Artistic Thresholding

Jie Xu Craig S. Kaplan[*]
David R. Cheriton School of Computer Science
University of Waterloo

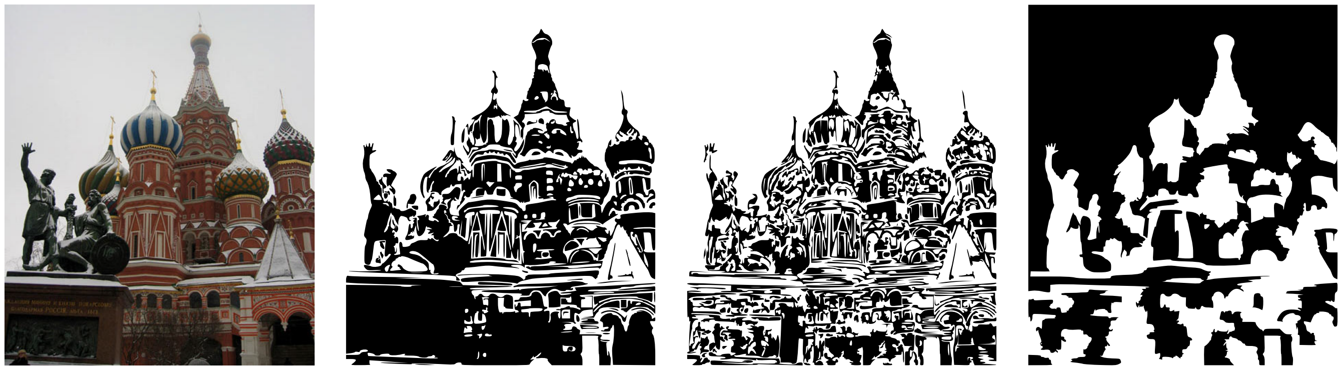

Figure 1: *A photograph of St. Basil's Cathedral in Moscow, rendered three different ways using artistic thresholding. Photograph used with permission from CNET Networks, Inc., Copyright 2008. All rights reserved.*

Abstract

We consider the problem of depicting continuous-tone images using only black and white. Traditional solutions to this problem include halftoning, which approximates tones, and line drawing, which approximates edges. We introduce "artistic thresholding" as a technique that attempts to depict forms in an image. We apply segmentation to a source image and construct a planar subdivision that captures segment connectivity. Our artistic thresholding algorithm is a combinatorial optimization over this graph. The optimization is controlled by parameters that can be tuned to achieve different artistic styles.

CR Categories: I.3.5 [Computer graphics]: Computational geometry and object modeling—Modeling packages J.5 [Arts and Humanities]—Fine arts G.2.2 [Discrete mathematics]: Graph theory—Graph algorithms

Keywords: Thresholding, Halftoning, Black and white, Segmentation, Region adjacency graph

1 Introduction

There are many algorithms for creating black and white representations of continuous-tone images. The majority of these algorithms are concerned with *halftoning*, the approximation of continuous tone using a distribution of black primitives on a white background [Ulichney 1987]. Halftoning was—and continues to be—a necessary means of overcoming limitations in output technologies. For the most part, computer displays have moved beyond the need for extensive halftoning, but print technology is still highly dependent on it. Halftoning has also found its way into non-photorealistic rendering research; examples include algorithms for approximating tone with stipples [Deussen et al. 2000; Secord 2002], screens [Ostromoukhov and Hersch 1995], pen strokes [Winkenbach and Salesin 1994], or the walls of a maze [Xu and Kaplan 2007].

However, halftoning need not be the only way create a black and white depiction of an image. There exists a great deal of traditional art in black and white that makes no attempt at halftoning. An immediate example is line drawing (see Figure 2(a)), which seeks to convey all information about form and texture from a few sparse visual cues embedded in contours. Aesthetically, we appreciate the "efficiency" of this encoding of a scene, and enjoy the experience of unpacking it.

Other traditional techniques use large regions of black and white instead of lines. Examples are shown in Figure 2. Some ink paintings use large black forms to carry all the salience of an image [Zheng 2000]. Papercutting, by its nature a binary medium, depicts an image by cutting holes in black paper [Xu et al. 2007]. Closely related is stencilling, as in the graffiti work of the enigmatic British artist Banksy. Many woodblock prints ignore tone (though masters of the medium may include hatching or stippling in their blocks). Contemporary examples include Frank Miller's *Sin City* comic and the computer animated film *Renaissance*. Both use pure black and white to reinforce the stark setting of their stories. The story "Exiles" in Neil Gaiman's *Sandman* series also makes heavy use of black and white, in a looser brush stroke style.

While the problem of producing line drawings from images has received attention in NPR research [Kang et al. 2007], depiction using large black and white regions remains underexplored. In this paper, we consider the problem of converting continous tone images into black and white representations inspired by examples like *Sin City* and *Sandman*.

