Fast Percentage Closer Soft Shadows using Temporal Coherence

Michael Schwärzler*
VRVis Research Center, Austria

Christian Luksch†
VRVis Research Center, Austria

Daniel Scherzer‡
Max-Planck-Institut für Informatik, Germany

Michael Wimmer§
Vienna University of Technology, Austria

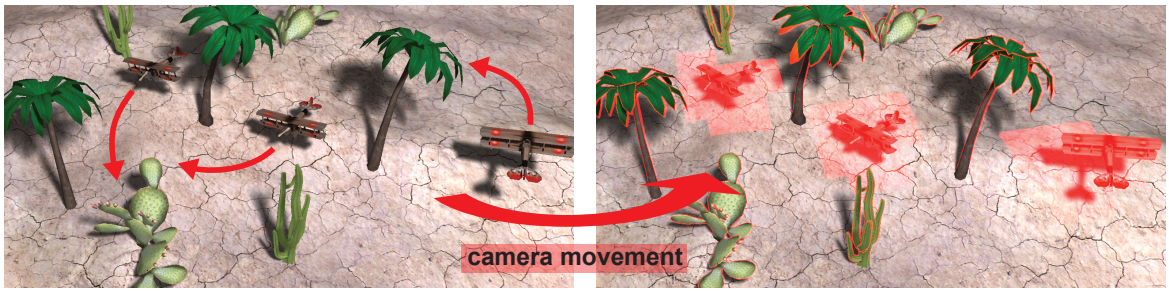

Figure 1: *Our new method improves the rendering performance of the* Percentage Closer Soft Shadows *method by exploiting the temporal coherence between individual frames: The costly soft shadow recalculation is saved whenever possible by storing the old shadow values in a screen-space* History Buffer*. By extending the shadow map algorithm by a so-called* Movement Map*, we can not only identify regions disoccluded by camera movement, but also robustly detect and update shadows cast by moving objects: Only the shadows in the areas marked red in the right image have to be re-evaluated. This saves rendering time and doubles the soft shadow rendering performance in real-time 3D scenes with both static and dynamic objects.*

Abstract

We propose a novel way to efficiently calculate soft shadows in real-time applications by overcoming the high computational effort involved with the complex corresponding visibility estimation each frame: We exploit the temporal coherence prevalent in typical scene movement, making the estimation of a new shadow value only necessary whenever regions are newly disoccluded due to camera adjustment, or the shadow situation changes due to object movement. By extending the typical shadow mapping algorithm by an additional light-weight buffer for the tracking of dynamic scene objects, we can robustly and efficiently detect all screen space fragments that need to be updated, including not only the moving objects themselves, but also the soft shadows they cast. By applying this strategy to the popular *Percentage Closer Soft Shadow algorithm (PCSS)*, we double rendering performance in scenes with both static and dynamic objects – as prevalent in various 3D game levels – while maintaining the visual quality of the original approach.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: soft shadows, temporal coherence, real-time

*e-mail:schwaerzler@vrvis.at
†e-mail:luksch@vrvis.at
‡email:scherzer@mpi-inf.mpg.de
§e-mail:wimmer@cg.tuwien.ac.at

1 Introduction

Shadows are employed in nearly every real-time 3D game that hits the market these days: They not only improve the quality of the rendered scene, but also contribute to a better scene perception. Still, most game engines apply only hard shadows (with filtered edges) and abstain from physically plausible soft shadows with varying penumbra sizes due to the high performance impact.

The main reason for this is the complexity of the visibility problem that is induced when area light sources are used instead of simple point lights. In order to calculate a physically exact soft shadow solution, either an infinite number of point light source samples distributed over the area light source would have to be taken and accumulated, or the visibility of the light source for each visible fragment in screen space has to be evaluated [Sintorn et al. 2008]. Single-sample soft shadow approaches try to approximate this theoretical solution by calculating a hard shadow first, and apply a blur filter kernel that is based on an approximation of the occluders between the light source and the shadowed surface. While methods like *Percentage Closer Soft Shadows (PCSS)* [Fernando 2005] or *Back-projection* techniques [Guennebaud et al. 2006; Guennebaud et al. 2007; Atty et al. 2006; Aszdi and Szirmay-Kalos 2006; Schwarz and Stamminger 2007; Baoguang et al. 2009] achieve a quality level that is hard to distinguish from a physically correct shadow, the computationally expensive occluder analysis in the pixel shader makes them prohibitively expensive to use in today's games.

Recently, various publications tackled the problem of reducing the computational effort for expensive operations in the pixel shader by introducing a so-called *history buffer* to exploit temporal coherence [Nehab et al. 2007; Scherzer et al. 2007; Sitthi-amorn et al. 2008a; Sitthi-amorn et al. 2008b]. The main idea is to re-use the calculated pixel values over several consecutive frames by storing them in a screen-space buffer and reprojecting them into the next frame. If the reprojected value is still valid (i.e. the fragment depths between the two frames is below a given threshold), it can be used to either omit a costly re-calculation or to improve the image quality.

In this work, we propose to combine the idea of reusing data from previously rendered frames with the expensive calculation of perceptually convincing soft shadows with varying penumbra size (see Figure 1): Soft shadow intensities calculated with the PCSS algorithm are stored in a history buffer and potentially reused in consecutive frames. In case of shadows generated by area light sources, this task can not be fulfilled with a simple per-fragment depth comparison, as moving objects cast complex shadows on completely different regions in the scene. We therefore propose a simple and easy-to-implement enhancement to the shadow map generation step, allowing shadows that have become invalid to be detected with a single texture lookup. We further discuss the issues introduced during the buffer reprojection, the limitations concerning moving light sources, and give details on our implementation and the achieved results.

