Stylized Shadows

Christopher DeCoro Forrester Cole Adam Finkelstein Szymon Rusinkiewicz

{cdecoro, fcole, af, smr}@cs.princeton.edu
Computer Graphics Laboratory
Princeton University

(a) Accurate Shadow (b) Shadow in Fine Art (c) Stylized Shadow

Figure 1: *Traditional computer graphics algorithms produce "accurate" shadows (left). Artists often deliberately render abstract shadows, such as the shadow with reduced contour detail in this painting by Vanderlyn, 1818 (middle). Our system offers controls for creation of stylized shadows (right).*

Abstract

While much research has focused on rendering physically-correct shadows, a "correct" shadow often exhibits unnecessary detail that distracts from the primary subject of the scene. Artists often prefer to have creative control over the rendered appearance of the shadow. This paper presents an algorithm offering control over stylized shadows, based on four intuitive parameters – *inflation*, *brightness*, *softness*, and *abstraction* – that together support a broad range of effects. The algorithm, which works largely in image space, can easily be incorporated into existing rendering pipelines, and is independent of scene geometry or shadow determination method.

1 Introduction

Shadows play a significant role in our perception of the world. They provide cues for light direction and atmospheric conditions (whether sunny, cloudy, dusty, etc.), as well as position and shape cues inherent in projections of an object that augment that of the viewing transformation. Works of art and animation commonly include shadows to provide such cues, which are particularly important when the viewer lacks 3D cues such as stereo and parallax. In cel animation and other media where a character and background may be rendered in different styles, shadows play the crucial role of anchoring the character in the scene by showing where the character's feet meet the ground. Finally, shadows offer a mood cue for dramatic effect (cheerful, spooky, etc.), with a well-developed vocabulary for cinematographers [Alton 1949].

Artists typically avoid the use of physically accurate ("correct") shadows for several reasons. First, correct shadows may be difficult to render. In addition, even in situations in which they are easy to produce (e.g. computer graphics applications using global illumination solvers) the resulting shadows may be too detailed and therefore distract from the subject. Fortunately, people tend not to notice or care whether shadows are physically correct. This provides the artist a large design space ranging from "plausible" to obviously abstract. See Figures 1b and 2 for examples from fine art.

This paper describes a method for stylizing shadows rendered in 3D scenes (Figure 1c). To control the visual qualities of the resulting shadow, we introduce four parameters that in combination can provide a broad range of stylization effects. They are:

1. *Inflation i* controls the size of the shadow, relative to the original, such that increased *i* gives the effect of a shadow emanating from a larger version of the shadow-casting object.
2. *Brightness b* is the intensity of the shadow region when fully occluded (or the effect of indirect illumination).
3. *Softness s* indicates the width of the transition region from fully occluded to fully visible, simulating the effect of an area light.
4. *Abstraction* α is a measure of shadow's accuracy; lower values yield more detailed, accurate shadows, whereas larger values produce rounder, simplified shadows.

We present an image-based algorithm that uses these parameters to produce stylized shadows from a shadow matte, as computed by standard techniques. Our implementation, which makes use of efficient Monte Carlo sampling techniques implemented on graphics hardware, provides interactive control of the light, camera, and shadow parameters, affording the artist a fluid environment for exploring this design space. Furthermore, because the

Figure 2: Artistic Stylized Shadows. *Artists regularly make use of shadow stylization for compositional purposes, as can be seen in these examples (by, clockwise from top-left: Fra Carnavale, Warhol, Nicholson, and Eddy). Note the use of highly simplified shadows in the left images, used to allow for shadow cueing, without adding distracting detail from the foreground object; in the bottom-left image, only a simple (though softened) circle is used. In the right images, distinct umbra and penumbra regions are shown, with varying stylizations in each.*

algorithm and controls are image-based, the design space itself affords more intuitive navigation for artists than would emerge from more conventional cinematic controls attached to lights in 3D. Another benefit of the approach is that it works for general visibility-determination methods, from shadow volumes [Crow 1977] and shadow maps [Williams 1978] used in a scanline-rasterization pipeline, to raytraced visibility used in an offline system.

Applications for this work include computer generated imagery for movies, cartoons, games, industrial and architectural design – essentially any context in which an artist is involved in the composition of the scene. For such applications artists in practice already use a variety of ad-hoc methods for abstracting shadows, for example by blurring either shadow maps or the resulting mattes, or by rendering shadows from pre-simplified or stand-in geometry. Thus the main contribution of this work is to provide a set of intuitive controls that work well in concert, together with an efficient algorithm that implements these controls such that they may be explored in an interactive setting.

2 Related Work

In art and cinematography, the interplay of light and shadow has a long-standing tradition for dramatic effect. In both live action and computer generated film, artists have employed a broad set of tools to compose abstract representations of shadows [Barzel 1997; Lowell 1992]. For example, a simple trick that has been used is to replace the "true" shadow casting geometry with a simpler form that is meant to suggest that geometry. In live action the lighting director may make use of a "gobo" (or "cookie", or "blocker") – a card with cutout shapes attached to the light and thereby casting shadows into the scene.

