Shape-Inspired Architectural Design

Weidan Xiong
Hong Kong UST
wxiongab@connect.ust.hk

Pengbo Zhang
Hong Kong UST
pzhangag@connect.ust.hk

Pedro V. Sander
Hong Kong UST
psander@cse.ust.hk

Ajay Joneja
Hong Kong UST
joneja@ust.hk

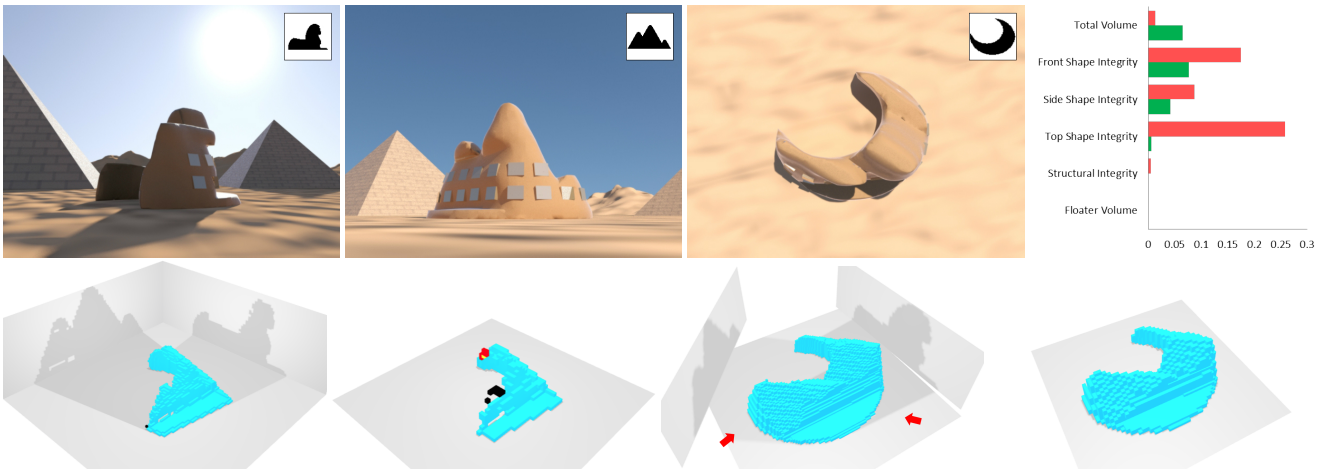


Figure 1: Design of a museum based on three binary images shown as insets with the corresponding rendered views in the top row. The second row shows the voxel grid of an initial design, a visualization of its structural and topological integrity, and an optimized design and its respective visualization. The values of the parameters controlling the shape before (red) and after (green) optimization are shown in the bar chart on top right.

ABSTRACT

We introduce a method to design architectural buildings that are inspired by shapes of non-architectural forms. The user inputs a few binary images, each providing an indicative shape for the building from a different viewpoint. A discrete visual hull corresponding to each binary image is generated. A voxel model is then constructed by intersecting the hulls corresponding to the images. The shape of the voxel model depends on the parameters of the projections. Real buildings must also obey some topological and structural constraints. We develop a shape metric to evaluate a given design in terms of topological, functional and structural requirements of the building. This allows us to optimize the building shape as a function of the parameters of the projection. The optimization problem is solved by means of an improved cuckoo search metaheuristic. The resulting voxel model is converted into a mesh. Finally, we apply a novel smoothing algorithm that produces a smooth surface while preserving sharp creases and roof structures. Several examples are presented in the paper to illustrate the methodology and results.

I3D '18, May 4–6, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5705-0/18/05…$15.00
https://doi.org/10.1145/3190834.3198034

ACM Reference Format:
Weidan Xiong, Pengbo Zhang, Pedro V. Sander, and Ajay Joneja. 2018. Shape-Inspired Architectural Design. In *I3D '18: I3D '18: Symposium on Interactive 3D Graphics and Games, May 4–6, 2018, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3190834.3198034

1 INTRODUCTION

Several modern architects use non-architectural shapes as an inspiration for the design of landmark buildings (see figure 2). Notable examples in recent years include the CCW building in Sydney (architect: Frank Gehry), the wood art museum in Harbin (architect: Ma Yansong) and the Birds nest stadium in Beijing (architects: Jacques Herzog and Pierre de Meuron). 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 the input. Our approach can be extended to use an arbitrary set of two or more such images.

Our approach is guided foremost by the idea that the generated model should faithfully mimic each input image from some viewpoint. One of the inputs is typically (although not necessarily) used as the plan view, and represents the footprint of the 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 also several other constraints that guide the design of buildings. These may be categorized as topological, physical or functional constraints. Some examples of such

©Peellden 2005 ©JesseW900 2014
(a) Taipei 101

©Joi Ito 2007 ©Andre Lage Freitas 2012
(b) Burj Al Arab

©Rmashhadi 2011 ©Cassandra Tiensivu 2006
(c) Olympic Pavilion

©Azerbaijan Republic 2012 ©Korrigan 2004
(d) Heydar Aliyev Center

Figure 2: Building designs inspired by non-architectural shapes (shown to the right or below).

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 a search process to generate low energy optimized designs. Again, our approach is extensible in the sense that additional constraints can be encoded to enrich the objective function, and existing constraint functions or their domains can be changed easily.

Contributions. We introduce a new approach to assist architects to quickly generate new designs inspired by images of non-architectural forms. We develop a model to find low energy optimized locations that yield faithful silhouettes of the images. The model allows encoding of physical as well as functional constraints. In achieving this main objective, we introduce:

- A modified cuckoo search algorithm that more efficiently solves our shape design problem. We hope that these modifications will find practical uses in other domains.
- A novel iterative mesh smoothing algorithm that yields a smooth surface while preserving sharp creases and roof structures. We show how this new technique generalizes well and is competitive with generic mesh smoothing techniques.

1.1 Relation to previous work

Automatic generation of architectural forms from image input has been explored in a few previous works. One approach is through the use of procedural modeling [Smelik et al. 2012]. Users can also construct a valid description of the building based on a set of grammatical rules with block units such as windows, walls, roof structures [Schwarz and Müller 2015]. The model geometry may be inferred via images of the building with machine learning techniques [Fan and Wonka 2016].

