Engineering Reports

WILEY

Engineering Reports Open Access

RESEARCH ARTICLE OPEN ACCESS

Monte–Carlo Techniques Applied to CGH Generation Processes and Their Impact on the Image Quality Obtained

Juan A. Magallón[1] | Alfonso Blesa[2] | Francisco J. Serón[1]

[1]Department of Computer Sciences, Universidad de Zaragoza, Zaragoza, Spain | [2]Department of Electronics Engineering, Universidad de Zaragoza, Teruel, Spain

Correspondence: Alfonso Blesa (ablesa@unizar.es)

Received: 11 May 2024 | **Revised:** 17 December 2024 | **Accepted:** 18 December 2024

Funding: This initiative is carried out within the framework of the EU-funded "Plan de Recuperación, Transformación y Resiliencia" (Next Generation). Instituto Nacional de Ciberseguridad (INCIBE).

Keywords: Computer Generated Hologram | Monte–Carlo | ray tracing

ABSTRACT

Computer graphics aim to create visual representations for screens, where depth is simulated. In contrast, Computed Generated Holograms (CGH) focus on encoding and recreating light patterns to generate a true 3D holographic image that appears as a physical object in space. Therefore, although both use digital models, the computation of CGHs necessitates additional phase-related calculations, which in turn escalate computational demands. These calculations often result in excessively long development times or, at worst, render the process unfeasible. In order to reduce computational time, Partial Monte–Carlo Sampling (PMCS) techniques for CGH generation are presented, integrating them into the whole process of generating a CGH for a synthetic 3D scene, from design to rendering. PMCS is based on the random choice of a subset of rays used to compute the CGH and relates the computation time spent to the quality of the reconstructed scene. Quantitative analysis shows that PMCS does not significantly compromise image quality. Both simulated and in-laboratory image reconstruction from holograms demonstrates consistent trends, showcasing improved quality with higher numbers of rays and increased resolution. Furthermore, we establish a direct relationship between image quality and computational time, which effectively addresses specific requirements.

1 | Introduction

Traditionally, graphical scenes have been depicted using 3D geometry, texture, and appearance attributes, encompassing both local and global reflectance characteristics. Across these representations, the treatment of light is consistently grounded in ray optics principles, with rendering methods typically involving projection, rasterization, or ray tracing techniques [1].

The transition from 2D scene representation to 3D adds more complexity and entails a shift in how information is modeled, stored, and represented. One solution to this challenge comes from holography. Holograms are sophisticated structures that capture the phase and amplitude of a scene wavefront, offering views from all possible perspectives through a given aperture. Holographic visualization techniques allow the inclusion of all vision features, including those related to depth [2]: occlusion, eye accommodation, convergence, and stereopsis.

Computer-Generated Holography (CGH) is based on calculating holograms by simulating light interference patterns, allowing 3D image projection without physical objects. CGHs are the result of virtually propagating light from a three-dimensional scene to

a designated hologram plane, where the complex-valued amplitude of the CGH is obtained.

CGH can be implemented on static recording media as Holographic Optical Elements (HOEs) [3, 4] or Holograms to capture or design three-dimensional scenes as a way to storage information [5]. Dynamic display devices (e.g., Spatial Light Modulators, SLM) can be used to reproduce the CGHs pattern.

Holographic Optical Elements (HOEs) designed with Computer-Generated Holography (CGH) enable precise light control and highly customizable light manipulation, supporting applications in diverse fields. In laser systems, they shape and split beams for precise processes like micro-machining and laser lithography [3, 6, 7]. HOEs CGH-based are used in holographic displays and security applications, creating intricate anti-counterfeiting holograms on currency and IDs [8].

In contrast, SLM-based dynamic media use pixelated arrays, limiting resolution but enabling real-time updates, suitable for interactive applications like augmented reality and virtual reality (HOEs also play a role in AR/VR headsets, where they guide light to produce clear, immersive visual) [9, 10]. Electro-holography relies on several key technologies, including Spatial Light Modulators (SLM), and CGHs. These applications present challenges [11] such as the need for significant computational resources, data Storage and Bandwidth, and complex algorithms for real-time processing (in this regard, this work addresses the need to reduce computation time) or visual quality assessment [12].

CGHs pose significantly greater mathematical demands compared to geometric optics. In other words, a computational time cost must be paid when attempting to provide the spatial insight offered by CGH compared to the 2D synthetic images or the pseudo 3D when using stereoscopy. Although recent advancements in computer technology facilitate hologram synthesis on desktop computers, creating a full-parallax hologram using ray-tracing methods may still require several hours or days, even with state-of-the-art hardware.

To better understand the computational load requirements of Computer Generated Holograms (CGHs), let's consider a discretized 3D scene with HDTV resolution equivalent (1920×1080 pixels). A Spatial Light Modulator (SLM) serves as the display device for CGHs under proper illumination. Assuming a current pixel size of 1μ m, achieving a sufficiently large viewing window (e.g., 27-inch monitors in a 16:9 format, equivalent to 65.8 cm wide and 33.6 cm high) necessitates approximately $3,45 \times 10^9$ pixels. Given that each pixel in the scene must transmit information to every pixel in the SLM, the computational effort required amounts to approximately $3,15 \times 10^{15}$ times that of a single ray. Spatial or temporal multiplexing capable of generating color entails three times the computational effort.

3D image reconstruction from CGH is achieved by recovering the original wavefront by appropriately illuminating it and propagating the resulting wave through space. For this purpose, computational methods based on the Fourier Transform formalism are employed [13]. Consequently, this image reconstruction is not a bottleneck in obtaining the final results.

The digital nature of CGHs, the use of coherent light sources and the resolution limitations imposed by current technology lead to issues in the generated images related to diffractive effects and speckle. So, speckle noise reduction and image quality enhancement techniques for CGH must be used. Some of them are related to CGH multiplexing (i.e., increasing computing time), another ones are related with modifying phase or intensity patterns (intensifying the utilization of computational resources during hologram recalculation). Examples include the application of neural computation [14, 15], optimization techniques for phase holograms [16], or the incorporation of realistic imagery with global illumination effects [17]. However, employing such feedback techniques does not allow for a deterministic estimation of the computational time required for hologram synthesis.

One of the main challenges in CGH is to render the scene as realistically as possible while achieving the desired results within an acceptable time-frame. Realistic rendering encompasses factors such as textures, surface properties, inhomogeneous lighting, and occlusions, all of which are essential to increase the realism of a scene designed on a computer. To achieve these characteristics, various CGH design algorithms have been developed [18], which can be categorized into point-cloud methods, geometric primitives and functions [19], layer-based methods, and ray-tracing [20, 21], among the most commonly used.

Current barriers to achieving fast and efficient Computer Generated Hologram (CGH) synthesis include [22] (To get realistic scenes) optimization of algorithms implies developing faster algorithms that maintain holographic fidelity, such as wavefront recording planes [23], Look-up tables [24], dynamic acceleration techniques [25], deep learning-based accelerations [26] or methods combining the above.

Limitations in display technology introduce additional challenges. Computed CGHs contain information on the complex-valued amplitude; however, most SLMs are designed to modulate only amplitude or phase. Consequently, these limitations lead to reductions in visual quality when compared to fully complex-valued modulation. The primary issues that degrade image quality are related to edge effects, speckle noise, contrast reduction, and the appearance of unwanted diffraction orders, among others. To mitigate these issues, substantial research efforts focus on complex-amplitude encoding [27], image restoration techniques [28], temporal averaging methods [29], non-iterative approaches [30], iterative methods [31, 32] and camera-in-the-loop techniques [14]. To designing optical devices based on holographic elements, algorithms must take in account the complex impulse response to optical basic stimulus [33] in base to actual SLMs technology.

Monte–Carlo integration offers a comprehensive approach to achieving physically accurate lighting simulations [1]. This method involves the utilization of a randomly selected subset of all rays that require tracing to produce an image of nearly flawless quality, thereby minimizing any perceptible degradation. Crucial considerations in this context include the optical path followed by the rays from their source to the CGH plane, incorporating relevant interactions with scene objects, as well as the intensity of the rays.

