number of nests $n = 25$. The convergence behavior of the algorithm can be seen in the graphs in Fig. 8(a) and supplemental material. The following sections describe how we generate new candidate solutions in the different stages of our algorithm.

### 3.4.1. Generating initial candidate solutions

One candidate solution is initialized with the default parameters, having equidistant orthogonal cameras. Another candidate solution starts with the parameters currently set in the user interface by the designer. The parameters of the remaining candidate solutions are then randomly generated using a uniform distribution over the range of valid parameter values.

We have found that we achieve faster convergence if we initialize some of the key parameters in a small fraction of the random candidate solutions. More specifically, we set camera angles $(\theta_1, \theta_2)$ in six of the solutions to $(0, \pi / 4)$, $(\pi / 4, 0)$, $(0, \pi)$, $((\pi / 4, 0)$, $0, \pi / 4)$, and in the case of perspective projection, we further set camera distances $d_1, d_2, d_3$ of seven solutions to uniformly distributed values within the parameter range (all three cameras with matching distance in each egg).

### 3.4.2. Generating Solutions via Lévy Flights

The Lévy flight is equivalent to a random walk, but with a random step size that is based on a Lévy distribution which has an infinite variance with an infinite mean. The simple version of Lévy distribution can be defined as: $\text{Lévy} \sim u = r^{-\alpha}$. We follow the original cuckoo search strategy of Yang and Deb [22] which uses
Algorithm 1: Modified Cuckoo Search

1. Generate initial population of \( n \) host nests (Section 3.4.1)
2. while termination condition not met in iteration \( k \) do
   3. Get \( n \) cuckoo eggs \( \{x_i^1\} \) by Lévy flights from \( \{x_i\} \) (Section 3.4.2)
   4. Calculate \( f(x_i^1) \) in parallel using \( n \) threads
   5. for each nest \( n_i \) do
      6. Choose random nest \( n_j \) to lay the egg
      7. if \( f(x_i^1) < f(x_j) \) then
         8. Replace \( x_j \) by \( x_i^1 \)
      end
   end
   10. Get \( n \) cuckoo egg \( \{x_i^2\} \) by mutation from \( \{x_i\} \) (Section 3.4.3)
   11. Calculate \( f(x_i^2) \) in parallel using \( n \) threads
   12. for each nest \( n_i \) do
      13. if \( f(x_i^2) < f(x_i) \) then
         14. Replace \( x_i \) by \( x_i^2 \)
      end
   end
   18. end
19. return \( x_{best} \)

Fig. 4. Generating a new candidate solution \( x_i^1 \) and mutating a solution \( x_i^1 \) with the strategy of ORIG, FLPR and ALPR.

Lévy flights when generating a cuckoo: \( x_i^1 = x_i + \alpha \text{Lévy}(\beta) \), based on a solution \( x_i \) from nest \( n_i \). In our experiments, the coefficient \( \beta = 1.5 \), which is standard in literature. The step size \( \alpha = 0.5 \) was determined empirically (see results section). The quality of the \( n \) newly generated cuckoos in \( n \) separate threads is evaluated. Finally, we select a random nest \( j \) and replace its solution \( x_j \) with the new solution \( x_i^1 \), if it has lower energy (i.e., if \( f(x_i^1) < f(x_j) \)).

3.4.3. Generating solutions by mutation

Instead of replacing the fraction \( p_n \) of the nests at the end of each iteration as original cuckoo search strategy (ORIG), we explored mutating only a fraction of our parameters, called as fixed likelihood parameter replacement (FLPR), as shown in Fig. 4. More specifically, when mutating the solution \( x_i \), we first let the new candidate solution \( x_i^1 = x_i \) and then update it based on the parameter differences between two random nests \( x_i \) and \( x_k \). Formally, each parameter \( x_i^1[v] \) in \( x_i^1 \) is adjusted as follows:

\[
x_i^1[v] = x_i[v] + \alpha_m [x_i[v] - x_k[v]] H(pa - \epsilon)
\]  

(6)

\[
H(p) = \begin{cases} 
1, & p > 0 \\
0, & p \leq 0
\end{cases}
\]  

(7)

where \( v = 1 \ldots \text{dim}(x) \), the step size coefficient \( \alpha_m \) is set as 0.5 empirically, \( H(\cdot) \) is a Heaviside step function, and \( \epsilon \) is a random number drawn from a uniform distribution over \((0, 1)\). This approach tends to reduce the step size over time as the solutions converge.

