Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

Variance Soft Shadow Mapping

Baoguang Yang[1,3] Zhao Dong[2†] Jieqing Feng[3] Hans-Peter Seidel[2] Jan Kautz[4]

[1]Autodesk [2]MPI Informatik [3]State Key Lab of CAD&CG, Zhejiang University. [4]University College London

(a) 131 fps (76k faces) (b) 148 fps (141k faces) (c) 110 fps (120k faces) (d) 25 fps (9700k faces)

Figure 1: *Different rendering results generated by our variance soft shadow mapping method without any precomputation.*

Abstract

We present variance soft shadow mapping *(VSSM) for rendering plausible soft shadow in real-time. VSSM is based on the theoretical framework of* percentage-closer soft shadows *(PCSS) and exploits recent advances in variance shadow mapping (VSM). Our new formulation allows for the efficient computation of (average) blocker distances, a common bottleneck in PCSS-based methods. Furthermore, we avoid incorrectly lit pixels commonly encountered in VSM-based methods by appropriately subdividing the filter kernel. We demonstrate that VSSM renders high-quality soft shadows efficiently (usually over 100 fps) for complex scene settings. Its speed is at least one order of magnitude faster than PCSS for large penumbra.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Color, Shading, Shadowing, and Texture

1. Introduction

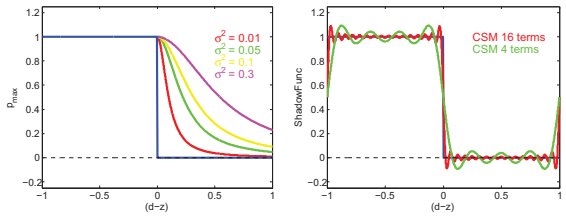
Shadow rendering is a basic and important feature for many applications. However, applications like computer games require shadow rendering to be very efficient — ideally less than 10ms per frame. *Shadow mapping* [Wil78] is a purely image-based shadow method which scales well with scene complexity. Hence it fulfills the strict requirement of game engines and has become the de facto standard for shadow rendering in computer games. While the original shadow mapping method only deals with hard shadows, a variant called percentage-closer soft shadows (PCSS) [Fer05] is sometimes used for rendering soft shadow. PCSS achieves visually plausible quality and real-time performance for small light source. Moreover, its implementation only incurs shader modification and is easy to be integrated into existing rendering system. The PCSS method contains the following main three steps. First, compute the average blocker depth for the current pixel by averaging all depth values within an initial filter kernel that are smaller than current pixel's depth. Then, the average blocker depth is used to compute the penumbra size. Note that PCSS assumes that the blockers/receivers are all planar and in parallel. Using this assumption, the penumbra can be easily computed based on similar triangles. Finally, a loop over all sampling points in the penumbra is performed to do shadow comparisons and to sum up the visibility to get soft shadows. The algorithmic pipeline of the PCSS method can be regarded as a general soft shadow mapping framework based on the planarity assumption.

1.1. Soft Shadowing with PCSS

Following the PCSS pipeline, several pre-filtering soft shadow mapping methods [Lau07, ADM*08] have been recently introduced. *Convolution soft shadow map* (CSSM) is built on pre-filterable shadow reconstruction functions using the Fourier basis. The reconstruction functions with different number of basis terms are shown in Figure 2(b). It is easy to see that the reconstruction curve of CSSM covers the whole range of $(d-z)$ values. Such a *double-bounded* pre-filtering function can be applied for both average blocker depth computation and soft shadow test, and fits very well into the PCSS framework. Yet, large amounts of texture

† Corresponding author: Zhao Dong. The first two authors contributed equally.

memory are required to store Fourier basis terms, making it less practical. Compared to CSSM, *Variance Shadow Maps* (VSM) [DL06] support pre-filtering based on a one-tailed version of Chebyshev's inequality and requires a much lower amount of texture memory. Unfortunately, there is no obvious way to correctly pre-filter average blocker depth values based on the VSM theory. In [Lau07], the average blocker depth evaluation step is therefore performed by brute-force point sampling of the depth map. The shadow reconstruction curve of VSM are shown in Figure 2(a). It is easy to see that this curve only bounds one side of shadow comparison function and is undefined when $(d-z) \leq 0$. Therefore, we call it *single-bounded* pre-filtering shadow function. Existing techniques [DL06] simply assume the shadow value is equal to 1 in this case. When the average depth value z_{Avg} of a filter kernel is bigger than or equal to the depth value d of the current point, this point will be assumed to be fully lit. When handling hard shadow or when the filter kernel is very small, this assumption is reasonable. However, when handling large kernel for soft shadow, this lit-assumption for the whole kernel can introduce incorrect result (lit pixels instead of partially shadowed). We refer to this as the "non-planarity" problem for *single-bounded* pre-filtering shadow functions [Sal08]. Such incorrectly-lit artifacts are more serious when the kernel size increases.