The simplest way to convert any colour image to black and white

[*]e-mail: {jiexu,csk}@cgl.uwaterloo.ca

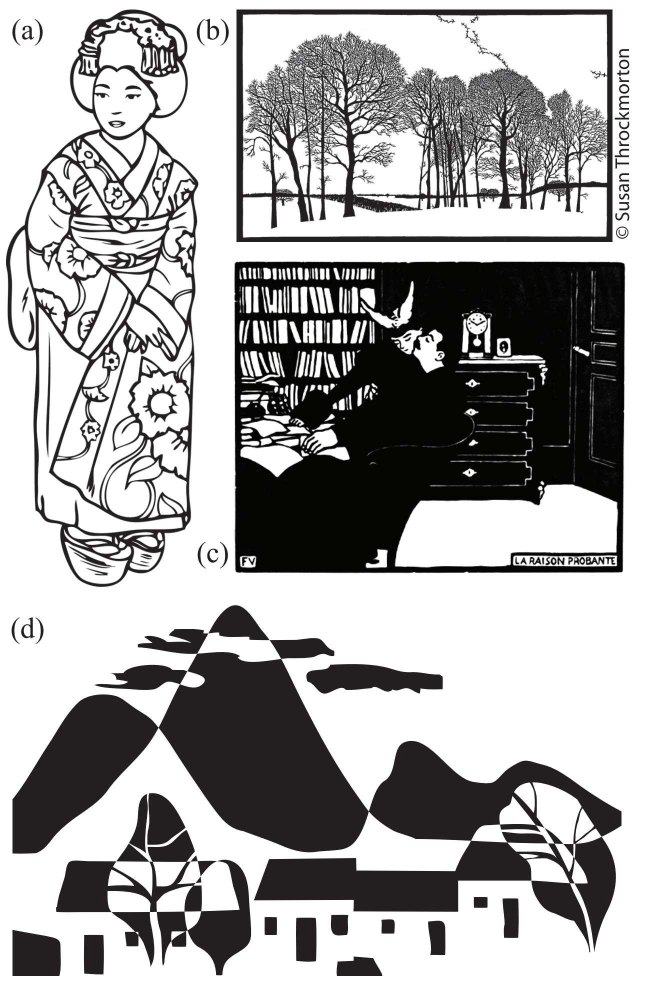

Figure 2: *Classic examples of black and white illustration without halftoning: a simple line drawing (a), a papercutting by Susan Throckmorton (b), Felix Vallotton's woodcut La Raison Probante (c), and a drawing by the authors (d), inspired by Chinese ink painting [Zheng 2000].*

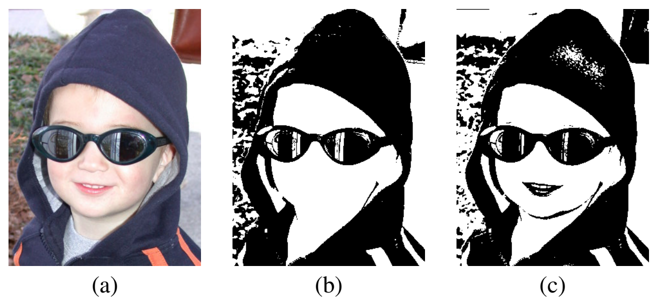

Figure 3: *Examples of a photograph (a) to which simple thresholding (b) and adaptive thresholding (c) have been applied.*

is undoubtedly *thresholding*. One chooses a cutoff value; a pixel is converted to black or white depending on whether its luminance is respectively below or above the treshold. Thresholding does not produce convincing abstractions of images (Figure 3). It does not respect features in the source image or recognize contrast between colours with similar luminance. Adaptive thresholding, shown in Figure 3(c), varies the cutoff value spatially depending on local tone. It can do a better job of extracting features, but suffers from many of the same problems.

One notable deficiency in thresholding is that an artist may choose to colour dark objects white or bright objects black in order to make them stand out from the background. These tone reversals are important for capturing all the details in a scene, and appear natural to the viewer. However, they could never be accounted for by simple thresholding. Our goal is to develop an algorithm that augments thresholding with a competing force that tries to preserve visibility of objects in the source image, and high-contrast edges in particular. We refer to any such algorithm as "artistic thresholding", or sometimes "wholetoning".

In this paper we present an optimization-based artistic thresholding algorithm. We construct a graph data structure based on segmentation of a source image (Section 2), and establish an energy function that measures the quality of different black and white colourings of the segments (Section 3). The algorithm searches for a black and white assignment that minimizes the energy function (Section 4). Our optimization is controlled via a collection of intuitive user-selected weights that can produce distinct results. The user can adjust weights in real time and observe the effect on the optimization process (Section 6). Sample images created using our system are presented in Figure 1.

1.1 Related work

Very little work in NPR is concerned with creating abstracted imagery via large regions of constant colour. DeCarlo and Santella [2002] used a hierarchical segmentation of an image to permit abstraction with spatially-varying level of detail. In their work, level of detail was guided by eye fixations, leading to a very natural distribution of detail around salient image features and high abstraction elsewhere. Similar results were achieved recently by Orzan et al. [2007]. They used edge information to guide the smoothing of features into large regions of constant colour. Wen et al. [2006] allowed the user to edit a segmentation interactively, and abstracted segment shapes into attractive coloured forms. All of these techniques focus on detail abstraction and preserve colours from the source image.

Gooch et al. [2004] used a perceptually motivated technique to convert photographs of human faces into black and white illustrations. They achieve a pleasing level of abstraction, though some amount of pixel-level detail survives the illustration process.

2 The region adjacency graph

The core of our artistic thresholding framework is a graph data structure that encodes the geometric and topological properties of small-scale features in an image. Our representation is similar to the "region adjacency graph" used occasionally in image segmentation and labeling algorithms [Horowitz and Pavlidis 1976], and we adopt that name for our purposes.

Given a source image, we use segmentation to subdivide it into regions. We use EDISON's synergistic image segmenter [Christoudias 2002], tuning it to oversegment. The small segments that are produced (related to the notion of "superpixels" in computer vision [Ren and Malik 2003]) support a wide range of abstracted results, while eliminating the distracting details associated with operations on individual pixels.

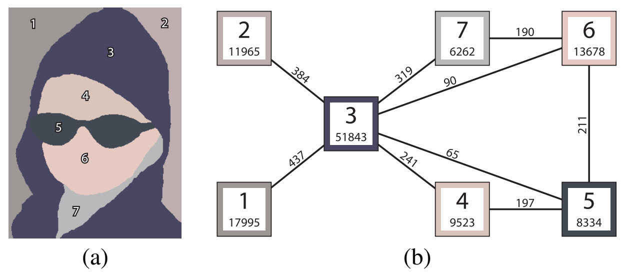

Figure 4: *A visualization of the region adjacency graph for a simple segmentation of the image in Figure 3(a). Every segment in (a) has a corresponding vertex in (b) with the same colour and label. Each vertex records the number of pixels in its segment. Each edge records the length of the shared boundary between its neighbouring segments.*

Let us assume that the source image has dimensions $W \times H$ (in pixels). Segmentation yields a set of N regions that we will denote using the indices $1,\ldots,N$. Each region is a (not necessarily connected) subset S_i of integer locations (x,y) in the image. We can think of the segmentation as a planar graph with a vertex for each region. Two vertices are connected by an edge when there are two pixels, one from each of the corresponding regions, that are adjacent horizontally or vertically.

