Scale-aware Black-and-White Abstraction of 3D Shapes

YOU-EN LIN, National Tsing Hua University
YONG-LIANG YANG, University of Bath
HUNG-KUO CHU, National Tsing Hua University

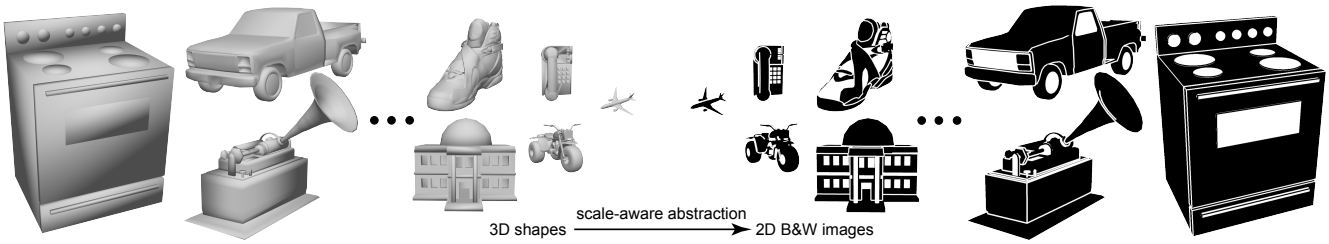


Fig. 1. Proposed computational framework for automatic generation of 2D black-and-white abstractions from 3D shapes which preserves geometric and structural properties at different scales. Scales of abstraction results in both paper and supplementary material were adaptively selected such that feature lines are recognizable and consistent over different scales.

Flat design is a modern style of graphics design that minimizes the number of design attributes required to convey 3D shapes. This approach suits design contexts requiring simplicity and efficiency, such as mobile computing devices. This 'less-is-more' design inspiration has posed significant challenges in practice since it selects from a restricted range of design elements (e.g., color and resolution) to represent complex shapes. In this work, we investigate a means of computationally generating a specialized 2D flat representation - image formed by black-and-white patches - from 3D shapes. We present a novel framework that automatically abstracts 3D man-made shapes into 2D binary images at multiple scales. Based on a set of identified design principles related to the inference of geometry and structure, our framework jointly analyzes the input 3D shape and its counterpart 2D representation, followed by executing a carefully devised layout optimization algorithm. The robustness and effectiveness of our method are demonstrated by testing it on a wide variety of man-made shapes and comparing the results with baseline methods via a pilot user study. We further present two practical applications that are likely to benefit from our work.

CCS Concepts: • **Computing methodologies** → **Computer graphics**;

Additional Key Words and Phrases: black-and-white image, scale-aware abstraction, joint 2D/3D analysis, layout optimization

ACM Reference Format:
You-En Lin, Yong-Liang Yang, and Hung-Kuo Chu. 2018. Scale-aware Black-and-White Abstraction of 3D Shapes. *ACM Trans. Graph.* 37, 4, Article 1 (August 2018), 11 pages. https://doi.org/10.1145/3197517.3201372

Authors' addresses: You-En Lin, Department of Computer Science, National Tsing Hua University, unlin@livemail.tw; Yong-Liang Yang, Department of Computer Science, University of Bath, y.yang@cs.bath.ac.uk; Hung-Kuo Chu, Department of Computer Science, National Tsing Hua University, hkchu@cs.nthu.edu.tw.

1 INTRODUCTION

With the recent development of mobile computing technologies, a fascinating design style called *flat design* has gained ever-increasing popularity. In contrast to 'skeuomorphic' design, which aims for a high degree of realism using ornamental design attributes (e.g., shadows, gradients, textures, or decorations, etc.), flat design relies on the minimum number of stylistic elements required for the illusion of three dimensions [Page 2014]. The resulting design outcomes hold a number of advantages, such as improved readability and legibility, easy compatibility of design style, and less network and memory cost, etc. These advantages serve a range of design contexts applicable to mobile apps, PC desktops, and webpages.

Among different styles of flat design, a specialization that abstracts 3D shapes into 2D binary images is particularly interesting. It applies the minimum set of colors (i.e., black and white) necessary for 2D design to create a concise abstraction of its 3D counterpart (see Figure 2). Due to the constraints on colors and the reduction of dimensions from 3D to 2D, how to preserve the defining characteristics of the 3D geometry and allow visual perception of the underlying 3D shapes become extremely important.

Various intriguing design elements have been used by designers to better infer 3D geometry. For example, as shown in Figure 2, to reflect shape occlusion and facilitate depth perception, designers often introduce a *halo-like gap region* on occluded sections. Designers add

Fig. 2. Example designs of 2D black-and-white abstractions of 3D shapes created by designers. Design elements such as halo-like gap region (red box), hollow line (blue box) and patch (green box) are used to better infer 3D geometry. Image courtesy of https://iconmonstr.com/.

hollow lines and *patches* to generate high contrast and emphasize the geometric features of individual shape components as well as the spatial relations between components. To facilitate the design process, these design elements and others have been collated into reference for practical applications [Abdullah and Hüber 2006]. Yet, even within these guidelines and with the help of modern graphics design tools (e.g., Adobe Illustrator), the design process still demands a high level of skill and is time-consuming. The designer needs to manually manage perspectives, trace boundary lines, and create design elements.

Some techniques exist for automatic shape depiction [Cole et al. 2008] and shape abstraction [Mehra et al. 2009; Yumer and Kara 2012]; However, the former can only generate 2D stylized drawing in a different context (line-based drawing), while the latter focuses on abstractions in 3D. By combining pieces from existing pictogram examples, Liu et al. [2016] present a 2D data-driven iconifiction framework without considering comprehensive geometric information in 3D. Therefore, we are still short of an effective tool for the present design context based on 3D shapes.

In this paper, we present a novel computational design framework to automatically generate 2D black-and-white abstractions from 3D man-made shapes (see Figure 1). The keys to our framework are a number of design principles selected from existing examples and design guidelines, and a set of shape analysis and layout optimization algorithms implementing these design principles. The result is an appealing black-and-white abstraction that well represents the characteristics of the input shape. Our framework takes advantage of the easy access to 3D man-made shapes provided by online repositories, allowing for a variety of quality results from shapes of different categories. Our optimization algorithm is also scale-aware in the sense that it can generate an adaptive abstraction layout according to the specific image size, such that only more prominent features are preserved at smaller scale to avoid visual cluttering. The efficacy of our framework is demonstrated through extensive comparison and a user study. We also exploit our results for two applications that can benefit from our black-and-white abstraction.

In summary, our work makes three major contributions:

- We present the first computational design framework capable of automatically generating 2D black-and-white abstractions of 3D man-made shapes.
- We devise a scale-aware layout optimization algorithm that enables an adaptive abstraction layout responsive to image size.
- We demonstrate novel applications based on the abstraction layout generated by our framework.

