McMaster University

Undergraduate Thesis

Department of Physics and Astronomy

Analyzing Computation Time for Holographic Reconstruction

Author:

Chloe Pomeroy

Supervisor:

Dr. Qiyin Fang

May 1, 2020

Abstract

Digital Holography has many applications and uses, particularly in the medical and biological fields. One of it's major drawbacks, however, is the need for the holograms to be reconstructed using phase and amplitude information collected at the sensor. Depending on the desired resolution, the holograms can take significant amounts of time to reconstruct and, for this reason, real-time analysis using holography isn't currently possible. Similarly, reconstructing videos takes quite a long time. In this project we analyze the affect of certain parameters on computation times using the HoloRec3D MATLAB toolbox. We find that decreasing the interval on z decreases the computation time, changing the pixel size has seemingly no effect on the computation time, and increasing the step size has a significant effect on the computation time. The ability to decrease computation time for reconstructions would bring digital holography even more into the medical and biological field and would greatly increase the benefits of using holography for things like urinalysis.

Contents

Chapter 1

Introduction

1.1 Background

Holography was first invented in 1948 by Dennis Gabor as a way to improve the resolution of the electron microscope. The resolving power of electron microscopes is limited by the spherical aberration of the lenses and trying to correct for the aberration is nearly impossible, which is why Gabor proposed holography as an alternative [1]. He realized that information about the amplitude and phase of the electron wave was contained in the diffraction pattern, and that it could be used to reconstruct the object field. He named it holography, because of the ability to record the 'whole' optical field using this principle. Because phase information contains depth information, holography can be used to reconstruct an image of an object's three-dimensional location. There are several types of holography, but in this project we will be focusing on in-line holography, where the reference beam is the same as the object beam and the object field is in general alignment with the reference beam [2].

One problem with holographic imaging is that there are two steps involved: image acquisition and image reconstruction and because reconstruction is necessary, real-time operation is difficult. For a number of applications, improving the computation time of holographic reconstruction would be beneficial. For example, holography is sometimes used in place of shadow imaging in urinalysis when scanning for bacteria because bacteria are transparent and affect the phase but not the intensity of the light that passes through them. In this case, real-time methods would be useful to monitor the development of infections [3]. Another application that would benefit is the holographic reconstruction of videos. Since each frame needs to be reconstructed separately, reconstruction of videos of samples can take a significant amount of computation time. In this project we look to analyze which parameters affect the computation time of a reconstructed hologram and what methods we could use to speed up reconstructions to achieve something closer to real-time analysis.

1.2 Types of Holographic Imaging

There are 2 main types of holography configurations that each have their own advantages. Gabor's initial configuration, called Gabor in-line holography, is simple to configure and has a large depth of field, but one major limitation is the presence of a twin-image artifact in the reconstructed holograms [4]. This twin-image occurs due to symmetry and often obscures the image of the object [5]. For sparse samples, the artifact can be eliminated by the simple assumption that the amplitude and phase of the light is uniform outside of the boundaries of individual objects. As a result, Gabor in-line holography is often used for particle imaging. It is still possible to reduce the twin-image artifact in samples that aren't sparse enough to make the previous assumption, but it takes a significant amount of computation time. The second main type of configuration, off-axis holography, bypasses the twin-image problem altogether. In an off-axis configuration, the reference and object waves are offset by an angle, which prevents the reference wave from overlapping with the twin image. Off-axis holography also has its disadvantages, as it's a much more complicated setup and requires a different method of reconstruction [4].

Luckily, for urinalysis and the other applications used in our lab, the samples are sufficiently sparse for us to easily eliminate the twin-image artifact in our reconstructions, and so this project will focus on in-line holography.

