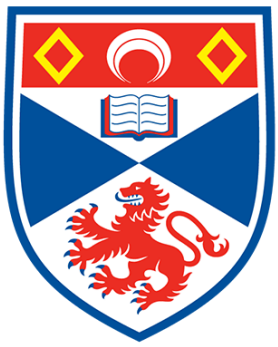
The Design of Computer-generated Holograms for Augmented and Virtual Reality Applications



University of
St Andrews

Alexander Richardson
ar376@st-andrews.ac.uk

Supervised by Professor Andrea Di Falco, Professor Jonathan Keeling and Dr Donatella Cassettari

August 2024

Contents

Abstract

Near-field Computer Generated Holography has the potential to revolutionise augmented and virtual reality. Singular variable decomposition with a cost-function minimisation approach can offer high-quality real-time hologram image generation. This approach, though promising, has not yet produced adequate image generation as the minimisation algorithm has struggled to converge to a correct solution efficiently. With certain improvements such as more appropriate starting conditions, this approach could yield impactful results.

Introduction

Computer-generated holography (CGHs) is the use of computer algorithms to generate holograms - three-dimensional images formed by interference from a laser or other coherent light sources. CGH has a broad range of applications, from medical imaging to education and training. This research focuses on the application of augmented and virtual reality (AR/VR) as it involves near-field holograms. CGH holds the potential to be the future of AR and VR with enhanced depth perception and improved image quality. Furthermore, CGH allows for a true 3d representation enabling users to naturally focus on different depths, combatting the fatigue of vergence-accommodation conflict. This future has yet to be realised due to complications regarding the time complexity of CGH algorithms, resulting in poor frame rates[1].

Recent advancements in near-field CGH have focused on overcoming the major challenges of computational complexity and low image quality[2], which are critical for creating high-quality, real-time holographic displays. Near-field CGH calculations are more computationally complex due to the nature of light propagation. Far-field CGH have the benefit of using the Fraunhofer approximation which simplifies the calculation with the assumption that the distance between the hologram and SLM is sufficiently large[3]. This allows the use of Fourier transforms to handle diffraction, greatly reducing computational load.

My investigation involves an SLM with 50×50 programable pixels. Each pixel is able to impose a phase retardation from 0 to 2π. The resulting digital image is calculated to reconstruct a target image of intensities in the near-field. The programming work was done in Python with significant aid from the Numpy library for matrix operations. The program first calculates a propagation matrix relating input pixels to output pixels, this matrix is then decomposed with singular variable decomposition to speed up the minimisation work. This decomposed propagation matrix is then used by a cost function and a cost derivative function which calculates the cost derivative with respect to each phase on the SLM. These functions are then used by the minimisation algorithm which attempts to find the phases which most reduce the cost. The decomposed propagation matrix is subsequently utilized by a cost function and its derivative function to compute the cost associated with certain phase configurations. These functions are then employed by the minimisation algorithm, which seeks to identify the phase configurations that most effectively minimise the cost. These phases can be used in conjunction with the propagation matrix to construct the final image. The Scipy and Matplotlib libraries were also used to a significant effect.

The Propagation Matrix

The propagation matrix relates input pixels to output pixels in the hologram image. It effectively assesses the contribution of each input pixel to every output pixel. To create this we used the kernel of the Rayleigh-Sommerfield integral with the exclusion of the complex amplitude as this can be later to used to get a resulting image. The propagation matrix labelled M is given by the following formula:

$$M = \frac{e^{krj}}{2\pi r^2}(-\ k + \frac{1}{r}) * dz$$

where

$$k = \frac{2\pi}{\lambda}$$

r is the distance from the input pixel to the output pixel
dz is the distance from the input plane to the output plane

This matrix is 2 dimensional with its size being all input pixels multiplied by all output pixels. Both the input and output pixels are stored as arrays of coordinates, allowing a 3d hologram surface to be represented with a single array. This matrix can be multiplied with a set of complex amplitudes, with one value per input pixel in order to calculate the resulting hologram image.

Singular Variable Decomposition

Singular variable decomposition (SVD) effectively factorises a matrix into a rotation, followed by a scaling followed by another rotation. Given a matrix A of size $i \times o$, SVD deconstructs A so that:

$$A = U\Sigma V^T$$

where

U is an $i \times i$ orthogonal matrix
Σ is an $i \times o$ diagonal matrix with singular values
V^T is an $o \times o$ orthogonal matrix

A cutoff value then can be picked to ignore smaller singular values which contribute less to the structure of the matrix. From this value, the three matrices can be truncated, leaving behind s singular values and a final structure where:

U_s is an $i \times s$ matrix
Σ_s is an $s \times s$ matrix
V_s^T is an $s \times o$ matrix

Reconstructing the matrix A_s gives us a lower-rank approximation. This approximation improves the efficiency of the cost function and therefore the overall minimisation.