2 RELATED WORKS

2.1 Black-and-white Image Stylization

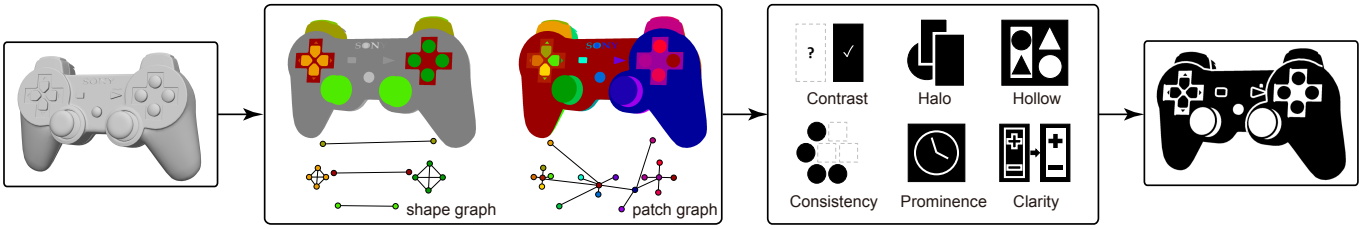
Image stylization studies the conversion of an input image into a new image of a specific style. Several works aim at generating stylized black-and-white images. Xu and Kaplan [2008] depict continuous tones using only black and white colors. They model the problem as graph optimization over a connectivity graph of the segmented input image. Although it also employs graph-based color labelling, our work is fundamentally different in the following aspects: 1) our input is 3D shapes; 2) our formulation is based on realizing flat design guidelines; and 3) our graph contains boundary nodes to create halo effects. Rosin and Lai [2010] abstract a given image with minimal tones (between 3 to 10). Their approach of abstracting image elements formed by lines and blocks is similar to ours, but they try to preserve information by using up to 10 colors instead of only black and white. Li and Mould [2015] enhance the contrast of a given image by converting it into a black-and-white representation. They use an efficient filter-based method instead of exhaustively optimizing tone assignments as suggested by previous works. Although these methods can generate black-and-white images, they are restricted to the 2D domain. Our method enables results from various views and better conveys the geometry of the underlying 3D shapes. It is possible to render a 2D image from 3D shape, followed by simply applying black-and-white image stylization. However, the result is heavily affected by rendering parameters, such as surface material and lighting conditions. Therefore, we propose a joint 2D/3D solution which is intrinsic to the geometry of the input shape.

2.2 Line Drawing of 3D Shapes

Many computer graphics methods have been proposed to depict 3D shapes through computer-generated line drawings [Rusinkiewicz et al. 2008]. There are two major approaches to this problem: image space solution and object space solution. Given that our input is a 3D shape, we mainly discuss the latter approach. Hertzmann and Zorin [2000] use surface silhouettes in which front-facing polygons meet back-facing ones to depict 3D shapes. DeCarlo et al. [2003] extract suggestive contours as feature lines based on surface curvature. Judd et al. [2007] propose apparent ridges that generalize curvature-based feature lines with view-dependent curvature. Cole et al. [2008] comprehensively evaluate the effectiveness of different methods through comparison with line drawings created by artists. While these methods can generate appealing line drawings from different views, they all focus on a line-based representation. This differs from our objective of abstracting 3D shapes in a patch-based manner. These line drawings cannot be effectively used for quality patch-based abstractions (see the comparison and user study in Section 7).

2.3 View-Dependent Shape Segmentation

3D shape segmentation investigates the decomposition of a shape into distinct parts that are semantically meaningful or practically useful, benefiting applications such as mesh parameterization, shape retrieval, and shape editing [Shamir 2008]. We aim at generating 2D abstractions from 3D shapes. The result is a set of 2D binary patches constituting the 2D projection of the 3D shape, which is obviously view-dependent. For view-dependent shape segmentation, Kolliopoulos et al. [2006] decompose 3D scenes with a normalized graph cut. Geometric information such as depth, normals, object ID and group ID are encoded in a multi-channel image, and the segmentation is performed in the image space. Eisemann et al. [2009] propose a view-dependent segmentation algorithm to convert a 3D scene into multiple vector graphics layers, which can be easily edited by artists to create 2D vector illustrations. Their segmentation algorithm is performed in 3D by first extracting visible surfaces, then



(a) Input shape (b) Joint 2D/3D shape analysis (c) Abstraction layout optimization (d) B/W abstraction

Fig. 3. Overview of proposed framework. Given an input 3D shape (a), our framework first performs joint 2D/3D shape analysis to encode geometric and structural properties into a shape graph and a patch graph (b). An abstraction layout optimization is then conducted according to design principles (c) to generate 2D black-and-white abstraction (d). For clarity, we only illustrate a subset of graph nodes in shape graph and patch graph, and the edge direction in patch graph is not shown.

constructing a graph description of the overlapping area, and finally using graph cut to divide the graph into non-overlapping layers. This geometric approach suits our needs for segmentation, despite its computational complexity. Since the hidden layers are unnecessary for our objective, we end up with an efficient segmentation approach in the image space which is suitable for our problem (detailed in Section 5.2).

2.4 Enhanced Depth Visualization

Designers often impose visual cues, such as the halo effect in 2D depictions, to emphasize 3D occlusion, and thereby enhance depth perception of the underlying 3D shapes. In the field of graphic design and visualization, a similar idea has been explored in several studies focused on altering the appearance of the depiction in a depth-dependent way. Appel et al. [1979] use line halo effects to eliminate hidden lines to enhance the wireframe-based shape visualization. Elber [1995] extends this idea by considering other visual properties of the lines, such as line width and color intensity. Luft et al. [2006] enhance depth perception of an image by unsharpening the depth map to create darkened halo effects around depth discontinuities. Bruckner et al. [2007] propose to use flexible volumetric halos to enhance depth perception. The halo effect used in this context is visually similar to ambient occlusion, a real-time technique for the rendering of approximate global illumination. Everts et al. [2009] compute GPU-accelerated line halos according to depth difference to generate black-and-white visualization of large datasets formed by a significant number of lines. In our work, we also take into account 3D depth and generate halo-like regions in occluded patches to enhance depth perception of the resulting 2D black-and-white abstraction. The difference is that we enable local halo effect based on an optimization approach. This not only avoids visual cluttering when the scale of the abstraction shrinks, but also strikes a good balance with other visual cues.

3 DESIGN PRINCIPLES

By investigating existing flat designs created by artists and documentations on design guidelines [Abdullah and Hüber 2006], we have identified a set of important design principles, which are used to guide subsequent analysis and optimization (see Figure 4).

Contrast. The majority of the layout, especially the patches incident to the white background, has to be black to clearly depict the foreground shape.

Halo. Designers often utilize the halo effect (white linear gap between two patches) to highlight the depth ordering of the underlying geometry. To avoid depth ambiguity, a linear gap is created within the occluded patch, which has a larger depth value along the viewing direction.

Hollow. An isolated patch enclosed entirely by another patch is often depicted in a different color for better comprehension.

Consistency. Patches originating from the same 3D shape component or repetitive components favor a consistent color. This helps to preserve the shape semantics, and allows easy shape and structure understanding.

Prominence. Prominent feature lines are often used to better convey details of the underlying geometric shape.

Clarity. Cluttered layout should be avoided to increase readability.

Note that in practice these principles are often in conflict. For example, highlighting all the prominent features at a small scale would cause visual cluttering, which goes against the principle of clarity. Halo and hollow effects cannot co-exist between two patches with both occlusion and inclusion relations. Thus, we leverage an optimization approach to generate abstraction layout in a comprehensive way (see Section 6).

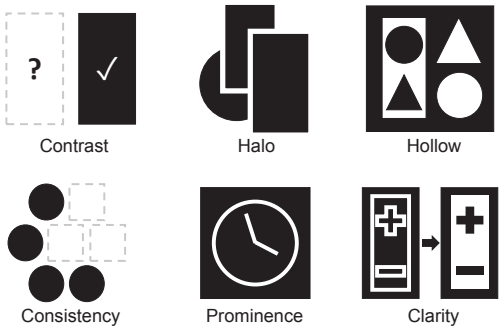
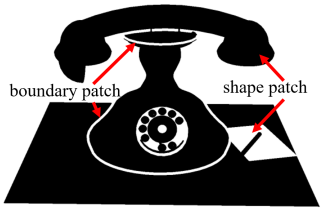


Fig. 4. Set of design principles applied to abstraction layout optimization.