Structural integrity of a design is an important criterion in architecture. But it is difficult to explore designs 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. [2014], (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 [Block and Lachauer 2011]. Efficient and robust techniques for statics-driven interactive mesh design have also been developed [Jiang et al. 2014; Tang et al. 2014]. An alternate approach is to integrate structural stability into a procedural modeling framework [Whiting et al. 2009]. Another approach is employing efficient physics engines. Our implementation uses a voxel-based solver that uses a simplified finite difference model for dynamic simulation of soft objects [Hiller and Lipson 2012]. This model uses several simplifying assumptions for computing dynamic distortions of a model. In our case, the only external forces on the model are gravity and the ground reaction (we neglect effects of wind loads), and the dynamic model is run long enough until each voxel has nearly zero velocity and acceleration.

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 2D sketches [Grimstead and Martin 1995] or engineering drawings [Dutta and Srinivas 1992]. This has been extended even to models of assembled parts (each with its assigned sketch-lines in each view) [Rivers et al. 2010]. Our approach borrows more from the visual hull introduced by Laurentini [1994]. 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. [2016]. 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 [Mitra and Pauly 2009]. However, none of these approach consider the structural integrity of generated models, nor other architectural needs, such as the roof flattening addressed by our smoothing algorithm which was tailored to this problem.

Our optimization 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. The optimization

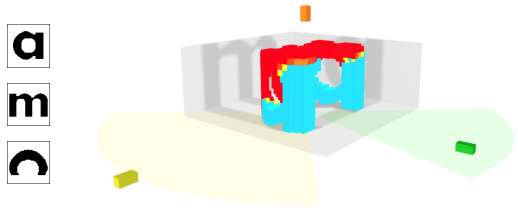


Figure 3: The user interface of our system with the camera and images of the ACM example in their initial default positions.

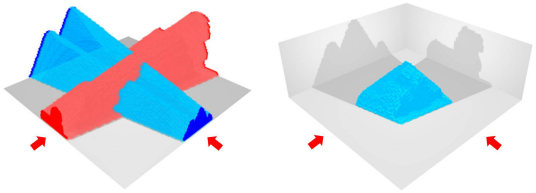


Figure 4: The perspective projections of the Sphinx and pyramids (left) are used to construct the visual hull (right).

model we face is non-linear, non-convex and discontinuous. We developed a search metaheuristic based on a modification of cuckoo search [Yang and Deb 2009] because of the effectiveness of this technique in solving problems of a similar nature. Finally, we employ a novel smoothing algorithm to create a plausible design.

2 INTERACTIVE DESIGN PROCESS

This section introduces our interactive architecture design system. We start with the three shape templates $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3$ (e.g., "A", "C", and "M" in figure 3). 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) using the current configuration as the input.

2.1 Discrete visual hull

By projecting the three shape templates from their respective camera positions, a visual hull [Laurentini 1994] can be constructed from the intersection of their silhouette cones (see figure 4). There are many algorithms for computing visual hulls in real time for different applications. Some are based on geometric computations or voxel representations [Loop et al. 2013], while others are image-based [Matusik et al. 2000]. For simplicity and efficiency, we use a voxel representation for the visual hull.

Optimizations. As we adjust and fine-tune the structure, we must update the voxel grid in real-time. Therefore, to improve efficiency during active user interaction, 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. When the user stops manipulating the parameters, the computation is updated using the full resolution grid. Table 1 lists all the resolutions used in our examples.

2.2 Objectives

When designing a building, there are several aesthetic, structural, and practical considerations. These are described below, with a description of how we measure its quality. 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 those of the shape templates. For each shape template j, we compute a new binary image $\mathbf{I}_j^*$ representing projection of the current voxel grid onto the image plane of camera j. We accomplish this by tracing a ray for each image pixel and setting it to 1 if it intersects an active voxel. We define the error as the Frobenius-norm between the desired shape template $\mathbf{I}_j$ and the current shape $\mathbf{I}_j^*$ integrated over all three images:

$$E_i = \sum_{j=1}^{3} \frac{||\mathbf{I}_j^* - \mathbf{I}_j||_F^2}{w_j}, \tag{1}$$

  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. There are several methods for structural analysis that could be employed in this setting. For efficiency and simplicity, we use a simple physical simulation. We assign each voxel a fixed mass, connect it to its six direct neighbors, constrain the position of the ground voxels, and simulate the system under the force of gravity until convergence. Areas of the building that undergo significant displacement indicate structural problems. For simplicity, the magnitude of a voxel's displacement is used as a proxy for structural instability within its neighborhood. In mechanics, the level of stress is not necessarily related to the absolute displacement. E.g., the maximum principal stress is felt at the fixed end of a cantilever under bending moment. However, the voxels at the free end (which experience the largest displacement) contribute the most to the stress at the fixed end, and thus serve as an indicator of a stress point in the neighborhood. Voxel displacements larger than ϵ_m voxel units denote moderate stress contributors (color coded in yellow in figure 5), whereas those larger than ϵ_s denote severe stress contributors (color coded in red). We measure structural integrity as the fraction of active voxels that contribute to severe stress:

$$E_s = g_{red}/g_a \tag{2}$$

  where g_{red} is the number of "red" voxels and g_a is the number of active voxels (i.e., a one voxel thick surface layer of the downsampled visual hull solid). In our results, we use weight $w = 50\text{kg}$, $\epsilon_s = 0.62$, $\epsilon_m = 0.5$. These parameters can be adjusted should the designer want to be more conservative.

In addition to the above criteria, we also consider the following practical considerations:

- *Total volume* $V_t = g_a/D^3$ measures the overall size of the building. It is the number of active voxels g_a normalized by

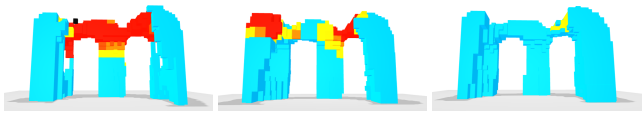


Figure 5: Different shapes generated by varying the model parameters, with voxel colors indicating the structural and topological measures. The stress intensity is denoted from yellow (light) to red (heavy).

total cubic dimensions of the voxel space D^3 in number of voxels.