1.2.1 In-Line Holography

In-line holography describes a type of holographic imaging setup in which the reference wave and the object wave are on the same optical axis. In this type of holography, there is one incident wave that passes through the sample. The part of the wave that hits the object is scattered before hitting the sensor and thus obtains a phase and amplitude difference from the incident wave. This scattered wave is called the object wave. The other part of the wave passes through the sample and hits the sensor without scattering, so there is no difference in the phase or amplitude between this wave and the incident wave [6]. This other wave is called the reference wave.

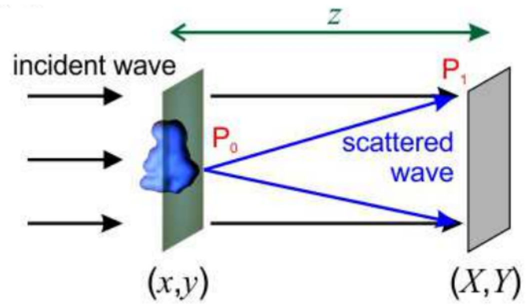


Figure 1.1 A graphic representation of an in-line holography setup with a plane wave. Reprinted from [6]

When the reference wave and the object wave meet each other at the sensor, the two waves are out of phase and an interference pattern appears. Through the interference pattern and the known information about the reference wave, it is possible to recover the phase and amplitude information of the object wave. It's important to note that the sample needs to be relatively sparse for this type of holography to work. In other words, this won't work if most or all of the incident wave is scattered, there needs to be a significant portion of the incident wave that reaches the sensor unchanged [7].

1.3 Reconstruction Methods

The most common way to reconstruct holograms is to multiply the hologram wave with the reference wave and then to back-propagate this wave into the object plane using the Fresnel-Kirchoff diffraction integral. This is known as the light back-propagation method of holographic reconstruction and is done computationally. Another method worth noting is the angular spectrum method, which uses no approximations and describes plane wave propagation using the propagation of its spectrum. When working with an off-axis holographic configuration, the angular spectrum method is used [4]. In this project we will focus on the light back-propagation method.

1.3.1 Light Back-Propagation Method

Each object can be described by a transmission function [6]:

$$t(x,y) = exp(-a(x,y))exp(i\phi(x,y)), \tag{1.1}$$

Where $a(x,y)$ is the absorption and $\phi(x,y)$ is the phase distribution when the incident wave scatters off the object. Transform functions are defined throughout literature and are different for objects with different shapes and sizes. As such, approximations about the expected imaged objects must be made and this method may not be ideal for imaging irregular objects. Equation 1.1 can also be rewritten as [6]:

$$t(x,y) = 1 + t'(x,y), \tag{1.2}$$

where $t'(x,y)$ is the change imposed on the incident wave by the object to create the object wave. This makes sense intuitively because it is clear that $t(x,y) = 1$ when the wave passes through to the sensor without scattering.

The wavefront past the object, called the exit wave, can be described:

$$U_{exit} = U_{incident} \cdot t(x,y) = U_{incident} + U_{incident} \cdot t'(x,y), \tag{1.3}$$

We can use the Fresnel-Kirchoff diffraction formula to decsribe the propagation of this wave towards the sensor. This formula is given by:

$$U_{detector}(X,Y) = -\frac{i}{\lambda} \iint U_{incident}(x,y) \cdot t(x,y) \frac{exp(ik|\vec{r} - \vec{R}|)}{|\vec{r} - \vec{R}|} dxdy, \tag{1.4}$$

where $k = \frac{2\pi}{\lambda}$.
The value $|\vec{r} - \vec{R}|$ is the difference between a point $\vec{r}$ in the object plane and a point $\vec{R}$ in the detector plane. The recorded hologram is given by:

$$\begin{aligned} H(X,Y) &= |U_{detector}(X,Y)|^2 \\ &= |R(X,Y)|^2 + |O(X,Y)|^2 + R*(X,Y)O(X,Y) + R(X,Y)O*(X,Y) \end{aligned}$$