4 OVERVIEW

Given an input 3D man-made shape, a viewing direction, and a specific scale, our framework automatically generates a 2D black-and-white abstraction of the input shape. We assume a triangle

mesh representation of the input shape due to its generality and popularity. The resulting abstraction is a monochromatic image with white background. The foreground is a black-and-white layout composed of patches in either color, altogether forming a 2D projection of the 3D shape. The projection can be either perspective or orthographic, depending on user specification. There are two types of patches in the abstraction layout: *shape patches* that abstract the input shape, and *boundary patches* that highlight either the geometric feature or the spatial structure of the input shape.



As shown in Figure 3, our framework consists of two main components, *joint 2D/3D shape analysis* and *abstraction layout optimization*. The former focuses on understanding the geometry and structure of the input 3D shape along with its 2D projection. More specifically, we jointly analyze the geometric and spatial relations between individual components in 3D and their 2D counterparts, i.e., patches in 2D. The output comprises a *shape graph* and a *patch graph*, which encode the geometric and structural properties of the input shape and its 2D counterpart (Section 5). This serves as the foundation for the latter component, which optimizes the color of each 2D patch, such that the black-and-white 2D patch layout optimally abstracts the geometry and structure of the input shape (Section 6).

5 JOINT 2D/3D SHAPE ANALYSIS

5.1 3D shape analysis

The input to our framework is a multi-component man-made shape represented by triangle mesh, denoted as $\mathcal{M} = \{C_1, ..., C_N\}$. We divide the whole mesh into individual components based on mesh face connectivity. To unify parameter settings, we normalize the input shape into a bounding box ranging from -1.0 to 1.0 along x, y, and z axis. We encode the structure of the multi-component shape using an undirected *shape graph* $\mathcal{G}_s(\mathcal{N}_s, \mathcal{E}_s)$, where graph node $C_i \in \mathcal{N}_s$ represents individual component, and graph edge $E_{ij} \in \mathcal{E}_s$ connects two congruent components C_i and C_j. For simplicity, we use C_i to denote both the i-th mesh component and its corresponding graph node. We rely on the same strategy applied in the work of [Yang et al. 2015] to identify congruent components. We first estimate the oriented bounding box (OBB) of each component. Then the congruency between two components is determined by checking if both OBBs and shapes can be well aligned (see Figure 5). Note that the adjacency relation between individual components is not considered here in 3D, since it can be easily affected by inter- and intra- occlusions when projecting the shape from 3D to 2D. We also extract geometric features by detecting sharp mesh edges, which allows for the highlighting of intra-component properties (details in Section 6.1).

5.2 2D shape analysis

The 2D counterpart of the input 3D shape is generated by camera projection using a standard graphics rendering pipeline. In order to preserve the structural properties of the input shape, we employ a specialized rendering-based segmentation algorithm which jointly considers the 3D shape and its 2D projection. We first render the 3D shape into a 2D image while recording triangle face ID in the frame buffer. Then at the pixel level, we perform a flooding algorithm by grouping pixels from neighboring faces according to mesh connectivity, resulting in a set of 2D patches each of which consists of pixels projected from connected triangle faces in a component. Figure 6 illustrates the process. Note that one mesh component may lead to multiple 2D patches due to occlusion by other component(s) or self-occlusion. In practice, we use a rendering resolution of 1600×1200. The viewing distance is 5.0 and field of view is 60°. Several results can be found in Figure 7.

We use a directed *patch graph* $\mathcal{G}_p(\mathcal{N}_p, \mathcal{N}_b, \mathcal{E}_a, \mathcal{E}_i)$ to encode the structural information of the patch segmentation. There are two types of graph nodes. In addition to patch node $P_i \in \mathcal{N}_p$ which represents individual 2D patch, we also create a boundary node $B_{ij} \in \mathcal{N}_b$ for two neighboring patches P_i and P_j that share a common boundary. The purpose of the boundary node is to better infer the 3D spatial relation between neighboring patches through the generation of a halo effect when appropriate (for details, see Section 6). For each patch node, we also record the ID of its corresponding mesh component. Graph edges comprise two types: *adjacency edges* that simply describe the adjacency between graph nodes, and *inclusion edges* that indicate where one patch node is entirely enclosed by another patch node. Two neighboring patches in the segmentation define two adjacency edges with the direction from the occluding patch node to occluded patch node, passing through the boundary node in-between. The occlusion between two neighboring patches is identified by locally checking the depth values from depth buffer. Inclusion edges always point to the enclosed patch node. We denote the set of adjacency edges and inclusion edges as $\mathcal{E}_a$ and $\mathcal{E}_i$, respectively. Illustrations of a shape graph and a patch graph can be found in Figure 8.

6 ABSTRACTION LAYOUT OPTIMIZATION

Based on $\mathcal{G}_s$ and $\mathcal{G}_p$ constructed above, the second step is to assign black or white color to each patch node $P_i \in \mathcal{N}_p$ along with determining whether the halo effect should be created for each boundary node $B_{ij} \in \mathcal{N}_b$, to form a monochromatic abstraction layout of the input shape. We formulate the layout optimization as a binary labelling problem, since the color set comprises only black or white, and the halo effect is either enabled or disabled. More specifically, for each patch node P_i, we assign a binary color label c_i for which

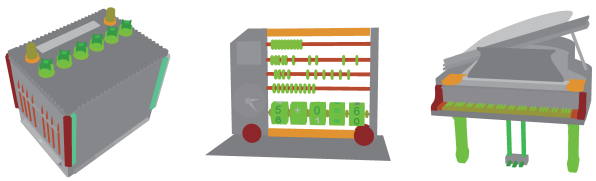


Fig. 5. We perform 3D shape analysis to identify congruent components (denoted by consistent coloring) and unique components, i.e., those without congruency (rendered in gray).

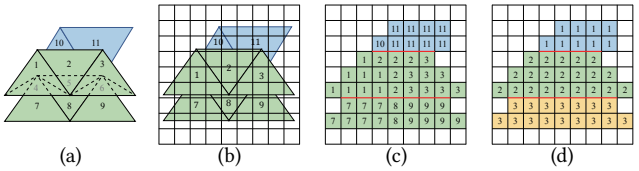


Fig. 6. Illustration of 2D patch generation. Given a 3D shape (a) with two components, its 2D counterpart is generated using a standard graphics rendering pipeline with hidden faces (face 4, 5, and 6) being eliminated (b), resulting in rasterized image (c). A flooding algorithm is then performed based on mesh connectivity to generate 2D patches (d). Boundaries between patches are highlighted in red.

$c_i = 0$ denotes black, and $c_i = 1$ denotes white. For each boundary node B_{ij}, we assign a binary label h_{ij} to indicate whether halo effect is enabled ($h_{ij} = 1$) or not ($h_{ij} = 0$). The resulting 2D abstraction is generated by maximizing the objective function as follows:

$$f(c_i\text{'s}, h_{ij}\text{'s}) = w_n f_n(c_i\text{'s}, h_{ij}\text{'s}) + w_c f_c(c_i\text{'s}) + w_b f_b(c_i\text{'s}), \quad (1)$$

where three energy terms f_n, f_c, and f_b represent respectively *neighbor* energy, *consistency* energy, and *background* energy, with weights w_n, w_c, and w_b used to control the relative influence. Each of these enery terms is tailored to realizing the design principles discussed in Section 3. We now elaborate each enery term in details.