- *Floater volume* $V_f = g_f/g_w$ measures the volume of floater objects, where g_f is the total number of voxels in the building components that are disconnected from the ground, and g_w is the total number of external voxels. Such disconnected regions are undesirable as they would 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 the first model on left in figure 5).

2.3 Parameters

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

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

where t_j, s_j, θ_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 overhead and does not include a y-axis rotation parameter.

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:

$$\mathbf{x_p} = \mathbf{x_o}^\frown(d_1, d_2, d_3) \tag{4}$$

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

At any stage, the user may render the model to get a better visualization of the shape. To implement this, we use a standard marching cubes library to create a faceted approximation of the model to which we can apply a user-selected texture.

3 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.

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

Objective function. We seek to minimize a function based on the objectives of section 2.2. The relative importance of these objectives is subjective. For the results in the paper, we considered a weighted combination of template integrity, structural integrity, total volume, and floater volume. Our energy function is

$$f(\mathbf{x}) = w_i E_i + w_s E_s + w_t(1 - V_t) + w_f V_f \tag{5}$$

ALGORITHM 1: Modified Cuckoo Search

```
Generate initial population of n host nests (section 3.2)
while termination condition not met do
    for each nest n_i do
        Get a cuckoo egg x' by Lévy flights from x_i (section 3.3)
        Choose random nest n_j to lay the egg
        if f(x') < f(x_j) then
            replace x_j by x'
        end
    end
    for each nest n_i do
        Get a cuckoo egg x' by mutation from x_i (section 3.4)
        if f(x') < f(x_i) then
            replace x_i by x'
        end
    end
end
return x_best
```

where E_i, E_s, V_t and V_f are the energy terms described above and computed based on parameters $\mathbf{x}$. In our results, we use the weights $w_i = 0.1$, $w_s = 5$, $w_t = 0.75$, and $w_f = 5$, 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.

3.1 Modified cuckoo search

Since the energy function $f(\mathbf{x})$ 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 simple cuckoo search 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 then end of each iteration, the algorithm stores the current best solution and drops a fraction p_a of the nests, replacing them by new random solutions. In our new *modified* cuckoo search, instead of randomizing *all* of the parameters in a candidate solution, we only modify selectively a fraction of the parameters. This modification, which is described in section 3.4, significantly improves the results.

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 figure 9. The following sections describe how we generate new candidate solutions in the different stages of our algorithm.

3.2 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 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 (θ_1, θ_2) in six of the solutions to (0, $\pi * 3/4$), ($\pi * 3/4$, 0), (0, π), (π, 0), (0, $\pi/4$), ($\pi/4$, 0), 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).

3.3 Generating solutions via Lévy flights

We follow the original cuckoo search strategy of Yang and Deb [2009] which uses Lévy flights when generating a cuckoo $\mathbf{x}'$ based on a solution $\mathbf{x}_i$ from nest n_i.

$$\mathbf{x}' = \mathbf{x}_i + \alpha_l \mathbf{Lvy}(\beta) \tag{6}$$

The Lévy flight is equivalent to a random walk, but with a step size that is based on a Lévy distribution. In our experiments, the coefficient $\beta = 1.5$, which is standard in literature. The step size α_l = 0.5 was determined empirically (see results section). Finally, select a random nest j and replace its solution $\mathbf{x}_j$ with the new solution $\mathbf{x}'$, if it has lower energy (i.e., if $f(\mathbf{x}') < f(\mathbf{x}_j)$).

3.4 Generating solutions by mutation

Instead of replacing the fraction p_a of the n nests at the end of each iteration as in the original cuckoo search strategy (ORIG), we explored mutating only a fraction of our parameters. We refer to this modification as *fixed likelihood parameter replacement* (FLPR). More specifically, when mutating the solution $\mathbf{x}_i$, we first let the new candidate solution $\mathbf{x}' = \mathbf{x}_i$ and then update it based on the parameter differences between two random nests $\mathbf{x}_j$ and $\mathbf{x}_k$. Formally, each parameter v in $\mathbf{x}'$, with probability p_a, is adjusted as follows:

$$\mathbf{x}'[v] := \mathbf{x}'[v] + \alpha_m * (\mathbf{x}_j[v] - \mathbf{x}_k[v]) * \mathbf{rand}(0, 1) \tag{7}$$

where the step size coefficient α_m is set as 0.5 empirically, and $\mathbf{rand}(0, 1)$ denotes a random number drawn from the 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_a for different parameters. This variant is refered as *adaptive likelihood parameter replacement*, or ALPR (see section 5). Empirically, we found that the following values give best results:

$$p_a = \begin{cases} 0.75 & \text{if } v \text{ is a camera angle } \theta_j \\ 0.6 & \text{if } v \text{ is a camera distance } d_j \text{ or image scale } s_j \\ 0.5 & otherwise \end{cases} \tag{8}$$

After processing all parameters, we then replace $\mathbf{x}_i$ by $\mathbf{x}'$ if it has lower energy, that is, if $f(\mathbf{x}') < f(\mathbf{x}_i)$. Otherwise, we retain the old solution.

4 MESH SMOOTHING

We use the marching cubes algorithm [Lorensen and Cline 1987] to extract a surface mesh from our final low energy solution. The resulting mesh often exhibits significant noise and features that are not suitable for architectural shapes. We considered several approaches for post-processing the mesh. We explored the rolling guidance filter (RGF) [Wang et al. 2015], however due to the nature of our voxelized structure it cannot produce satisfactory results. We also considered the joint bilateral filter [Zhang et al. 2015], which uses the normal field to iteratively guide the smoothing process. However, the output cannot satisfy the architectural requirements.

