Holographic Particle Image Velocimetry: Computational Simulation and Reconstruction

by

Craig P. Earls
B.S. Physics
University of California at San Diego, 1989

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degrees of
Naval Engineer's Degree
and
Master of Science in Aerospace Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1999

Author ..... Craig Paul Earls
Department of Ocean Engineering
June 1, 1999

Certified by .. Jerome H. Milgram
Jerome H. Milgram
Professor of Ocean Engineering
Thesis Supervisor

Accepted by ..... Arthur B. Baggeroer
Art Baggeroer
Ford Professor of Engineering
Chairman, Departmental Committee on Graduate Studies

Read by ..... Mark Drela
Mark Drela
Associate Professor of Aeronautics and Astronautics
Thesis Reader

Accepted by .....
Jaime Perraire
Professor of Aeronautics and Astronautics
Chairman, Departmental Committee on Graduate Studies

Holographic Particle Image Velocimetry: Computational Simulation and Reconstruction

by

Craig P. Earls

B.S. Physics
University of California at San Diego, 1989

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degrees of

Naval Engineer's Degree

and

Master of Science in Aerospace Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author .....
Department of Ocean Engineering
June 1, 1999

Certified by .....
Jerome H. Milgram
Professor of Ocean Engineering
Thesis Supervisor

Accepted by .....
Art Baggeroer
Ford Professor of Engineering
Chairman, Departmental Committee on Graduate Studies

Certified by .....
Mark Drela
Associate Professor of Aeronautics and Astronautics
Thesis Reader

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chairman, Departmental Committee on Graduate Studies

Holographic Particle Image Velocimetry: Computational Simulation and Reconstruction

by

Craig P. Earls

Submitted to the Department of Ocean Engineering
on June 1, 1999, in partial fulfillment of the
requirements for the degrees of
Naval Engineer's Degree
and
Master of Science in Aerospace Engineering

Abstract

Holographic Particle Image Velocimetry (HPIV) is a novel technique for measuring the complete fluid flow around a body. Advances in computing power make this technique practicable for the first time.

Currently popular techniques for experimentally determining fluid flow around a test body rely on measuring the flow at a single point and moving the sample point during the experiment. This implicitly time averages the data. HPIV can measure the flow field in a volume nearly instantaneously by sequentially "holographing" the seeded flow field.

Until recently the only practical method of processing the HPIV data was to physically reconstruct the optical image from the hologram and scan through the resulting volume using microscope optics. This is a very slow process.

Recent advances in computing power make it possible to directly analyze the digitized holograph (digitized using a high resolution scanner) This reduces the post-processing time from several hours to less than 1-hour with much less expensive equipment.

The primary contribution of this thesis is an advanced multi-platform graphical user interface hologram synthesizer for use in calibrating digital HPIV reconstruction algorithms.

Thesis Supervisor: Jerome H. Milgram
Title: Professor of Ocean Engineering

Acknowledgments

Many people have have supported me through my time at MIT, and in producing this thesis. Those who directly contributed to my successful completion of the X-IIIA curriculum include CDR Mark Welsh, and Professor Jerry Milgram.

My non-academic life was immeasurably enriched by my wife Adrienne, my dog Duncan, and my cats, Tiggerr, Emma, and Lizzy.

The most important people to acknowledge here are my father and mother, Michael and Karon Earls. Obviously without them, I would never have gotten here. Thanks, Dad and Mom...

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

This thesis presents an advanced multiplatform graphical user interface hologram synthesizer to produce calibration holograms of seeded flow fields. The synthesizer, HoloSynth, runs under X11R6 with OpenGL, and Microsoft Windows NT. Complete source code is contained in Appendix A. A brief introduction to particle holography and holographic particle velocimetry is presented as well as results from computational reconstruction of the hologram generated by the synthesizer.

1.2 Motivation

Until very recently, particle image velocimetry (PIV) was limited to measuring fluid flow velocities only in a single plane. A sheet of laser light was passed through the flow which contained neutrally buoyant particles. A double exposure of the particles onto either film or a Charge Coupled Device (CCD) camera allows the two velocity components in the illuminated plane to be determined.

The PIV method has recently been extended to three dimensions by Katz et. al. [1] by making a hologram of the flow field. Three dimensional Holographic Particle Image Velocimetry (HPIV) uses either in-line or off-axis holographic techniques (see Section 2.2). Once the hologram has been recorded, the image is reconstructed by reversing the recording process and using a three dimensional optical scanner to

digitize the image for analysis by computer. This process can take several hours.

An alternative to scanning the reconstructed image is to digitize the hologram and allow a computer to reconstruct the image in software. The recent advances in computing power should make this process far faster than the mechanical/optical scanning process now used.

Digitizing the hologram can be done in two ways. The initial method avoids the three dimensional scanning of the reconstructed image by using a high performance flat bed scanner to digitize an enlargement of the transparency produced in the normal holographic process. Alternately, a high resolution scanner can be used directly on the hologram transparency. The flat bed scanning process is well understood, since the transparency can be enlarged to overcome resolution problems with the scanner. However, it still requires recording the hologram on film with the attendant film processing.

A more advanced technique would be to capture the hologram using a Charge Coupled Device (CCD) Camera. The hologram would be immediately available in digital form, and no film or processing would be involved.

Regardless of the hologram digitization technique the computational machinery for reconstructing the image must be calibrated. It far easier to produce a synthetic hologram for calibration than to set up a tightly controlled experiment and actually record a hologram.

1.3 Requirements

The initial goals of computationally reconstructed HPIV are to be able to accurately determine the flow field in a cubic volume a few inches on a side. The fluid speeds will be typical of those used in hydrodynamics research. The quantifiable requirements are;

- Must be able to resolve a 10μm particle. This corresponds to the most conveniently sized neutrally buoyant seed particles used for Laser Doppler Anemometry.

- Accurate coverage within a $125cm^3$ (5cm cube) volume. This will rise as the technique is perfected, and computing power increases.

- Unambiguously detect fluid speeds $< 10m/s$. This accounts for free stream plus allowance for turbulence and model scale factors.

The first and second requirements dictate the size and resolution of the recording device.

Since holography is essentially a linear process, multiple exposures can be used instead of a rapid succession of exposures. So rather than design a system to rapidly capture two separate holograms, the recording medium is exposed two or more times without being altered between exposures. This results in multiple real images for each particle in the volume, the analysis process must identify the multiple images with a particular particle. This alleviates the need for a high speed film traversal mechanism and removes CCD output data rate as a concern.

The third requirement sets the pulse lengths and pulse intervals of the laser used to generate the hologram. If the pulses are too long the diffraction patterns will be smeared from the particle motion. The greater the interval between pulses, the harder it will be to uniquely identify particle images with a single particle. The pulse length and interval requirements determine the recording device sensitivity and laser output power.

Chapter 2

Review of Holographic Technique

2.1 Overview

A hologram is a two dimensional recording that can produce a three dimensional image. Typically electromagnetic (light) waves are recorded, but recording of acoustic waves and matter waves have also been made[2]. Electromagnetic holography differs from normal photography in that it encodes phase information as well as intensity information on the recording medium.

Recording media respond only to the intensity of the incoming wavefront, not its phase. To encode the phase using normal media, holography depends on using a coherent reference beam to interfere with a scattered or diffracted wave, or "object wave", from the objects under observation. As a result of this interference, the intensity at the recording plane depends on both the phase and amplitude of the object wave.

The original image is reproduced simply by illuminating the hologram with coherent light using geometry similar to that used to record the hologram. If the wavelength used for reconstruction differs from the wavelength used for recording, the image will be enlarged or reduced.

The reconstruction process, while simple, is not amenable to rapid quantitative data collection. Quantitative analysis generally involves using three dimensional optical scanning of the reconstructed image. The ability to computationally re-

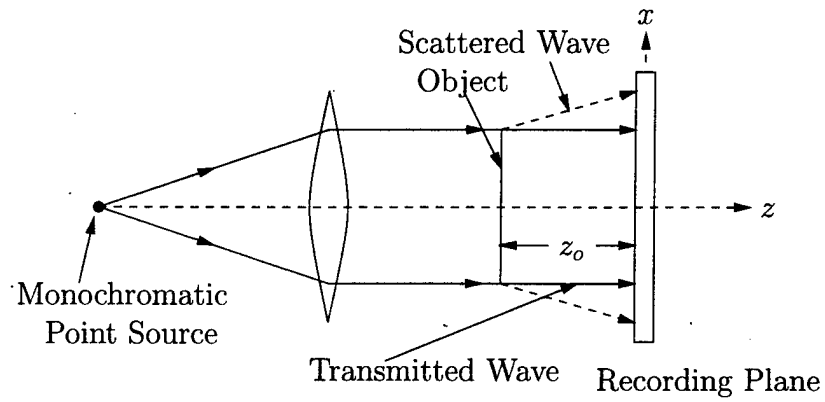


Figure 2-1: Basic in-line hologram geometry

construct the image would dramatically increase the utility of holographic measurement techniques.

2.2 In-line (Gabor) Holography

In-line holography, or Gabor Holography after its inventor, describes a holographic apparatus in which the reference beam is usually oriented perpendicular to the recording media and illuminates the objects from behind. It is not a useful technique for objects that are large with respect to the recording medium, but if one is interested primarily in the three dimensional position of small objects and not their three dimensional shape, in-line holography can be very effective. It is commonly used to size particulates and aerosols suspended in fluids, and it has been used for traditionally reconstructed HPIV[3].

Figure 2-1 shows the basic geometry and coordinate system for recording a plane in-line hologram. The object is illuminated by the reference source. The diffracted wave, or object wave, interferes with the wave from the reference source, or reference wave. This interference pattern is recorded by the medium in the recording plane and constitutes the hologram itself.

If the reference and object complex wave amplitudes measured at the recording plane are given by $\bar{R}(x,y)$ and $\bar{O}(x,y)$ respectively, the intensity at the recording

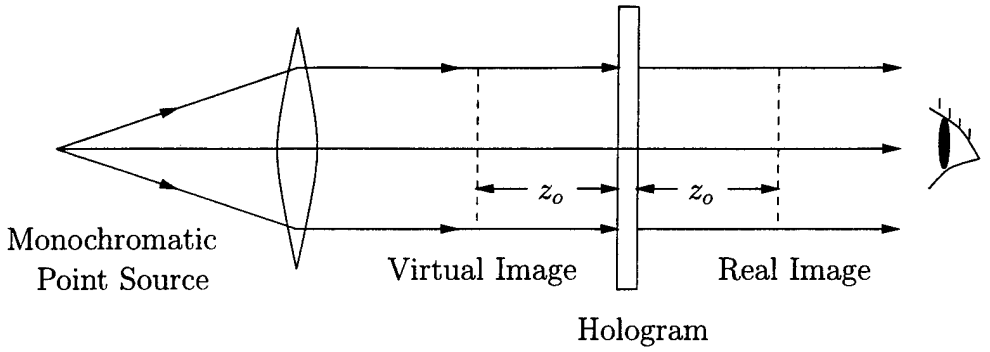


Figure 2-2: In-line hologram reconstruction geometry

plane is given by,

$$I(\bar{x},\bar{y}) = \left|\bar{R}(x,y) + \bar{O}(x,y)\right|^2 \tag{2.1}$$

Since the recording plane is perpendicular to the plane reference wave, the complex amplitude of the reference wave is a constant, r. Expanding (2.1) yields

$$\begin{aligned} I(\bar{x},\bar{y}) &= \left|r + \bar{O}(x,y)\right|^2 \\ &= r^2 + \left|\bar{O}(x,y)\right|^2 + r\bar{O}(x,y) + r\bar{O}^*(\bar{x},\bar{y}) \end{aligned} \tag{2.2}$$

The intensity is recorded in any acceptable manner, usually as a positive transparency on film. Most reasonable recording media respond nearly linearly to incident wave intensity. The transmittance, $t(x,y)$, of a positive transparency can be adequately described by

$$\begin{aligned} t(x,y) &= \tau_b - \beta T I(\bar{x},\bar{y}) \\ &= \tau_b - \beta T\left[r^2 + \left|\bar{O}(x,y)\right|^2 + r\bar{O}(x,y) + r\bar{O}^*(\bar{x},\bar{y})\right] \end{aligned} \tag{2.3}$$

where τ_b is a constant background transmittance, T is the exposure time, and β is a constant of the recording medium.

Figure 2-2 shows an in-line hologram being physically reconstructed. The wavefront transmitted by the hologram is simply the incident wave multiplied by

the transmittance of the hologram and can be written,

$$\begin{aligned}\psi(x,y) &= rt(x,y) \\ &= r\left(\tau_b + \beta T r^2\right) + \beta T r \left|\bar{O}(x,y)\right|^2 \\ &\quad + \beta T r^2 \bar{O}(x,y) + \beta T r^2 \bar{O}^*(x,y) \end{aligned} \tag{2.4}$$

The first term in (2.4) represents the uniformly attenuated reference wave. The second term is normally very small, since typically $\bar{O}(x,y) \ll r$. It is called the "halo" and under proper conditions can be neglected. The third term is identical to, within a multiplicative constant, with the original scattered wave. It forms an image z_o from the hologram on the side opposite the observer. This is the virtual image, which is what the observer perceives. The fourth term represents a wavefront similar to the originally scattered wavefront, but with opposite curvature. It converges to form a real image z_o from the hologram, between the hologram and the observer.[4]

The observer then sees the virtual image, with a superimposed, out-of-focus real image along with the diminished reference beam. The two images instead of one and are the biggest drawbacks to in-line holography.

2.3 Off-axis (Leith-Upatneiks) Holography

A simple way to separate the out-of-focus real image from the virtual image is to provide a separate path for the reference beam and the object wave. Figure 2-3 shows the basic geometry for recording an off-axis hologram.

Because the reference beam is no longer perpendicular to the recording plane, it cannot be simplified to a constant, rather, it can be expressed as,

$$\bar{R}(x,y) = r\exp(j2\pi\xi_r x) \tag{2.5}$$

where $\xi_r = (\sin\theta)/\lambda$, since only the phase of the reference beam varies across the recording plane.

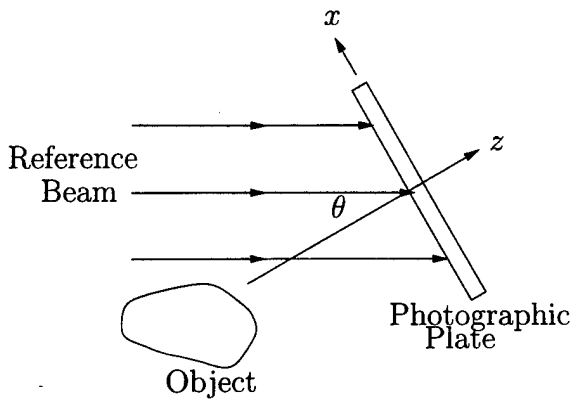


Figure 2-3: Off-axis hologram geometry

In a fashion similar to the derivation of (2.2) the intensity at the recording plane is found to be,

$$I(x,y) = r^2 \left|\bar{O}(x,y)\right|^2 + 2r \left|\bar{O}(x,y)\right| \cos\left[2\pi\xi_r x + \phi(x,y)\right] \tag{2.6}$$

Figure 2-4 shows the geometry for an off-axis hologram reconstruction. Assuming the recording medium is linear, the transmitted wave complex amplitude is found in a similar manner,

$$\begin{aligned} \psi(x,y) &= \tau_b r \exp(j2\pi\xi_r x) \\ &\quad +\beta T r \left|\bar{O}(x,y)\right|^2 \exp(j2\pi\xi_r x) \\ &\quad +\beta T r^2 \bar{O}(x,y) \\ &\quad +\beta T r^2 \bar{O}(x,y) \exp(j4\pi\xi_r x) \end{aligned} \tag{2.7}$$

Once again, the first term in (2.7) is simply the attenuated reference beam. The second term is a "halo" term, similar to the neglected term in (2.7). The third term is identical to the original scattered wave and forms a virtual image in the same position as the original object. The fourth term is, again, the real image, this time shifted from the optical axis by approximately twice the angle the reference beam makes with the recording plane. Hence the out of focus real image does not interfere with the vitual image.