Neighbor energy. This energy term highlights the feature and spatial relation between neighboring patches, and is defined on every adjacent node triplet (P_i, B_{ij}, P_j) from $\mathcal{G}_p$, where $P_i, P_j \in \mathcal{N}_p$, $B_{ij} \in \mathcal{N}_b$ and $E_i^{ij}, E_{ij}^j \in \mathcal{E}_a$. For each triplet (P_i, B_{ij}, P_j), different abstraction layouts can be generated according to the values assigned to c_i, h_{ij}, and c_j. We further classify the triplet layouts into four types according to layout characteristics, including 'merge', 'contrast', 'halo', and 'invalid'. The last type enables the halo effect in the wrong context, i.e., when two neighboring patches are with different colors. As a result, it is not favored by our optimization. The other three types are suitable for different scenarios. The 'contrast' type layout is good for highlighting spatial inclusion between two adjacent patches which are close in depth. The 'halo' type is effective to reflect the occlusion relation between two adjacent patches with significant depth difference, since the halo effect highlights the depth ordering. The 'merge' type is only preferred when patches are small and the 'contrast' or 'halo' type cannot be realized due to lack of space or recognition. Table 1 lists all possible layouts

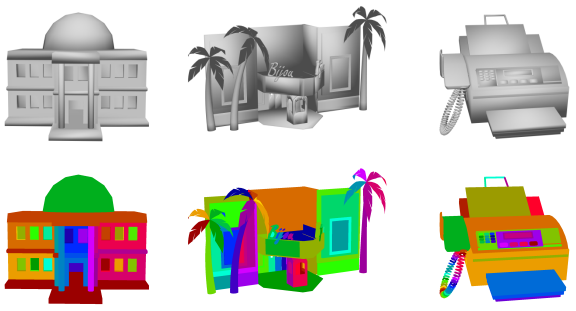


Fig. 7. Examples of 2D patches (bottom) generated from 3D shapes (top).

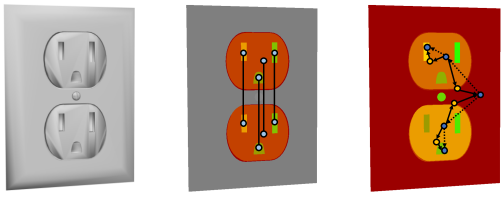


Fig. 8. Input shape (left) with corresponding shape graph (middle) and patch graph (right). Only a subset of graph nodes is shown for clarity. In shape graph, undirected edges connect congruent parts. In patch graph, directed edges either connect patch node and boundary node (through solid adjacency edges), or two patch nodes of which one encloses the other (through dashed inclusion edges). The direction of the edge indicates occlusion and inclusion relations.

Table 1. Possible labellings and corresponding layout types for a patch-boundary-patch triplet. B=black, W=white, D=disable halo, E=enable halo.

P_i	B	W	B	W	B	W	B	W
B_{ij}	D	D	D	D	E	E	E	E
P_j	B	W	W	B	B	W	W	B
Type	merge	merge	contrast	contrast	halo	halo	invalid	invalid

and corresponding types. In the following, we define quantitative measurements for scoring the quality of four layout types.

For 'contrast' type, the score is defined as follows:

$$s_{con}(c_i, h_{ij}, c_j) = ScaleContrast(P_i, P_j) * Relative(P_i, P_j) * \big(1 - d(B_{ij})\big) * Inclusion(P_i, P_j), \quad (2)$$

where $ScaleContrast(P_i, P_j)$ is an activation-type function which only enables color contrast when patch width is large enough for the patch to be recognized, $Relative(P_i, P_j)$ measures how much P_i and P_j are correlated, $d(B_{ij})$ is the normalized depth difference at B_{ij}, and $Inclusion(P_i, P_j)$ reflects if P_j is enclosed by P_i. Note that due to the involvement of the activation-type function, the constituent functions in the quality score are combined via multiplication rather than addition. This only allows color contrast if the patch size is reasonable, a feature which cannot be easily realized using addition. More specifically, the individual functions are defined as follows:

$$ScaleContrast(P_i, P_j) = \max\left(0, 1 - \frac{Width}{\min\big(2D(P_i), 2D(P_j)\big)}\right), \quad (3)$$

where $Width$ is the minimum recognizable patch width (5 pixels in our implementation), $D(P_i)$ is the maximum distance transform value within patch P_i, estimating patch width from patch boundary.

$$Relative(P_i, P_j) = b(P_i, P_j) * \min\big(a(P_i), a(P_j)\big), \quad (4)$$

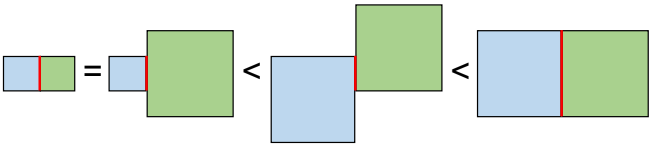


Fig. 9. Illustration of the relative score in Equation 4 for measuring the correlation between two adjacent 2D patches (colored squares). It encourages patches with large area and distinct shared boundary (red line) for better visual recognition.

where $b(P_i, P_j)$ is the normalized length of the shared boundary between P_i and P_j, $a(P_i)$ is the normalized area of P_i, same for P_j. It can be seen in Figure 9 that P_i and P_j are strongly correlated if they can be easily recognized individually (i.e., with large patch area), and they are in good contact (i.e., with long boundary length). Note that similar measurements were also used in [Xu and Kaplan 2008].

$$Inclusion(P_i, P_j) = \begin{cases} W_I, E_i^j \in \mathcal{E}_i \\ 1, E_i^j \notin \mathcal{E}_i \end{cases} \quad (5)$$

This returns W_I if P_i and P_j are connected by an inclusion edge in $\mathcal{G}_p$, otherwise 1. We use $W_I = 1.5$ in our implementation to further encourage contrast for inclusion patches.

For 'halo' type, the score is defined as follows:

$$s_{halo}(c_i, h_{ij}, c_j) = ScaleHalo(P_i, P_j) * Relative(P_i, P_j) * d(B_{ij}) \quad (6)$$

where $Relative(P_i, P_j)$ and $d(B_{ij})$ are the same as for 'contrast' type. Similar to $ScaleContrast$, $ScaleHalo(P_i, P_j)$ is an activation-type function that only allows halo effect with appropriate patch width, which is defined as follows:

$$ScaleHalo(P_i, P_j) = \max\left(0, 1 - \frac{Width}{\min\left(D(P_i), D(P_j)\right)}\right). \quad (7)$$

Unlike $ScaleConstrast$, here we use $D(P_i)$ instead of $2D(P_i)$ to further restrict the patch size. Since the halo region is created on the occluded patch, both of them need to be visible in the final layout.

The last two types 'merge' and 'invalid' are simply defined as constants:

$$s_{merge}(c_i, h_{ij}, c_j) = 1e{-}8, \quad (8)$$

and

$$s_{invalid}(c_i, h_{ij}, c_j) = 0. \quad (9)$$

Note that the 'invalid' case is unfavorable, and thus has no quality score. The 'merge' case is favored when both 'contrast' and 'halo' are not activated; a small positive value is then returned.

