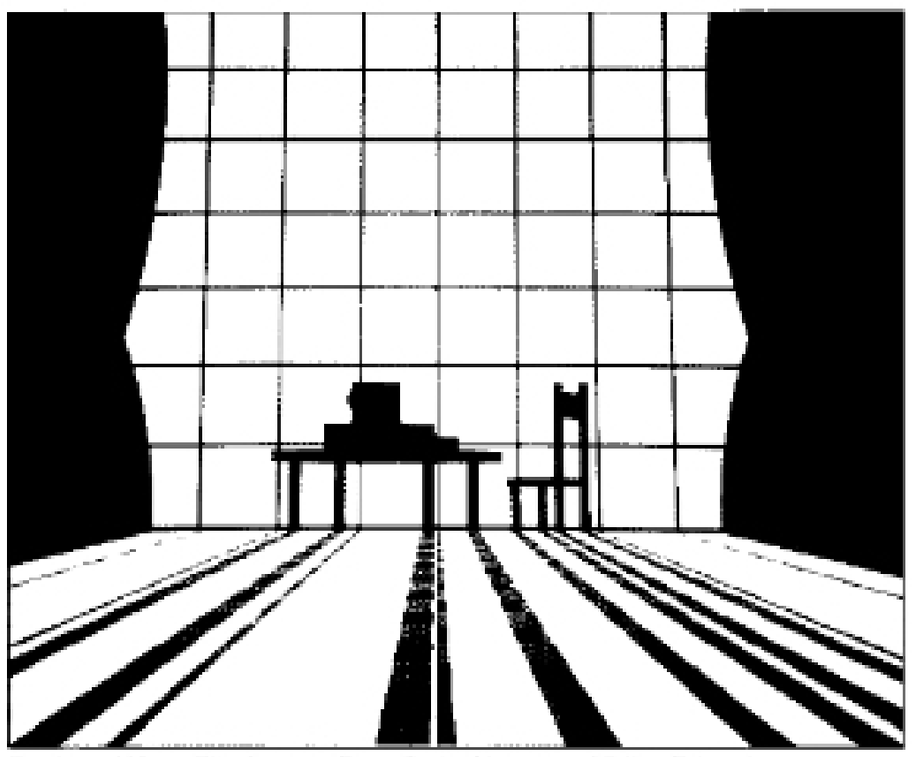
Survey

A Survey of Shadow Algorithms

Andrew Woo, Pierre Poulin, and Alain Fournier
University of Toronto



Courtesy of Mayer Dippé and Pete Shirley, University of British Columbia

Essential to realistic and visually appealing images, shadows are difficult to compute in most display environments. This survey characterizes the various types of shadows. It also describes most existing shadow algorithms and discusses their complexities, advantages, and shortcomings. We examine hard shadows, soft shadows, shadows of transparent objects, and shadows for complex modeling primitives. For each type, we examine shadow algorithms within various rendering techniques.

This survey attempts to provide readers with enough background and insight on the various methods to allow them to choose the algorithm best suited to their needs. We also hope that our analysis will help identify the areas that need more research and point to possible solutions.

A shadow—a region of relative darkness within an illuminated region—occurs when an object totally or partially occludes the light. A transparent object does not necessarily attenuate the light it occludes. In fact, it can concentrate light. However, as is traditional in image synthesis, we will consider a region to be in

0272-17-36 90 1100-0013$01.00 © 1990 IEEE

Table 1. Common notation used in this survey.

Symbol	Definition
E	Average number of edges per polygon
P	Number of points simulating an extended light source
R	Linear resolution of some buffer used
n	Number of primitives in the scene
$p \times q$	Resolution of the image in pixels