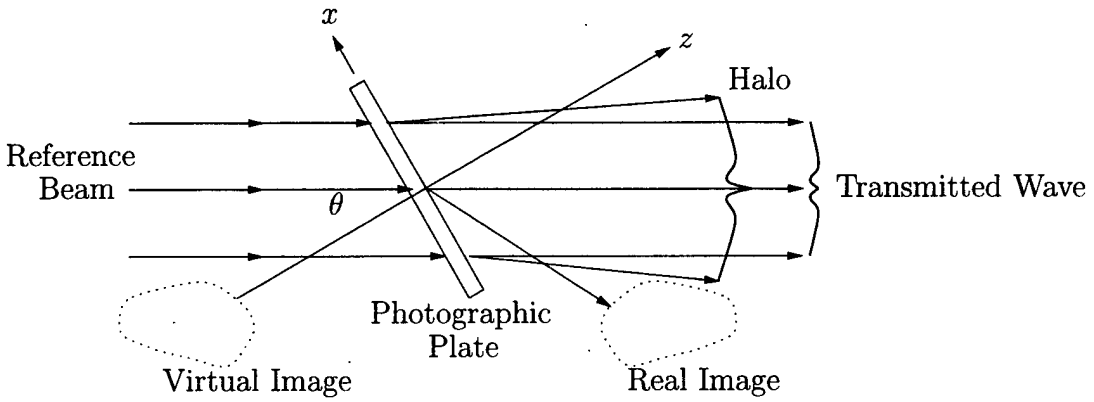


Figure 2-4: Off-axis hologram reconstruction geometry

At first blush it would seem off-axis holography is the answer to the virtual image problem for HPIV. If purely physical reconstruction is to be used, then off-axis holography is eminently practicable. The fatal flaw comes when considering digitizing the hologram for use in computational reconstruction. The reference beam now introduces a rapidly fluctuating intensity term that quadruples the resolution requirement of the recording medium and digitization system. For this reason off-axis holography will not be discussed further in this thesis.

2.4 Diffraction

Holography is fundamentally about diffraction. Goodman[5] provides a detailed derivation of the Huygens-Fresnel principle for a wave from constant z,

$$\psi(x_h, y_h) = \frac{z}{j\lambda} \iint_{\Sigma} \psi_z(x, y) \frac{\exp(jkr_{01})}{r_{01}^2} dx\, dy \qquad (2.8)$$

where, $r_{01} = \sqrt{z^2 + (x_h - x)^2 + (y_h - y)^2}$, and x_h and y_h are coordinates in the recording plane.

This expression, while physically meaningful, is not easily evaluated. Two fundamental simplifications are common depending on the geometry of the optical problem being analyzed. The approximations are the Fresnel and the Fraunhofer approximations, valid in the near and far fields, respectively.

2.4.1 The Fresnel Approximation

A simpler and more useful expression of the Huygens-Fresnel integral can be found by simplifying r_{01} using the paraxial approximation, $z^2 \gg x^2 + y^2$.

If $b < 1$, then,

$$\sqrt{1+b} = 1 + \frac{1}{2}b - \frac{1}{8}b^2 + \cdots \tag{2.9}$$

Using (2.9) with $b = \left(\frac{x_h - x}{z}\right)^2 + \left(\frac{y_h - y}{z}\right)^2$,

$$\begin{aligned} r_{01} &= \sqrt{z^2 + (x_h - x)^2 + (y_h - y)^2} \\ &= z\sqrt{1 + \left(\frac{x_h - x}{z}\right)^2 + \left(\frac{y_h - y}{z}\right)^2} \\ &\approx z\left[1 + \frac{1}{2}\left(\frac{x_h - x}{z}\right)^2 + \frac{1}{2}\left(\frac{y_h - y}{z}\right)^2\right] \end{aligned} \tag{2.10}$$

Substitution of (2.10) into (2.8) yields little simplification unless carefully chosen terms are dropped. For the r_{01}^2 term in the denominator of (2.8) it is considered acceptable to drop all terms but z. However, small changes in the r_{01} occurring in the numerator can cause significant changes in the value of the exponential and cannot be omitted. These large changes are generated in part by the multiplication of r_{01} by k, usually a very large quantity, and the sensitivity of the exponential to phase changes in its argument. The resulting simplification yields the Fresnel-Kirchoff Diffraction Integral:

$$\psi(x_h, y_h) = \frac{e^{jkz}}{j\lambda z}\iint_{-\infty}^{\infty} U(x,y)\exp\left\{j\frac{k}{2z}\left[(x_h - x)^2 + (y_h - y)^2\right]\right\}dx\,dy \tag{2.11}$$

$$= \frac{e^{jkz}}{j\lambda z}e^{j\frac{k}{2z}(x_h^2 + y_h^2)}\iint_{-\infty}^{\infty} U(x,y)e^{j\frac{k}{2z}(x^2+y^2)}e^{-j\frac{2\pi}{\lambda z}(x_h x + y_h y)}dx\,dy \tag{2.12}$$

Where (2.12) differs from (2.11) only through factoring the term $\exp\left[\frac{jk}{2z}(x_h^2 + y_h^2)\right]$ outside the integral. This approximation holds when the observer is in the *near field*, or Fresnel zone, of the diffracting aperture. Typical HPIV geometry will

place the recording medium in the far field of the individual particles but in the near field of the ensemble.

2.4.2 The Fraunhofer Approximation

Additional simplifications to the Fresnel Diffraction Integral can be made if the system meets the Fraunhofer approximation,

$$z \gg \frac{k(x^2+y^2)_{\max}}{2} \tag{2.13}$$

Simplifying (2.12) yields,

$$\psi(x_h, y_h) = \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x_h^2+y_h^2)} \iint_{-\infty}^{\infty} U(x,y) e^{-j\frac{2\pi}{\lambda z}(x_h x + y_h y)} dx\, dy \tag{2.14}$$

which is valid in the *far field*. The integral is recognizable as a two dimensional Fourier transform.

2.4.3 Convolution interpretation of Fresnel-Kirchoff Integral

The Fresnel-Kirchoff Integral,

$$\psi(x_h, y_h) = \frac{e^{jkz}}{j\lambda z} \iint_{-\infty}^{\infty} U(x,y) \exp\left\{ j\frac{k}{2z}\left[(x_h - x)^2 + (y_h - y)^2\right]\right\} dx\, dy \tag{2.15}$$

can easily be viewed as a convolution with kernel,

$$h_z(x,y) \equiv \frac{e^{jkz}}{j\lambda z} \exp\left[j\frac{k}{2z}\left(x^2+y^2\right)\right] \tag{2.16}$$

yielding

$$\psi_z(x_h, y_h) = \iint_{-\infty}^{\infty} U(x,y) h_z(x_h - x, y_h - y) dx\, dy = U(x_h, y_h) \otimes h_z(x,y) \tag{2.17}$$

This expression determines the field diffracted from a planar object, or an ensemble of co-planar planar objects. If a large number of very small non-coplanar objects are in the field of view, (2.17) can be extended:

$$\psi(x_h, y_h) = \sum_n U_n(x_h, y_h) \otimes h_{z_n}(x_h, y_h) \tag{2.18}$$

where the sum is taken over n planes within the volume. Each object plane, $U_n(x, y)$, contains at least one diffracting object and is located z_n from the recording plane. The planes need not be evenly spaced.

2.5 Laser Characteristics

2.5.1 Fundamentals

At the atomic level, light is always generated by an atom transitioning from one energy state, E_0, to a lower energy state, E_1. When this transition occurs the atom emits a photon with frequency $\nu = (E_0 - E_1)/h$, where h is Planck's constant. This emission is a probabilistic occurrence and may occur spontaneously provided the atom has sufficient internal energy. This is spontaneous emission.

A second form of emission, first suggested by Einstein in 1917, is called stimulated emission. In the case of stimulated emission, a photon collides with an excited atom. The incident photon must have exactly the same energy as the transition the atom will undergo. When this condition is met, the atom will make the transition, emitting a second photon. The emitted photon is in phase with the incident photon and travels in the same direction.

Stimulated emission is fundamental to laser operation. A material is induced to have a majority of its atoms in an excited state (called a "population inversion"). Since the majority of the atoms in the material are excited to the same state, they have identical transition energies. Initial spontaneous emission by a small number of the atoms will stimulate emission in others. The stimulated photons will in turn stimulate further emission in a run-away chain reaction. This in itself will not cause light amplification, but if the material is mounted within an optical

cavity (two parallel reflectors), the stimulated emission will preferentially orient itself along the axis of the cavity. Radiation not parallel to the axis will be lost to the surroundings and not contribute to the electromagnetic wave building up in the cavity. Since the stimulated emission is always in phase, the radiation traveling along the cavity axis constructively reinforces and the amplitude rapidly increases forming a longitudinal vibration mode within the cavity. If an end of the cavity is only partially reflective, then some of the coherent amplified signal will be released, this is the usable laser beam.

2.5.2 Establishing the Population Inversion

The process of establishing the population inversion is called 'pumping'. Pumping can be accomplished using flash lamps, usual in the case of ruby lasers, or electrical discharge, as in the case of most gas lasers. Photochemical reactions can also stimulate the population inversion but that technique is very rare.[6]

The cavity material characteristics fundamentally determine what type of pumping will be appropriate. Continuous beam lasers must obviously use a sustainable pumping technique, such as tungsten filament incandescent lamps, or the aforementioned electrical discharge.

The length of the pulse in non-switched pulse lasers is directly related to the length of the pulse length of the pump. Flash lamp pumped ruby lasers typically provide pulses on the order of several milliseconds long. The length of the pulse also varies from pulse to pulse as the condition of the flash lamp changes.

The minimum time between lasers pulses in non-switched lasers is directly related to the requirements of the pump. Flash lamps typically must cool immediately after use. The amount of cooling is directly related to the power output of the lamp, and can be on the order of minutes. This limits the pulse repetition frequency of the unswitched laser.

Switching can reduce pulse lengths to femtoseconds (1×10^{-15} seconds) and pulse intervals similarly. Switching is discussed in some detail in Section 2.5.5.

2.5.3 Energy Levels

When common laser materials are pumped, the resulting atomic population usually has many more than one possible transition state available. The likelihood that an atom undergoes a particular transition depends on the quantum mechanics of the particular material. In any event, when the population inversion starts to break down there can be a large number of longitudinal oscillation modes, each producing a separate wavelength in the cavity. Typically, a small number of closely spaced wavelengths will predominate, but the resultant beam is not strictly monochromatic without special modifications to the laser (see Section 2.5.7). [6]

2.5.4 Coherence Length

The preceding discussion leaves the impression that all light generated in the laser cavity is coherent. This is not the case. Not all stimulated photons are descended from the same spontaneously emitted photon, and the Heisenberg uncertainty principle places a lower limit on the precision with which the wavelengths are reproduced. So even a laser oscillating in only a single longitudinal mode has some degree of incoherence in its output.

One measure of the quality of a laser's output is termed "coherence length". This length is the maximum beam path length within which the beam can coherently interfere with itself. The practical ramification of this is that the holographic system must have all path lengths from laser output to recording planes less than the coherence length of the laser. [7]

2.5.5 Switching

It is clear that the amplification properties of the laser cavity are critically dependent on the alignment and quality of the end mirrors. Misalignment or surface imperfections can destroy the amplification effect and turn the laser into an expensive flash bulb.

This sensitivity can be turned to the laser users advantage. If the mirror properties can be accurately controlled then the amplification process can be controlled

independently of the pumping process.

Borrowing a term from electrical engineering, the ability of the cavity to resonate is called it's 'Q'. Systems that switch the amplification process on and off by controlling the resonance qualities of the cavity are called 'Q-switched' or 'Q-spoiled' lasers.

Q-switching has a subtle benefit. The population inversion does not occur instantaneously at the beginning of the pumping sequence. If the cavity cannot oscillate due to low Q, the population inversion can build to a much higher level than required for lasing. If the Q is suddenly increased, allowing the cavity to oscillate, the peak energy in the pulse can be much higher than if the cavity was allowed to oscillate continuously through the pumping sequence. So the power available in a Q-switched pulse can be much higher than in an unswitched pulse.[6]

An additional benefit of Q-switching is the ability to produce multiple pulses out of a single pumping process. This allows accurate control of continuously pumped lasers, and multiple rapid pulses out of short pulse pumped lasers such as ruby lasers.

2.5.5.1 Electrooptical Q-Switching

Several physical phenomena allow the electrical modulation of an optical signal. The most commonly used in Q-switching lasers is embodied in the Pockels Cell.

Pockels Cells are formed from a crystal, typically potassium dihydrogen phosphate (KDP). KDP has two distinct optical axes. With an electrical voltage applied across the crystal the index of refraction on one axis is different than the other. An incident polarized wave is split into two portions, one portion traveling more slowly than the other. The two waves are recombined upon exiting the cell, resulting in a wave that is elliptically, circularly, or linearly polarized depending on the applied voltage.

The two voltages of interest for Pockels Cell Q-switching are termed the quarter-wavelength ($\lambda/4$) voltage, and the half-wavelength ($\lambda/2$) voltage. The quarter-wavelength voltage will result in a circularly polarized beam at the output of the cell. The half-wavelength voltage will result in a wave linearly polarized 90° to

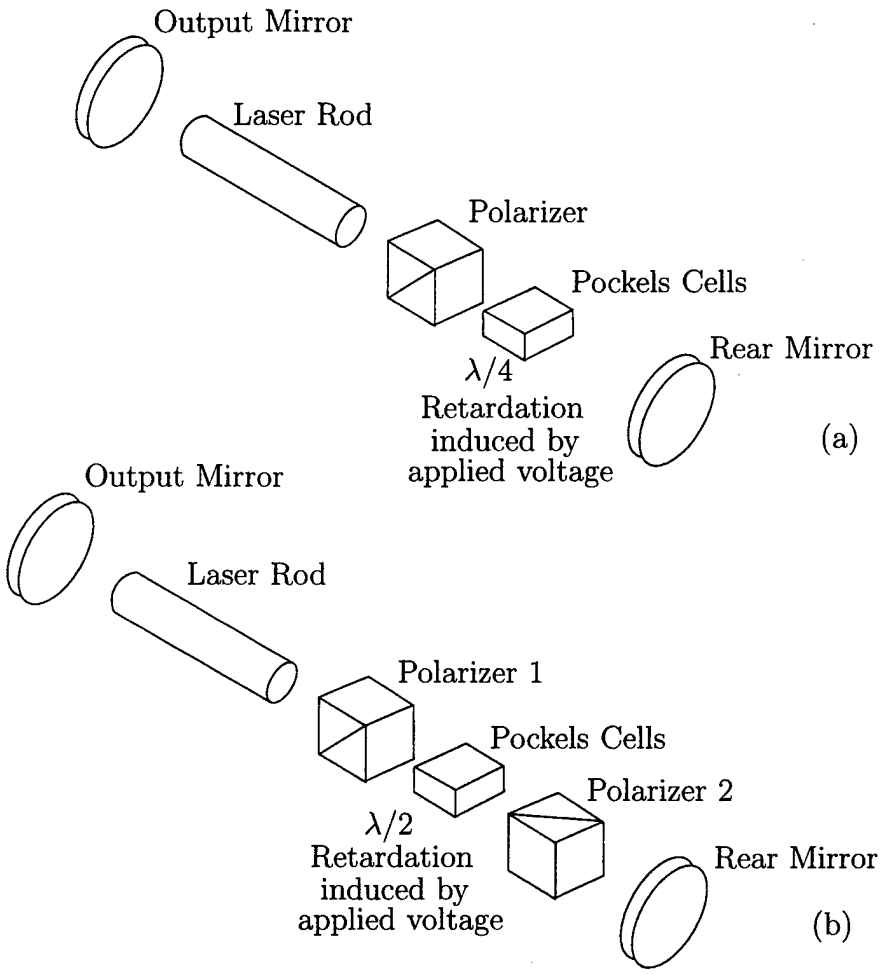


Figure 2-5: Pockels Cell Q-switch at (a) quarter-wave and (b) half wave retardation voltage.

the input wave.

