Binary Shading using Appearance and Geometry

Bert Buchholz[+] Tamy Boubekeur[+] Doug DeCarlo[†,*] Marc Alexa[*]

[+] Telecom ParisTech - CNRS *LTCI* [*] TU Berlin [†] Rutgers University

Abstract

In the style of binary shading, shape and illumination are depicted using two colors, typically black and white, that form coherent lines and regions in the image. We formulate the problem of assigning colors in the rendered image as an energy minimization, computed using graph cut on the image grid. The terms of this energy come from two sources: appearance (shading) and geometry (depth and curvature). Our contributions are in the use of geometric information in determining colors, and how this information is incorporated into a graph cut approach. This optimization yields boundaries between black and white regions that tend towards being shorter and to run along geometric features like creases. We show a range of results, and demonstrate that this approach produces more coherent images than simpler approaches that make local decisions when assigning colors, or that do not use geometry.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: Color, shading, shadowing, and texture

back

1. Introduction

Depiction of an object requires at least one foreground and one background color. Oftentimes it is desirable or necessary to restrict the depiction to just two colors; examples are artistic black-and-white illustrations and images generated from stencils. In technical terms, such images are commonly referred to as bilevel or binary images. We consider the problem of generating binary images by means of rendering a 3D shape.

There are several ways of conveying tone using just two colors, notably half-toning [FS76, Uli87], which can also be used for the creation of artistic images [OH95, OH99]. We are rather concerned with depictions that mostly assign black color to dark regions and white color to bright regions of the image, inspired by the popular black and white style of several graphic novels, and recent graffiti art.

The black and white illustration by comic book artist Alex Ross displayed in figure 1 is a quintessential example of this style. In analyzing drawings such as these, we find that the composition combines aspects of tone reproduction with rendering of geometric features. As expected, large black or white regions in the drawing correspond to low and high levels of illumination, respectively. Furthermore, high local contrast in illumination can override the absolute illumination levels. For instance, a coherent area which is darker than surrounding regions might be depicted in black, even when the illumination levels are all fairly high. Finer details are often depicted this way, such as the musculature on the arms and torso in the image on the right. At the same time, the particular boundaries between black and white are selected to effectively convey the shape. These boundaries might run along surface features like ridges and valleys, such as along the ridge of the nose or the boundary of the brow. Or they might run along occluding contours (discontinuities in depth), such as on the cape. They also appear to be as short as possible, given these constraints. Thin lines, which are often haloed, may also be used to further delineate parts of the shape. In many situations, including such lines is crucial—the bottom of the cape is drawn in white so it is easily separated from the black background, but the lines along the occlusions on the bottom of the cape are what clearly convey its wing-like nature. In other work, the effects of highlights or shadows can be largely discounted when artists make these decisions.

Figure 1: *Hand drawn illustration of the* Batman *comic (DC) by Alex Ross. Note that assignment of white and black color combines aspects of tone reproduction with rendering of geometric features (see main text).*

In terms of rendering, a pixel may be black or white because of the illumination of the shape, or because the contrast between neighboring pixels of different color helps to better convey the geometric structure of the shape, regardless of the shading. These goals are often contradictory, meaning that taking into account only *appearance* or only *geometry* is insufficient. Here, appearance refers to various terms that are commonly used in realistic shading (section 4), while geometry refers to geometric properties of the shape relative to screen coordinates (section 5). Note that a binary image could be generated from the appearance of the shape alone by thresholding [KZ93, LL98, MG08, XK08] or from geometry using only toon-shaders [Dec96, BTM06].

An artist would need to make a range of decisions to arrive at a successful depiction using only black and white, balancing the two contradictory goals. Thus, we model the process of assigning black or white to the pixels as an *optimization* problem that takes into account both appearance and geometry.

The optimization problem to be solved is essentially a segmentation of the image into black and white pixels. As in other recent approaches, we model the image as an undirected graph and determine the solution using graph cut energy minimization [BVZ01]. Although the form of the function being optimized is restricted, these restrictions allow for efficient algorithms for its solution. (Note that because of its connection to the maximum flow problem, the typical terminology used for parts of the graph are related to flows. Here, instead, we use terms related to colors and pixels, and note the flow terms parenthetically.) In this framework, there is a node in the graph for each of the pixels in the image. There are also two additional nodes in the graph (terminal nodes) that represent the color white (the source node) and the color black (the sink node). The edges in the graph are either connections between neighboring pixels, connections from a pixel to the white node, or connections from a pixel to the black node. Each edge has a non-negative weight (capacity) that signals the strength of the connection. Initially, edges connect pixels to each of their neighbors, and to both the white and black nodes. Any cut in the graph that separates the white and black nodes (the source from the sink) is a possible depiction in black and white. An appropriate depiction in a binary style is determined by computing the minimal cut—the globally minimal set of edges (by summing their weights) that when removed separates the white from the black pixels. An overview of this approach is diagrammed in figure 2.

The main contributions of our work are the use of geometric information in this style of depiction, and further, modeling the separate influence of appearance and geometry elegantly as weights on the two different types of edges in the graph:

- appearance defines the weights of edges connecting pixels to the white and black nodes, and specifies the tendency for pixels to be black or white,
- geometry defines the weights of edges connecting neighboring nodes in the image, and specifies the tendency for pixels to be different colors because of geometric structures.

The primary challenge here is to assign weights to edges so that the resulting minimal cut represents a coherent binary image. After an overview in section 3, we describe how weights are assigned using appearance (section 4) and geometry (section 5).

