

Frame-to-Frame Coherent Halftoning in Image Space

Mike Eissele Daniel Weiskopf Thomas Ertl

Institute of Visualization and Interactive Systems
University of Stuttgart, Germany
{eissele | weiskopf | ertl}@vis.uni-stuttgart.de

Abstract

We present an interactive technique for generating halftoning animations in image space. We specifically consider a halftoning approach that uses screening methods to achieve shading, brightness, and texture by placing varying patterns. Our technique transports the screening patterns in a frame-to-frame coherent manner according to the velocity of each fragment on the image plane. We show how the algorithm is mapped to programmable graphics hardware to achieve interactive frame rates. Our approach can be used for any screening-based halftoning technique, and therefore different non-photorealistic image-space rendering styles, including dithering, screening, stippling, and hatching, can be utilized to generate real-time frame-to-frame coherent animations.

1. Introduction

Generating drawings using different patterns to convey shading and texture is an effective and widely used technique. These methods, named *halftoning* [21], simulate a wide color spectrum with only a few discrete color values. In terms of non-photorealistic rendering (NPR), halftoning is applied to introduce patterns in the drawing in order to achieve a clearer and more pleasant display style. For example, many illustrations in textbooks on archaeology or medical science are drawn in a stippling style, even if photographs of good quality are available. The presented technique uses screening as the basis for halftoning, where different tones are represented through predefined patterns or figures [12].

For the automatic generation of halftoning images, two possible approaches have evolved: halftoning in object space as illustrated by Freudenberg et al. [3] or Praun et al. [14], and halftoning in image space as, e.g., shown by Secord [16]. Both approaches have specific advantages and disadvantages.

In object-space methods, frame-to-frame coherence can quite easily be achieved by attaching the halftoning structure to scene objects. In this way, halftoning patterns always move with their corresponding object, and therefore temporal coherence is preserved. Typically, halftoning elements are stored in the form of a texture; texture coordinates are specified in a way to “glue” these textures onto the objects, thereby automatically linking the halftoning fragments with the object surface [3, 14]. In pen-and-ink illustrations the halftoning fragments are often drawn beyond the object borders to achieve a more natural drawing style. Approaches in object space cannot easily fulfill this demand, as the halftoning process is restricted to the object surface. Moreover, a prerequisite for object-space methods is a 3D scene with adequately assigned texture coordinates and therefore input data without the information of the complete 3D scene cannot be processed. Another important drawback of object-space techniques shows up for screening illustrations. A screening process of the complete image plane is not possible, as screening is applied separately for each object surface. Moreover, the occurrence of the screening patterns should be independent of the object scale and orientation, which is hard to achieve in object-space approaches.

Conversely, image-space techniques overcome many restrictions of object-space methods. The halftoning fragments can easily cross object borders, as the halftoning operation is performed in the image plane. Often, image-space algorithms rely only on 2D information and, therefore, can also be applied to video sequences. In contrast to object-space methods, image-space approaches can easily meet the requirement that screening patterns are rendered independently of the object scale or orientation. On the other hand, the main problem for image-space methods is how to maintain temporal coherence since, in general, the halftoning process is independent of the objects.

In this paper, we address the issue of frame-to-frame coherent halftoning in image space in order to preserve the advantages of image-space techniques for animated sequences. We present a rendering technique that transports the halftoning structures in image space according to the

motion of the underlying objects. It is capable of generating time-coherent animations of all halftoning techniques that are based on gray-scale mappings by a underlying screening matrix. Therefore many display styles such as ordered dithering [18], clustered dot screening [19], or void and cluster screening [19] can be implemented. Additionally, stippling [16] or hatching [3] illustrations can be simulated with appropriate screening matrices. Moreover, it is demonstrated that the rendering algorithm can be mapped to graphics hardware, ensuring the generation of non-photorealistic animations in real time.

The remainder of this paper is organized as follows. In Section 2, we review previous work. The main algorithm is described in Section 3. A discussion of the implementation and results follows in Sections 4 and 5. The paper ends with a short conclusion and an outlook on possible future work.

2. Previous Work

The field of NPR methods has been an active area of research; an overview can be found in the textbooks by Gooch and Gooch [4], Strothotte and Schlechtweg [18], and in the SIGGRAPH Course on NPR [5].