(a) VSM Func (b) CSM Func

Figure 2: *Comparison between different pre-filtering shadow functions. The blue line represents the heaviside step function for the shadow test. d represents the depth value of current point and z represents the depth value sampled from shadow map with a filter kernel.*

1.2. Our Method

Motivated by aforementioned problems, *Variance Soft Shadow Map* (VSSM) is introduced to enable real-time, high-quality soft shadow rendering with low-memory cost. Our key contributions are:

1. Derivation a novel formula for estimating average blocker depth, which is based on VSM theory.
2. An efficient and practical filter kernel subdivision scheme that handles the "non-planarity" lit problem for *single-bounded* VSM shadow functions. The subdivision scheme can be either in uniform way or in adaptive way which is based on linear quad-tree traversal on the GPU. Such a divide-and-rule strategy succeeds in efficiently removing incorrect-lit.

2. Related Work

A complete review of existing shadow algorithms is beyond the scope of this article and we refer the reader to Woo et al. [WPF90], Hasenfratz et al. [HLHS03], and to a recent course [EASW09] for a detailed overview. In this section, the most related pre-filtering shadow mapping techniques and soft shadow mapping methods will be introduced.

2.1. Soft Shadow Mapping with Backprojection

In recent work [AHL*06, GBP06], researchers have transferred ideas from classical discontinuity meshing [DF94] to the shadow mapping domain. Although these backprojection-based methods stem from physically correct theory, crude approximations of blocker geometry may yield either incorrect occluder fusion or light leaking. The work by Guennebaud et al. [GBP07] and bitmask soft shadows [SS07] remove most of these problems, but increase the algorithmic complexity or computation time. More recently, Yang et al. [YFGL09] accelerate backprojection soft shadow mapping by introducing a hierarchical technique, which results in better performance for large penumbra but is still complex for real applications.

2.2. Hard Shadow Mapping with Pre-Filtering

Edge anti-aliasing is a classical problem for hard shadow mapping [Wil78]. Unfortunately, standard filtering cannot be applied directly to the shadow map, because the shadow test has to be carried out before the filtering takes place [RSC87]. Besides brute-force point sampling, several pre-filtering shadow mapping methods [DL06, AMB*07, AMS*08, Sal08] have been proposed recently to solve this problem. The general idea is to transform the standard shadow test function into a linear basis, which then enables the use of readily available filtering functions, such as mip-mapping or summed-area tables (SAT) [Cro84].

The *variance shadow map* (VSM) [DL06] is a probabilistic approach that supports shadow pre-filtering. The shadow test is based on one-tailed version of Chebyshev's inequality which only gives an upper bound of the result. Usually the upper bound value will be directly taken as the shadow test results. The reconstructed function of VSM is shown in Figure 2(a). It is easy to see when the variance σ^2 becomes bigger, the reconstructed results of VSM will become worse. This will produce noticeable high frequency light leaking artifacts for scenes with high depth complexity. Lauritzen et al. [LM08] successfully suppress light leaking by partitioning the shadow map depth range into multiple layers. However, the incorrectly-lit due to "non-planarity" problem still exists since there is no correct definition for the left side of shadow test function.

Convolution shadow maps (CSM) [AMB*07] are based on the same theory as CSSM [ADM*08]. As mentioned in Sec. 1.1, its shadow reconstruction function is "double-bounded". Hence the "non-planarity" problem does not exist

for CSM. Yet, the shadow quality depends on the truncation order and high order will incur impractical memory requirements for storing basis textures.

Exponential shadow maps (ESM) [AMS*08] [Sal08] use the exponential function to approximate the heaviside shadow test function. Since the exponential function is also *single-bounded*, it relies on standard percentage closer filtering (PCF) for "non-planarity" regions [AMS*08].For hard shadows, those regions are small and PCF is efficient enough. However, for soft shadows, these regions become bigger and PCF becomes too expensive.

2.3. Soft Shadow Mapping with Pre-Filtering

One key insight of PCSS [Fer05] is that its step (1) and (3) are based on brute-force point sampling of the depth map. When the area light size becomes large, many sampling points (e.g., 30×30) are required to avoid banding artifacts.Based on PCSS, more efficient prefiltering-based methods have been proposed, such as *convolution soft shadow map* (CSSM) [ADM*08] and *SAT-based variance shadow map* (SAVSM) [Lau07], which we have already discuss in Sec. 1.1. Despite the correct estimation of the average blocker depth in SAVSM (brute-force), it may still show "non-planarity" lit problems.

3. Overview

An overview of the VSSM algorithmic steps are given in Algorithm 1 and we refer to the line numbers as (L*xx*) in the text. First, we generate a normal shadow map and two textures based on it (L2-L4): a summed-area table (SAT) and a min-max hierarchical shadow map (HSM). Then for each visible scene point P, we do the following: Firstly, the initial filter kernel w_i (blocker search area) is computed (L7) by intersecting the shadow map plane with the frustum formed by P and the light source. We then sample the average depth value z_{Avg} in w_i from the SAT texture and the min-max depth range in w_i from min-max HSM. Comparing the depth value d of P with the min-max depth range, we can quickly find the fully-lit and fully-blocked (lit/umbra) scene points and ignore them for following soft shadow computation (L10). Then for the scene points that are left and which are potentially penumbra, our VSSM method checks whether w_i is a "non-planarity" kernel or not. The condition here is whether $z_{Avg} \geq d$. If w_i is not a "non-planarity" kernel, the average blocker depth will be estimated directly using a new formula (L15), which will be introduced in section 4. If w_i is a "non-planarity" kernel, w_i needs to be subdivided either uniformly or adaptively to compute the average blocker depth (L12-L13). The kernel subdivision scheme will be explained in detail in section 5. After getting the average blocker depth, the actual penumbra kernel w_p can be computed (L16). Note, the computation for kernel size in this step is similar to L7, and just the shadow map plane is substituted by the average blocker depth plane. Finally, the variance-based soft shadow value of penumbra kernel w_p can be evaluated either directly or using the kernel subdivision scheme.