The overall neighbor energy f_n is defined by enumerating all triplet scores in different cases as explained above:

$$f_n(c_i\text{'s}, h_{ij}\text{'s}) = \sum_{E_i^{ij}, E_{ij}^j \in \mathcal{E}_a} s_n(c_i, h_{ij}, c_j), \quad (10)$$

where

$$s_n(c_i, h_{ij}, c_j) = \begin{cases} s_{con}(c_i, h_{ij}, c_j), \text{case 'contrast'} \\ s_{halo}(c_i, h_{ij}, c_j), \text{case 'halo'} \\ s_{merge}(c_i, h_{ij}, c_j), \text{case 'merge'} \\ s_{invalid}(c_i, h_{ij}, c_j), \text{case 'invalid'} \end{cases} \quad (11)$$

Consistency energy. This energy term encourages color consistency between two patches P_m and P_n if their corresponding components are the same or congruent. It is defined as follows:

$$f_c(c_i\text{'s}) = \sum_{i_m = i_n, E_{i_m i_n} \in \mathcal{E}_s} \text{XNOR}(c_m, c_n) * \min(a(P_m), a(P_n)), \quad (12)$$

where i_m is the component ID of P_m, XNOR is the exclusive-nor operator, and $a(P_m)$ is the area of patch P_m. Function $\min(a(P_m), a(P_n))$ is used to weigh the influence according to patch size.

Fig. 10. Black-and-white abstraction layouts without (left) and with (right) feature lines highlighted. Note that the asymmetry of the left layout (with feature lines) is due to the fact that the input 3D model is not symmetric (although it looks so without shading).

Background energy. This energy term encourages the color contrast between the background and the patches incident to the background. It is defined as follows:

$$f_b(c_i\text{'s}) = \sum_{P_k \in \bar{\mathcal{N}}} \text{XOR}(c_k, \bar{c}) * Relative(P_k, \bar{P}), \quad (13)$$

where $\bar{\mathcal{N}}$ is the set of graph nodes incident to background patch $\bar{P}$, $\bar{c} = 1$ denotes white background color, XOR is the exclusive-or operator, and $Relative$ is the correlation function defined as before.

Optimization. We use belief propagation [Yedidia et al. 2003] to maximize the objective function defined in Equation 1 and obtain the optimal assignment of patch colors and halo effect labels. We empirically set $w_n = 1.0$, $w_c = 1.0$, $w_b = 5.0$ in all experiments.

6.1 Highlighting sharp features

The optimized abstraction layout primarily considers the geometric and structural properties at an inter-component level. The intra-component geometric properties (e.g., feature lines), except self-occlusions, have yet to be considered. To better address this, we extract sharp edges as prominent features for each mesh component due to their effectiveness of representing man-made shapes [Gal et al. 2009]. The sharp edges are defined as mesh edges shared by triangles with acute dihedral angle ($< 80°$). Then, we project sharp edges to 2D and trace 2D line segments by a flooding algorithm. In practice, we involve a feature line $\mathcal{L}$ into the final layout only if i) it is salient, and ii) it does not affect the appearance of the existing layout. The former criterion is modeled as $Length(\mathcal{L}) > W_F * Width$, where $Length(\mathcal{L})$ is the length of $\mathcal{L}$, and W_F weighs the saliency and is fixed to 10 in our implementation. The latter is ensured by enforcing $D(P_{\mathcal{L}}) > Width$, where $P_{\mathcal{L}}$ is the patch where $\mathcal{L}$ locates. This concept echoes that of the activation of halo features. Figure 10 presents a comparison between layouts with and without sharp features highlighted.

6.2 Abstraction layout realization

Once we have estimated how to abstract patch nodes, boundary nodes, and sharp features, we can now render the final abstraction layout. First, we assign color to each patch P_i according to the optimal color label c_i, resulting in a black-and-white segment layout, where each segment corresponds to a shape patch in the final layout. Then we generate boundary patches on top of the initial shape

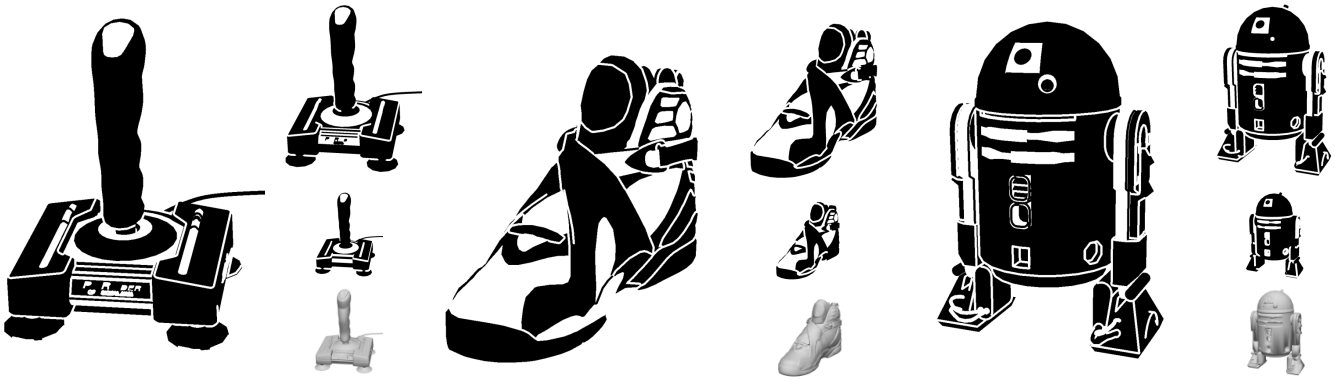


Fig. 11. Black-and-white abstractions generated at different scales ($s = 0.7, 0.3, 0.1$).

patches as follows. For each boundary node B_{ij} with binary label $h_{ij} = 1$, we extract the boundary between two neighboring patches P_i and P_j. Then we use a disk-shaped mask with radius $Width$ centered at each boundary pixel to sculpt a halo region (also a boundary patch) on the occluded patch P_j based on boolean operation. It is easy to see that the width of the halo region is $Width$, which is the same parameter indicating the minimum recognizable patch width as used in the layout optimization. Boundary patches induced by sharp feature lines are generated similarly. For a 2D feature line $\mathcal{L}$, we use a disk-shaped mask with radius $Width/2$ centered at each pixel of $\mathcal{L}$ to sculpt a feature line region (also a boundary patch) with width equal to $Width$.

6.3 Scale-aware layout generation

Our layout optimization and generation algorithm is highly flexible. It can be used to generate abstraction layouts at multiple scales, where more detailed features are preserved at a large scale to better depict the input shape, while only prominent features are kept at a small scale to avoid visual cluttering and improve readability. By default, the resolution of the projected 2D image is 1600×1200, which corresponds to the largest scale. A straightforward approach to generate layout at a smaller scale is to reduce 2D patch sizes by increasing viewing distance of the input shape, and keep the same $Width$ in the optimization. However, it largely affects the quality of patch segmentation in Section 5.2 and easily leads to errors in segmentation results. This is because small triangle faces could be eliminated due to rasterization when using a low-resolution image. Instead, we propose computing all layouts in the highest resolution by varying $Width$ (from 5 to 35 in our implementation) in the optimization. The final result is obtained by down-sampling the optimized layout such that the minimum recognizable patch width is still 5. To simplify the parameter setting, we leverage a scaling factor $0 \leq s \leq 1$ for intuitive control. Specifically, $s = 1$ corresponds to the largest scale with $Width = 5$, while $s = 0$ implies the smallest scale with $Width = 35$, and s is linearly mapped to $Width$ for in-between values. Figure 11 shows examples of abstraction layouts generated at three different scales. Note that the result at the smaller scale is not simply down-sampled from the large-scale result but optimized separately. It is possible to increase the $Width$ value beyond 35 to produce even smaller abstractions. Its limit is restricted by the size of the projected shape.

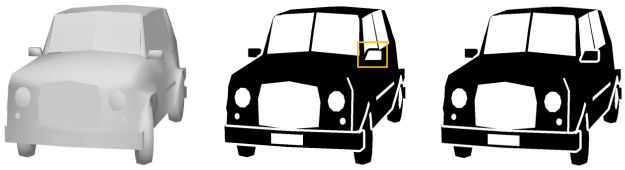


Fig. 12. User editing example. Given an input shape (left), the automated result has different colors for two back mirrors (middle), as the reflective symmetry between components is not considered in our framework. In this context, the user can specify black color for the highlighted back mirror, and our framework performs constrained optimization to generate the new result (right).