The quarter-wavelength retardation system shown in Figure 2-5(a) is a 'pulse off' Q-switch. During the initial phase of pumping, the $\lambda/4$ voltage is applied to the cell. The output of the Pockels cell is circularly polarized, after reflection the circularly polarized beam reenters the Pockels cell and emerges linearly polarized 90° to the input beam. The polarizer then prevents the beam from reentering the laser rod, spoiling the amplification. When the voltage is removed, the Q is restored, and oscillation can start.

The half-wavelength retardation system shown in Figure 2-5(b) is a 'pulse on' Q-switch. With no voltage applied the Pockels cell does not alter the polariza-

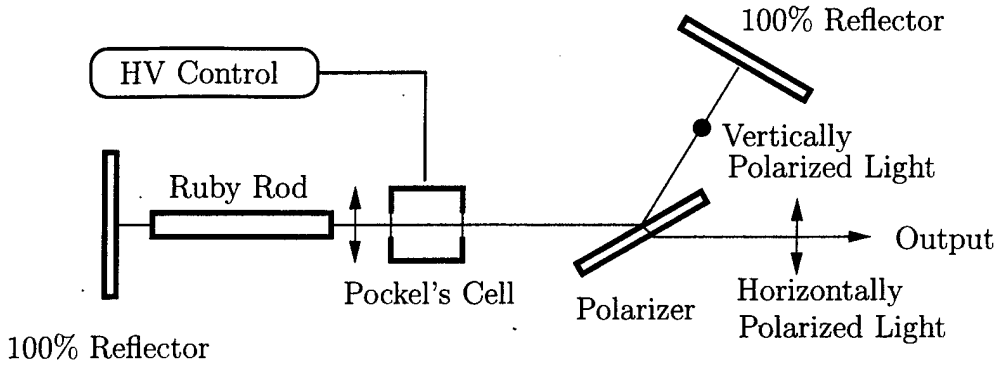


Figure 2-6: Schematic Cavity Dump

tion of the transmitted beam, and the crossed polarizers prevent oscillation from occurring. When switched on the Pockels cell shifts the polarization 90° allowing the beam to pass uninterrupted through the two polarizers.

The half-wave retardation voltage for KDP is 7500V. KDP is also water soluble. Normally the crystal is hermetically sealed to prevent exposure to atmospheric moisture, but hydrophobic materials and extremely high voltages merit caution when used in a hydrodynamics experiment.

2.5.5.2 Mechanical Q-Switches

There are two common methods of mechanically Q-switching a laser. They both rely on mechanically rotating a reflective element within the laser cavity. The most obvious is to rotate the mirror such that it is only periodically parallel to the fixed reflector at the opposite end of the cavity. This system requires extremely critical alignment of the shaft upon which the mirror rotates. The alternative to rotating a mirror is to use a roof prism, which alleviates the alignment requirement.

Mechanical switching is not as flexible as electrooptical switching, and is probably not practicably capable of implementing the three pulse technique described in Section 3.1.

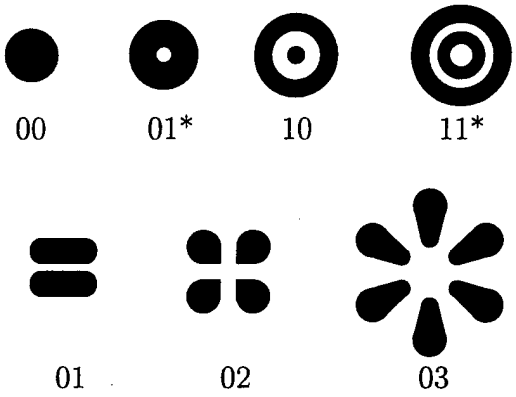


Figure 2-7: Cylindrical Transverse Electromagnetic Modes

2.5.5.3 Cavity Dumping

Extremely short pulses can be generated using a "cavity dump". Cavity dumps allows the use of two 100% reflectivity end mirrors. When the laser power is maximized, the cavity dump redirects the beam outside of the cavity, rapidly dumping the entire optical energy of the cavity into the output beam. The pulse length is reduced to the round trip transit time within the cavity. Cavity dumps can be used to switch intermittently and continuously pumped lasers.

A schematic of a cavity dumped ruby laser is shown in Figure 2-6. Initially, no voltage is applied to the Pockels cell. When the flash lamp is fired the horizontally polarized wave is passed through the polarizer and not allowed to regenerate. Vertically polarized light is kept within the cavity and builds up the vibration mode within the cavity. When the energy stored in the Ruby has peaked, the Pockels cell is switched to its half-wave retardation voltage. The vertically polarized light is rotated to horizontally polarized light by the Pockels cell and passes through the polarizer and out of the cavity. The combination of the polarizer, off-axis mirror and the Pockels cell essentially forms a voltage controlled mirror, whose reflectivity can be change from 100% to 0% and back to 100% during the cycle.[6, 8]

2.5.6 Transverse Electromagnetic Modes

Laser cavities do not only oscillate longitudinally. There are also transverse modes of vibration which are very similar to the vibration modes of a drum head. Laser specifications typically include a description of which transverse modes are present in the output. The Transverse Electromagnetic Modes (TEM) are described using the symbol TEM_{mnq}. The first two indices describe the number of transverse modes, while the third (often omitted in specification) describes the number of longitudinal modes. Figure 2-7 shows cylindrical transverse modes, along with the indices describing them. Rectangular modes are also possible. It is not difficult to find TEM_{00} ruby lasers suitable for HPIV use.

2.5.7 Output Frequency Control

It is possible to dramatically reduce the number of longitudinal modes present in the output. The simplest method is to use a Fabry-Perot etalon. This is simply a very small optical cavity that resonates at precisely the frequency desired in the laser cavity. The etalon effectively spoils the cavity Q for other frequencies, preventing them from building up.

The simplest form of etalon is simply a thin glass plate. Internal reflections within the plate selection interfere with the wave to produce a sharp resonance peak. The effective thickness of the plate can be changed by rotating the etalon about an axis perpendicular to the cavity axis.

Other etalon designs are possible. Alternate designs include wedge shaped plates that can be tuned by translating the plate, as well as designs that form an optical cavity using an air gap between two plates. The latter design can sometimes be adjusted by varying the gas pressure or temperature within the cavity.[6, 8]

2.5.8 Laser Construction

Commercial lasers for field use are often manufactured in such a way as to limit the ability of the user to alter the laser. This increases laser reliability. Commercial

lasers for laboratory use are a different matter. As discussed above there are many additional components that can be added to the laser cavity to customize and control the laser output. Laboratory lasers typically mount all components, including the mirrors and laser rod, on a long rail with additional space for adding polarizers, etalons, Q-switches and cavity dumps.

Chapter 3

HPIV System Design

3.1 Timing

The requirements to unambiguously determine speeds less than 10m/s inside a volume 5 cm on a side impose severe restrictions:

1. Exposure times must be short enough to 'stop' the motion. In a photograph the motion allowed during the exposure is related to the feature size of the object. During the photographic exposure the smallest features of an object should not move more than approximately 1/10th their size or the image will be unacceptably blurred. Likewise an acceptable amount of particle motion during holographic exposure has been found to be approximately 1/10th of a particle diameter[9].

2. The interval between exposures must be short enough to maintain the identity of the particles, so tracking and velocity determination is possible. This restricts the mean free path length of a particle between exposures to less than the distance between particles. There is an additional problem of determining which particle image is 'oldest', so that direction of travel can be determined.

3. The interval between exposures must be short enough to ensure a majority of the particles captured in the first exposure are within the covered volume during the following exposures.

For a 10μm particle, these requirements become:

1. A 1μs exposure results in 1/10th diameter of travel during the exposure.
2. A 10μs exposure interval results in 10 particle diameters of travel between exposures at 10m/s.

The exposure interval also places an upper limit on the concentration of seed particles. Assuring the particle mean free path between exposures is less than the average distance between particles requires that

$$\rho_p \leq \frac{1}{\frac{4}{3}\pi(100\mu\text{m})^3} \approx 2 \times 10^5 \frac{\text{particles}}{\text{cm}^3} = 4 \times 10^6 \frac{\text{particles}}{\text{in}^3}$$

Where ρ_p is the seed particle density, and the mean free path of the particles is ten particle diameters, 100μm.

The direction problem is subtler. Keeping the average inter-particle distance greater than the mean free path length between exposures significantly simplifies the task of identifying pairs of particle images as being the same particle. Unfortunately, even when the two images of a given particle have been identified, there is a 180° ambiguity in the velocity of the particle. A simple solution to this problem is to use three pulses rather than two, with differing time intervals between pairs of pulses. For example, the second 1μs pulse is sent 5μs after the first, then the third is sent 10μs after the second. The pulse intervals are sufficiently short to permit particle identification, while the difference in intervals allows determination of the time order of the images. An additional advantage of the triple exposure technique could be estimation of fluid acceleration. This technique assumes the flow speed will not change appreciably over the distances traveled by particle during the exposure sequence. If the particle were to radically decelerate to less than half of the speed it traveled during the first set of exposures, it would be possible to confuse the time order of the exposures.

3.2 Recording and Digitization

3.2.1 CCD Requirements

The ultimate goal for computationally reconstructed HPIV is a completely digital process for determining the flow field throughout the volume. The most likely candidate for the 'digital recording plane' is a Charge Coupled Device (CCD) Array. Unfortunately, the resolution of CCD arrays is significantly less than holographic film.

3.2.1.1 Characteristics of CCD arrays

A CCD Array is a semiconductor architecture that transfers charge through storage areas. The CCD array is essentially an array of charge "buckets". Photons are captured by the buckets and converted to charge, the buckets only perform this conversion during a controllable integration period. More photons, or higher energy photons, hitting the array during the integration period result in a higher element charge. After integration the CCD Array support electronics scan through each bucket and report the charge level in each bucket. This charge is proportional to the intensity of the image focused on the array.[10]

For holographic applications the fundamental characteristic of CCD Arrays that differentiates them from film is the limited number of recording elements. High quality photographic film can have grains smaller than 1μm over an area 12-20 centimeters square, or better than 120,000×120,000 recording elements. The best CCD arrays currently commercially available have element sizes on the order of 10μm, but the total array size is only about 5 centimeters on a side, yielding an array of approximately 5000×5000 elements. This is the root of the limitation in holographic applications.

3.2.1.2 CCD Resolution Requirements

3.2.1.2.1 Diffraction patterns of circular objects Parrent and Thompson analytically derived the diffraction pattern for an opaque circular object in a

coherent field. [11] For a circular particle, the field and intensity, respectively, are:

$$\begin{aligned}\psi(r) &= \cos kz - \frac{ka^2}{r}\sin\left(kz+\frac{kr^2}{2z}\right)\Lambda_1\left(\frac{kar}{z}\right) \\ &\quad + i\left[\sin kz + \frac{ka^2}{r}\cos\left(kz+\frac{kr^2}{2z}\right)\Lambda_1\left(\frac{kar}{z}\right)\right] \\ I(r) = |\psi(r)|^2 &= 1 - \frac{2ka^2}{r}\sin\frac{kr^2}{2z}\Lambda_1\left(\frac{kar}{z}\right) + \frac{k^2a^4}{r^2}\left[\Lambda_1\left(\frac{kar}{z}\right)\right]^2 \quad (3.1)\end{aligned}$$

Where, r is measured from the center of the object projected into the recording plane, and $\Lambda(\alpha) = J_1(\alpha)/\alpha$, a Bessel function of the first kind divided by its argument.

An illustrative example is plotted in Figures 3-1 and 3-2. The parameters more characteristic of HPIV systems result in plots that do not acceptably show the salient features of the intensity distribution.

3.2.1.2.2 Recording the Diffraction Pattern Equation (3.1) describes the diffraction pattern recorded for a single particle with circular cross section. All previous discussion has assumed this diffraction pattern is perfectly recorded. In reality the recording medium has limited bandwidth (*i. e.* a grain), and limited physical extent.

Figure 3-1 shows a high frequency sine function modulated by a lower frequency $\Lambda(\alpha)$ envelope function. The envelope function is dependent on both particle diameter and particle distance from the recording plane. The high frequency sine function is dependent only on particle distance. Accurately recording the high frequency component is the most critical factor in obtaining good depth discrimination.

Vikram[3] recommends recording the first three side lobes formed by the bessel envelope function. The spacing of the fine fringes, Δx, in the third lobe can be

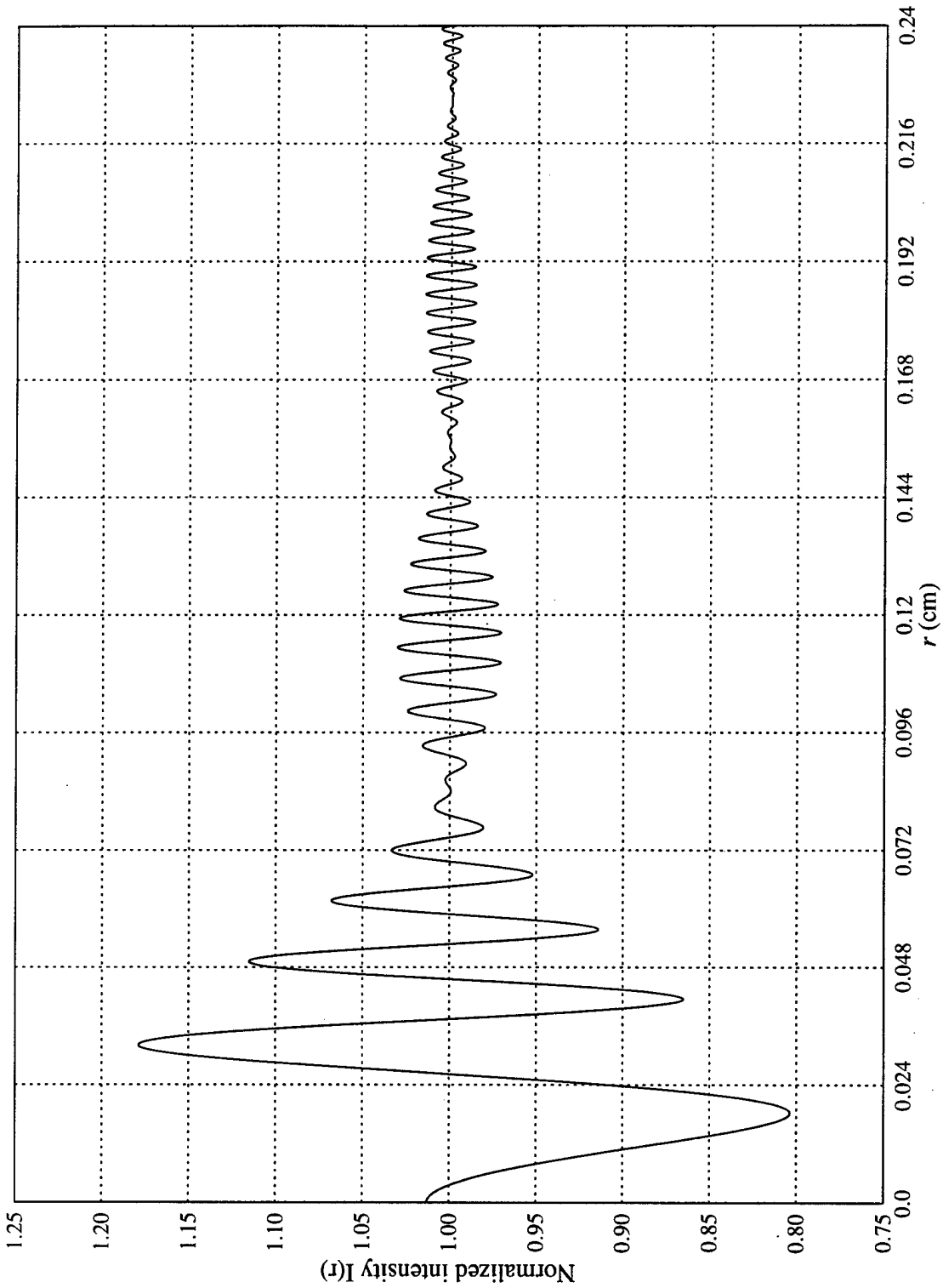


Figure 3-1: Diffraction intensity of a circular object with $a = 50\mu$m, $\lambda = 0.6943\mu$m, and $r = 10$cm