Algorithm 1 Overview of VSSM algorithm

```
1  Render scene from light center:
2    Render normal variance depth map
3    Generate summed-area table (SAT) for the depth map.
4    Render the min-max hierarchical shadow map (HSM)
5      for the depth map
6  Render scene from view point. For each visible point P:
7    Compute the initial kernel w_i (blocker search area)
8    Check if P is lit or umbra using the HSM
9    if (P is lit or umbra)
10     return the shadow value accordingly
11   if (w_i is "non-planarity" kernel)
12     Subdivide filter kernel
13     Estimate average blocker depth using novel formula
14   else
15     Estimate average blocker depth using novel formula
16   Compute penumbra kernel w_p based on average blocker depth
17   if (w_p is "non-planarity" kernel)
18     Subdivide filter kernel and evaluate soft shadow value
19   else
20     Evaluate soft shadow value directly
21 Render the final image using the visibility factors
```

4. Variance Soft Shadow Mapping

In this section, we introduce the theory about how to efficiently estimate average blocker depth for VSSM.

4.1. Review of Variance Shadow Maps

Variance shadow maps are based on the one-tailored version of Chebyshev's inequality. Let x be a random variable drawn from a distribution with mean μ and variance σ^2, then for $t > \mu$:

$$P(x \geq t) \leq p_{\max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2} \tag{1}$$

Considering t represents the current point's depth d, and x represents the sampled depth z from the shadow map, the quantity $P(x \geq t)$ in Eq. 1 represents the fraction of texels over a filter kernel that will fail the depth comparison, which is exactly the same as the result of PCF sampling. Since $\mu = E(x) = x$ and $\sigma^2 = E(x^2) - E(x)^2$, $E(x)$ and $E(x^2)$ can be generated on-the-fly to pre-filter the shadow test.

Note, only in the particular case of a single planar occluder at depth d_1, casting a shadow onto a planar surface at depth d_2, the upper bound of Eq. 1 will be equal to the shadow test result. In most other cases, Eq. 1 will not provide an exact value, but a close approximation (Fig. 2).

4.2. Estimating Average Blocker Depth

In order to fit VSM into the PCSS framework, the difficult problem is how to efficiently estimate the average blocker (first step in PCSS, see Sec. 1).

Considering a filter kernel w and the current point's depth t, the pre-filtered depth value z and its square z^2 can be sampled from the VSM. Based on linear filtering, the sampled z is actually the average depth value z_{Avg} in w. The depth

values for all the texels in w can be separated into two categories: (1) the depth values which are $\geq t$ and the average of this kind of depth values is defined as z_{unocc}, (2) the depth values which are $< t$ and the corresponding average value is defined as z_{occ}. Let's assume there are N samples in total in filter kernel w. N_1 of them are $\geq t$ and N_2 of them are $< t$. The following equation holds:

$$\frac{N_1}{N} z_{unocc} + \frac{N_2}{N} z_{occ} = z_{Avg} \tag{2}$$

It is easy to see $\frac{N_1}{N}$ and $\frac{N_2}{N}$ correspond to shadow test results $P(x \geq t)$ and $P(x < t) = 1.0 - P(x \geq t)$. Therefore, Eq. 2 can be written as:

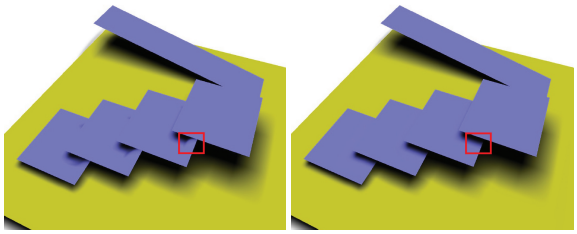
$$P(x \geq t) z_{unocc} + (1.0 - P(x \geq t)) z_{occ} = z_{Avg} \tag{3}$$

Therefore, the average blocker depth value z_{occ} is:

$$z_{occ} = (z_{Avg} - P(x \geq t) z_{unocc}) / (1.0 - P(x \geq t)) \tag{4}$$

z_{Avg} is known and $P(x \geq t)$ can be evaluated based on Chebyshev's inequality. The only unknown variable left is the average non-blocker depth value z_{unocc}. Observing that in the aforementioned two-plane scene setting, $P(x \geq t)$ is accurate and in this case, $z_{unocc} = t$. We therefore assume $z_{unocc} = t$ and use it for general cases as well. This assumption generates high-quality soft shadows in all our experiments.