In this context, an important question arises: Can we reduce the computational cost associated with CGH synthesis? This paper addresses this question by introducing the concept of Partial Monte–Carlo Sampling (PMCS) for 3D scenes CGH computation. Following this, another key question emerges: What impact do PMCS-based computational techniques have on the image quality of CGHs?

Our CGH methodology enables the generation, processing, and rendering of holograms from synthetic 3D objects. Therefore, this study primarily focuses on evaluating holograms generated using PMCS methodologies. The primary objective is to alleviate the computational burden involved in CGH generation while simultaneously assessing the quality of the reconstructed image compared to images obtained from CGHs using all available rays. To provide comprehensive insights, this study encompasses image acquisition through simulation and laboratory experimentation.

Section 2 presents the basic elements of the process of creating a synthetic 3D scene, its storage in a CGH, and subsequent retrieval, introducing PMCS to the calculus process. Section 3 shows the results obtained during CGH workflow. Section 4 demonstrates the results relating PMCS-based computation times to image quality, and finally, Section 5 summarizes the main contributions and future lines of work.

2 | Theoretical Foundations

A hologram is a recording of an interference pattern that can reproduce a 3D light field using diffraction. In general, a hologram is a recording in the form of an interference pattern. It can be created by capturing light from a real scene, or it can be generated by a computer, in which case it is known as a computer-generated hologram (CGH), which can show virtual objects or scenes in 2D or 3D.

With the aim in mind of generating realistic scenes from a virtual initial scene in 2D or 3D using CGH, it is necessary to use and combine several well-known techniques and procedures. The first technology to use is named Computer Graphics, which provides tools to describe a scene [1]: it is necessary to take into account the geometry, behavior of materials under light, the lighting through the description of the behavior of the different possible light sources used, camera position or algorithmic parameters, as some examples.

The second technology used has to do with the calculation of the CGH. Essentially consists of understanding how the wavefront, resulting from the interaction of light with the matter described in the previous step, propagates from the 2D or 3D scene to the plane where we let's record the hologram [34, 35]. This plane being digital, it is defined through pixels organized in a matrix of the appropriate dimensions. For it, this wavefront can be registered as a matrix of complex numbers representing the amplitude and phase observed by each basic element (pixel) of this plane.

The third technology has to do with the reconstruction of the 2D or 3D scene, it is based on understanding how, the amplitude and phase information recorded in the hologram perturbs the propagation of a well-defined wavefront (for example, a flat wavefront) and how it evolves through space [36].

This process can be carried out through mathematical modeling (simulation) or in the laboratory. In the case of the laboratory, it is necessary, to transfer the adequate information to a physical device known as a spatial light modulator (SLM). In our case, it records the components of the phases.

The results obtained both in simulation and in the laboratory demonstrate the coherent nature of the light used to illuminate the hologram: Diffractive effects or speckle [37] appear, severely limiting the quality of the obtained image. Therefore, the last part of the process followed deals with improvements in image quality. There are different techniques for this such as statistical filters (e.g., time multiplexing) or filters based on transformed operators.

The indicated process would be the usual one for obtaining CGHs and visualizing them, starting from a suitable point cloud belonging to a 3D scene. Now let's do the following reflection, if we start from a scene 3D sampled by N points and the hologram is composed of $W \times H$ pixels, the number of initial rays that must be launched is $N \times W \times H$ once the illumination that each point in the scene receives from the interaction of light with matter has been calculated. It can be said that the process is very expensive, numerically speaking. In order to alleviate this effort, this article proposes using a selective Monte–Carlo sampling, which will be described later.

The entire process followed appears summarized in Figure 1, and in the following subsections, we detail some relevant aspects of each stage of it.

2.1 | Implementation Details of About Aspects of Computer Graphics

In contemporary society, the widespread adoption of photorealistic computer graphics is evident across various sectors, including film, video games, product design, and architecture. A key aspect driving advancements in image synthesis is the utilization of physics-based rendering techniques. These methods focus on accurately modeling light scattering physics to achieve both visual authenticity and predictive capabilities. Across industry and research in physics-based rendering, there exists a robust framework supporting these standards.

Our geometrical model is used to define our scene polygon-based methods, and the ray tracing paradigm is used for scene rendering. Ray tracing, in particular, is a widely recognized technique employed for modeling illuminated scenes in computer graphics. This method revolves around tracing individual rays of light as they traverse through a virtual scene, interacting with its surfaces along the way. Through these interactions, ray tracing facilitates the generation of highly realistic images, making it an indispensable tool in the field of computer graphics.

An exhaustive classification of different physically based rendering can be found on authoritative source [1] where different of these algorithms and their source code are adequately described.

2.2 | Implementation Details About Aspects of CGH

Computer Generated Holograms (CGH) are implemented on a real or virtual Spatial Light Modulator (SLM), which can be

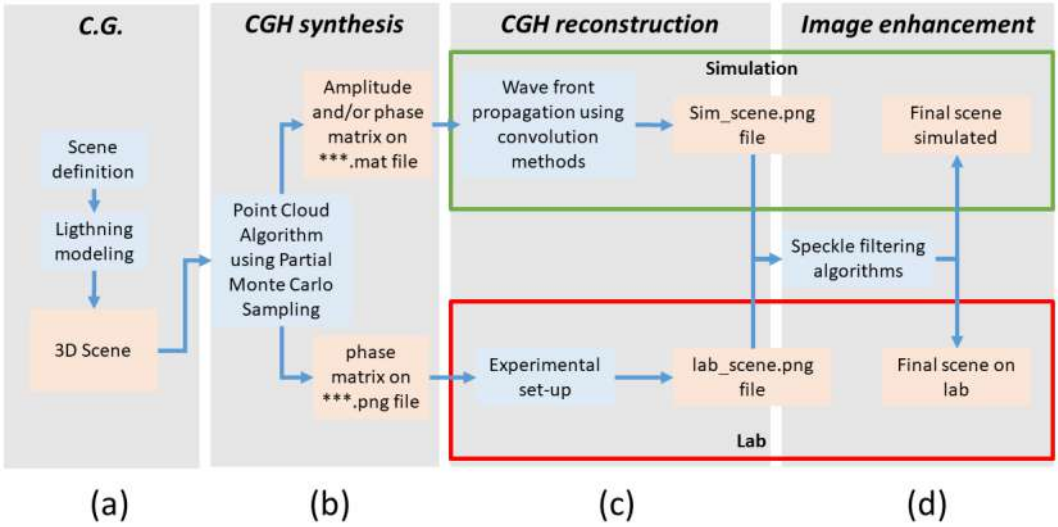

FIGURE 1 | Main workflow. Four phases are defined: (a) Scene synthesis (by Computer Graphics), (b) CGH synthesis, (c) Scene reconstruction, and (d) Image enhancement. The latter two can be carried out either through simulation or in the laboratory. Processes used in each stage are highlighted in blue, while data/files obtained are highlighted in orange.

represented as a matrix structure of $W \times H$ elements (pixels) of a given size. In this work, we use the geometrical characteristics of the phase only SLM PLUTO device [38] (1920×1080 pixels, 8 μm) to perform the registration and reconstruction processes of the calculated CGH and to compare the simulation results with those obtained in the laboratory. In the case of SLM in the laboratory, only the phase information calculated for each SLM pixel can be considered. For the simulated SLM we can also include the amplitude information. To generate a hologram for a 3D scene, it is necessary to capture a portion of the light wavefront produced by the scene on a specified plane (SLM plane). This plane is divided into $W \times H$ pixels, and each of the S scene points selected, generates a spherical wave that contributes to each one of all SLM pixels.

Various approaches are employed for CGH synthesis. These methodologies can be broadly categorized into several techniques [17]. Firstly, there are methods grounded in point clouds, which involve tracking the propagation of discrete points within the scene through space. These approaches often incorporate approximations, such as the Fresnel approximation. Secondly, geometric primitives serve as the foundation for another set of techniques. Additionally, layer-based methods exploit the convolution properties of a wavefront in one plane with a kernel, enabling the propagation of the wave to another plane through transform operators [39]. These approaches incorporate approximations, such as the Fresnel approximation.

In the present work, we use a hybrid technique of point clouds (based on discretizing the scene on samples) and ray tracing (for modeling light transport). The main concept involves selecting the point cloud of the scene based on a specific set of rays originating from each element of the CGH that each acts as a point of view (an eye of the classical ray tracing method) and extending towards its illumination sources.