In this equation, the second term is assumed to be small, and the last two terms are what cause the interference pattern at the detector. Before it's possible to reconstruct the hologram, it must be normalized by dividing out the background image. The first term in equation 1.5 is equivalent to the result we would get if there were no object present in out sample at all, and so when we divide out the background image, that term goes to one. We can write the normalized hologram as:

$$H_0(X,Y) = \frac{H(X,Y)}{B(X,Y)} - 1 \approx \frac{R*(X,Y)O(X,Y) + R(X,Y)O*(X,Y)}{|R(X,Y)|^2} \tag{1.5}$$

Since we subtracted one here, $H_0(X,Y)$ is just the interference term.
We can now reconstruct the hologram by multiplying equation 1.6 by the reference wave and then applying the Fresnel-Kirchoff diffraction integral:

$$U(x,y) \approx \frac{i}{\lambda} \iint R(X,Y)H_0(X,Y)\frac{exp(-ik|\vec{r} - \vec{R}|)}{|\vec{r} - \vec{R}|}dXdY, \tag{1.6}$$

This constructed wave corresponds to $t'(x,y)$ and so by adding one we can find $t(x,y)$, which then allows us to be able to find $a(x,y)$ and $\phi(x,y)$ by equation 1.1. The following steps summarize the reconstruction process [6]: 1. Calculate the Fourier transform of $H_0(X,Y)$
2. Simulate the Fourier transform of The Fresnel function
3. Multiply these two results together
4. Take the inverse Fourier transform of the result found in 3. This result will give $t'(x,y)$
Repeating this process many times for different values of $z = |\vec{r} - \vec{R}|$ gives the ability to reconstruct a three-dimensional image of the object plane.

Chapter 2

Methods

2.1 Image Reconstruction

In order to achieve a holographic image, the hologram must be reconstructed using the phase and amplitude information picked up by the sensor. The light back-propagation method and the spectral angle method mentioned in Chapter 1 are examples of these reconstruction methods. Many groups have implemented computer algorithms to reconstruct these images, such as the HoloRec3D MATLAB toolbox, which uses the light back-propagation method as described in section 1.3.1. In summary, The wave information collected at the detector is propagated backwards towards the object using the Fresnel-Kirchoff diffraction integral, and then using inverse Fourier transforms we are able to recover information about the optical field surrounding the object. Using this method, it's possible to create three dimensional images that include depth information [6]. For this project we will use MATLAB and the HoloRec3D toolbox, which was made available for download online [8].

2.1.1 Sample Image

I reconstructed the same image for each computation. this is the sample image I used for the reconstructions:

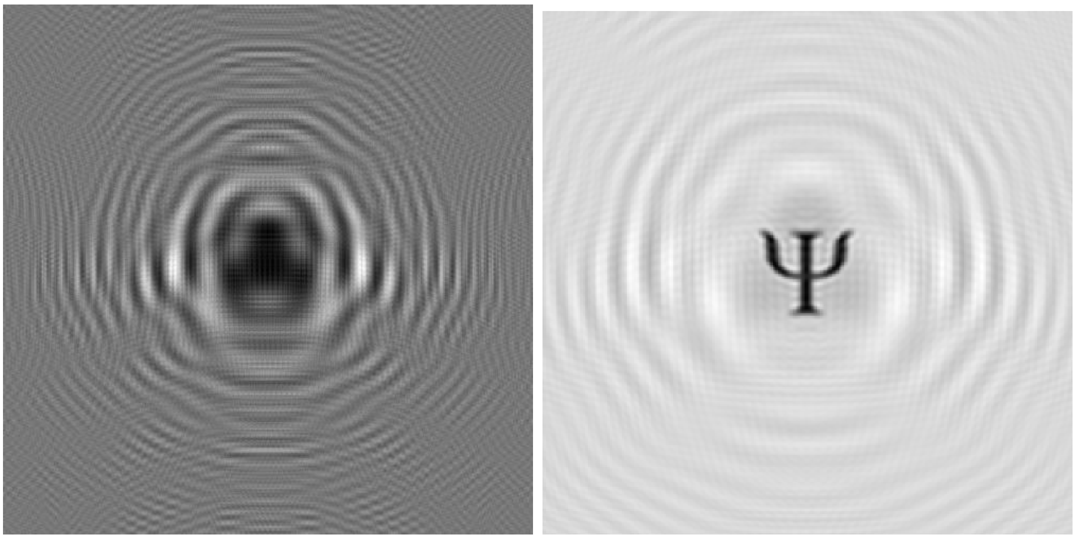


Figure 2.1 The test image used during the reconstructions in this project. These are both the same image.

