# Using Path Tracing for Computer Generated Holography

INAN XU, UC San Diego, USA

This paper explores the use of path tracing for computer-generated holography (CGH), focusing on rendering high-fidelity holograms through modified Lajolla path tracing techniques. The approach simulates light transport to calculate the complex amplitude for hologram generation, which is then reconstructed using Fourier transforms. Although bugs in the pipeline affected the phase image and reconstruction quality, the results were compared with stochastic gradient descent (SGD)-based methods from the Odak library.

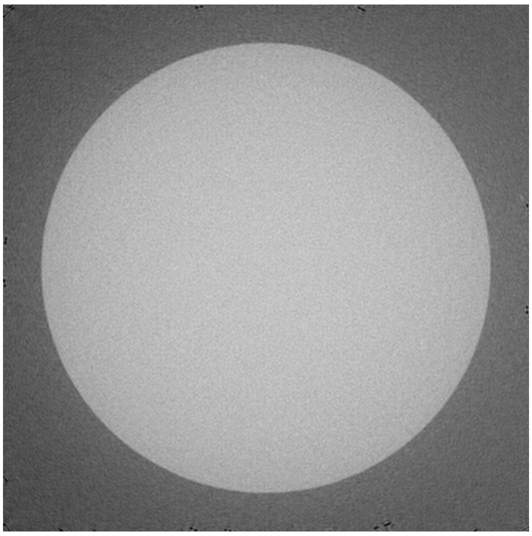

Fig. 1. A hologram reconstruction of the vol_path_test2.xml scene.

## 1 INTRODUCTION

Computer-generated holography (CGH) has a wide range of applications, from optical displays to microscopy and 3D imaging. Traditionally, CGHs are synthesized using the point-cloud method, where a wave is numerically propagated through space, interacting with discrete points to reconstruct the complex light field. While effective, this approach can be computationally expensive and difficult to scale for high-resolution holograms.

Path tracing offers an alternative framework for generating CGHs by simulating the transport of light rays instead of computing direct wave interactions at discrete points. By leveraging existing path tracing techniques, we can improve both the accuracy and efficiency of hologram generation. This approach aligns with modern advancements in physically based rendering, making it a promising direction for realistic and high-fidelity CGH synthesis.

In this report, I present the modifications made to the LaJolla renderer to support hologram generation using path tracing. I detail the implemented method, discuss key challenges in the pipeline, and analyze existing bugs that lead to deviations from expected results. Through this, I aim to provide insights into the feasibility of path-traced CGH and its potential improvements.

Author's address: Inan Xu, inxu@ucsd.edu, UC San Diego, San Diego, California, USA.

## 2 METHOD

The core approach is identical to the approach found in homework 2: "Multiple chromatic heterogeneous volumes with absorption and multiple-scattering with both phase function sampling and next event estimation, with surface lighting", except that I treat the light values as monochromatic. The maximum number of bounces, BSDFs, etc. and other path-tracing related parameters are kept the same.

### 2.1 Phase

The core rendering loop of Lajolla divides the image into tiles split among CPU threads. For each pixel $(u, v)$ in the output image, we sample $s$ rays based on the sampling density per pixel and calculate the radiance for each pixel by accumulating the contribution $L$ from each ray.

For simplicity, I only consider values from the red channel of the radiance. Each path tracing function was modified to also return the vertex of intersection. This is necessary to calculate $r_n$ for the complex amplitude $E(u, v)$. I accumulate each ray's contribution to the complex amplitude using the following equation, where $A_i$ is equal to the light returned from path tracing. In implementation, the real and imaginary components are derived using the relation $e^{in} = \cos n + i \sin n$, and there are checks to avoid dividing by 0 or $NaN$.

$$E(u,v) = \sum_{i=0}^{s} \frac{A_i}{r_i} \exp\left(i\frac{2\pi}{\lambda} r_i\right)$$

$$r_i = \sqrt{(x-u)^2 + (y-v)^2 + z^2}$$

From here, the phase value of the hologram is computed as follows. First, the quadratic phase factor is computed and multiplied with each value in $E$. Then, I perform a fast Fourier Transform to calculate the complex amplitude. I encode the phase component of the complex amplitude using the kinoform method to the output image, since the amplitude has mostly minimal effect on the output hologram. Each pixel is multiplied with a phase factor, passed into an arctan, and divided by $2\pi$ to normalize within $[0, 1]$.

$$\begin{aligned} H(\xi,\eta) &= \mathcal{F}\left\{E(u,v) \times \exp\left\{i\frac{\pi}{\lambda z} \times \left(\left(u\frac{\lambda z}{Mp}\right)^2 + \left(v\frac{\lambda z}{Np}\right)^2\right)\right\}\right\} \\ &\quad \times \frac{\exp\{i\frac{2\pi}{\lambda}z\}}{i\lambda z} \times \exp\left\{i\frac{\pi}{\lambda z}\left(\xi^2+\eta^2\right)\right\} \qquad (6) \\ Phase(\xi,\eta) &= \arctan\{H(\xi,\eta)\} + \pi \qquad (7) \end{aligned}$$

The majority of additional runtime cost comes from the fast Fourier transforms, but this occurs outside of the core rendering loop. Since it occurs outside of the rendering loop, it has a cost proportional to the output image size which is relatively minimal for my 512x512 images.