We show a range of results in section 6. In particular, we argue that with our approach it is possible to generate results that cannot be produced with simpler approaches such as thresholding or using local decisions. Following this, we discuss the effects of using our method in animation in section 7.

2. Background and previous work

We view the generation of binary images as a segmentation problem, i.e. assigning regions in the image to foreground or background. Several instances of this problem have lately been modeled as a max-flow/min-cut problem—the so called graph cut approach [BVZ01, BK04, RKB04].

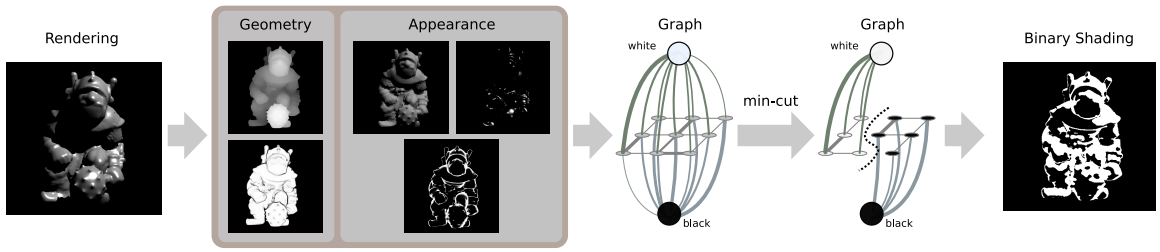


Figure 2: *Several sources of information about the appearance and geometry of the scene influence the edge weights in a graph representing the image. The black and white assignment is computed by finding the minimum cut in the graph. Thus, the output respects geometric features, and not only appearance.*

Our goal is to create large regions of black and white pixels that convey the essential features of the shape. Perhaps the simplest such operation on images that works towards this goal is thresholding. For most inputs, a global threshold (see Lee et al. [LCP90] for an overview) will not preserve important features, which is why thresholds are typically adapted locally [KI86, YB89].

Recently, algorithms for the binarization of photographs have been considered for the purpose of creating artistic or abstract images. In particular, Mould and Grant [MG08] as well as Xu and Kaplan [XK08] are considering the same problem we do, albeit starting from an image and not a 3D scene. In both approaches, desirable properties of a binarization are described using an energy function that is a weighted sum of terms. This energy function is optimized to yield the image. While the approaches differ in how the energies are defined, their general strategy is similar: there are terms for encouraging similarity to the input (in terms of absolute and local contrasts), preservation of image features (i.e. as region boundaries), and smoothness of regions in the result. Mould and Grant optimize the energy by casting it as a min-cut problem while Xu and Kaplan use simulated annealing [KGV83].

A closely related problem to binary shading is the creation of a papercutting [XKM07] or stencil [BRO08]. In this case, regions of black and white are replaced with where material is cut out, and where it is retained. However, there is an additional constraint, which has been the main focus of the research: ensuring the result is a single connected piece of material. The binarization step is performed using simple thresholding, and the pieces are connected with lines that minimize particular cost measures.

Performing binarization in a 3D rendering pipeline has also been studied in the context of real-time stylized shading. Spindler et al. [SDM06] combine a simple toon-shader with haloed lines. Stylized forms of shading that are exaggerated or expressive [RBD06, VBGS08, VPB*09] combine enhancements for increasing contrast locally with adjustments of illumination based on the surface geometry. However, these decisions are made locally—each point is shaded independently, and thresholded. As a consequence, boundaries between regions will not necessarily run along surface features, and thus lack an element of coherence that our approach provides (see section 6 for comparisons).

Artistic black and white rendering is quite well studied for other styles than the one we are considering here. In particular, drawing feature lines in black or white [DFRS03, LMLH07, JDA07] results in informative renderings, which might be combined with toon-shaders to produce a different black and white rendering style [DR07].

3. Overview

We model the problem of assigning black or white to each pixel in the image as a graph cut: each pixel is a node in a graph; all pixels are connected to two nodes that separately represent white and black (the terminal nodes—source and sink). In addition, each node is connected to a set of nodes representing neighbors in the image. We define the weights (capacities) of the edges in this graph using the full data available from the 3D scene, namely appearance and geometric surface properties. Finding a cut that disconnects the white from the black node effectively assigns the pixel nodes to either black or white. Our goal is to find the cut with minimum total edge weight, and we do this using existing optimization methods [BVZ01].

Assume we wish to compute a binary image of $m \times n$ pixels in size. Our shading technique proceeds in three steps (summarized in Figure 2):

1. **Rendering**: Generate arrays $\mathbf{A}_i \in [0,1]^{m \times n}$ capturing different channels of the appearance of the object and $\mathbf{G}_i \in [0,1]^{m \times n}$ capturing geometric properties of the shape relative to screen space. The arrays can be generated using any rendering technique. We choose ray-tracing for the flexibility it offers during experiments.
2. **Graph construction:** Edge weights from pixel nodes are defined based on the values in the arrays $\mathbf{A}_i$ and $\mathbf{G}_i$.
   - **Appearance—edges to white and black**: For pixel (x,y) in the image, let $W[x,y]$ and $B[x,y]$ be the

Figure 3: *The effect of changing weights w on edges connecting pixels to white. Throughout these changes, $b = 0.5$.*

weights of edges that connect its node to the white and black nodes, respectively. These are defined as the linear combination of $W_i[x,y]$ and $B_i[x,y]$ (using w_i and b_i), where i is the index of the appearance component $\mathbf{A}_i$:

$$W[x,y] = \sum_i w_i W_i[x,y]$$
$$B[x,y] = \sum_i b_i B_i[x,y] \quad (1)$$

The derivation of W_i and B_i from $\mathbf{A}_i$ is explained in section 4. The values w_i and b_i manage the relative influence of these different appearance components (see figure 3).