We improve the results considerably by fine-tuning the values of \( p_n \) for different parameters \( x_i[v] \) (adaptive likelihood parameter replacement, or ALPR) (see Fig. 4 and Section 5). Empirically, we have found that the following values give best results: \( p_n = 0.75 \) if \( x_i[v] \) is a camera angle \( \theta \), \( p_n = 0.6 \) if \( x_i[v] \) is a camera distance \( d \) or image scale \( s \) and \( p_n = 0.5 \) otherwise.

After processing all parameters, we evaluate the quality of \( n \) newly generated cuckoos in \( n \) separate threads. We then replace \( x_i \) by \( x_i^1 \) if \( x_i^1 \) has lower energy, that is, if \( f(x_i^1) < f(x_i) \). Otherwise, we retain the incumbent solution.

4. Interactive space planning process

This section introduces our interactive inner space planning process for early stage concept design. Given an architectural design \( M \), the user may arrange its internal space via our interface. We provide four regular categories of space unit: cuboid, cone, cylinder and half ellipsoid. The user can add arbitrary instances of these primitives to form a set of desired internal units \( U_i \) (e.g., \( U_1, U_2, U_3, U_4 \) in Fig. 5).

For each unit, \( U_i \), at any point of the arrangement process, the user can specify its size through the interface. All the adjustable size parameters are shown in the left column of Fig. 5. Furthermore, the user may change the orientation and spatial location of each unit (horizontal planar movement and vertical movement (see element \( U_2 \))). By default, all units are located on the ground floor.

We provide a visualization of space arrangement as a 3D floor plan, by breaking up the 3D space into different levels. The user may change the height of ground floor and regular floors as well as the interval between each floor (see the right plan in Fig. 5).

At any stage of the planning process, given the fixed size and vertical location of all units, the user can run our algorithm to obtain a nearly optimal solution for the orientation and planar location of all units.

4.1. Objectives

In the design process, the user is allowed to add, delete or modify any arbitrary unit. We provide a few simple criteria to measure the quality of the arrangement on each unit in \( U_i \) as itemized below. Note that all categories of volumes are approximated and measured in terms of number of voxels.

- **Internal volume** \( V_{int,i} \) measures the volume of unit \( U_i \) located inside overall design \( M \).
- **External volume** \( V_{ext,i} \) measures the volume of unit \( U_i \) located outside overall design \( M \).
- **Collision volume** \( V_{coll,i} \) measures the volume of intersection of unit \( U_i \) with all other units (see the intersection of \( U_i \) and \( U_j \) in Fig. 5). Collision voxels are only counted once if more than two units are intersecting.
- **Space volume** \( V_{space,i} = V_{int,i} + V_{ext,i} \) measures the volume of unit \( U_i \).

Since traversing the voxel grid is time consuming, to obtain these measurements in real-time, we only visit the voxels that are located inside or on the bounding box of each unit (see the colored wireframe in Fig. 5).

While traversing each voxel within the bounding box of each unit \( U_i \), we use the implicit equation of the unit to determine whether the voxel lies inside the shape of \( U_i \). Then, depending on whether the voxel lies inside \( M \), we classify it as part of internal or external volume. During this traversal, voxels that lie within the implicit representation of multiple units are classified as part of the collision volume.
4.2. Space planning

Many functional spaces have requirements on floor location. Consequently, among all adjustable parameters, we do not modify the user-specified unit size and floor location, and optimize orientation and planar position of the units. We seek to optimally arrange all the units within the overall architecture shape $M$ with the given user preferences.