### 2.2 Reconstruction

With the phase image, we can reconstruct the hologram. Actual evaluation methods for computer generated holography often use optical reconstruction techniques using spatial light modulators (SLMs) to display the hologram and cameras to capture at various focal lengths. Since I lack that equipment, I evaluated using numerical reconstruction. To do this, I calculate the phase factor and take the Fourier transform to build the reconstructed image.

## 2.3 Implementation

The Fourier transforms utilized functions the FFTW library, specifically fftw_complex for storing the results and fftw_plan_dft_2d to perform the FFT. For reconstruction, I implemented a script in Python that processed the EXR file and handled the S-FFT calculations using OpenCV and numpy. Odak reconstructions were implemented in Python using instructions from the official Odak documentation. The code can be found on GitHub at https://github.com/ixukw/cgh_pathtracer.

## 2.4 Debugging

Unfortunately, I did not get my hologram pipeline working fully. One bug exists in the calculations for the phase image. The periodic nature of the Fourier transform is incorrectly shown in my output image. This is not caused by the tiling parallelization, as the images do not changed based on the tile size. Additionally, occasional runs of the renderer will return $NaN$ for all pixels in the image, likely caused by an overflow issue.

## 3 EVALUATION

To evaluate my rendering pipeline, I compare my output of the phase image and the reconstructed with outputs from the Odak library [2]. The Odak library uses stochastic gradient descent to approximate the phase and reconstruction images. I used their "Bandlimited Angular Spectrum" propagation method for the stochastic gradient descent (SGD) approximation of the phase and reconstruction images. Note that the EXR format was unsupported for the Odak library, despite the authors being optics researchers. This meant all evaluations were done in PNG.

There are three parameters related to evaluation: light wavelength, propagation distance, and pixel pitch. Because my implementation ignores the blue and green channels, I chose the red wavelength 6.23e-7 that was used in Chen et al[1]. for all evaluations. The pixel pitch is the resolution, which was held constant.

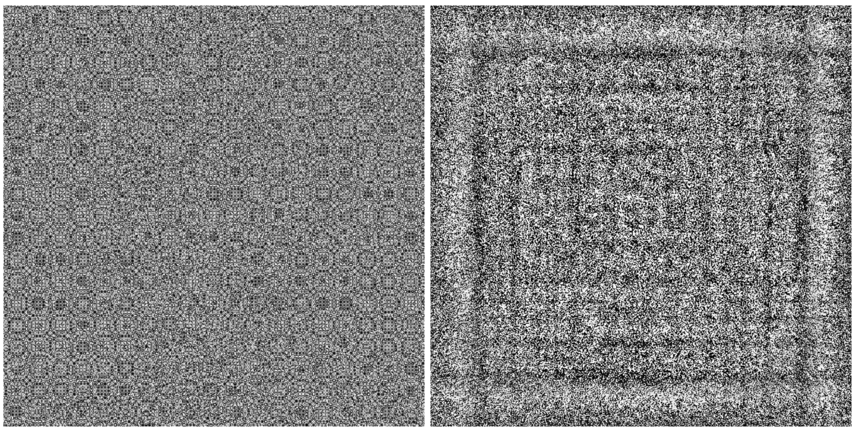

Fig. 2. Phase images for the volpath_test2.xml scene. The left image is mine, while the right image is the approximation from Odak. Notice the periodic circles.

Figure 3 shows the results of reconstruction using Odak. My phase image is clearly incorrect, but observe the subtly stronger circle warping in the center of the image. I also wrote my own