We augment this purely topological description of the segments with information that will allow us to construct artistic thresholding algorithms. With each region S_i we associate c_i, the average colour of the pixels in S_i, and the area A_i, the number of pixels in S_i. For every pair of distinct indices i and j, we let $l_{i,j}$ denote the length of the shared boundary between S_i and S_j. This length is measured in terms of the number of adjacencies in the sense given above; if S_i and S_j are not connected by a graph edge, $l_{i,j}$ is zero (and hence an explicit graph data structure is not required). Figure 4 illustrates a simple segmentation, together with its region adjacency graph.

This simple representation of a segmentation enables a wide variety of non-photorealistic rendering algorithms. Here we envision artistic thresholding as a technique that assigns a value $b_i \in \{\text{black}, \text{white}\}$ to every region. We freely abuse this definition, thinking of b_i sometimes as a colour defined in the same colour space as c_i, sometimes as a boolean value (where black is true and white is false), and sometimes as real-valued a grey level (where black is 0 and white is 1). Given an assignment $\{b_1,\ldots,b_N\}$, We also define $B = \{i \in 1,\ldots N | b_i = \text{black}\}$, the current set of black regions.

We would also like to allow a designer to divide the image more coarsely into high-level features, and have those features interact with the low-level segmentation. If explicit features are provided, we simply divide any segmentation regions that cross feature boundaries. We will use these features in Section 3.3. In principle, these two levels could be extended into a full segmentation hierarchy, such as the one constructed by DeCarlo and Santella [2002]; we have found two levels adequate for our purposes.

3 Evaluating the quality of an assignment

A natural first approach to artistic thresholding is an energy-based optimization framework. There are 2^N possible assignments to the b_i; we search over that space of assignments for one that minimizes an objective function. In this section, we build the objective function, based on an intuitive sense of the quality of an assignment. We believe that this quality depends on several competing forces. The tradeoff in these forces is manifested by an objective function that is a weighted sum of individual measurements. The designer can adjust the weights to bias the search.

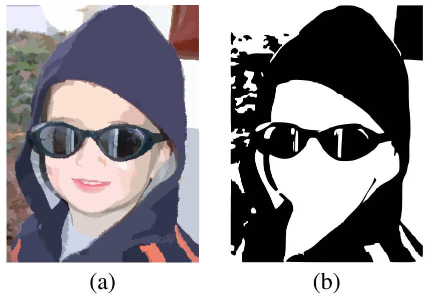

Figure 5: *An example demonstrating the use of C_{col} in isolation. The photograph from Figure 3 is shown segmented in (a). The optimized result is then shown in (b).*

3.1 Colour matching

One term in the objective function must measure how well the binary assignment to each region approximates the original colour of that region. We define C_{col}, which evaluates the overall difference between the binary assignment and the source image pixels:

$$C_{\text{col}} = \left(\sum_i A_i d(c_i, b_i) \right) / (WH) \; .$$

Here, $d(c_1, c_2)$ is a function that computes the difference between two colours, producing a result in the range $[0,1]$. We divide by the total image area to normalize the cost to the range $[0,1]$ (we will seek to normalize all cost functions in a similar way). When C_{col} is used in isolation, the optimal assignment can be found easily by setting b_i to be the thresholded luminance of c_i. For reference, Figure 5 shows an image produced by minimizing C_{col}. The result is already more attractive than the pixel-based thresholding of Figure 3: pixels are collected into segments, resulting in a less noisy image.

3.2 Area matching

In some cases a designer might wish to control the overall proportion of black used in the final image. Given a user-supplied target value $T_{\text{area}} \in [0,1]$, we define

$$C_{\text{area}} = \left| \left(\sum_{i \in B} A_i \right) / (WH) - T_{\text{area}} \right| \; .$$

Figures 6(a) and (d) demonstrate the effect of optimizing C_{area}.

3.3 Boundary contrast

The most interesting costs associated with an assignment measure the impact of contrast (or lack of it) between adjacent segments on

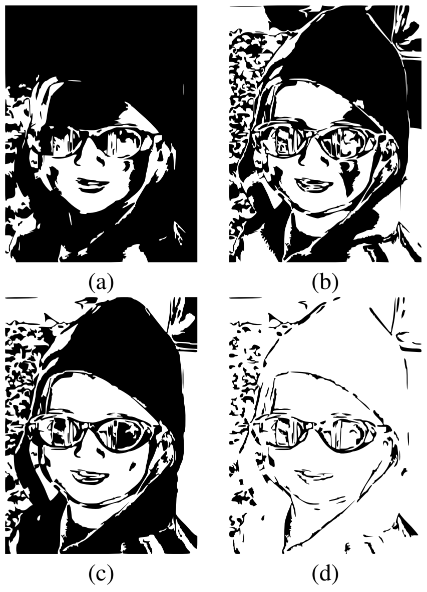
(a) (b) (c) (d)

Figure 6: *Demonstrations of the area and boundary costs. The image in (a) was optimized to be 80% black. Image (b) demonstrates the use of C_{alike} in isolation. In (c), we combine C_{alike} and C_{opp} with comparable values to achieve a balanced composition. In (d) we combine all three costs, aiming for 10% black.*

the quality of the final assignment. Boundaries between segments tend to contain the image's edges as a subset. We would like to preserve the visibility of those edges by ensuring that adjacent segments with contrasting colours are assigned opposite binary values. Conversely, similarly-coloured segments should be assigned identical binary values.