Where shadows have been used in traditional cel animation, they have been hand drawn, typically as light, blobby shapes that serve to anchor the characters in the scene without distracting from the action. Petrovic et al. [2000] developed a method for semi-automatic creation of shadow mattes for cel animation, based on the hand drawn artwork in the scene. While not directly concerned with abstraction, the shadows resulting from their system tend to be abstract for two reasons. First, they are based on abstract characters. Second, they are produced by casting lights from simple geometry created by "inflating" the hand drawn artwork. Central to our process for abstracting shadows is an inflation algorithm, which though different from that of Petrovic enjoys the property of creating smooth shadow mattes.

In computer cinematography, the lighting director applies a spectrum of tricks to control shadows through lighting for various artistic goals [Barzel 1997], for example making many or all lights respect the shadow map of the "key" light (to simplify cast shadows), "cheating" the shadow map away from the key light (to position the shadow for better dramatic composition), or inserting invisible, often simplified, geometry into the scene (for the sole purpose of creating shadows that would not otherwise appear). A recent system based on deep frame buffers allows artists to *interactively* explore the broad space of effects available from complex CG lights [Pellacini et al. 2005]. The method proposed here might well complement such a system, as it focuses specifically on interactive control of the abstraction of shadows. Finally, in computer cinematography a variety of image processing techniques are commonly used to adjust shadows in the final compositing stage, for example blurring the shadow matte to create softer shadows. Our framework also supports such effects, but in such a way that they can respond to geometric properties of the scene. For example, we support blurring less near occluders to simulate a narrowing penumbra.

Researchers have described various methods for controlling lighting parameters by directly manipulating the resulting shadows. For example, the method of Poulin and Fournier [1992] allowed a designer to transform shadow volumes in wireframe view, while that of Pellacini et al. [2002] let the artist drag shadows directly in an interactively rendered image. A number of researchers have developed techniques for optimization of lighting parameters in order to achieve various design goals (including shadows), e.g. [Kawai et al. 1993; Shacked and Lischinski 2001; Gumhold 2002; Lee and Hao 2006]. Together, these systems provide an array of intuitive controls for positioning and shaping shadows, but are not directly concerned with stylization or abstraction, the focus of this paper. While not considering stylization, the previous work of [DeCoro and Rusinkiewicz 2007] has also considered post-render processing of the visibility buffer, as in our approach. This is done mostly for increased performance (visibility buffers are rendered at low resolution, and then resampled), but this work also demonstrates that the resampling step can be used to soften shadows.

3 Stylized Shadows Algorithm

For a scene with l directional lights, the standard method for computing the direct, shadowed illumination of a point x is:

$$L_o(x,\vec{\omega}_o) = \sum_l \rho(x,\vec{\omega}_l,\vec{\omega}_o) S_l(x) L_l(\vec{\omega}_l), \qquad (1)$$

such that the exitant radiance L_o from x in the direction $\vec{\omega}_o$ is computed from the reflectance ρ, incident radiance L_l, and a shadowing function $S_l : x \to [0,1]$. This scale factor denotes the proportion of lighting received from the l-th light, and in real-time applications is commonly the binary visibility $V_l(x)$ of the light relative to the point x. $V_l(x)$ is commonly called the *shadow matte*.

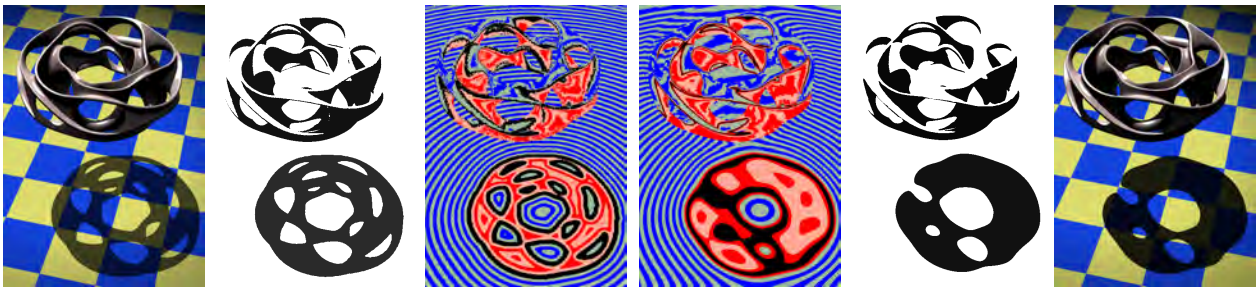


Figure 3: Overview. *Instead of an accurate shadow (left) our algorithm produces a stylized shadow as follows: (1) with the visibility computation resulting from a conventional shadow algorithm, (2) a signed distance transform to the shadow boundary is found – red and blue stripes denoting negative and positive isovalues; (3) a blur is applied, followed by (4) a transfer function, such as a simple threshold, yielding a shadow matte which is used in (5) lighting the scene.*