Most previous work is focused on imitating various artistic drawing styles for still images, while we restrict our overview on previous work to topics directly related to this paper: pen-and-ink styles and halftoning with the focus on animated renditions. Secord [16] presents a method to generate stipple drawings, where the points are distributed according to a Voronoi diagram. An interactive technique is also mentioned, where the area for each pixel is mapped to a precomputed point distribution according to the intensity. Jodoin et al. [8] describe a system that generates sequences of strokes from an initial training sequence. An approach for screening visualizations by Ostromoukhov and Hersch [12] uses a set of predefined figures and interpolates the contours of these for all intensity values.

Another field of research specifically addresses the problem of generating NPR animations. Freudenberg et al. [3] propose a concept to generate interactive NPR animations in object space using graphics hardware. Praun et al. [14] introduce tonal art maps (TAM) for hatching in object space. These precalculated mip-map stroke textures directly address the problem of maintaining coherence across scales and object orientations.

Meruvia Pastor and Strothotte [9] describe an approach for frame-coherent stippling in object space. Secord et al. [17] redistribute an initial point arrangement through a probability density function (PDF), derived from a single input image, to achieve frame-coherent stippling animations. Cunzi et al. [1] present an image-space approach to animate the background canvas in NPR illustrations. They use a limited set of transformations to animate the back-

ground paper according to the camera movements. Ostromoukhov and Hersch [13] present a technique to generate a large stochastic dither matrix to generate stippling illustrations through a screening process. In an earlier publication by Ostromoukhov [11], random spatial structures are used as point primitives and rendered with a screening technique.

For the image-space method of this paper, we adopt the concept of a G-buffer as proposed by Saito and Takahashi [15]. The G-buffer generalizes the framebuffer to allow for an extended set of attributes per pixel. These attributes may include color, depth, normal vector, texture coordinates, and even more complex parameters. The final rendering step uses only these attributes of the intermediate G-buffer for generating the resulting image.

The transport of halftoning patterns is accomplished by texture advection. Texture advection was originally developed for visualizing vector fields. Recently, Jobard et al. [7] and van Wijk [20] have presented variants of texture advection specifically designed for time-dependent flows. Our implementation makes use of the GPU-based version [22] of Jobard et al.’s approach.

3. Algorithm

3.1. Overview

Screening techniques combine a gray-scale or color image with an underlying screening mask to obtain the final rendering. Figure 1 shows a typical example for a screening technique—threshold screening. Here, the gray value of a fragment from the original picture is compared to entries in a screening matrix. The final pixel is black when the original gray value is below the corresponding entry, otherwise white. Other screening approaches may use more complicated ways to determine the color of the final pixel. However, in threshold screening the screening mask, which covers the complete image space, is decomposed into smaller matrices. These screening matrices are located on a uniform grid, filling the screening mask. Thus, the position of elements of the screening mask is independent of the motion of the underlying scene. Therefore, previously known screen-

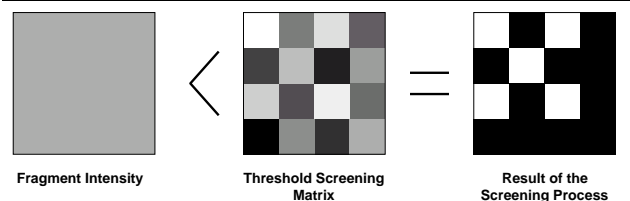


Figure 1. Threshold screening technique.

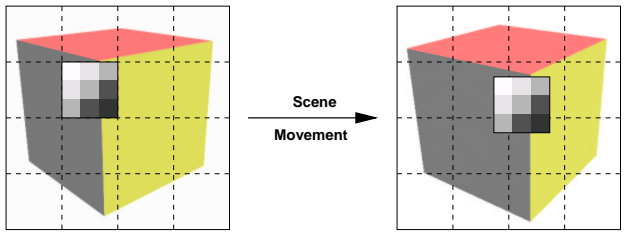


Figure 2. Advection of a screening matrix.

ing techniques suffer from the “shower door effect”, where objects of the scene appear to swim through the texture.

To overcome this fundamental drawback of such an image-space technique, we propose an approach that advects the screening mask in image space according to the velocity of each pixel in the image plane. In this way, a coupling of the screening mask and the moving objects of the scene is established. Figure 2 shows two snapshots taken from a rotating cube to illustrate how a single screening matrix is moving with the underlying object. The image-space velocities of the fragments—in the context of computer vision also referred to as optical flow [5, 6]—can either be calculated directly from the 3D scene, for example, by using rigid body kinematics, or they are computed from video sequences [6]. In both variants, the G-buffer framework is extended by an additional attribute to provide the velocity of fragments with respect to the image plane.