While it may now seem straightforward to compute the average blocker depth value, the new formula relies on the VSM shadow value $P(x \geq t)$. As mentioned already, the shadow reconstruction function of VSM is just "single-bounded". If $z_{Avg} \geq t$, this will break the prerequisite of Chebyshev's inequality and the "non-planarity" lit problem can occur (Fig. 3(a)). In order to evaluate the average blocker depth correctly, we need to correct $P(x \geq t)$, which we propose to do using subdivision. In the following section, we will explain the kernel subdivision scheme which deals with the "non-planarity" problem (Fig. 3(b)).



(a) Without kernel subdiv. (b) With Kernel subdivision

Figure 3: *Comparison between without kernel subdivision and with kernel subdivision.*

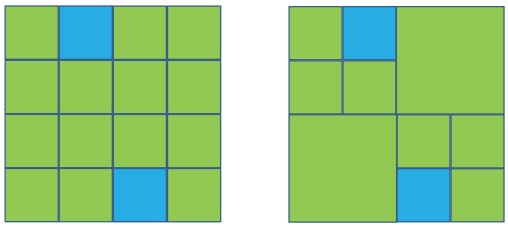
5. Non-Planarity Problem and its Solution

In this section, we introduce the uniform and the adaptive filter kernel subdivision schemes to handle the "non-planarity" problem.

5.1. Motivation for Kernel Subdivision

For an arbitrary filter kernel w of a scene point P, the "non-planarity" problem occurs if $z_{Avg} \geq t$. Here, t represents the current point's depth d. Standard VSM will assume that the shadow value equals to 1 in this case. When w is small, it is reasonable since the depth values of most texels in w are likely to be bigger than t. However, when the size of w increases, only part of the texels in w contain bigger depth value than d. Therefore, the errors due to the lit assumption for the whole w becomes obvious.

Following the concept of divide-and-rule, we propose to subdivide the kernel w into a set of sub-kernels $\{w_{ci}, i \in [1 \ldots n]\}$. Depending on whether $z_{Avg} \geq d$ in w_{ci}, all the sub-kernels can be categorized into two parts. For the normal sub-kernels fulfilling $z_{Avg} < t$, the Chebyshev's inequality still holds and we can compute the average blocker depth and soft shadow based on it. For the "non-planarity" sub-kernels fulfilling $z_{Avg} \geq t$, there are two options: (1) assuming each of them to be lit or (2) using normal PCF sampling to do the shadow test. Option (1) is similar to the previous VSM strategy. However, since the sub-kernel w_{ci} is much smaller than initial kernel w, the "non-planarity" lit problem can be effectively suppressed. Option (2) is a good alternative, since few cheap PCF samplings (2×2) can generate rather accurate results for w_{ci}. In our implementation, option (2) is chosen.



(a) Uniform kernel subdivision (b) Adaptive kernel subdivision

Figure 4: *Illustration of* 4×4 *uniform and adaptive subdivision for filter kernel.*

5.2. Uniform Kernel Subdivision Scheme

Since the corner points of the initial kernel w are known, it is straightforward to subdivide it into equal-sized sub-kernels. As illustrated in Fig. 4(a), the whole quad represents the initial kernel w and each sub-quad inside of it represents a sub-kernel w_{ci}. We loop over each w_{ci} and check whether it is a "non-planarity" kernel or not. In Fig. 4, the blue sub-quad represents "non-planarity" kernels and the green one represents the normal kernels. Here we conceptually separate all the sub-kernels into two groups: the normal sub-kernel group $w_{cj}$and the "non-planarity" sub-kernel group w_{ck}. In following, we will illustrate how to estimate average blocker depth using the uniform kernel subdivision scheme.

Let's first consider the normal sub-kernel group: To compute $P(x \geq t)$ using Eq. 1 for the whole group, the mean

value μ and the variance σ^2 needs to be determined. More specifically, the $E(x)$ and $E(x^2)$ from all the sub-kernels in this group need to be computed. We define the size of w_{cj} to be T_{cj}, and arrive at the following formulas to compute μ and σ^2 for the normal sub-kernel group:

$$\mu = \sum_j (E(x)_{cj} \cdot T_{cj}) / \sum_j T_{cj}$$
$$\sigma^2 = \sum_j (E(x^2)_{cj} \cdot T_{cj}) / \sum_j T_{cj} - \mu^2 \quad (5)$$

During the loop, $E(x)_{cj}$ and $E(x^2)_{cj}$ can be sampled from the SAT texture for each w_{cj}. Then, $E(x)_{cj} \cdot T_{cj}$, the sum of all texels' depth in w_{cj}, is accumulated. The same accumulation happens for $E(x^2)_{cj} \cdot T_{cj}$. Finally, the depth and depth square sums are divided by the accumulated sub-kernel size (Eq. 5) to get μ and σ^2. Hence, the VSM shadow reconstruction function can be evaluated. The average blocker depth d_1 in the normal kernel group can be evaluated using Eq. 4.

For the "non-planarity" sub-kernel group, we apply standard PCF sampling for each w_{ck}: m points are sampled in w_{ck} and the sum of all the blocker depth samples are computed. Since the size of w_{ck} is small, usually $m = 2 \times 2$ is enough. In following steps, both the sum of all the blocker depth and the sum of all the blocker sub-kernel size are accumulated. Similar as before, we can get the average blocker depth d_2 of the "non-planarity" sub-kernel group.