ALGORITHM 2: Mesh smoothing

Input mesh M_0, number of iterations k
for *s=1 to k* **do**
  Compute face normals $\{\mathbf{n}_i\}$ of mesh M_{s-1}
  Compute $\{\mathbf{g}_i\}$ by applying DMF to $\{\mathbf{n}_i\}$ (eq. 10)
  Compute $\{\bar{\mathbf{g}}_i\}$ by roof flattening on $\{\mathbf{g}_i\}$ (eq. 14)
  Compute $\{\bar{\mathbf{n}}_i\}$ by applying bilateral filter to $\{\bar{\mathbf{g}}_i\}$ (eq. 15)
  Generate M_s by updating vertices of M_{s-1} based on $\{\bar{\mathbf{n}}_i\}$ (eq. 18)
end
Output mesh M_k

Inspired by architectural designs, we propose an iterative smoothing method that yields a mostly smooth surface while preserving sharp creases. Common architectural designs often exhibit mostly smooth surfaces with a few shape-defining sharp creases (figure 2d). Oftentimes, nearly horizontal regions at the top are flattened to serve as a balcony or roof. Our approach can consider adjusting the surface at the top of the structure to allow for a flat roof, which may be desirable for most buildings.

In each iteration, we first apply the deformable mean filter to generate a smooth normal field $\{\mathbf{g}_i\}$ (section 4.1). We then adjust normals of nearly horizontal surface regions which are candidate roof structures and apply a bilateral filter to further smooth small-scale features, generating $\{\bar{\mathbf{n}}_i\}$ which is used to update the mesh (section 4.2). Algorithm 2 outlines this process, where the number of iterations is determined by the user as described in section5.2. Next, we describe each of the stages.

4.1 Deformable mean filter

The traditional fixed-shape filter traverses a 1-ring neighborhood of facet f_i (see figure 6) and takes the average normal $\{\mathbf{n}_j\}$ weighted by triangle area:

$$\mathbf{g}_i = \frac{\sum_{f_j \in N_i} A_j \mathbf{n}_j}{||\sum_{f_j \in N_i} A_j \mathbf{n}_j||} \tag{9}$$

where N_i is the set of faces that share a common edge or vertex with f_i, and A_j, $\mathbf{n}_j$ are the area and normal of face f_j.

Results are improved by considering a wider neighborhood beyond the 1-ring of f_i. Zhang et al. [2015] consider all 1-ring neighborhoods of all f_j adjacent to f_i, and set g_i to be the average normal of the f_j neighborhood with least normal variation. We propose an efficient approximation that does not need to consider the entire 2-ring neighborhood of f_i. We adapt the notion of deformable kernel proposed in [Dai et al. 2017] to area weighted mean filtering. While traversing the neighbor f_j of f_i, instead of computing its 1-ring average normal, we simply interpolate its normal with the normal of just one randomly selected neighbor of f_j, as shown in figure 6. Note that a different random neighbor is drawn for each f_j in each iteration.

Using a mean filter with a larger neighborhood may have the undesirable effect of smoothing out details and creases [Yagou et al. 2002]. In order to address that, we introduce a weight w_j to the deformable mean filter, which considers normal similarity in

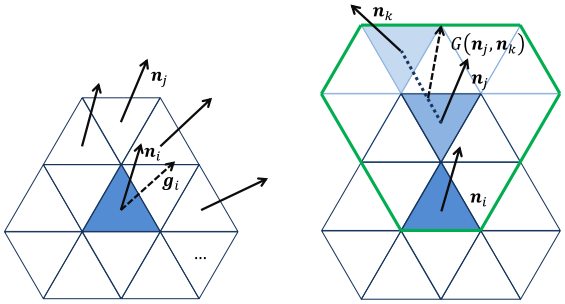


Figure 6: Traversing the 1-ring neighborhood of n_i to attain g_i (left). Instead of directly computing the average normal n_i using n_j, the deformable neighbor filter interpolates n_j with the normal of a random neighbor (right).

order to reduce the impact of normals with significantly different orientation. Formally,

$$\mathbf{g}_i = \frac{\sum_{f_j \in N_i, f_k \in N_j} w_j A_j A_k G(\mathbf{n}_j, \mathbf{n}_k)}{|| \sum_{f_j \in N_i, f_k \in N_j} w_j A_j A_k G(\mathbf{n}_j, \mathbf{n}_k)||} \tag{10}$$

$$w_j = \mathbf{max}(\mathbf{n}_i \cdot G(\mathbf{n}_j, \mathbf{n}_k), 0) \tag{11}$$

$$G(\mathbf{n}_j, \mathbf{n}_k) = aA_j\mathbf{n}_j + (1-a)A_k\mathbf{n}_k \tag{12}$$

$$a = \begin{cases} 0.5 & k = i \\ \dfrac{||c_k - c_i||}{||c_k - c_i|| + ||c_j - c_i||} & \text{otherwise} \end{cases} \tag{13}$$

where c_i is the centroid of face f_i,. Note that $G(\mathbf{n}_j, \mathbf{n}_k)$ is the interpolated normal between f_j and f_k according to their relative Euclidean distances to f_i.

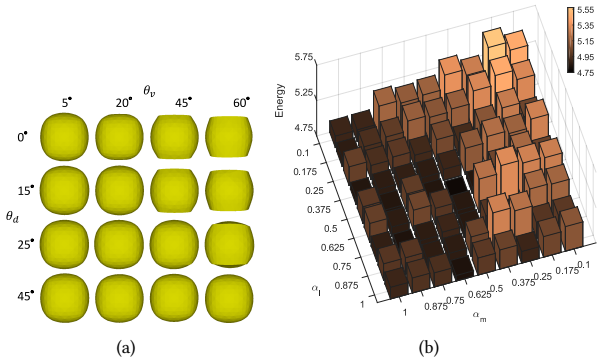
The combination of weights w and a results higher influence to faces that are both closer in Euclidean distance as well as normal orientation to f_i. The effects of introducing each of these modifications separately are shown in figure 10 and discussed in the results section.

4.2 Bilateral filtering with roof flattening

From figure 10, we can see that some undesirable small-scale features still remain. To further address this issue while preserving the large-scale structure, we apply a bilateral filter [Zheng et al. 2011] in each iteration. Prior to applying the bilateral filter, we adjust the resulting normals $\{\mathbf{g}_i\}$ from the the deformable mean filter, to yield flatter roof structures.

