Development of an Optoelectronic Holographic Platform for Otolaryngology Applications

Ellery Harrington, Worcester Polytechnic Institute

December, 2009

APPROVED:

_____

Professor Michael Gennert, CS Thesis Advisor and Head of Department

_____

Professor Cosme Furlong, ME Thesis Advisor

_____

Professor Matt Ward, Thesis Reader

By

Center for Holographic Studies and Laser micro-mechaTronics

Mechanical Engineering Department

Worcester Polytechnic Institute

Worcester, MA 01609-2280

Abstract

In this thesis work, we develop an optoelectronic holographic platform to facilitate otologists' ability to quantitatively study and diagnose disorders of the tympanic membrane (TM) and middle ear of humans in full-field-of-view. The holographic platform consists of a laser delivery system, a handheld interferometer, and corresponding software, which allow nanometer scale 3D measurements of deformations of the TM.

The holographic platform will be deployed and used in a medical research environment with future developments leading to the use of the platform in the clinic.

For more information about the software described in this thesis, please contact:

Professor Cosme Furlong, WPI-ME/CHSLT, cfurlong@wpi.edu.

Table of Contents

Table of Figures

1. Introduction & Background

1.1. Purpose of Project

The tympanic membrane (TM), also known as the eardrum, is part of the middle-ear which serves the function of transmitting sound to the cochlea in the inner ear [Rosowski, 1996].

Most existing studies of the TM are limited to measurements on a single point or a series of points [Huber et al, 2001; Whittemore et al, 2004]. While this is a useful approach for research and diagnoses, having a large field of view is much more useful for the diagnosis of TM and middle-ear disorders, and for determining the effectiveness of surgical procedures. Being full-field-of-view in nature, optoelectronic holography can provide doctors with important information about the TM, both experimental and clinical, that is not obtainable through other methods [Hulli, 2008].

1.2. History of Project & Acknowledgements

This project started in 2007 with the intent of performing research on human tympanic membranes and developing a clinical instrument. With a team of mechanical engineering students, Ellery served as the sole computer science student on the project starting in early 2007, to provide a software package which allows for communication to all necessary hardware and the ability to perform all necessary tasks in software.

In developing the system, Ellery worked closely with fellow M.S. candidates Nesim Hulli and Nikhil Bapat, post-doctoral students Maria Hernandez-Montes and Jorge Mauricio Flores, the project's advisor Professor Cosme Furlong, and several undergraduate students who all contributed greatly to the project. Dr. John Rosowski is serving as the project manager and is performing experiments along with his Mass. Eye and Ear Infirmary (MEEI) coworkers including Jeffrey (Tao)

Cheng, Michael Ravicz, Antii Aamisalo, and Saumil Merchant [Rosowski, et al., 2009; Hulli, 2008; Balboa, Dobrev, Fossett, 2008; Dwyer, Maccaferri, Wester, 2008; Karasic, Largesse, Lincoln, 2008; Furlong, et al, 2008].

The other WPI students who contributed to this project are all mechanical engineering students, while the MEEI employees have a medical or hearing & speech background. As a computer science student, Ellery assisted in some developments of optics, but primarily developed the software and integrated hardware components related to it.

1.3. Middle Ear

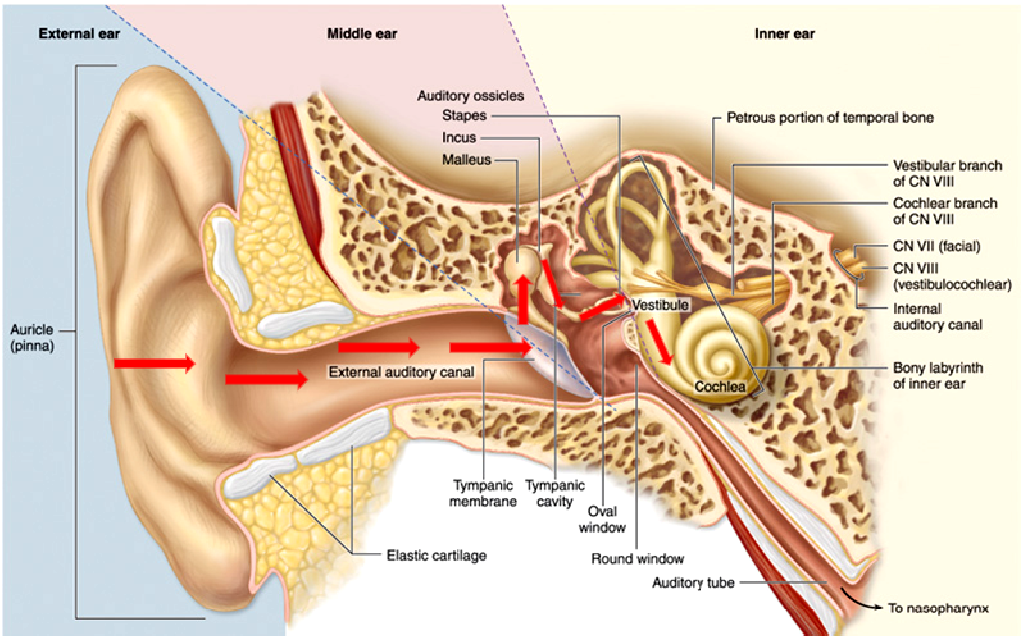


Fig. 1. Anatomy of the human ear: The cross section of human ear showing divisions of the outer, middle, and inner ears [Wikipedia, 2008].

As shown in Fig. 1, the human ear is divided into three sections: outer, middle and inner. The middle ear consists of the TM (or eardrum), several muscles and three bones. It is responsible for transmitting energy from airborne compressional waves to the cochlea [Rosowski, et al., 2009].

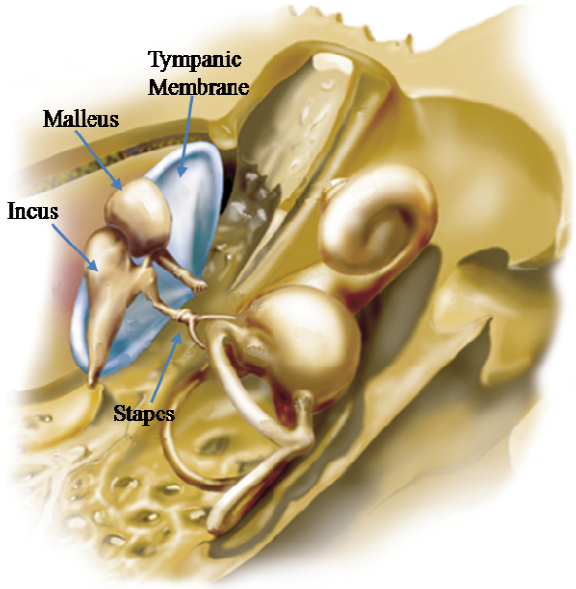
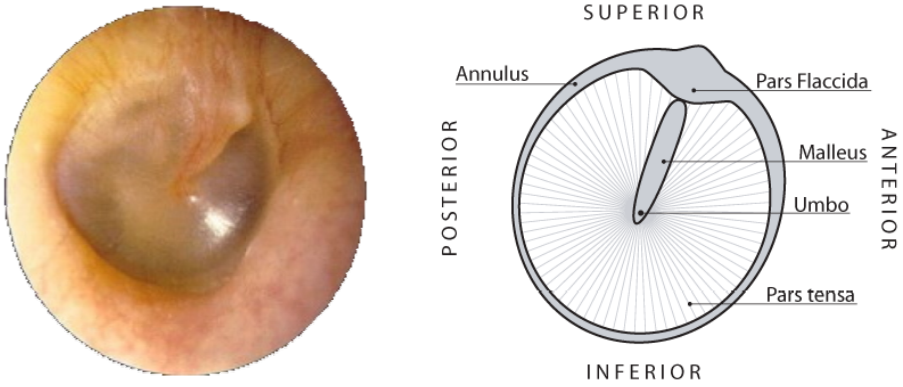


Fig. 2. The middle ear [Csillag, A., 2005].



(a) (b)

Fig. 3. (a) normal lateral surface of a tympanic membrane as seen through an otoscope [Grundman et al., 2008]; (b) schematic of the lateral surface of a right tympanic membrane [Sundberg, 2008].

The middle ear and tympanic membrane are shown in Fig. 2 and 3, respectively. As seen from the ear canal the TM is a concave tent-like structure, where the umbo, located near the center of the TM, is the most depressed part. The malleus is attached to the inner surface, and transmits sound to the incus.

The middle ear can be a point of failure for humans. When surgery is to be performed on the TM, diagnosis is normally performed with visual inspection from an otoscope. This is a subjective process that could use improvement, as it doesn't provide quantitative information about the TM's reaction to sound, or its ability to behave normally. To better study and diagnose the TM, nanometer-scale measurements in full-field-of-view are necessary. The goal of this project is to make this tool available for both experimentation and clinical use.

The TM's structure is shown in Fig. 3. It is a layered membrane, consisting of the lateral (outermost), lamina propria (middle), and medial (innermost) layers [Sanna et al., 2003]. The exterior edge, the annulus, is made of a thicker cartilaginous tissue to secure the TM to the auditory canal [Sundberg, 2008]. The TM ranges in thickness between 55 to 140 micrometers [Kuypers et al., 2006].