- **Geometry—edges to neighbors**: Let $N[x_0,y_0,x_1,y_1]$ be the weight of edges connecting pixel (x_0,y_0) to (x_1,y_1). These weights are the product of a positive constant $\tilde{N}$ and factors N_j that are derived from the geometric components $\mathbf{G}_j$:

$$N[x_0,y_0,x_1,y_1] = \tilde{N}\prod_j N_j[x_0,y_0,x_1,y_1]. \quad (2)$$

The details of deriving N_j from $\mathbf{G}_j$ are discussed in section 5. $\tilde{N}$ controls the relative weight of these edges compared to edges connected to white and black.

3. **Minimal cut**: Based on assignment of weights, a minimal cut is approximated following Boykov et al. [BVZ01]. This minimum cut defines an assignment of pixels to white or black, based on the two connected components: one contains the white node, and one contains the black node.

The approximation of the minimum cut is fast enough to enable the user to iteratively improve the result by adjusting parameters used to combine the various appearance and geometry terms.

4. Appearance contributions to the graph

The appearance of an object depends on many material and illumination parameters and it is beyond the scope of this paper to attempt a systematic exploration. We demonstrate our approach to binary shading using simple appearance components:

Diffuse: The diffuse component is computed using a Lambertian reflection model: its value is the cosine of the angle between the surface normal and lighting direction. It is independent of the viewpoint.

Specular: The specular component takes into account the viewing position, the light source and the surface normal. The Beckmann distribution is used to calculate the specularity of a point.

Head-lamp: By placing a light at the camera, and using a Lambertian model, we shade based on surface orientation with respect to the camera. This term helps separate the object from the background by being darkest along the silhouette (on smooth parts of the surface).

While our approach is not limited to this list of appearance attributes, we have found that it generates enough variability for different binary rendering styles. We generate all values by ray-tracing and remap them onto the range $[0,1]$ into arrays $\mathbf{A}_i[x,y]$. We show the influence of different appearance components (upper row) on the weighted binary shading (lower row) in Figure 4.

The assignment of weights based on the $\mathbf{A}_i[x,y]$ alone will only consider global features. We know from previous work that we must consider local contrasts as well. Conse-

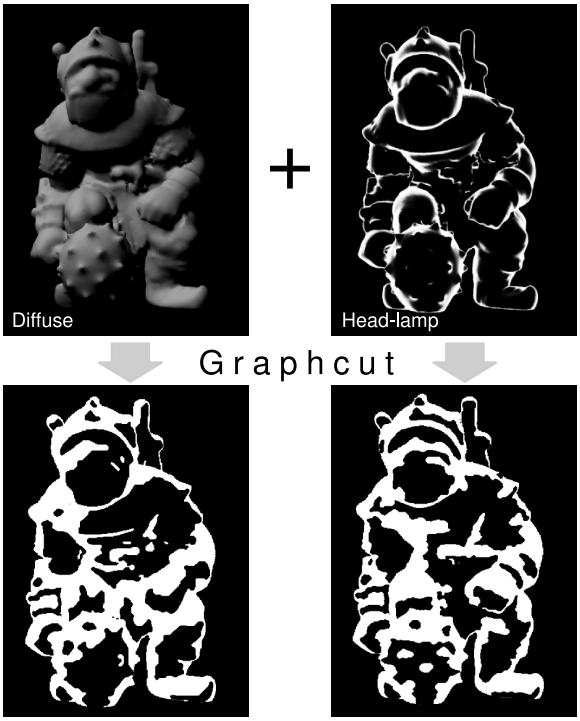


Figure 4: *Top: Appearance arrays rendered from the 3D model. Bottom: influence of these arrays on binary shading; diffuse (left), diffuse and head-lamp (right).*

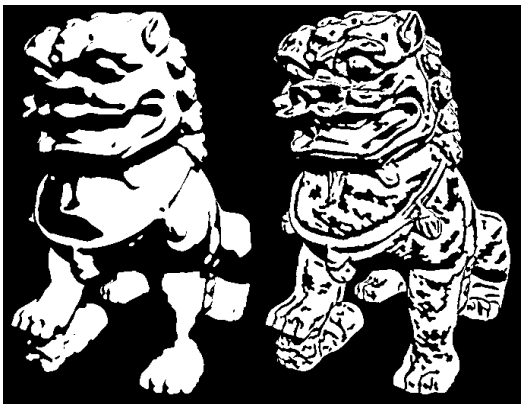


Figure 5: *Global (left) versus local (right) thresholding.*

quently, we model both global and local features explicitly. For the global contribution of each component we simply use $\mathbf{A}_i[x,y]$ and $1-\mathbf{A}_i[x,y]$ as contributions to the weights $W_i[x,y]$ and $B_i[x,y]$, respectively.

For modeling the local contribution to the weights we relate the values in $\mathbf{A}_i$ to spatial averages of $\mathbf{A}_i$. Let the average value $\overline{\mathbf{A}}_i[x,y]$ be based on a neighborhood in image space, i.e.

$$\overline{\mathbf{A}}_i[x,y] = \sum_{u,v} \mathbf{A}_i[u,v] e^{-\frac{(x-u)^2+(y-v)^2}{h_i^2}} \tag{3}$$

where h_i is a parameter to control the size of the neighborhood. For practical purposes we cut off the Gaussian weight function by only considering pixels within a square window.