Roof flattening. To satisfy the need of creating a balcony or flat roof, we introduce a normal adjustment step for triangles that are nearly horizontal. The agressiveness of the adjustment is determined by two parameters: θ_v which determines an angular distance threshold to the vertical direction (z-axis), and θ_d which prevents the adjustment if the neighborhood is not sufficiently flat. Formally the updated normal is given by

$$\bar{\mathbf{g}}_i = \begin{cases} \mathbf{z} & \mathbf{g}_i \cdot \mathbf{z} > cos\theta_v \quad \& \quad \mathbf{g}_i \cdot \mathbf{g}_j < cos\theta_d \ \ \forall j \\ \mathbf{g}_i & \text{otherwise} \end{cases} \tag{14}$$



(a) (b)

Figure 7: (a) The result of using different θ_v and θ_d for roof flattening; (b) The performance of cuckoo search with different α_l and α_m

As illustrated in figure 7(a), note that by increasing the angular threshold θ_v, the flattened region generally increases. Meanwhile, by increasing θ_d, only regions that exhibit lower normal variation are considered, and thus the flattened region generally decreases.

Bilateral filtering. Given the normal field $\{\bar{\mathbf{g}}_i\}$, we apply the bilateral filter, resulting in an updated set of normals $\{\bar{\mathbf{n}}_i\}$:

$$\bar{\mathbf{n}}_i = \frac{\sum_{f_j \in N_i} A_j K_s(c_i, c_j) K_r(\bar{\mathbf{g}}_i, \bar{\mathbf{g}}_j) \bar{\mathbf{g}}_j}{|| \sum_{f_j \in N_i} A_j K_s(c_i, c_j) K_r(\bar{\mathbf{g}}_i, \bar{\mathbf{g}}_j) \bar{\mathbf{g}}_j ||} \tag{15}$$

$$K_s(\mathbf{c_i}, \mathbf{c_j}) = \mathbf{exp}(-\frac{||c_i - c_j||^2}{2\sigma_s^2}) \tag{16}$$

$$K_r(\bar{\mathbf{g}}_i, \bar{\mathbf{g}}_j) = \mathbf{exp}(-\frac{||\bar{\mathbf{g}}_i - \bar{\mathbf{g}}_j||^2}{2\sigma_r^2}) \tag{17}$$

After applying the bilateral filter, we update each vertex position according to $\{\bar{\mathbf{n}}_i\}$:

$$v_i^{t+1} = v_i^t + \frac{1}{|F_i|} \sum_{j \in F_i} \bar{\mathbf{n}}_j [(c_j^t - v_i^t) \cdot \bar{\mathbf{n}}_j] \tag{18}$$

where F_i is the set of faces incident on vertex v_i.

Results discriminating the contributions of each of these steps are discussed in section 5.

5 RESULTS

We first explored the parameter space of the coefficients α_l and α_m as well as the number of iterations. 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 best performance with higher parameter values (≥ 0.5) (see figure 7(b)). For the results in the paper, we used $\alpha_l = \alpha_m = 0.5$.

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 30 iterations. This is shown in figure 9. No significant drop in the energy was observed beyond 30 iterations.

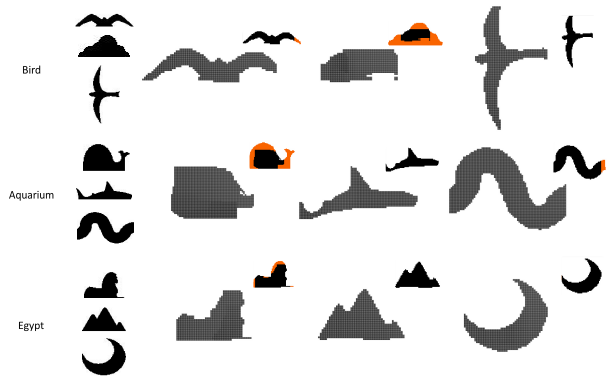


Figure 8: Shadow Art for our architectural designs. The desired shapes are shown on the left. The three columns to the right show the results, with inset figures depicting the amount of deformation from the original shape (in red).

5.1 Verifying the modified cuckoo search

Next we compare the performance of our modified cuckoo algorithm. Six sets of experiments were conducted. Initially, all examples were run by using ORIG in figure 9. Next, we ran a modified cuckoo search with a *fixed likelihood parameter replacement* (FLPR) approach. Then, we ran a different variation of the search in which each parameter is dropped based on an adaptive probability (ALPR). Finally, we also compared with strategies using simulated annealing (SA), genetic algorithm (GA), and particle swarm optimization (PSO). We explored parameter space of these other strategies using the same amount of potential solutions so as to best minimize our energy function.

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. Refer to table 1 for processing times, and figure 9 for the convergence results. In all experiments, our ALPR approach outperformed the other techniques.

5.2 Mesh smoothing

Next, we present results of our smoothing algorithm. Recall that the goal of our method is to smooth the surface preserving large scale sharp creases while also favoring a flat roof structures where applicable. Figure 10 shows the progression of the results after including each of our proposed modifications. It starts with the classical mean filter, then includes the crease preserving weight w from equation 11, the deformable mean filter without and with w, the bilateral filter approach, and finally our combined two-stage approach that includes the deformable mean filter with w and the bilateral filter. This approach yields a smoother surface with sharper creases. Figure 11 compares our results with different techniques applied to our Egypt museum building design. Note again the sharper well-defined creases and smoother surfaces from our approach. The processing times of all techniques are shown in table 1. Finally, figure 12 shows results applied to standard meshes used in literature for benchmarking denoising techniques. We chose the parameters for each denoising techniques which produce visually the best results. The D_{mean}, D_{max} and D_{RMS} in table 2 is measured by Hausdorff distance to the original noise free mesh [Cignoni et al. 1998]. Note that for this last comparison, we do not perform roof flattening. The supplemental material contains additional comparisons of our smoothing method.

5.3 Example designs

Table 1 shows the statistics of using our system to design four examples. Figures 1 and 13 illustrate their designs and rendered results. In each figure, the top 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. 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.

Figure 8 show the examples generated by the shadow art algorithm [Mitra and Pauly 2009]. 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.

6 CONCLUSION

We propose a new technique 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 modified cuckoo search to find low energy optimized initial design, followed by a novel smoothing algorithm to produce an plausible final building shape. 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.

ACKNOWLEDGEMENTS

This work was partly supported by Hong Kong GRF grant #618513.