The following issues are introduced by an advection of the screening mask. First, screening matrices may move to positions that are no longer aligned with the underlying grid, which is shown in the right image in Figure 2. A second problem is caused by velocity fields that contain regions of divergence, convergence, and discontinuity. For example, divergence and convergence appear in the case of a rotating object or camera zooming during the animation; discontinuities show up at object borders. In these areas, in the following referred to as *distortion areas*, the velocity field causes distortions in the screening mask if only a simple advection is performed. In addition, the spatial frequencies of the mask might change.

Stated differently, image-space advection of screening masks is subject to the following contradicting requirements: (1) screening matrices should have equal size and be aligned to an underlying grid, (2) screening matrices should accurately follow the motion of the objects. Since there is no optimal solution for both requirements, we propose a way to combine the solutions for both aspects.

We introduce multiple screening masks to realize the above requirements, each with the same size as the image plane: (1) a static, non-advected mask C , which accounts for screening matrices aligned to the underlying grid; (2) a screening mask S , which is advected to follow the motion of

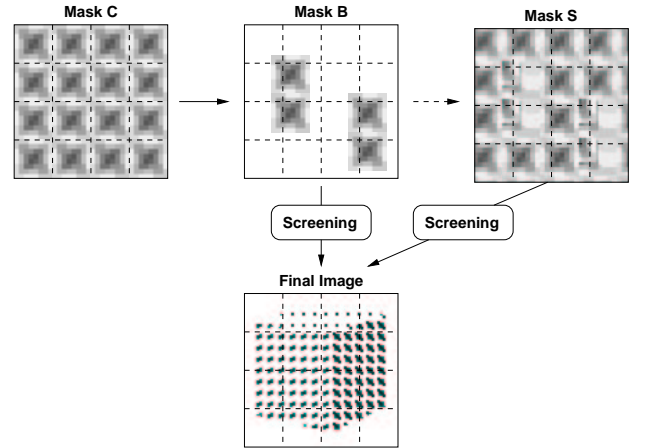


Figure 3. Combining the different masks.

the objects; and (3) a so-called *blend-in screening mask* B , used to couple the above two masks and make possible a smooth insertion of new screening matrices. Figure 3 illustrates the different masks and how they are combined for the generation of the final image.

Mask S is advected solely based on the velocities of the scene objects. Whenever the artifacts caused by distortion areas become too large, new elements of the mask C replace these regions of S . The blend-in screening mask allows these new elements to be blended in smoothly and thus eliminates popping artifacts. In the actual implementation, masks C and B are not combined to form another mask; rather both are applied to render intermediate halftoning images that are afterwards combined to the final image.

The complete process can be split into three main calculation paths: calculation of the final image, update of the blend-in screening mask B , and update of screening mask S . Figure 4 illustrates the flow chart for the combination of all three tasks. In the following subsections, the details of these three steps are discussed.

3.2. Calculation of the Final Image

To generate the final image, the results of the screening process with the mask S and the blend-in mask B have to be combined. Therefore, each fragment of the screening mask S has an appended status information that consists of the total advection distance in x - and y -direction. Each time the fragment is advected, these distance counters are updated accordingly. The scale for the counters are chosen in a way that an absolute value of 1.0 corresponds to an advected distance equal to the extent of the screening matrix in this direction. Additionally, the blend-in screening mask B holds a timer that is incremented each frame and thus reflects the age of the fragment in the B mask. A timer value of 1.0 corresponds to the maximum age of a fragment.

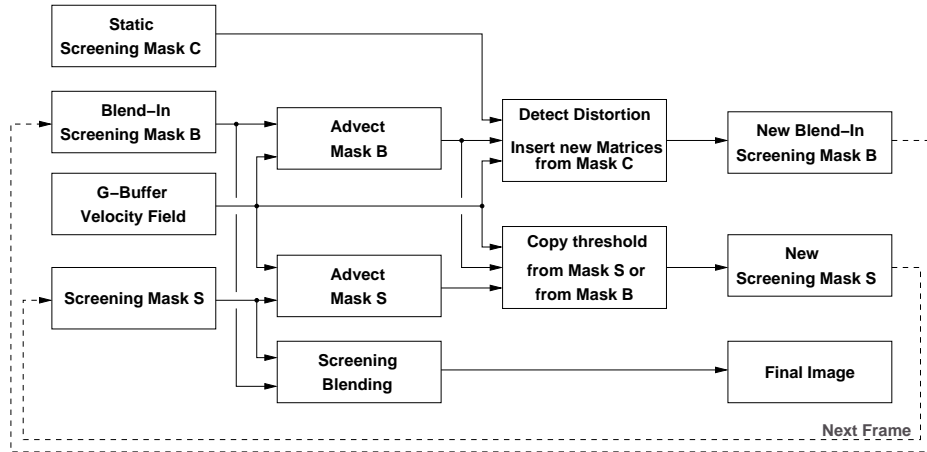


Figure 4. Calculation scheme for one time step of an animation.