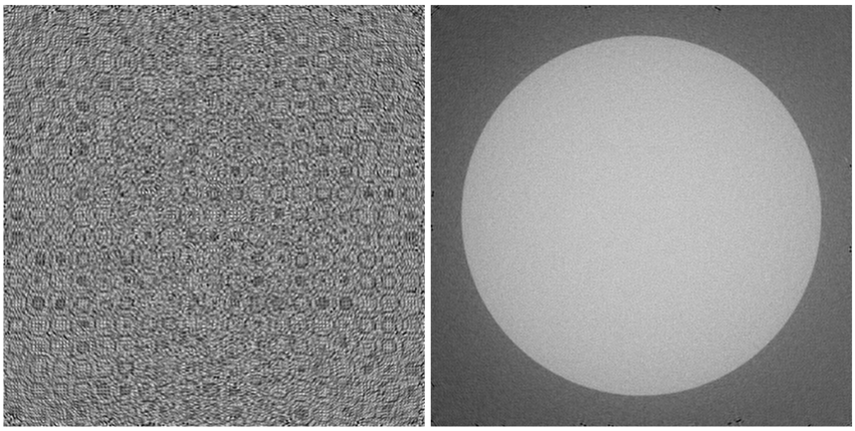

Fig. 3. Reconstruction using calculate_amplitude from the Odak library. Left: reconstruction using my phase image. Right: a grouth truth reconstruction using Odak's SGD approximation.

reconstruction to test, but there was clearly a bug in my implementation. I analyzed the difference between my phase image and the point cloud's. In theory, the phase images should be interchangable. Figure 4 displays the two phase images of my output and the correct version.

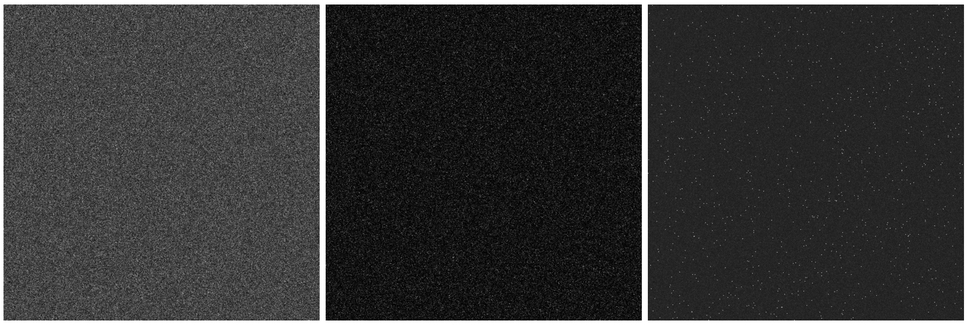

Fig. 4. Reconstruction images of the Cornell box using my bugged method. Left: Reconstructed hologram using path tracing approach. Center: Reconstruction using phase image generated from point cloud method. Right: Image difference between the two.

## 3.1 Choice of Rendering Algorithm in Lajolla

There was no significant difference between rendering algorithms in Lajolla. If the rendering algorithm did not support a scene, (vol_path_tracing_1 cannot work for the Cornell box scene), the output was effectively random. However, using multiple bounces versus a single bounce for the scenes did not have a major difference. I did not test on complex scenes like the hetereogeneous volume, though I would like to once the pipeline is working correctly.

## 4 TAKEAWAYS AND FUTURE WORK

I learned a lot about how computer generated holography works and its applications. Surprisingly, there are a lot of advancements in the field for point-cloud methods and machine learning approximations, but not many for path tracing methods. I think a lot of path tracing techniques could be leveraged to both increase realism and improve runtime using GPUs.

### 4.1 Parallelization

The primary advantage to path tracing is its easy parallelizability. All of the CGH-related calculations during the core rendering loop is easily parallelizable since they do not cause any data dependency issues. The S-FFT calculations to compute the CGH may need a more advanced parallelization strategy for sufficiently large images since it occurs outside of the rendering loop and only executes on a single thread. I was considering using Slang to implement this project and explore performance comparisons with existing methods, but chose not to due to unfamiliarity with the language.

### 4.2 Chromatic Holograms

There is significant research in chromatic holograms for practical applications like XR. This project can be easily modified to support chromatic holograms by calculating the phase for each channel and concatenating them during reconstruction.

### 4.3 Focusing

I am particularly interested in the "focusing" nature of holograms, where images can be produced with depth of field rather than full sharpness found in the images rendered for this course. Path tracing may naturally produce noiser images based on distance, but holograms achieves depth of field mathematically. I want to explore this area further for this project, possibly looking into if we can reuse computations.

## REFERENCES

[1] Xindi Chen, Aiming Ge, Dongsheng Zhu, Qiuyang Wang, Jiangbo Wu, and Shuo Liu. 2023. Photorealistic Rapid Computer-Generated Holography Employing an Enhanced Path Tracing Technique With Sequence Generated Trial. *IEEE Photonics Journal* 15, 5 (2023), 1–12. https://doi.org/10.1109/JPHOT.2023.3314525

[2] Koray Kavakli and Kaan Akşit. 2022. Introduction to Odak: a Differentiable Toolkit for Optical Sciences, Vision Sciences and Computer Graphics, In Frontiers in Optics + Laser Science 2022 (FIO, LS). *Frontiers in Optics + Laser Science 2022 (FIO, LS)*, FTu1A.1. https://doi.org/10.1364/FIO.2022.FTu1A.1