For this project, the propagation matrix was minimised with a cutoff of 10^{-3}. This reduced the number of singular variables to 885 from 2500. The singular variables were plotted for the target image (*Figure 1*) with roughly 885 of the values being equal. This indicates an inability to reduce the matrix further. This value was initially expected to be lower. Regardless, this technique still improves the efficiency of the cost function.

Design of the Cost Function

The cost function should have a global minimum that corresponds to the desired pattern. The design of the cost function represents a significant challenge as it must be efficient enough for the minimisation whilst ensuring no local minima are present. Additionally, a cost derivative function with respect to the phases was used in order to use the SVD propagation matrix and reduce the time complexity of the algorithm. The cost function subscripts refer to the dimensionality where i is the number of input pixels, o is the number of output pixels and s is the number of singular values left after SVD. The following equation was used:

$$C = \sum(P_o - \lambda|Y_o|^2)^2$$

using

$$Y_0 = \sum_s U_{os}\sigma_s Y_{si}$$

using

$$Y_{si} = (V^t)_{si} E_i e^{i\phi_I}$$

where

P_o is the target image

λ is some scaling parameter

$U_{os}\sigma_s(V^t)_{si}$ corresponds to $U\Sigma V^T$

E_i is the intensity envelope

ϕ are the phases

This cost function resembles a mean squared error calculation, importantly, this cost function has a scaling parameter, which rewards the minimisation algorithm for representing the shape of the image rather than just the average intensity.

A cost function derivative was also used to ensure the minimisation made use of the most efficient derivative. The following function was used making reference to the above constants and terms:

$$\frac{\partial C}{\partial \phi_i} = -\ 4\lambda \mathbb{R}(i \sum_s z_s Y_{si})$$

using

$$Z_s = \sum_s (P_o - \lambda \left|Y_o\right|^2) Y_o^* U_{os} \sigma_S$$

A cost derivative with respect to the scaling factor is also required making use of the above terms:

$$\frac{\partial C}{\partial \lambda} = -\ 2 \sum_o (P_o - \lambda \left|Y_o\right|^2) \left|Y_o\right|^2$$

These partial derivatives were calculated with the intention of avoiding $i \times n$ operations in order to speed up execution. Both partial derivatives were confirmed with a simple finite difference calculation.

Minimisation

The minimisation method chosen was the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm which is an iterative method for solving unconstrained non-linear optimisation problems. This algorithm is a second-order derivative method requiring the inverse of the Hessian, the limited memory variant updates this calculation at each iteration rather than a complete recalculation[4]. The limited memory variant provides considerably better time complexity for many-variable problems. BFGS has an approximate time complexity of $O(n^2)$ whilst L-BFGS has a time complexity of $O(kn)$ where $k << n$. The minimisation was initially run with bounds placed on the possible phases however these were removed to improve convergence as it stops the phases from reaching a bound and having to backtrack. The effectiveness of certain algorithms can be very problem-specific, so testing various minimisation algorithms for this specific problem was required[(5)].

Results

The minimisation suffered from long execution times and incomplete results as the algorithm was unable to converge onto a solution. Many attempts were made at fine-tuning the optimisation such as altering the initial starting conditions however the results were similarly weak. An example target image is shown below along with two calculated images with a scaling and non-scaling cost function.