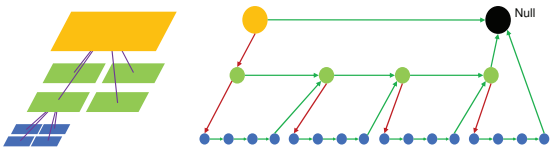
After getting d_1 and d_2, the average blocker depth d over the whole kernel w can be computed by combining d_1 and d_2 weighted by the corresponding blocker kernel size separately. Note there is a reasonable acceleration strategy: the variance σ^2 represents the depth value variation in each w_{ci}. Hence, when the z_{Avg} in w_{ci} is bigger than current depth d, and if the σ^2 is also less than a small *Threshold*, such a w_{ci} can probably be treated as fully-lit.

5.3. Adaptive Kernel Subdivision Scheme

To achieve better subdivision granularity control, we propose an adaptive kernel subdivision scheme, as shown in Fig. 4(b). Compared to the uniform scheme, adaptive kernel subdivision processes sub-kernels in hierarchical way. Its performance is a balance between the hierarchical culling gain and traversal cost. Usually when the number of sub-kernels is large (≥ 64), adaptive subdivision achieves better performance.

Since our filter kernel is always a 2D square, we can construct a quad-tree in the 2D domain (Fig. 5(a)). For the filter kernel w, the root node of the quad-tree represents w itself and each tree node represents a sub-kernel $\{w_{ci}, i \in [1 \ldots n]\}$. Be aware the different w_{ci} are not equal-sized anymore and they could exist in different levels of tree hierarchy. In Algorithm 2, we show the steps for computing the final soft shadow value based on adaptive kernel subdivision scheme.

Standard quad-tree traversal depends on recursive operations, which is not easily implemented on a stackless GPU.



(a) Kernel Hierarchy (b) Linear quad-tree traversal

Figure 5: *Linear quad-tree traversal on 2D filter domain.*

Algorithm 2 Adaptive kernel subdivision algorithm

```
1  Define VSMEx = 0, VSMEx2 = 0, VSMArea = 0
2         PCFArea = 0,PCFBArea = 0,LitArea = 0
3  Start from the root node of kernel w, TreeNode = root
5  While( TreeNode != Null ):
6    TreeNode w_ci is current node
7    Compute texcoords UV and kernel size T_ci
8    Sample the E(x)_ci and E(x^2)_ci from SAT
9    Compute the variance σ^2
10   If (E(x)_ci ≥ d And σ^2 < Threshold)
11     TreeNode = TreeNode.Next
12     LitArea = LitArea + T_ci
13   Else
14     If (E(x)_ci < d)
15       VSMEx = VSMEx + E(x)_ci × T_ci
16       VSMEx2 = VSMEx2 + E(x^2)_ci × T_ci
17       VSMArea = VSMArea + T_ci
18       TreeNode = TreeNode.Next
19     Else
20       If (TreeNode is not a leaf)
21         TreeNode = TreeNode.Child
22       Else
23         Sample m points inside the kernel w_ci of TreeNode.
24         PCFArea = PCFArea + T_ci
25         PCFBArea = PCFBArea + T_ci × m̄/m
26         //m̄ is the number of occluding samples
27         TreeNode = TreeNode.Next
28 End While
29 Compute shadow reconstr. value L from VSMEx and VSMEx2
30 Compute visibility L1 based on PCFArea and PCFBArea
31 Final visibility is computed using L, 1.0 and L1,
32   weighted by VSMArea, LitArea and PCFArea separately
```

We borrow the idea from [Bun05] and successfully apply the GPU-based linear quad-tree traversal for our 2D filter kernel domain. To achieve the linear traversal, each quad-tree node needs to define two pointers (as shown in Fig. 5(b)): 'Child' pointer (red), which points to the first child node, and the 'Next' pointer (green), which points to the next tree node on the linear traversal path. After carefully setting up the 'Next' pointer for each tree node [Bun05], we avoid the usual recursive operation and enables a linear forward traversal on the GPU.

Note, computing soft shadow value needs to consider the fully-lit sub-kernels. In Algorithm 2, we define a variable *LitArea* to record the size of all the fully-lit sub-kernels. In L10-L12, when both $E(x)_{ci} \geq d$ and $\sigma^2 < Threshold$, the current w_{ci} is fully-lit, and its all child nodes can be ignored. So we accumulate the *LitArea* and move to the next

treenode. If $E(x)_{ci} < d$, w_{ci} is a normal sub-kernel. As before, we accumulate the sum of $E(x)_{ci}$ (L15), the sum of $E(x^2)_{ci}$ (L16) and the sum of sub-kernel size T_{ci}(L17). After accumulation, we then move on to the next treenode (L18). Excluding from above two cases, the last case is when $E(x)_{ci} \geq d$ and $\sigma^2 \geq Threshold$. In this situation, we should consider whether the current treenode is a leaf or not. If the current treenode is a non-leaf node, go down the tree hierarchy to its child node (L21). Otherwise, if the current treenode is a leaf node, we resort to use PCF for the visibility computation (L23-L26) as before. When it is done, we move to the next treenode. After the whole traversal is finished, the final visibility can be evaluated (L29-L32). Note, here all the three sub-kernel regions: *VSMArea*, *LitArea* and *PCFArea* are required to compute the final result.