**Parameters.** The parameter vector $u$ consists of all parameters for each unit $u_i$. For each space unit $U_i$, we allow horizontal planar displacement $(x, y)$ and rotation $\theta_i$ around the Z-axis.

$$u_i = (x, y, \theta_i)$$  \hfill (8)

To assess the quality, or suitability of a solution which arranges all the $N$ units in $\{U_i\}$, we employ several measurements:

- **External ratio** measures the volume of internal spaces that are located outside design $M$ according to parameter vector $u$. $E_e$ is calculated relative to the total volumetric space, while $E_i$ is normalized by the volume of each individual unit:

$$E_e = \frac{\sum V_{e,i}}{\sum V_{s,i}}, \quad E_e = \frac{1}{N} \sum \frac{V_{e,i}}{V_{s,i}}$$  \hfill (9)

Note that the terms are zero if all units are completely inside the building. For unit sets with large difference on element volume, $E_e$ emphasizes more on the quality of large units and tends to neglect the performance of small ones. On the other hand, $E_e'$ gives all the units equal weight. With the same external ratio, the larger elements will have significantly larger external volume.

- **Collision ratio** measures the volume of intersection among all the units in $\{U_i\}$. Similarly, $E_c$ is calculated relative to the total volume space, while $E_c'$ is normalized by the volume of each individual unit:

$$E_c = \frac{\sum V_{c,i}}{\sum V_{s,i}}, \quad E_c' = \frac{1}{N} \sum \frac{V_{c,i}}{V_{s,i}}$$  \hfill (10)

Again, the terms are zero if there is no intersection between any pair of units. For the unit sets with large difference on element volume, $E_c$ emphasizes more on the quality of large units while $E_c'$ gives all the units equal weight.

Initially, we seek to minimize an energy function based on a weighted combination of external and collision ratio: $f(u) = w_{e}E_{e} + w_{c}E_{c}$. We employ our fine-tuned ALPR approach (Section 3.4.3) to find good approximations of the optimal solution. However, the complexity of the problem increases as the size and variety of the unit sets increases. The results tend to get trapped at local optima, sacrificing the integrity of smaller units.

Although such solutions have low energy value, solutions with good performance on each unit irrespective of their size may be preferred.

To overcome this problem, we explore another energy function based on a weighted combination of unit-wise external and collision integrity:

$$f(u)^{'} = w_{e}E_{e} + w_{c}E_{c}$$  \hfill (11)

By adjusting the weights, one can increase the importance of small units at the expense of the placement of larger units. Despite the fact that both options have some undesirable properties, they are jointly helpful in pruning unwanted solutions. Unlike exterior design where users may desire to set different weights and priorities for different terms, here we seek to find configurations that are local optima in both. Furthermore, in some cases, we need to trade off between eliminating external volume and reducing collision volume. Hence, we employ three objective functions to seek a Pareto optimal solution among these objectives.

4.3. Optimization

We employ the framework of our modified cuckoo search algorithm (as Algorithm 1) and further extend our ALPR (Section 3.4) into multi-objective form. We then employ our multi-objective ALPR to attain feasible solutions for the interior space planning problem.

**Parameters.** For a given set $\{U_i\}$ consisting of $N$ space units, our optimization has $\dim(u) = 3 \times N$ parameters.

**Objective function vector.** To achieve viable solutions, we seek to minimize a function vector: $f_{\text{space}}(u) = \{f_{1}(u), f_{2}(u), f_{3}(u)\}$, based on following objectives mentioned in Section 4:

$$f_{1}(u) = E_e, \quad f_{2}(u) = E_c, \quad f_{3}(u) = f(u)^{'}$$  \hfill (12)

where $E_e, E_c$ and $f_{3}(u)$ are the energy terms described in Section 4 and computed based on parameters $u$. In our results, we use the weights $w_{e} = 0.4$ and $w_{c} = 1.0$, which we found to be a good compromise between the objectives. The weights can be easily adjusted before or during the design session based on user needs.