We consider the difference of appearance attributes and their local averages:

$$\mathbf{A}'_i[x,y] = \mathbf{A}_i[x,y] - \overline{\mathbf{A}}_i[x,y]. \tag{4}$$

The sign of $\mathbf{A}'_i[x,y]$ determines whether pixel (x,y) is locally brighter or darker, and thus whether it should tend towards white or black. Thus we weight white as $\max(0,\mathbf{A}'_i[x,y])$ and black as $\max(0,-\mathbf{A}'_i[x,y])$, thereby only increasing one of the weights (the other is zero). These local terms are combined with the global measure so that:

$$\begin{aligned} W_i[x,y] &= g_i \mathbf{A}_i[x,y] &&+ (1-g_i)\max(0,\ \mathbf{A}'_i[x,y]) \\ B_i[x,y] &= g_i(1-\mathbf{A}_i[x,y]) &&+ (1-g_i)\max(0,-\mathbf{A}'_i[x,y]) \end{aligned} \tag{5}$$

where the parameter $g_i \in [0,1]$ controls the balance between global and local features. Figure 5 shows the difference between thresholding globally and locally. Beyond this, we ensure that the weights W and B are strictly positive by adding a small fixed value, which prevents pixels from being permanently bound to either white or black.

5. Geometry contribution to the graph

Nodes in the graph are connected to their 8-neighborhood. These connections result in a tendency of neighboring pixels to be assigned to the same terminal, despite potentially varying bias towards white or black based on the terminal weights. This yields the desired large black and white regions as well as smoothness of region boundaries.

We set the maximum neighbor edge weight to $\tilde{N}$. Larger values for $\tilde{N}$ lead to smoother and fewer regions in the final result—see figure 6. This value is then modulated (i.e. multiplied with factors between zero and one) to account for naturally connected or disconnected regions in the image based on the geometry. Here we consider as geometric features the depth and normal variation (i.e. curvature) relative to screen space.

Occluding contours are important cues for shape understanding. However, the local geometry of the shape could lead to similar appearance across contours. In such situations, the coherence enforced by the minimum cut would enforce the same color for pixels across contours. Consequently, we modulate the neighbor edge weights with the factor:

$$N_0[x_0,y_0,x_1,y_1] = e^{-\frac{(z[x_0,y_0]-z[x_1,y_1])^2}{d^2}} \tag{6}$$

where $z[x,y]$ is the distance to the camera for pixel (x,y), and the parameter d encourages cuts to be along depth discontinuities as it approaches zero.

It is also important to convey important features on the shape. Following the works of Lee et al. [LVJ05] and Gal and Cohen-Or [GCO06] we understand features as high local variation of curvature. Consequently, we assume that coherent regions on the shape have the property that the curvature at a point is similar to an average over the curvatures in a neighborhood of the point. Such regions meet in points whose curvatures greatly differ from the average curvature.

Rather than estimating curvatures on the shape, we follow the idea of Judd et al. [JDA07] and consider the derivatives of normals relative to screen space directions. Let $\mathbf{n}[x,y]$ be

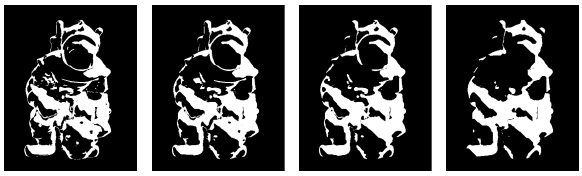


Figure 6: ***Influence of edge weights*** *In these examples, the maximum neighbor weight $\tilde{N}$ is modulated. Note that small disconnected regions are successively connected when increasing weight. The left image has zero neighbor weight, and is equivalent to thresholding.*

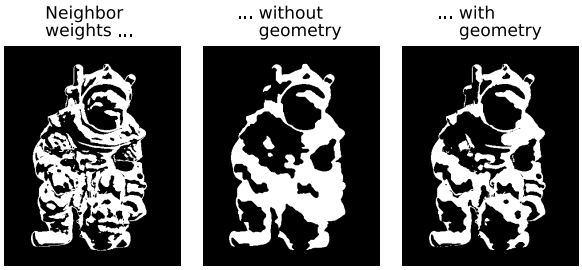


Figure 7: *The left-most image shows a segmentation with neighbor weights equal to zero which results in mere thresholding. In the middle image the neighbor weights are set to a constant value of $\tilde{N} = 0.5$, whereas the right image has the same base $\tilde{N}$ but uses the geometry terms to modulate the neighbor weight.*

the unit vector normal gathered from the 3D scene at pixel (x,y). Then we assign a directional curvature measure to pairs of neighboring pixels $[x_0,y_0],[x_1,y_1]$ as

$$\kappa[x_0,y_0,x_1,y_1] = d_{\mathbb{S}^2}(\mathbf{n}[x_0,y_0],\mathbf{n}[x_1,y_1]) \tag{7}$$

where $d_{\mathbb{S}^2}$ is the geodesic distance on the Gauss sphere.

Based on this measure, we modulate the weight of edges by relating this curvature measure to the average curvature measure in the neighborhood $[\bar{x},\bar{y}] = \left[\frac{x_0+x_1}{2},\frac{y_0+y_1}{2}\right]$

$$\overline{\kappa}[x_0,y_0,x_1,y_1] = \sum_{[u,v]} \kappa[\bar{x},\bar{y},u,v]\, e^{-\frac{(\bar{x}-u)^2+(\bar{y}-v)^2}{c^2}} \tag{8}$$

where c controls the neighborhood size. This yields the edge weight modulation

$$N_1[x_0,y_0,x_1,y_1] = e^{-\frac{\left(\kappa[x_0,y_0,x_1,y_1]-\overline{\kappa}[x_0,y_0,x_1,y_1]\right)^2}{k^2}} \tag{9}$$

where k is a user parameter to attenuate the effect of feature lines. The effect of considering curvature for attenuating edge weights is demonstrated in figure 7.