2 Related Work

We give a short overview on publications related to our work. We refer the interested reader to the book recently published by Eisemann et al. [Eisemann et al. 2011] and to [Hasenfratz et al. 2003] for an in-depth overview on real-time (soft) shadows, and to a state-of-the-art report on temporal coherence techniques in real-time applications by Scherzer et al. [Scherzer et al. 2011].

2.1 Real-time Soft Shadow Mapping

The *Shadow Mapping* algorithm is an image-based algorithm proposed by Reeves et al. [Reeves et al. 1987] in 1978, and has been extended and adopted in numerous further publications. The basic idea is to first render the scene from the position of the light source and store the depth values in a texture, called shadow map. This discrete scene representation, containing the distances to all sampled surface points illuminated by the light, is then used in a second pass rendered from the point of view of the camera to calculate the shadow value: Each visible fragment is transferred to light space, and the distance to the light source is compared to the value stored in the shadow map.

Although being fast and easy to use, this algorithm suffers from aliasing and undersampling artifacts. Shadow filtering methods like *Percentage Closer Filtering (PCF)*, where the distance is also compared to the neighboring values in the shadow map to generate a "shadow percentage", or pre-filtering techniques like *Variance Shadow Maps* [Donnelly and Lauritzen 2006], *Convolution Shadow Maps* [Annen et al. 2007] or *Exponential Shadow Maps* [Annen et al. 2008b] reduce this artifacts by blurring the shadow edges. Apart from reducing the artifacts, the introduced "softness" of the shadow is also perceptually more convincing than hard shadow borders, since nearly all light sources in reality have a certain extent. Still, these blurry shadows do not reflect the fact that the size of the penumbra (the "half-shadowed" area, from which the light source is partly visible) varies depending on the distance relations between the light source, occluders and receiver.

In order to simulate more accurate soft shadows with varying penumbra sizes, *Backprojection* techniques ([Guennebaud et al. 2006; Guennebaud et al. 2007; Atty et al. 2006; Aszdi and Szirmay-Kalos 2006; Schwarz and Stamminger 2007; Baoguang et al. 2009]) use a single shadow map not only for depth comparison, but employ it as a discretized representation of the scene. The visibility factor for a screen-space pixel is calculated by back-projecting the shadow map texels onto the light source, where the amount of occlusion is estimated. These approaches produce perceptually convincing results in many cases, but are prone to artifacts and require a costly blocker search in the shadow map.

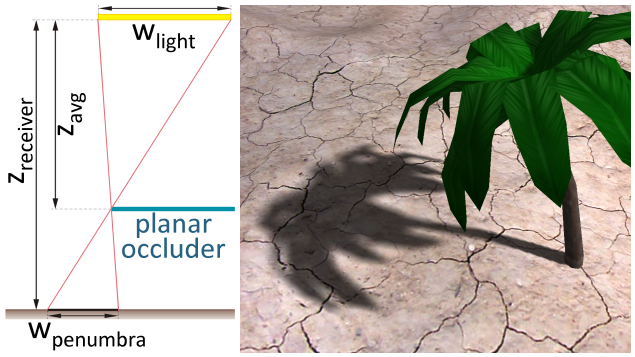

Figure 2: *The PCSS algorithm: By estimating a penumbra width based on light size, average occluder distance and receiver distance (see Eq. 1), the filter kernel size is adapted, generating visually plausible varying penumbra sizes.*

Percentage Closer Soft Shadows (PCSS) [Fernando 2005] extend the *PCF* filtering method to support variable kernel sizes in order to simulate varying penumbra sizes (see Figure 2, right): An average blocker distance z_{avg} is first calculated by searching for values in the shadow map that are smaller than current pixel's depth within an initial kernel. Under the assumption that blockers and receiver are planar and in parallel, the penumbra size $w_{penumbra}$ is estimated based on similar triangles (see Figure 2, left) using the relations between pixel depth $z_{receiver}$ and light source size w_{light}, and the filter kernel is adjusted accordingly:

$$w_{penumbra} = w_{light} \frac{(z_{receiver} - z_{avg})}{z_{avg}}. \quad (1)$$

While the approach is comparably easy to implement and produces visually pleasing soft shadows, the vast amount of texture lookups (blocker search + large filter kernel) have a negative impact on rendering performance. We therefore propose a way to overcome this expensive calculation by exploiting temporal coherence techniques (see Section 2.2). Based on this idea of an additional blocker search for the estimation of the penumbra filter kernel size, adoptions of the prefiltering techniques mentioned above have been proposed to follow the PCSS pipeline [Yang et al. 2010; Annen et al. 2008a]. While the achievable performance of prefiltering techniques is superior to a simple PCSS version, the implementation and handling is rather complex.

2.2 Data Caching / Temporal Coherence

Reusing data from previous frames by reprojecting it into the current frame has been independently proposed by Nehab et al. [Nehab et al. 2007] and Scherzer et al. [Scherzer et al. 2007], and is referred to as *Reverse Reprojection*. The idea is to store per pixel information in an off-screen buffer - the so-called *history buffer*, *payload buffer* or simply *cache*. This buffer is viewport-sized and filled for all visible rasterized surface points. In consecutive frames, the cached data is reprojected according to the scene motion, and reused in the current frame whenever possible: By comparing the stored depth against the current depth (within a given tolerance), it is decided whether the pixel was visible in the previous frame, and the data stored in its buffer location can usually be safely reused (except for changes in the shading signals, e.g. a moving light source, specular highlights, etc.). In disoccluded regions or areas that lay outside the view frustum in the previous frame, the information has to be recalculated. It has to be pointed out that due to the lookup in the history buffer and the corresponding resampling, a reprojection

error is introduced whenever the viewpoint changes, see Section 3.3.