**Pareto Optimum.** In multi-objective optimization, we want to find a variable vector $x^{*}$ to optimize the function vector $f(x^{*}) = (f_{1}(x^{*}), \ldots, f_{k}(x^{*}))$. Formulated by Pareto [41], $x^{*}$ is Pareto Optimal if $\forall i \in \{1, \ldots, k\}$: $f_{i}(x^{*}) \leq f_{i}(x)$ and $\exists i \in \{1, \ldots, k\}: f_{i}(x^{*}) < f_{i}(x)$. That is to say, $x^{*}$ is a Pareto optimal when there exist no $x$ that can decrease any energy item while not increasing any other in the function vector.

In Algorithm 1, we then can define the dominance relationship between function vectors $f$ and $f^{'}$ as $f \prec f^{'}$, if and only if $\forall i \in \{1, \ldots, k\}$: $f_{i} \leq f_{i}^{'}$ and $\exists i \in \{1, \ldots, k\}: f_{i} < f_{i}^{'}$. 

---

Fig. 5. The architectural interior design process of our system with the ACM example, including the input, result of optimized space arrangement, and 3D floor plan. The plans include four space elements: foyer, department area, and multi-floor atrium (two red cuboids, a green cylinder, and a purple cone). (Please refer to the web version of this article for the interpretation of the color legend.)
The Pareto optimum always consists of a set of solutions called non-dominated solutions or Pareto front [42]. During the process of decreasing the external volume measured by $f_1$ and $f_2$, we use $f_3$ to trade off between generation of internal and collision volume. A Pareto optimal solution would indicate that the smallest volume of external and collision voxels has been achieved.

**Candidate pruning.** To improve the efficiency of the searching process by generating qualified solutions, during egg initialization and generation of solutions via Levy flights, we regenerate parameters $\mathbf{u}$ until all the center of units $\{U_i\}$ are located inside the current building model $\mathbf{M}$. Referring to Fig. 5, with the current parameter $\mathbf{u}$, the center of $U_2$ is located inside $\mathbf{M}$ while the others are outside. Hence we regenerate $\mathbf{u}$ until it results in a plan with all the centers located inside $\mathbf{M}$. Experimentally, with fine-tuned parameters, we found this pruning strategy increased the quality of solution without increasing the running time used to converge to solutions.

A considerable number of candidates are needed in this complex form. We achieved best results letting the number of nests $n = 50$. We fine-tune the values of $p_a$ for different parameters in our multi-objective ALPR (see Section 5). Empirically, we have found that the following values give best results: for the $k$th variable where $k = 1 \ldots \dim(\mathbf{u})$, $p_s = 0.5$ if $u[k]$ is planar position $(x, y)$ and $p_s = 0.625$ if $u[k]$ is rotation angle $\theta_z$. Please refer to the supplemental material for the pseudo-code of our algorithm.

**Exterior shape modification.** In some cases, if a large-scale space is required, the corresponding units might not be able to be perfectly placed inside the current building design $\mathbf{M}$ in any case. To examine the possibility of making all units entirely inclusive while maintaining the overall shape and scale of the building, we employ our ALPR to minimize $f_3$ by perturbing parameters $\mathbf{x}$ (mentioned in Section 3.3) in a user given range of the parameter domain. The range is set to 10\% by default. Two perturbed result are shown in Figs. 6(a) and 9(b).

**Global design process.** For some projects, the requirements on interior functional rooms serve as constraints to the exterior shape design process at the beginning of the design pipeline. We explored the possibility on integrating the problems of exterior shape design with inner space planning. Since a trade-off between exterior shape and interior plan are expected, we provide the option on minimizing a global objective function: $f_{\text{global}}(\mathbf{x}, \mathbf{u}) = w_{\text{shape}}f_{\text{shape}}(\mathbf{x}) + w_{\text{space}}f_{\text{space}}(\mathbf{u})$, with our ALPR. The value on weights $w_{\text{shape}}$ and $w_{\text{space}}$ are subjective. Two results are shown in Figs. 6(b) and 9(c). In our results, we found $w_{\text{shape}} = 0.1$ and $w_{\text{space}} = 0.9$ provide a good compromise.