6. Performance and Comparisons

We have implemented our approach on the CPU, using the graph cut code of Boykov and Kolmogorov [BK04]. Table 1 shows rendering times for a laptop with a 1.83 GHz Core 2 Duo over different input mesh sizes and different viewpoints. Note that the times are good enough for interactive optimization of the results even on this platform.

The time required to fill the graph and generate the minimum cut, as well as the ray-tracing time, depend mostly on the number of pixels covered by the object, while being largely independent from the geometric complexity (i.e. by using a kD-Tree). Note that we use ray-tracing as a simple way to evaluate and explore various appearance properties.

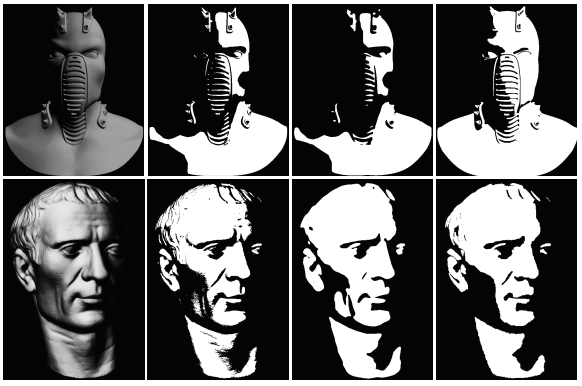
Torus model (4096 triangles)			
Nodes/Pixels	2290	17493	48272
Raytracing	0.74	1.76	2.50
Graph (fill)	0.076	0.33	0.74
Graph (cut)	0.0022	0.007	0.011
Overall	0.82	2.10	3.25
Caesar model (774164 triangles)			
Nodes/Pixels	5404	37030	118173
Raytracing	1.81	2.40	3.67
Graph (fill)	0.18	0.60	1.70
Graph (cut)	0.0067	0.019	0.056
Overall	2.00	3.02	5.43

Table 1: *Timings (in seconds) for different numbers of triangles and filled pixels.*

When rendering performance is of higher concern, GPU rasterization may be used instead.

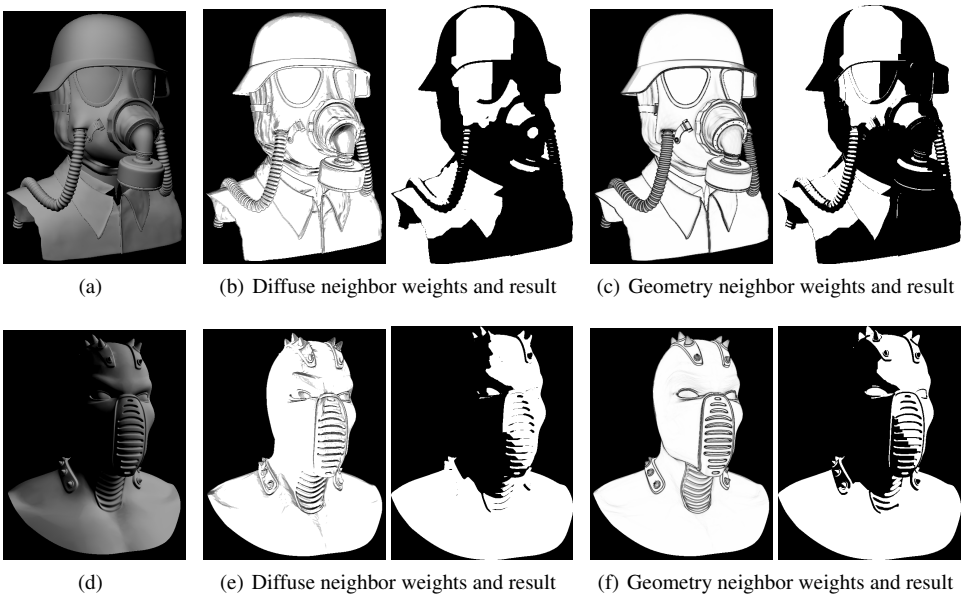
In the remainder of this section, we compare our technique to different families of binary image creation methods.

Uniform Image Thresholding and Local Binary Shading. Our approach to binary shading achieves better results than direct uniform image thresholding because it exploits more information about the scene. In particular, the contributions and influence of appearance components and geometric structure are taken into consideration explicitly. To visualize the difference this makes we compare threshold-



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

Figure 8: ***Comparison to thresholding*** *(a) diffuse input, (b) direct uniform thresholding, (c) uniform thresholding over a Gaussian blurred input (10x10 kernel) and (d) our approach using the unblurred diffuse input for setting the terminal weights, but adding neighbor weights from geometry. This effects homogeneous regions where geometry is smooth, while still conveying features—without searching for a compromise among Gaussian kernels.*



(a) (b) Diffuse neighbor weights and result (c) Geometry neighbor weights and result

(d) (e) Diffuse neighbor weights and result (f) Geometry neighbor weights and result

Figure 9: *Comparison to an image based binarization approach. All results are generated with the same input for terminal weights, namely only global information from a diffuse shading, seen in (a) and (d). For the image based approach ((b) and (e)), neighbor weights are derived from the diffuse shading only (therefore being similar to what can be achieved with image-space only approaches like Mould and Grant [MG08]), while we use geometric information in our approach ((c) and (f)). The neighbor weights are visualized on the left side of each pair as the normalized sum of all neighbor weights of each node (black meaning no edge weight at the corresponding pixel) while the segmentation result is displayed on the right side of each pair.*

ing a shaded image as well as a smoothed version of the shaded image. For the Ninja model, smoothing removes important geometric features and the result without smoothing looks better, while for the Caesar model the geometric details make smoothing necessary. For a fair comparison, we use the same input for setting the terminal weights in our approach (i.e. using only global information from the diffuse component). While preserving features, we can still achieve homogeneity in smooth regions, and the result automatically adapts to the necessary amount of detail. Figure 8 shows the results.