6.4 Optional user control

Although our algorithm runs automatically, our framework also provides design freedom by allowing the user to specify color for shape patches. The layout optimization adopts the user constraints by fixing the corresponding color labels while optimizing the other variables. A user editing example is shown in Figure 12.

7 RESULTS AND EVALUATION

We have tested our framework on a variety of man-made shapes in different categories from [Chen et al. 2003]. Figure 13 shows a gallery of automated abstraction results, which fulfill the design principles and well represent the characteristics of the input shape. Our framework is also capable of generating abstraction results at different scales, where the level of details is adaptive to the specific scale, enabling good readability of the resulting abstraction layout. In addition to Figure 11, more results can be found in the supplementary material.

7.1 Performance

Our computational framework is integrated into a GUI application using C++. All the experiments are conducted on a moderate machine with 3.80GHz CPU, 16GB memory, and NVIDIA GTX 1070 graphics card. The computational performance is mainly affected by

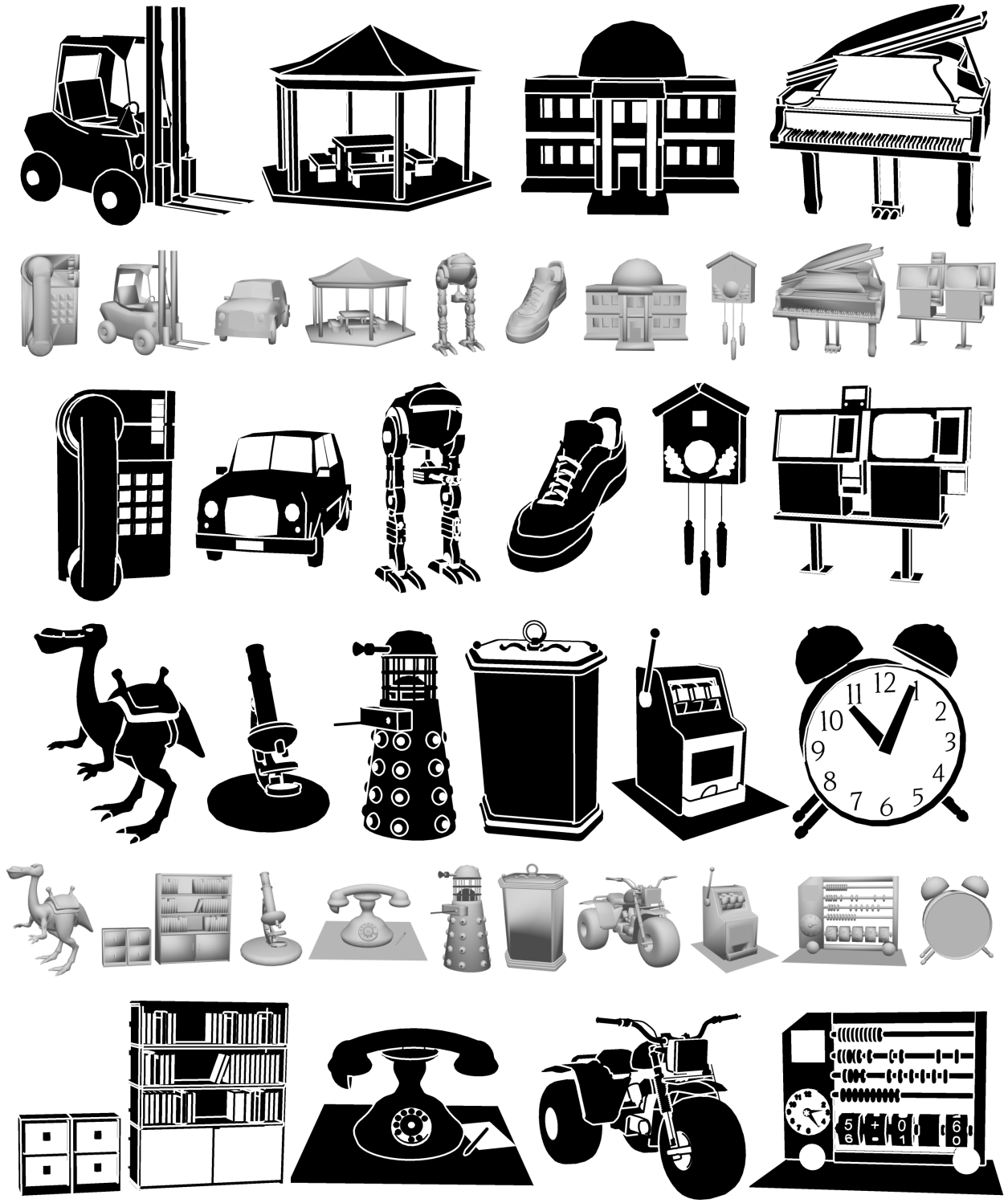


Fig. 13. Gallery of black-and-white abstractions generated by proposed framework on a variety of man-made shapes with scale $s = 0.5$.

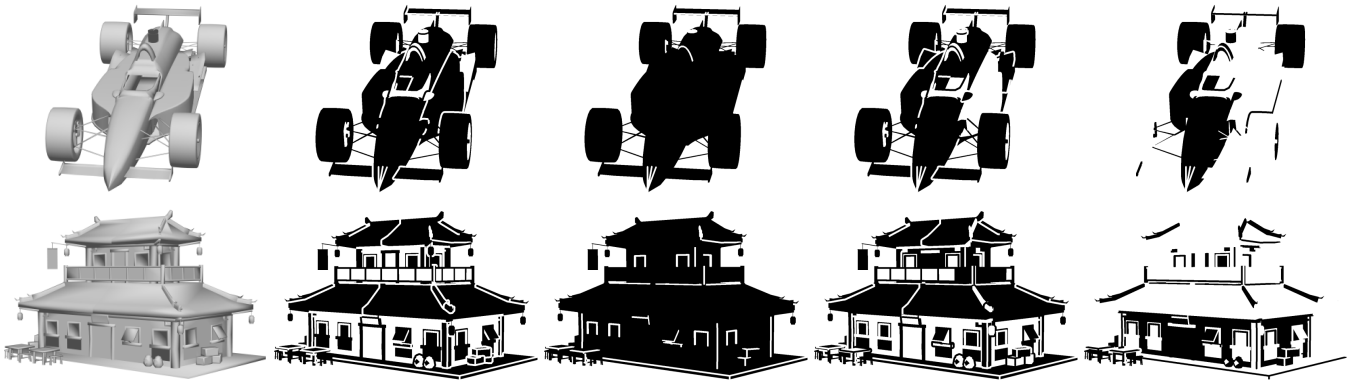


Fig. 14. Evaluation of energy terms: (left to right) input shape, results of including all energy terms, results of excluding neighbor term, results of excluding consistency term, results of excluding background term.

(i) the number of triangles and components of the input 3D shape due to congruence detection; and (ii) the number of patches that are involved in layout optimization. The timing statistics over all experiments exhibit an interactive performance, which ranges from less than 0.5 second for shapes with thousands of triangles to close to 1 second for the most complex shapes (rare cases). We also leverage the shader capability of modern GPU to further accelerate the generation of halo and feature line region in Section 6.2.