The maximum of these values is calculated to determine the weight for blending the results of the screening mask S and the blend-in mask B . This ensures that the fragment is completely blended when it has either been advected by the size of one screening matrix in x - or y -direction or the maximum time for the blending has elapsed. The timer restricts the duration of the blending process to a user-specified threshold, while the direction counters accelerate the blending depending on the advection velocity.

3.3. Update of the Blend-In Screening Mask

The first step for this part of the algorithm advects the blend-in screening mask B according to the velocity values in the G-buffer attribute and increments the timer for each fragment. A backward Lagrangian-Eulerian advection (LEA) scheme [7] is applied to transport elements of B by displacing them at each time step. LEA makes use of nearest-neighbor sampling in the backward advection lookup to maintain the contrast of the original input texture and to preserve the structure of the screening mask. Unfortunately, a straightforward implementation of advection with nearest-neighbor sampling would result in an animation that is partitioned into regions of constant motion, within which the integer part of the displacement is constant. This unpleasant quantization of the velocity vectors is described in more detail in [7]. Fractional coordinates are used to overcome this problem: These additional coordinates determine the subtexel position of each element of the input texture and are updated for each time step according to the velocity field. In this way, subtexel motion is possible and the animation is no longer partitioned into regions of quantized velocities.

The primary purpose of the blend-in mask is to detect the distortion regions in the velocity field and schedule these for a replacement in the blend-in mask B with new matrices

from the static screening mask C . As a measure for the distortion, the difference of the velocity at the current fragment position and at the previous position is considered. Unfortunately, the velocity field can contain a wide range of velocity values and therefore simple thresholds for finite differences are not appropriate. Thus, our implementation actually computes the ratio of both velocities. If the ratio is below a given divergence threshold, the fragment is assumed to be located in a divergence region and therefore marked for an overwrite with the static screening mask C . A ratio above the convergence threshold also initiates an overwrite of the fragment. Additionally, the distance counters are examined to check whether the fragment has been advected for more than the extent of the screening matrix, which also initiates an overwrite by the value of the static screening mask C .

To ensure that only complete matrices are inserted, the distortion regions have to be exactly aligned with the layout of the static screening mask C as shown in Figure 5. All fragments of a newly inserted screening matrix receive an initial value of 0.0 for both the timer and the distance counters. Along the subsequent frames, the contribution of the blend-in screening mask B to the final image increases monotonically, depending on the distance and the timer value, until one of the values reaches 1.0 and the information of the blend-in screening mask B of the current fragment is transferred to the screening mask S .

3.4. Calculation of the Screening Mask

For each frame, the screening mask S is advected according to the velocity value for each pixel. Due to the distortion areas in the velocity field the advected screening mask S contains areas in which the screening matrices are skewed. The method described in Section 3.3 inserts new matrices in the blend-in screening mask B to smoothly re-

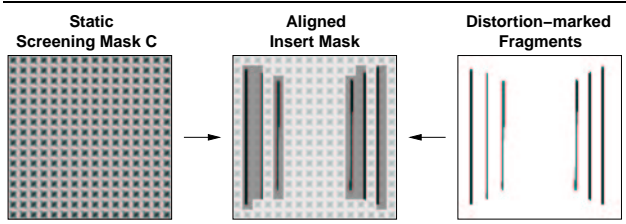


Figure 5. Alignment of the insert mask to ensure complete matrix insertion.

place the distorted matrices in the screening mask S over a couple of frames. Therefore, during the calculation of the screening mask S it is only important to detect when the blend-in screening mask B has been completely blended. Then the screening matrices are copied from the mask B to the screening mask S .