REFERENCES

P. Block and L. Lachauer. 2011. Closest-Fit, Compression-Only Solutions for Free Form Shells. In *Proceedings of the IABSE-IASS Symposium 2011*, Vol. 2011. London, UK.

Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. 1998. Metro: Measuring error on simplified surfaces. In *Computer Graphics Forum*, Vol. 17. Wiley Online Library, 167–174.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable Convolutional Networks. *arXiv preprint arXiv:1703.06211* (2017).

D Dutta and Y L Srinivas. 1992. Reconstruction of curved solids from two polygonal orthographic views. *Computer-Aided Design* 24, 3 (1992), 149–159.

Lubin Fan and Peter Wonka. 2016. A Probabilistic Model for Exteriors of Residential Buildings. *ACM Trans. Graph.* 35, 5, Article 155 (July 2016), 13 pages. https://doi.org/10.1145/2910578

Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. 2003. Bilateral mesh denoising. In *ACM transactions on graphics (TOG)*, Vol. 22. ACM, 950–953.

I. J. Grimstead and R. R. Martin. 1995. Creating Solid Models from Single 2D Sketches. In *Proceedings of the Third ACM Symposium on Solid Modeling and Applications (SMA '95)*. ACM, New York, NY, USA, 323–337. https://doi.org/10.1145/218013.218082

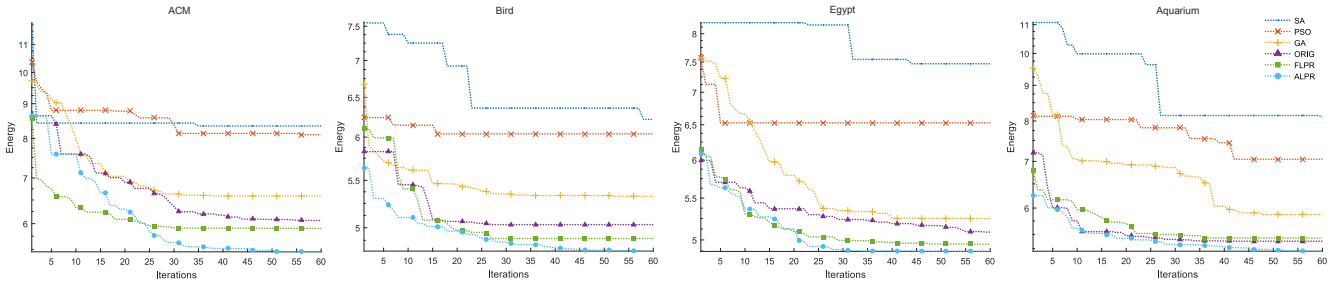


Figure 9: Convergence of the different optimization strategies after 60 iterations.

Table 1: Processing time of all methods with 60 iterations used in the comparison. The timings were measured on an Intel i5-4590.

Input	Optimization Methods and Time(min)						Smoothing Methods and Time(s)							statistic	
Building	SA	PSO	GA	ORIG	FLPR	ALPR	BMD	NIFP	FEFP	BNF	L0	GMN	Ours	Voxel Grid Res	$\|F\|$
ACM	63	64	79	171	207	224	3.88	59.94	1.49	1.69	104.21	26.80	30.18	$123 \times 123 \times 50$	12396
Bird	46	43	65	76	85	119	8.17	180.89	1.67	1.84	105.01	37.40	12.79	$154 \times 154 \times 22$	11736
Egypt	127	133	250	300	325	415	3.21	70.67	1.26	1.40	171.45	34.35	13.93	$123 \times 123 \times 45$	13320
Aquarium	141	162	243	423	490	545	2.95	62.46	1.44	1.34	146.80	26.29	14.50	$174 \times 174 \times 37$	12568

Table 2: Error of all methods used in the comparison relative to the original noise-free mesh.

Model	Error($\times 10^{-2}$)	BMD	NIFP	FEFP	BNF	L0	GMN	Ours
Twelve	D_{mean}	1.425	1.252	0.894	0.955	1.681	**0.479**	0.551
$\|F\| = 9216$	D_{max}	13.567	10.733	5.715	7.047	10.150	3.446	**2.912**
$\|V\| = 4610$	D_{RMS}	1.946	1.622	1.186	1.278	2.204	**0.645**	0.710
	Time(s)	2.8	1.0	17.8	2.2	252.8	86.4	114.6
Sphere	D_{mean}	9.848	6.201	6.978	6.873	10.308	4.827	**4.399**
$\|F\| = 20882$	D_{max}	70.314	44.603	47.070	39.756	32.715	28.711	**22.319**
$\|V\| = 10443$	D_{RMS}	13.031	7.861	8.614	8.410	12.271	6.173	**5.520**
	Time(s)	11.1	3.8	114.5	2.0	304.9	90.5	66.8

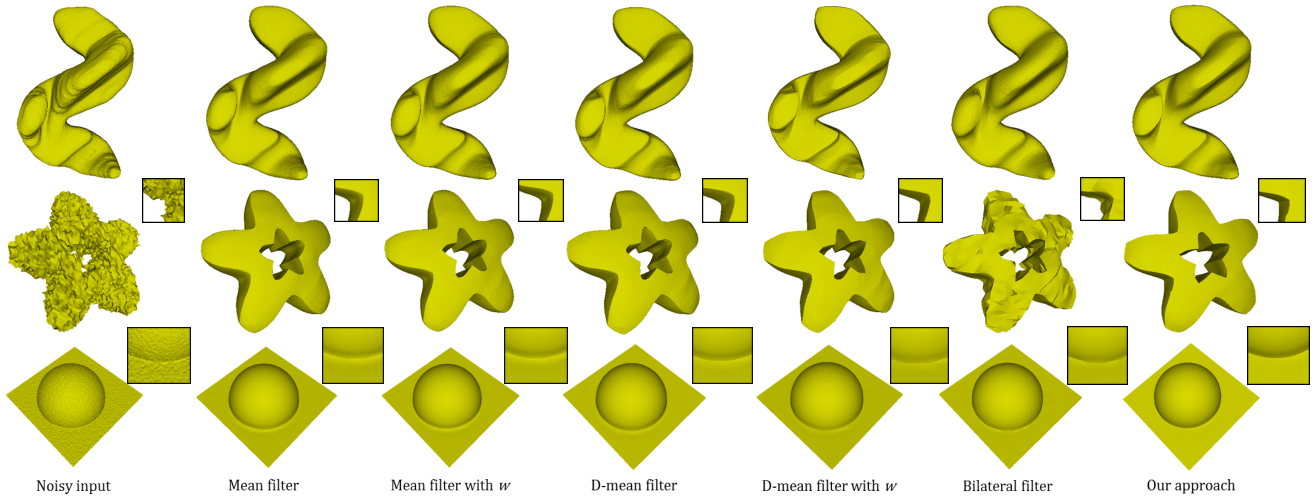


Figure 10: Progression of smoothing results using each of our proposed modifications. D-mean filter is the deformable mean filter from section 4.

