GPU-Accelerated Point-Based Holograms

Sascha Fricke[†] **Reinhard Caspary**[⋆] **Susana Castillo**[†] **Marcus Magnor**[†]

[†] *Institut für Computergraphik, TU Braunschweig, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany*
[⋆] *Cluster of Excellence PhoenixD, Leibniz Universität Hannover, Welfengarten 1A, 30167 Hannover, Germany*

[†] {*fricke, castillo, magnor*}*@cg.cs.tu-bs.de* [⋆]*reinhard.caspary@phoenixd.uni-hannover.de*

Abstract: We present a novel fast method to compute point-based Computer Generated Holograms with occlusion on the GPU, using modern hardware capabilities. Our method offers a promising new route towards real-time calculation of high-resolution holograms.

1. Problem statement

Point-Based (PB) holography algorithms offer a flexible way of generating holograms from a point cloud with arbitrarily and sparsely distributed points in three dimensions. However, they can suffer from poor algorithmic scaling, i.e. $O(n \times m)$ with n object points and m hologram points, and while acceleration via thread scheduling on Graphics Processing Unit (GPU)'s is well suited to solve this limitation, doing it efficiently is a non-trivial problem [1]. In this work, we propose a new method vastly simplifying the combination of both approaches, enabling fast occlusion processing, and thus being able to generate PB holograms in interactive time.

In PB methods, an Object Point (OP) emits a spherical wave that intersects with the hologram plane, where the resulting wavefront is recorded. Superimposing these wavefronts from multiple points creates an interference pattern. One of the first algorithms to propose using the GPU for Computer Generated Holography (CGH), Petz and Magnor [2], used the hardware fixed-function pipeline in conjunction with hardware texturing to accelerate the interference computations. This method was restricted by the lack of floating point precision and programmable shaders provided by the hardware of the time. Later, Chen and Wilkinson [1] introduced the use of Compute Unified Device Architecture (CUDA) to accelerate this process further. The calculation time of PB methods scales with the propagation distance, since the radius of the point spread function rises with the propagation distance. Therefore, Shimobaba *et al.* [3] introduced the concept of the Wavefront Recording Plane (WRP), a virtual plane close to the object. The wavefront from the WRP to the desired hologram plane is then propagated using angular spectrum propagation. In these works, as in any CGH method, efficient occlusion detection and culling are crucial parts . Light field methods [4] are based on individual rays and therefore intrinsically take occlusion into account. Occlusion can be computed using either an approximation [1], raycasting [5], or a combination of raycasting with layer-based methods [6]. All of these techniques trade quality for a computational overhead. We improve both aspects by using modern hardware capabilities, leading to real-time calculation of high-resolution holograms, which is required by many applications like near-eye displays for augmented reality.

2. Method

Our algorithm follows the established PB method [1]. We store the phase distribution for a point of a certain distance similarly to [7]. The point spread function of each OP is limited to a circle with a radius derived from the grating equation. The pixel pitch $p = w_x/H_x = w_y/H_y$ of a hologram of size w_x, w_y and number of pixels H_x, H_y, leads to a maximum reconstruction angle at the wavelength λ: $\theta_{\max} = \arcsin\left(\frac{\lambda}{2p}\right)$. The cone defined by $\theta_{\max}$ projects a circle with radius $r = z_0 \tan\theta_{\max}$ onto the hologram plane limiting the influence of each OP (x_0, y_0, z_0).

Our algorithm splits the circular shape of the Point Spread Function (PSF) into a fan of triangles (Fig. 1(E)). For each segment, an occlusion ray is cast from its center to the respective OP using the programmable raytracing pipeline of the GPU. In case of occlusion, this segment is marked in a mask and thus ignored by later stages. Hence, the overall complexity of occlusion processing is independent of size and resolution of the hologram and depends only on the number of OPs and the fan subdivision. Since the occlusion computation prevents parts of the PSF circle from being rasterized, casting occlusion rays reduces the rasterization costs, leading to a net improvement on both quality and performance. Nominally, our method computes holograms of $300k$ OPs in 185 ms with occlusion vs. 179 ms without.

The point spread functions of neighboring OPs overlap to a high degree. This introduces a race condition when the contributions of different OPs to a certain hologram pixel need to be stored for summation. Synchronization using atomic operations can have a big performance impact and increases implementation complexity. Therefore,