The light emitted by a scene can be modeled as a set of spherical waves $\mathcal{U}_i(\boldsymbol{q})$, originating from each point on the object $\boldsymbol{p}_i$ and reaching every point $\boldsymbol{q}$ on the CGH. The *complex amplitude* of each spherical wave at a point $\boldsymbol{q}$ on the hologram plane is given by (see Figure 2a):

$$\mathcal{U}_i(\boldsymbol{q}) = U_i(\boldsymbol{q})\, e^{i\varphi_i(\boldsymbol{q})} \tag{1}$$

where $U_i(\boldsymbol{q})$ and $\varphi_i(\boldsymbol{q})$ represent, respectively, *real amplitude* and *phase* of the wave at the CGH plane, which can be obtained as

$$\begin{aligned} U_i(\boldsymbol{q}) &= U_{i,0}\frac{1}{\|\boldsymbol{q}-\boldsymbol{p}_i\|} \\ \varphi_i(\boldsymbol{q}) &= \varphi_{i,0} + \frac{2\pi}{\lambda}\|\boldsymbol{q}-\boldsymbol{p}_i\| \end{aligned} \tag{2}$$

where $U_{i,0}$ is the wave amplitude at its origin $\boldsymbol{p}_i$, phase offset $0 \leq \varphi_{i,0} \leq 2\pi$ (or initial phase) is the phase of the wave at its origin (an initial condition for each point of the scene), λ is wavelength, and $\|\boldsymbol{q}-\boldsymbol{p}_i\|$ is the distance the wave has travelled from point $\boldsymbol{p}_i$ to point $\boldsymbol{q}$.

The complete object wave $\mathcal{U}(\boldsymbol{q})$ is the superposition of all the elementary waves $(\mathcal{U}_1(\boldsymbol{q}), \ldots, \mathcal{U}_n(\boldsymbol{q}))$ originating on all points in our scene:

$$\mathcal{U}(\boldsymbol{q}) = \sum_F \mathcal{U}_i(\boldsymbol{q}) = \sum_{i=1}^{n_F} \mathcal{U}_i(\boldsymbol{q}) = \sum_{i=1}^{n_F} U_i(\boldsymbol{q}) e^{i\varphi_i(\boldsymbol{q})} \tag{3}$$

with F the full set of samples over the scene, numbered from 1 to n_F.

2.3 | Implementation Detail of Monte–Carlo Sampling: Utilizing all or a Portion of the Visible Points in the Scene

Calculation of the object wave $\mathcal{U}(\boldsymbol{q})$ over the CGH plane is performed by discretization of the CGH, so a set of discrete points (pixels) $\boldsymbol{q}_j$ is chosen, evenly distributed on the CGH, like in traditional computer imagery, and a discrete set of $\mathcal{U}(\boldsymbol{q}_j)$ values are determined. What we are effectively calculating, for every pixel in

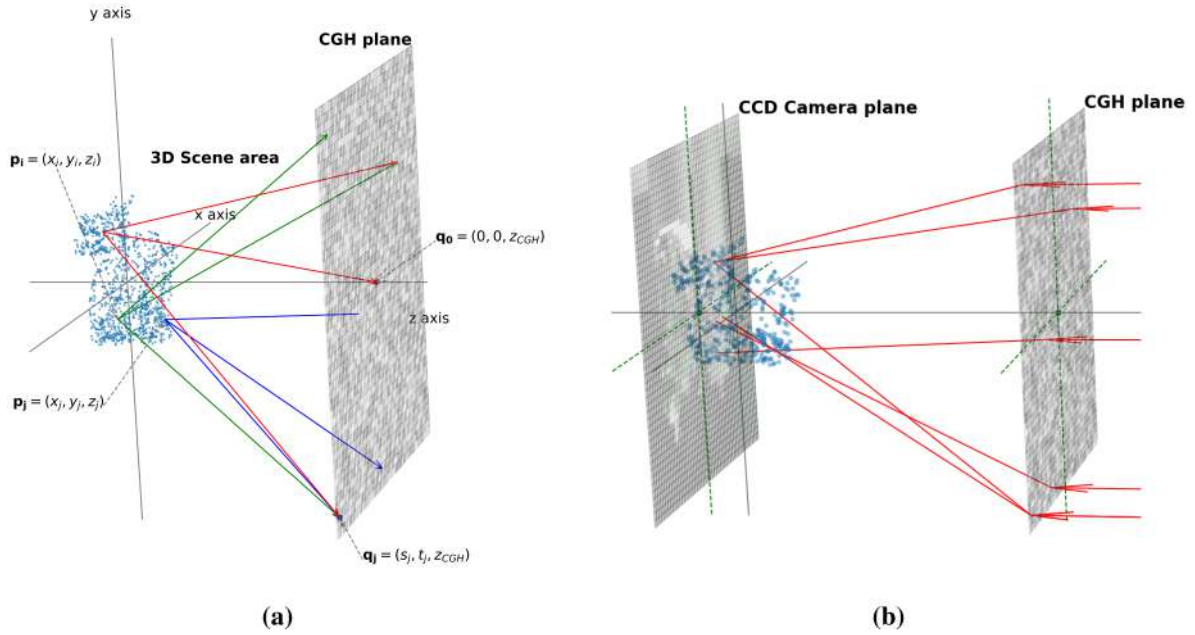

FIGURE 2 | General schemes: (a) Simplified geometry of wave propagation from a generic object to the hologram plane: $\boldsymbol{p_i} = (x_i, y_i, z_i)$ are points on the 3D scene, while points $\boldsymbol{q_j} = (s_j, t_j, z_{CGH})$ are the CGH pixel coordinates. Solid lines are the calculated light paths. On the PMSC algorithm, each 3D scene point contributes to an established percentage of random CGH pixels. (b) Reconstruction basis: The CGH is illuminated with a planar wavefront. The stored amplitude and phase information (or phase only) perturbs this wavefront, and the scene is reconstructed. Only points on the CCD plane are focused.

the hologram, is the integral of light received from the scene, and we sample the scene to compute this integral. Given the complexity of this equation, it must be calculated numerically. A suitable algorithm for this task could be a Monte–Carlo algorithm. Initially, we can employ a brute-force sampling approach. In computer graphics (CG), it has been demonstrated that Monte–Carlo methods are efficient for computing such integrals, thus, we can utilize them in this context.

The first use of the Monte–Carlo method involves randomly sampling the scene, so a subset S of points from the full set F of all possible scene points is selected to render the CGH, like in Computer Graphics algorithms, so the numerical approximation for the wave value becomes:

$$\mathcal{U}(\boldsymbol{q_j}) = \sum_{S} \mathcal{U}_i(\boldsymbol{q_j}) = \sum_{i=1}^{n_S} \mathcal{U}_i(\boldsymbol{q_j}) = \sum_{i=1}^{n_S} U_i(\boldsymbol{q_j}) e^{i\varphi_i(\boldsymbol{q_j})} \tag{4}$$

with S the selected subset of samples over the scene, numbered from 1 to n_S, and with $n_S < n_F$. The choice of the sample point set S is defined by the Computer Graphics algorithm used to render the scene.

As a form of attempted optimization of the indicated method, we introduce a Partial Monte–Carlo Sampling (PMCS) technique by selecting only a random subset R of the S scene samples to calculate the value at each hologram pixel. For this method to be accurate, the subset must adhere to a known probability density function $p(s)$ (PDF) (which determines the weight of each sample) and be different for each hologram pixel. Our calculation now becomes:

$$\mathcal{U}(\boldsymbol{q_j}) = \sum_{R} w_i \cdot \mathcal{U}_i(\boldsymbol{q_j}) = \sum_{i=1}^{n_R} w_i \cdot \mathcal{U}_i(\boldsymbol{q_j}) = \sum_{i=1}^{n_R} w_i \cdot U_i(\boldsymbol{q_j}) e^{i\varphi_i(\boldsymbol{q_j})} \tag{5}$$

with R the random set of sample points chosen from S, numbered from 1 to n_R, with $n_R \ll n_S$, and w_i the weight of sample i, defined by the PDF of the sampling as $w_i = \frac{1}{p_i} = \frac{1}{p(\mathcal{U}_i)}$.