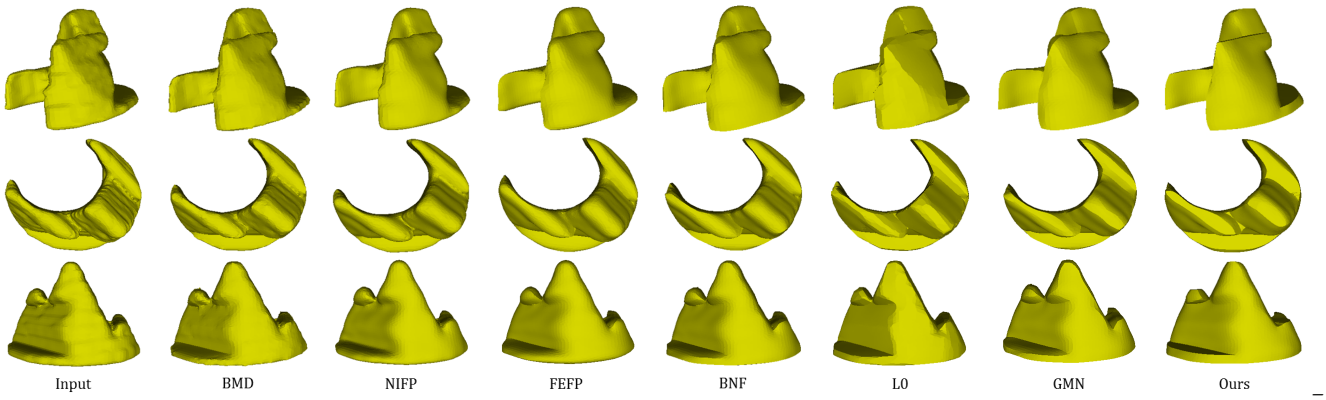


Figure 11: Comparison of different methods applied on Egypt building model. BMD: Bilateral mesh denoising [Fleishman et al. 2003]; NIFP: Non-iterative, feature preserving mesh denoising [Jones et al. 2003]; FEFP: Fast and effective feature preserving mesh denoising [Sun et al. 2007]; BNF: Bilateral normal filtering for mesh denoising [Zheng et al. 2011]; L0: Mesh denoising via L0 minimization [He and Schaefer 2013]; GMD: Guided mesh normal filtering [Zhang et al. 2015].

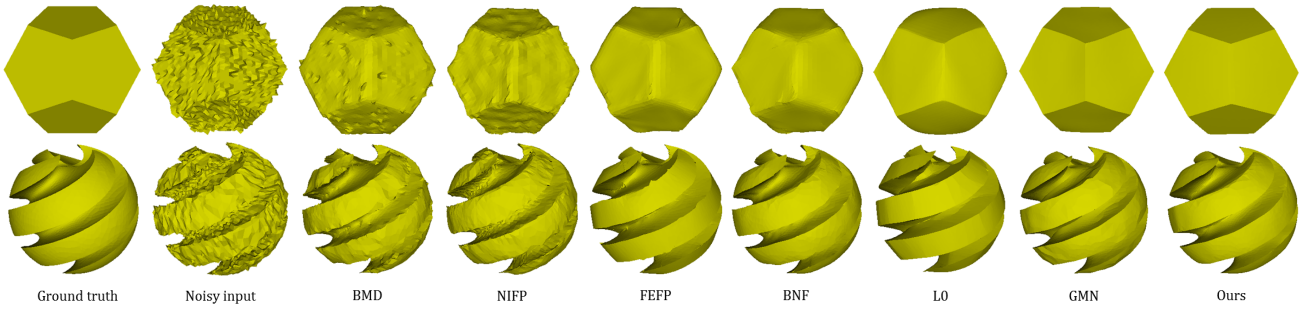


Figure 12: Comparison of the same methods from figure 11 applied on a 3D noisy model.

Lei He and Scott Schaefer. 2013. Mesh denoising via L 0 minimization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 64.

Jonathan D. Hiller and Hod Lipson. 2012. Dynamic Simulation of Soft Heterogeneous Objects. *CoRR* abs/1212.2845 (2012). http://arxiv.org/abs/1212.2845

Caigui Jiang, Chengcheng Tang, Marko Tomicic, Johannes Wallner, and Helmut Pottmann. 2014. Interactive modeling of architectural freeform structures - combining geometry with fabrication and statics. In *Advances in Architectural Geometry*, P. Block, J. Knippers, and W. Wang (Eds.). Springer.

Thouis R Jones, Frédo Durand, and Mathieu Desbrun. 2003. Non-iterative, feature-preserving mesh smoothing. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 943–949.

A. Laurentini. 1994. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Trans. Pattern Anal. Mach. Intell.* 16, 2 (Feb. 1994), 150–162. https://doi.org/10.1109/34.273735

Charles Loop, Cha Zhang, and Zhengyou Zhang. 2013. Real-time high-resolution sparse voxelization with application to image-based modeling. In *Proceedings of the 5th High-Performance Graphics Conference*. ACM, 73–79.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, Vol. 21. ACM, 163–169.

Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J Gortler, and Leonard McMillan. 2000. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 369–374.

Niloy J Mitra and Mark Pauly. 2009. Shadow art. In *ACM Transactions on Graphics*, Vol. 28. 156–1.