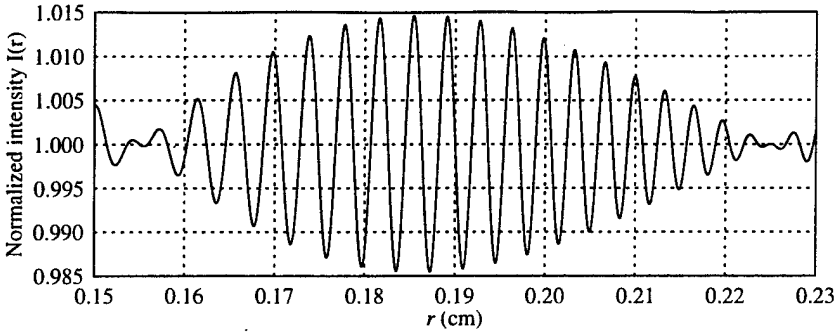


Figure 3-2: Enlarged plot of the third side lobe in Figure 3-1

found using the argument of the sine term in (3.1), $kr^2/2z$:

$$\begin{aligned} \frac{kx_2^2}{2z} - \frac{kx_1^2}{2z} &= 2\pi \\ x_2^2 - x_1^2 &\approx 2x\Delta x = 2\lambda z \\ \Delta x &= \frac{\lambda z}{x} \end{aligned} \tag{3.2}$$

Equation (3.2) should be evaluated in the region containing the third side lobe. The mid-radius of the third side lobe, $\bar{r}_3$ is:

$$\begin{aligned} J_1(\alpha_n)/\alpha_n &= 0 \quad (n = 1, 2, ..., \infty) \\ &\therefore \quad \alpha_n = 3.832,\ 7.06,\ 10.173,\ 13.323,\ 16.470... \\ \alpha_n &= \frac{kar_n}{z} \\ \bar{r}_n &= \frac{r_n + r_{n-1}}{2} = \frac{z}{2ka}(\alpha_n + \alpha_{n-1}) \\ \bar{r}_3 &= \frac{z}{2ka}(\alpha_3 + \alpha_2) = 17.189\frac{z}{2ka} \end{aligned} \tag{3.3}$$

Equations (3.3) and (3.2) give $\bar{r}_3 = 0.189$cm and $\Delta x = 3.65 \times 10^{-3}$cm for the parameters plotted in Figures 3-1 and 3-2. Using $z = 5$cm and $a = 10\mu$m, with $\lambda = 690$nm, more reasonable for HPIV purposes, $\bar{r}_3 = 0.472$cm and $\Delta x = 7.31 \times 10^{-4}$cm.

The parameter $\bar{r}_3$ dictates the minimum size of the hologram, with no magnification, and is easily met. The parameter Δx dictates the minimum grain of

the hologram recording medium. Since enlargement or reduction of the hologram during recording is easily accomplished using lenses, the absolute size of the fringe spacing is not important. Rather, the important quantity is the ratio the fringe spacing to the total size of the system, so that the fine fringes produced by particles at opposite extremes of the volume can be resolved. The sampling theorem suggests that the recording medium should be able to sample at twice the highest spatial frequency required, or 1/2 the fringe spacing. To achieve the desired accuracy, a CCD array would have to have

$$\frac{2 \cdot 0.05\text{cm}}{7.31 \times 10^{-4}\text{cm}} \approx 13680 \text{ pixels square.}$$

This resolution will adequately record the particle diffraction patterns. Unfortunately no CCD arrays are commercially available that reach this resolution. The best currently available are in the range of 5000 pixels square.

3.2.2 Exposure Requirements

While the performance characteristics of CCD arrays are well known it is difficult to predict the amount of power incident on the array from the suspended particles.

The resolution analysis was performed assuming the finest necessary fringes were detected. The incident power depends heavily on the power output of the laser, the absorption of the fluid, and the reflectivity of the particles, and the viewing window material.

The exposure problem is different in character from the resolution problem. The resolution problem has to wait for better CCD technology, or lower the accuracy requirements for HPIV significantly. The exposure problem can be solved many ways, the simplest (but not necessarily the least expensive) way is to increase the laser power.

Chapter 4

Simulation

4.1 Motivation

A primary factor preventing HPIV from becoming an easily and widely used technique is the data reduction effort. As previously mentioned, physical reconstruction techniques are simple, but require costly equipment, and inordinate amounts of time. If computational reconstruction methods can be perfected, HPIV could be become more practical.

There are several possibilities for computational reconstruction algorithms, one of which (Onural's method) is presented in Section 5.1. The methods must be evaluated with realistic input in order to gauge their effectiveness. Obtaining realistic input is difficult. There have been experiments done using holographic techniques to study small ocean biological specimens, but obtaining the actual holograms produced is difficult. An additional difficulty is that you don't know precisely what was contained in the original image so calibrating the reconstruction algorithm is nearly impossible.

4.2 Hologram Synthesis

There are several methods applicable to synthesizing holograms similar to those expected from HPIV. The most straightforward methods calculate the individual contribution from each particle to each pixel on the simulated recording plane.

These methods are the most accurate and flexible but suffer from excessive computational requirements when large numbers of particles are being simulated.

One way around this problem is hinted at by Equation (2.18):

$$\psi(x,y) = \sum_n U_n(x,y) \otimes h_{z_n}(x,y)$$

We can subdivide the region being simulated into a large number of thin slabs. Within each slab there can be a large number of particles, but each will be treated as if it had the same z coordinate. We then calculate the contribution of each slab to the total hologram rather than the contribution from each particle. With a large number of particles we can achieve significant performance increases with little degradation of quality.

Evaluating (2.18) is quite efficient. Using the convolution theorem:

$$\begin{aligned} \psi(x,y) &= \sum_n U_n(x,y) \otimes h_{z_n}(x,y) \\ &= \sum_n \mathcal{F}^{-1}\left[\mathcal{F}\left[U_n(x,y)\right] \cdot \mathcal{F}\left[h_{z_n}(x,y)\right]\right] \\ &= \mathcal{F}^{-1}\left[\sum_n \mathcal{U}_n(\omega_x,\omega_y) \cdot \mathcal{H}_{z_n}(\omega_x,\omega_y)\right] \end{aligned} \tag{4.1}$$

Where $\mathcal{U}_{z_n}(\omega_x,\omega_y) = \mathcal{F}\left[U(x,y)\right]$ and $\mathcal{H}_{z_n}(\omega_x,\omega_y) = \mathcal{F}\left[h_{z_n}(x,y)\right]$. But $\mathcal{H}_{z_n}(\omega_x,\omega_y)$ can be analytically evaluated [12],

$$\mathcal{H}_{z_n}(\omega_x,\omega_y) = \mathcal{F}\left[h_{z_n}\right] = \exp\left[-j\frac{\lambda z_n}{4\pi}\left(\omega_x^2+\omega_y^2\right)\right] \tag{4.2}$$

yielding,

$$\psi(x,y) = \mathcal{F}^{-1}\left[\sum_n \mathcal{U}_n(\omega_x,\omega_y) \cdot \exp\left[-j\frac{\lambda z_n}{4\pi}\left(\omega_x^2+\omega_y^2\right)\right]\right] \tag{4.3}$$

The algorithm to evaluate (4.3) is shown schematically in 4-1, and computer code to perform the algorithm is given in Sections A.1.1 and A.1.2.

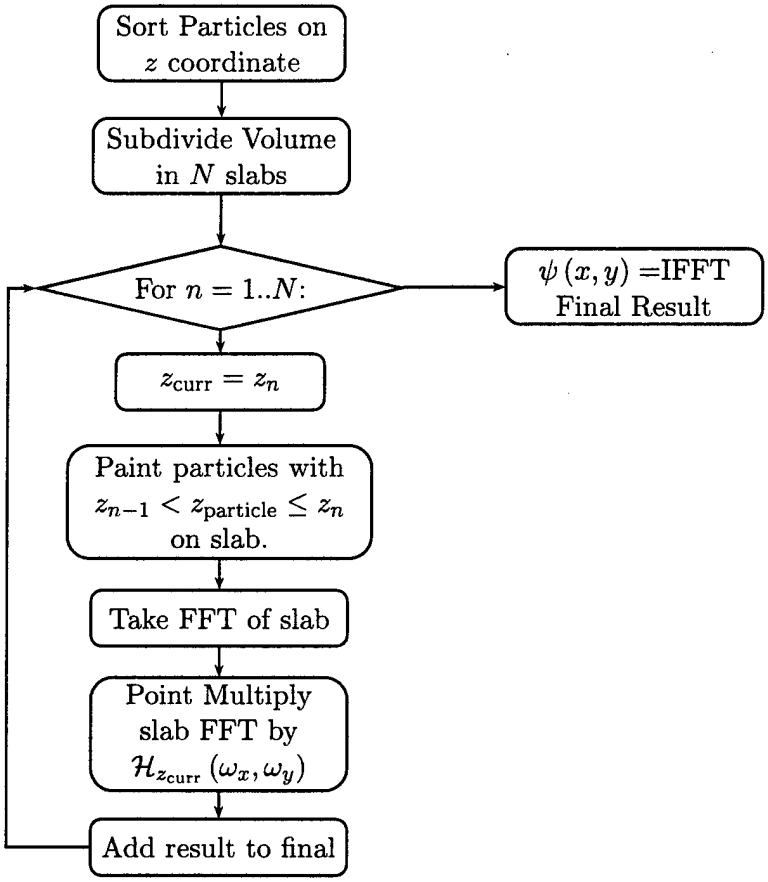


Figure 4-1: Hologram synthesis algorithm

4.3 HoloSynth Simulator

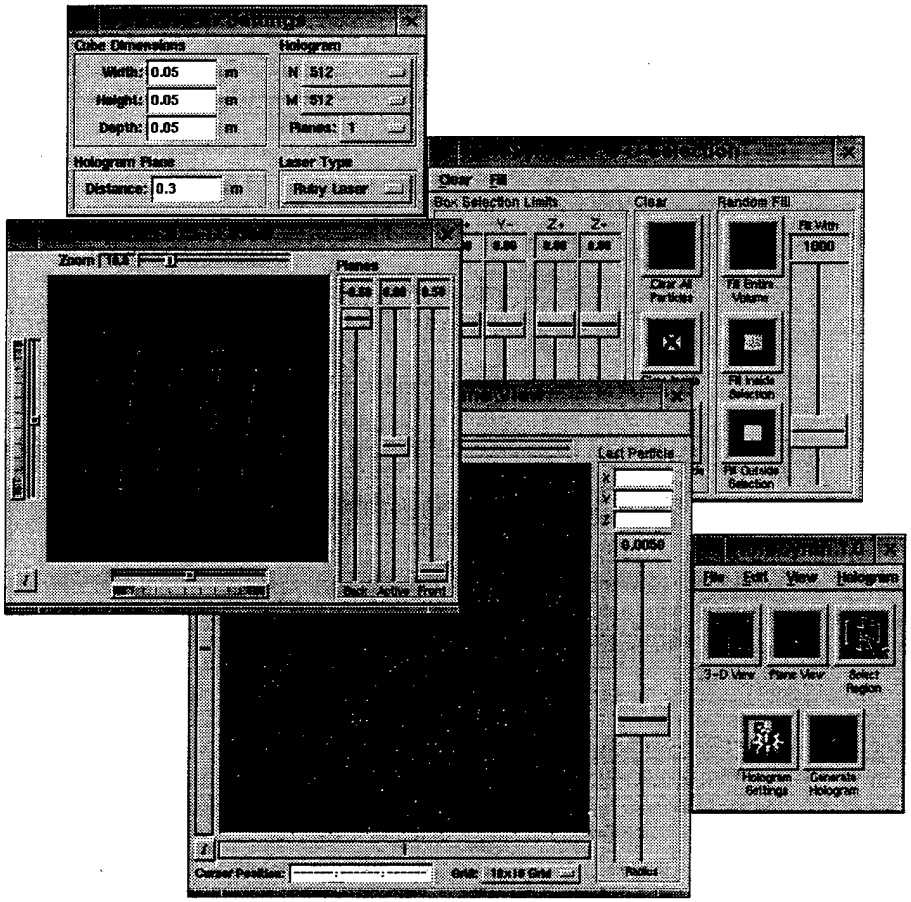


Figure 4-2: HoloSynth user interface

4.3.1 Introduction

HoloSynth assists in synthesizing holograms similar to those obtained from Holographic Particle Image Velocimetry. It provides the user with a simple way to place any number of particles into a cubic volume in preparation for generating a synthetic hologram. The user can manually place particles anywhere in the 3-dimensional region as well as fill arbitrary rectangular solid regions with randomly distributed particles.

HoloSynth uses the method for synthesizing holograms described in Section

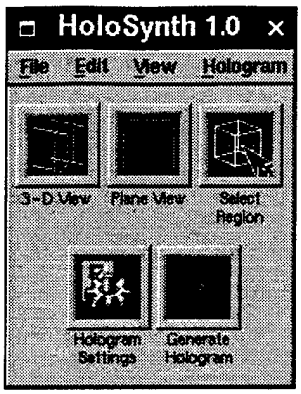


Figure 4-3: HoloSynth main controls

4.2. In addition it prepares an input file for an alternate hologram synthesizer so that different synthesis algorithms may be easily compared, without modifying the HoloSynth source code. The output file is also useful in analyzing the results of reconstruction algorithms.

HoloSynth was developed under Red Hat®Linux version 5.1 and tested under Microsoft Windows NT®.

4.3.2 The Interface

4.3.2.1 The Main Controls

Figure 4-3 shows the main controls for HoloSynth.

Each button will open up a new window that controls a particular aspect of HoloSynth. The windows are:

1. 3-D View of the current particle configuration.

2. Plane View, which allows manual placement of particles on an arbitrary plane.

3. Select Region, which allows selection of an arbitrary rectangular solid region within the view space. From this control panel you can:

   - Clear all or any part of the volume.

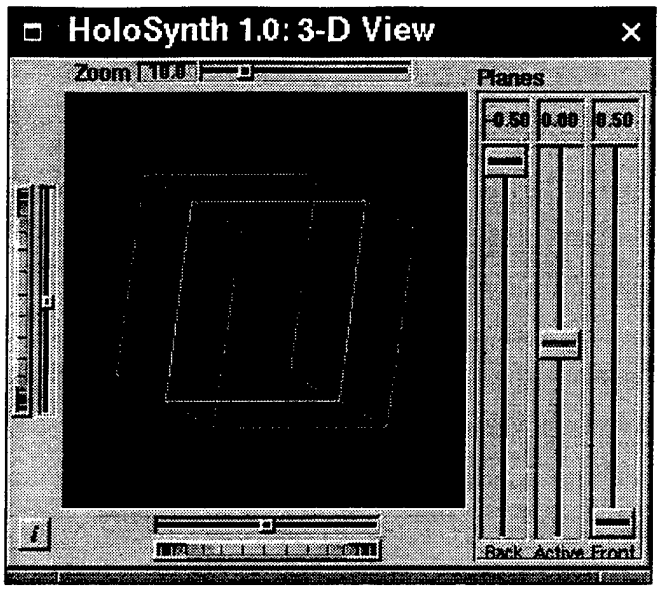


Figure 4-4: 3-D View window

- Fill all or any part of the volume with randomly placed particles.