Since a healthy TM is necessary for good hearing function, the assessment of functional disorders of the TM is a critical part of the differential diagnosis of hearing disorders.

1.4. Holographic Laser Interferometry

Holographic laser interferometry enables the measurement of nanometer-scale deformations of an object, with precision to a fraction of the laser beam wavelength. The object, having nanometer-scale vibrations or movements, is then observed with the laser beam and

photographic equipment, giving a view of the object's movement or vibration when the imagery is taken.

Originally, holographic laser interferometry was accomplished using holographic plates as a recording medium, making laser holography an expensive and time-consuming process.

In the 1980's, a new method was developed to use beam phase stepping with a camera and digital storage [Stetson, Brohinsky, 1985; Stetson, Brohinsky, 1987]. This process has been refined and further developed as improving technology allowed for it, giving the ability for live processing and display of the reconstructed hologram.

1.5. The Optoelectronic Otoscope

Our optoelectronic holographic otoscope (OEHO) platform consists of three major components: a laser delivery system, an otoscope interferometer with imaging optics for sample observation (optical head), and computer with software controls [Furlong, et al., 2008]. The entire layout is shown in **Fig. 4**.

The laser delivery system consists of a solid state laser, with a fiber-optic subsystem for splitting of the laser beam into an object and reference beam, and amplitude and phase modulation of the beams. These components are labeled in Fig. 5.

The two beams from the delivery system are coupled into the optical head setup, which uses the object beam to illuminate the object and feeds the reference beam directly into the camera. A speaker positioned in closely to the object of interest is used to excite the object. These beams are then recombined using a beam splitter, resulting in a combined image observed by the camera. The components of the optical head are labeled in Fig. 6.

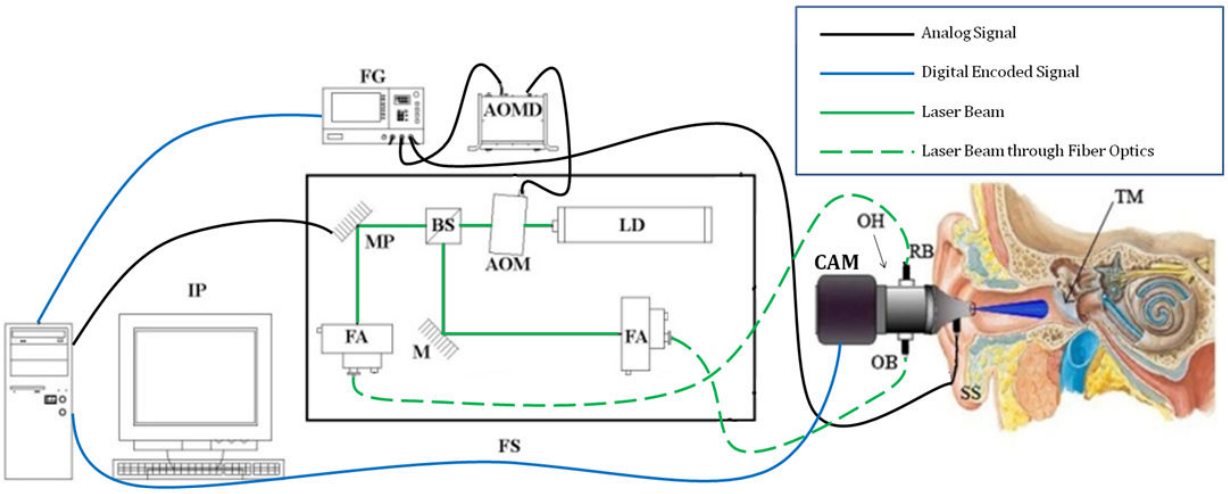


Fig. 4. Schematic of the OEHO system.

TM: Tympanic Membrane under investigation; OH: otoscope head containing imaging optics; CAM: digital camera; FS: fiber optic subsystem containing a laser (LD), accousto-optic modulator (AOM), beam splitter cube (BS), piezoelectric modulator (MP), mirror / MP operating as a mirror (M), and laser-to-fiber components (FA) to provide an object beam (OB) and reference beam (RB); OH and FS are controlled by the image processing computer (IP). SS is the integrated sound source. FG is the frequency generator which generates sound for stimulus at the sound source (SS) and provides the timing input to the AOM driver (AOMD), which acts as a stroboscopic switch [Furlong et al., 2008].

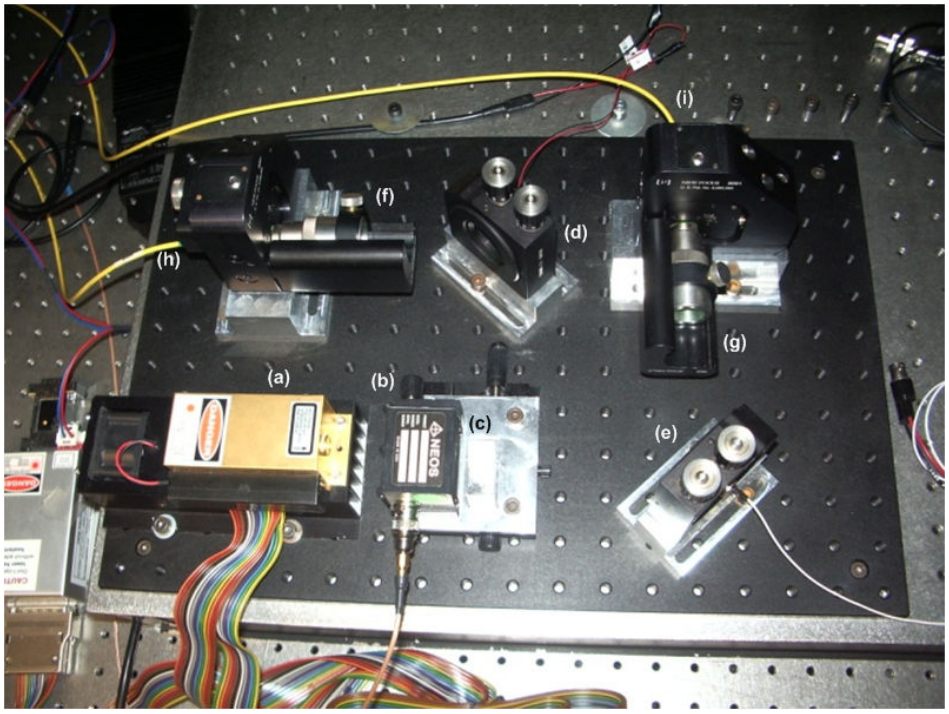


Fig. 5. The laser delivery system consists of (a) a solid state laser, (b) an acousto-optic modulator for use with stroboscopic mode, (c) a beam splitter cube, (d) and (e) two piezos, (d) used for phase-shifting the laser beam, and (f) and (g) laser-to-fiber couplers, (h) and (i) fiber-optic cables.

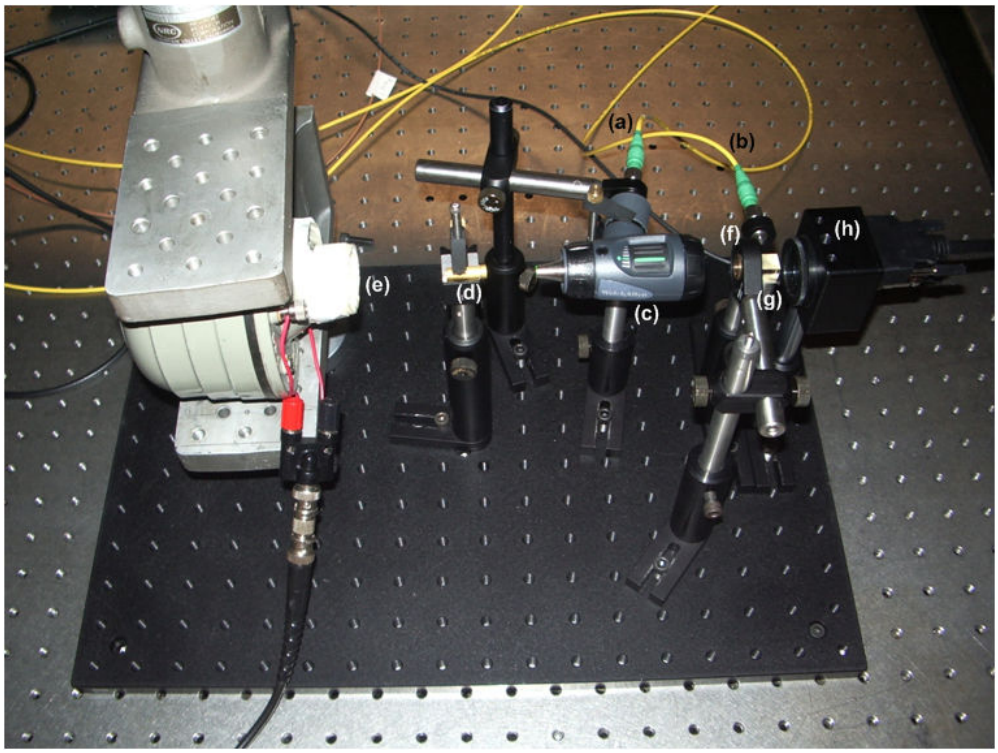


Fig. 6. The optical head consists of (a) and (b) fiber-optic cables, (c) Welch Allyn Otoscope with optical fibers and adjustable lens, (d) sample: here, a latex membrane, (e) speaker, (f) lens, (g) beam combiner, (h) Silicon Imaging camera.