Alec Rivers, Frédo Durand, and Takeo Igarashi. 2010. 3D Modeling with Silhouettes. *ACM Trans. Graph.* 29, 4, Article 109 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778846

Michael Schwarz and Pascal Müller. 2015. Advanced Procedural Modeling of Architecture. *ACM Trans. Graph.* 34, 4, Article 107 (July 2015), 12 pages. https://doi.org/10.1145/2766956

Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2012. A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum* (2012). https://doi.org/10.1111/cgf.12276

Xianfang Sun, Paul Rosin, Ralph Martin, and Frank Langbein. 2007. Fast and effective feature-preserving mesh denoising. *IEEE transactions on visualization and computer graphics* 13, 5 (2007).

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. Graph.* 33, 4, Article 70 (jul 2014), 9 pages. https://doi.org/10.1145/2601097.2601213

Matthew Trager, Martial Hebert, and Jean Ponce. 2016. Consistency of Silhouettes and Their Duals. In *Proceedings of the CVPR (CVPR 2016)*. IEEE, 10.

Peng-Shuai Wang, Xiao-Ming Fu, Yang Liu, Xin Tong, Shi-Lin Liu, and Baining Guo. 2015. Rolling guidance normal filter for geometric processing. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 173.

Emily Whiting, John Ochsendorf, and Frédo Durand. 2009. Procedural Modeling of Structurally-sound Masonry Buildings. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 112, 9 pages. https://doi.org/10.1145/1661412.1618458

Hirokazu Yagou, Yutaka Ohtake, and Alexander Belyaev. 2002. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing, 2002. Proceedings*. IEEE, 124–131.

Xin-She Yang and Suash Deb. 2009. Cuckoo search via Lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. IEEE, 210–214.

Wangyu Zhang, Bailin Deng, Juyong Zhang, Sofien Bouaziz, and Ligang Liu. 2015. Guided mesh normal filtering. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 23–34.

Youyi Zheng, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. 2011. Bilateral normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics* 17, 10 (2011), 1521–1530.

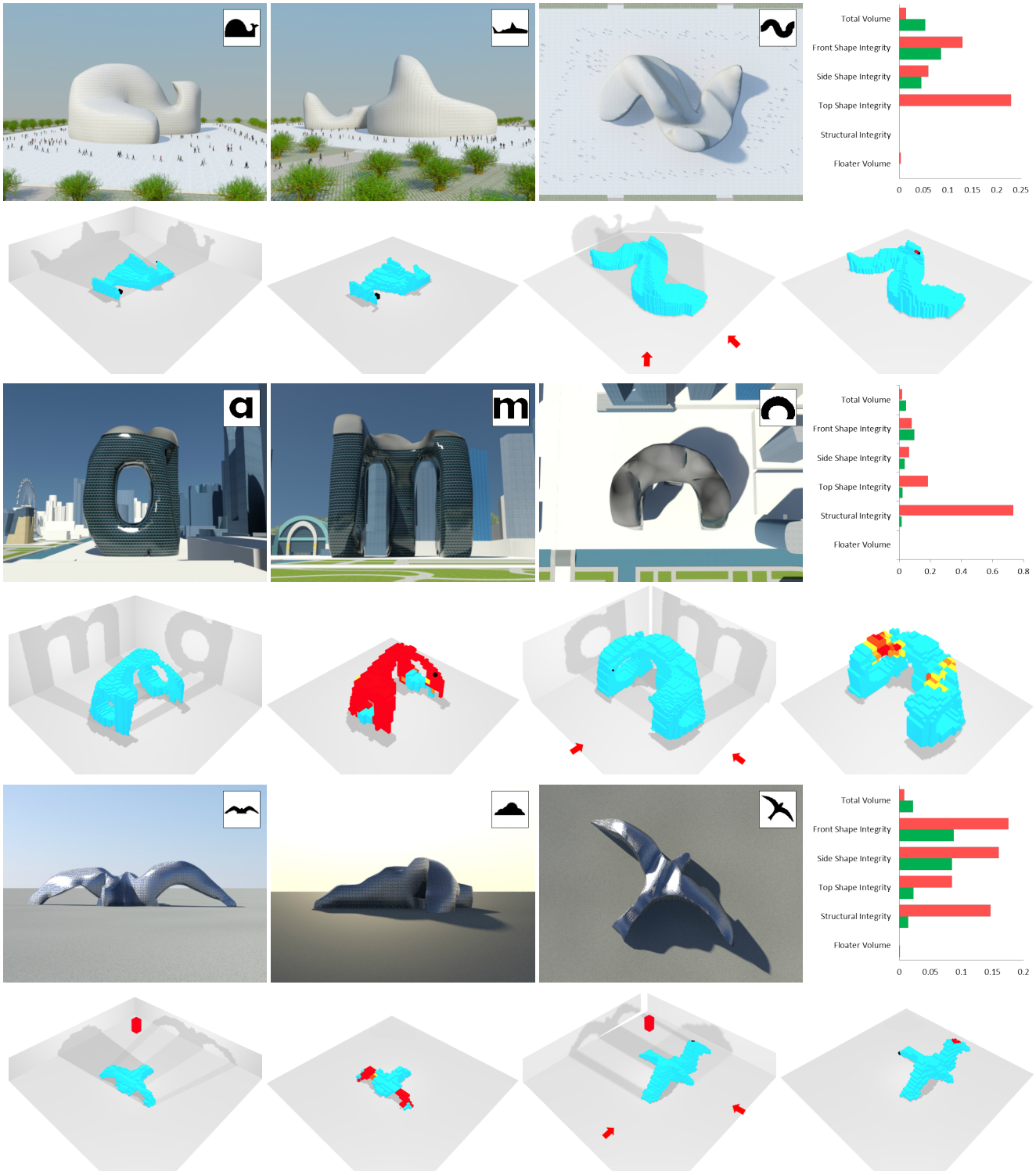


Figure 13: Design of an aquarium building inspired by marine creatures (top), a potential future ACM headquarters (middle) and a bird shaped building (bottom). Each example is based on three binary images shown as insets with the corresponding rendered views in the top row. The second row shows the voxel grid of an initial design, a visualization of its structural and topological integrity, and an optimized design and its respective visualization. The values of the parameters controlling the shape before (red) and after (green) optimization are shown in the bar chart on top right.