7.2 Effectiveness of energy terms

In the layout optimization, we devise three energy terms to realize our selected design principles. In this section, we evaluate the effectiveness of each energy term by vanishing the corresponding weighting coefficient (i.e., w_n, w_c, or w_b) during the optimization. A visual comparison is shown in Figure 14. It can be seen that without the neighbor term, the halo and hollow effects cannot be generated to highlight the spatial relation between patches. Excluding the consistency term can easily lead to an ambiguous layout with repetitive components in opposite colors, causing trouble for understanding correlations between patches. Lack of a background term is also problematic since boundary patches are often confused with the background of the same color. Optimization using all three energy terms ensures high-quality results.

7.3 Comparison with baseline methods

We compare our abstraction results with three baseline methods that also generates 2D binary representations from 3D shapes. Figure 15 (left) shows a comparison of different models. The first method (SC) is based on suggestive contour [DeCarlo et al. 2003], a 3D surface-based method that detects salient feature lines by analyzing surface curvature. We simply compute a 2D projection of the whole shape and assign black color to it, followed by overlaying white feature lines from suggestive contours. Since suggestive contour depends on surface curvature, it can be easily affected by the robustness of the curvature estimation method. Although it works very well for smooth surfaces with high mesh quality, man-made shapes with low quality measurements (e.g., irregular triangle shape, spiky geometry, seams and gaps) usually cause misplaced and scattered feature lines. The second method (DoG) is carried out by rendering the normals of an input 3D shape into an RGB normal map, based on which we apply the differential of Gaussian image filter to extract feature lines. Since it is an image-based method, the result is less sensitive to mesh quality, and thus more robust than suggestive contours. However, it has no halo and hollow features to support spatial and structural inference. Inconsistent normal orientation due to flipped faces can also cause problems. The third method (AMR) is artistic minimal rendering [Rosin and Lai 2010] with two tones, which takes the shaded image of the 3D shape as input. The result can be influenced by different shading effects due to lighting conditions. And missing geometric details can easily occur, especially at small scale when the image resolution is low, since the method is not intrinsic to shape geometry. We also conducted a user study to validate the effectiveness of our method over three baselines as follows.

7.4 User study

Since the ultimate judgement of the quality of 2D abstraction lies in human perception, we conducted a pilot user study via an anonymous online web survery with 100 participants to evaluate our approach over three baseline methods. The participants were mostly academic students with computer science background. We randomly picked 20 examples and presented to the participants the input 3D shape along with results generated from different methods in random order. The scale-awareness of our results was evaluated by generating abstractions at two different scales ($s = 0$ and $s = 1$). The results of other baseline methods are normalized to the same scale as ours. For surface-based method (SC), we resized the 2D projection of the 3D shape while keeping the width of the feature lines. For image domain methods (DoG and AMR), we generated results at different resolutions. During the trial, each participant was asked to select the best result according to one of the following three questions that respectively correspond to three visual metrics - 'conciseness', 'recognizability', and 'representativeness': Q-1: Which is the most *concise* result that can help distinguish between shapes? Q-2: Which is the most *recognizable* result at one glance? Q-3: Which is the most

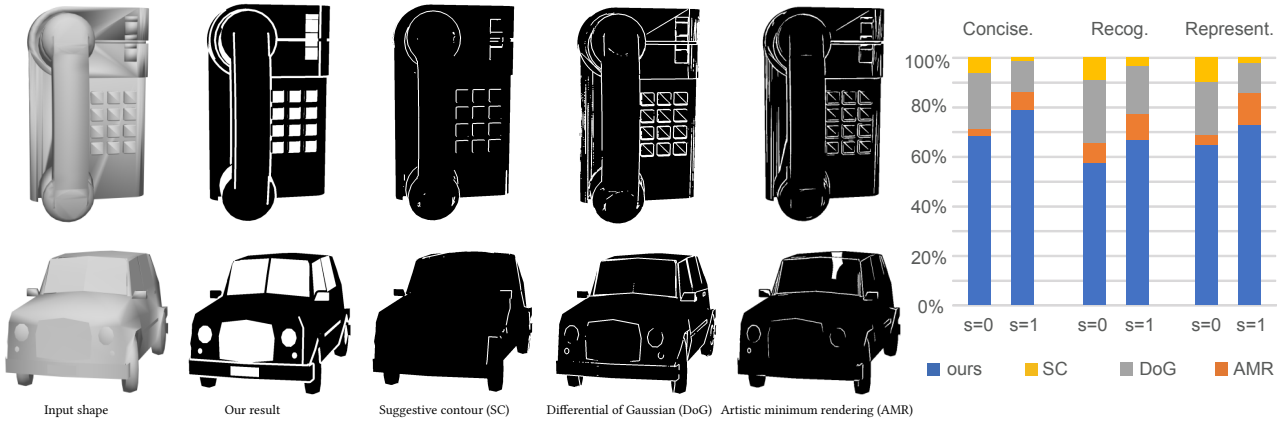


Fig. 15. Comparison with baseline methods. (Left) Visual comparison of results respectively generated by our method, suggestive contour (SC), differential of Gaussian (DoG), and artistic minimum rendering (AMR). (Right) User study statistics generated by evaluating quality of our results over baseline methods at two different scales.

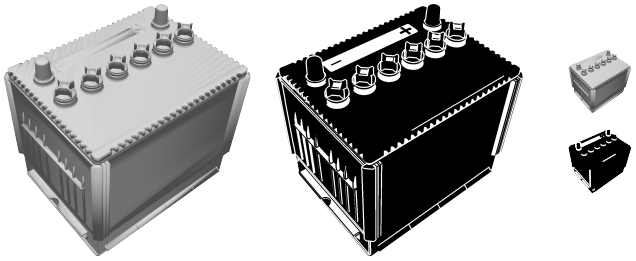


Fig. 16. An example involved in the user study that shows our result at small scale (right, $s = 0$) is less preferred than the one at large scale (middle, $s = 1$) due to the over-simplifcation of fine-grained features.

representative result with preferable appearance? Note that these questions were asked separately and properly distributed among participants to avoid bias.

The statistics of the user study are shown in Figure 15 (right). Our results receive the most positive responses in 'conciseness' as our goal is to generate concise 2D abstraction of 3D shapes. The preference of our method declines slightly for 'recognizability' and 'representativeness' due to the simplification of feature lines. The less preference of our result at small scale over large one is because our method retains only the most prominent features at small scale when comparing with other methods. Figure 16 shows an example of such a 'less-favorable' case. In general, our approach outperforms other baseline methods at both scales by striking a good trade-off between visual cluttering and over-simplification, resulting in a concise abstraction while preserving shape features.

8 APPLICATIONS

In this section, we discuss two applications that benefit from our black-and-white abstraction framework: flat vector graphics design and thumbnail image compression.

8.1 Flat vector graphics design

By fulfilling the design principles, our black-and-white abstraction is particularly suited to flat vector graphics design. As shown in Figure 17, through the use of Potrace [Selinger 2003] or Image Trace in Adobe Illustrator, our result can be effectively converted to its vectorized counterpart with less geometric complexity (e.g., number of anchors) than the other method. This facilitates the design process and serves as a good starting point for further edits by designers in the creation of icons [Bernstein and Li 2015], pictograms [Abdullah and Hüber 2006], and flat vector graphics in general. Note that besides geometric elements, the designers also rely on conceptual elements to better infer the functionality of the design. As shown in Figure 2, a thunder shape is added to the big chimney to indicate that it is used for generating electric power. Computational generation of such conceptual elements is beyond the scope of the present work.