Depending on the application, the reprojected data can be used to improve both image quality [Scherzer et al. 2007; Scherzer and Wimmer 2008; Yang et al. 2009] and performance (by either reducing the need for expensive recalculations [Nehab et al. 2007; Sitthi-amorn et al. 2008a; Sitthi-amorn et al. 2008b], or by distributing expensive calculations into multiple frames [Scherzer et al. 2009]). Reiner et al. [2012] use a similar cache structure to increase the rendering performance in an interactive procedural modeling system.

2.3 Temporal Coherence and Shadows

Temporal Coherence in shadow calculation has been scientifically discussed twice: Scherzer et al. [Scherzer et al. 2007] exploit temporal coherence to converge to pixel-correct hard shadows. And in Scherzer et al. [Scherzer et al. 2009], the costly calculation of physically correct soft shadows is spread over multiple frames, so that the accumulated shadow values in the history buffer converge to the exact solution. In contrast to these publications, we concentrate on increasing rendering performance for plausible soft shadows, and suggest a stable solution for dynamic scene objects - an unsolved issue for both proposed techniques.

3 The Algorithm

We describe how to combine the idea of exploiting *Temporal Coherence* with the generation of soft shadows using the *PCSS* method (Section 3.1). Especially the robust updates of soft shadows cast by moving scene objects (Section 3.2) and the handling of the reconstruction error introduced during the reprojection (Section 3.3) are non-trivial issues and require special considerations.

3.1 Shadow Reprojection

We closely follow the *Reverse Reprojection* pipeline in order to reuse soft shadow information (see Figure 3): Each fragment's shadow value, computed via PCSS (implemented exactly as in the NVIDIA PCSS white paper [Kevin et al. 2008]), is not only used to illuminate the corresponding scene surface, but also stored together with the clip space scene depth in a 2-channel off-screen viewport-sized history buffer (which is set as a second render target). If the depth test as described in Section 2.2 passes, the old shadow value is reused. Since reading the old buffer information and writing the new data in the same rendering pass is not allowed on current GPUs, we use two buffer textures and switch them in ping-pong style. Assuming a static scene configuration, the costly PCSS evaluation has to be only performed in regions of disocclusions or in areas which have previously been outside the screen borders after camera movements, significantly reducing the rendering load: Instead of a minimum of 16 texture lookups for the blocker search and the filtering step each, only a single bilinear lookup in the history buffer has to be executed.

3.2 Detecting Moving Objects

Whenever an object in the scene moves, the shadows cast *on* this object as well as the shadow cast *by* it change, so that the corresponding fragments in screen space cannot be reconstructed from the history buffer and need to be recalculated. The invalidation of shadow information that is cast *on* a dynamic object is trivial, as these fragments automatically fail the depth test described in Section 3.1. For the shadows cast by dynamic objects onto static objects anywhere in the scene, this depth test is completely irrelevant (as of course no change in depth is induced by shadows), making

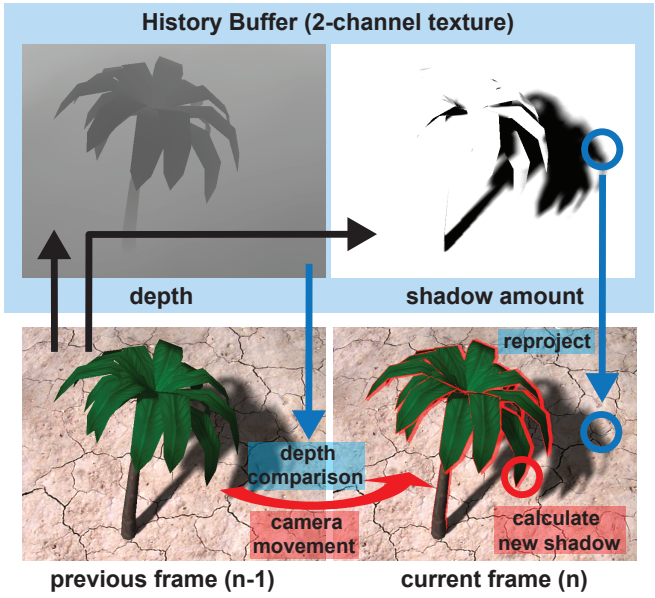

Figure 3: *Shadow reprojection: Depth and soft shadow amount from frame $n-1$ are stored in the* History Buffer. *In frame n, the buffer is reprojected, and the depth values are compared. If an occlusion (marked red) is detected, a new soft shadow value has to be estimated using the* PCSS *algorithm; otherwise, the stored shadow can be re-used.*

the invalidation a much more complex task, especially in the case of soft shadows (see Figure 4).