We propose an alternative formulation of $S_l(x)$ for providing stylistic control over the appearance of shadows, by defining an operator on $V_l(x)$. Our algorithm applies the control parameters in five steps for each light l (shown in Figure 3):

1. Render the visibility buffer $V_l(x)$ corresponding to the l-th light.
2. Compute a signed L_p-averaged distance transform $D(V_l)$
3. Filter D with a Gaussian G, producing $G \otimes D(V_l)$
4. Apply a transfer function f, yielding $S_l(x) = f(G \otimes D(V_l))$.
5. Light the scene with $S_l(x)$ according to Eq.1

3.1 Control Parameters

Inflation: We first consider the inflation parameter i. By increasing the value of this parameter, we approximate inflating the original mesh by inflating the shadow represented in the matte, or rather, creating an offset curve at distance i from the original shadow. We compute a distance transform D applied to V, such that $D(V(x))$ is equal to the distance (according to a metric we define shortly) from x to the original shadow contour. This contour is intuitively the boundary between shadowed and unshadowed regions as indicated by V. Therefore the isocontour $D(V(x)) = i$ is equivalent to an inflation at distance i of the hard shadow, which we can represent by applying a threshold transfer function $f_i(D) = \text{threshold}_i(D)$. Note that once computed, this representation allows interactive modification of i, as we only require recomputing f from $D(V)$, rather than recomputing $D(V)$ itself. To allow for deflation, as well as inflation, we consider D as a signed transform by negating $D(V(x))$ for all points x such that $V(x) = 0$ (points that are shadowed in the original shadow matte).

Were we to use the standard Euclidean L_2 metric, however, offset curves would display cusps and other sharp artifacts due to the discontinuous nature of the distance field in the area of the medial axis. We would prefer a distance transform that is smooth, and thus we adapt an existing method proposed in the literature for inflation of meshes.

Previous work by Peng et al. [2004] produced a surface representation of layered concentric shells, suitable for producing inflated or deflated representations. Unlike the naïve approach of displacement along the normal (corresponding to the L_2 metric) shells generated with this method are guaranteed to be free from cusps and other visually unpleasant artifacts. The method is based on the *L_p-averaged distance metric* defined over R^3 relative to a surface. The parameter p allows for a trade-off between smoothness (with small values of p) and approximation to the Euclidean distance metric, which it approaches as p goes to infinity. While [Peng et al. 2004] addressed surfaces in R^3, we limit our use of this metric to curves defined in R^2, producing:

$$D_p(V(x)) = \left(\int_C \frac{1}{|x-y|^p} dy \right)^{-1/p} \Bigg/ \left(\int_C dy \right)^{-1/p} \qquad (2)$$

in which C is the shadow contour in V, and where the denominator is a normalizing term for the integral. In practice, we have found that $p = 8$ provides a practical trade-off between smoothness and accuracy, and is used for the results in this paper. Unlike L_2, this metric produces a distance transform D that is free from discontinuities and medial axis effects. Furthermore, the average transform exhibits better temporal coherence than the the Euclidean transform, since it is not as sensitive to changes in the Voronoi structure of the sample points. In Figure 4 we show a comparison of the isocurves for the Euclidean L_2 transform and the L_8-averaged

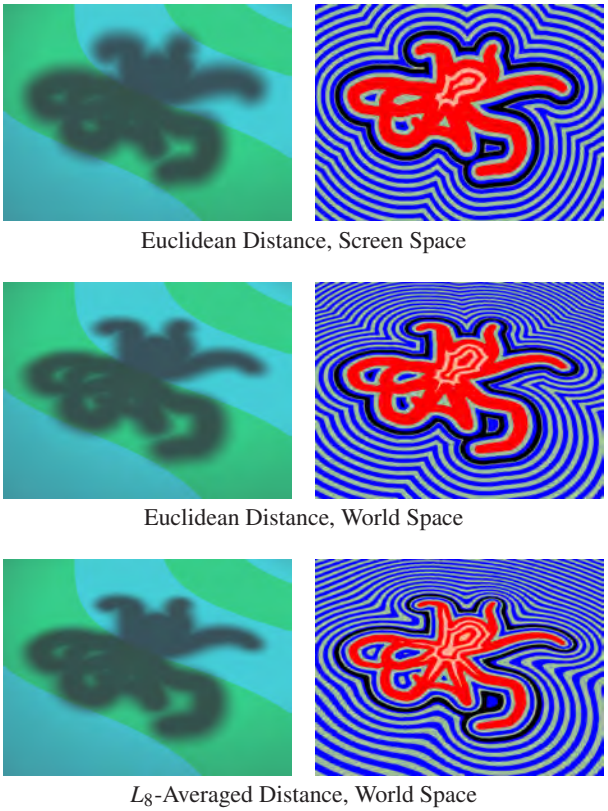


Figure 4: Distance Transforms. *Distance transform of the original hard shadow matte (red region) of the Octopus scene (shown later in Figure 9). The black line represents the inflated shadow contour. The Euclidean metric in screen space displays discontinuities in the soft shadow contours, and does not account for perspective foreshortening or the orientation of the ground plane. While the Euclidean metric in world space corrects the latter, unpleasant discontinuity artifacts are clearly noticeable. The L_8-averaged metric in world-space corrects both problems (Section 3.2).*