We divide the set E of edges of the segment graph into two groups E_{alike} and E_{opp}. An unordered pair (i, j) is in E_{alike} if $b_i = b_j$ (i.e., the segments are both black or both white); otherwise the edge is in E_{opp}. We can now define

$$C_{\text{alike}} = \left(\sum_{(i,j)\in E_{\text{alike}}} l_{i,j} d(c_i, c_j)^{1/5} \right) / \left(\sum_{(i,j)\in E} l_{i,j} \right) \text{ , and}$$

$$C_{\text{opp}} = \left(\sum_{(i,j)\in E_{\text{opp}}} l_{i,j} (1 - d(c_i, c_j)^{1/5}) \right) / \left(\sum_{(i,j)\in E} l_{i,j} \right) .$$

The cost C_{alike} measures how effectively the assignment uses similar binary values for similarly-coloured adjacent segments. On its own, this cost is theoretically minimized by letting E_{alike} be empty, corresponding to a 2-colouring of the region adjacency graph. In practice, we obtain very busy assignments that approximate 2-colourings, as shown in Figure 6(b). Conversely, minimizing C_{opp} comes from placing every edge in E_{alike}, which can be achieved by assigning every segment the same binary value. When these two costs are given comparable weights, they cooperate to achieve a balanced use of contrast. Most interestingly, optimizing these costs can lead to a segment being assigned a binary value that contradicts its luminance, if that assignment improves the overall depiction of shapes via edges.

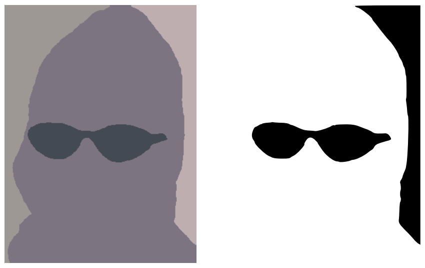
(a) (b)

Figure 7: *An example of minimizing the group homogeneity measure C_{group} in (b), based on the user-supplied features shown in (a).*

Most of the time, we expect that adjacent segments will have similar colours because of spatial coherence, which suppresses the effectiveness of these measurements. Empirically, we found that taking the colour differences to the power of $1/5$ magnifies small differences and evens out the cost functions. Note also that constrasts are multipled by boundary lengths, and normalized by the total length of boundaries in the image.

When high-level features are provided by the designer, we modify C_{alike} to further penalize feature edges that are not depicted via segment contrast:

$$C_{\text{alike}} = \left(\sum_{(i,j)\in E_{\text{alike}}} l_{i,j} d(c_i, c_j)^{1/5} p_{i,j} \right) / \left(\sum_{(i,j)\in E} l_{i,j} p_{i,j} \right) .$$

Here, $p_{i,j} = 1$ when segments i and j belong to the same high-level feature, and some real number greater than 1 when they do not.

3.4 Feature homogeneity

When the image is divided explicitly into high-level features, we may sometimes wish to assign binary values as homogenously as possible within each of those features. Assume that there are M features. For a given assignment, let u_k and v_k denote the number of black and white segments within feature k. Then we let

$$C_{\text{group}} = \left[\sum_{k=1}^{M} \left(1 - \frac{|u_k - v_k|}{u_k + v_k} \right) \right] / M .$$

This cost is minimized by assigning all segments within the same high-level feature identical binary values. On its own, C_{group} can produce very attractive images, but deceptively so: the quality arises almost entirely from the salience of the user-provided features (see Figure 7). This cost can be used profitably in conjunction with the others, as a way to produce a "calmer" assignment.

3.5 Total cost

We must define a single energy function that can be minimized via optimization. We adopt an approach that has been seen before in NPR [Hertzmann 2001; Kim and Pellacini 2002], computing the overall energy as a weighted sum of individual terms:

$$C_{\text{total}} = \frac{w_{\text{col}} C_{\text{col}} + w_{\text{alike}} C_{\text{alike}} + w_{\text{opp}} C_{\text{opp}} + w_{\text{group}} C_{\text{group}}}{w_{\text{col}} + w_{\text{alike}} + w_{\text{opp}} + w_{\text{group}}} .$$

The weights are non-negative real numbers, not all zero. As with the individual costs, we divide by the sum of the weights to normalize the quality measurement. This normalization becomes especially important in the next section, where C_{total} is used as the objective function in a simulated annealing optimization. If the overall scale of the weights were allowed to drift, the resulting cost function would behave differently with respect to the optimization's cooling schedule.

4 Optimization

The goal of artistic thresholding is to find an assignment $\{b_1, \ldots, b_N\}$ that minimizes the total cost C_{total} defined in the previous section. Except for very simple cases, it is too expensive to search over all 2^N assignments for the optimum. Instead, we use an optimization framework based on simulated annealing [Press et al. 1992, Chapter 10]. Simulated annealing is a robust, general purpose optimization algorithm. It is particularly useful when the geometry of the configuration space is difficult to characterize.

We initialize the b_i randomly to black or white, and keep track of the current best assignment. The optimizer repeatedly constructs perturbations of this assignment and tests their costs. If a perturbed assignment has a lower cost, it is accepted unconditionally as the new best answer; if the cost is higher, it is accepted with a small probability, temporarily degrading the solution in the hopes of avoiding local minima in the configuration space. We use a cooling schedule to decrease the acceptance probability at an exponential rate. Optimization continues until a specified number of iterations have passed with no changes to any of the b_i, at which point we report the best assignment found. A similar approach was used by Agrawala and Stolte [2001] to render route maps.

The question remains of how to construct perturbations of the current assignment. A natural approach is to flip a random b_i and check if that improves the overall assignment. However, in practice this approach is too local. There may be situations where overall cost can be decreased by changing the values of several nearby segments in tandem, even though the cost goes up if any one of them is flipped in isolation.