Figure 4: *Shadows from moving object.* Left: *Correctly shadowed scene of a moving toy airplane using PCSS.* Right: *Naive reprojection using depth comparison only recognizes that the shadow* on *the airplane needs to be updated, but does not indicate the need for a shadow recalculation in the regions marked with red ellipses.*

For this reason, we extend the shadow mapping algorithm by storing not only the depth from the view of the light source, but also the information whether an object is currently moving, in a lightweight binary mask buffer with the same size as the shadow map. This buffer is set as a second render target, and can therefore be written to in parallel to the depth map. We use an 8 bit buffer and output a value of "1" (which is stored as "255") in the shader for a texel where a moving object is visible, and "0" otherwise, and refer to this buffer by calling it *Movement Map* (see Figure 5). By looking up this information during the shadowing pass with a preliminary texture fetch, it can immediately be decided whether a shadow recalculation is necessary for the fragment, or if the history buffer should be investigated. Additionally, shadow values in whose calculation moving objects have been involved are marked as such in the history buffer, so that they get updated when the dynamic object casting the shadow has "moved on", avoiding the shadow leaving a "trail" (see the accompanying video for a demonstration of the ro-

Figure 5: *Shadow mapping extension for dynamic objects.* Left: *A scene with a static palm, casting a shadow onto the moving toy airplane.* Middle: *The depth map as known from the shadow mapping algorithm.* Right*: The corresponding* Movement Map*, indicating regions in which a moving object casts a shadow. Note that by rendering dynamic objects first, the parts of the airplane that are* occluded by the palm *can be stored as well. After the moving objects are stored, mipmaps are generated and used for an efficient lookup during the shadowing pass.*

bustness of this method even when objects in the scene are moving very quickly).

A problem with this approach lies in the concept of using only a single hard shadow map for the generation of physically plausible soft shadows through filtering. The information regarding moving objects is only valid for the original hard shadow, and needs to be extended to reflect the penumbra region, i.e. the region that is partly visible from the area light source. We divide the penumbra itself into two regions that require special attention: the *inner penumbra*, representing the part that lies inside the hard shadow borders and is connected to the fully shadowed *umbra* region, and the *outer penumbra*, extending the hard shadow and fading out until the surface is fully lit. The problems associated with these regions are (see also Figure 6 and Figure 7):

(I) *Inner penumbra of static objects*: The inner penumbra of a static object lying closer to the light source than a moving object on the same "light ray" will block an update of the area the moving object cast a shadow on, as the value in the movement map is "0" (see blue ellipse in Figures 6 and 8).

(II) *Outer penumbra of moving objects*: Whenever an objects moves, the approaching soft shadow is *larger* than the "tagged" area in the movement map described above (see red ellipse in Figures 7 and 8). It would therefore be necessary to search for information in the map within a given radius through costly texture lookups, annilihating the algorithmic speedup gained by using the history buffer as described in Section 3.1.

We solve these two difficulties by refining our strategy on how the movement map is filled and used: By first rendering all moving objects into both the depth map and the movement map, then releasing the movement map as a render target, and finally rendering the remaining objects, the fragments of the misleading static elements (problem *I*) are not depicted in the movement map anymore, ensuring that all inner penumbra regions are updated correctly. At the same time, the depth map itself represents the scene with the correct ordering of objects and depth values as usual.

In order to solve problem *II* with the outer penumbra of moving objects, we exploit the fact that (in contrast to the depth map) the movement map can be prefiltered, and use the hardware-accelerated mipmap generation procedure to efficiently create an image pyramid of the movement map. Since we use an 8 bit buffer, a value of "255" leaves a footprint over at least 5 mip levels using the pixel-averaging mipmap generation algorithm, which proved to be sufficient in all our tests (in case of a 1024^2 depth map, level 5 represents a search radius of 32 texels). This allows us to search for a moving object within a given radius by simply checking if the value of a texel in the desired mip layer is *larger than zero* with only a single texture lookup. The selection of the correct mip level is equivalent to the calculation of the initial occluder search radius in the PCSS algorithm (see Equation 2), and is estimated by taking the area light source size w_{light} (in UV space) and the distance of the fragment $z_{receiver}$ to it into account [Kevin et al. 2008]:

$$r_{search} = \frac{w_{light} * (z_{receiver} - d_{Nearplane})}{z_{receiver}} \tag{2}$$

Note that the size of the light source in UV space w_{light} is a customizable parameter and has to be set by the programmer according to scene scale and is "something you can change with artistic preference" [Kevin et al. 2008]. The corresponding mip level of a shadow map with size w_{SM} can thus be found by evaluating

$$l_{mip} = \lceil \log_2 (2 * r_{search} * w_{SM}) \rceil, \tag{3}$$

where w_{SM} is the shadow-map resolution. With the movement map prepared this way, we can efficiently and robustly detect *soft* shadows of dynamic objects for any fragment in screen space, allowing us to quickly decide whether a new shadow value has to be calculated due to object movement, or if the history buffer should be checked for reusable data.

3.3 Reconstruction Error

Reprojecting the history buffer from the previous frame to the current one always comes at the cost of a certain reconstruction error in case of camera movement. This problem is equivalent to transforming a 2D image, and requires resampling of discrete data. Since the best available native reconstruction filter on today's graphics hardware is bilinear interpolation, state-of-the-art publications in the area of temporal coherence [Scherzer et al. 2011] advise to sample the history buffer accordingly, as most of them reuse the stored data only for one or a few frames.

While this strategy may be sufficient in many use cases, we have made the observation that keeping and reprojecting shadow values for several hundreds or thousands of frames using bilinear interpolation introduces an amount of additional softness in the shadow that is noticeable in certain scene configurations. Interestingly, this yields both positive and negative aspects for the soft shadows (see Figure 9):

- The additional blur leads to a *reduction of banding artifacts* that can occur when the penumbra size is large and not enough samples are used during the PCF step. By applying our proposed reprojection strategy, these artifacts disappear automatically after a few frames.
- Unfortunately, the blurriness falsifies the penumbra size drastically in *contact areas*, where shadow casters and shadow receivers touch each other. In such regions, the shadow is nearly a hard shadow (i.e. there is hardly any penumbra visible), and the introduced softening is therefore visually disturbing.