Figure 1 - Target Hologram Image

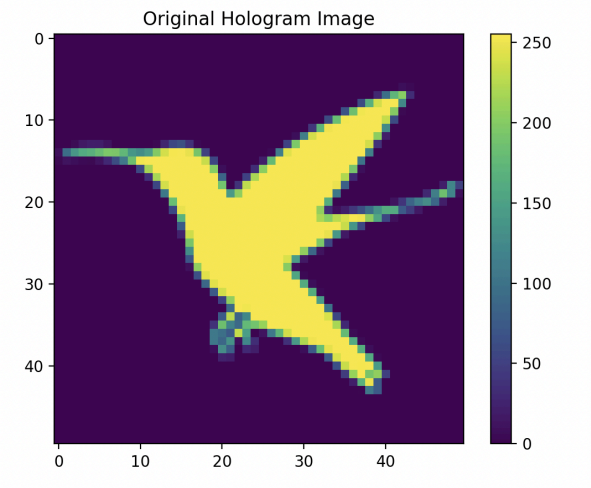


Figure 2 - Calculated Hologram Image without Scaling

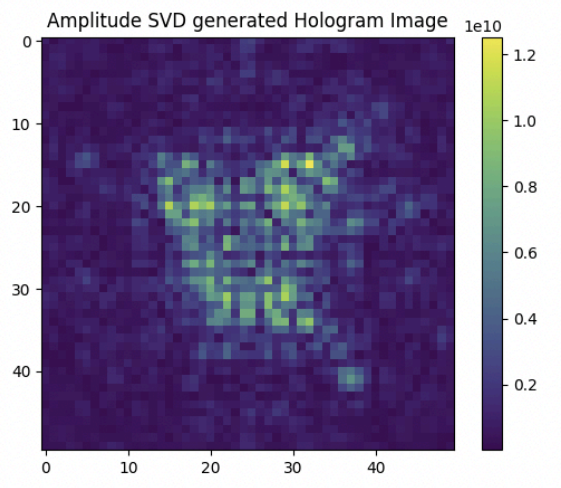


Figure 3 - Calculated Hologram Image with Scaling

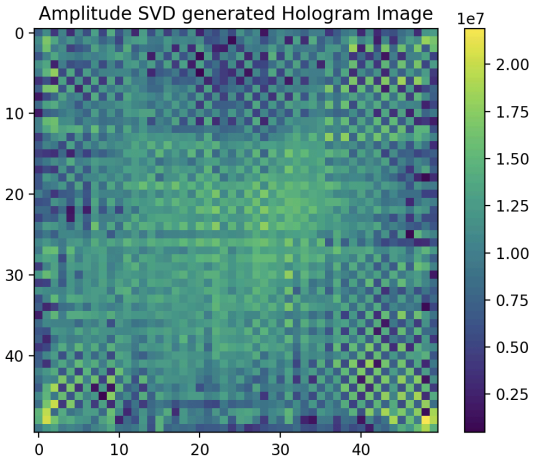
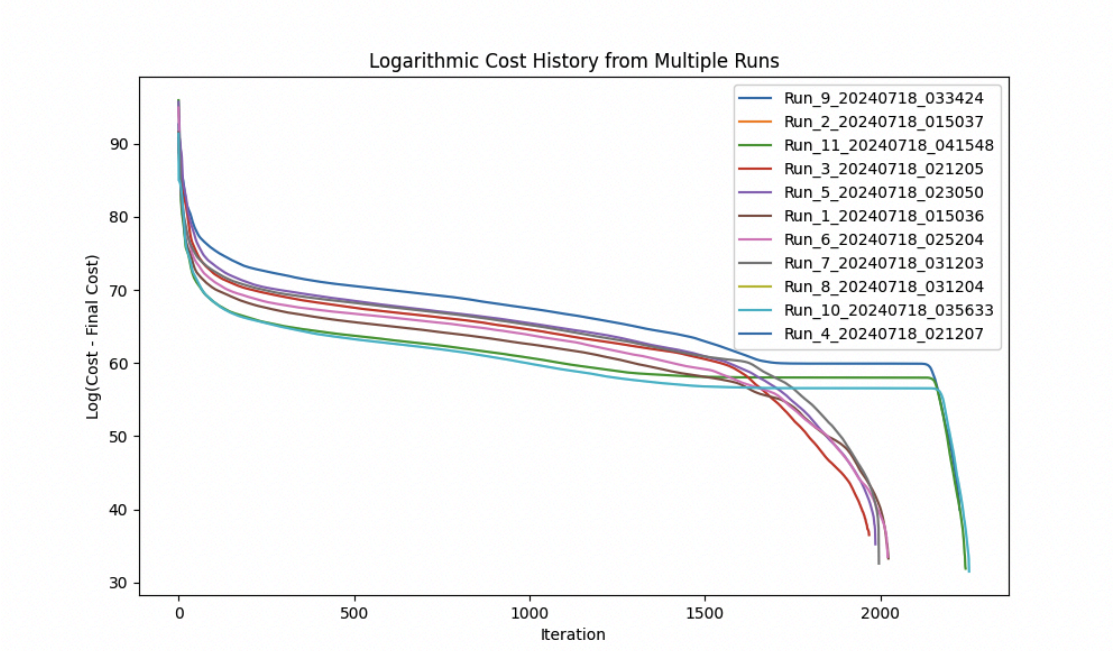