6. Implementations and Discussion

6.1. Min-Max Hierarchical Shadow Map

When generating the min-max hierarchical shadow map (HSM), there are two options: Mip-map and N-buffers [DÓ5]. Mip-maps can be generated very efficiently and also require little texture memory. However, the introduced error tends to be obvious when sampling from high mip-map level. In contrast, N-buffers can ensure accurate min-max sampling results for arbitrary filter kernel sizes with more memory and generation time. For the scene setting of Fig. 1(b), generating a 1024×1024 HSM, takes 3ms with N-buffers and only 1ms with mip-maps. In our experiments, the HSM using mip-maps already works very well, even for complex scenes. Hence we choose mip-maps for HSM generation.

6.2. Number of Sub-Kernels

Applying uniform kernel subdivision to evaluate both average occluder depth and soft shadows, the number of sub-kernels in these two steps can be represented as $m \times m$ and $n \times n$ respectively. If the number is too low, the "non-planarity" sub-kernel will have relatively larger size, so that a low number of PCF samplings (2×2) will not be enough to avoid perceptible artifacts (Fig. 6(a)). In Fig. 6(b), we increase m to 5 and the artifacts are successfully removed. In fact, we find that $m = 5$ works well in most of our experiments. Furthermore, average blocker depth evaluation is less sensitive to precision compared with shadow computation. Hence, n is usually larger than m.

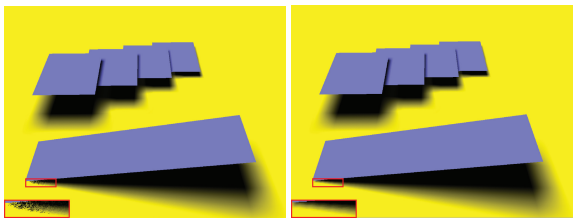
Our adaptive subdivision is based on a full quad-tree of 2D sub-kernels. If the height of the quad-tree is H, there are maximally 4^H leaf nodes corresponding to 4^H sub-kernels. If taking $H = 3$, our experimental results show that the quality of adaptive subdivision is basically the same as for uniform subdivision (Fig. 6(b) and (c)).

6.3. Combining Different Subdivision Schemes

The performance of adaptive subdivision is a balance between the hierarchical culling gain and traversal cost. When the number of sub-kernels is small (like $m = 5$), the traversal cost could hinder the performance. A better strategy is to use uniform subdivision for evaluating average occluder depth ($m = 5$) and adaptive subdivision for the soft shadow ($H = 3$) separately. As shown in Fig. 6(d), such a combination gives the same quality but provides the best performance.

6.4. SAT Precision and Contact shadow

We adopt summed-area tables (SAT) to pre-filter the shadow map. However, it is well known that SAT suffers from numerical precision loss for large filter kernel. Following [Lau07], the 32-bit integer format is used for SAT generation to achieve stable shadow quality. However, in contact shadow areas, where the blocker and receiver are placed very closely, the precision of integer SAT is still not enough and can introduce small errors (Fig. 7(a)) for the average blocker depth z_{Avg}. We observe that in contact shadow areas, the difference between z_{Avg} and d is very small [ADM*08], and hence the corresponding penumbra size is also very small and applying several PCF samplings for shadow is usually sufficient. In our experiments, the contact shadow sub-kernels are detected by checking the difference between z_{Avg} and d. If the difference is smaller than a threshold value ε, a 3×3 jittered bilinear PCF sampling [Bav08] is applied for evaluating soft shadow. In our experiments, $\varepsilon = 0.01 \cdot r$ and r is the bounding sphere radius of input scene. The experimental results demonstrate such a strategy can avoid precision artifacts and generate convincing contact shadows (Fig. 7(b)).



(a) Contact Noise (b) Noise Fixed

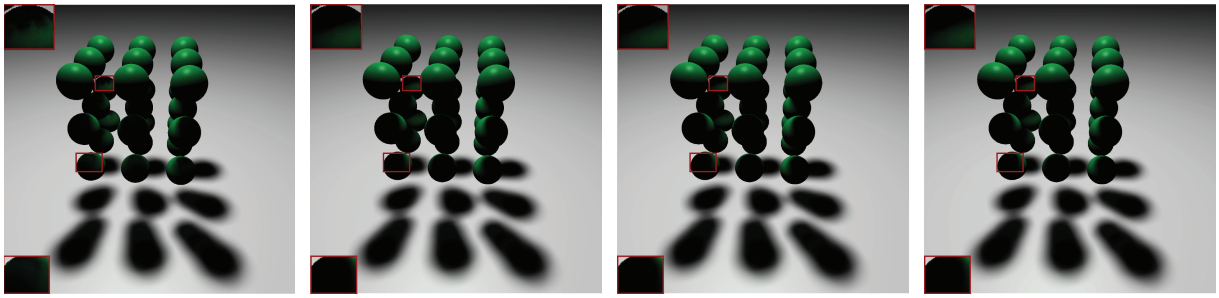
Figure 7: *Fixing contact shadow noise.*

6.5. Threshold Selection

In Algorithm 2, there is a *Threshold* value which is used to identify nearly-planar regions that can be safely marked as fully lit. In all of our tests, $Threshold = 0.0001 \cdot r$ works well and r is the bounding sphere radius of input scene.