## 5. Results

For the exterior shape design process, we first explored the parameter space of the coefficients $\alpha_l$ and $\alpha_m$ and the number of iterations required to converge to a good solution. We estimated these by an empirical study over a set of examples, with an expectation that the convergence speed and result do not change significantly for small variations. We achieved the best performance with higher parameter values ($\geq 0.5$) (see the first chart in Fig. 7). For the results in this paper, we used $\alpha_l = \alpha_m = 0.5$. We also explored the choice on period $p$; based on the empirical results, we recommend $p = 1, 2, 3$ (see the supplemental materials). The selection on period is open to the user; we use $p = 1$.

The other important parameter is the termination condition. The objective function is discontinuous, non-linear, and each iteration is expensive since it requires solving the deformation model multiple times. A series of trial runs across the range of our examples all converged after approximately 60 iterations. This is shown in Fig. 8(a).
Fig. 8. Convergence results of the different optimization strategies after 60 iterations (a), of interior optimizations (b), after perturbation optimizations (c), and of the global optimizations integrating exterior and interior design process (d). The convergence statistics are shown in Tables 1–2.

Table 1
Statistics of our example designs. All times are in minutes.

<table>
<thead>
<tr>
<th>Building</th>
<th>Voxel grid Resolution</th>
<th>Mesh(#f)</th>
<th>#Iters</th>
<th>Time</th>
<th>Time(2)</th>
<th>Time(3)</th>
<th>#Iters</th>
<th>Units</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bird</td>
<td>154 × 154 × 22</td>
<td>11736</td>
<td>60</td>
<td>12.7</td>
<td>6.5</td>
<td>4.3</td>
<td>30</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>ACM</td>
<td>123 × 123 × 50</td>
<td>12396</td>
<td>60</td>
<td>33.5</td>
<td>16.7</td>
<td>11.9</td>
<td>60</td>
<td>4</td>
<td>10.4</td>
</tr>
<tr>
<td>Egypt</td>
<td>123 × 123 × 45</td>
<td>13320</td>
<td>60</td>
<td>47.8</td>
<td>24.6</td>
<td>15.5</td>
<td>45</td>
<td>3</td>
<td>12.8</td>
</tr>
<tr>
<td>Aquarium</td>
<td>174 × 174 × 37</td>
<td>12568</td>
<td>60</td>
<td>64.2</td>
<td>31.3</td>
<td>20.1</td>
<td>45</td>
<td>3</td>
<td>17.9</td>
</tr>
</tbody>
</table>

*Timings when performing structural integrity once every 2 or 3 iterations. Result figures available in supplemental material.

For the internal space planning process, we explored the parameter space of the coefficients $\alpha_l$ and $\alpha_m$. By varying $\alpha_l$ and $\alpha_m = 0.1, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1$, we found that the parameters $\alpha_l = 0.375$ to 0.625 and $\alpha_m = 0.375$ to 0.625 work best (see Fig. 7). With $\alpha_l = 1$, the algorithm sometimes failed at the solution-pruning stage in Section 4.3.

Additionally, we estimate the convergence speed and results on several examples. Each unit takes around 15 iterations to converge. Thus we dynamically determine the number of iterations according to the number of space instances (see Fig. 8(b)).

Next, we estimate the termination condition on exterior shape perturbation. The convergence on the optimizations could be observed in Fig. 8(c) within 10 iterations. Moreover, we integrate the exterior design and interior planning processes into a global optimization. Since the exterior problem requires more time to converge under the constraint of internal elements, we run the global optimization with at least 60 iterations. The convergence was observed within $60 + 15 \times N$ iterations (see Fig. 8(d)). The running times were measured on a workstation with a 10-core Intel i9 CPU.

5.1. Verifying the modified Cuckoo search

Next, we measure the performance of our modified cuckoo algorithm. Six sets of experiments were conducted. Initially, all examples were run by using ORIG in Fig. 8(a). Next, we ran a modified cuckoo search (FLPR). Then, we ran a different variation of the search (ALPR). Finally, we also compared with strategies using simulated annealing (SA), genetic algorithm (GA), and particle swarm optimization (PSO). We explored the parameter space of these other strategies (with the same amount of potential solutions) so as to best minimize our energy function. More precisely, we performed multiple runs for each algorithm and report the best result. Refer to Fig. 8(a) for the convergence results and supplemental material for the performance on the average and standard deviation of the final energy.