4. Implementation

Our implementation is based on C++ and DirectX 9.0 [10]. The G-buffer framework utilizes the render-to-texture functionality of DirectX to store the G-buffer attributes of each pixel in textures. The velocity attribute stores the velocity in screen space multiplied by the frame-to-frame time, which yield the distance the fragment moves in one time step. For rotating objects the magnitude of the velocity near the rotation center is very low, whereas the velocities further away from the rotation center could be very high. Thus the velocity field could contain a huge range of the values, and for that the G-buffer uses a texture format with 32-bit float values in each color component to store distances. Our image-space rendering algorithm uses the Pixel Shader 2.0 functionality through the DirectX effect files feature [10]. A similar implementation based on OpenGL can be achieved via the CgFx framework [2].

For the calculation of the final image the screening technique is applied separately for the screening mask S and the blend-in mask B . We use screening patterns that encode thresholds in textures, and apply a smooth threshold function [3] to evaluate the screening. The results are blended according a factor f , which is calculated from the maximum of the timer and the two distance counter values, and finally written to the frame buffer. If f is greater than 1.0, the blend-in mask has already been completely blended and copied to the screening mask S . In this case the blend-in screening mask has no contribution to the final image and only the result of the screening with the mask S is applied for the final image.

During the update of the blend-in screening mask B the distance values are used to apply a hardware-based back-

ward texture advection [22] to the mask, utilizing dependent texture lookups and maintaining fractional coordinates. The extra information for the timer and the distance counters for each fragment are encoded in the free components of the blend-in mask and calculated as described in Section 3.2 during the blend-in screening mask update.

Additionally, the distortion areas have to be detected, by computing the ratio of the G-buffer's distance value at the position in the previous frame to the distance at the current position. If the ratio is below the divergence threshold, in the implementation we used 0.3, the current fragment will be regarded as distorted, a difference above the convergence threshold, in the implementation 1.1 is used, also marks the fragment as distorted. The total advection distance of a fragment is used as a third criterion to determine a distortion. It is compared to the corresponding extent of the screening matrix in each direction. If the advection exceeds the matrix dimension in either direction, the fragment is considered to be distorted and is scheduled for an override.

In a subsequent rendering pass, the size of these distortion areas are enlarged to fit the size of a screening matrix, which is illustrated in Figure 5. Therefore, a render target with the size of the image plane divided by the size of the screening matrix is used, where each fragment corresponds to one screening matrix in the image plane. To determine whether a fragment of the downsized render target has to be set as distorted, all fragments of the corresponding screening matrix in the image plane should be tested if at least one of them is marked. To access all fragments of an 8×8 screening matrix 64 texture lookups are needed, which is more than most of today's hardware can provide. The values to be read are binary flags of 0 or 1. To determine a distortion of a fragment in the downsized distortion mask it is enough to know if at least one fragment of the corresponding screening matrix is marked as distorted. Bilinear texture filtering accesses a 2×2 block of texels within a single texture lookup operation and combines the four values to a single interpolated value. It is sufficient to check this result against 0 to determine if at least one texel has a value greater than 0. Therefore only 16 texture lookups are necessary to calculate the enlarged distortion area for a 8×8 screening matrix. For very big screening matrices a hierarchical multipass approach can be applied, whereby each pass reduces the number of fragments which map to a screening matrix until one fragment maps to one matrix.

In an additional pass this render target is resized to the output resolution using nearest-neighbor sampling. Now each area of distortion is aligned with the size of the screening matrix and, depending on the insertion mask, either the advected value of the blend-in screening mask from the previous frame is used or a new value is inserted from the static screening mask C .

In the update process of the screening mask S an advec-

tion is performed, exactly in the same way as for the blend-in mask. Afterwards the distance counters and the timer, which are stored in the blend-in mask B , are checked if one of them exceeds their maximum of 1.0, which will initiate a copy of the screening mask value from the blend-in mask B to the screening mask S . This transfer of the screening value is also triggered if the fragment of the blend-in mask B is scheduled for an override with the static mask C , since otherwise the content of the screening mask B would be lost.

5. Results

Figure 7 shows a sequence of a rotating car in a computer-generated animation. For these renderings, the velocity at each vertex of the mesh is evaluated using rigid body kinematics in object space and projected to the image space. The linear interpolation of the vertex attributes within each polygon of the mesh is used to calculate the velocity of the fragments. The illustrations demonstrate a scale and orientation independent representation of the screening patterns over the entire image plane. The correspondence of the screening patterns are shown in the two enlarged parts at the bottom of the figure. They indicate that the motion of the screening patterns is coherent with the motion of the underlying scene. In areas of extreme distortion at the left side of the car, some screening matrices are still distorted. In these cases it is hard for the algorithm to decide whether an advection of the matrix or a replacement should be performed.