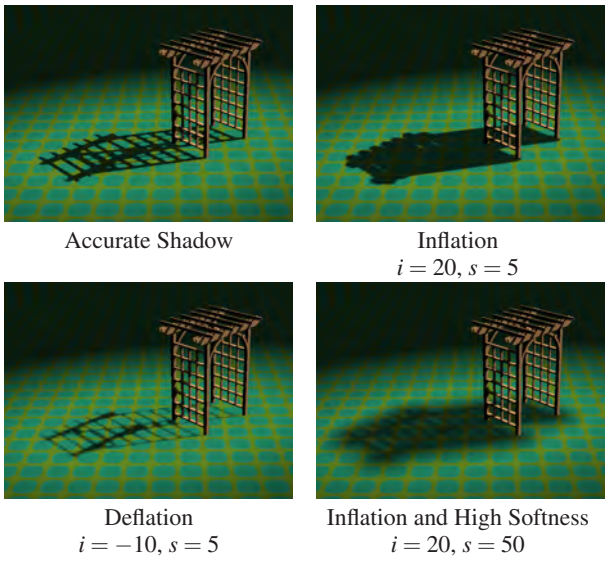


Figure 5: Inflation. *From the original shadow matte, we can produce alternate shadows of various inflations, either using a constant value of i, or varying i according to the approximate occluder distance.*

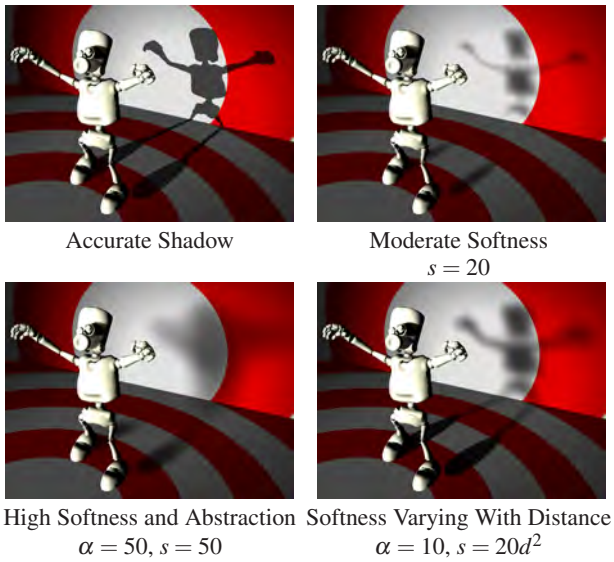


Figure 8: Softness. *We can approximate the effect of a soft area light by changing the softness parameter. The bottom-left figure uses geometric information to harden shadows near shadow casters as discussed in Section 3.2*

transform. Figure 5 then demonstrates the effect of varying the inflation parameter. Note that as the shadow is inflated, it maintains a smooth, visually appealing shape.

Brightness: We will use the brightness term, b, to represent the effect of ambient lighting in the shadowed region. Implicitly, if there exists some light reaching the area fully in shadow, it has reached it through indirect means. Brightness represents the "darkest" a shadow can be, or in our framework the minimum of the transfer function f and therefore the lowest visibility. Note that, without loss of generality, we do not assign a parameter for the maximum value of f, which is always 1; an object outside of the shadow is fully visible to the light. To attenuate the lighting at that object, one would either inflate the shadow, or correspondingly decrease the power of the light.

We can use the brightness parameter to combine multiple lights, which generalizes to multiple regions of the same shadow, such as a highly stylized umbra and penumbra. This effect can be achieved using a low brightness shadow for the umbra, and an inflated or softened shadow with higher brightness. When visibilities are computed separately, and used to modulate lighting that is accumulated in the final rendering, this produces effects reminiscent of painted shadows (see the Warhol and Nicholson works in Figure 2).

Figure 6: Umbra and Penumbra

Softness: After applying the inflation and brightness parameters, we are still left with hard shadows, possibly inflated or deflated from the original. The softness parameter is used to add continuous variation in intensity to the shadow rendering.

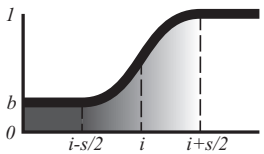


Figure 7: Transfer Function.