We mitigate this problem by operating on subgraphs instead of individual segments. Given a vertex in the region adjacency graph, we generate a connected subgraph contaning that vertex. The number of vertices in the subgraph can be controlled by the user; we have found that three to five vertices provide a good balance between performance and quality. We construct new assignments for all possible combinations of binary values within this subgraph, and emit the lowest-energy combination as the chosen perturbation. (We also sometimes choose one of the other combinations, in order to inject additional randomness into the search.)

Note that we do not need to recompute C_{total} from scratch every time some b_i changes. Most of the energy terms depend only on the relationship between a segment and its immediate neighbours. When one segment changes colour, we can compute the effect of this change on those cost terms and add the difference to the current cost. We further exploit this fact when testing all combinations of binary assignments within a subgraph. We iterate over the combinations in an order given by a Gray Code [Weisstein 2008], so that the combinations can be tested by flipping a single b_i at a time.

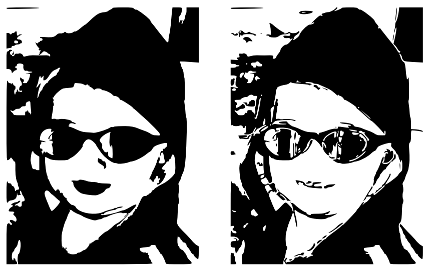

Figure 8: *Demonstrations of the postprocessing operations discussed in Section 5. We smooth out isolated pockets of black and white in (a) using graph-based morphological operators. In (b), we superimpose edges between high-contrast segments that were assigned the same binary value.*

5 Postprocessing

Once the optimization is complete, we can simply render every pixel in S_i using colour b_i. We have also experimented with several postprocessing operations that can improve the overall quality of our results.

The designer may optionally apply the standard OPEN and CLOSE morphological operators, adapted here from the pixel grid to our region adjacency graph. These operators can help to eliminate small, isolated pockets of contrasting colour, smoothing out noise and increasing the level of abstraction in the thresholded image. For each segment, we control which of its neighbours participate in the computation of OPEN and CLOSE. A user-controllable area threshold prevents small adjacent segments from having an influence. An edge threshold excludes adjacent segments that are too different in colour, helping to preserve image edges.

While we wish to draw a clear distinction between artistic thresholding and previous work on line drawing, a few well-chosen lines can enhance some of our results. We search over all pairs of adjacent segments. When two segments are given the same binary assignment but have a colour difference beyond a user-selected threshold, we draw the boundary between them as an edge. These edges can help to reinforce object boundaries in the source image that were missed by the optimizer. In practice, we find that many images are comprehensible without these edges.

Figure 8 demonstrates the application of the morphological operators and edges.

Finally, we vectorize the binary image to produce a high-quality scalable result. We have obtained good results using the Potrace library [Selinger 2003]. All results presented here have been vectorized in this way.

6 Implementation

We have constructed an interactive software implementation of artistic thresholding. The interface is designed to support a continuous, dynamic interaction between the designer and the optimization process. The current best binary assignment is displayed during optimization. The designer can modify the weights in the cost function, immediately affecting the algorithm. In order to allow weight changes to have a significant effect on the assignment, each adjustment causes a small increase in annealing temperature. The

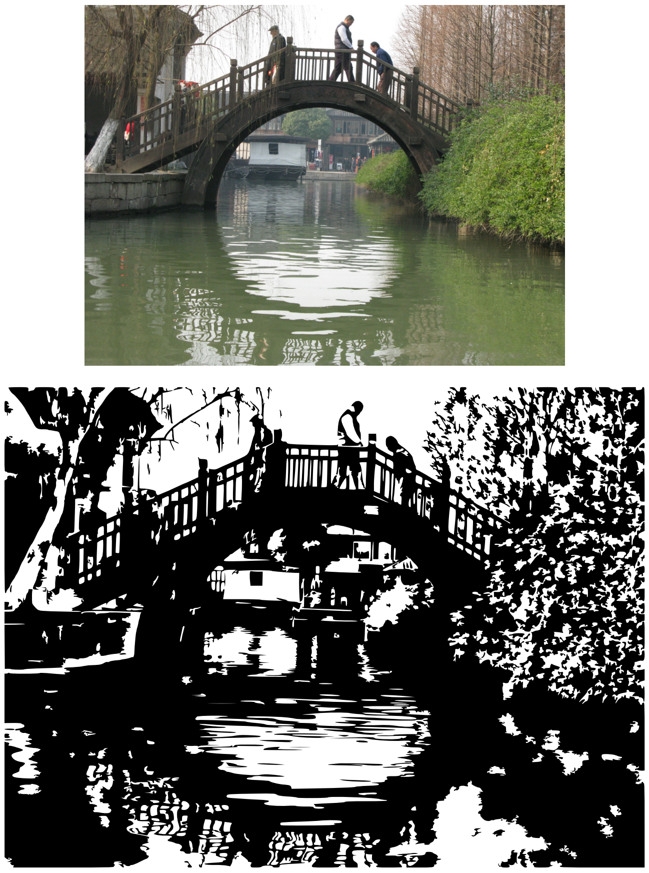

Figure 10: *A wholetoned drawing based on a photograph of a bridge in Wuzhen. Photograph by Carsten Ullrich.*

designer can also fix binary values for individual segments when desired; such values are held constant during optimization.

In our implementation, we set the initial annnealing temperature to 100. Every time the temperature changes, we iterate over the vertices in the region adjacency graph. For each vertex, we construct a random connected subgraph containing that vertex and optimize it as described in Section 4. After visiting all vertices, we set the temperature to 99% of its current value. When the current best assignment has survived 200 temperature changes, we stop the optimization process.