The software, called LaserView, to control this hardware and run the necessary algorithms is written in Visual C++. LaserView is designed to interface with a selection of high-speed, high-resolution cameras [Hulli et al, 2007]. In order to create a holographic image, the software uses a data acquisition board to output a time-varying analog voltage, synchronized with each frame captured from the camera. This output is sent to a nanometer-resolution piezoelectric device (the piezo), which modulates the phase of the laser beam. The software captures frames from the camera, at specific laser phase states, and then stores them in memory for processing. The software displays processed images in two distinct modes: time-averaged for characterization of dynamic events, and double-exposure for detection of 3D motions that correspond to changes between two different states of deformation of the object of interest [Pryputniewicz, 1996].

2. Motivation & Related Work

2.1. Previous studies: Tonndorf and Khanna

Analog holographic time-averaged imagery has been performed in great detail on the tympanic membranes of post-mortem specimens by Tonndorf and Khanna starting several decades ago [Tonndorf J, Khanna, S.M., 1972]. These studies produced data on the behavior of tympanic membranes under stimulation using time-average techniques on film, rather than digitally as the techniques that we use were not yet possible. While very successful, the results of the experiments were limited due to the complexity and time requirements. Having the capability to perform the experiments digitally gives much more flexibility to the operators, allowing for such features as live display and stroboscopy.

Continuing similar experiments using new techniques, we are able to perform research on the tympanic membrane which had not been performed; something that is needed by the medical community for both research and diagnosis.

3. Requirements

3.1. Resolution

A human eardrum has the approximate diameter of 10 mm, requiring that the minimum field of view for the system be near 10mm × 10 mm. The resolution and size of the image determines the spatial sensitivity of the measurement system. While a large image size is most desirable, an optimal size must be used as to keep live image processing as efficient as possible. Live image processing is dependent on running an algorithm on every pixel in a series of acquired images. Based on a series of experiments with various samples (synthetic materials, animal TMs), an image size of 800 × 800 pixels was found to give high quality results which would not greatly

improve with increasing the resolution, while keeping the image size small enough for fast live image processing. While the actual field of view varies based on the sample being view, Using a field of view of 10mm × 10 mm the spatial resolution is 12.5 nm per pixel. This resolution

Using cameras with a pixel size of 6.7μm × 6.7μm and a lens to focus on the sample, the optimal configuration was determined through a series of experiments which tested image quality, and has a resolution of 57.0 line pairs/mm, a depth of field of 5 mm, and a field of view of 10 × 10 mm^2 [Hernández-Montes, 2008; Karasic, Largesse, Lincoln, 2008].

3.2. Quality of Imagery

The quality of the imagery recorded by the computer is dependent on several factors including signal-to-noise ratio of the imagery, ratio of the object and reference beams, and the alignment of the optics and laser beams. To maximize the quality, adjustments are performed to change the alignment of the two laser beams, otoscope, and other optics, as the position of the object.

3.3. Speeds of Acquisition

The desired speed of acquisition, the frame-rate, varies depending on the type of experiment being performed. This speed usually falls between 20 – 40 frames per second. This speed can be limited by numerous factors, including the camera's capabilities, shutter speed, transfer speed, synchronization with the piezo, and computer processing power for transferring, processing and displaying images.

The speed of acquisition is a major factor when using samples which are moving. A live subject, human or otherwise, may have movements in the inner ear area of interest due to such factors as muscle movement, heartbeat, or respiratory system. The subject's movements add a

restraint on the amount of time which four sequential images may be taken to produce useful digital holograms; movements of the entire body or ear area can dwarf the nanometer-scale movements of the tympanic membrane, rendering the smaller movements undetectable. A faster speed or acquisition can help to eliminate this problem.

3.4. Modes of Operation

3.4.1. Time-Averaged

Time-Averaged mode allows for the viewing of the near-instantaneous movement of an object. This is accomplished using a time varying fringe-locus function, $\Omega_t(x, y, t)$ related to the vibrating object [Pryputniewicz, 1985, 1987, 1989].

The intensity distribution of the interferogram sampled by the camera is

$$I_t(x, y, t) = I_o(x, y) + I_r(x, y) + \qquad (1)$$

$$2A_o(x, y)A_r(x, y)cos[\Delta\phi(x, y) + \Omega_t(x, y, t) + \Delta\theta_n].$$

where I_o is the temporally constant Intensity of the reflected object beam, I_r is the temporally constant intensity of the reference beam, A_o and A_r are amplitudes of the object and reference beam, $\Delta\phi$ is the phase difference between the object and reference beams when the object is at rest, Ω_t is the change in the phase difference produced by motion of the object and $\Delta\theta_n$ is the change in the phase difference due to the controlled alterations in the optical path length of one of the beams.

Because each frame is taken over a period of time, dependant on the camera's shutter speed, the intensity is averaged over a period, Δt. Therefore, the observed intensity is:

$$I(x, y) = \frac{1}{\Delta t}\int_t^{t+\Delta t} I_t(x, y, t)dt \qquad (2)$$

Images are recorded at four separate imposed path length differences ($\Delta\phi1 = 0$, $\Delta\phi2=0.25$, $\Delta\phi3=0.5$ or $\Delta\phi4=0.75$ cycles). With the integration over the recording time necessary for each image:

$$I_{t_n}(x,y) = I_o(x,y) + I_r(x,y) + 2A_o(x,y)A_r(x,y)cos[\Delta\phi(x,y) + \Delta\theta_n]M[\Omega_t(x,y)], \quad (3)$$

Where $M[\Omega_t(x,y)]$ is the characteristic function that modulates the interference of the two fields due to the motion of the object. $I_o(x,y)$ and $I_r(x,y)$ are intensities of object and reference beams, $A_o(x,y)$ and $A_r(x,y)$ are amplitudes of object and reference beams.

To implement this in software, images are continuously acquired in succession at a constant frame rate and exposure time, synchronized with the piezo for changing the phase of each image. With each new image, an algorithm is run on the previously acquired four images, taking into account the phases of each image, which correspond to the voltage sent to the piezo. The resulting image is then displayed on screen.

3.4.2. Double Exposure

Double-exposure holography is based on the principle of extracting data from the interference pattern formed by recombining object and reference beams. I_n, the combination of both beams where F_o and F_r refer to the complex light fields of the object and reference beams, respectively, is defined as:

$$I_n = (F_o + F_r)(F_o + F_r)^* = |A_o|^2 + |A_r|^2 + 2A_oA_rcos[(\phi_o - \phi_r) + \Delta\theta_n] \quad (4)$$

where A_o and A_r are the amplitudes of the object and reference beams, respectively, ϕ_o and ϕ_r are the phases of the object and reference beam, respectively, and $\Delta\theta$ is the phase step introduced at each frame by the piezo [Stetson, Brohinsky, 1985; Creath, 1985].

By introducing the fringe-locus function Ω, which contains information about the shape or deformation of the sample of interest, I_n' can be defined as [Creath, 1985]:

$$I_n' = I_o + I_r + 2A_oA_rcos[\Delta\phi + \Omega + \Delta\theta_n] \quad (5)$$

To solve for Ω, a system of equations is used: *D*, a cosinusoidal image, and *N*, a sinusoidal image [Creath, 1985]:

$$D = 64{A_o}^2{A_r}^2\cos(\Omega) \quad (6)$$

$$N = 64{A_o}^2{A_r}^2\sin(\Omega) \quad (7)$$

These are combined to produce the displayed double exposure image [Stetson, Brohinsky, 1985]:

$$\Omega = \tan^{-1}\left(\frac{N}{D}\right) \quad (8)$$

Using this information, double exposure is performed using a set of four 'reference' images and four 'live' images. The reference images are gathered by an operator pressing the "new reference" button. The live images are the four most recent images from the camera, each at a different phase. Deformations are extracted from the interference pattern, which is generated by the combination of the object and reference beams [Furlong, C., and Pryputniewicz, R. J., 1998; Hulli et al., 2007; Pryputniewicz, 1996]. To perform this, the user selects the time of reference by pressing a button. The software stores the reference images in memory and continues acquiring new images. The resulting double-exposure image is generated by using

$$\mathrm{I} = \mathrm{ATAN2}((I_0 - I_2)(R_1 - R_3) - (I_1 - I_3)(R_0 - R_2), (I_0 - I_2)(R_0 - R_2) + (I_1 - I_3)(R_1 - R_3)) \quad (9)$$

where I_0 is the most recent image acquired, I_1 through I_3 are the three previously acquired images. R_0 through R_3 are reference images, each of which has the same phase value as its I counterpart: i.e. I_x and R_x have the same phase [Stetson, Brohinsky, 1987; Hernández-Montes, et al., 2008].