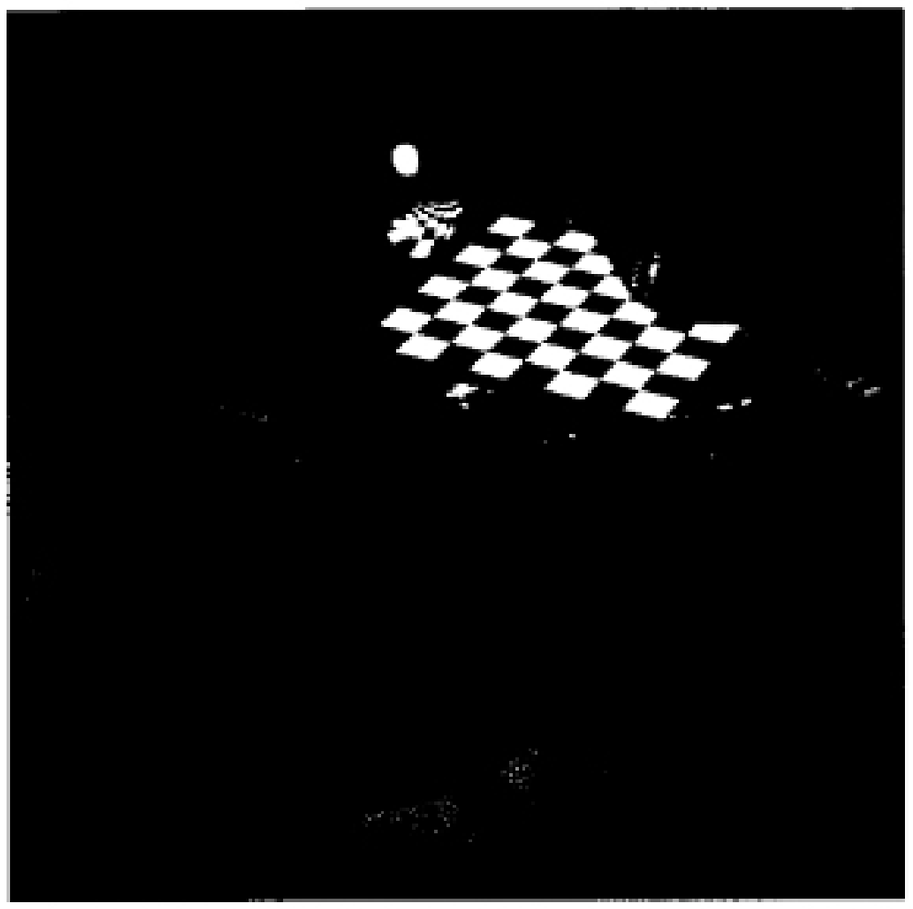


Figure 1. Hard shadows in ray tracing. Ray tracing offers a simple technique to produce sharp shadows. In this image, we used a single directional light source. Notice how the shadow of the table looks sharp and well defined over the objects in shadow.

shadow if there exists an occluding surface between light and the region of interest, whether transparent or not.

A shadow does not necessarily look like a darker region; colored shadows can appear in different ways in a scene. For example, let's say you used two different colored light sources. A region occluded from only one light source by an opaque object will lie in the shadow of this light source, but the color of the second light source can influence the region. An object can also act as a filter, with the color of the shadow region depending on the wavelength spectrum filtered through the object. Moreover, diffraction of light through a transparent object might provide color within a shadow.

Using almost any measure of the quality of an image, the computation of shadows is essential. They cause some of the highest intensity contrasts in images; they provide strong clues about the shapes, relative positions, and surface characteristics of the objects; they can indicate the approximate location, intensity, shape, and size of the light source(s); and they represent an integral part of the total effect in architecture with many objects in the environment. In fact, in some circumstances the shadows constitute the only components of the scene, as in shadow-puppet theater and in pin screen animation (developed by Alexander Alexeieff).

In the only previous comprehensive discussion of shadow algorithms, Crow[1] discussed shadow generation by classifying the type of algorithms used. He distinguished three general categories of shadow algorithms. This useful distinction has since inspired many developments. However, the intervening years have seen the advent of many new modeling primitives and rendering techniques. Within each one, new shadow algorithms have been developed. Amanatides,[2] Thalmann and Thalmann,[3] and Foley et al.[4] described some of the existing shadow algorithms within the general context of image synthesis. However, they dealt mainly with shadows created by opaque objects, and even within this framework their discussion was incomplete.

In this survey, we examine the algorithms according to the type of shadows produced: hard shadows and soft shadows caused by opaque objects, shadows caused by transparent objects, and shadows caused by more complex modeling primitives, such as parametric and implicit surfaces, particle systems, volume-filling media, and surface self-shadowing. Within each type, we examine algorithms as they relate to the specific rendering techniques for which they were developed. We hope that after reading this survey, implementors will be aware of nearly all the existing shadow algorithms and will have sufficient information to choose an algorithm given the type of rendering, primitives, and effects desired.

Of course, many problems related to shadows have not been solved (or solved well). With this in mind, we have tried to identify gaps and suggest improvements to known algorithms, as well as pointing out directions for new ones.

Complexity of algorithms

As a consistent basis for comparison, all shadow complexity order statistics assume a single light source and polygonal surfaces. The complexity consists mainly of three components: storage usage, preprocessing runtime complexity, and runtime complexity during actual rendering (referred to as shadow rendering complexity from here on). The storage complexity is stated with respect to the total storage requirements for shadow determination. The runtime complexities represent the additional cost of shadow computation over implementations that do not compute shadows. The shadow rendering complexity is also stated with respect to each pixel unless otherwise indicated.

Note that, in general, we give worst-case complexity. Whenever practical, we also give an estimate of average complexity, but this is risky without an analysis of scene statistics.

Table 1 summarizes common notation used throughout this survey.

Hard shadow generation

This section discusses shadow algorithms for hard shadows, requiring the display of the umbra section alone (as illustrated in Figure 1). Calculation of hard shadows involves only the determination of whether or not a point in the scene lies in the shadow of opaque objects. This is a binary decision problem on top of the shading model. In other words, multiply a value of either 0 or 1 by the light intensity, indicating in shadow or not in shadow, respectively.