Notice that the original cuckoo search could get trapped in a local minimum for a large number of iterations, making it less viable in our application due to the relatively high cost per iteration. In all experiments, our ALPR approach outperformed the other techniques.

5.2. Example of architectural designs

Table 1 shows the statistics of using our system to design four examples. Figs. 1(c), 1(d), 2(b), 5 and 9(a), illustrate designs and rendered results. For the example of Egypt and Aquarium design, the first row shows three views of the optimized model with their corresponding binary images shown as insets. The bar chart on the top right shows the statistics of key parameters before (in red) and after (in green) optimization. Bigger values are expected for the statistics in upper part with a green right arrow, while smaller values are expected for the lower part with a green left arrow. Note the significant improvement in these statistics after performing the optimization. The second row shows the voxel grid of an initial design, a visualization of its structural and topological integrity, and an optimized design with its respective visualization.

Fig. 10 shows the examples generated by the Shadow Art algorithm [12]. Since Shadow Art targets a different application, it does not directly consider architectural objectives such as structural integrity, floaters, ground area, a flat roof structure, and overall shape smoothness. Thus, results are not directly applicable to our domain.

5.3. Examples of space planning

The internal space plans on examples generated by our shape inspired architectural design are shown in Figs. 1(c), 5, 6, 9 and
Fig. 9. (a) Design of an Aquarium based on three binary images shown as insets with the corresponding rendered views in the first row. The second row shows the voxel grid of an initial design, a visualization of its structural and topological integrity, and an optimized design along with its visualization. (b) The result of the optimized arrangement on space elements (top) and perturbed exterior design with optimized arrangement (bottom) of the Aquarium. (c) The exterior design and interior plan of Aquarium design obtained by integrating them into a global problem. The first two columns of each example show the initial plan (top left), a visualization of its floor plan (bottom left), and the optimized respective plans (right). The third column shows the voxel grid of the exterior design and a visualization of its structural and topological integrity. The values of the parameters controlling the exterior shape before (red) and after (green) optimization are shown in the bar chart. (Please refer to the web version of this article for the interpretation of the color legend.).

Fig. 10. Results of Shadow Art applied to our architectural problem based on desired templates shown on the left column. The inset figures depict the deformed target shape in black and the error relative to the desired templates in orange. (Please refer to the web version of this article for the interpretation of the color legend.).

Fig. 11. The space plans include an airport lounge and a strip of functional area (two red cuboids). The optimized plan emphasizes on shape integrity of elements (left), and exclusion between units (right).

11. Referring to the first row of Figs. 6(a) and 9(b), a portion of the units cannot be accommodated entirely inside the overall shape of the conceptual design. Thus, we further modify the design by perturbing the parameters in the shape design algorithm within a 10% range in the value space (see results to the second row). Note the growth on integrity of templates. Trade-offs are made to improve integrity of space elements. The processing time and number of iterations for these examples are shown in Table 1.

By varying the weights on external ratio \( w_e \) and collision ratio \( w_c \), the user can trade off between inclusiveness of units under overall shape or exclusion between units. Optimized results on bird shape design with: \( w_c = 0.1, w_e = 5 \) and \( w_c = 5, w_e = 0.1 \) are shown in Fig. 11. With preference on units integrity, the left result allows intersection between these two elements best shape integrity. While with preference on exclusion, the bigger lounge was partly pushed out from the overall design in the right result. Trade-offs are highlighted by dashed rectangles. Statistics are shown in Table 1.

Furthermore, we investigate the global optimization problem of integrating exterior shape design with inner space planning. We ran our global optimization on the Egypt design and Aquarium design with the same set of units shown in Figs. 6(b) and 9(c). Note that new configurations on camera positions are found to maximum accommodate space units inside overall shape. The
statistics about making modification on exterior shape as a post-processing stage and integrating exterior and interior design as a global problem are shown in Table 2.