The aim of the Monte–Carlo algorithms usage is to reduce noticeably the number of samples (in our case, waves) needed to calculate the value of each pixel on the CGH.

2.3.1 | PMCS Algorithm

To perform the integration across the entire set of discrete sources S, we use a Monte–Carlo integration approach. This method enables the calculation of the complex wavefront $\mathcal{U}(\boldsymbol{q_j})$ at a specific point on the hologram $\boldsymbol{q_j}$. The contribution $\mathcal{U}_i(\boldsymbol{q_j})$ from each source wave within the set S is cumulatively summed at this point (Equation 4). So, to apply Monte–Carlo methods we need to reformulate the sum as the calculation of an average value, as follows:

$$\mathcal{U}(\boldsymbol{q_j}) = n_S \left(\frac{1}{n_S} \sum_{S} \mathcal{U}_i(\boldsymbol{q_j}) \right) = n_S \left(\frac{1}{n_S} \sum_{i=1}^{n_S} \mathcal{U}_i(\boldsymbol{q_j}) \right) \tag{6}$$

Monte–Carlo integration allows for a reduction in the number of sources utilized for the calculation of the average. By selecting a randomized subset R of samples from S, where each sample ($\mathcal{U}_i$ wave) is chosen according to a probability density $p(\mathcal{U}_i)$, the integral can be computed as:

$$\mathcal{U}(\boldsymbol{q_j}) = n_S \left(\frac{1}{n_R} \sum_{i=1}^{n_R} w_i \cdot \mathcal{U}_i(\boldsymbol{q_j}) \right) = n_S \left(\frac{1}{n_R} \sum_{i=1}^{n_R} \frac{1}{p(\mathcal{U}_i)} \cdot \mathcal{U}_i(\boldsymbol{q_j}) \right) \tag{7}$$

with all terms defined as in Equations (4) and (5). Notice that, as in any Monte–Carlo algorithm, as the PDF affects the distribution of the samples, we must cancel it with the weights.

The choice of the probability density function $p(\mathcal{U}_i)$ can significantly enhance calculation accuracy, particularly when there is some prior knowledge about the distribution of $\mathcal{U}_i$. For instance, by favoring samples that are more likely to contribute higher values (based on a greater probability due to greater wave amplitude U_i), the accuracy of the integration can be improved. In scenarios such as ours, where a fully comprehensive complex scene serves as the source and no prior insight into the wavefront is available, the simplest choice for $p(\mathcal{U}_i)$ is a uniform random distribution, resulting in $p(\mathcal{U}_i) = \frac{1}{n_S}$. The resulting illumination of a complex scene can not be precisely predicted, and defining a numerical approximation to the real PDF of $\mathcal{U}_i$ would be feasible, but also very time-consuming as the number of source points on the scene can rise up to several million.

Moreover, the standard approach of defining a discrete cumulative distribution function (or $cdf(\mathcal{U}_i)$) and using it to have a precise sampling of $p(\mathcal{U}_i)$, via the inverse transform of a uniform random variable ξ, results in a very inefficient algorithm when implemented on the GPU (as our initial intent was to use also GPUs to speed up the calculations). This is caused by the inherently stateful algorithms used in random number generation, which implies the use of global information (variables) to store the RNG state, a fact that is very poorly managed by GPU hardware, causing a huge slow-down of the algorithm. Our intent to solve this problem is described in Subsection 2.4.2.

Uncertainty of the Monte–Carlo algorithm can be analyzed as follows. Instead of stating the algorithm as taking n_r samples for each pixel of the CGH, and repeating the process for each pixel, we could think about it as taking just *one* sample for each pixel to generate a sampled version $\mathcal{U}_s$ of the CGH, and repeating it n_r times. The mean square error ϵ of the approximation can be calculated as:

$$\epsilon = E((\mathcal{U} - \mathcal{U}_{n_r})^2) = (\mathcal{U} - \mathcal{U}_{n_r})^2 + \frac{V[\mathcal{U}_{n_r}]}{n_r}$$

where $\mathcal{U} - \mathcal{U}_{n_r}$ represents the *bias*, and $V[\mathcal{U}_{n_r}]$ the *variance* of the approximation. The method is not biased, as in the limit $n_r = n_s$, if we use all the available samples the result is exact. The variance can be approximated as

$$V[\mathcal{U}_{n_r}] = \frac{1}{n_r - 1}\sum_{s=1}^{n_r}(\mathcal{U}_s - \mathcal{U}_{n_r})^2$$

The root mean square error can be stated as $\sqrt{\frac{V[\mathcal{U}_{n_r}]}{n_r}}$, so the convergence of the method follows the $\frac{1}{\sqrt{n_r}}$ rule, as any MC algorithm. A possibility to measure $\mathcal{U}_s - \mathcal{U}_{n_r}$ (the *difference* between two wavefronts) is to measure the difference between the reconstructed images using the correlation coefficient. This behavior can be verified in Figure 8, both in every independent graph of CC vs sample rate and in the CC values vs the number of averaged images (N).

Despite this reduction in calculations, computational hologram synthesis still requires intensive processing. Therefore, from a practical perspective, it is essential to employ parallel processing techniques.

2.4 | Implementation Details About Parallel Processing, for CPU and GPU

Both lighting simulation (in order to generate sampling points in the 3D scene) and CGH calculation from those sampling points are computationally expensive processes, that require an insanely high amount of computing power. Parallel computing techniques can be applied to reduce computation costs.

2.4.1 | Parallel Path-Tracing

Lighting simulation for 3D scene rendering is performed with a Monte–Carlo path-tracing algorithm. This kind of algorithm is widely described in the literature [1]. Our implementation uses standard C++ threads to implement a shared memory parallel algorithm for rendering in CPUs, and CUDA parallel primitives on GPUs. Moreover, both CPUs and GPUs can be used at the same time: a shared queue of rendering tasks (image zones) is generated and every processor and/or GPU picks the next zone in the queue to render. The number of zones in which the image is partitioned is adjusted so it gives a good dynamic load balancing between processors of different computing power (CPUs and GPUs).

The result of the lighting simulation step is an enhanced version of what in common renderers is just an image frame buffer, which we call *geometry buffer* or *g-buffer* (in the style of deferred rendering algorithms). It contains also all the information needed for the CGH calculation: besides the color (light spectrum) in a given scene point, it also stores ray information, point depth, position, and initial phase for our wavefront.

2.4.2 | Parallel CGH Calculation

It should be noted that path-tracing rendering algorithms focus on the calculation of light *intensity* (power) on scene points, but wave propagation requires using the *complex amplitude* of the light wavefront, information obtained in the rendering phase must be corrected to use amplitude instead of intensity.

CGH calculation involves propagating the amplitude of the wavefront from the sampling points in the *g-buffer* to the pixels on the hologram plane. For a typical full HD resolution of 1920 × 1080 in both the sampling grid and the CGH array, and estimating about 100 *flops* (floating point operations) the cost of calculating each point-to-point interaction, this leads to about a 400 TFlops (Tera-flops) of calculations. Calculation of each value on the CGH can be performed independently of any other value, so this process can be parallelized over the domain of the CGH area.

This can be reduced by means of PMCS techniques, as depicted in Figure 2a, selecting a subset of R source points on the *g-buffer* as described in Subsection 2.4.1. This selection must comply with some requirements:

- The source point set chosen must be random for each pixel on the CGH (see Figure 2a).
- As the source point is a discrete set, repetition of source points for the same CGH pixel must be avoided.

To accomplish these needs, a vector of indexes on the source points is built and randomly shuffled. For the calculation of each pixel on the SLM, a random position on this vector is selected, and next R indexes are used to choose source points on the *g-buffer*, so it guarantees that each pixel gets a random subset of indexes not correlated with neighboring pixels.

This process could also be used on GPU to speed-up the calculation of the CGH. But applying this technique in GPU poses one additional problem: the speed of random access to source points from the *g-buffer* is greatly affected by the amount of cache of the processing unit. As GPUs offer much lower cache sizes than CPUs, this random access to data lowers the overall performance of the algorithm on the GPU. It is still an open research area on how to modify the algorithm to make it more cache-friendly while maintaining the true randomness needed for the Monte–Carlo algorithm to be correct. In this work, only the CPU implementation is addressed.