Figure 2 shows that the algorithm with the non-scaling cost function was able to produce some features of the desired hologram pattern, however, the overall pattern is still relatively weak. The scaling cost function was introduced to better reward the minimisation algorithm for finding the shape and yet *Figure 3* shows a chaotic image with no similarity to *Figure 1*. Despite the worsened image, the scaling approach was still decided as the more appropriate cost function. If a perfect result were to be achieved, the scaling cost function would have to be used due to fundamental issues with the non-scaling cost function. This issue is a priority of average intensity rather than shape. The scaling cost function was investigated further with the logarithmic cost history tracked from multiple runs to monitor the progression of the algorithm, the results can be seen below in *Figure 3*.

Figure 3 - Logarithmic Cost History Runs with Random Starting Conditions



The optimisation attempts can broadly be placed into two categories. Some runs terminated after very few iterations, whilst the majority terminated after ~ 2000 iterations. Both categories of runs failed to converge on a solution, with the error message 'Desired error not necessarily achieved due to precision loss.'. This indicates the algorithm encountered some limitations regarding floating point arithmetic. There was also some unusual behaviour with the scaling factor, many runs finished with a negative scaling factor which is indicative of failure to converge. The optimisation algorithm could benefit from a better choice of starting variables. An appropriate choice of phase and scaling factor could allow for an early convergence. The scaling factor could also be constrained in a range of appropriate values preventing the negative value.

In the context of this project, it is unlikely that local minima are cause for concern. It is far more likely that the structure of the space has many flat regions that make optimisation difficult. The fact this happens repeatedly reduces the effectiveness of potential mitigations such as random restarts. Some sort of memory mechanism could prevent the revisiting of certain areas.

It is possible the cost function could be amended to better the algorithm's ability to minimise, however, there are no obvious improvements to make. The introduction of the scaling factor greatly increased the minimisation computation time however it was required to better generate images.

The impact of singular variable decomposition whilst reducing the complexity of the matrix around 65% was expected to be more impactful. Its effectiveness should be explored with other hologram images as well as with different constants such as varying wavelength and distance between planes. This may lead to a greater reduction of singular variables and thus reduce computational load.

Conclusion

Near-field CGH has the possibility to revolutionise AR and VR technology. Whilst this new approach to calculating holograms has failed to realize this, there is potential in these numerical methods to yield significant results. For certain problems, it is possible that singular variable decomposition can provide a greater reduction of singular variables. With changes to the minimisation algorithm such as an improved choice of starting variables, it could see significant improvement in its ability to converge onto to the correct result.

Acknowledgements

I would like to thank Lord Laidlaw and the Laidlaw Foundation team for the opportunity to conduct this research. I would also like to thank my research advisors, Professor Andrea Di Falco, Professor Jonathan Keeling and Dr Donatella Cassetari for all their support and guidance.

References

(1)*Advancing real-time 3D holographic display: A breakthrough in computer-generated holography* (2024) *SPIE, the international society for optics and photonics*. Available at: https://spie.org/news/advancing-real-time-3d-holographic-display-a-breakthrough-in-computer-generated-holography#_=_(Accessed: 31 August 2024).

(2)Pi, D., Liu, J. and Wang, Y. (2022) *Review of computer-generated hologram algorithms for color dynamic holographic three-dimensional display*, *Nature News*. Available at: https://www.nature.com/articles/s41377-022-00916-3 (Accessed: 21 August 2024).

(3)Konijenberg, S., Adam, A.J.L. and Urbach, H.P. (2022) *6.6: Fresnel and Fraunhofer approximations*, *Physics LibreTexts*. Available at: https://phys.libretexts.org/Bookshelves/Optics/BSc_Optics_(Konijnenberg_Adam_and_Urbach)/06:_Scalar_diffraction_optics/6.07:_Fresnel_and_Fraunhofer_Approximations (Accessed: 31 August 2024).

(4)Kumar, U. (2022) *Numerical optimization based on the L-BFGS method*, *Medium*. Available at: https://towardsdatascience.com/numerical-optimization-based-on-the-l-bfgs-method-f6582135b0ca (Accessed: 31 August 2024).

(5)Liu, D.C. and Nocedal, J. (2021) *On the limited memory BFGS method for large scale optimization - mathematical programming*, *SpringerLink*. Available at: https://link.springer.com/article/10.1007/BF01589116 (Accessed: 31 August 2024).