In physically-based rendering, the softness of a shadow is proportional to the area of the light source; rather than viewing $V(x)$ as a binary-valued function in which a light is either visible or occluded relative to a point, visibility instead varies continuously across the penumbra region. We can consider then, two separate shadow contours: one deflated from what would be the location of the hard shadow, delineating the umbra from the penumbra, one inflated, delineating the outermost boundary of the penumbra.

Using our distance transform representation, we can extract these contours directly, and because of our use of the smooth L-averaged metric, the shadow intensity will vary smoothly between the two contours. Therefore, we can define softness as the width of the transition from $f(D) = b$ to fully visible $f(D) = 1$.

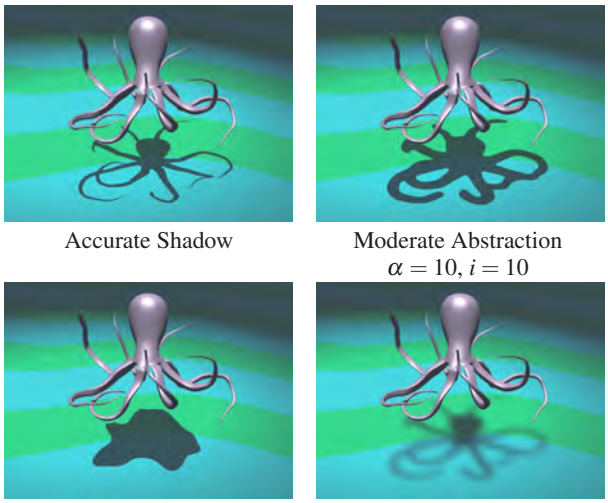
Given these parameters, we can now specify our transfer function. We assume a monotonic falloff from the center of the shadow outwards, and prefer that the function be at least C^1 continuous to avoid visual discontinuities. We therefore choose as a transfer function the following, where smoothstep is a C^2 Hermite polynomial as commonly defined in shader languages. In Figure 8 we can see the effect of changing softness.

$$f(D) \quad = \quad (1-b)^{-1}\,\text{smoothstep}(D, i-s/2, i+s/2)+b. \quad (3)$$

Abstraction: We define least abstract shadows to be the original hard shadows, and the most abstract to be a perfect circle. Therefore, we will define the abstraction parameter α as a limit on the curvature detail of the visibility isocurves. As the abstraction parameter increases, the isocurves lose sharp detail and become rounder.

We implement abstraction by convolving the distance function with a Gaussian kernel of standard deviation α. Under appropriate conditions (away from the medial axis) it can be shown that this has the effect of applying a low-pass filter to the isophote (contour normal) curvature, in which nearly all of the high-frequency curvature detail has been attenuated.

Figure 9 shows the result of filtering the L_8-averaged isocurves seen previously in Figure 4, causing the previously sharply-curving shadows to become smoother and more rounded and leading to a "blobby" or cartoon appearance. However, note that they still maintain their hard transition from light to dark and do not appear blurred, thus mimicking the shadows frequently seen in traditional cartoons.



Accurate Shadow

Moderate Abstraction
$\alpha = 10, i = 10$

High Abstraction
$\alpha = 70, i = 10$

Abstraction and Softness
$\alpha = 20, s = 20$

Figure 9: Abstraction. *By applying a Gaussian filter to the distance transform, we are able to reduce the high detail in the contours, allowing the later thresholding step to produce rounder, more abstract shadows.*

3.2 Geometric Information

As described so far the algorithm proceeds entirely in screen space without considering any world-space geometric information for visible points x. However, we have found the use of geometric information of the scene to give more intuitive shadow behavior, although it is conceivable that another implementation might strictly use image-space information for stylistic effect. The three additional values our system uses per pixel are the world-space position, normal, and approximate distance to occluder. As a practical matter, these can be computed at rasterization time and stored in separate render targets, and thus add trivial overhead.

World-space Distance Metric: In our computation of the L_8 averaged distance transform $D(V)$, we interpret distances in world space, rather than screen space, by using the stored 3D position of each pixel. This allows for important effects such as proper foreshortening of shadows away from the camera. This effect is clearly noted in Figure 4. Additionally, this prevents the shadowing from areas nearby in screen-space, yet distant in world-space, from bleeding across each other. As can be seen in the illustrations, especially Figures 8 and 9, an unshadowed foreground object may cross a shadowed background without artifact on either object.

Normal Discontinuities: To prevent artifacts resulting from sampling across discontinuities, we apply an angular threshold based on the normal when computing the distance transform and blur. If the angle between the normal of x and any point y is greater than 45°, we ignore y for the purpose of computing the distance metric or blur. This has some intuitive, if not more rigorous, basis as lighting is a function of normal, and so different normals will lead to different illumination. Importantly, it works well in practice, and we have used this for all illustrations.

Non-constant Stylization Parameters: In physically realistic rendering, we note that certain properties of shadows vary relative to scene geometry, such as an increase in softness and brightness of a shadow at greater distances from the occluder. Therefore, we allow our stylization parameters to vary as functions of such geometric information.