4. The Synthesizer Settings window which controls the physical characteristics of the simulation used for the hologram synthesis.

Through the file menu the user can open and save particle fields, and exit HoloSynth.

4.3.2.2 The 3-D View

The 3-D View, shown in Figure 4-4, provides a three dimensional view of the current particle configuration. There are three basic functions controlled by this window: the magnification (zoom) of the view, the angle and translation of the view, and the position of the active and two clipping planes. No direct editing takes place in this window, it is merely a visualization aid.

The viewing "camera position" is controlled by the rollers and sliders to the left and below the viewing window. The slider above the viewing window controlling the magnification of the camera.

The various view control planes are controlled using the slide bars to the right of the 3-D view window. They appear as colored squares drawn around the periphery

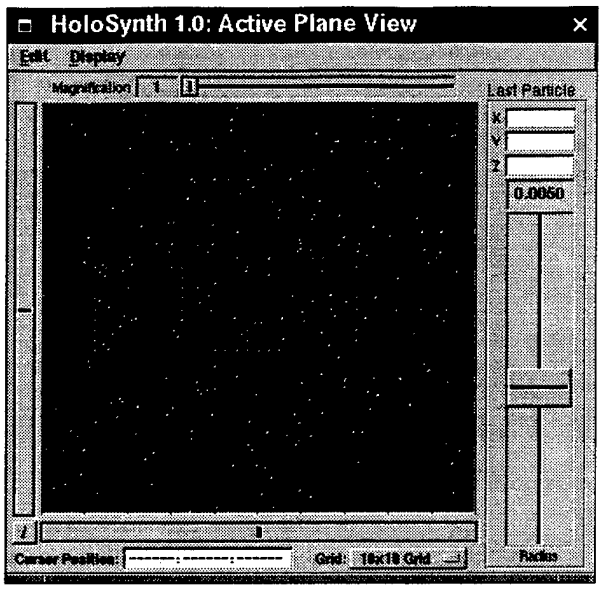


Figure 4-5: HoloSynth plane editor

of the cube. Only the active plane is visible in Figure 4-4, the clipping planes are hidden at the ends of their travel. On a color display the active plane is bright green, the rear cliiping plane is blue, and the front clipping plane is magenta. The active plane determines the z-coordinate that the plane editor uses when adding new particles. The front and rear clipping plane restrict the range of z coordinates visible in both the plane view and the 3-D view. In all cases, particles lying on the active plane are visible within both views. If the user wants to view *only* the particles on the active plane, then he places both clipping planes at the same limit of travel.

4.3.2.3 The Plane View

The plane view projects all particles visible in the 3-D view onto a single plane. Particles that are on the active (green) plane in the 3-D view are highlighted green in the plane view. Particles can be manually positioned or deleted on the active plane.

To add a particle, simply left click within the black portion of the active plane view. Accurate positioning of particles can be performed in two ways:

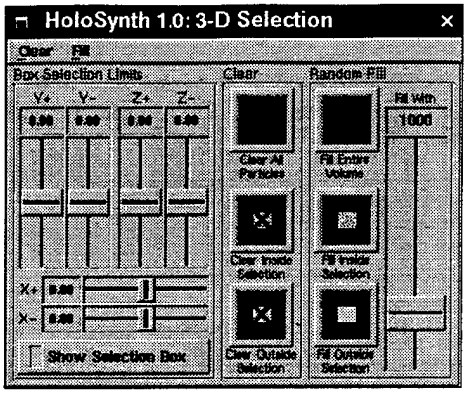


Figure 4-6: 3-D region selection

1. Turn on the coordinate grid. At the bottom of the plane view window is a selection list allowing presentation of grids of various densitites. High grid densities are convenient at high magnifications. While a grid is displayed the particles automatically snap to the nearest grid intersection. This can be controlled by the `Display->Snap to Grid` menu item, and can also be toggled using the 'g' key.

2. Use the real-time coordinate diplay. Any time the mouse is over the edit window its complete 3-D coordinate is show at the bottom of the window.

Deletion is accomplished by dragging with the right mouse button over the region containing the particles to be deleted. A red selection border will appear showing the selected region. To delete inside the selected region, use the menu item, `Edit->Delete Inside Selection` or the delete key. Using the `Edit->Delete Outside Selection` menu item will clear all particles visible in the plane editor but outside the selection box. The region deleted is bounded in the x and y directions by the selection box, and in the z dimension by the clipping planes. If a particle is visible in the plane editor, regardless of whether it is on the active plane, it can be deleted.

4.3.2.4 Select Regions

The 3-D Selection window, shown in Figure 4-6, allows the user to define a solid rectangular sub-volume within the 3-D view and perform the following operations:

1. Clear the entire volume.
2. Clear all particles inside the selected sub-volume
3. Clear all particles outside the selected sub-volume
4. Fill the entire volume with randomly placed particles. The slider at the right of the window controls how many particles will be added.
5. Fill the selected sub-volume with randomly placed particles. The slider at the right of the window controls how many particles will be added.
6. Fill outside the selected sub-volume with randomly placed particles. The slider at the right of the window controls how many particles will be added.

The sub-volume is displayed as a translucent yellow box with the 3-D view. Each wall of the selection region is controlled by its own slider. The region will only be displayed in the 3-D view if the `Show Selection Box` button has been selected.

4.3.2.5 Hologram Synthesis

Figure 4-7 shows the synthesizer controls. This allows easy control of the physical size of the simulation as well as recording plane distance and number of planes used to subdivide the simulation volume. The generated hologram will be displayed in a new window when the computations have completed. If the product is satisfactory, the user may save the hologram to a file using the Truevision Targa format or the Adobe Photshop Raw format.

4.3.3 Requirements

HoloSynth is written completely in C++. The Graphical User Interface uses the freely available platform independent user interface toolkit FLTK[13]. The

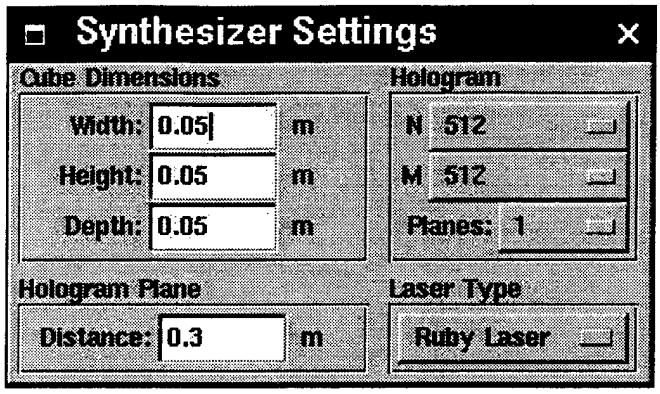


Figure 4-7: Synthesizer settings

hologram synthesizer use the FFTW fast fourier transform library available freely from MIT[14]. The program runs under X-Windows, and Microsoft Windows with OpenGL[15] or the Mesa 3-D graphics library (an OpenGL functional equivalent)[16].

4.3.4 Program Architecture

At its heart HoloSynth is a fancy list editor, editing a list of particle coordinates and radii. The core data structure is, appropriately, a linked list of particle objects. This allows an arbitrary number of particles to be added, restricted only by the memory available.

Figure 4-8 shows the schematic relationship between all program modules that make up HoloSynth. The complete source code to HoloSynth is presented in Appendix A.1.

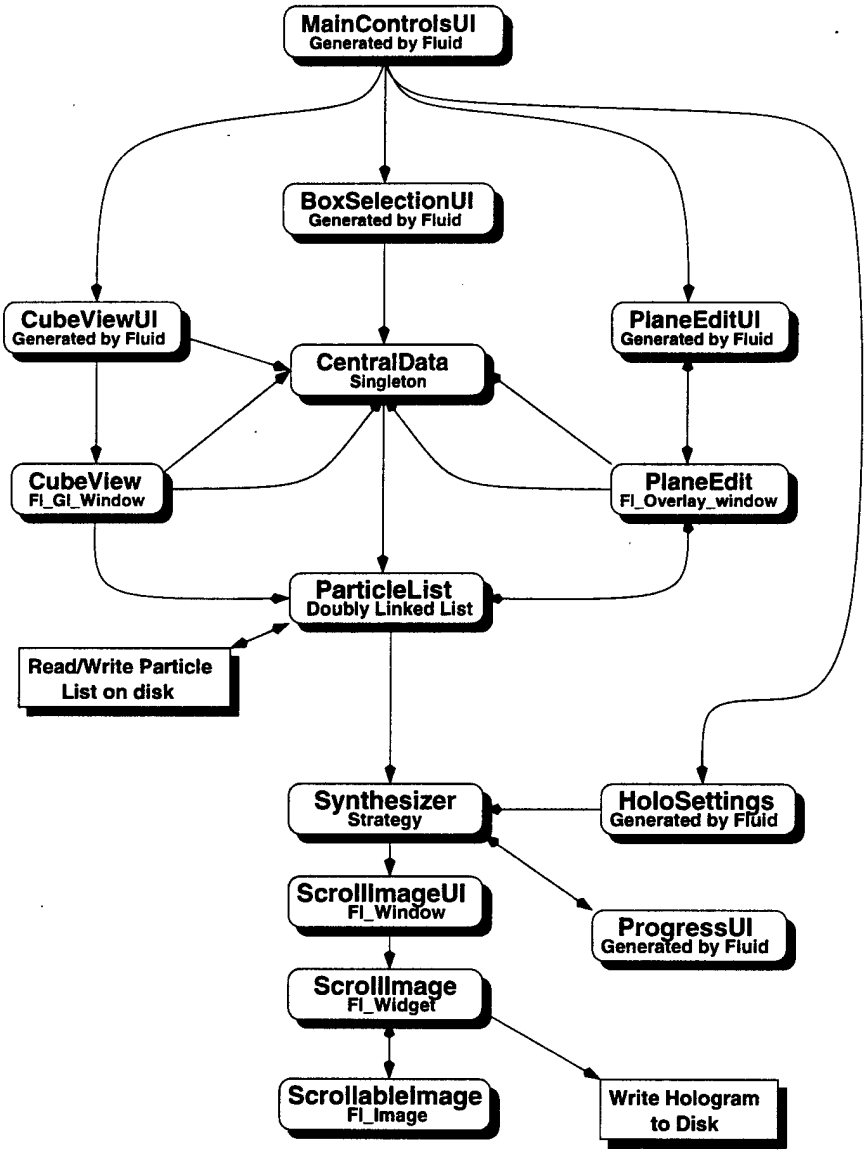


Figure 4-8: HoloSynth architecture

Chapter 5

Computational Reconstruction

5.1 Onural's Method

In 1985, Levent Onural proposed a technique based on a truncated inverse filter[12, 17]. The method is summarized here, source code is presented to perform the algorithm in Sections A.2.1 and A.2.2.

Starting from (2.17),

$$\psi_z(x,y) = U(x,y) \otimes h_z(x,y)$$

We let $U(x,y) \equiv 1 - a(x,y)$ where $a(x,y)$ is the opacity of the given plane. Now write the recording plane intensity,

$$\begin{aligned} I_z(x,y) &= \psi_z(x,y)\,\psi_z^*(x,y) \\ &= |(1 - a(x,y)) \otimes h_z(x,y)|^2 \\ &= 1 - a^*(x,y) \otimes h_z^*(x,y) - a(x,y) \otimes h_z(x,y) \\ &\quad + |a(x,y) \otimes h_z(x,y)|^2 \qquad (5.1) \\ &= 1 - a(x,y) \otimes [h_z^*(x,y) + h_z(x,y)] \\ &= 1 - a(x,y) \otimes 2\mathrm{Re}\{h_x(x,y)\} \qquad (5.2) \end{aligned}$$

Where $|a(x,y) \otimes h_z(x,y)|^2$ was dropped as insignificant in (5.1). Equation (5.2)

shows the intensity modeled as a DC shifted linear system with impulse response,

$$g(x,y) = 2\mathrm{Re}\{h_z(x,y)\} \tag{5.3}$$

No inverse of $g(x,y)$ in the normal sense exists, However, Onural showed in [12] that a series approximation to the Fourier transform of g can be found. If $\mathcal{G}(\omega_x,\omega_y) = \mathcal{F}[g(x,y)]$ then,

$$\mathcal{G}_M^{-1}(\omega_x,\omega_y) = \cos\frac{\lambda z}{4\pi}\left(\omega_x^2+\omega_y^2\right) + \sum_{k=1}^{2^{M-1}-1}\left\{\frac{M-1-[\![\log_2 k]\!]}{M}\right\} S_k(\omega_x,\omega_y) \tag{5.4}$$

where $[\![\log_2 k]\!]$ is the integer obtained by truncating the value of $\log_2 k$ and,

$$S_k(\omega_x,\omega_y) = (-1)^k \cos\frac{\lambda z}{4\pi}(2k+1)\left(\omega_x^2+\omega_y^2\right) \tag{5.5}$$

yielding,

$$g_M^{-1}(x,y) = \mathcal{F}^{-1}\left[\mathcal{G}_M^{-1}(\omega_x,\omega_y)\right] \tag{5.6}$$

Looking back at the hologram recording given by (5.2),

$$\begin{aligned} a(x,y)\otimes 2\mathrm{Re}\{h_z(x,y)\} &= 1-I(x,y) \\ \mathcal{F}[a(x,y)]\,\mathcal{G}(\omega_x,\omega_y) &= \mathcal{F}[1-I(x,y)] \\ a_z(x,y) &= \mathcal{F}^{-1}\left[\mathcal{F}[1-I(x,y)]\,\mathcal{G}_{M,z}^{-1}(\omega_x,\omega_y)\right] \end{aligned} \tag{5.7}$$

yielding the original opacity function.

5.2 Simulation Results

5.2.1 Synthesis and Computational Reconstruction

Figure 5-2 is a portion of a hologram generated by HoloSynth. The particle configuration consisted of 2328 particles. 2000 of the particles were randomly distributed

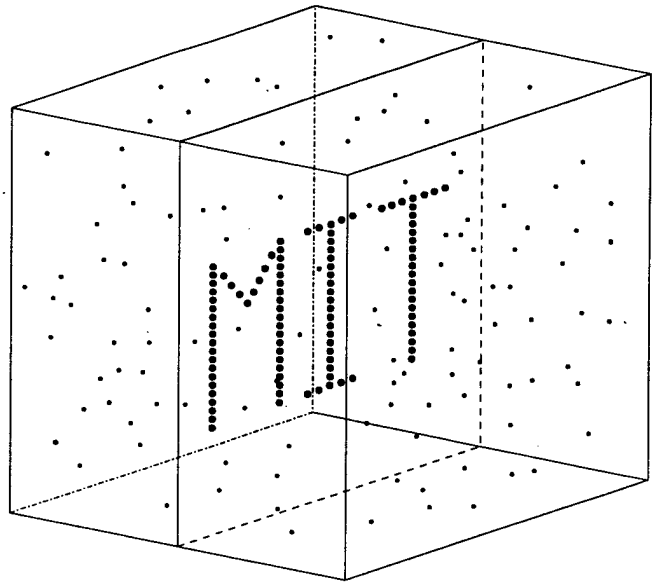


Figure 5-1: Schematic of the particle configuration used to generate Figure 5-2

throughout a 5cm cube. Along the center plane parallel to the wavefront 328 particles were positioned to spell the letters "MIT". Figure 5-1 illustrates a schematic of that configuration. The synthetic hologram itself was 2048×2048 pixels, and the volume was subdivided into 1024 planes, for a z-axis discretization of 0.04mm. The simulated particle diameter was 25μm. The portion of the hologram shown in 5-2 is 512×512 pixels and its upper left corner is 205 pixels from the left extremity of the original and 741 pixels from the top. Although it contains the 'M' from "MIT" , it is impossible to distinguish from the diffraction patterns produced by the other particles.

Figure 5-3 through 5-8 show the results of reconstructing Figure 5-2 at various distances between 30.0cm and 30.5cm from the recording plane using Onural's method with $M = 5$. The images generated are the opacity functions, with black being transparent, white being opaque.