3.4.3. Stroboscopic Mode of Acquisition

The third operational mode is stroboscopic. In time-averaged and double exposure modes, the camera's shutter integrates over a period of time, usually 10 – 30 milliseconds. Using a shorter shutter speed will not produce effective imagery, as not enough light is available. Stroboscopic mode allows for using only a small portion (usually 5% - 15%) of the stimulus wavelength. To do this, a dual-channel signal generator is used with one channel set to a sine wave used for stimulating the object of interest. The second channel has the same frequency but uses a square wave to use an acousto-optic modulator (AOM) to enable and disable the laser beam by blocking its path. This generates the same effect as a strobe light, only showing the desired portion of the stimulus wave. The offset of the square wave can also be automatically adjusted to change the phase relative to the stimulus [Hernández-Montes, et al., 2008].

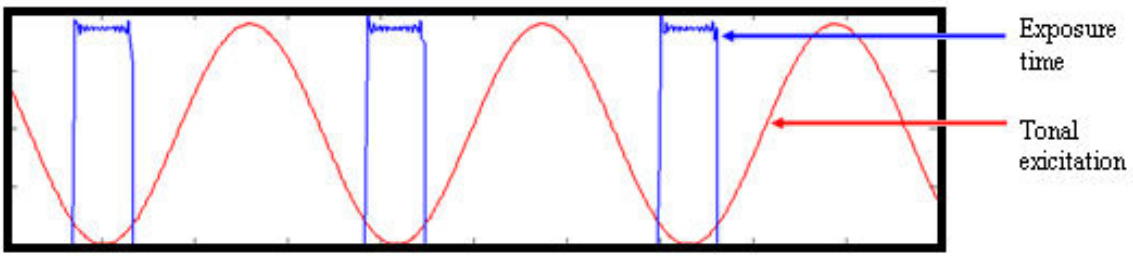


Fig. 7. Stroboscopic mode: synchronization of the illumination with the object excitation.

3.5. Phase Unwrapping

As explained above, using the equations in double exposure operation mode, the software can produce a 2-dimensional phase map, showing the displacements of an object over the complete camera viewport. This phase map is wrapped: it has 2π discontinuities from the minimum value to the maximum value [Bushman, 1993].

To unwrap these images, a complex and processor-intensive algorithm is run on the wrapped phase image. The unwrapping algorithm currently in use on this project is written in the IDL programming language, along with several helper applications to read in RTI files, apply filters, unwrap the phase, and save the phase map and 3D models. These have been in used since before Ellery's involvement in the project. Fringe Automator is a piece of software written in IDL by Ellery for streamlining the unwrapping process: it allows for processing imagery taken from LaserView through several IDL utility programs (including the unwrapper) with little or no user intervention.

4. Design

4.1. Computer

In selecting the requirements for a computer to run LaserView, four major factors were considered: processor speed, memory, hard disk, and connectivity.

Since a major component of the software is live processing and display of video, a powerful processor is required to run LaserView effectively. Since the software uses multiple threads to perform tasks simultaneously, a dual-core or dual-processor system can be utilized for the best results. A processor speed of 3 GHz or higher is ideal so that processed imagery can be shown at a high framerate.

Memory usage in LaserView is relatively low during normal usage, and a significant amount of memory is not necessary to run the program; 100 MB is sufficient. However, during certain operations, when a large amount of data must be stored quickly, a larger amount of memory is required. This is the case when saving a series of images in quick succession: in order to prevent losing synchronization or slowing acquisition, writing to disk is avoided and therefore captured images are stored in memory to be written to disk later. Depending on the nature of the images to be captured, a larger amount of memory is recommended. Assuming 1.2 MB per image, 2 GB of RAM will allow for storage of about 1700 images, which will be sufficient for most purposes. Adding in other memory usage requirements (1 GB in total for normal operating system memory usage and LaserView operation), systems requiring rapid acquisition should have at least 3 GB of memory.

A large amount of hard disk space is required when performing certain experiments. Since a single capture in reference mode can save as many as 9 images, which when saved uncompressed at 1.2 MB per image, is approximately 10 MB written in a single capture. A large hard drive is utilized, in conjunction with a DVD writer for backup and archival purposes.

Camera connectivity can be accomplished with several interfaces: IEEE 1394 (FireWire) and Camera Link are two of the most common used in industry. To allow for a variety of camera testing and future expansion, computers with the capability for both interfaces are used. IEEE 1394 is a standard communication protocol for many devices and is standard on many hardware devices. Camera Link is a communication protocol used for only machine vision, and normally uses a PCI Camera Link capture card to interface with the camera. Computers purchased for the project have both PCI-X and PCI-e (PCI Extended and PCI Express, respectively) to support a variety of Camera Link frame grabbers.

A sample configuration of some of the computers used is listed in appendix B.

4.2. Cameras

Cameras can be connected to a computer by a variety of methods: USB, IEEE 1394 (FireWire), Camera Link, gigE (1000BASE-T Ethernet), analog, and proprietary PCI or PCMCIA cards. For the OEHO system, we have tested IEEE 1394 and Camera Link connections. Camera Link has the key advantage of faster data rates, but IEEE 1394 is much more widely available, making it possible to deploy more easily on multiple computer systems without the need for extra hardware configuration.

Using Camera Link, we have run the OEHO system using EPIX brand PIXCI Camera Link frame grabber PCI cards in conjunction with Silicon Imaging brand cameras. LaserView is specifically configured to use the SI-1280M [Silicon Imaging, Inc., 2009], Silicon Imaging's 6.7µm × 6.7µm, 1280 × 1024 pixel monochrome camera, connected to an EPIX PIXCI frame grabber. The model of the frame grabber can differ since they use a shared application programming interface (API), but the only camera model explicitly supported is SI-1280M since commands for each model of camera differ greatly.

PixeLink cameras are also supported using IEEE 1390. Since PixeLink cameras share an API, PixeLink cameras are interchangeable. Development was done with the PL-A741 camera, which has the same pixel pitch and resolution as the Silicon Imaging camera: 6.7µm × 6.7µm per pixel, and a 1280 × 1024 resolution.

A third camera choice, QICam by QImaging, is also supported, but support in LaserView is buggy due to issues with the API causing hardware failures on some computers. This camera is not currently in use for any setups, but may be used in the future.

The bit depth of cameras is an extremely important component to the selection of cameras. Many cameras are only capable of 8-bit imagery (256 shades of gray) which is generally not

sufficient to generate high-quality holographic images. The cameras used are capable of 10-bit and 12-bit grayscale imagery which provides 1024 and 4096 shades of gray respectively.

Supported camera models and specifications are listed in appendix B.

4.3. Data Acquisition Card

National Instruments manufactures a large selection of DAQ cards, which allow for digital input and output as well as analog input and output. This is required for synchronizing the piezo, as well as several other optional features of the software. When a voltage is sent to the piezo, it changes the optical path length by a value relative to the voltage sent.

The resolution of analog output and input are a critical part of selecting a DAQ card. The varying voltage output to the piezo is reliant on the precision of the DAQ card. Early in the project, the system used 12-bit analog input and output processing with a range of 20 volts. Since input and output ranges are +- 10v, 12-bit DAQ cards give a resolution of 4.88 mV. This is sufficient for running the piezo, but newer generations of DAQ cards provide 16-bit input and output. This gives the much higher resolution of 0.305 mV input and output.

The DAQs used in this project are PCI and PCI-express, installed in the computer running LaserView. The card must be connected to a screw terminal connector block for connection to each device which uses analog or digital communication through the DAQ card: piezo, shutters, and analog trigger. The numbered ports in the screw terminal connector block correspond with the line numbers given in the reference manual for the specific DAQ card being used. The input/output line channel numbers are configured in settings.h.

Supported DAQs are listed in appendix B.

4.4. Function Generator

For use of stroboscopic mode, a function generator is required to output two synchronized signals. This function generator, the Tektronix AFG3102 [Tektronix, Inc., 2009], is connected though USB via National Instruments' VISA virtual interface. The connection to the device is managed through the VISA interactive control, which gives the connection ID set in LaserView's settings.h to establish the connection.

4.5. Laser

The wavelength of a laser is indicative of the color it projects onto the object of interest. Since the human TM is semi-transparent, the color is a very important component, as various colors will exhibit varying reflectivity and therefore different results of imagery as they are observed and recorded by the OEHO [Karasic, Largesse, Lincoln, 2008].

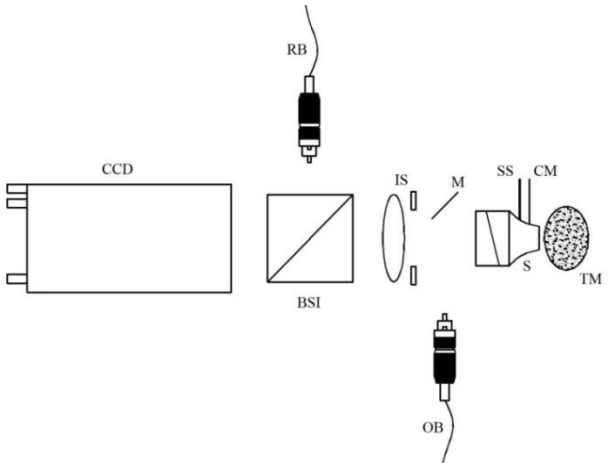
Experiments were performed to measure effectiveness of various laser wavelengths. Due to the high price of lasers, it is prohibitively expensive to perform a comprehensive test with varying lasers' frequencies. After testing several wavelengths, a 532 nm wavelength green laser is being used for experimental purposes.