Figure 6 depicts a rotating pot. This sequence shows that the patterns of the screening algorithm follow the texture of the pot while it rotates. This scene contains much less severe distortions areas, which leads to better results in the illustration of the screening patterns.

Further video material can be accessed via the internet at <http://www.vis.uni-stuttgart.de/~eissele/eguk04/> on our project page to demonstrate the frame-to-frame coherent screening algorithm.

Table 1 shows the measurement of the framerates in frames per second for different screen resolutions. The framerates with and without the generation of G-buffer attributes are displayed, i.e., the last row shows the performance numbers for the screening mechanism itself. As test system an AMD Athlon 1.4 Ghz system with an ATI Radeon 9700 PRO graphics adapter was used. The scene used for measuring was the car model of Figure 7.

6. Conclusion

We have presented an approach for frame-to-frame coherent screening in image space. The appearance of the screening patterns are independent of camera zoom, object scale, and object orientation and the system advects

Viewport size	256×256	512×512	1024×1024
with G-buffer	238	65	16.4
without G-buffer	295	68	18.6

Table 1. Framerates in frames per second.

the screening matrices in order to link the screening matrices with the object surfaces to maintain temporal coherence. The method is targeted for an implementation on programmable graphics hardware, which leads to real-time frame rates. Since the input to our technique only requires the intensity and the velocity in image space, our approach is also applicable to video sequences. The proposed method can also be applied to solve other, similar problems as, e.g., stated by Cunzi et al. [1]. Thus, other techniques could also benefit from our approach and achieve better results.

In future work, the system could be extended by a better prediction when and where new screening matrices have to be inserted into the blend-in screening mask to achieve a better image quality. Additionally, a greater variety of pen-and-ink rendering styles could be implemented.

References

- [1] M. Cunzi, J. Thollot, S. Paris, G. Debunne, J.-D. Gascuel, and F. Durand. Dynamic canvas for immersive non-photorealistic walkthroughs. In *Graphics Interface*, pages 121–130, June 2003.
- [2] R. Fernando. CgFX overview. <http://www.developer.nvidia.com/Cg>, 2003. Nvidia Corporation.
- [3] B. Freudenberg, M. Masuch, and T. Strothotte. Real-time halftoning: A primitive for non-photorealistic shading. In *Proceedings of the 13th Eurographics Workshop on Rendering Techniques*, pages 227–231, 2002.
- [4] B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. AK Peters Ltd, 2001.
- [5] S. Green, D. Salesin, S. Schofield, A. Hertzmann, P. Litwinowicz, A. Gooch, C. Curtis, and B. Gooch. *Non-Photorealistic Rendering*. SIGGRAPH 99 Course Notes, 1999.
- [6] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *NPAR 2000: First International Symposium on Non-Photorealistic Animation and Rendering*, pages 7–12, 2000.
- [7] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-Eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):211–222, 2002.
- [8] P.-M. Jodoin, E. Epstein, M. Granger-Piché, and V. Ostromoukhov. Hatching by example: A statistical approach. In *NPAR 2002: Second International Symposium on Non-Photorealistic Animation and Rendering*, pages 29–36, 2002.