In Figure 5-3(a) the 'M' is clearly visible on a computer monitor with some slight clouds from out-of-focus particles. Figure 5-4 is the same hologram reconstructed at 30.1cm. Here there is little visible difference from the previous figure. This is an artifact of the way the decoder transform real valued intensities into

gray scale bitmap images.

To translate from an arbitrary field of real number to a field of 8 bit integers, the decoder first finds the maximum and minimum values present in the real array, it then scales and translates each of these values to fill the range from 0 to 256, in order to maximize use of dynamic range available in an 8 bit gray scale image.

The 30.1cm reconstruction has a much lower maximum intensity than the 30.0 cm reconstruction. Volume reconstruction, where many planes will be reconstructed at a time, in order to reconstruct flow fields, will have to keep track of this scaling issue between planes.

The important thing to note about this series of decodes, is that while the source hologram was constructed using a z discretization of 0.04mm, the reconstruction algorithm z-discrimination was of the order of millimeters. Thus, the original image could have been discretized into 64 steps, or 0.78mm slabs, requiring 1/16th the computation time with little or no loss of reconstruction accuracy.

5.2.2 Image Postprocessing

The next step in reconstruction is to automatically detect particles in each plane and register their x, y, and z coordinates, rather than simply displaying an image. Repeatedly reconstructing at different depths and detecting the particles will replace the time consuming 3-D scanning.

In Figures 5-3 through 5-8 subfigure (b) and (c) show the results of applying some simple image processing techniques to the original reconstructions to enhance the output. Subfigure (b) in each figure applies 'edge detection', which simply filters the image marking any region where the gradient of the image exceeds a certain threshhold. In the case of the grayscale images being used here, the gradient is simply the difference in grayscale value between adjacent pixels.[18] Subfigure (c) in each figure 'threshholds' the image, changing a pixel to white (value 255) or black (value 0) depending on whether or not the pixel value exceeds a preset threshhold.[1]

[1]The image processing was accomplished with the freely available GNU Image Manipulation

The combination of edge detection and threshholding can be used very effectively to automatically detect particle positions. At the end of the image processing, there are only regions of black (value 0) with very small regions of white (value 255). The white regions correspond to particles and can easily be automatically tabulated and mapped into x and y coordinates.

Program (GIMP, `http://www.gimp.org`), which is very similar to Adobe Photoshop®.

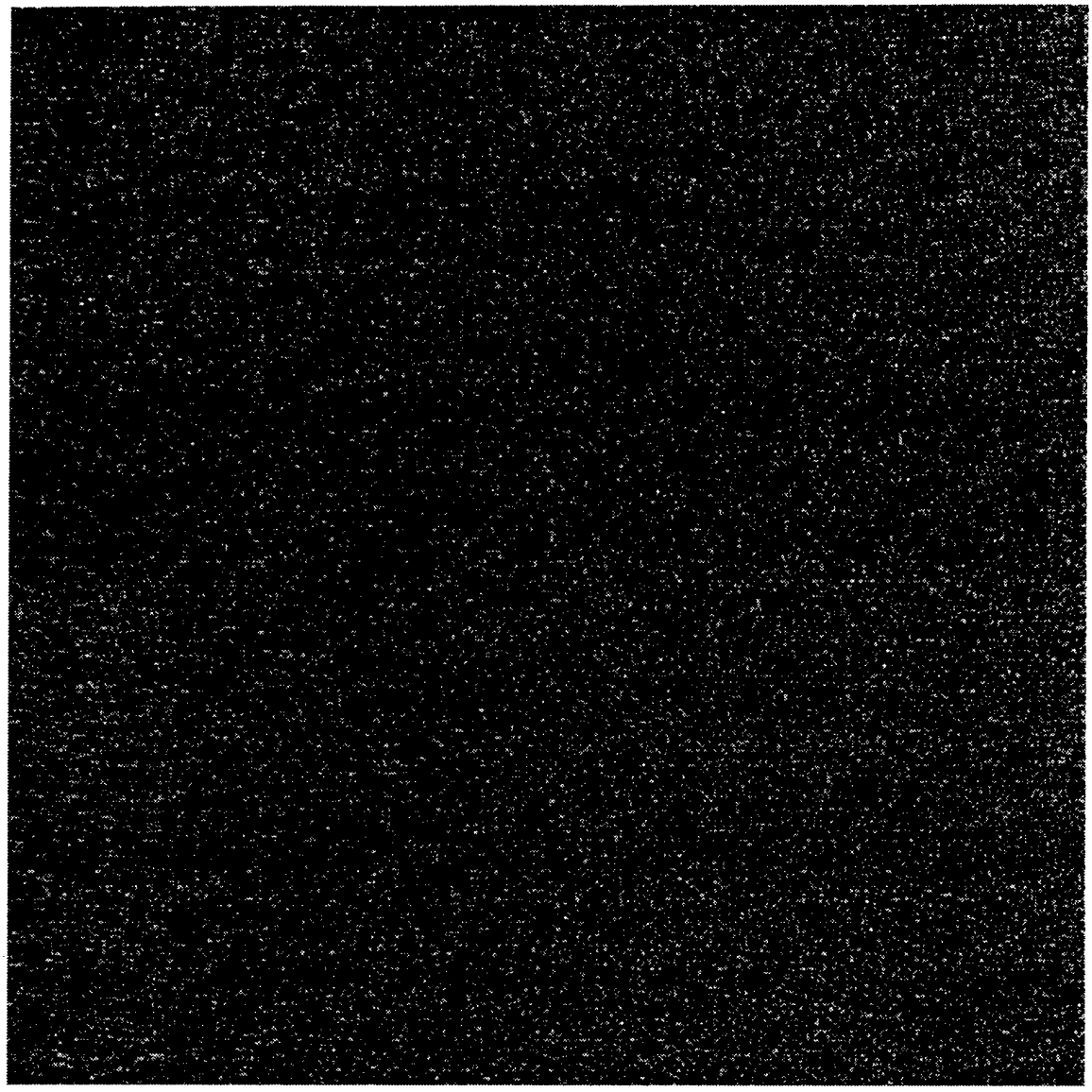
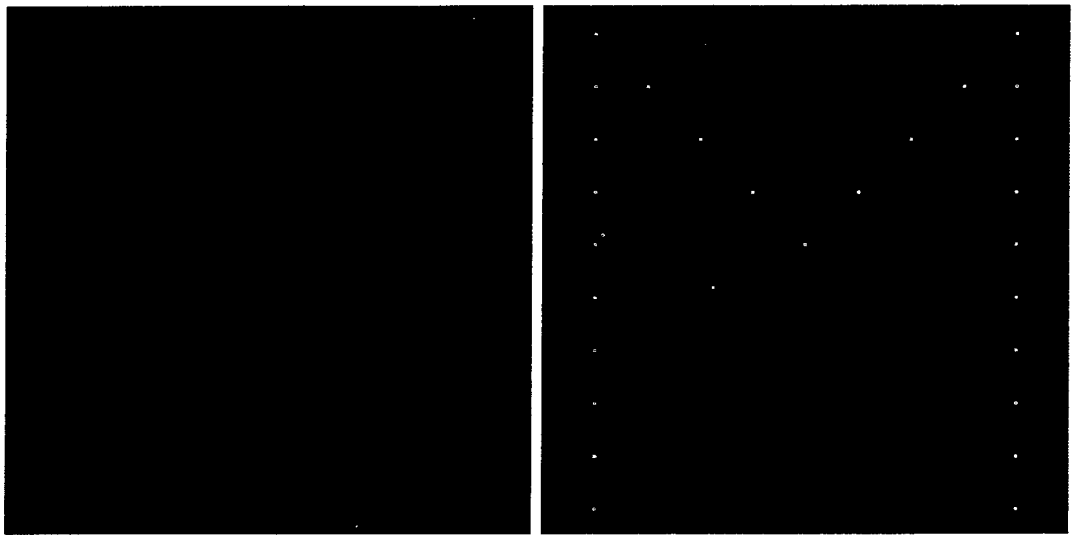
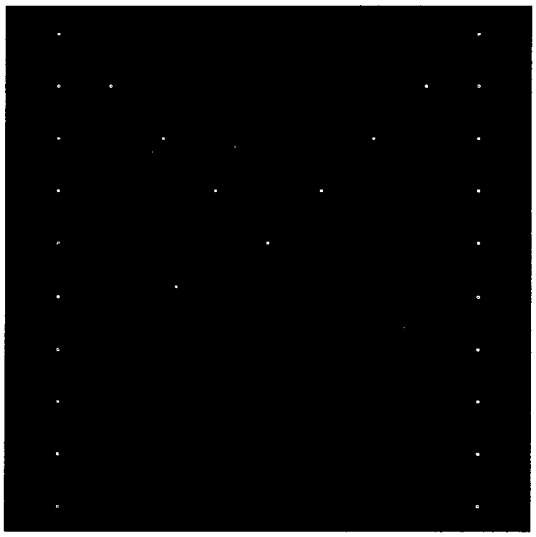


Figure 5-2: Portion of a synthetic hologram. The original synthetic hologram was 2048×2048 pixels. This portion is 512×512 pixels, cropped from the original hologram to make the images presentable in printed form.



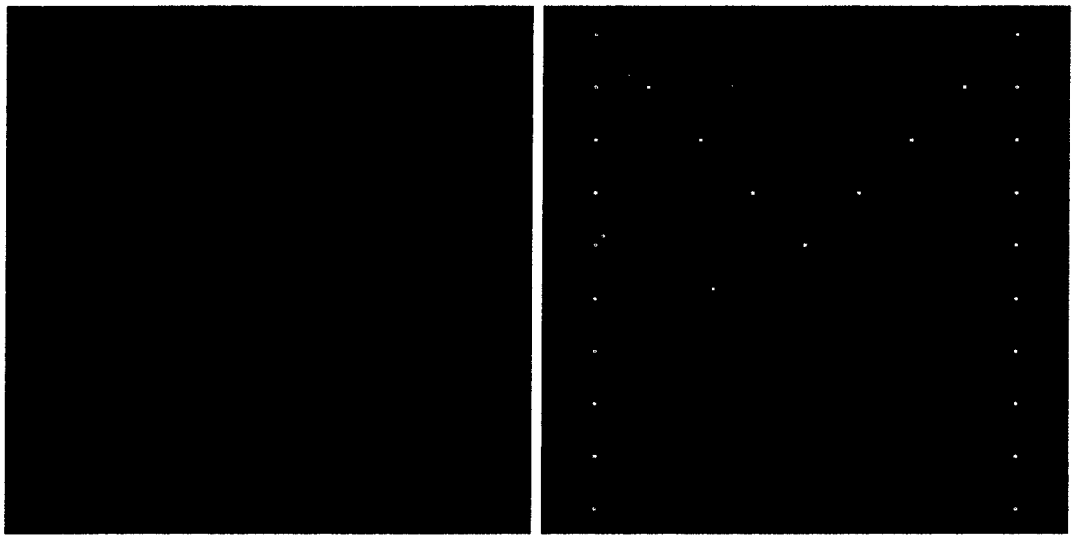
(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.



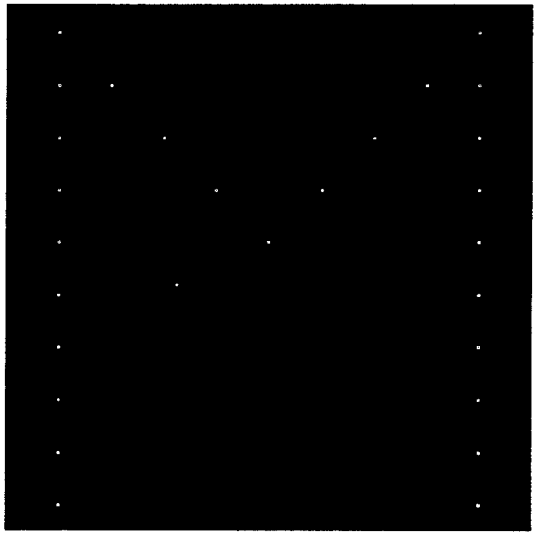
(c) Image after enhancing with edge detection and threshholding.

Figure 5-3: Figure 5-2 decoded at a depth of 0.300m.



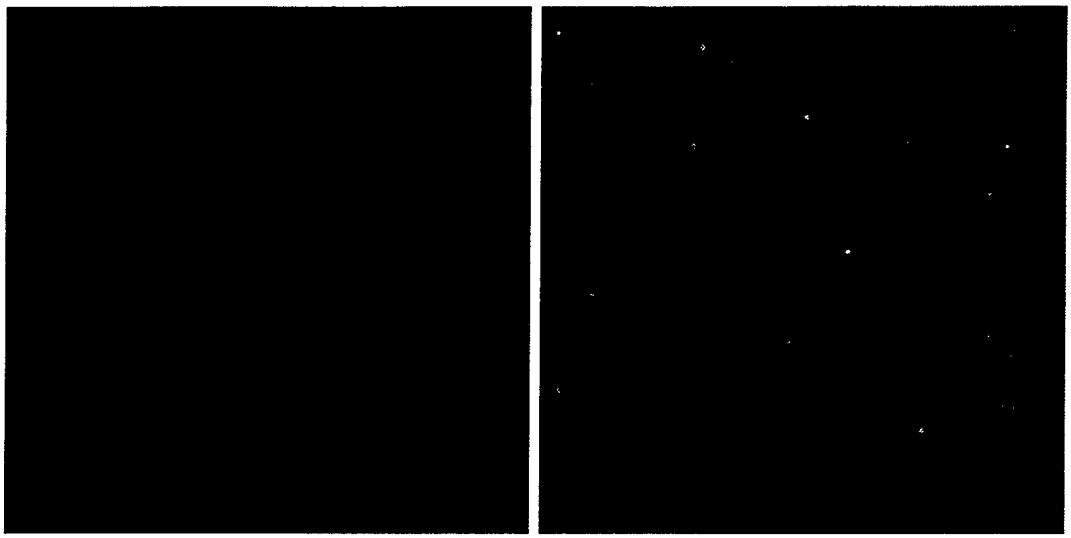
(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.



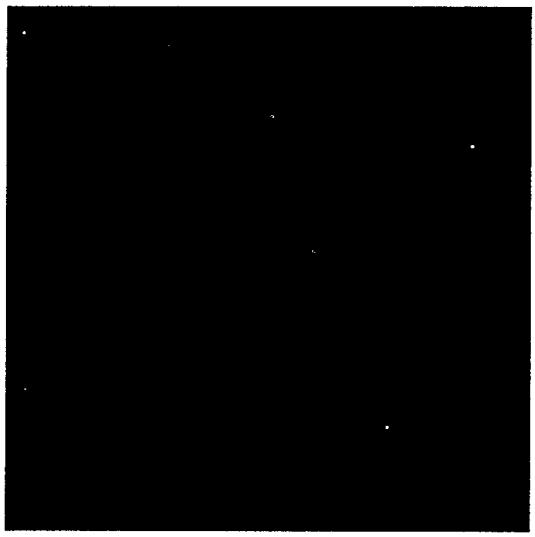
(c) Image after enhancing with edge detection and threshholding.

Figure 5-4: Figure 5-2 decoded at a depth of 0.301m.



