Adaptive Gaussian Points for Faster and Better Computer-Generated 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: To achieve better quality and generation speed, we generate holograms from adaptively sized and placed points, modeled as 2D Gaussians. We compute them with a Rayleigh-Sommerfeld convolution directly in a compact, radially symmetric look-up table.

1. Problem Statement

For several decades, the research community has been looking for ways to improve the quality and efficiency of Computer Generated Holography (CGH). Among the proposed techniques, and since its proposal by Mark E. Lucente [1], one of the most revisited for improving computational speed is the use of Look-Up Tables (LUTs). While their advantages on the CPU are widely known [2–6], most recent previous work found that the use of a LUT on graphics hardware (GPU) can lead to a significant overhead due to the look-up cost [7].

In this work, we propose a new GPU implementation of a LUT for CGH with a three-folded goal: *Increased generation efficiency, increased quality of the generated hologram, and reduced number of required points.*

Chen *et al.* [3] indicated that the non-uniform distribution of points leads to better holograms with fewer points. Regrettably, their occlusion computation did not allow them to benefit from this. Following their insights, we distribute points non-uniformly to represent complex scenes more efficiently but, differently to them, we rely on an approximate hardware-accelerated occlusion computation. We approximate the Point Spread Function (PSF) as hardware-rasterized triangle-fans, and check each triangle segment for occlusion using hardware-raytracing. Thus, we are able to overcome the problems Chen *et al.* [3] encountered while also avoiding the expensive memory look-ups that Yanagihara *et al.* [7] discussed.

Our LUT stores pre-propagated Gaussians in a very cheap, radially symmetric representation [4, 6]. To improve storage efficiency in comparison to previous work [2, 5], we organize the LUT into a square image, where each row corresponds to a point at a given distance. The radius of the PSF is limited by the grating equation. This gives us a certain amount of extra space in our LUT rows which we use to store multiple versions of pre-propagated Gaussians of different sizes. Finally, we can select the appropriate point size during hologram computation using the point density information from our non-uniform distribution.

2. Method: The Look-Up Table

We scatter points on the surface of objects based on a grid of shaded images of a scene with different viewpoints. The shading is used as an input to control the point density, where brighter areas get more points under the assumption that they deserve more detail. The point scattering is done by transforming a 2,3-Halton series using the inverted Cumulative Distribution Function (CDF) derived from the shading images. The quasi-random Halton-sequence, being low-discrepancy, has a more uniform point density and thus offers better surface coverage than pseudo-random placement [8].

The LUT contains the radial symmetric point spread functions for individual distances z. For each row ξ_v, the distance is derived as:

$$z = z_{\min} + \frac{\xi_v}{N}(z_{\max} - z_{min}). \tag{1}$$

Given a square hologram ($H \times H$) of ($N \times N$) pixels, the radius r_z in pixels of the PSF is $r_z = r\,z$, where:

$$r = \frac{1}{p}\tan\left(\sin^{-1}\left(\frac{\lambda}{2p}\right)\right), \tag{2}$$

where p is the pixel pitch (H/N), and λ is the wavelength.

We define a square LUT and set its resolution equal to that of the hologram ($H \times H$). This ensures that the sampling density in the LUT is equal to the pixel pitch along the horizontal and vertical axes. The trade-off is that