Depending on the scene configuration it can therefore be necessary to make sure the accumulated reprojection error does not become too large in order to avoid "oversmoothing". We have evaluated two strategies to overcome the problem (see Section 5 for details):

Using third-order (bicubic) texture sampling by manually implementing it as a shader function, the amount of blur introduced during the reprojection is significantly smaller, retaining small penumbras. Note that the applied bicubic filter has to reconstruct the his-

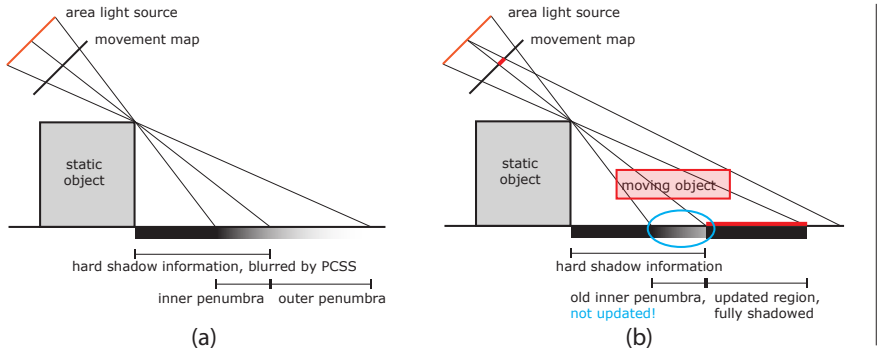

Figure 6: *Problems with inner penumbras of static objects: In a scene with static objects only, the movement map stays empty (a). If a moving object enters the view frustum of the light source (b), and static objects are rendered prior to moving objects, the data in the movement map is not properly set as seen in Figure 5 due to z-buffering! This prevents a proper update and causes artifacts in inner penumbra regions of static objects (blue ellipse). See Figure 8 for a visualization of the artifacts caused by such special scene configurations as well as the results in Figure 11 for an example on how to correctly handle these cases.*

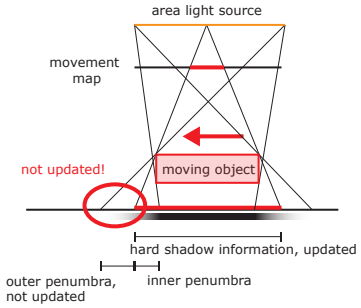

Figure 7: *Problems with outer penumbras of moving objects: If only a single lookup in the highest mip level (i.e. the "hard shadow" boundaries) is used to evaluate shadow updates, the outer penumbra regions of moving objects are not properly updated (red ellipse).*

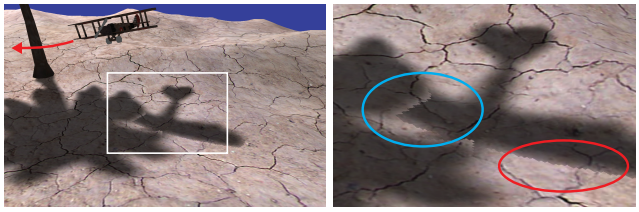

Figure 8: *Artifacts in the penumbra regions (as explained in Figure 6 and Figure 7) that are removed by our movement map generation strategy (see the results in Figure 11 for a proper handling of these cases).*

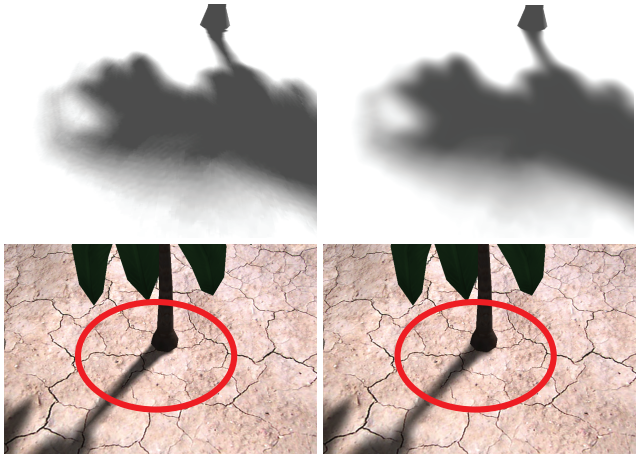

Figure 9: *The blur introduced by bilinear reconstruction has both positive and negative effects.* Top Row: *Typical PCSS band artifacts (left) disappear after the camera has been moved and the shadow has been reprojected several frames (right).* Bottom Row: *Shadows in "contact regions" with a small penumbra (left) become too soft due to reprojection (right).*

tory buffer by *interpolation*, and not by *approximation*, as otherwise the shadow information "loses energy" and becomes unusable within a few frames. Unfortunately, this makes the use of an optimized B-Splines filter with only 4 bilinear texture fetches as proposed by Sigg and Hadwiger [Sigg and Hadwiger 2005] impossible. Instead, we implemented a Catmull-Rom interpolation with 16 nearest neighbor texture fetches for the history buffer lookup, which is used in frames whenever the camera has moved. While this conceptually solves the problems with the reprojection error, the application is only feasible if a high-quality PCSS version with more than 16 texture lookups is used (e.g. 64 for the PCF step) – or if future generations of GPUs support a corresponding bicubic texture sampling in hardware.

Alternatively, a refresh strategy as described in [Scherzer et al. 2011] can be used to update the bilinearly reprojected texels before the accumulated error becomes noticeable. By dividing the screen into groups in a grid, tiled regions can be updated periodically using a global clock. An *amortized sampling* strategy ensures that newly calculated shadow values do not completely replace the old ones, but that they are gradually blended, avoiding visible transitions caused by the update pattern.