The domain of light sources truly generating hard shadows is restricted to point[5,6] and directional light (see Figure 2).

In general, we can consider hard-shadow determination just as difficult as visibility determination from the eye, because shadow determination is visibility determination with respect to the light source. However, hard-shadow determination is actually simpler to deal with. You do not need to calculate the closest visible object; just determine the existence of an object between the light source and the point of interest.

Fake shadows

The simplest approach employs special-purpose shadow algorithms that work only under certain circumstances. For example, you might use a real-time shadow generator that takes into account only shadows projected on a floor.[7] Such short-cut algorithms

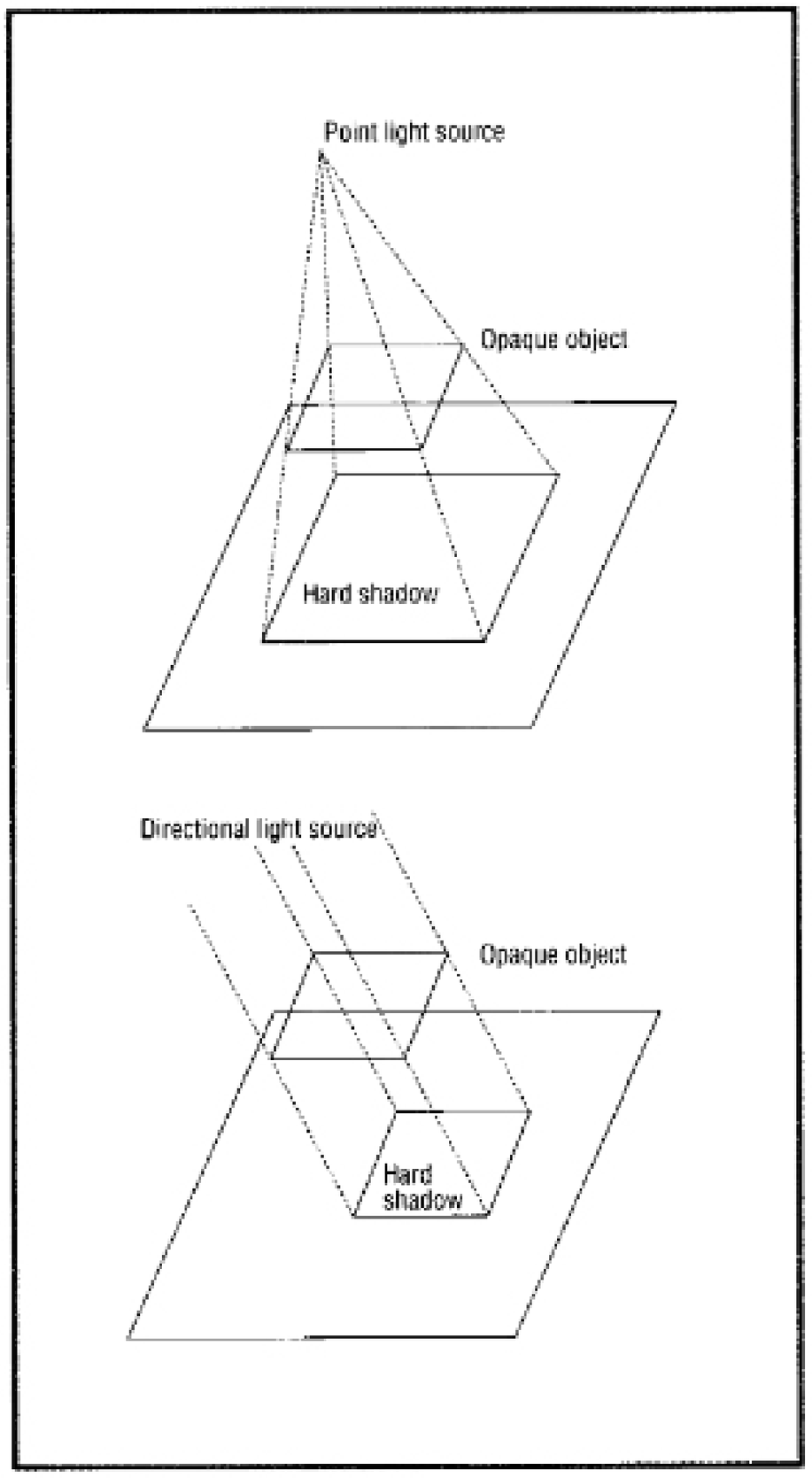


Figure 2. Hard shadows.

tend to be faster computationally than "honest" algorithms (discussed in the upcoming subsections) and can be very effective in the appropriate context, such as video games.

Shadow generation during the scanning phase

Appel[8] and Bouknight and Kelley[9] suggested shadow generation during the display phase using an extended scan-line approach. During the display phase, polygonal boundaries are projected down onto the scanned object to form shadow boundaries, clipped within the boundaries of the scanned object,