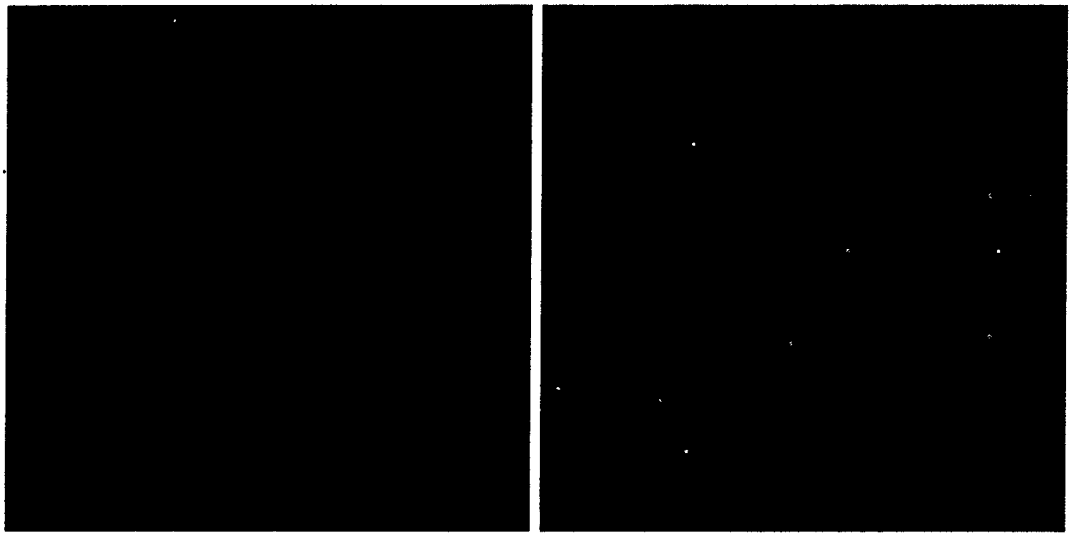
(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.



(c) Image after enhancing with edge detection and threshholding.

Figure 5-5: Figure 5-2 decoded at a depth of 0.302m.

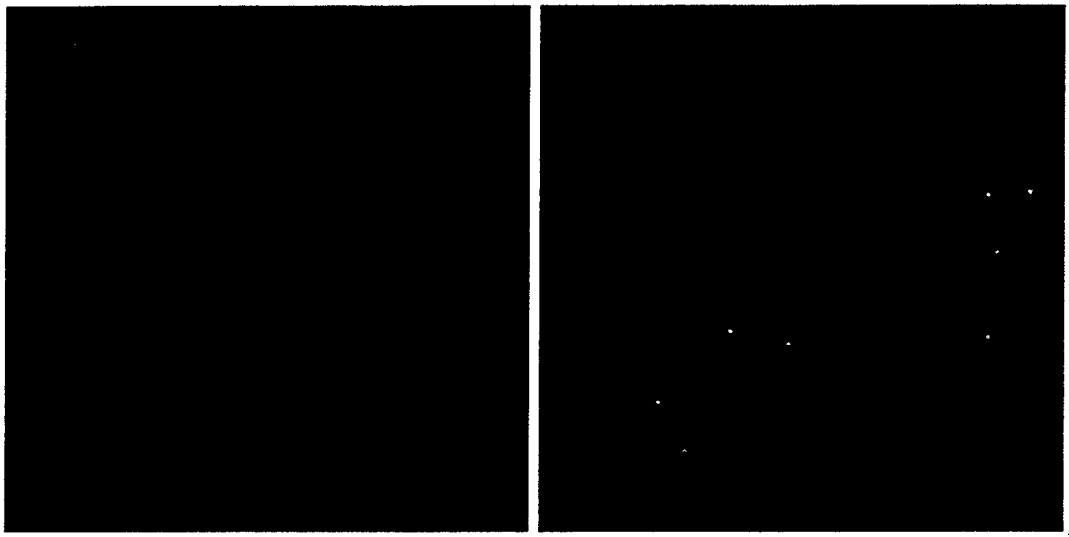


(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.

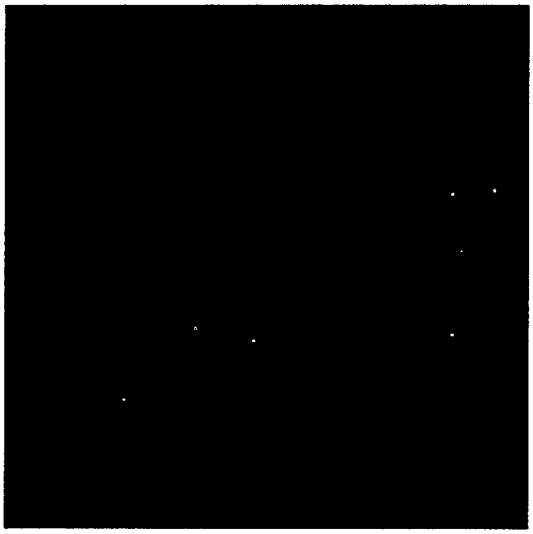
(c) Image after enhancing with edge detection and threshholding.

Figure 5-6: Figure 5-2 decoded at a depth of 0.303m.



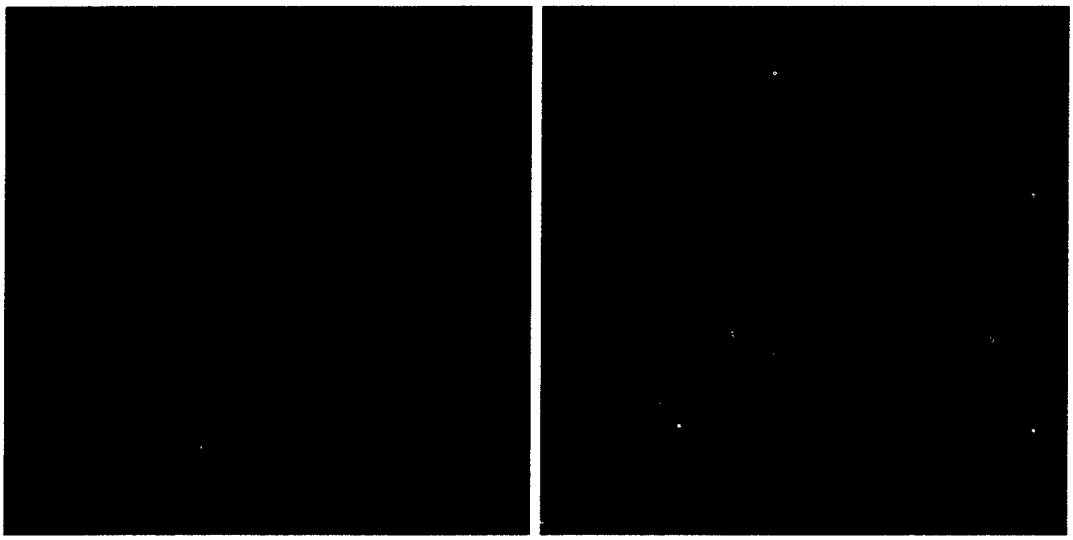
(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.



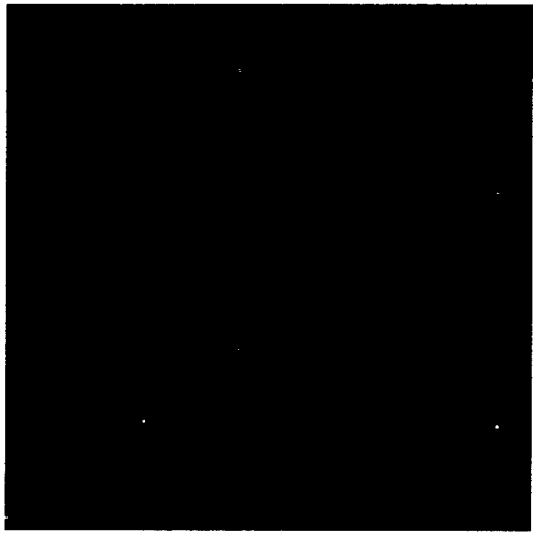
(c) Image after enhancing with edge detection and threshholding.

Figure 5-7: Figure 5-2 decoded at a depth of 0.304m.



(a) Raw Reconstructed Image

(b) Image after enhancing with edge detection.



(c) Image after enhancing with edge detection and threshholding.

Figure 5-8: Figure 5-2 decoded at a depth of 0.305m.

Chapter 6

Conclusions and Recommendations

It is clear that computational reconstruction of HPIV is practicable. The procedure that Onural proposed in 1985 was not then practical due to the computational power commonly available. Modern computing can easily handle realistic holograms and the numeric techniques of normal PIV could be extended to deal with the particle identification problem in HPIV. The simulator presented in this thesis will be useful in any setting when attempting to validate particular reconstruction algorithms.

Future work should concentrate on extending the reconstruction prosses fully into three dimensions and developing particle identification techniques. Automatically determining the velocity vector for a particle is the ultimate goal.

Appendix A

Source code

A.1 `HoloSynth`: Hologram Synthesizer.

This is the source code for HoloSynth, a program for easily creating particle configuration and holograms for use calibrating computational reconstruction algorithms.

A.1.1 OnuralSynthesizer.hpp

```
#ifndef ONURALSYNTHESIZER_H
//!include guard
#define ONURALSYNTHESIZER_H 1

#include "Synthesizer.hpp"
#include "ParticleList.hpp"
#ifndef FLMATH
#define FLMATH
#include <FL/math.h>
#endif
#include "RGBImage.hpp"

/*!\class OnuralSynthesizer
 *
 *\brief The algorithm use for actually generating the hologram
 *
 *The forward diffraction algorithm implement a method described in
 * "Digital Decoding of In-Line Holograms" a PhD thesis by Leven Onural
 * */

class OnuralSynthesizer:public Synthesizer
{
public:
    //! created with a reference to the current particle list
```

```
    OnuralSynthesizer(ParticleList *pl);
    ~OnuralSynthesizer();
    //! Generate, organized all the FFTs and diffraction calculations.
    int generate();

private:

    RGBImage *hologram;

    void diffract(fftw_complex *image);
    void clearField(fftw_complex *field, fftw_real val);
    void plotParticle(particle *c,fftw_complex *plane);
    void plot(int p_X, int p_Y, fftw_complex *plane);
    void addto(fftw_complex *a1, fftw_complex *a2);
    void MagtoRGB(fftw_complex *data);

    void cleanup();

    fftwnd_plan fftwFwdPlan;
    fftwnd_plan fftwBwdPlan;

    fftw_complex *oplane;
    fftw_complex *hplane;
};


#endif
```

A.1.2 OnuralSynthesizer.hpp

```
#include <FL/Fl.H>
#include "ProgressUI.hpp"
#include "ScrollImageUI.hpp"
#include <FL/Fl_Image.H>
#include "OnuralSynthesizer.hpp"

#include "StatusBox.hpp"
#define MIN(a, b) (((a) < (b)) ? (a) : (b))
#define MAX(a, b) (((a) > (b)) ? (a) : (b))


OnuralSynthesizer::OnuralSynthesizer(ParticleList *pl)
        :Synthesizer(pl)
{
}

OnuralSynthesizer::~OnuralSynthesizer()
{
    cleanup();
```

```
}

int OnuralSynthesizer::generate()
{
    int progresscount=0;

    progressUI→setMax(5+5*d_Planes);
    progressUI→update(0);
    progressUI→reset();
    progressUI→show();
    Fl::wait(0.0);

    particle *curr;
    int pixels, i1, pcount, Ncalc, Mcalc;

    fftw_real zgrain, Zmin, Zmax;



/*************************************************************
 * Initialization.
 *************************************************************/

    pixels=2*(2*N/2+1)*2*M;

    Ncalc=2*N; //Double the size for padding out the edge effects.
    Mcalc=2*M;

    if(progressUI→setStatus("Allocating memory...")){
        cleanup();
        progressUI→hide();
        return 0;
    };

    hplane=(fftw_complex *) calloc( pixels,sizeof(fftw_complex));
    oplane=(fftw_complex *) malloc( pixels*sizeof(fftw_complex));

    progressUI→update(++progresscount);

    zgrain=(fftw_real)1/(2*d_Planes);//The thickness of a HALF plane

    if(progressUI→setStatus("Generating FFTW Plans...")){
        cleanup();
        progressUI→hide();
        return 0;
    };


    fftwFwdPlan=fftw2d_create_plan(Ncalc, Mcalc, FFTW_FORWARD,
```

```
FFTW_IN_PLACE);
    fftwBwdPlan=fftw2d_create_plan(Ncalc, Mcalc, FFTW_BACKWARD,
FFTW_IN_PLACE);

    progressUI→update(++progresscount);
    //Remember we will eventually need to split out the vector end of things.

    pList→sortListOnZ();


    curr=pList→first();

    for(i1=1;i1<=d_Planes;i1++){


        Fl::wait(0);

        if(progressUI→setStatus("Clearing Object Plane...")){
            cleanup();
            progressUI→hide();
            return 0;
        };

        clearField(oplane,1.0);
        progressUI→update(++progresscount);

        Zmin=(i1-1)*2*zgrain-0.5;
        Z=(Zmin+zgrain)*depth;//Physical Z
        Zmax=Zmin+2*zgrain;

        pcount=0;

        if(progressUI→setStatus("Plotting Particles...")){
            cleanup();
            progressUI→hide();
            return 0;
        };

        while((curr!=NULL)&&(curr→Z>Zmin)&&(curr→Z<=Zmax)){
            plotParticle(curr,oplane);
            pcount++;
            curr=curr→next;
        }
        progressUI→update(++progresscount);

        if(pcount>0){


            if(progressUI→setStatus("Performing Forward FFT...")){
```

```
                cleanup();
                progressUI→hide();
                return 0;
            };

            fftwnd_one(fftwFwdPlan, oplane, 0);
            progressUI→update(++progresscount);

            if(progressUI→setStatus("Diffracting..")){
                cleanup();
                progressUI→hide();
                return 0;
            };
            diffract(oplane);
            progressUI→update(++progresscount);

            if(progressUI→setStatus("Adding...")){
                cleanup();
                progressUI→hide();
                return 0;
            };
            addto(hplane,oplane);
            progressUI→update(++progresscount);
        }
        progresscount=5*i1+2;
        progressUI→update(progresscount);
    }
    hplane[0].re=1000000;

    progressUI→setStatus("Performing Backward FFT...");
    progressUI→update(++progresscount);
    fftwnd_one(fftwBwdPlan, hplane, 0);
    progressUI→setStatus("Calculating Intensity...");

    MagtoRGB(hplane);

    //The hologram is finished, now display it...

    Fl_Image *image=new Fl_Image(hologram→data(), N, M);
    progressUI→setStatus("Displaying...");
    progressUI→update(++progresscount);
    ScrollImageUI *si=new ScrollImageUI(image , 0,0, 500,500);
    si→show();

    progressUI→hide();
    cleanup();

    return 1;
}
```

```
void OnuralSynthesizer::addto(fftw_complex *a, fftw_complex *b)
{
    int i1;
    for (i1=0; i1 < ( 4 * N * M ); i1++){
        if((fabs(b[i1].re)>=0.0000001)&&(fabs(b[i1].im)>=0.0000001))
        {
            a[i1].re += b[i1].re;
            a[i1].im += b[i1].im;
        }
    }
}

void OnuralSynthesizer::clearField(fftw_complex *field, fftw_real val)
{

    int i1;
    for(i1=0; i1 < ( 4 * N * M ); i1++){
        field[i1].re = val;
        field[i1].im = 0.0;
    }

}
void OnuralSynthesizer::plotParticle(particle *c,fftw_complex *plane)
{
    int x, y, r;

    x=(int)((c->X + 0.5)*N);
    y=(int)((0.5 - c->Y)*M);
    r=(int)(c->R*N);

    switch (r)
        {
            case 6:
                plot(x-1, y+1, plane);
                plot(x-1, y-1, plane);
                plot(x+1, y-1, plane);
                plot(x+1, y+1, plane);
            case 2:
            case 3:
            case 4:
            case 5:
                plot(x, y-1, plane);
                plot(x, y+1, plane);
                plot(x-1, y, plane);
                plot(x+1, y, plane);
            default:
                plot(x, y, plane);
```

```
        }

}

void OnuralSynthesizer::plot(int p_X, int p_Y, fftw_complex *plane)
{
    int pos=2 * p_Y * N + p_X;
    if(pos>0){
        plane[ pos].re=0.0;
        plane[ pos].im=0.0;
    }

}

#define F_PI 4.0*3.1415926535
void OnuralSynthesizer::diffract(fftw_complex *image)
{
    int x, y, Ncalc, Mcalc;
    long ik;
    fftw_real lz, cc;
    fftw_real wx, wy, a, ca, sa;

    Ncalc=2*N;
    Mcalc=2*M;

    lz=L*(distance-Z);
    ik=0;


/* FFTW puts frequency space origin at the upper right corner of the
 * array. Positive frequencies build from the upper left corner
 * diagonally down through the upper left quadrant. The other corners
 * of the array the array are ALSO at the origin, so the very center
 * of the array is the HIGHEST frequency.
 *
 * In order to use the diffraction filters we MUST take that into
 * account, that is what the strange trigraph statements take care of
 * below */
    for (y=0;y<Mcalc;y++)
    {
        wy = (y<M) ? y/height : (y-Mcalc)/height;
// wy = (y<M) ? y/height : (Mcalc-y)/height;
        wy * = wy;
        for(x=0;x<Ncalc;x++){

            wx=(x<N) ? x/width : (x-Ncalc)/width;
// wx=(x<N) ? x/width : (Ncalc-x)/width;
                a=lz*(wy+wx*wx)*3.1415926535;
// a=lz*(wy+wx*wx)/F_PI;
```

```
            ca=cos(a);
            sa=sin(a);
            cc=image[ik].re*ca-image[ik].im*sa;
            image[ik].im=image[ik].re*sa+image[ik].im*ca;
            image[ik].re=cc;
            ik++;
        }
    }
}

void OnuralSynthesizer::MagtoRGB(fftw_complex *data)
{
    int i, j, ij, grey, Ncalc, Mcalc;
    fftw_real v, vmax, vmin, v1, v2;



    Ncalc=2*N;
    Mcalc=2*M;

    hologram=new RGBImage(N,M);

    vmax=0.0;
    vmin=1e6;
    ij=0;

    //Scale the result to take maximum advantage of 8bit range
    for(i=0;i<Mcalc*Ncalc;i++){
            v1=data[i].re;
            v2=data[i].im;
            v=sqrt(v1*v1+v2*v2);

            vmax=MAX(v, vmax);
            vmin=MIN(v, vmin);
    }

    vmax=1/(vmax-vmin);


    RGBColor c;
    for(i=0;i<M;i++){
        for(j=0;j<N;j++){
            ij=i*Ncalc+j;

            v1=data[ij].im;
            v2=data[ij].re;
            v=(sqrt(v1*v1+v2*v2)-vmin)*vmax;
            data[ij].re=v;
```

```
                grey=(int)floor((fftw_real)(v*255.0));
                        c.B=grey;
                        c.R=grey;
                        c.G=grey;
                        hologram→plot(j, i, c);
            }
        }

}

void OnuralSynthesizer::cleanup()
{
    free(hplane);
    free(oplane);
    fftwnd_destroy_plan(fftwFwdPlan);
    fftwnd_destroy_plan(fftwBwdPlan);
}
```