2.2 Computer Specifications

2.2.1 Software

For this project we used MATLAB R2019B and the HoloRec3D Toolbox [8]. Only essential background processes were operating while performing the reconstructions.

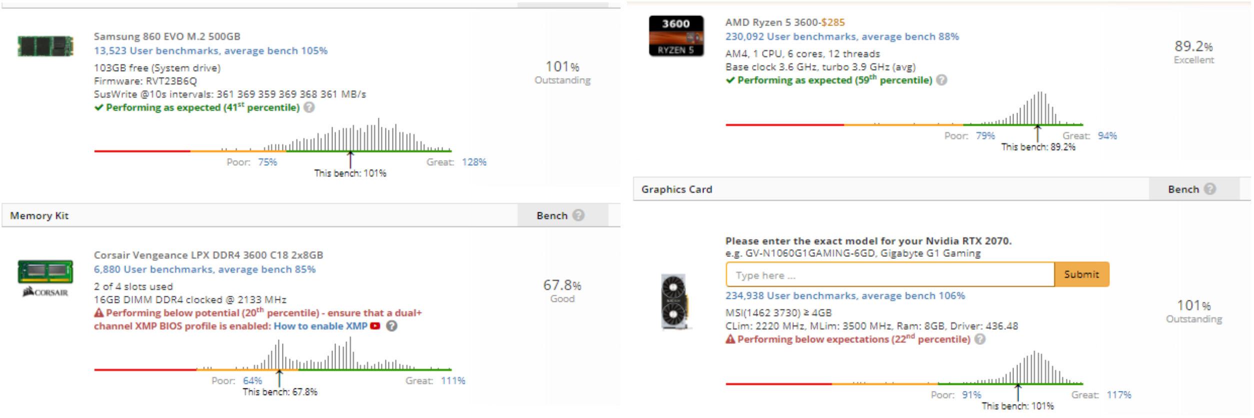
2.2.2 Hardware

MATLAB programs are processed using the central processing unit (CPU) unless they're written with specific functions that call the graphics processing unit (GPU) [9]. As such, the toolbox used in this project is processed by the CPU and thus the hardware with the greatest effect on the computation time is the CPU. The reconstructions were done on a PC with the following components:

- CPU: Ryzen 5 3600
  - 6 cores
  - 3.6GHz per core base frequency
- GPU: RTX 2070
  - 1410 MHz base clock

- SSD: Samsung 860 Evo SATA M.2

- Memory: 2x8Gb DDR4 3600



Samsung 860 EVO M.2 500GB
13,523 User benchmarks, average bench 105%
103GB free (System drive)
Firmware: RVT23B6Q
SusWrite @10s intervals: 361 369 359 369 368 361 MB/s
✓ Performing as expected (41st percentile)

Memory Kit | Bench

Corsair Vengeance LPX DDR4 3600 C18 2x8GB
6,880 User benchmarks, average bench 85%
2 of 4 slots used
16GB DIMM DDR4 clocked @ 2133 MHz
⚠ Performing below potential (20th percentile) - ensure that a dual+ channel XMP BIOS profile is enabled: How to enable XMP

AMD Ryzen 5 3600-$285
230,092 User benchmarks, average bench 88%
AM4, 1 CPU, 6 cores, 12 threads
Base clock 3.6 GHz, turbo 3.9 GHz (avg)
✓ Performing as expected (59th percentile)