2.5 | Implementation Details About Scene Reconstruction

Once the CGH has been calculated, the next step is to use it to generate a realistic view of the 3D scene, in the sense of full depth and parallax perception for the observer. To accomplish this task, the original light wavefront must be reconstructed, by modifying a simple wavefront (typically a flat beam with the direction of propagation perpendicular to the CGH plane), with the information stored on the CGH.

2.5.1 | Wavefront Propagation from CGH

The wavefront at any plane parallel to the initial one can be calculated, effectively simulating wavefront propagation. This propagation process between two parallel planes can be achieved using Fourier optics, commonly employing kernels such as the Fresnel, angular spectrum, Fourier transform, or Rayleigh–Sommerfeld kernels, each suited to specific propagation conditions and distances in wavefront simulation [13].

This method only requires a couple of Fourier transforms to propagate the wave, so it can be implemented very efficiently via FFT. If wave section U_0 (complex amplitude) is known at a $z = 0$ coordinate, the value U_z at another z position can be calculated as:

$$U_z = \mathcal{F}^{-1}(\mathcal{F}(U_o).P(z)) \tag{8}$$

where $P(z)$ is a propagation function that depends only on the distance z, and $\mathcal{F}$, $\mathcal{F}^{-1}$ are the direct and inverse Fourier transform operators.

Given that all information contained within a CGH resides on a single plane, the propagation of the wavefront can be reformulated as for a planar wavefront. To propagate a planar wavefront, methods both in the spatial domain (convolution) or in the frequency domain (angular spectrum method—ASM) can be used. Both kinds of methods can be efficiently computed numerically using the Fast Fourier Transform (FFT) algorithm [13]. In this work, the ASM is used.

2.5.2 | Improvement of Final Scenes: Filtering

Speckle noise reduction and enhancement of image quality in CGH can be achieved through the several techniques grouped on [22] and [40]:

- Time-averaging methods: these methods reduce speckle noise by generating multiple CGHs and presenting them at a very fast rate to the observer, allowing his visual system to average the images due to vision persistence, which averages the images. They require generating as many holograms as deemed necessary for averaging, which results in a considerable increase in computation time. However, there are proposals that significantly reduce this time to calculate a single CGH (e.g., [41]), making image enhancement no longer the bottleneck of the entire calculation process.
- Non-iterative statistical methods: they are based on statistical analysis of speckles, so spatial filters can be optimally designed to reduce noise.
- Iterative methods: an iterative algorithm is used, that allows the tuning of some parameter of the hologram reconstruction (initial random phase, for example), by means of a measure of the reconstruction error.

In this work, we have selected a process similar to the averaging methods described in [28]. The chosen method involves introducing random phases to the scene points for each calculated CGH. The speckle pattern changes and statistical filtering techniques can be used to minimize its effect on the final image.

3 | Results of the Simulation and Laboratory

3.1 | Scene Defined

An image of a scene is generated. These figures are the result of stage (a) in Figure 1. To obtain it, we use a Cornell-Box-like scene, with semi-specular and diffuse surfaces (Figure 3a), including two well-known objects in the field of Computer Graphics, illuminated by four-point sources. The side walls are diffuse, colored green and red respectively, while the back wall is semi-specular. The lighting is based on four-point sources located at the upper corners of the Cornell box. The final scene used to generate the CGH is shown in Figure 3b. Both images have a resolution of 1920×1080 pixels.

3.2 | Calculation of the CGH

To compute the CGH, all values of the wavefront at the hologram plane must be calculated. Every source we have defined in the previous step must contribute to every pixel of the hologram.

(a) (b)

FIGURE 3 | Designed synthetic scene, inspired by the Cornell Box model. (a) Global view of the scene. (b) Detail of the global scene used in this work. (See Figure 1a).

For the final results, the algorithm used to generate the complete set of holograms (which takes into account the PMCS described in Section 4) can be represented as follows:

```
Define scene geometry and material properties
Define hologram recording frame size WxH and
 position
Define number N of holograms to average
Calculate the full set S of source points in
 the point cloud:
  For each pixel (i,j) out of (W, H) in the
   hologram frame
    Trace a ray into the scene
    Intersect the scene and select a point
     for the source point cloud
    Perform path tracing to simulate light
     propagation in the scene and determine
      illumination of the source point
For each percentage to test in PMCS
 Define the size of subset R of points from
   full set S to be used
 For each hologram in N
  Set a random initial phase for source points
    in the point cloud
  For each pixel (i,j) out of (W, H) in the
   hologram frame
     Select random subset R of source points
      from full set S
     Propagate waves from random subset R of
      points to hologram pixel
Average the N holograms to get the final
  result
```

This algorithm is an extension of a previous one developed for 2D scenes [42]. It is important to select a different subset R of source point set S for each pixel in the hologram, for the Monte–Carlo algorithm results to be correct. The result of the final calculation, when random subset R is the full set of source points S, is used as reference image.

The result is the CGH, which is essentially a matrix of complex numbers containing information on the amplitude and phase reaching each pixel of the hologram. The phase must be converted into an image file format (e.g., PNG) compatible with SLMs that allow only phase modulation. Figure 4 illustrates this information.

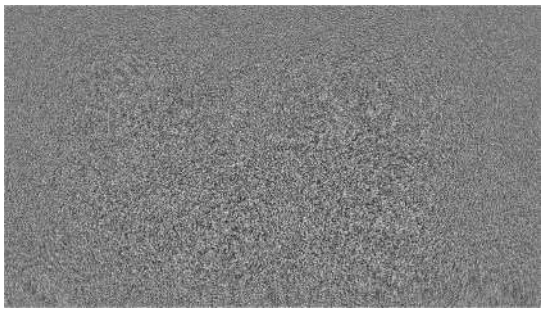

FIGURE 4 | Phase information from the CGH generated from the Figure 3b is encoded in a PNG-format file. (See Figure 1b).

3.3 | View Reconstruction from CGH

Figure 2b depicts the basic reconstruction process. To generate the actual image of the scene, light must follow the inverse path it previously took in the recording process. This is achieved by illuminating the CGH plane with a perpendicular plane wave. Consequently, a specific plane of the objects can be focused using a plane (referred to as the CCD camera plane in the figure). By shifting this plane along the z-axis, the entire scene can be explored.

In CGH computations, a random phase is introduced to broadly diffuse the object light, however, random interference arises at the reconstructed image plane, leading to the presence of speckle noise. Figure 5 shows it both on simulated and laboratory-reconstructed images.

3.4 | Improvement of Final Scenes: Filtering

Results are shown in Figure 6, for both simulated images (Figure 6a) and images obtained in the laboratory (Figure 6b). In any case, the use of image enhancement techniques always demands increased computational time.

In Figure 6a,b, the slight blur effect on the teapot is noticeable, due to being out of focus because of the depth of field, as the CGH

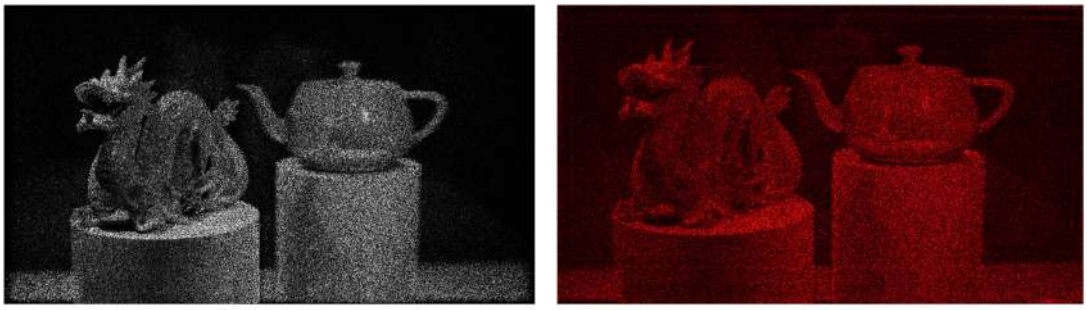

FIGURE 5 | Scenes obtained from the CGH prior to the filtering process. (a): Simulated. (b) Lab. (See Figure 1c).