A.2 Decoder: Onural Reconstruction Algorithm.

This is the source code for an Onural reconstruction algorithm.

A.2.1 OnuralDecoder.hpp

```
#ifndef ONURALDECODER_HPP
#define ONURALDECODER_HPP 1
#include "Decoder.hpp"
#include "ProgressUI.hpp"
#include <fftw.h>
class OnuralDecoder: public Decoder
{
public:
    OnuralDecoder();
    virtual uchar* decode(uchar *data, int N, int M,
                          float W, float H, float L,
                          float D);

private:
    int d_ImageN;
    int d_CalcN;
    int d_ImageM;
    int d_CalcM;

    float d_ImageW;
    float d_ImageH;
    float d_ImageLambda;
    float d_decodeDist;

    fftw_complex *d_Field;

    uchar * d_data;


    void OnuralDecoder::noDC1quad(fftw_complex *field, fftw_real averageValue);

    fftw_real OnuralDecoder::aprinv(fftw_real zz, int Mstage);

    void OnuralDecoder::onural(fftw_complex *hg,
                               int N, int M,
                               fftw_real L, fftw_real Z,
                               fftw_real X, fftw_real Y,int Mstage);


    fftw_real Magnitude(fftw_complex *field);

    void uchar2field();
    uchar *field2uchar();
```

```
    fftw_real *clipField();


    uchar *OnuralDecoder::mainLoop();
        ProgressUI *progress;
};
#endif
```

A.2.2 OnuralDecoder.cpp

```
#include "OnuralDecoder.hpp"
#include <fftw.h>
#include "StatusBox.hpp"
#include <FL/Fl.H>
#include <math.h>

#define PI 3.14159265359
#define TPI 6.28318530718
#define MIN(a, b) (((a) < (b)) ? (a) : (b))
#define MAX(a, b) (((a) > (b)) ? (a) : (b))

OnuralDecoder::OnuralDecoder()
{
    d_ImageN=0;
    d_ImageM=0;

    d_ImageW=0;
    d_ImageH=0;
    d_ImageLambda=0;
    d_decodeDist=0;
    d_data=0;
        progress=new ProgressUI();
}

uchar *OnuralDecoder::decode(uchar* data, int N, int M, float W, float H, float L,
float D)
{
    d_ImageN=N;
    d_CalcN=2*N;

    d_ImageM=M;
    d_CalcM=2*M;

    d_ImageW=W;
    d_ImageH=H;
    d_ImageLambda=L;
    d_decodeDist=D;
    d_data=data;
```

```
    return mainLoop();
}


/* Remove the DC component from only the first Quadrant, the other
 * three quadrants are ALREADY ZERO*/

void OnuralDecoder::noDC1quad(fftw_complex *field, fftw_real averageValue)
{
    int i, j, pos;
    for(i=0; i<d_ImageM; i++){
        for(j=0; j<d_ImageN; j++){
            pos=i*d_CalcN+j;
            field[pos].re -= averageValue;
        }
    }
}

fftw_real OnuralDecoder::aprinv(fftw_real zz, int Mstage)
{
    fftw_real cc,ss,powKa;
    int k,powK;

    cc=cos(zz);
    ss=0;

    if(Mstage>1){
        ss=2*sin(zz);
        cc+=ss*sin(2*zz);
        ss* =2*cos(2*zz);
    }
    for (k=3;k<=Mstage;k++){
        powK=4<<(k-3); // powK=2**(k-1), fine for small k, and FAST

        powKa=(fftw_real)powK*zz;

        cc+=ss*sin(powKa);
        ss* =2*cos(powKa);
    }
    return 2*cc/Mstage;

}

void OnuralDecoder::onural(fftw_complex *hg,
        int N, int M,
        fftw_real L, fftw_real Z,
        fftw_real X, fftw_real Y,int Mstage)
{
```

```
    int x, y, m12, n12;
    long ik;
    fftw_real alpha, lz, wx2, wy2;

    lz=L*Z;

    m12=M/2;
    n12=N/2;

    ik=0;


    for (y=0;y<M;y++)//M
    {
        wy2 = (y<m12) ? y/Y : (y-M)/Y;
        wy2 * = wy2;
        for(x=0;x<N;x++)//N
        {
            wx2 = (x<n12) ? x/X : (x-N)/X;
            wx2 * = wx2;

            alpha=PI*lz*(wy2+wx2);

            hg[ik].re* =aprinv(alpha, Mstage);
            ik++;
        }
                if(progress→update(3)){
                        progress→hide();
                        return;
                };

    }
}


fftw_real OnuralDecoder::Magnitude(fftw_complex *field)
{
    int i;
    fftw_real v,sumV, v1, v2;

    sumV=0;

    for(i=0; i < d_CalcN*d_CalcM; i++){
            v1=field[i].re;
            v2=field[i].im;
            v=sqrt(v1*v1+v2*v2);
            field[i].re=v;
            field[i].im=0;
            sumV+=v;
```

```
    }


    return sumV/(d_CalcN * d_CalcM);

}

void OnuralDecoder::uchar2field()
{
    int arraysize=d_CalcN*d_CalcM*sizeof(fftw_complex);
    int fieldpos, ucharpos, x, y;
    uchar minval, maxval;
    fftw_real scale;

    d_Field=(fftw_complex *)calloc(arraysize, 1);//calloc clears to zero...
    minval=0;
    maxval=255;

    for(y=0; y< d_ImageM; y++ ){
        for(x=0; x<d_ImageN; x++){
            ucharpos=3*(x + y*d_ImageM);
            minval=MIN(minval, d_data[ucharpos]);
            maxval=MAX(maxval, d_data[ucharpos]);
        }
    }
    scale=1.0/(fftw_real)(maxval-minval);

    for( y=0; y< d_ImageM; y++ ){
        for( x=0; x<d_ImageN; x++){

        fieldpos=x + y*d_CalcN;
        ucharpos=3*(x + y*d_ImageN);

        d_Field[fieldpos].re=((fftw_real)(d_data[ucharpos]-minval))*scale;
        }
    }

}

uchar *OnuralDecoder::field2uchar()
{
    int arraysize=3*d_ImageN*d_ImageM*sizeof(uchar);
    int fieldpos, ucharpos, x, y;
    fftw_real minval, maxval, v, vre, vim;
    fftw_real scale, val;

    uchar *retval;

    retval=(uchar* )calloc(arraysize, 1);//calloc clears to zero...
```

```
    minval=1e20;
    maxval=0;

    for(y=0; y< d_ImageM; y++ ){
        for(x=0; x<d_ImageN; x++){
            fieldpos=x + y*d_CalcM;
            vre=d_Field[fieldpos].re;
            vim=d_Field[fieldpos].im;
            v=sqrt(vre*vre+vim*vim);

            minval=MIN(minval, v);
            maxval=MAX(maxval, v);
            d_Field[fieldpos].re=v;

        }
    }
    scale=255.0/(maxval-minval);

    for(y=0; y< d_ImageM; y++ ){
        for(x=0; x<d_ImageN; x++){
            fieldpos=x + y*d_CalcN;
            ucharpos=3*(x + y*d_ImageN);

            val=(uchar)((d_Field[fieldpos].re-minval)*scale);
            retval[ucharpos]=(uchar)val;
            retval[ucharpos+1]=(uchar)val;
            retval[ucharpos+2]=(uchar)val;

        }
    }
    return retval;
}


uchar* OnuralDecoder::mainLoop()
{

    fftw_real DC;
    int progresscount=0;

        progress->setMax(5);
    progress->update(0);
    progress->reset();
    progress->show();

        if(progress->setStatus("Transforming Input...")){
        progress->hide();
```

```
    return 0;
};



fftwnd_plan ForwardFFTPlan, BackwardFFTPlan;

uchar2field();

    progress→update(++progresscount);
if(progress→setStatus("Making FFTW Plans...")){
    progress→hide();
    return 0;
};

    ForwardFFTPlan=fftw2d_create_plan(d_CalcN, d_CalcM,
                                      FFTW_FORWARD,
                                      FFTW_ESTIMATE|FFTW_IN_PLACE);

BackwardFFTPlan=fftw2d_create_plan(d_CalcN, d_CalcM,
                                       FFTW_BACKWARD,
                                       FFTW_ESTIMATE|FFTW_IN_PLACE);


DC=4*Magnitude(d_Field); //The 4 handles the doubling of arraysize



noDC1quad(d_Field, DC);

    progress→update(++progresscount);
    if(progress→setStatus("Calculating Forward transform...")){
    progress→hide();
    return 0;
};


fftwnd_one(ForwardFFTPlan, d_Field, 0);
d_Field[0].re=0;
d_Field[0].im=0;

    progress→update(++progresscount);
if(progress→setStatus("Starting Onural reconstruction...")){
    progress→hide();
    return 0;
};

onural(d_Field, d_CalcN, d_CalcM, d_ImageLambda, d_decodeDist,
       d_ImageW, d_ImageH, 5);
```

```
        progress→update(++progresscount);
    if(progress→setStatus("Calculating the Reverse Transform...")){
        progress→hide();
        return 0;
    };


    fftwnd_one(BackwardFFTPlan, d_Field, 0);

        progress→update(++progresscount);
    if(progress→setStatus("Complete")){
        progress→hide();
        return 0;
    };

    uchar* decoded=field2uchar();

    free(d_Field);

        progress→hide();
    return decoded;

};
```

References

[1] J. Zhang, B. Tao, and J. Katz. Turbulent Flow Measurement in a Square Duct with Hydrid Holographic PIV. *Experiments In Fluids*, To Be Published 1997.

[2] Thierry Loyau Jean-Claude Pascal and Paul Gaillard. Broadband acoustic holography reconstruction from acoustic intensity measurements. i: Principle of the method. *Journal of the Acoustic Society of America*, 84(5):1744–1750, November 1988.

[3] Chandra S. Vikram. *Particle Field Holography.* Cambridge Studies in Modern Optics. Cambridge University Press, 1992.

[4] P. Hariharan. *Optical Holography.* Cambridge Studies in Modern Optics. Cambridge University Press, 2 edition, 1996.

[5] Joseph W. Goodman. *Intoduction to Fourier Optics.* McGraw Hill Companies, Inc., 1996.

[6] Walter Koechner. *Solid-State Laser Engineering*, volume 1 of *Springer Series in Optical Sciences.* Springer, 4 edition, 1996.

[7] Jeff Hecht. *Understanding Lasers: An Entry-Level Guide.* IEEE Press, 2 edition, 1993.

[8] Jeff Hecht. *The Laser Guidebook.* McGraww-Hill Book Company, 1986.

[9] J. Crane, P. Dunn, P. H. Malyak, and B. J. Thompson. Particulate velocity and size measurements using holographic and optical processing methods. In *SPIE Proceedings*, volume 348, pages 634–642, 1982. Contained in [20].

[10] Gerald C. Holst. *CCD Arrays. Cameras and Displays.* JCD Publishing, 1996.

[11] G. B. Parrent Jr. and B. J. Thompson. On the fraunhofer (far field) diffraction patterns of opaque and transparent objects with coherent background. *Optica Acta*, 11(3):183–193, July 1964. Contained in [20].

[12] Levent Onural. *Digital Decoding of In-Line Holograms.* PhD thesis, State University of New Yourk at Buffalo, February 1985.

[13] Bill Spitzak and Michael Sweet. FLTK: The Fast Light Toolkit. `http://www.fltk.org/`.

[14] Matteo Frigo and Steven G. Johnson. FFTW: The Fastest Fourier Transform in the West. `http://theory.lcs.mit.edu/~fftw/`.

[15] Silicon Graphics Inc. The OpenGL standard. `http://www.opengl.org`.

[16] Brian Paul. The Mesa 3-d Graphics Library. `http://www.mesa3d.org`.

[17] Levent Onural and Peter D. Scott. Digitial decoding of in-line holograms. *Optical Engineering*, 26(11):1124–1132, November 1987. Contained in [20].

[18] Jae S. Lim. *Two Dimensional Signal and Image Processing.* Signal Processing Series. Prentice Hall, 1990.

[19] Winston E. Kock. *Laser and Holography: An Introduction to Coherent Optics.* Dover, 2 edition, 1981.

[20] Chandra S. Vikram, editor. *Selected Papers on Holographic Particle Diagnostics.* Number 21 in SPIE Milestone Series. The Society of Photo-optical Instrumentation Engineers Optical Engineering (SPIE) Press, Bellingham, Washington, 1990.

[21] B. J. Thompson. Diffraction by opaque and transparent particles. *SPIE Journal*, 2(43-46), December 1963. Contained in [20].

[22] R. E. Brooks, L. O. Heflinger, R. F. Wuerker, and R. A. Briones. Holographic photography of high-speed phenomena with conventional and q-swirched ruby lasers. *Applied Physics Letters*, 7(4):92–94, August 1965. Contained in [20].

[23] Brian J. Thompson. Holographic methods for particle size and velocity measurement - recent advances. *SPIE Proceedings*, 1136:308–326, 1989. Contained in [20].

[24] C. Knox. Holographic microscopy as a technique for recording dynamic microscopic subjects. *Science*, 155(3739):989–990, 1966. Contained in [20].

[25] James A. Liburdy. 1987. *Applied Optics*, 26(19):4250–4255, October 1987. Contained in [20].