6. User feedback

Since there is no prior work on creating buildings from non-organic shapes to the best our knowledge, we performed a user study to examine the usability of our interactive system and the effectiveness of our optimization algorithms on optimizing the user-created designs. The one-on-one user studies were performed on a computer running our prototype software.

Template library and participants. We recruited 22 participants for this user study: 16 male and 6 female participants with age ranging from 20 to 40. Four participants were professional architects, four were engineers from an architectural company, and the remaining participants were postgraduate students or university staff. We created a library of 99 shape templates. Participants could choose three arbitrary shape templates as input or provide their own.

Tasks and procedure. The study began with a brief 15-minute tutorial about the background and basic operations of our interface. Participants were free to ask questions to better understand the system at this stage. Then, the participants were asked to create their designs. After tuning all the parameters through our interface, the participants were free to arrange the internal space of their design. Finally, the participants were encouraged to give advice and express their opinion on any part of our system, and were asked to give scores on satisfaction and applicability of using our interface. All scores range from 1 (lowest) to 5 (highest).

6.1. Results and discussion

Creating 3D designs requires intensive spatial imagination. In the process of generating designs from input templates, our interface greatly reduces the requirement on spatial imagination in achieving satisfying designs. Without making efforts to imagine the resultant voxel design, users can make decisions and improvements on selection of templates according to the immediately updated model.

Moreover, when enhancing the preliminary model, it is often difficult to imagine the resulting model after exploring different parameter combinations on cameras. Most participants prefer to only fine-tune the parameters on image templates. However, the improvement on designs in both physical and aesthetic aspects could be more significant with suitable configuration of cameras.

We ran our optimization based on designs created by participants and obtained several interesting feasible designs. The participants referred to their designs as: skyscraper, hillside studio and theme museum. We show comparisons of three user designs and optimized solutions in Fig. 12 along with statistics of both designs. More results are shown in the supplemental material. Note the improvement on template integrity of each design, as well as improvement on structural integrity of examples on some of the results. Furthermore, even a tiny perturbation on rotation angle can reduce the volume of a structurally problematic area (see the supplemental material). Hence, our optimization algorithms reduced by a large amount the information participants need to consider in the design process.

User Interface. All of our participants from different backgrounds confirmed that the GUI is easy to learn and use. Most of the users took about five to ten minutes to get familiar with functions provided in our interface. The learning speed is approximately ten minutes on average (refer to the top left chart in Fig. 13).

After experiencing the design system, all participants expressed interest in generating their own design through our interface and gave the score of 4.3 on average (refer to the top right chart in Fig. 13). Also, suggestions on differentiating the color code of structural integrity and internal units, employing more straightforward graphical icons, and enhancing rendering effects are given by users to improve the visual effect of our prototype system.

Applicability. For the applicability of our exterior and interior design processes, the users gave scores of 4.3 and 3.9 on average respectively (refer to the two charts at the bottom in Fig. 13). By analyzing the ratings, we find that experts gave higher scores on interior design while participants with other backgrounds gave higher score on exterior design.

Further assessment and guidance by employing AI techniques on how to choose consistent templates and how to effectively fine-tune the parameters to enhance design quality are suggested as future work. A simple guidance on how to choose templates as a good starting point are suggested in the supplemental material. Some combinations of templates are not able to generate a plausible architectural design or preserve the shape of the templates even after optimization (see the model on the left in Fig. 14). Furthermore, sometimes the exterior design cannot provide sufficient interior space to accommodate the user specified units (see the right plan in Fig. 14). The size of internal units need to be changed in this case. Moreover, participants made several suggestions for additional functions, such as scaling, allowing modifications on the 2D templates in the interface, and interactive modification on 3D voxel models (add/reduce small fraction of volume), among others, to address these issues as future work.

Expert feedback. Interviews were conducted to collect the feedback from four architects, two from industry and two from academia.

The industry architects find the process of creating designs through our interface enjoyable. They commented that the exterior design is an attractive process and the generated model could trigger inspiration or serve as preliminary design for their future work. On the other hand, they appreciate the choice to use 3D space entities in space planning, while having the visualization of internal arrangement both in the form of floor plans and 3D plan at the same time. From their experience, most architectural designs are developed based on requirements of several internal spaces. Thus, integrating the process of exterior design with interior design into a single problem is more intuitive.