Our method would easily allow for functions of arbitrary parameters, though we focus our implementation on functions of the approximate distance of a point x to its occluder, which we denote $d(x)$. To motive our desire to allow the shadow parameters to vary with $d(x)$, we observe that to mimic realistic shadowing, softness should increase with $d(x)$, as the projected area of the occluder decreases relative to the projected area of the light. Brightness should also increase with $d(x)$, as more ambient illumination reaches x. This combined effect is often referred to as "hardening" of the shadow near occluders, and is an important perceptual cue. We demonstrate its use in Figures 8 and 11. We have found that quadratic functions of $d(x)$ allow suitable control; more complex functions could be used as needed.

The determination of accurate distances from points to occluders is not a simple problem; however, we have found that for the purpose of setting shadowing parameters, only rough approximations are required to achieve a wide variety of useful stylistic effects. While the algorithm to compute $d(x)$ is orthogonal to our method as a whole, for the figures shown here we use a simple distance between the shaded point and a coarse occluder geometry represented by a small number (1 to 2) of points or lines that is computed in the shader on rasterization. We stress that other implementations could use more accurate distances, such as those computed with raytracing.

3.3 Monte-Carlo Filtering

Both the averaged distance transform (Step 2) and the Gaussian blur (Step 3) require integration over a potentially large domain of the shadow matte. In order to implement these efficiently, we perform the algorithm in a GPU fragment shader using a probabilistic Monte Carlo approach. Note that while rendering efficiency is a secondary concern to stylistic control, it is nevertheless useful in rapidly configuring parameters in order to develop an artistic style. We use a slightly different approach in each step.

Distance Transform: The L-averaged distance is defined on the distance from a point x to a contour C; however, using a uniform spatial sampling over the entire shadow matte V, the probability of finding a pixel $y \in C$ is low. Furthermore, the distance transform must be defined a significant distance from the shadow boundary, in order to allow for shadows with high inflation or softness, requiring sampling over a large region. Clearly, a uniform sampling of V would be inefficient.

To address this problem, we selectively sample only over pixels on shadow boundaries. Currently, we implement this by transferring the visibility buffer V from the GPU back to the CPU, detecting the edges on the CPU, and sending back to the GPU a list of coordinates of pixels in C. When rendering a pixel $x \in D(V)$, the GPU shader performs Monte Carlo integration by sampling a random set of pixels $y \in C$, and using them to compute the L_8-averaged distance to x.

Gaussian Blur: The Gaussian blur may also need to sample over a large portion of the image; fortunately, however, its effect is limited by the falloff of the Gaussian filter kernel. We combine Monte Carlo integration with a windowed approach: each pixel x will randomly sample a disk of pixels of radius 3α around x, where α is the abstraction parameter and filter standard deviation. We have found that both importance sampling the Gaussian kernel, and using a quasi-random Halton distribution [Press et al. 1992], instead of a uniform distribution has a noticeable effect on reducing sampling noise. Additionally, while combined 1-D filters cannot be used to reduce the number of operations, as the function contains discontinuities in world-space and therefore the kernel cannot be separated, we can perform two 2-D convolutions of $\sqrt{1/2}$ the width to achieve the same result as a larger kernel, with significantly fewer computations. Because of these optimizations, the Gaussian blur causes only minor overhead.

Sampling: By using Monte Carlo integration, our approach allows for a continuous time-quality trade-off; while the performance is good even at high quality, fewer samples can be used for slightly

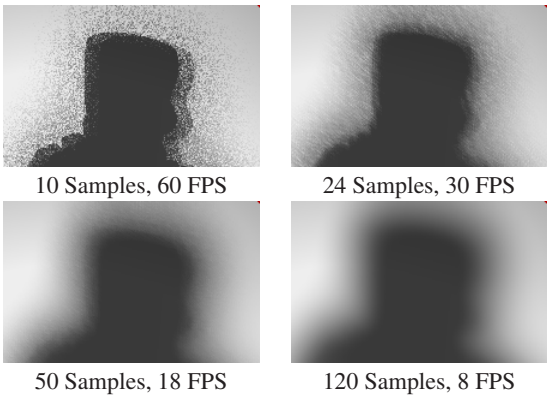


Figure 10: Performance vs. Quality. *As we take fewer samples, we can see that the quality of the image relative to the reference (measured as root-mean squared error per pixel) decreases; however, the frame rate increases significantly as well.*

less accurate results when previewing shadows. Further, we only need to resample the distance transform (the slowest-converging and therefore most time-intensive portion of our algorithm) when the camera or light changes. Inflation, brightness and softness can be modified without refiltering, allowing for fast ($>$ 100 fps for many scenes) modification. Abstraction requires only a recomputation of the rapidly-converging Gaussian blur, and which can be updated in real-time ($>$ 50 fps). Additionally, we can automatically reduce the number of samples while moving camera or light to allow for rapid preview.