Although refreshing the fragment pixels reduces the achievable performance speed-up factor by approximately 10%, we opted for this solution in most of our application scenarios, as it is highly configurable, simple to implement, predictable and stable. Bicubic texture sampling has a larger performance impact and reduces the achievable speedup by about 20% on today's graphics hardware. Still, the decision on whether the reprojection error needs to be minimized and what strategy is applicable depends on the scene configuration, the area light source size, the amount of movement and the desired shadow quality.

4 Implementation

Our proposed technique to increase the rendering performance of the PCSS algorithm can be implemented on all current shader-based rendering frameworks. We have implemented the algorithm in a C++ framework using DirectX10 for testing and evaluation purposes. Based on the PCSS example provided in the NVidia white paper [Kevin et al. 2008], we render the depth map into a 32 bit render target with a size of 1024^2, and simultaneously use an 8 bit texture render target of the same size as the movement map. Note

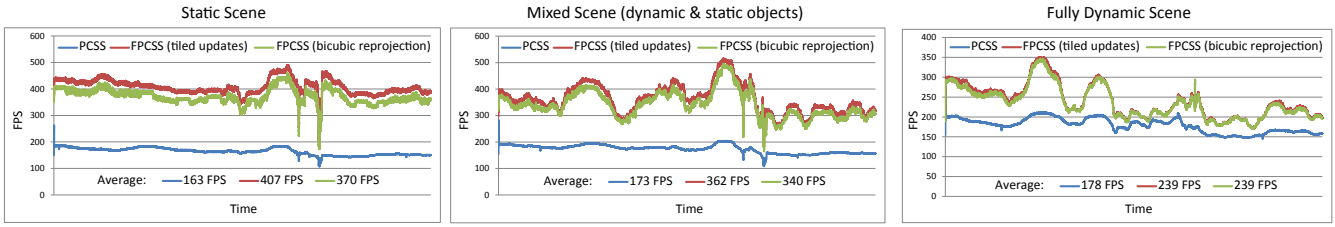

Figure 10: *Benchmark comparison of different scenes (see the accompanying video for a visual analysis of the recorded benchmarks). The standard PCSS algorithm, our new method with a tiled update strategy, and our new method with bicubic reprojection have been tested (FullHD resolution, 64 samples for both the blocker search and in the filtering step). In all three scenes, the same camera path has been used for the walkthrough. In the first frame of the recording, the history buffers have been filled with data from preceding frames.* Left: *In a static scene, where only the camera was moving, an average speedup factor of 2.5 could be achieved.* Middle: *In a scene with both static and moving objects (representing a typical game scene), the average performance was increased by factor 2.* Right: *Even in a scene with only dynamic shadow casters, the scene could be rendered at 130% of the speed of the PCSS version.*

that *conceptually*, it would be sufficient to simply use a DirectX Depth-Stencil Resource to save both the depth and the movement information, but due to API limitations, we have to use two separate render targets: The first issue is that a depth resource can only store depth values between 0 and 1, but the original PCSS algorithm uses linear depth values. Secondly, no mipmaps can be generated for a DirectX stencil buffer resource.

For the history buffer, we use two screen-size texture resources with two 32 bit channels each, where one texture is set as a render target and stores the current information, and the other acts as the lookup buffer for the information from the previous frame. After each rendered frame, the two textures are swapped ("ping-pong"). While in the first channel of the history buffer the current shadow value is saved, the second channel stores the depth and (encoded by a negative sign) the information whether the shadow originates from a moving object. The usage of such screen size buffers is related to *deferred shading* approaches that have become a popular method in today's 3D games, and does therefore well integrate in such rendering systems.

5 Evaluation and Comparison

We have tested our algorithm in three different scenarios in terms of necessary shadow updates and visual quality (see Figures 10 and 11): First, we evaluated our method in a completely static scene, then replaced some objects by moving toy airplanes, and finally tried to challenge our algorithm with a scene where all shadow casters are moving. In these three scenes, we used exactly the same camera path for the scene walkthrough. The system used for the tests consists of an Intel Core i7 920 CPU with 6GB of RAM and a Geforce GTX 580 GPU. See also the accompanying video for a visualization of the results (we also demonstrate the robustness of our approach with a very fast-moving shadow caster there).

We compared three different algorithms per scene in our benchmarks: The standard PCSS algorithm, our method with a tiled region update strategy, and our method with bicubic reprojection (see Section 3.3). In the PCSS step, we used 64 samples for both the blocker search and the filtering step, allowing us to simulate a large area light source with visually plausible penumbras. The screen view port was set to a resolution of 1920x1080 (FullHD) for the benchmarks.

As can be seen in Figure 10, our proposed algorithm outperforms the standard PCSS algorithm in all three scenarios. It is easily comprehensible that the greatest performance improvement (250% of the PCSS frame rate) can be achieved whenever most of the shadow values in the scene can be reused (i.e. the scene is static). Still, a significant performance boost (130%) can even be gained in a fully dynamic scene where all shadows cast by the moving objects have to be recalculated! This effect can be explained by the fact that shadowless regions (in contrast to the standard PCSS algorithm) do not need to perform the costly blocker search at all, but can rely on the information in the movement map that is fetched with a single texture lookup. In a scene with both static and dynamic objects, comparable to situations often found in 3D games, the average frame rate is doubled.

In general, our proposed method benefits from the computational complexity inherent in the chosen soft shadow algorithm: the more expensive shader instructions can be saved through the reprojection, the higher is the speed-up. The relative performance boost would therefore be even higher in a PCSS version with 128 texture lookups for the blocker search and 128 lookups for the PCF filtering step, but lower when using a version with only 32 lookups each. Note that the chosen PCSS light source size itself cannot be seen as a direct influence factor for the prospective performance, as the evaluation of pixels to recalculate takes place in screen-space only: The camera position and the scene configuration have a significantly larger impact (e.g. if the camera is very close to a penumbra region of a dynamic object, nearly the whole screen has to be updated in the next frame – even if the light source itself is comparatively small).