4.6. Optic System

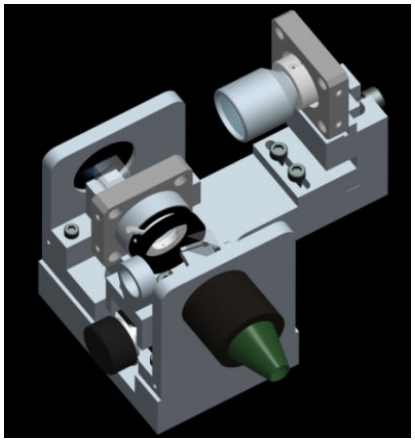
The entire basis for holographic interferometry is based on the use of optics; using the proper beam splitters, fiber optics, and lenses are all critical to the quality of results. Fourteen lenses were tested for their varying focal lengths, resulting in the use of the THORLABS AC127-030-A1 achromatic lens, which has a diameter of 12.7 mm and focal length of 30 mm. Several revisions

of an otoscope head subsystem were created [Dwywer, Maccaferri. Wester, 2008; Karasic, Largesse, Lincoln, 2008].

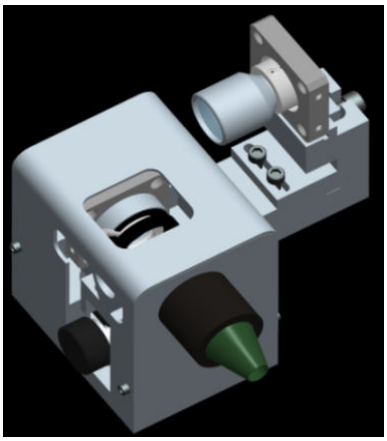
(a)



(b)



(c)



(d)

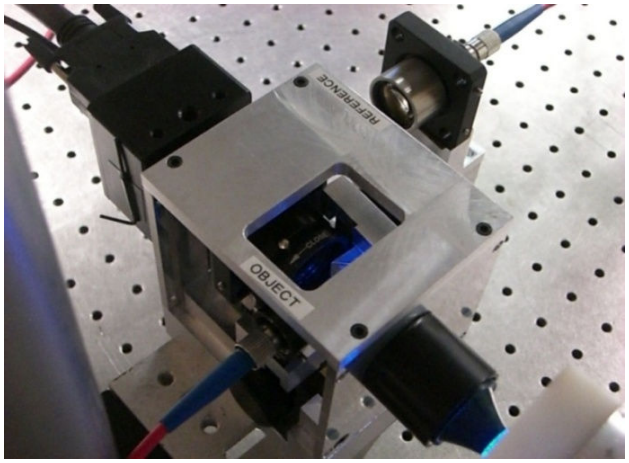


Fig. 8. Optical Head: (a) schematic model depicting the major components; (b,c) CAD model depicting the major components; and (d) fabricated subsystem. [Hulli, 2008]

4.7. Software Architecture

LaserView is developed using simple Visual C++ architecture, where events and basic logic are handled using functions in the application windows' classes, and more intensive operations are handled by task- or hardware-specific classes, run as concurrent threads in some cases when concurrent operation is needed.

5. Implementation & Development

5.1. Constructing the OEHO

Ellery assisted the mechanical engineers on the project in developing and constructing a fully-functional optoelectronic holographic otoscope system. This involved the machining of several custom parts, as well as the acquisition and testing of numerous optical and electronic devices.

5.2. Software Development

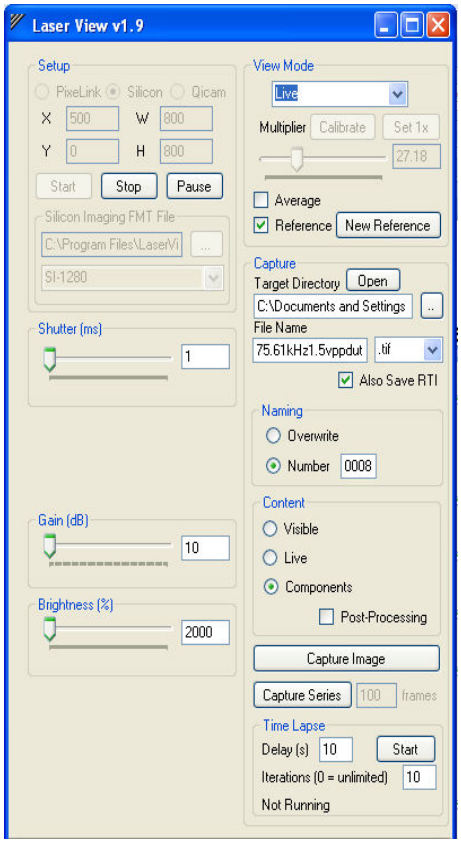


Fig. 9. The LaserView control window.

5.2.1. Platform

With a choice of platforms and programming languages, Visual C++ .NET was selected for its ability to perform well at fast image processing and display on Windows PCs, including low-level algorithm support. As much of Ellery's prior programming experience has been in C and C++, Visual C++ was the most natural choice.

5.2.2. Architecture, Classes and Structure

The LaserView program is divided into a series of classes. The core of the application is in the ControlWindow class, which houses the main user interface and allows for configuration of

hardware. ImageWindow is also a major part of the software's architecture, handling storage of images from the camera, processing, and display of the live image.

All communication with cameras is handled through the abstract CameraControl class' subclasses, one for each brand of camera: SiliconImagingCameraControl, PixelLinkCameraControl, and QicamCameraControl. CameraControl contains the declaration for several abstract functions which are called on regardless of which camera is in use. Most notable are: connect (establish a connection to the camera), disconnect (close the connection to the camera), configure (set up the camera in a ready state), startStream (start acquisition stream), stopStream (stop acquisition stream), setOption (change a parameter on the camera such are frame rate or shutter speed), getValue (get a parameter's current value), and getImage (get a newly acquired image from the camera). This structure allows for additional cameras to be added in the future, by further subclassing CameraControl.

National Instruments data acquisition cards communications are supported through the use of the NICard6733 class, which allows for 2-way communication between the computer and National Instruments DAQs utilizing the DAQmx API. This is used by other classes to control analog and digital output and input.

The TekAFG3102 class allows for communication through National Instruments' VISA protocol to a Tektronix AFG3102 (and other similar models) signal generator. Stroboscopic mode is currently the only place where this is utilized currently.

The additional tools which allow for monitoring, measurement, and analysis are each self-contained in a Windows Form class, allowing for use by the controlWindow.

The interaction between classes is shown in Fig. 10.