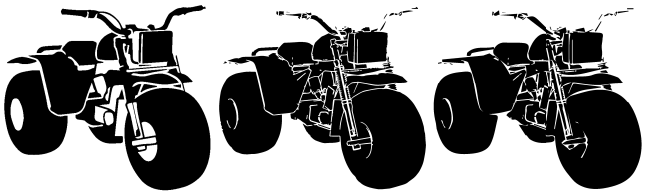


Fig. 17. Proposed black-and-white abstraction can be effectively converted to vector graphics format (left). The other two results are vectorized versions of the same DoG image with different geometric complexity defined by the amount of anchors. We can see that our result requires much less geometric components (i.e., 800 anchors) to represent the original shape when comparing to the middle one (2800 anchors) and is superior than the right one (1600 anchors) in perserving the prominent features.

8.2 Thumbnail image compression

In many online 3D model databases (e.g., https://free3d.com/), a shaded thumbnail image in full color is used for assisting the user to navigate the database. These thumbnail images often require relatively high resolution in image space and colour spaces. Although this might suit contexts such as categorized websites, the

network and memory cost of visualizing a large database would be too high. Our abstraction result can be encoded in a compressed 8-bit gray-scale image or a vector graphics format. As a result, network bandwidth and disk space can be largely saved (up to 91% according to our test) with moderate trade-off for details.

9 DISCUSSION

In this paper, we present a computational framework for the automatic generation of 2D black-and-white abstractions from 3D man-made shapes at multiple scales. We identify a set of relevant design principles to help formulate the problem, and devise a set of shape analysis and optimization algorithms conforming to these design principles. We demonstrate the proposed framework on a variety of man-made shapes, and evaluate the results through extensive comparison with baseline methods in a pilot user study. Experimental results show that our black-and-white abstraction presents a concise representation of the input shape, and well preserves its geometric and structural information in 3D at differing levels of detail. Further, we apply our abstraction results to two application scenarios to showcase its practical value.

Limitations. While our framework is robust to shapes with low mesh quality measurements, such as irregular sampling and triangulation, models with poor connectivity due to many duplicated mesh vertices and/or fragmented components can be problematic in the extraction of shape patches in 2D. Mesh repair algorithms and user interactions could be applied at a pre-processing stage to improve mesh connectivity. Another limitation is that our framework mainly focuses on man-made shapes with multiple components but not organic shapes with continuous geometry (e.g., the Stanford bunny). To adopt such shapes into our framework, existing techniques in geometry processing field such as feature extraction and mesh segmentation can be applied to analyze the geometric and structural properties.

Future work. To improve the semantics of abstractions at multiple scales, we would like to expand upon this work by exploring the high-level structural properties, such as symmetry and hierarchy. There also exists scope for further research into other flat design styles. For example, a popular design style that emphasizes patch boundaries over the patch itself. Some of the present patch-based abstraction principles (e.g., halo, hollow) can also be useful therein. Further, the current procedural optimization approach is devised in an unsupervised manner. How to automatically learn design patterns from existing examples would be an interesting avenue of reasearch. Last but not least, we plan to utilize our results for other applications that involve the 2D abstraction of 3D shapes, such as paper cutting - a characteristic 2D form that mimicks real objects in 3D. In this case, additional physical constraints such as connectivity between patches need to be considered.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their comments and suggestions. We also thank all the anonymous users for participating the user study. The work was supported in part by the Ministry of Science and Technology of Taiwan (106-3114-E-007-008 and 105-2221-E-007-104-MY2), and CAMERA, the RCUK Centre for the Analysis of Motion, Entertainment Research and Applications, EP/M023281/1.

REFERENCES

Rayan Abdullah and Roger Hüber. 2006. *Pictograms, Icons & Signs: A Guide to Information Graphics.* Thames and Hudson Ltd.

Arthur Appel, F. James Rohlf, and Arthur J. Stein. 1979. The Haloed Line Effect for Hidden Line Elimination.. In *Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '79).* 151–157.

Henk Bekker, Jos B.T.M. Roerdink, Tobias Isenberg, and Maarten H. Everts. 2009. Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), 1299–1306.

Gilbert Louis Bernstein and Wilmot Li. 2015. Lillicon: Using Transient Widgets to Create Scale Variations of Icons. *ACM Trans. Graph.* 34, 4 (2015), 144:1–144:11.

Stefan Bruckner and Eduard Gröller. 2007. Enhancing Depth-Perception with Flexible Volumetric Halos. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1344–1351.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum* 22, 3 (2003), 223–232.

Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. 2008. Where Do People Draw Lines? *ACM Trans. Graph.* 27, 3 (2008), 88:1–88:11.

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive Contours for Conveying Shape. *ACM Trans. Graph.* 22, 3 (2003), 848–855.

Elmar Eisemann, Sylvain Paris, and Frédo Durand. 2009. A Visibility Algorithm for Converting 3D Meshes into Editable 2D Vector Graphics. *ACM Trans. Graph.* 28, 3 (2009), 83:1–83:8.

Gershon Elber. 1995. Line illustrations is computer graphics. *The Visual Computer* 11, 6 (1995), 290–296.

Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. 2009. iWIRES: An Analyze-and-edit Approach to Shape Manipulation. *ACM Trans. Graph.* 28, 3 (2009), 33:1–33:10.

Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00).* 517–526.

Tilke Judd, Frédo Durand, and Edward Adelson. 2007. Apparent Ridges for Line Drawing. *ACM Trans. Graph.* 26, 3 (2007).

Alexander Kolliopoulos, Jack M. Wang, and Aaron Hertzmann. 2006. Segmentation-based 3D Artistic Rendering. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques (EGSR '06).* 361–370.

Hua Li and David Mould. 2015. Contrast-Enhanced Black and White Images. *Comput. Graph. Forum* 34, 7 (2015), 319–328.

Yiming Liu, Aseem Agarwala, Jingwan Lu, and Szymon Rusinkiewicz. 2016. Data-driven Iconification. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (Expresive '16).* 113–124.

Thomas Luft, Carsten Colditz, and Oliver Deussen. 2006. Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Trans. Graph.* 25, 3 (2006), 1206–1213.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J. Mitra. 2009. Abstraction of Man-made Shapes. *ACM Trans. Graph.* 28, 5 (2009), 137:1–137:10.

Tom Page. 2014. Skeuomorphism or Flat Design: Future Directions in Mobile Device User Interface UI Design Education. *Int. J. Mob. Learn. Organ.* 8, 2 (2014), 130–142.

Paul L. Rosin and Yu-Kun Lai. 2010. Towards Artistic Minimal Rendering. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering (NPAR '10).* 119–127.

Szymon Rusinkiewicz, Forrester Cole, Doug DeCarlo, and Adam Finkelstein. 2008. Line Drawings from 3D Models. In *ACM SIGGRAPH 2008 Classes (SIGGRAPH '08).* 39:1–39:356.

Peter Selinger. 2003. Potrace: a polygon-based tracing algorithm. In *In http://potrace.sourceforge.net.*

Ariel Shamir. 2008. A survey on Mesh Segmentation Techniques. *Computer Graphics Forum* 27, 6 (2008), 1539–1556.

Jie Xu and Craig S. Kaplan. 2008. Artistic Thresholding. In *Proceedings of the 6th International Symposium on Non-photorealistic Animation and Rendering (NPAR '08).* 39–47.

Yong-Liang Yang, Jun Wang, and Niloy J. Mitra. 2015. Reforming Shapes for Material-aware Fabrication. *Comput. Graph. Forum* 34, 5 (2015), 53–64.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. Exploring Artificial Intelligence in the New Millennium. Chapter Understanding Belief Propagation and Its Generalizations, 239–269.

Mehmet Ersin Yumer and Levent Burak Kara. 2012. Co-abstraction of Shape Collections. *ACM Trans. Graph.* 31, 6 (2012), 166:1–166:11.