FIGURE 6 | Scenes enhanced through N-image multiplexing techniques. ($N = 20$) (a) Simulated. (b) Lab. (See Figure 1d).

is visualized with the focus plane on the dragon's head. This effect is not as observed in the figures of Figure 5 because of the speckle. The reflection of both objects on the semi-reflective back wall is also observed.

In the case of images obtained in the laboratory, edge effects are observed due to the coherent nature of light (horizontal and vertical fringes) and to the fact that, due to the type of SLM used (PLUTO-2.1 phase only Spatial Light Modulator with 15.36 × 8.64 mm size and 8μ m of pixel pitch), amplitude information of the CGH can not be used.

4 | Partial Monte–Carlo Sampling (PMCS)

In the preceding sections, we observed that the process depicted in Figure 1 for designing a digital hologram (CGH), as well as the reconstruction and enhancement of the obtained scene, demands increasingly significant computational effort. Among all the described phases, the one related to CGH computation is the most resource-intensive.

Since the calculation time heavily depends on the hardware utilized, we have adopted the criterion of considering the time required for computing a hologram with all rays as the unit of time. Accordingly, the estimated time to complete each of the phases described in Figure 1 is presented in Table 1.

Several series of CGHs have been generated by varying the percentage of rays reaching from the scene to the CGH, following the Monte–Carlo criterion described in Section 2.3. Figure 7 shows

TABLE 1 | Estimated computation time for each of the phases indicated in Figure 1. The computation time of the CGH is taken as the reference scale. (a) Lighting simulation, (b) CGH synthesis, (c) Image reconstruction, (d) Image enhancement.

	(a)	(b)	(c)	(d)
Time cost (u.a.)	0.21	1	0.0001	0.0026

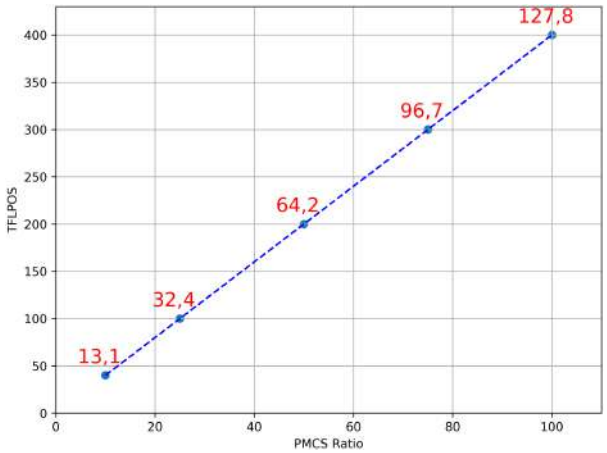

FIGURE 7 | Time cost for CGH synthesis using several PMCS samples ratios (percentage of total samples used, $(n_R/n_S) \times 100$): 10%, 25%, 50%, 75% and 100%. The X-axis represents the percentage of rays used in the PMCS algorithm. The blue points correspond to the values on the Y-axis (Tera-FLOPS required to complete the computation of a CGH). The included data (in red) indicates the time used (in minutes).

that time cost is close to linear with PMCS ratio rays used. Time data are obtained using an AMD EPYC 7313P CPU (16 cores/32 threads at 3.0 GHz, with 128 Mb cache). Due to the Monte–Carlo algorithm shuffles data, the cache memory plays a relevant role in improving this computing time.

It exists an almost linear relationship between the number of rays used and the computation time required for CGH calculation. However, it is essential to understand how this variation affects the quality of the final image.

There are various methods for comparing images with one another. Among them, The Pearson Correlation Coefficient [43] quantifies the linear relationship between the pixel values of two images. While it is useful for evaluating overall similarity and identifying linear patterns, it does not directly capture structural or shape differences, as it reduces the comparison to a single numerical value. In contrast, calculating the difference between images is ideal for detecting and visualizing specific, localized changes, and it is represented as an image. It allows for the detection of precise and localized changes between two images, making it useful for highlighting fine details, identifying movements, or detecting structural changes. To facilitate the comparison between an image I and a reference image R, we use the Pearson Correlation Coefficient (CC) defined as

$$CC = \frac{\sum_{N,M}^{p,q}(I_{p,q} - \bar{I})(R_{p,q} - \overline{R})}{\sqrt{\left(\sum_{N,M}^{p,q}(I_{p,q} - \bar{I})^2\right)\left(\sum_{N,M}^{p,q}(R_{p,q} - \overline{R})^2\right)}} \tag{9}$$

where I and R denote the two images being compared, p and q represent pixel row and column coordinates, $\bar{I}$ and $\overline{R}$ denote the average intensity values of images I and R, respectively, and $I_{p,q}$ signifies the intensity values of pixels at p, q in the image under comparison with the reference image $R_{p,q}$.

The Correlation Coefficient ranges from −1 to 1, where 0 signifies a lack of correlation between the images, while 1 implies a precise match between them and −1 indicates a perfect negative linear relationship. This metric allows us to evaluate similarity in terms of intensity patterns, to compare different versions of the same image or to detect linear changes. So, this metric allows for the assessment of changes in the quality of reconstructed images when varying parameters such as the ray ratio used in the PMCS algorithm or the number of images used to reduce speckles.

The original scene (Figure 3b) converted to grayscale serves as the reference image. The image for comparison is obtained by propagating the wavefront to a specific plane (in our case, coinciding with the position of the dragon's head). Therefore, there will always be a difference between images due to the blur of the rest of the scene (e.g., observe the difference in the teapot between Figures 3b and 6a).

In Figure 8, it can be observed that the worst case occurs when using a single CGH with a low percentage of rays (5%). As either of the two variables is increased, the overall performance improves. The best case is achieved when using a higher number of images and 100% of possible rays for CGH synthesis. Remarkably, this variation is not linear, as a high correlation coefficient

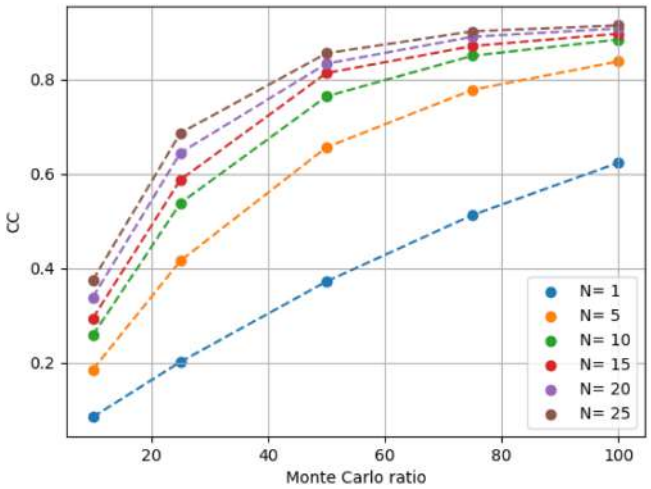

FIGURE 8 | Correlation Coefficient vs. PMCS ratio used (abscissas) and N (number of single CGH used to improve final images).

can be maintained even with a significant decrease in rays (i.e., computing time).

In Figure 9, a qualitative comparison of the details of images' behavior can be observed when modifying the percentage of rays (represented on the abscissa axis of Figure 8) and using multiple multiplexed images to minimize the effects of speckle (series represented with various colors in the same figure).

In this figure, the worst-case scenario is observed (i.e., only one CGH using a PMCS sampling of 10%), with a $CC = 0.05$, and the best scenario (25 CGHs for filtering using a PMCS sampling of 100%), with a $CC = 0.95$. As seen in Figure 8, the loss of image quality when decreasing these variables is not linear, maintaining values close to the ideal case (The CC values in these cases are around 0.9–0.8). This slight decrease makes it feasible to assess the savings of a considerable amount of computing time (e.g., 50%).

In the last column of Figure 9, for completeness, some of the details of the images obtained in the laboratory are presented. It is observed that the behavior is equivalent to those generated by simulation, although it is always slightly worse because only the phase information of the calculated CGH is utilized (see Figures 6b and 5b).

5 | Conclusions

This paper presents the fundamental elements to generate a CGH from a synthetic 3D Computer Graphics scene including the PMCS algorithm in the calculation process.