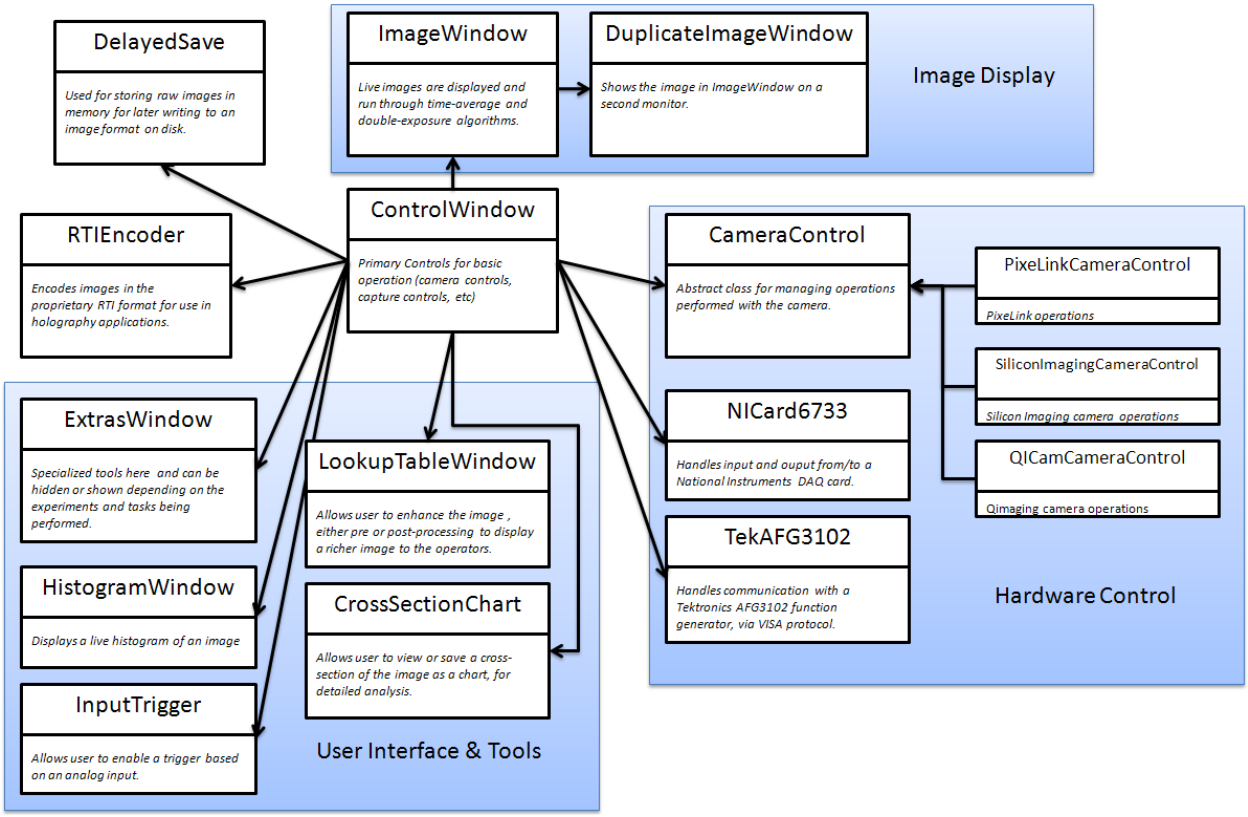


Fig. 10. Class communication structure of LaserView

5.2.3. Multi-Hardware Capabilities

Due to varying requirements, availability of hardware, and configurations, LaserView can be compiled with or without various hardware supported. With the ability to use three brands of cameras, some computer setups may have different combinations of cameras drivers installed and available. Automatic shutters can be installed on the holographic subsystem to achieve automatic beam-ratio calculation. A data acquisition card may optionally be present. Changes to the availability of hardware are made at compile time in the settings.h file, and can also be automated to build multiple configurations of the software simultaneously.

Although configuration at runtime is desirable, some of the utilized hardware requires that certain libraries be present in order to run the software, even when not utilized. Using compile-

time directives allows for multiple configurations utilizing various hardware and settings depending on the computer and external hardware available.

5.2.4. Camera Selection and Initialization

Because there is a choice of three camera brands to use, LaserView allows for the selection of the camera on startup. When a camera brand is selected, LaserView instantiates the 'cam' CameraControl object with an instance of the class which corresponds to the camera requested.

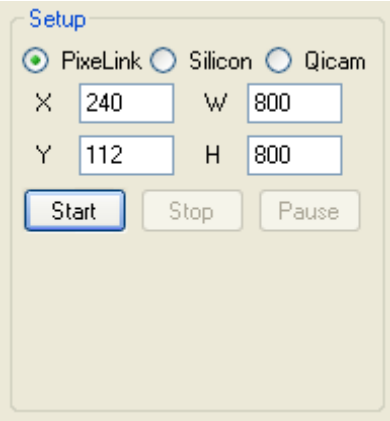


Fig. 11. The camera setup dialog.

Here, the code run when the PixeLink radio button is clicked disconnects from the current camera object if it is connected, then initiates a new connection to the PixeLink camera.

```
if (radioButtonPixeLink->Checked){
     if (cam != NULL)
           cam->disconnect();
     cam = new PixeLinkCameraControl();
}
```

This cam object is then used for all operations which require interaction with the camera.

5.2.5. Synchronization

In order to acquire usable imagery, the camera must be synchronized with the piezo. If there is any part of the camera exposure where the piezo changes from one state to the next, that frame is invalid and cannot be used to create a useful holographic image.

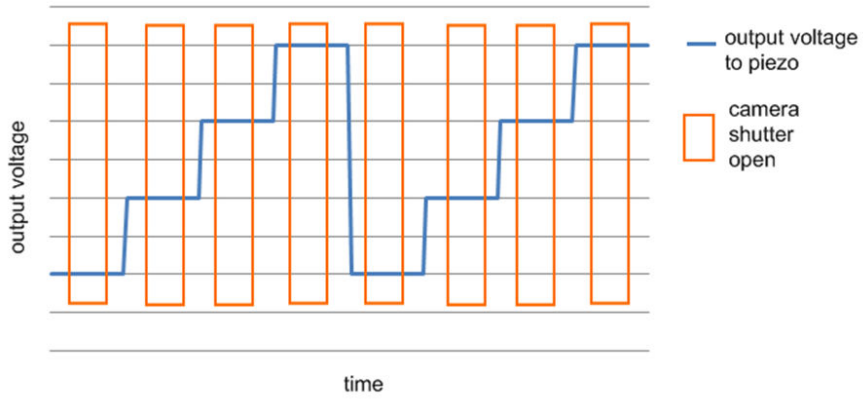


Fig. 12. Synchronization of camera and piezo.

In Fig. 12, the blue stepping function represents the ideal voltage value sent to the piezo (actual voltage has some variance). The orange boxes represent the time periods where the camera lens is open to light and integration takes place.

To manage this synchronization, the CameraControl object can register an event handle. The event handle is then signaled each time an image is captured, which signals another process to perform a synchronized action, such as changing the shifter output voltage.

5.2.6. Image Processing Algorithms

Parts of the development of the image processing algorithms are based on previous works in digital interferometry. Most notably, "BladeApp," which was developed as a tool for machine vision on razor blades, contributed portions of the algorithms for implementation of modulation

and optical phase view modes. The image processing algorithms were written based on both the equations presented earlier (time-average and double-exposure) and the previous work in BladeApp.

Two primary considerations in writing the routines to process the imagery are speed and accuracy. Each pixel requires computation of pixels from a series images using a square root or arctangent, and must be calculated individually. For example, one of the commonly used image sizes on this project, 800 × 800 pixels, requires 640,000 pixels to be individually calculated several times per second to allow the display of a live fluid image. Years ago, this was possible by using a preprogrammed graphics pipeline. With improvements in processor speed, these calculations have become feasible for real-time display in software.

To improve performance, the square root function is re-implemented by using a lookup table: each possible input integer (2^{10} or 2^{12} possible values, depending on the camera's bit depth) is pre-calculated and stored in an array. This method uses a relatively small amount of memory (up to 16KB) to provide constant-time access to square roots.

5.3. Primary Features

5.3.1. Camera Controls

To acquire high-quality imagery, certain camera parameters must be adjusted. This includes frame rate and shutter speed, as well as modifiers such as brightness, contrast, and gain. As these change with the way the experiments are done, they may be adjusted during an experiment.

Implementation is handled differently for each camera model, as the APIs require different communication methods. The PixeLink and QImaging APIs provide a library of functions to adjust

the camera's settings, but the Silicon Imaging requires generating raw commands to be sent to the camera.

5.3.2. Image Acquisition and Processing

For storage of imagery, the *RawImage* is a typedef of a pointer to the *short int* primitive, allowing storage up images of arbitrary dimensions with up to 16 bits per pixel. The raw image is allocated using *malloc*, with a size of *imageWidth* * *imageHeight* * 2 bytes.

LaserView works by repeatedly acquiring imagery from the camera, storing it, while simultaneously processing it and displaying it to the user. Several threads are run simultaneously to perform these tasks.

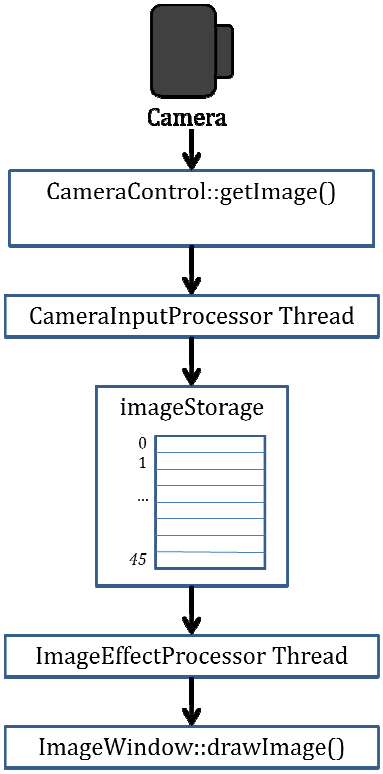


Fig. 13. Flow diagram of image acquisition, processing, and display.

This is accomplished using an array of type *RawImage*, and size *IMAGE_STORAGE_SIZE* (currently defined as 45 images). More images are saved in memory than necessary to avoid the possibility of overrunning the array when processing is slow. This array, referred to as *imageStorage* is allocated once based on the width and height of images being acquired and accessed circularly (the image at position 0 follows the image at position IMAGE_STORAGE_SIZE – 1). The *ImageWindow* class keeps track of the current position in *imageStorage*, giving the ability to use the most recently acquired images in the *ImageEffectProcessor* thread. These sets of imagery can then be processed by the *imageEffectsProcessor* thread, depending on the view mode that is selected.

5.3.3. View Modes

View modes enable the operator to switch between a selection of views to give various ways to view the imagery.

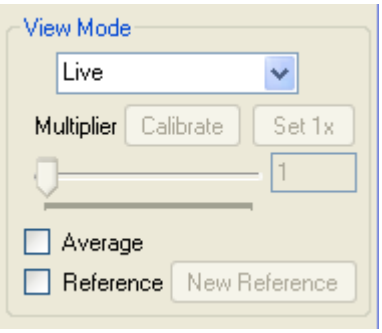


Fig. 14. View mode dialog.