Graphics Card | Bench

Please enter the exact model for your Nvidia RTX 2070.
e.g. GV-N1060G1GAMING-6GD, Gigabyte G1 Gaming

234,938 User benchmarks, average bench 106%
MSI(1462 3730) ≥ 4GB
CLim: 2220 MHz, MLim: 3500 MHz, Ram: 8GB, Driver: 436.48
⚠ Performing below expectations (22nd percentile)

Figure 2.2 The results of a benchmark test of the PC used for the reconstructions. The benchmark test was performed using the software User Benchmark

2.3 Measuring Computation Time

To measure and compare the computation time in MATLAB, we ran the profiler before running each simulation. After each simulation, we recorded the computation time for the function titles "Classical_reconstruction>pushbutton3_Callback". We recorded the self time as oposed to the total time because the self time is what is relevant to our analysis, as the self time is the time spent in a function excluding the time spent in it's child functions. One issue is that self time also includes overhead resulting from the process of profiling. When recording the computation time, we ran each simulation with the same parameter values 4 separate times and took the mean of the four recorded values.

Chapter 3

Results

For these reconstructions, we used the light back-propagation reconstruction part of the toolbox, and we specifically looked at Fresnel Transform as our method of reconstruction. When looking at changing one parameter, all other parameters were kept constant.
Note: Although they aren't visible because the values are so small, all graphs in this section do contain error bars, since each point on the graph is an average result over four trials.

3.1 Image results

By changing the reconstruction parameters, the reconstructed images themselves change. When the bounds on z were changed, the images got blurrier as z became further away from the object. When the lower bound was fixed, the images looked the same but contained additional frames for higher z values, all of which were less and less in focus the larger the upper bound for z became. When the upper bound was fixed, we could see that the images were becoming blurrier as the lower bound was raised, because the back-propagation was starting farther and farther away from the object.

Changing the pixel size didn't change the appearance of the resultant image in any way. In contrast, decreasing the step size resulted in images with many more frames, each frame wasn't necessarily more in focus, but because there were more total frames, these holograms did contain crisper frames than those with bigger

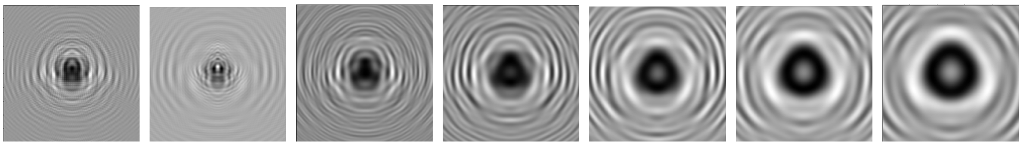


Figure 3.1 Each of these images were taken at the starting frame for different lower bounds, from left to right: 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, and 1.3

step sizes.

3.2 Changes to the Computation Time

3.2.1 Bounds on z

For this section, only the lower and upper bounds were changed between reconstructions. All other values were fixed as follows:

$\lambda = 0.532e - 6$

Pixel Size = 7e-6

Fill Factor of Camera = 0.8367

Step size = 0.01

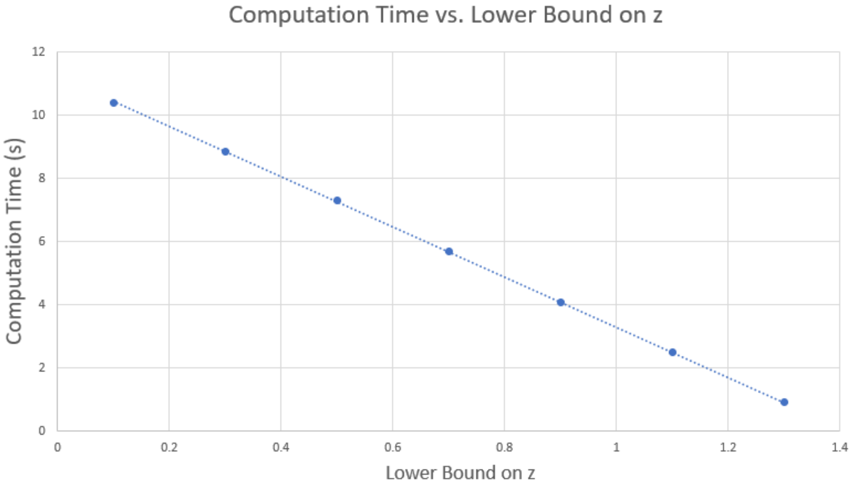


Figure 3.2 A graph demonstrating the relationship between the change in the lower bound and the change in computation time, for a fixed upper bound 1.4. This trendline has a slope of -7.9299

There is very clearly a linear relationship between the bounds on z and the computation time. By averaging the absolute values of the two slopes, we find

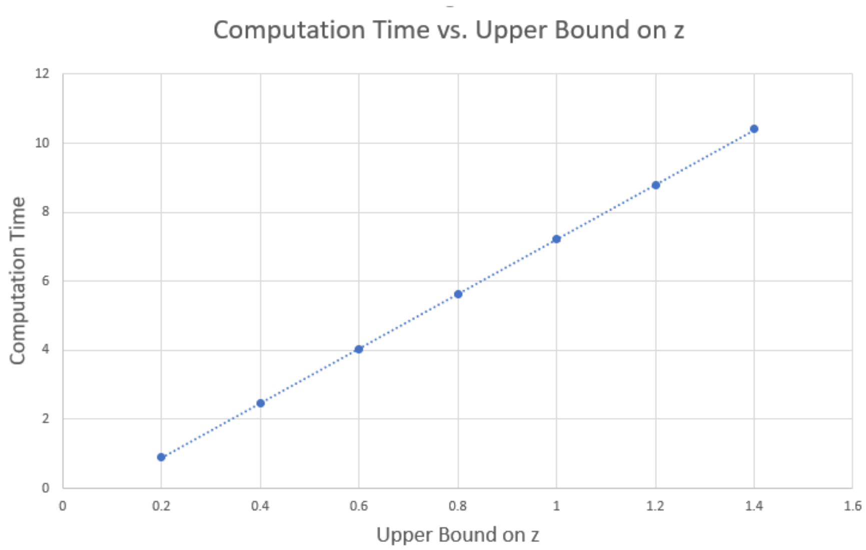


Figure 3.3 A graph demonstrating the relationship between the change in the upper bound and the change in computation time, for a fixed lower bound 0.1. This trendline has a slope of 7.9123.

that these two variables are related by a slope around 7.92 ± 0.02.

3.2.2 Step Size

For this section, only the step size was changed between reconstructions. All other values were fixed as follows:

$\lambda = 0.532e - 6$

Pixel Size = 7e-6

Fill Factor of Camera = 0.8367

Lower Bound = 0.1

Upper Bound = 0.8

This graph doesn't have a clear trendline, but it does look somewhat like there may be an exponential relation between the two variables.

3.2.3 Pixel Size

As with the previous sections, only the pixel size was changed while doing these reconstructions. All other values were fixed as follows:

$\lambda = 0.532e - 6$

Fill Factor of Camera = 0.8367

Lower Bound = 0.1

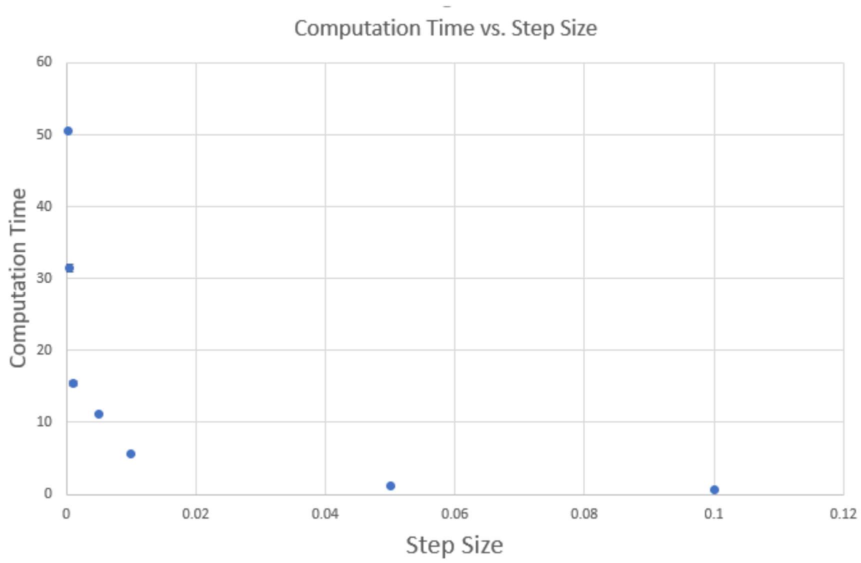


Figure 3.4 A graph demonstrating the relationship between the change in the step size and the change in computation time

Upper Bound = 0.8

Step Size = 0.01

There doesn't seem to be a consistent relation between the pixel size and the computation time, however the computation time also doesn't stay constant while the pixel size changes. These measurements were carried out multiple times each and are replicable, which makes it even more interesting that they seemingly have no reason.

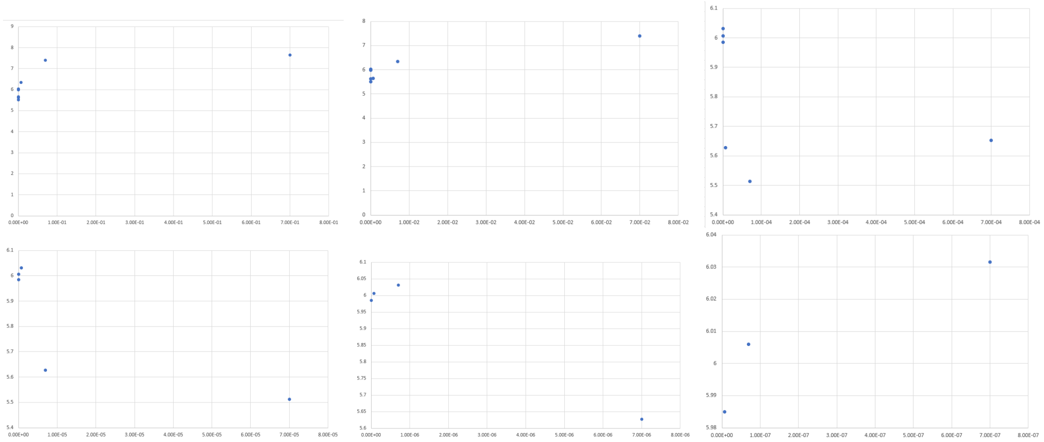


Figure 3.5 These graphs demonstrate the relationship between pixel size (x-axis) and computation time, in seconds (y-axis). Each graph is slightly more zoomed in than the last, as each of the tested values for pizel size is 10 times larger than the last, which makes it difficult to see the relationship between the points.

Chapter 4

Summary

4.1 Conclusion

A major limitation to the field of digital holography is the computation time for reconstructions. If it were possible to cut down computation times without compromising the resolution, then holographic videos could be reconstructed closer to real time than is possible today. From my results, it's clear that pixel size has no meaningful effect on the computation time and so it doesn't need to be a concern when designing holographic setups.

Changing the bounds on z does increase the computation time, however it doesn't always improve image resolution. Therefore, it can be said that it isn't particularly beneficial to increase the area of z that you are examining. If an object is known to be within certain z values, there is no benefit to performing the reconstruction beyond those values. In fact, one way to decrease computation time may be to minimize the z interval, which is especially doable in applications where the position of the object is somewhat known.