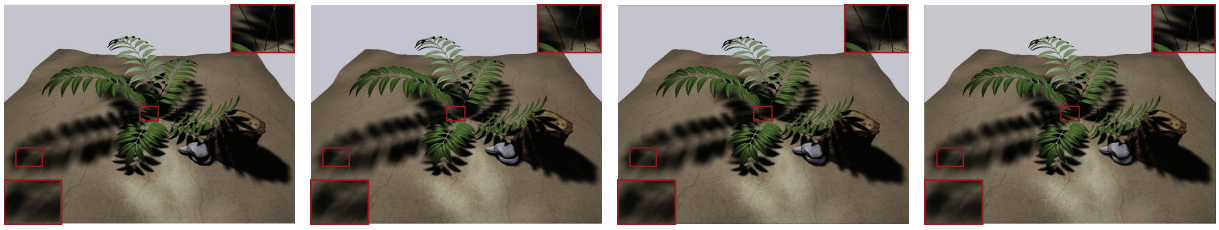
7. Results

Our experiments were run on a PC with a quad-core 2.83GHz Intel Q9550 CPU, an NVIDIA GeForce GTX 285, and 4GB of physical memory. Except for comparison in Fig. 6, all the result images are using mixed kernel subdivision scheme: 5×5 uniform subdivision for estimating average blocker depth, and $H = 3$ adaptive subdivision for computing soft shadow. The screen resolution for rendering is always 1024×768.



(a) $m = 3, n = 8$ (102 fps) (b) $m = 5, n = 8$ (98 fps) (c) $H = 3$ (94 fps) (d) $m = 5, H = 3$ (117 fps)

Figure 6: *Comparison between different subdivision cases. m and n represent the number of sub-kernels when using uniform kernel subdivision. H represent the height of quad-tree when using adaptive kernel subdivision.*



(a) Ground Truth (Ray-Traced) (b) VSSM (148 fps) (c) PCSS (10 fps) (d) Backprojection (19 fps)

Figure 8: *Shadow quality comparison of several methods (SM size* 512×512*, scene has 212K faces): ray-tracing (a), our VSSM method using mixed subdivision scheme (b), percentage closer soft shadows [Fer05] (c), backprojection [GBP07] (d).*

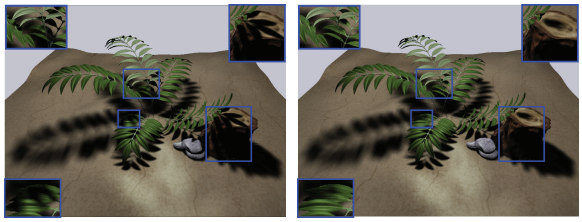
Scene	GBuf	SM	HSM	SAT	Shadow	Total
Balls	2.0	0.1	0.1	2.5	3	7.7
Sponza	2.5	0.6	0.3	4.4	2.0	9.8
Soldier	13.8	10.6	0.3	4.3	3.8	32.8

Table 1: *Performance (milliseconds) breakdown for different scenes (SM:*1024×1024*).*

SMRes	GBuf	SM	HSM	SAT	Shadow	Total
512	1.7	0.17	0.18	2.1	2.5	6.65
1024	2.0	0.25	0.3	4.8	3.5	10.85
2048	2.0	0.5	1.0	17.9	4.4	25.8

Table 2: *Performance (milliseconds) breakdown using different SM resolution for plant scene (141k faces).*

Table 1 provides the performance breakdown for different scenes: Balls (55k faces) in Fig. 6, Sponza (72k faces) in Fig. 1 (a) and Soldier (9700k faces with 100 instances) in Fig. 1 (d). Each row contains timings for: generate G-buffer data (GBuf), render shadow map (SM), generate mipmap HSM (HSM), generate summed-area table (SAT), and soft shadow pass (Shadow). The final column (Total) is the sum of each step's timing. From the data, we can see that SAT usually takes a significant ratio of the running time. The running time of soft shadow pass also depends on how many screen pixels locate in penumbra. In the soldier scene, penumbras appear in many areas, so the soft shadow pass takes more time. Also since the geometric burden in the soldier scene is very high, the GBuf and SM become the bottleneck of rendering. Table 2 provides the performance breakdown for the Plant (142k faces) in Fig. 8 using different SM resolution. With increasing resolution, the SAT becomes the bottleneck and also increases the sampling cost in the soft shadow pass.