7 Results and discussion

Figures 9 and 10 show some results produced using our artistic thresholding algorithm. We have found our implementation to be successful on a wide variety of source images. A typical result requires only a few minutes of processing (disregarding the segmentation step) and very little user interaction. In most cases, it is not necessary to fix the binary values for any segments manually.

In Figure 11, we compare our technique to drawings that could be produced with more traditional image processing methods. We have found that a combination of bilateral filtering, blurring and thresholding can produce attractive results, as in Figure 11(c). However, this method misses some of the edges and details brought out by artistic thresholding. Also, it is ultimately tied to traditional

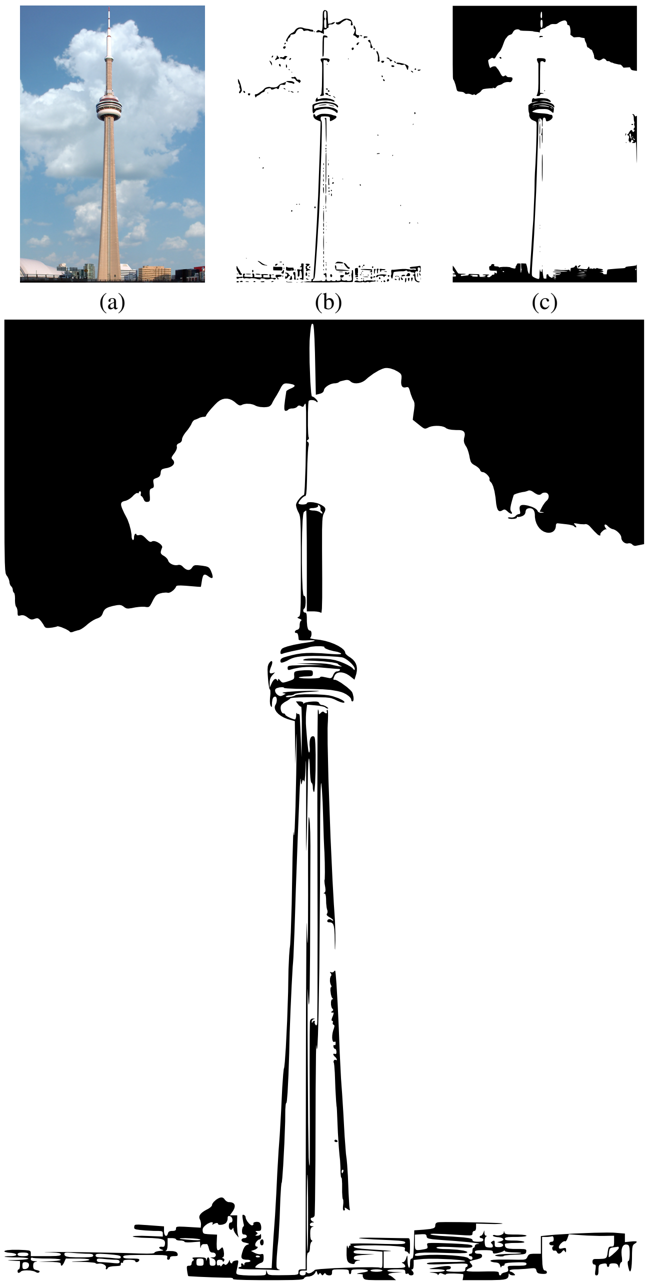

Figure 11: *A wholetoned drawing based on a photograph of the CN Tower, together with related drawings produced via traditional image processing approaches and vectorization. The original photograph is shown in (a). Blurring and adaptive thresholding produces the drawing in (b). In (c), we apply bilateral filtering, blurring and thresholding, leading to a result with some similarities to ours.*

luminance-based thresholding, and could never produce more abstract results like the rightmost image in Figure 1.

The algorithm produces interesting results across a wide range of weights, and the responsive interface invites the designer to manipulate the weights interactively in a dialogue with the optimizer. Our implementation is an engaging and effective tool for creative exploration of this artistic style.