As demonstrated in the close-up images of Figure 11 and the accompanying video, the perceivable differences between the different algorithms are negligible and hardly noticeable. Even if shadows in contact areas become softer than the PCSS version due to bilinear reprojection, the tiled region update strategy combined with amortized sampling quickly covers up the introduced blur within a few frames.

6 Discussion and Conclusion

We have presented a new method to *accelerate the computationally expensive PCSS algorithm* by exploiting *temporal coherence* techniques and by extending the shadow-mapping algorithm by a so-called *movement map* – a light-weight 8 bit buffer storing the location of moving objects in light space. By pre-filtering this map through mipmap generation, it can be easily decided with a single texture lookup whether the soft shadow value stored in a screen-space history buffer can be reused or has to be re-estimated.

The algorithm is easy to integrate into an existing rendering framework, and can be robustly used for all kinds of different scenes.

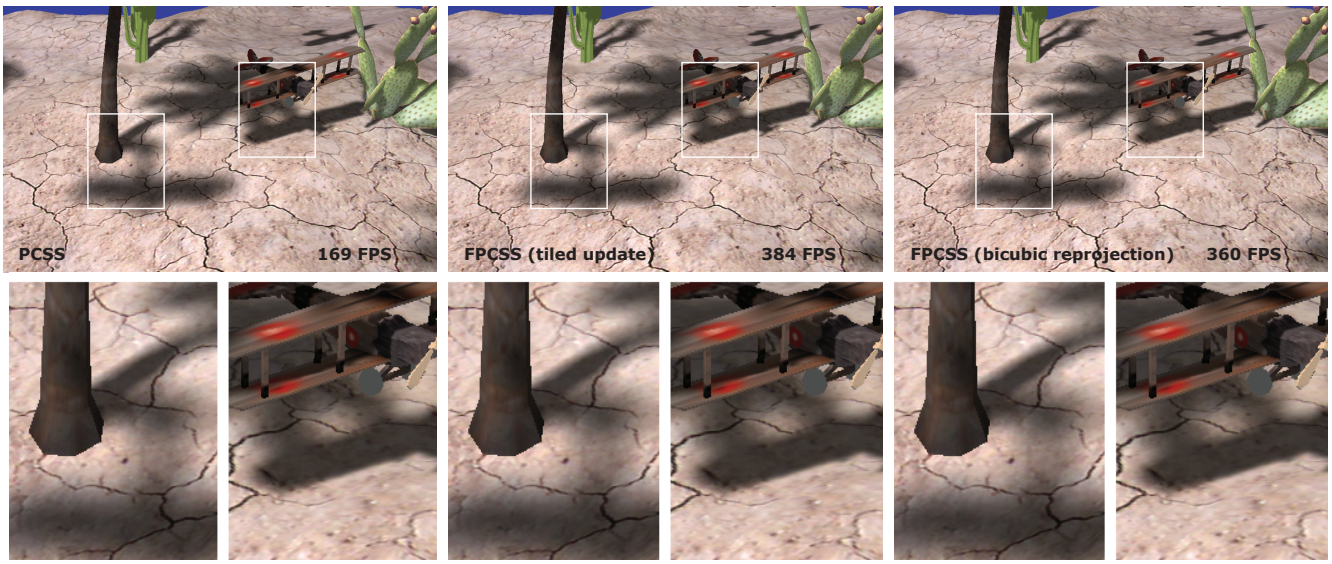

Figure 11: Top Row: *Result screenshots from the benchmark scene with static and dynamic objects.* Left: *PCSS, 169 FPS.* Middle: *Fast PCSS with tiled updates and amortized sampling strategy, 384 FPS.* Right: *Fast PCSS with bicubic reprojection of the history buffer, 360 FPS.* Bottom Row: *Close-up views of critical areas (contact shadows, overlapping penumbras) show that the achievable visual quality of our new approach is nearly equal to the quality of the significantly slower PCSS version.*

The achievable performance gain (between 250% in static scenes and 130% in fully dynamic scenes) comes at the cost of *memory consumption*, though: Apart from the 8 bit buffer for the movement map, two more 32 bit 2-channel screen-size buffers have to be allocated for the history buffer.

A limitation of our proposed algorithm is its application in scenes with *moving light sources*: While our algorithm can robustly handle such cases by setting all values in the movement map (i.e. in the "view frustum") of the light source to "255", obviously no speedup can be achieved. In this worst case, our algorithm is minimally slower (2-5%) than the standard PCSS algorithm, as it has to perform the mipmap generation and the additional movement map lookup. In scenarios with both static and moving light sources (e.g. with a static sun light source from above, and a headlight on a car driving around in the scene), using our method can still improve rendering performance: as long as the moving light source does not force an update of all fragments in screen space, at least the soft shadow from the static light source can be efficiently reused.

It has furthermore to be pointed out that the due to the varying amount of pixels that need to be updated, the frame rate of our approach is not as constant as when using the original PCSS method (see Figure10). This may be of concern whenever the overall rendering performance of the application is close to a critical threshold, e.g. 30 FPS.

We hope to be able to extend our idea to further soft shadow algorithms in order to make their use feasible in 3D games and applications – especially the real-time calculation of physically correct soft shadows in dynamic scenes is still a challenge yet to be solved.