It elaborates on the key aspects of CGH synthesis and the retrieval of the original scene, introducing Partial Monte–Carlo Samples (PMCS). We outline the significant steps in the reconstruction process. Additionally, this work showcases the relationship between computation times and image quality through demonstrated results.

Section 2 presents the theoretical foundations of the different techniques used. They are presented following four phases: (a)

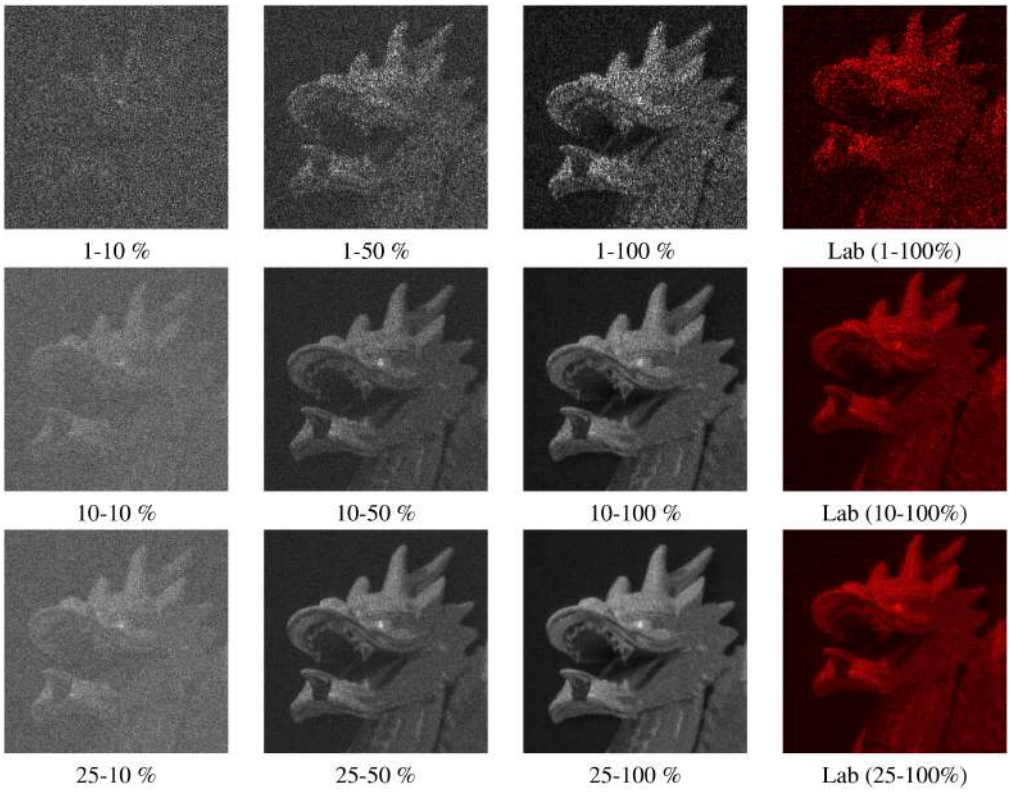

FIGURE 9 | Details of images obtained with various percentages of rays used (PMCS) and the number of images used to reduce speckle. They correspond to some of the data represented in Figure 8. The rows correspond to the number of images used for averaging and speckle removal (1,10 and 25), and the columns correspond to the percentage of rays used in the Monte–Carlo algorithm for synthesizing the CGHs (10%, 50%, and 100%). For completeness, laboratory results are presented for images obtained using 100% of rays (last column).

Scene synthesis (by Computer Graphics), (b) CGH synthesis, (c) Monte–Carlo Sampling, (d) Parallel processing, (e) Scene reconstruction, and (f) Image enhancement. The latter two can be carried out either through simulation or in the laboratory.

Section 3 presents in detail the results of the simulation and laboratory of the most prominent aspects of CGH synthesis of the synthetic 3D scene selected: it describes the process of CGH calculation using the full ray-tracing approximation, wavefront propagation for the reconstruction of a realistic (with depth and parallax perception) scene view, and enhancements of this process via filtering to reduce speckle effects.

And finally, Section 4 evaluates PMCS as a method to reduce the computational cost. Since the calculation time heavily depends on the hardware utilized, we have adopted the criterion of considering the time required for computing a hologram with all rays as the unit of time. This section shows the results relating computation times to image quality using the Correlation Coefficient (*CC*) to relate both variables, both in software and in-lab reconstruction processes, showing that computing time can be estimated depending on the quality desired, and that PMCS can generate images at similar visual and measured quality levels as the full ray-tracing solution, at a fraction of the computational cost. The utilization of PMCS enables a reduction in the computation time of an SLM without significantly compromising the final quality achieved. The results are consistent with previous work [42], which focused on CGH synthesis for 2D scenes. In addition, this work includes the possibility of changing the scene's lighting patterns and improving the final result by means of speckle filtering techniques.

Future Work: Future work will be related to improving the PMCS algorithm on specific architectures such as GPU in order to reduce even more this computing time. Another line of work is related to the use of PMCS to include full parallax in the scene obtained from the CGH.

Author Contributions

Juan A. Magallón: conceptualization; software; investigation; methodology; data curation; formal analysis; visualization; writing – review and editing. **Alfonso Blesa:** conceptualization; methodology; investigation; formal analysis; writing – review and editing; writing – original draft; funding acquisition; project administration; resources. **Francisco J. Serón:**

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

Data results may be obtained from the authors upon reasonable request.

References

1. M. Pharr, J. Wenzel, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation* (Cambridge, MA: MIT Press, 2023).

2. B. C. Kress, "Optical Architectures for Mixed-Reality Headsets," *Optical Architectures for Augmented, Virtual, and Mixed-Reality Headsets* 258 (2020): 258–268.

3. S. H. Lee and W. Däschner, "Low Cost High Quality Fabrication Methods and CAD for Diffractive Optics and Computer Holograms Compatible With Micro-Electronics and Micro-Mechanics Fabrication," *Diffractive Optics and Optical Microsystems* 5 (1997): 133–138, https://doi.org/10.1007/978-1-4899-1474-3_12.

4. H. P. Herzig, *Micro-Optics: Elements, Systems and Applications* (London, UK: Taylor & Francis, 1997).

5. E. V. Rabosh, N. V. Petrov, and N. S. Balbekin, "Analog-To-Digital Conversion of Information Archived in Display Holograms: I. Discussion," *Journal of the Optical Society of America A* 40, no. 4 (2023): B47–B56, https://doi.org/10.1364/JOSAA.478498.

6. P. Zhou and J. H. Burge, "Optimal Design of Computer-Generated Holograms to Minimize Sensitivity to Fabrication Errors," *Optics Express* 15, no. 23 (2007): 15410–15417, https://doi.org/10.1364/OE.15.015410.

7. C. C. Guest, H. Farhoosh, K. S. Urquhart, S. H. Lee, and M. R. Feldman, "Computer Aided Design of Computer Generated Holograms for Electron Beam Fabrication," *Applied Optics* 28, no. 16 (1989): 3387–3396, https://doi.org/10.1364/AO.28.003387.

8. C. Martinez, O. Lemonnier, F. Laulagnet, et al., "Complementary Computer Generated Holography for Aesthetic Watermarking," *Optics Express* 20, no. 5 (2012): 5547–5556, https://doi.org/10.1364/OE.20.005547.

9. Y. L. Li, D. Wang, D. Wang, N. N. Li, Q. H. Wang, and Q. H. Wang, "Fast Hologram Generation Method Based on the Optimal Segmentation of a Sub-CGH," *Optics Express* 28, no. 21 (2020): 32185–32198, https://doi.org/10.1364/OE.403252.

10. X. Yang, H. B. Zhang, and Q. H. Wang, "A Fast Computer-Generated Holographic Method for VR and AR Near-Eye 3D Display," *Applied Sciences* 9, no. 19 (2019): 4164, https://doi.org/10.3390/APP9194164.

11. K. Yamaguchi, M. Onishi, and Y. Sakamoto, "Evaluation of Transmission Performance for Streaming CGH Video Based on Computer Holography and Electro-Holography," *International Conference on Information Networking* 1 (2024): 689–694, https://doi.org/10.1109/ICOIN59985.2024.10572059.