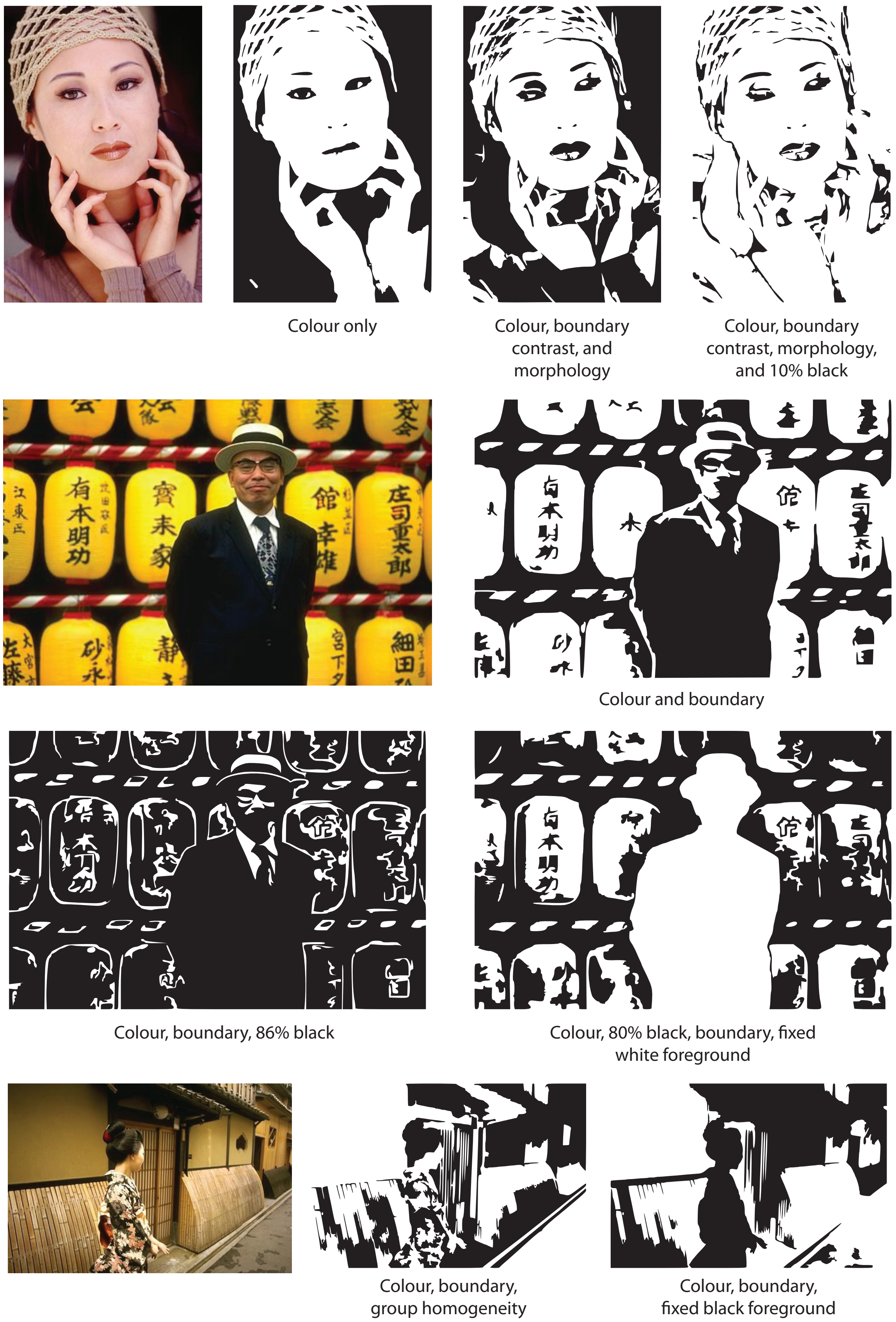
Colour only
Colour, boundary contrast, and morphology
Colour, boundary contrast, morphology, and 10% black
Colour and boundary
Colour, boundary, 86% black
Colour, 80% black, boundary, fixed white foreground
Colour, boundary, group homogeneity
Colour, boundary, fixed black foreground

Figure 9: *Sample results produced using our artistic thresholding algorithm. We summarize the settings used to produce each result by listing the non-zero weights and target area (if applicable).*

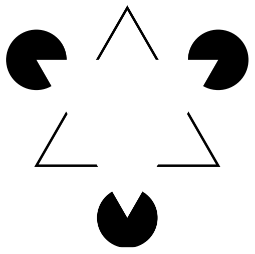

Figure 12: *The Kanizsa triangle, an example of illusory contours.*

As mentioned in Section 2, we support manually specified high-level features. They can be provided by tracing salient boundaries in a drawing program. In some cases, we can make use of pre-existing feature data. The images in Figure 9 all come from the Berkeley Segmentation Dataset [Martin et al. 2001], and are accompanied by human feature identification data. In Figure 9, high-level features were used for results that have fixed foregrounds or that made use of C_{group}.

Despite the seemingly unpredictable nature of simulated annealing, we found that our implementation was quite stable: given a source image and its region adjacency graph, re-running the optimization with the same weights produced nearly identical results. Differences were minor and did not affect the visual character of the thresholded images. Quantitatively, the images differed from each other over only a few percent of their pixels.

As with many algorithms in graphics and vision, we begin with a finely segmented image and treat segments as atomic entities. Conceivably our algorithm could be modified to operate directly on the planar graph induced by image pixels. However, segments provide a basic level of image abstraction and noise reduction that would be difficult to achieve at the pixel level. A graph of individual pixels would probably produce less attractive results in the presence of C_{alike} and C_{opp}. Looked at another way, we suspect that any sufficiently robust artistic thresholding algorithm would necessarily include a step that is equivalent to segmentation. We prefer to trust in the high quality of published segmentation algorithms.

8 Future work

There is an interesting relationship to be explored between artistic thresholding and line drawing. This paper takes a first step by drawing missed edges explicitly after optimization is complete. We would like to investigate how an understanding of edges can be incorporated directly into the optimization. We might extend our binary assignment to include a black or white value for each edge in the region adjacency graph. We would then have to modify the cost functions to evaluate assignment quality in the presence or absence of these edges. At a minimum, we would want a term that simply minimizes the total length of all edges drawn, in order to encourage the binary segments to carry most of the salience in the result.

Even when adjacent segments with contrasting colours are given the same binary value, it need not follow that their shared boundary must be drawn. Our perceptual systems are wired to infer portions of object contours that are obsured by lack of contrast. Perhaps the most famous demonstration of this effect is the "Kanizsa triangle", shown in Figure 12. Missing contours are inferred so strongly that we actually see an edge where none is present. These "illusory" or "subjective" contours have been demonstrated in many contexts [Parks 1984], and are usually attributed to a Gestaltist explanation: a triangle occluding three circles is the simplest interpretation of the image. This effect is used in practice, as in the boundaries between the hosues in Figure 2(d). We would like to explore the problem of how well image edges can be represented even when partially invisible, taking inference into account. This effect is enabled uniquely by artistic thresholding: line art drawings do not exhibit illusory contours to the same degree.