We provide performance figures for different sample counts in Figure 10, from an implementation of our algorithm running on a 3Ghz Intel Core CPU, and a GeForce 8800 GPU. This is taken from the "worst-case" situation for our algorithm: moving the light or camera with high softness and no abstraction, which would otherwise remove the noise introduced by the distance transform.

4 Discussion and Future Work

We have presented a system for flexible shadowing that allows a wide array of stylized effects. As seen in Figure 11, the style of the shadows may be adapted according to the control of the artist, and may be used to produce widely different styles of rendering. One immediate direction for future work is an improved algorithm for computing the L-averaged distance transform, which is currently the slowest part of our algorithm. Note that, interestingly, the CPU-GPU communication is not the bottleneck; rather, it is the large number of samples required for low-variance distance computation. This suggests that one solution is to use a biased method that trades off accuracy for a reduction in noise.

We note that many of the choices we presented for our system could be generalized in alternate implementations. For example, we made the assumption that the shadow softness always exhibits a monotonic fall-off from the center, but this need not always be the case. The isocurve renderings in this paper were created by substituting an appropriate stepped transfer function; other artistic effects could be created similarly. Additionally, control parameters could be made functions of the surface material properties, or of the viewing angle to approximate specular reflectance. However, we believe that this implementation provides intuitive control for a majority of applications.

Our implementation contains certain limitations that would need to be addressed for use in a production environment. First, per-object controls over shadow parameters would be required. Moreover, even with support for multiple objects, static parameters for an entire scene might not allow for flexible artistic control; parameters that achieve the desired result for a given combination of light direction and camera angle may be inappropriate for others. Therefore, we would suggest keyframing of the stylization parameters as a reasonable solution, effectively making the parameters a function of time. Practical implementations might also control the parameters as functions of other variables, such as light and camera direction, or as non-linear functions. Finally, although we consider the parameters discussed as the most intuitive, one could consider adding additional parameters, such as anisotropic elongation of the shadow away from the light or additional possibilities for non-photorealistic rendering of the shadows (e.g., outlining, hatching, or stippling).

Another area of future work involves extending shadow stylization to the case of area light sources. While the simplest approach might sample the light and apply our stylization transfer function for each sample independently, this would not be computationally efficient. Instead, we believe that it is possible to express the result of the stylization computation in terms of the area-light shadow intensities themselves, as well as their gradients.

Acknowledgments

This work is partially supported by the Sloan Foundation and the National Science Foundation, grants CCF-0347427 and IIS-0511965. We would like to thank UC Berkeley, AIM@Shape, and DeEspona for providing models used in this work.

References

ALTON, J. 1949. *Painting With Light*. Republished in 1995 by University of California Press, Berkeley, CA.

BARZEL, R. 1997. Lighting controls for computer cinematography. *Journal of Graphics Tools 2*, 1, 1–20.

CROW, F. C. 1977. Shadow algorithms for computer graphics. In *(Computer Graphics) Proceedings of SIGGRAPH 1977*, ACM Press, New York, NY, USA, 242–248.

DECORO, C., AND RUSINKIEWICZ, S. 2007. Subtractive Shadows: A flexible method for shadow level-of-detail. Tech. Rep. TR-781-07, Princeton University.

GUMHOLD, S. 2002. Maximum entropy light source placement. In *Proceedings of IEEE Visualization 2002*, 275–282.

KAWAI, J. K., PAINTER, J. S., AND COHEN, M. F. 1993. Radioptimization - goal based rendering. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 147–154.

LEE, C. H., AND HAO, X. 2006. Geometry-dependent lighting. *IEEE Transactions on Visualization and Computer Graphics 12*, 2, 197–207. Member-Amitabh Varshney.

LOWELL, R. 1992. *Matters of Light & Depth*. Lowel-Light Manufacturing Inc., New York, NY.

PELLACINI, F., TOLE, P., AND GREENBERG, D. P. 2002. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics 21*, 3 (July), 563–566.

PELLACINI, F., VIDIMCE, K., LEFOHN, A., MOHR, A., LEONE, M., AND WARREN, J. 2005. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Transactions on Graphics 24*, 3 (Aug.), 464–470.

PENG, J., KRISTJANSSON, D., AND ZORIN, D. 2004. Interactive modeling of topologically complex geometric detail. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 635–643.

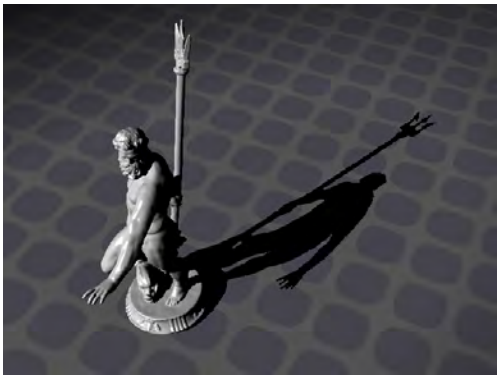
PETROVIC, L., FUJITO, B., WILLIAMS, L., AND FINKELSTEIN, A. 2000. Shadows for cel animation. In *Proceedings of ACM SIGGRAPH 2000*, 511–516.