12. F. Navarro, S. Castillo, F. J. Serón, and D. Gutierrez, "Perceptual Considerations for Motion Blur Rendering. ACM Transactions on Applied," *Perception* 8, no. 3 (2011): 1–15, https://doi.org/10.1145/2010325.2010330.

13. J. W. Goodman, "Introduction to Fourier Optics," in *Introduction to Fourier Optics*, 3rd ed. (Englewood, Colorado: Roberts & co. Publishers, 2017), 1.

14. L. Shi, B. Li, C. Kim, P. Kellnhofer, and W. Matusik, "Towards Real-Time Photorealistic 3D Holography With Deep Neural Networks," *Nature* 591, no. 7849 (2021): 234–239, https://doi.org/10.1038/s41586-020-03152-0.

15. Y. Peng, S. Choi, N. Padmanaban, and G. Wetzstein, "Neural Holography With Camera-In-The-Loop Training," *ACM Transactions on Graphics* 39, no. 6 (2020): 1–14, https://doi.org/10.1145/3414685.3417802.

16. L. Chen, H. Zhang, L. Cao, and G. Jin, "Non-iterative Phase Hologram Generation With Optimized Phase Modulation," *Optics Express* 28, no. 8 (2020): 11380, https://doi.org/10.1364/oe.391518.

17. D. Blinder, M. Chlipala, T. Kozacki, and P. Schelkens, "Photorealistic Computer Generated Holography With Global Illumination and Path Tracing," *Optics Letters* 46, no. 9 (2021): 2188, https://doi.org/10.1364/ol.422159.

18. P. W. M. Tsang, T. C. Poon, and Y. M. Wu, "Review of Fast Methods for Point-Based Computer-Generated Holography [Invited]," *Photonics Research* 6, no. 9 (2018): 837, https://doi.org/10.1364/prj.6.000837.

19. Y. Pan, Y. Wang, J. Liu, X. Li, and J. Jia, "Fast Polygon-Based Method for Calculating Computer-Generated Holograms in Three-Dimensional Display," *Applied Optics* 52, no. 1 (2013): A290–A299, https://doi.org/10.1364/AO.52.00A290.

20. M. Sun, M. Sun, M. Sun, et al., "Acceleration and Expansion of a Photorealistic Computer-Generated Hologram Using Backward Ray Tracing and Multiple Off-Axis Wavefront Recording Plane Methods," *Optics Express* 28, no. 23 (2020): 34994–35005, https://doi.org/10.1364/OE.410314.

21. T. Ichikawa, K. Yamaguchi, and Y. Sakamoto, "Realistic Expression for Full-Parallax Computer-Generated Holograms With the Ray-Tracing Method," *Applied Optics* 52, no. 1 (2013): A201–A209, https://doi.org/10.1364/AO.52.00A201.

22. D. Blinder, T. Birnbaum, T. Ito, and T. Shimobaba, "The State-Of-The-Art in Computer Generated Holography for 3D Display," *Light: Advanced Manufacturing* 3, no. 3 (2022): 572–600, https://doi.org/10.37188/lam.2022.035.

23. T. Shimobaba, J. Weng, T. Sakurai, et al., "Computational Wave Optics Library for C++: CWO++ Library," *Computer Physics Communications* 183, no. 5 (2012): 1124–1138, https://doi.org/10.1016/J.CPC.2011.12.027.

24. M. E. Lucente, and E. M. Lucente, "Interactive Computation of Holograms Using a Look-Up Table," *Journal of Electronic Imaging* 2, no. 1 (1993): 28–34, https://doi.org/10.1117/12.133376.

25. S. C. Kim, M. W. Kwon, E. S. Kim, and X. B. Dong, "Fast Generation of Video Holograms of Three-Dimensional Moving Objects Using a Motion Compensation-Based Novel Look-Up Table," *Optics Express* 21, no. 9 (2013): 11568–11584, https://doi.org/10.1364/OE.21.011568.

26. L. Cao, K. Liu, J. Wu, and X. Sui, "High-Speed Computer-Generated Holography Using an Autoencoder-Based Deep Neural Network," *Optics Letters* 46, no. 12 (2021): 2908–2911, https://doi.org/10.1364/OL.425485.

27. L. G. Neto, Y. Sheng, and D. Roberge, "Full-Range, Continuous, Complex Modulation by the Use of Two Coupled-Mode Liquid-Crystal Televisions," *Applied Optics* 35, no. 23 (1996): 4567–4576, https://doi.org/10.1364/AO.35.004567.

28. V. Bianco, P. Memmolo, M. Leo, et al., "Strategies for Reducing Speckle Noise in Digital Holography," *Light: Science & Applications* 7, no. 1 (2018): 1–16, https://doi.org/10.1038/s41377-018-0050-9.

29. J. Amako, H. Miura, and T. Sonehara, "Speckle-Noise Reduction on Kinoform Reconstruction Using a Phase-Only Spatial Light Modulator," *Applied Optics* 34, no. 17 (1995): 3165–3171, https://doi.org/10.1364/AO.34.003165.

30. E. Buckley, "Real-Time Error Diffusion for Signal-To-Noise Ratio Improvement in a Holographic Projection System," *Journal of Display Technology* 7, no. 2 (2011): 70–76.

31. R. Gerchberg, "A Practical Algorithm for the Determination of Phase From Image and Diffraction Plane Pictures," *Optik* 35 (1972): 237–246.

32. P. Chakravarthula, E. Tseng, T. Srivastava, H. Fuchs, and F. Heide, "Learned Hardware-In-The-Loop Phase Retrieval for Holographic Near-Eye Displays," *ACM Transactions on Graphics* 39, no. 6 (2020): 186, https://doi.org/10.1145/3414685.3417846/SUPPL_FILE/3414685.3417846.MP4.

33. E. Zlokazov, "Methods and Algorithms for Computer Synthesis of Holographic Elements to Obtain a Complex Impulse Response of Optical Information Processing Systems Based on Modern Spatial Light Modulators," *Quantum Electronics* 50, no. 7 (2020): 643–652, https://doi.org/10.1070/QEL17291/XML.

34. P. Hariharan, "Optical Holography: Principles," *Techniques and Applications* 1 (1996): 163–180, https://doi.org/10.1017/CBO9781139174039.

35. K. Matsushima, "Introduction to Computer Holography," *Series in Display Science and Technology* 1 (2020): 153–186, https://doi.org/10.1007/978-3-030-38435-7.

36. E. Cuche, P. Marquet, and C. Depeursinge, "Digital Holography for Quantitative Phase-Contrast Imaging," *Optics Letters* 24, no. 5 (1999): 291–293.

37. J. Goodman, "Some Fundamental Properties of Speckle," *Journal of the Optical Society of America* 66, no. 11 (1976): 1145–1150.

38. PLUTO-2 Phase Only Spatial Light Modulator Reflective Holoeye Photonics AG.

39. P. Lobaz, "Reference Calculation of Light Propagation Between Parallel Planes of Different Sizes and Sampling Rates," *Optics Express* 19, no. 1 (2011): 32, https://doi.org/10.1364/oe.19.000032.

40. Z. S. Li, Y. W. Zheng, Y. L. Li, D. Wang, and Q. H. Wang, "Method of Color Holographic Display With Speckle Noise Suppression," *Optics Express* 30, no. 14 (2022): 25647–25660, https://doi.org/10.1364/OE.461294.

41. P. J. Christopher, R. Mouthaan, V. Bheemireddy, and T. D. Wilkinson, "Improving Performance of Single-Pass Real-Time Holographic Projection," *Optics Communications* 457 (2020): 1–15, https://doi.org/10.1016/j.optcom.2019.124666.

42. A. Blesa, J. Magallón, and F. J. Serón, "Partial Monte Carlo Sampling for Computer Generated Holograms," *Engineering Reports* 6, no. 1 (2024): e12673, https://doi.org/10.1002/eng2.12673.

43. A. Kaur, L. Kaur, and S. Gupta, "Image Recognition Using Coefficient of Correlation and Structural SIMilarity Index in Uncontrolled Environment," *International Journal of Computer Applications* 59 (2012): 32–39, https://doi.org/10.5120/9546-3999.