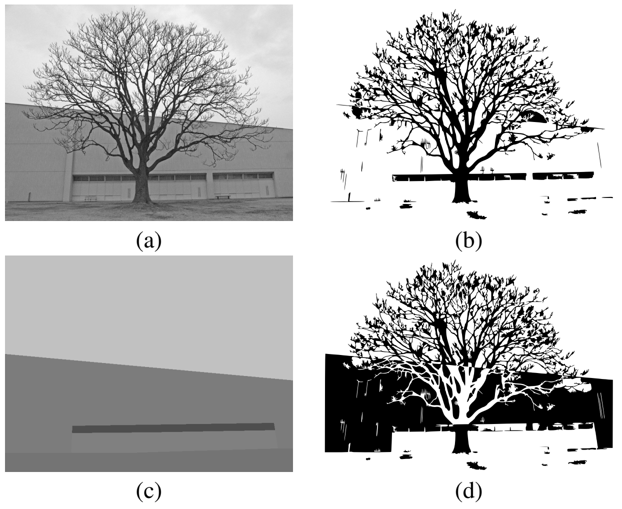

Figure 13: *An example of how an "exclusive or" effect may be achieved via carefully constructed user features. The tree in (a) yields a single large segment, and therefore cannot contrast with both the building and the sky in (b). In (c) we force a feature for the building to cross through the tree. The boundary of this feature splits the tree into two segments that can be given opposite colours. Photograph by atemzeit, used with permission.*

Another effect we cannot easily achieve is the use of "exclusive-or" to depict foreground objects on top of a background that varies between black and white. We have encountered many examples where thin foreground objects such as trees or table legs are drawn in white on black segments and black on white segments (see Figure 2(d)). The object is then visible everywhere, and can be perceived as a cohesive whole even though it varies between black and white. Because we start by segmenting a flat image, it is difficult to discover opportunities to use this effect. We can contrive to achieve it by specifying high-level features that deliberately cross through foreground objects. Those objects are then broken into multiple segments, allowing the optimizer to make independent assignment choices in the sub-segments. Figure 13 shows an example. An alternative would be to work from a 3D scene or $2\frac{1}{2}$D layers, in which case we can decide on an XOR-like compositing rule when the layers are flattened. We used a similar approach in our earlier work on papercutting [Xu et al. 2007].

Given the compelling appearance of the film *Renaissance*, it is natural to consider the application of artistic thresholding to video. If segments could be tracked through a video sequence (perhaps using a keyframe approach [Agarwala et al. 2004]), we speculate that the objective function could be modified to evaluate the cumulative cost of an assignment across all frames. Alternatively, a technique such as Video Tooning [Wang et al. 2004] could be used to find a low-cost assignment directly on the space-time volume equivalent of the region adjacency graph.

Finally, it would be interesting to augment the objective function by taking into account further measures of salience in the source image. Salience could be painted by hand or derived from eye-tracking data [DeCarlo and Santella 2002]. It might also be computed automatically; Collomosse and Hall [2005] used an automated salience algorithm to drive a genetic algorithm for painterly rendering. Salience would probably be used to annotate edges in the region adjacency graph with an importance value, which would then affect boundary contrast costs.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Thanks also to Bill Cowan, Doug DeCarlo, Aaron Hertzmann, Neil Kaplan, and David Mould for assistance and productive discussions. Thanks to Susan Throckmorton, Carsten Ullrich, and atemzeit for permission to use the images in Figure 2(b), Figure 10, and Figure 13. This research was supported by NSERC.

References

AGARWALA, A., HERTZMANN, A., SALESIN, D. H., AND SEITZ, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 584–591.

AGRAWALA, M., AND STOLTE, C. 2001. Rendering effective route maps: improving usability through generalization. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 241–249.

CHRISTOUDIAS, C. M. 2002. Synergism in low level vision. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 4*, IEEE Computer Society, Washington, DC, USA, 40150. EDISON code available at `http://www.caip.rutgers.edu/riul/research/code/EDISON/`.

COLLOMOSSE, J. P., AND HALL, P. M. 2005. *Genetic paint: a search for salient paintings*, vol. 3449 of *Lecture Notes in Computer Science (Proc. EvoMUSART)*. Springer-Verlag, 437–447.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. *ACM Trans. Graph. 21*, 3, 769–776.

DEUSSEN, O., HILLER, S., VAN OVERVELD, C., AND STROTHOTTE, T. 2000. Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3, 41–50.

GOOCH, B., REINHARD, E., AND GOOCH, A. 2004. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph. 23*, 1, 27–44.

HERTZMANN, A. 2001. Paint by relaxation. In *CGI '01: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 47.

HOROWITZ, S. L., AND PAVLIDIS, T. 1976. Picture segmentation by a tree traversal algorithm. *J. ACM 23*, 2, 368–388.

KANG, H., LEE, S., AND CHUI, C. K. 2007. Coherent line drawing. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 43–50.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 657–664.

MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, 416–423.

ORZAN, A., BOUSSEAU, A., BARLA, P., AND THOLLOT, J. 2007. Structure-preserving manipulation of photographs. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 103–110.

OSTROMOUKHOV, V., AND HERSCH, R. D. 1995. Artistic screening. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 219–228.

PARKS, T. E. 1984. Illusory figures: A (mostly) atheoretical review. *Psychological Bulletin 95*, 2, 282–300.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, second ed. Cambridge University Press. ISBN 0-521-43108-5. Held in Cambridge.

REN, X., AND MALIK, J. 2003. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, vol. 1, 10–17.

SECORD, A. 2002. Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 37–43.

SELINGER, P. 2003. Potrace: a polygon-based tracing algorithm. `http://potrace.sourceforge.net/potrace.pdf`, September.

ULICHNEY, R. 1987. *Digital Halftoning*. The MIT Press.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. F. 2004. Video tooning. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 574–583.

WEISSTEIN, E. W., 2008. Gray code. From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/GrayCode.html`.

WEN, F., LUAN, Q., LIANG, L., XU, Y.-Q., AND SHUM, H.-Y. 2006. Color sketch generation. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 47–54.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 91–100.

XU, J., AND KAPLAN, C. S. 2007. Image-guided maze construction. *ACM Trans. Graph. 26*, 3, 29.

XU, J., KAPLAN, C. S., AND MI, X. 2007. Computer-generated papercutting. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG'07)*, IEEE Computer Society, Washington, DC, USA, 343–350.

ZHENG, C. 2000. *New Decorative Landscape Pattern Design*, first ed. Zhejiang People's Art Publishing House. ISBN 7-5340-1002-0.