(a) VSSM (148 fps) (b) SAVSM (183 fps)

Figure 9: *Shadow quality comparison of between VSSM (a) and SAT-based variance shadow map (SAVSM) [Lau07] (b).*

The result images shown in Fig. 8 compare the shadow quality of several different algorithms including a ray-traced reference image. We analyze two situations in particular, large penumbrae and multiple blockers shadows (close-ups in red squares). Overall Shadows rendered with VSSM are very close to the reference. For the large penumbrae, the results of all methods are close to the reference and just a little

bit of banding can be noticed in PCSS case. In the case of multiple blockers, the difference between our method and the reference becomes more obvious. It is because VSSM is based on planar assumption of PCSS and will average the blocker depth so that the umbra is underestimated. The rendering result of PCSS exhibits the same effect as VSSM. Backprojection method can achieves more physically correct result, but its performance is slow for real-time applications. We further compare VSSM with SAVSM [Lau07], see Fig. 9. All the three close-up regions contain multiple depth layers. Hence, for SAVSM the "non-planarity" lit case happens. The side-by-side comparisons clearly show that our kernel subdivision scheme successfully removes incorrectly lit areas at very reasonable performance cost.

7.1. Limitations

Our VSSM method shares the same failure cases as PCSS. The PCSS method assumes that all blockers have the same depth within the filter kernel. Such a "single blocker depth assumption" essentially flattens blockers. When the light size becomes bigger, this assumption is more likely to be violated and umbrae tend to be underestimated. Furthermore, PCSS only generates one depth map from the center of the light source. When using a single depth map to deal with blockers of a high depth range, single silhouette artifacts [AAM03] may appear. Nevertheless, in most cases, the soft shadow generated by VSSM is visually plausible and looks very similar to the ray-traced reference.

8. Conclusions and Future works

In this paper, we have presented *variance soft shadow mapping* (VSSM) for rendering plausible soft shadow. VSSM is based on the theoretical framework of *percentage-closer soft shadows*. In order to estimate the average blocker depth for each scene point, a novel formula is derived for its efficient computation based on the VSM theory. We solve the classical "non-planarity" lit problem by subdividing the filtering kernel, which removes artifacts. As future work, we would like to apply our kernel subdivision method to exponential shadow mapping [AMS*08], which is also a *single-bounded* pre-filterable shadow mapping method.

Acknowledgements

Part of the research at UCL is funded by EPSRC (EP/E047343/1) and the TSB (Q2047E), and part of the research is funded by the NSF of China (60933007, 60873046), the 973 program of China (2009CB320801).

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. Graph. 22*, 3 (2003), 511–520.

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3 (2008), 1–8.

[AHL*06] ATTY L., HOLZSCHUCH N., LAPIERRE M., HASENFRATZ J.-M., HANSEN C., SILLION F.: Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum 25*, 4 (2006).

[AMB*07] ANNEN T., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Convolution shadow maps. In *Proc. of EGSR* (2007), vol. 18, pp. 51–60.

[AMS*08] ANNEN T., MERTENS T., SEIDEL H.-P., FLERACKERS E., KAUTZ J.: Exponential shadow maps. In *GI '08: Proceedings of Graphics Interface 2008* (2008), pp. 155–161.

[Bav08] BAVOID L.: Advanced soft shadow mapping techniques. In *GDC 2008* (2008).

[Bun05] BUNNELL M.: Dynamic ambient occlusion and indirect lighting. In *GPU Gems 2*. Addison Wesley, 2005.

[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *Proc. of SIGGRAPH '84* (1984), pp. 207–212.

[DÓ5] DÉCORET X.: N-buffers for efficient depth map query. *Computer Graphics Forum 24*, 3 (2005).

[DF94] DRETTAKIS G., FIUME E.: A fast shadow algorithm for area light sources using backprojection. In *SIGGRAPH '94* (1994), pp. 223–230.

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proc. of I3D '06* (2006), pp. 161–165.

[EASW09] EISEMANN E., ASSARSSON U., SCHWARZ M., WIMMER M.: Casting shadows in real time. In *ACM SIGGRAPH Asia 2009 Courses* (Dec. 2009).

[Fer05] FERNANDO R.: Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches* (2005), p. 35.

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time soft shadow mapping by backprojection. In *Proc. of EGSR* (2006), pp. 227–234.

[GBP07] GUENNEBAUD G., BARTHE L., PAULIN M.: High-quality adaptive soft shadow mapping. *Computer Graphics Forum 26*, 3 (2007).

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (2003), 753–774.

[Lau07] LAURITZEN A.: Summed-area variance shadow maps. In *GPU Gems 3*, Nguyen H., (Ed.). 2007.

[LM08] LAURITZEN A., MCCOOL M.: Layered variance shadow maps. In *Proc. of GI* (2008), pp. 139–146.

[RSC87] REEVES W., SALESIN D., COOK R.: Rendering antialiased shadows with depth maps. In *Proc. of ACM SIGGRAPH* (July 1987), pp. 283–291.

[Sal08] SALVI M.: Rendering filtered shadows with exponential shadow maps. In *ShaderX 6.0 - Advanced Rendering Techniques* (2008), Charles River Media.

[SS07] SCHWARZ M., STAMMINGER M.: Bitmask soft shadows. *Computer Graphics Forum 26*, 3 (Sept. 2007), 515–524.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proc. of ACM SIGGRAPH* (1978), pp. 270–274.

[WPF90] WOO A., POULIN P., FOURNIER A.: A survey of shadow algorithms. *IEEE Computer Graphics & Applications 10*, 6 (1990), 13–32.

[YFGL09] YANG B., FENG J., GUENNEBAUD G., LIU X.: Packet-based hierarchal soft shadow mapping. *Computer Graphics Forum* (2009).