Real-time binary shaders are often similarly based on thresholding a mix of appearance properties (or other quantities such as normals), implemented on the GPU for performance reasons [VBGS08]. While the context is different from image thresholding, the results are necessarily the same, as the assignment of pixel color is based on thresholding local information. Our use of global optimization here means our approach can make more coherent decisions across significant parts of the image and shape, albeit at the price of not running at interactive rates. Figure 10(a) shows results with clearly visible coherent decisions, such as the shading boundaries that run roughly along the ridges on the head of the chameleon—these particular results can be compared directly to the black and white results from [VBGS08] (see figure 10(b)), which lack this coherence.

Of course, these advantages come at the price of computation time, and even if our implementation can be significantly improved using rasterization (see Table 1), local binary shading methods are still methods of choice for applications such as games.

Variational Image Binarization. While the advantage over simple image thresholding might be rather obvious, it is important to point out that even global variational techniques for thresholding are less convincing if they are based on a single shaded depiction of the object. The reason is, again, that geometric structures might be lost in the shaded image, and no thresholding technique could ever revive them.

We have used our approach to implement an artistic image thresholding approach similar to the one of Mould and Grant [MG08] (which is also based on graph cut). In particular, we derive neighbor weights from the input images by comparing neighboring pixels (for details see [MG08]). We apply this thresholding implementation to a diffuse rendering and compare to the result of our approach, which has been similarly limited to use only the diffuse channel as an appearance attribute. Thus, both approaches start from the same appearance information, but only our technique

(a) (b)

Figure 10: *(a) Our approach.* ***Left and center****: Global terms only ($g_i = 1$). Left: Head-lamp and specular terms, with the head-lamp term giving the basic sink and source weights and the specular term only adding to the source. $\tilde{N} = 0.8$, $c = 0.07$. Center: Diffuse term combined with haloed occluding contours. $\tilde{N} = 1.1$, $c = 0.1$.* ***Right****: Mix between local and global diffuse term, $g_i = 0.1$, $\tilde{N} = 0.15$, $c = 0.07$. When the local term is used, the terminal weights are smaller as compared to the global term, thus $\tilde{N}$ is set lower. (b) Results obtained by Vergne et al. [VBGS08].*

uses geometric information for the modulation of neighbor weights. Figure 9 shows the effect of the additional information, emphasizing surface structures that are hard to derive from the diffuse shading.

On the other hand, using "invisible information" suggests our binary images differ more from realistic shading. One might argue that the restriction to only two levels makes it harder to convey both appearance and structure, and in our approach we specifically opt for conveying structure rather than appearance.

Line Drawing. Finally, it is interesting to compare the results of binary shading to line drawings (see figure 11). Line drawings implicitly define the regions that we try to assign constant colors; conversely, binary shading implicitly defines feature lines as boundaries between regions. In this sense, the two approaches are dual.

However, we feel that concavities can be communicated more clearly using larger black regions (as can be seen in the golf ball example in Figure 11). A combination of toon shading and line art rendering [DR07] that includes suggestive contours and suggestive highlights produces results that bear some similarity to our style.

As is obvious from the dual relationship, the fundamental idea of optimizing the result in image space could also be applied to line art rendering, potentially helping to connect otherwise disconnected feature lines or help to fill empty spaces—something artists report doing [CGL*08]. Similarly, global optimization could help combining different types of lines [DR07].

7. Conclusions and Discussion

Our approach to binary shading offers a flexible framework to control the effects of appearance and geometry on the shape and placement of black and white regions in a binary rendering. The geometry term controls how regions partition the surface, with boundaries that often run along surface features. Changing the underlying rendering procedures allows for different styles.

User interaction might help in reaching visually pleasing results with less effort. We have implemented a user interface based on learning the component statistics of regions that the user suggests to be white or black (similar in spirit and design to Rother et al. [RKB04]). However, we find that the control of style is comparable to adjusting the parameters directly using sliders—at least for us. As we lack conclusive information on the usability of this interface for other users, we leave this for future work. A similar strategy based on deriving probability distributions could be used for learning a model from a large collection of artist data. However, no such data set is available at present.

We have also conducted experiments on the degree of temporal coherence achieved by our technique, when applied to each frame separately. The notion of temporal coherence for styles such as binary shading is hard to define, since jumps must happen on occasion—fading in and out is not possible with just black and white. In typical situations, we found that, apart from visibility changes, large regions do not drastically change from one frame to the other. Thus,

Figure 11: *Depictions of a golf ball: line art rendering using suggestive contours [DFRS03] as well as a toon shader combined with suggestive contours and suggestive highlights [DR07], and our black and white results.*