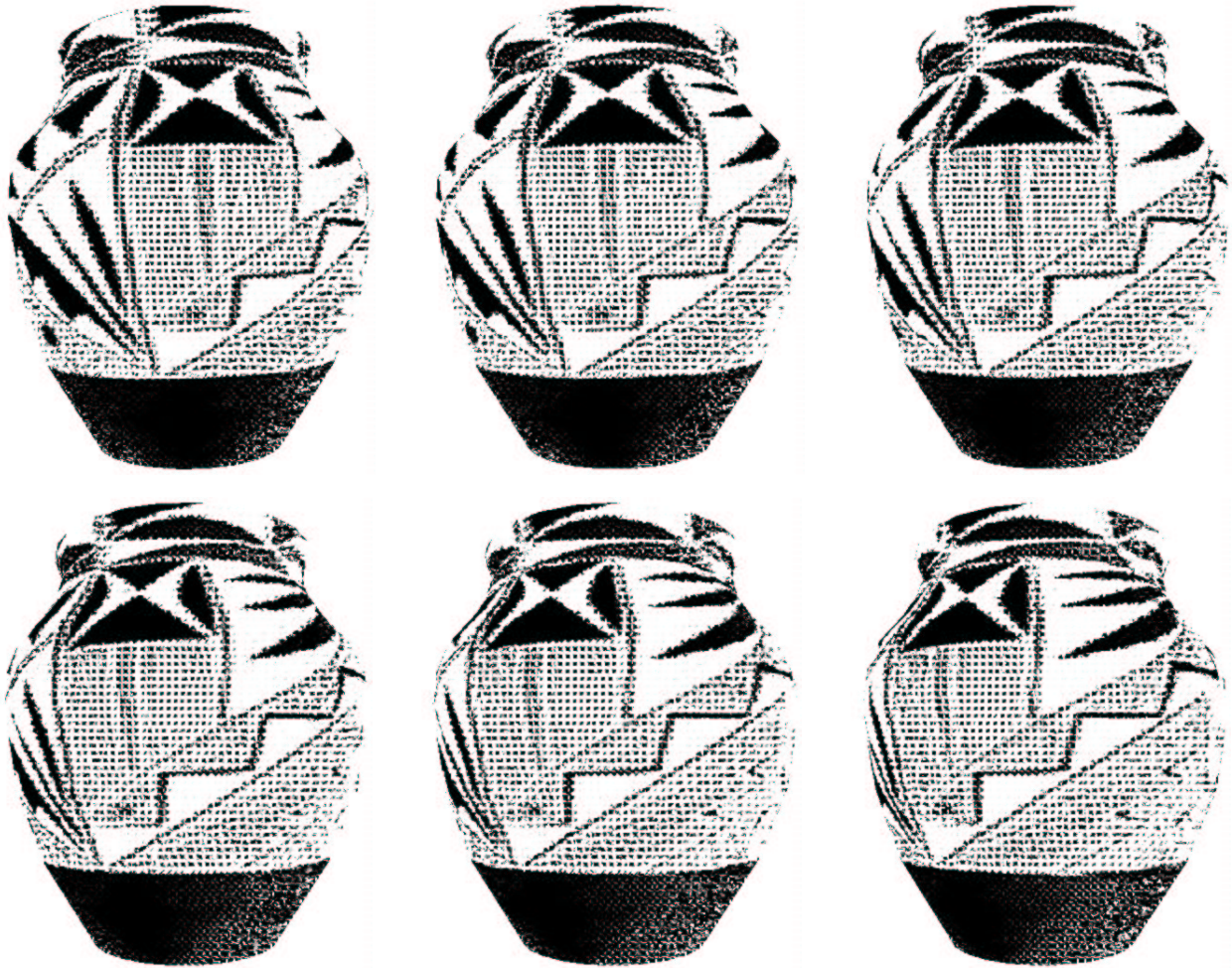


Figure 6. Screening illustration of a rotating pot.

- [9] O. E. Meruvia Pastor and T. Strothotte. Frame-coherent stippling. In *Proceedings of Eurographics (Short Paper)*, volume 21, pages 145–152, 2002.
- [10] Microsoft. DirectX 9.0 programmer's reference. <http://www.microsoft.com/downloads>, 2002. Microsoft Corporation.
- [11] V. Ostromoukhov. Pseudo-random halftone screening for color and black&white printing. In *Proceedings of the 90th congress on advances in non-impact printing technologies*, pages 579–582, 1993.
- [12] V. Ostromoukhov and R. D. Hersch. Artistic screening. In *Proceedings of ACM SIGGRAPH 95*, pages 219–228, 1995.
- [13] V. Ostromoukhov and R. D. Hersch. Stochastic clustered-dot dithering. *Journal of Electric Imaging (Special issue on Color Imaging)*, 8(5):439–445, 1999.
- [14] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, pages 579–584, 2001.
- [15] T. Saito and T. Takahashi. Comprehensible rendering of 3-D shapes. In *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, volume 24, pages 197–206, 1990.
- [16] A. Secord. Weighted Voronoi stippling. In *NPAP 2002: Second International Symposium on Non-Photorealistic Rendering*, pages 27–43, 2002.
- [17] A. Secord, W. Heidrich, and L. Streit. Fast primitive distribution for illustration. In *Proceedings of the 13th Eurographics Workshop on Rendering Techniques*, pages 215–226, 2002.
- [18] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics – Modeling, Rendering, and Animation*. Morgan Kaufmann, 2002.
- [19] R. Ulichney. A review of halftoning techniques. In *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts V*, volume 3963 of *Proceedings of SPIE*, pages 378–391, 2000.
- [20] J. J. van Wijk. Image based flow visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.
- [21] O. Veryovka and J. Buchanan. Halftoning with image-based dither screens. In *Graphics Interface*, pages 167–174, 1999.
- [22] D. Weiskopf, G. Erlebacher, M. Hopf, and T. Ertl. Hardware-accelerated Lagrangian-Eulerian texture advection for 2D flow visualization. In *Workshop on Vision, Modeling, and Visualization VMV '02*, pages 77–84, 2002.