Moreover, the architects expected to explore our interface as an add-on in professional design software. As a next step, they recommended to encode more information into our optimization formulation, such as conditions on wind load and sunlight for exterior design, and grammatical rules or correlation between space elements for interior design. Moreover, providing simple 3D shape units (e.g. cube) as foundation for the exterior design process is useful as well.

The architects from academia appreciated our attempt in transforming the design processes into a computational problem. Furthermore, they commented that our interface would be practical for massing phase (early design phase). Generally, architecture

<table>
<thead>
<tr>
<th>Building</th>
<th>Shape modification</th>
<th>Global problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Iters</td>
<td>Time</td>
</tr>
<tr>
<td>Egypt</td>
<td>10</td>
<td>0.8</td>
</tr>
<tr>
<td>Aquarium</td>
<td>10</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Table 2: Processing time of shape modification and combining two design steps as a global problem. All times are in minutes.

W. Xiong, P. Zhang, P.V. Sander et al. Computer-Aided Design 142 (2022) 103120
is the combination of firmness, commodity and delight. With real-time rendering on design, visualization on structural integrity and design statistics, our platform indeed provides alternatives for designers to interact with clients and users in the early stage.

To be more specific, as architectures are symbols in the space, our exterior design methodology would be useful in such application that architecture itself holds the function to convey explicit information to observer (e.g. landmark buildings). Such design language has been analyzed by Venturi et al. [43] about the ubiquitous symbolism of geometrical and architectural forms in Las Vegas. Overall, the architects felt that the design experience was easy to understand and exciting to explore.

The second example in Fig. 12, referred to as studio, was designed by an in field architect. The third example in Fig. 12, referred to as theme museum, was designed by an architect from academia. During the design process, while novices focused mostly on the choice of silhouettes and category of rooms, architects spent time on analyzing the daylighting, exterior details, and location of interior rooms. Furthermore, they considered how the building would look like with people walking through the space.
Thus the floor plan of their design could further assist them on this process.

7. Conclusion and future work

In this paper, we propose a new framework, called ShapeArchit, for initial design of architectural buildings from images. Such a tool is of interest to architects who use, for example, organic forms as an inspiration. The 3D shape is derived from the visual hull of the images. The parameters controlling the size and location of the images control the shape of the building. An evaluation of the building is made as a weighted sum over several factors, including topological factors, structural stability, input conformance, and other functional requirements. Using this objective function, we run a parallel modified cuckoo search to find low energy optimized initial design. As a following step, with a preliminary design, we introduce a technique for initial arrangement on architectural inner space. Our tool is used to provide possible plans with given three-dimensional space elements. We evaluate plans with given parameters of size and position on shape items. To find low-energy solutions minimizing our objectives simultaneously, we run a modified multi-objectives cuckoo search algorithm. Several examples were created to indicate that our approach can produce viable shapes for projects including buildings, theme park structures and even artificial landscaping forms.

User studies were conducted to estimate the effectiveness of our method. The participants included practicing architects, architecture professionals, as well as postgraduate students and university staff with diverse backgrounds. The studies confirm that our GUI is easy to learn and use. Furthermore, in most cases, the optimized version of the design was found to improve upon the input user-specified design. In future work, we will focus on improving the accuracy, efficiency, and usability of our interface. One possible way is to employ a more direct structural analysis approach to replace our current stress simulation. Another way is to use a mesh-model instead of voxel model, since it is capable of preserving more details on given image; however, this approach would imply the development of an alternate mechanics model. Finally, employing multi-objective optimization algorithms to solve our exterior design problem is also worth exploring.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the Departments of CSE, IEDA, and ECE of HKUST. The authors also would like to thank our reviewers for their helpful comments and all the participants in our user study. Part of this research was funded by HK RGC GRF, Hong Kong Special Administrative Region grant #16213519.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2021.103120. The supplementary material contains a video introducing our interactive interface, and a document with additional data on optimization algorithms, discussion on templates selection and additional results from user study.

References