POULIN, P., AND FOURNIER, A. 1992. Lights from highlights and shadows. In *1992 Symposium on Interactive 3D Graphics*, vol. 25, 31–38.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press.

SHACKED, R., AND LISCHINSKI, D. 2001. Automatic lighting design using a perceptual quality metric. *Computer Graphics Forum 20*, 3, 215–226.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *(Computer Graphics) Proceedings of SIGGRAPH 1978*, ACM Press, New York, NY, USA, 270–274.



Accurate Neptune Shadow

Accurate Filigree Shadow

$\alpha = 13 + 4d - 8d^2$, $i = -2d^2$, $s = 12 - 4d^2$

$\alpha = 20$, $i = 4$, $s = 1$

$\alpha = 5$, $i = -4$, $s = 10$

$\alpha = 7$, $i = -4$, $s = 5$

$\alpha = 20 + 10d$, $i = 5 + 10d$, $s = 50$

$\alpha = 20$, $i = 10$, $s = 25$

Figure 11: Varying control parameters. *The first row shows the hard, unabstracted shadow. The second row shows abstracted but hard shadows. Note that though the shadow is abstract, critical detail (such as the trident) is preserved. In the third row the shadow is shrunk and softened while the character of the shape is preserved. In the final row the shadow is blurred and lightened to remove detail, while maintaining a sense of the light position.*

Stylized Shadows

Christopher DeCoro
Forrester Cole
Adam Finkelstein
Szymon Rusinkiewicz


Accurate Shadow

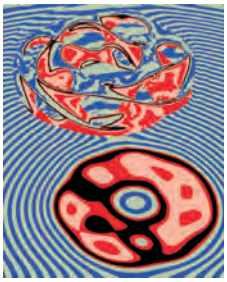

Shadow in Fine Art


Stylized Shadow


Accurate Shadow

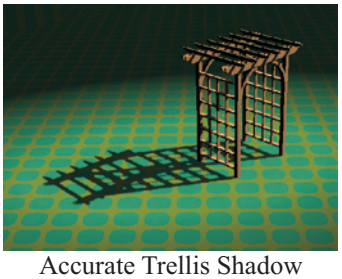

1. Visibility

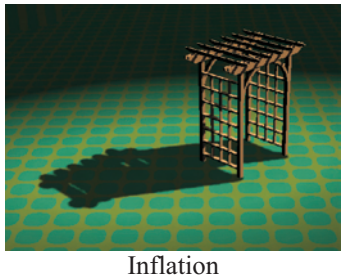

2. Dist. Transform

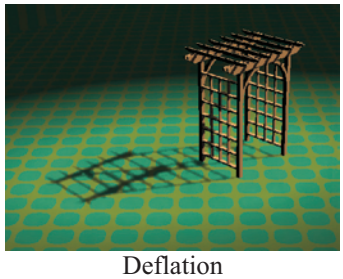

3. Blur

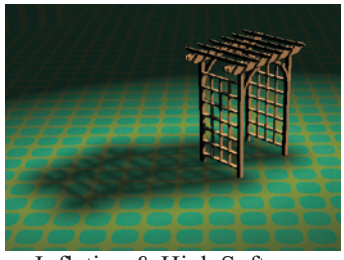

4. Threshold


5. Light


Accurate Trellis Shadow


Inflation


Deflation


Inflation & High Softness


Accurate Robot Shadow

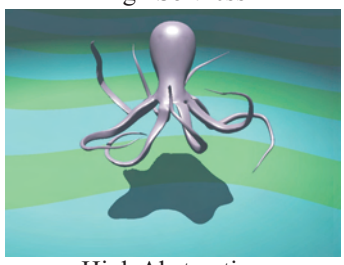

Moderate Softness

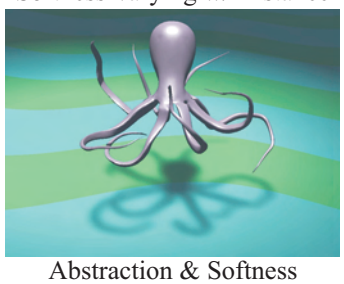

High Softness


Softness Varying w/ Distance


Accurate Octopus Shadow


Moderate Abstraction


High Abstraction


Abstraction & Softness

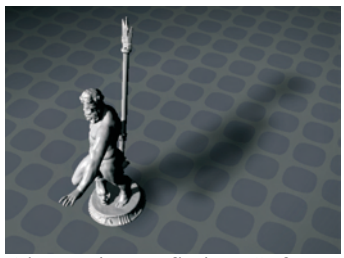

Accurate Filligree Shadow


Abstraction & Inflation


Deflation


Abstraction, Inflation, Softness


Accurate Neptune Shadow


Abstraction, Inflation, Softness


Deflation


Varying Abstraction