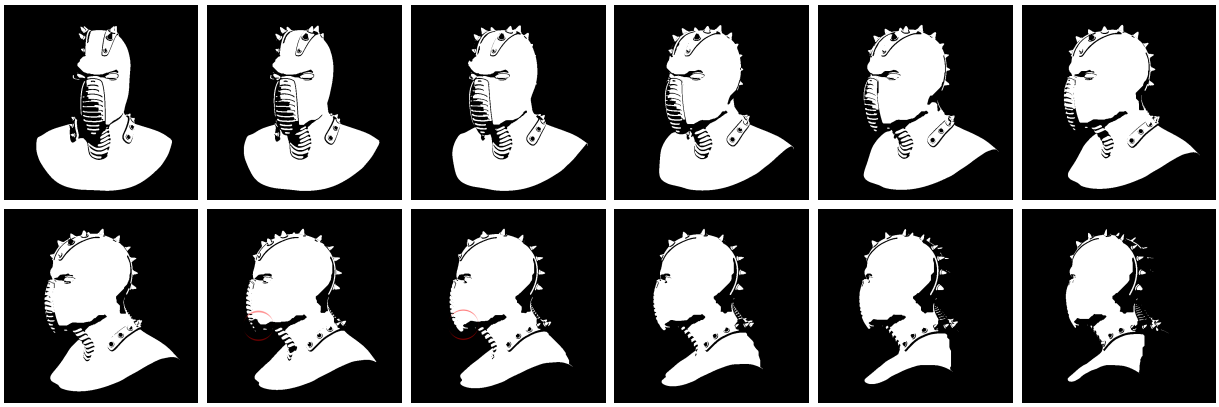


Figure 12: *Binary shading applied to animation, showing every other frame of a sequence. Top row: Regions appear and disappear progressively, and coherently. Bottom row: failure case with tangential regions (circled in red), where the incoherence is easily perceptible.*

the algorithm often gives acceptable results when performed on a per-frame basis—see Figure 12(top). This comes from the fact that the energy implicitly defined on the graph is strongly attached to object space quantities (from geometry and appearance), while the particular screen-space local connectivity has a minor impact on the graph flow and perceived region boundaries. However, this general observation fails on regions which, during the animation, quickly become viewed edge-on. In such cases, the screen-space projection effects significant differences from frame-to-frame. This appears as a sudden large change between black and white—see Figure 12(bottom). Inspired by work such as [WXSC04], we seek to improve temporal coherence through the analysis of a spatiotemporal volume of images, where each image in the stack represents a particular moment in time. In our case, the graph can be constructed to correspond to this volume. We experimented with different approaches for the weights and the connectivity of the nodes between the frames, but none delivered more coherent but still visually satisfactory results. Temporal coherence of this style presents a significant challenge and is left for future work.

We feel that the idea of using different appearance attributes could be exploited further: mirror or glossy reflections would aid the depiction of plastic or metallic surfaces, a style often used in manga drawings. More appearance and geometry channels might also enable or improve the success of learning attribute statistics from user data.

We have focused on generating binary images. Clearly, extending the approach towards more colors, a set of dithering or hatching styles, or other sets of attributes is possible. The main idea is to render appearance attributes into channels and to compute the screen space properties from the different attributes, rather than combining the attributes before rendering. This basic idea clearly has applications in a variety of rendering techniques.

Acknowledgements This work has been partially supported by an Institut Telecom's *Futur&Rupture* Grant. D. DeCarlo acknowledges support from the Alexander von Humboldt Foundation and the National Science Foundation under grant CCF-0541185.

References

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE trans. on PAMI 26*, 9 (Sept. 2004), 1124–1137.

[BRO08] BRONSON J., RHEINGANS P., OLANO M.: Semi-automatic stencil creation through error minimization. In *Proc. NPAR* (2008), pp. 31–37.

[BTM06] BARLA P., THOLLOT J., MARKOSIAN L.: X-toon: an extended toon shader. In *Proc. of NPAR* (2006), DeCarlo D., Markosian L., (Eds.), ACM, pp. 127–132.

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell 23*, 11 (2001), 1222–1239.

[CGL*08] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 88:1–88:11.

[Dec96] DECAUDIN P.: *Cartoon-Looking Rendering of 3D-Scenes*. Tech. Rep. 2919, INRIA, 1996.

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (2003), 848–855.

[DR07] DECARLO D., RUSINKIEWICZ S.: Highlight lines for conveying shape. In *NPAR* (2007), pp. 63–70.