Acknowledgements

We wish to express our thanks to Andreas Reichinger (VRVis Research Center) for his insightful and valuable comments on this work. The competence center VRVis is funded by BMVIT, BMWFJ, and City of Vienna (ZIT) within the scope of COMET - Competence Centers for Excellent Technologies. The program COMET is managed by FFG.

References

ANNEN, T., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2007. Convolution Shadow Maps. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, Eurographics Association, Grenoble, France, 51–60.

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2008. Real-time, All-Frequency Shadows in Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008) 27*, 3, 34:1–34:8.

ANNEN, T., MERTENS, T., SEIDEL, H.-P., FLERACKERS, E., AND KAUTZ, J. 2008. Exponential Shadow Maps. In *GI '08: Proceedings of Graphics Interface 2008*, Canadian Information Processing Society, Toronto, Ont., Canada, 155–161.

ASZDI, B., AND SZIRMAY-KALOS, L. 2006. Real-Time Soft Shadows with Shadow Accumulation. In *Eurographics 2006 Short Presentations*, Eurographics Association, 53–56.

ATTY, L., HOLZSCHUCH, N., LAPIERRE, M., HASENFRATZ, J.-M., HANSEN, C., AND SILLION, F. 2006. Soft Shadow Maps: Efficient Sampling of Light Source Visibility. *Computer Graphics Forum 25*, 4.

BAOGUANG, Y., FENG, J., GUENNEBAUD, G., AND LIU, X. 2009. Packet-Based Hierarchal Soft Shadow Mapping. *Computer Graphics Forum 28*, 4, 1121–1130.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance Shadow Maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ACM Press, New York, NY, USA, I3D '06, 161–165.

EISEMANN, E., SCHWARZ, M., ASSARSSON, U., AND WIMMER, M. 2011. *Real-Time Shadows*. A.K. Peters.

FERNANDO, R. 2005. Percentage-Closer Soft Shadows. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, ACM Press, New York, NY, USA, 35.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-Time Soft Shadow Mapping by Backprojection. In *Eurographics Symposium on Rendering (EGSR 2006), Nicosia, Cyprus*, Eurographics Association, 227–234.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2007. High-Quality Adaptive Soft Shadow Mapping. *Computer Graphics Forum 26*, 3, 525–534.

HASENFRATZ, J.-M., LAPIERRE, M., HOLZSCHUCH, N., AND SILLION, F. 2003. A Survey of Real-Time Soft Shadows Algorithms. In *Eurographics 2003 State of the Art Reports*, Eurographics Association.

KEVIN, M., FERNANDO, R., AND BAVOIL, L. 2008. Integrating Realistic Soft Shadows into Your Game Engine. Tech. rep., NVIDIA Corporation, 02.

NEHAB, D., SANDER, P. V., LAWRENCE, J., TATARCHUK, N., AND ISIDORO, J. R. 2007. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, Eurographics Association, 25–35.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, 283–291.

REINER, T., LEFEBVRE, S., DIENER, L., GARCÍA, I., JOBARD, B., AND DACHSBACHER, C. 2012. A runtime cache for interactive procedural modeling. *Computers & Graphics 36*, 5, 366 – 375.

SCHERZER, D., AND WIMMER, M. 2008. Frame Sequential Interpolation for Discrete Level-of-Detail Rendering. *Computer Graphics Forum (Proceedings EGSR 2008) 27*, 4, 1175–1181.

SCHERZER, D., JESCHKE, S., AND WIMMER, M. 2007. Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence. In *Rendering Techniques 2007 (Proceedings of Eurographics Symposium on Rendering)*, Eurographics Association, J. Kautz and S. Pattanaik, Eds., 45–50.

SCHERZER, D., SCHWÄRZLER, M., MATTAUSCH, O., AND WIMMER, M. 2009. Real-Time Soft Shadows Using Temporal Coherence. In *Advances in Visual Computing: 5th International Symposium on Visual Computing (ISVC 2009)*, Springer, Lecture Notes in Computer Science, 13–24.

SCHERZER, D., YANG, L., MATTAUSCH, O., NEHAB, D., SANDER, P. V., WIMMER, M., AND EISEMANN, E. 2011. A Survey on Temporal Coherence Methods in Real-Time Rendering. In *Eurographics 2011 State of the Art Reports*, Eurographics Association, 101–126.

SCHWARZ, M., AND STAMMINGER, M. 2007. Bitmask Soft Shadows. *Computer Graphics Forum 26*, 3, 515–524.

SIGG, C., AND HADWIGER, M. 2005. Fast Third Order Texture Filtering. *Published in the Book "GPU Gems 2"*.

SINTORN, E., EISEMANN, E., AND ASSARSSON, U. 2008. Sample-based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008) 27*, 4 (June), 1285–1292.

SITTHI-AMORN, P., LAWRENCE, J., YANG, L., SANDER, P. V., AND NEHAB, D. 2008. An Improved Shading Cache for Modern GPUs. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, Eurographics Association, Aire-la-Ville, Switzerland, 95–101.

SITTHI-AMORN, P., LAWRENCE, J., YANG, L., SANDER, P. V., NEHAB, D., AND XI, J. 2008. Automated Reprojection-Based Pixel Shader Optimization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008) 27*, 5, 127.

YANG, L., NEHAB, D., SANDER, P. V., SITTHI-AMORN, P., LAWRENCE, J., AND HOPPE, H. 2009. Amortized Supersampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009) 28*, 5, 135.

YANG, B., DONG, Z., FENG, J., SEIDEL, H.-P., AND KAUTZ, J. 2010. Variance Soft Shadow Mapping. *Computer Graphics Forum 29*, 7, 2127–2134.