Live: Live mode is simply the image stream from the camera displayed. When an image is captured from the camera, it is stored in the array of images and the position is updated. The image window continuously monitors this and updates the display with the most recently acquired image. Since there are no calculations done, very little CPU time is needed to display this imagery. This mode is used for setting up the system, making sure that beam alignments are done properly and that the object is in focus.

Modulation: Modulation mode, used for time-average, shows the near-instantaneous changes in phase on the image. This is useful for viewing an image's vibration when a stimulus is applied. When in reference mode, the image is compared against most recently stored reference images, and shows fringes of constant deformation related to the change between the current state and the reference state, similar to the lines of a contour map. A section of C++ code showing the processing for modulation mode is available in Appendix A.

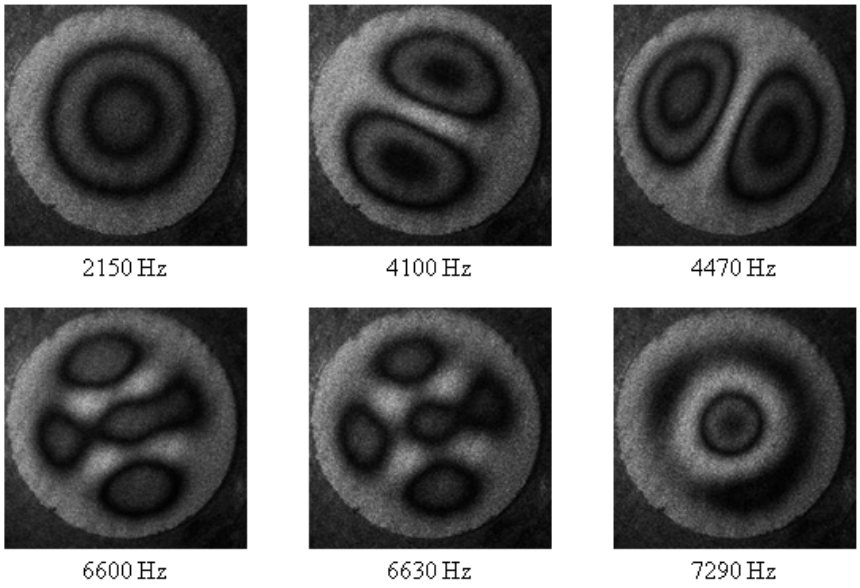


Fig. 15. Time-averaged interferograms of a copper plate stimulated at varying frequencies [Hulli, 2008].

Fig. 15 gives a best-case scenario for the time-averaged mode: the copper plate is rigid and reflects light much better than translucent TMs.

Optical Phase: Optical Phase view mode is used for viewing a wrapped fringe pattern on an object. It can be used for double exposure mode. The wrapped phase imagery can then be unwrapped, giving a displacement map of the object of interest.

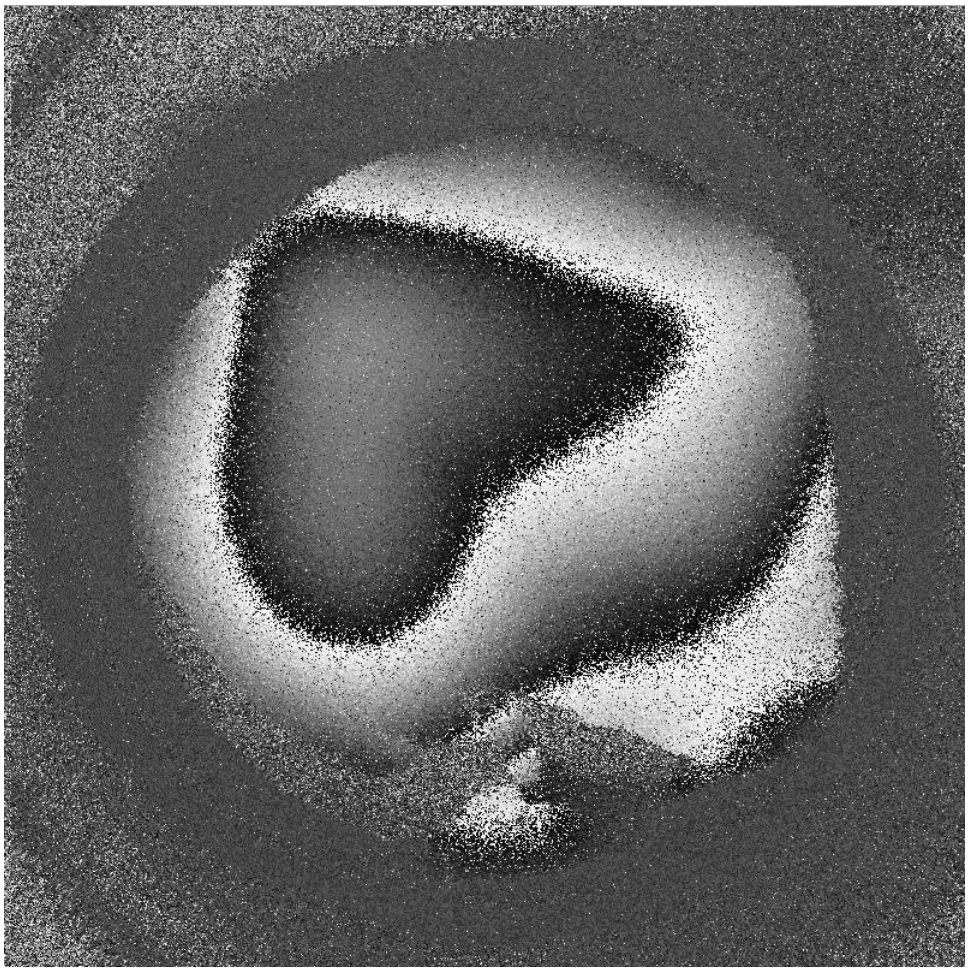


Fig. 16. An optical phase image of a piece of latex material, used for testing purposes.

Normally, when optical phase mode is used in reference mode, the absence of change is shown as black (value 0). Because the phase wraps from black to white, this can generate patterns of white and black pixels next to each other, which can be distracting to the operator and make distinguishing small color changes more difficult. To combat this, optical phase mode can also be displayed with a 180° offset, so that no change is indicated by the mid-range gray. A code sample for optical phase mode is available in Appendix A.

Reference Images: Both modulation and optical phase modes support reference mode, where four reference images (at the four phases) are stored constantly in memory. The live phase-shifted images are then compared to the reference images. This gives the ability to take a reference at any time, and then view the change between that reference and the current state of the object.

Calibration: In modulation mode, the imagery generated is initially in floating point format, therefore having an arbitrary range. Each pixel is converted into an integer for display. By multiplying the full image by a scalar, the bit-depth is increased to better match the range of the camera's bit depth.

An optimal multiplier can be calculated and set automatically by using the calibrate function of LaserView.

Picture-In-Picture: For testing purposes, Picture-In-Picture gives the ability to view both modulation and live mode at the same time. This gives the ability to view the object as well as its motion so that changes can be made and their results can be observed immediately.

Picture-in-Picture is implemented by overlaying a sub-sampled image on a full image.

Image Averaging: Small uncontrolled motions of the specimen due to acoustic noise (e.g. from the air handler), temperature gradients, or uncontrolled mechanical vibrations can generate brief flickerings in the interferometric image, which can be reduced by averaging consecutive images.

5.3.4. Image Capture

Image Formats: Saving an image to disk can be done in several image formats, including JPEG, BMP, and TIFF. Because the images produced by the system may be used for various purposes after being saved, a wide variety of formats is supported.

The proprietary RTI format is also supported [Furlong, 2000]. This format facilitates storage of uncompressed 16-bit monochromatic images which can be read by other specialized interferometry software for later processing. In addition to the storage method of most other image formats which store pixels as integer values, RTI also allows for storage of the 'float' type, giving images the ability to store a bitmap using 32-bit single-precision floats, and giving an arbitrary range of values. This gives the ability to store displacements in any unit desired and high precision, rather than a fixed scale of integers.

A control in the software gives the option of saving an RTI and a second format. This gives the ability to save in JPEG format for qualitative analysis, as well as RTI format for unwrapping and analyzing further. JPEG is a lossy format but is useful for image distribution for qualitative observation.

Selection of Content to be Saved: LaserView gives the option to save three types of image: live, visible, or components. Live saves the current image straight from the camera, with no processing. Visible saves the processed image which is currently displayed in the image window, such as modulation or optical phase. Using the components option saves the image being shown on screen, as well as the component images which were used to create it. This allows further processing of the imagery, including unwrapping and 3-dimensional surface generation.

5.3.5. Series, Video, and Time Lapse Capture

To allow recording motion or a series of images, LaserView gives the option to record a series of images and store them in a video file or sequentially numbered image files. A series of images can be recorded and saved as separate image files, or as a single video file.