Finally, we found that the step size greatly increased the computation time of the reconstruction, but it also increased the resolution that we were able to find. The main question we have now is how much resolution is able to be lost in return for computation time? When the step size was 0.0001, we found that the average computation time was 315 ± 4 seconds, which is reasonable for only reconstructing one image, however, if we were attempting to reconstruct a video at 60 frames per second, one minute of video would take approximately 5 hours and 15 minutes to reconstruct. This may be too long for some medical and biological applications. When the step size was set to 0.001, the average computation time was 15.4 ± 0.3 seconds, which translates to around 15.4 minutes to reconstruct one minute of video. This seems much more reasonable and quite close to real time, however, we must make a decision whether to compromise the resolution in favour of computation speed or to compromise the computation speed in favour of image resolution.

4.2 Future Goals

The next step to this project would be to find a way to quantify resolution and numerically determine how much resolution is lost for each second gained in computation time. Being able to quantify what specific resolution is needed for each application of these principles would make it easier to determine exactly how quickly, with today's software, a hologram with the bare minimum resolution requirements could be reconstructed. Additionally, as mentioned briefly in section 2.2.2, MATLAB runs on the CPU, but it is possible to write a MATLAB program in such a way that the GPU is acting instead of the CPU. Multiple groups [9, 10] have done holographic reconstructions using GPUs instead of CPUs and have found success in achieving faster computation times. An interesting continuation of this project may be to rewrite HoloRec3D in terms of MATLAB functions that process on the GPU and observe how this could change or improve computation times.

References

[1] Gabor, D. “A New Microscopic Principle.” *Nature*, vol. 161, no. 4098, 1948, pp. 777–778., doi:10.1038/161777a0.

[2] Kim, Myung K. “Principles and Techniques of Digital Holographic Microscopy.” *Journal of Photonics for Energy*, 2010, p. 018005., doi:10.1117/6.0000006.

[3] Mahoney, Eric, et al. “Review—Point-of-Care Urinalysis with Emerging Sensing and Imaging Technologies.” *Journal of The Electrochemical Society*, vol. 167, no. 3, 2020, p. 037518., doi:10.1149/2.0182003jes.

[4] Yu, Xiao, et al. “Review of Digital Holographic Microscopy for Three-Dimensional Profiling and Tracking.” *Optical Engineering*, vol. 53, no. 11, 2014, p. 112306., doi:10.1117/1.oe.53.11.112306.

[5] Latychevskaia, Tatiana, and Hans-Werner Fink. “Solution to the Twin Image Problem in Holography.” *Physical Review Letters*, vol. 98, no. 23, 2007, doi:10.1103/physrevlett.98.233901.

[6] Latychevskaia, Tatiana, and Hans-Werner Fink. “Practical Algorithms for Simulation and Reconstruction of Digital in-Line Holograms.” *Applied Optics*, vol. 54, no. 9, 2015, p. 2424., doi:10.1364/ao.54.002424.

[7] Ozcan, Aydogan, and Euan Mcleod. “Lensless Imaging and Sensing.” *Annual Review of Biomedical Engineering*, vol. 18, no. 1, 2016, pp. 77–102., doi:10.1146/annurev-bioeng-092515-010849.

[8] M. Seifi, C. Fournier and L. Denis, ”HoloRec3D : A free Matlab toolbox for digital holography” (2012).

[9] Masuda, Nobuyuki, et al. “Computer Generated Holography Using a Graphics Processing Unit.” *Optics Express*, vol. 14, no. 2, 2006, p. 603., doi:10.1364/opex.14.000603.

[10] Garcia-Sucerquia, J., et al. “Real Time Numerical Reconstruction of Digitally Recorded Holograms in Digital in-Line Holographic Microscopy by Using a Graphics Processing Unit.” *Photonics Letters of Poland*, vol. 2, no. 4, 2010, doi:10.4302/plp.2010.4.12.