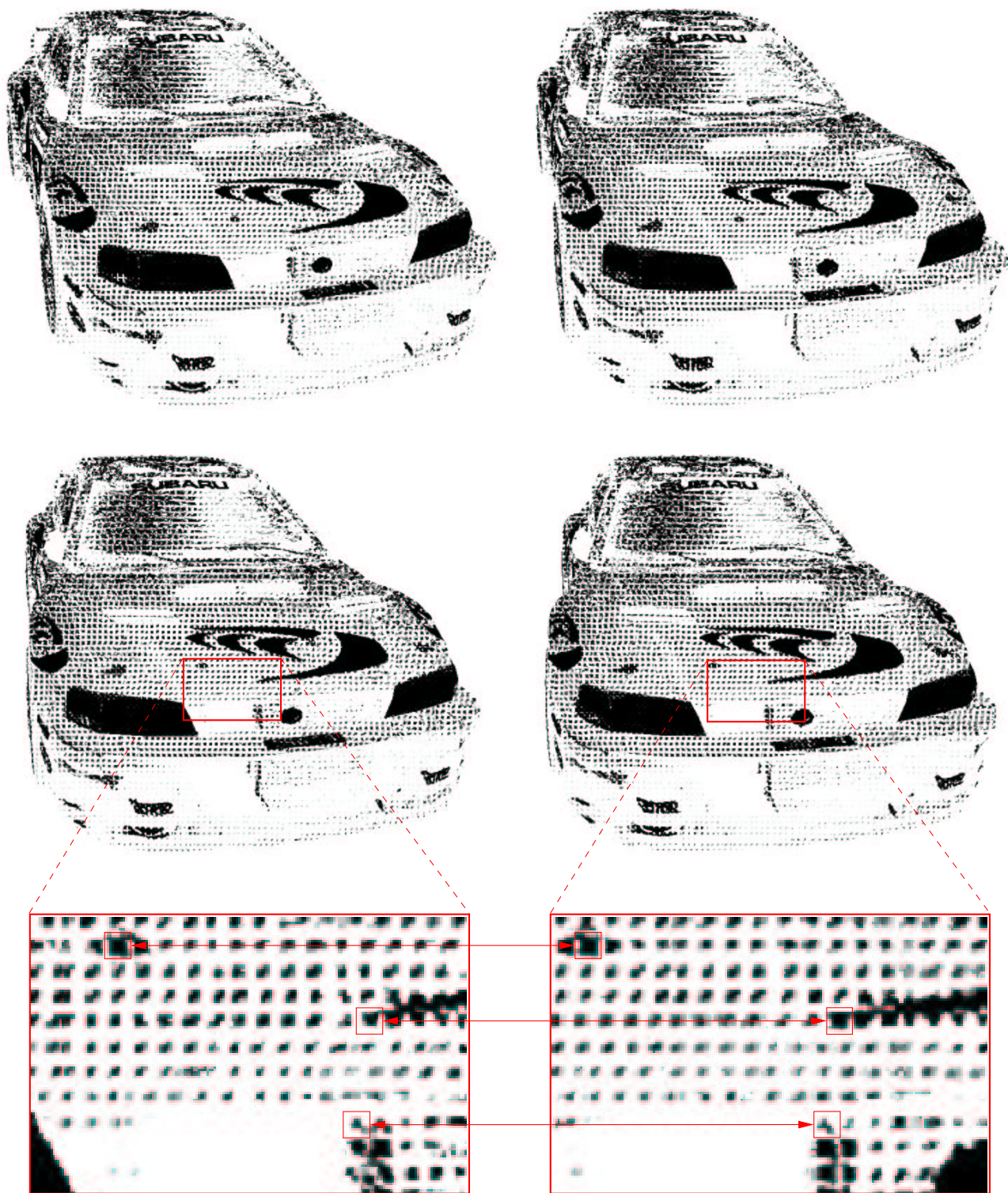


Figure 7. Sequence of a computer-generated animation.