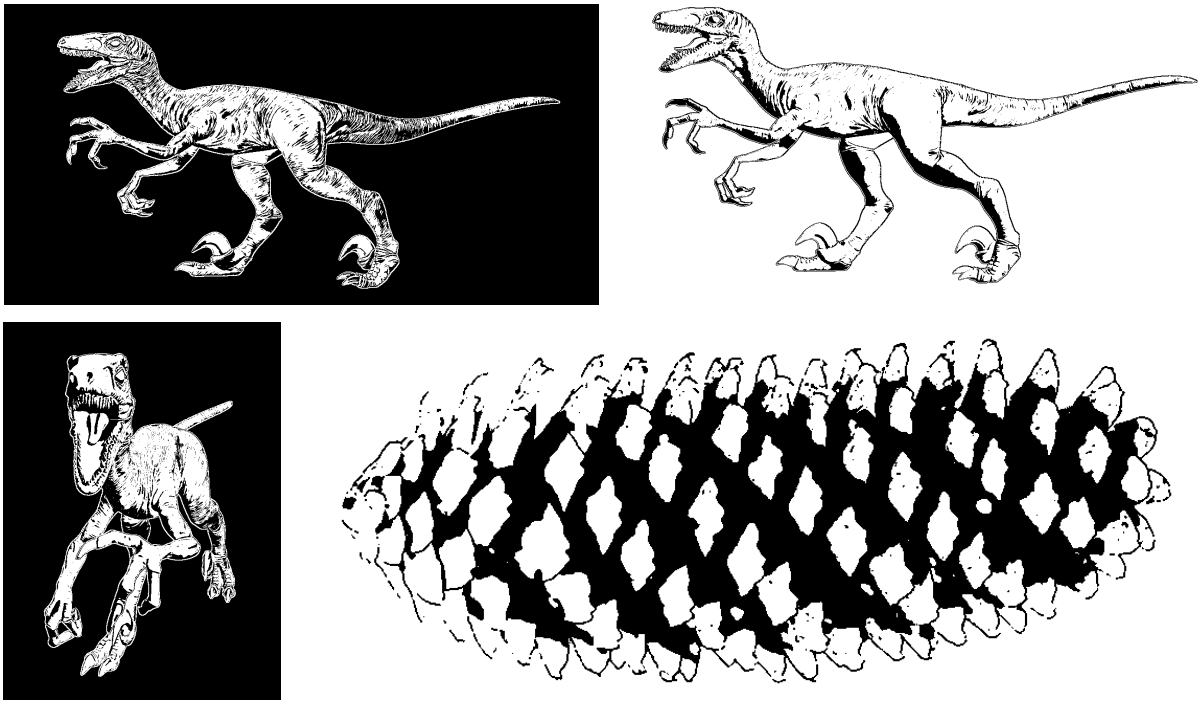


Figure 13: ***Top and bottom-left****: Results using only the local diffuse term ($g_i = 0$ for diffuse with neighborhood size radius of 4 pixels) combined with haloed occluding contours. The base neighbor weight is $\tilde{N} = 0.2$ and modulated using our curvature geometry term. The respective neighborhood size control is $c = 0.2$ for the images on the top and $c = 0.1$ for the image on the bottom-left.* ***Right-bottom****: Our method applied to RGBN data [TFFR06]; in this case, contours are drawn on top using the provided discontinuity map, after median filtering and thresholding.*

[FS76] FLOYD R. W., STEINBERG L.: An adaptive algorithm for spatial greyscale. *Proc. of the Society for Information Display 17*, 2 (1976), 75–77.

[GCO06] GAL R., COHEN-OR D.: Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics 25*, 1 (Jan. 2006), 130–150.

[JDA07] JUDD T., DURAND F., ADELSON E. H.: Apparent ridges for line drawing. *ACM Trans. Graph. 26*, 3 (2007), 19.

[KGV83] KIRKPATRICK S., GELATT C. D., VECCHI M. P.: Optimization by simulated annealing. *Science 220* (1983), 671–680.

[KI86] KITTLER J., ILLINGWORTH J.: Minimum error thresholding. *Pattern Recognition 19*, 1 (1986), 41 – 47.

[KZ93] KAMEL M., ZHAO A.: Extraction of binary character/graphics images from grayscale document images. *Graphical Models and Image Processing 55*, 3 (May 1993), 203–217.

[LCP90] LEE S. U., CHUNG S. Y., PARK R. H.: A comparitive performance study of several global thresholding techniques for segmentation. *Comp. Vis., Gfx, & Im. Proc. 52* (1990), 171–190.

[LL98] LEUNG C. K., LAM F. K.: Maximum segmented image information thresholding. *GMIP 60*, 1 (Jan. 1998), 57–76.

[LMLH07] LEE Y., MARKOSIAN L., LEE S., HUGHES J. F.: Line drawings via abstracted shading. *ACM Transactions on Graphics 26*, 3 (July 2007), 18:1–18:5.

[LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. *ACM Trans. on Graph. 24*, 3 (Aug. 2005), 659–666.

[MG08] MOULD D., GRANT K.: Stylized black and white images from photographs. In *NPAR* (2008), pp. 49–58.

[OH95] OSTROMOUKHOV V., HERSCH R. D.: Artistic screening. In *Proc. of SIGGRAPH* (1995), pp. 219–228.

[OH99] OSTROMOUKHOV V., HERSCH R. D.: Multi-color and artistic dithering. In *Proc. of SIGGRAPH* (1999), pp. 425–432.

[RBD06] RUSINKIEWICZ S., BURNS M., DECARLO D.: Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics 25*, 3 (July 2006), 1199–1205.

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (2004), 309–314.

[SDM06] SPINDLER M., DËĘHRING N. R. R., MASUCH M.: Enhanced cartoon and comic rendering. In *Proc. of Eurographics Short Papers* (2006), pp. 141–144.

[TFFR06] TOLER-FRANKLIN C., FINKELSTEIN A., RUSINKIEWICZ S.: Illustration of complex real-world objects using images with normals. In *NPAR* (2006), pp. 111–119.

[Uli87] ULICHNEY R.: *Digital Halftoning*. MIT Press, 1987.

[VBGS08] VERGNE R., BARLA P., GRANIER X., SCHLICK C.: Apparent relief: a shape descriptor for stylized shading. In *Proc. of NPAR* (2008), ACM, pp. 23–29.

[VPB*09] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SCHLICK C.: Light warping for enhanced surface depiction. *ACM Trans. Graph. 28*, 3 (2009), 1–8.

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. *ACM Trans. on Graph. 23*, 3 (2004), 574–583.

[XK08] XU J., KAPLAN C. S.: Artistic thresholding. In *Proc. of NPAR)* (2008), pp. 39–47.

[XKM07] XU J., KAPLAN C. S., MI X.: Computer-generated papercutting. In *Pacific Graphics* (2007), pp. 343–350.

[YB89] YANOWITZ S. D., BRUCKSTEIN A. M.: A new method for image segmentation. *Comp.Vis., Gfx & Im. Proc. 46*, 1 (1989), 82–95.

Figure 14: *Input models used in figures 10 and 13.*