5.3.6. Dual Monitor Capabilities

Experimentation is usually done with at least two people: one operating the computer and one controlling the specimen and optical equipment. Depending on the layout of equipment, the software includes optional dual monitor support, showing the same image on a second monitor, if installed on the system. This lets the computer operator have complete control, while showing a view of the experiment to other people in the room.

To implement this, a new window is opened automatically and maximized on the area occupied by the secondary monitor. As each image is displayed in the main image window, it is copied to the duplicate image window and displayed. This window also supports a status text overlay to alert the operator to events that are occurring, such as changing view modes or capturing images.

5.4. Tools and Other Features

5.4.1. Shifter Voltage Selection and Calculation

Laser holography requires that a piezo be used to modulate the phase of the laser beam to create the effect which makes this process possible. This is done by sending a varying voltage to the piezo via the data acquisition card's analog out capabilities. The voltages are based on the wavelength of the laser and the sensitivity that relates the voltage drive to the piezo and the motion of the piezo. The variables can be entered into the voltage calculation tool to automatically update the voltages which are output.

5.4.2. Phase Calculation

Rather than using the voltage calculation tool, these voltages can be calculated experimentally instead, through use of the phase calculation tool. This is useful if certain information about the setup is unknown.

5.4.3. Lookup Tables

A lookup table transforms an image using a function: each pixel input value has an output value. LaserView implements a lookup table dialog box to make a user-defined lookup table function and apply it to the video stream. Both pre-processing (run on live video) and post-processing (run on processed video) are supported.

Lookup tables can be generated from a combination of linear, exponential, and other functions, giving full control of the video processing. Implementation is done by setting an array of integers the size of the image's bit depth as the lookup table is modified. This enables constant time access for transforming each of the image's pixels.

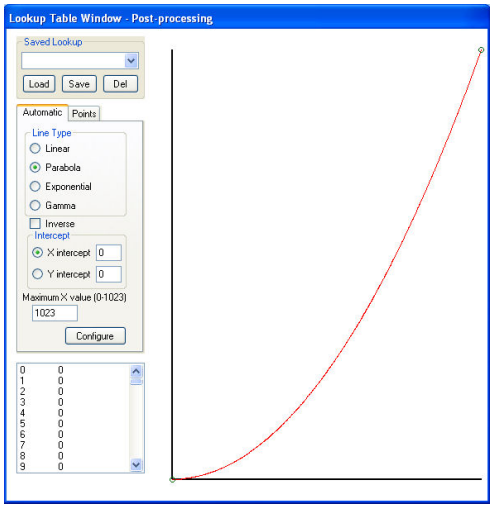


Fig. 17. A lookup table. The x-axis indicates input values from an image; the y-axis indicates the output values.

5.4.4. Stroboscopic Mode

Stroboscopic mode enables acquiring imagery at a near-instantaneous period in time, in relation to a stimulus wave. Since the camera cannot acquire effective imagery at a near-instantaneous period in time due to the lack of light in that period, the image is acquired over a longer period with a strobe effect on the laserbeam. To do this, two synchronized signals must be generated: the stimulus sent to a speaker, and the strobe signal to activate illumination. LaserView includes a tool to automatically configure a signal generator to output these two signals at a given frequency, phase, and duty cycle. When these items are set and the stroboscopic process is started, LaserView communicates with the Tektronix signal generator to initialize the synchronized analog outputs. While some of this behavior is possible with the National Instruments DAQ card, the same level of accuracy would not be achieved.

Once the signal generator has been started, constant communication is not necessary as it runs independently from the camera and piezo. However, the user has the ability to dynamically 'sweep' the phase and frequency of the strobe signal using the stroboscopic control panel. With each change made, the new values are sent to the function generator, which immediately starts outputting the new signals.

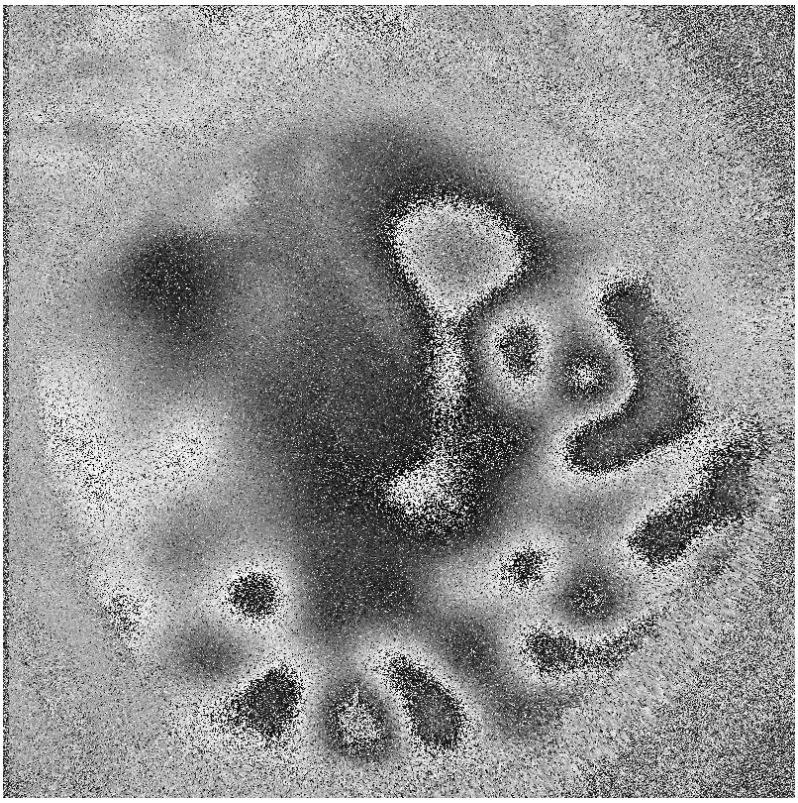


Fig. 18. Chinchilla TM vibrating in stroboscopic mode (double exposure, optical phase) at 11.2 kHz, with a phase offset of 45°.

5.4.5. Exposure Time Selection

In stroboscopic mode, the inherent difference in timing for the camera and strobe can introduce flickering in the frames as they are taken directly from the camera. The camera could be exposed for x strobes in one frame, but exposed for $x + 1$ strobes on the next frame. If x is large, i.e. for high frequencies, the difference between a single extra strobe is not noticeable and does not cause problems in the algorithms. However, for lower frequencies (< 1kHz) this can cause noticeable problems such as flickering in live images and poor quality digital holograms after algorithms are performed.

To eliminate this problem, the camera's exposure time is adjusted to a multiple of the period of the strobe. This ensures the same number of strobes occurs in each frame. For instance,

if the strobe is running at 300 Hz, the camera's exposure time must be 3.33*n* ms where *n* is an integer. This calculation and automatic setting can be performed using LaserView by pressing a button. LaserView uses the largest *n* where the resultant exposure time is valid for the current frame rate.

5.4.6. Computer Speed Control

Slow computer speed can be a contributing factor to losing synchronization between components. This can occur when the camera is integrating the image at the wrong time relative to the piezo's change due to image processing or another process using too much CPU time. If the piezo's position is changed while an image is being acquired, the laser's beam is shifted during the camera's acquisition which produces an image that is not useful. To avoid this behavior, especially on slower computers, a delay may be added to acquisition and/or processing in order to maintain proper synchronization.

To implement this, LaserView records the system time at the beginning of each capture and process, then checks the system time at the end. If not enough time has elapsed, a delay will be added.

5.4.7. Region of Interest

A rectangular region of interest (ROI) can be selected to run calculations on. The two operations that support the use of an ROI are beam ratio calculation and histogram. To perform a selection, the user clicks a button and is prompted to select the rectangle on the image window. The ROI can be displayed or hidden, and reselected as needed. When the ROI is defined, the histogram and beam ratio calculations are performed only on the area within the selected ROI.

5.4.8. Beam Ratio Calculation

The beam ratio is the ratio between the intensities of the reference beam and object beam, as observed by the camera. This is dependent on the object being observed, since its appearance may affect the amount of light from the object beam.

To measure this ratio, each beam must be measured individually. LaserView prompts users to cover one beam at a time, and then uses the data from those images to calculate the ratio. The ratio is updated live on an overlay so that changes can be seen as the optics are adjusted.

To make the process easier the software supports communication with hardware shutters to hide the individual beams, one at a time, and perform the calculations automatically. The shutters are controlled by the data acquisition card, which sets a digital signal to open or close the shutters.

5.4.9. Cross Section Viewer

LaserView's cross section tool gives the ability to measure and view the intensity of an image along a straight line. The line is defined by the user, horizontally, vertically, or by selecting two arbitrary points on the image. The intensity of the image along the line is plotted in real-time, giving a visual representation which can be used for measuring relative or absolute displacements and for achieving the optimal focus. An instantaneous representation of the raw data can be exported, giving the ability to perform detailed analysis.

5.4.10. Histogram

The histogram displays a live representation of the distribution of intensities of the image being displayed. As shown in Fig. 19, the x-axis represents each possible intensity in the image, and

the y-axis represents its distribution. This allows for calibration of camera settings, and comparisons between objects or lighting.

Implementation is done by iterating over each pixel of the image and keeping an array of each value's number of occurrences. That array is then displayed graphically as shown in Fig. 19.

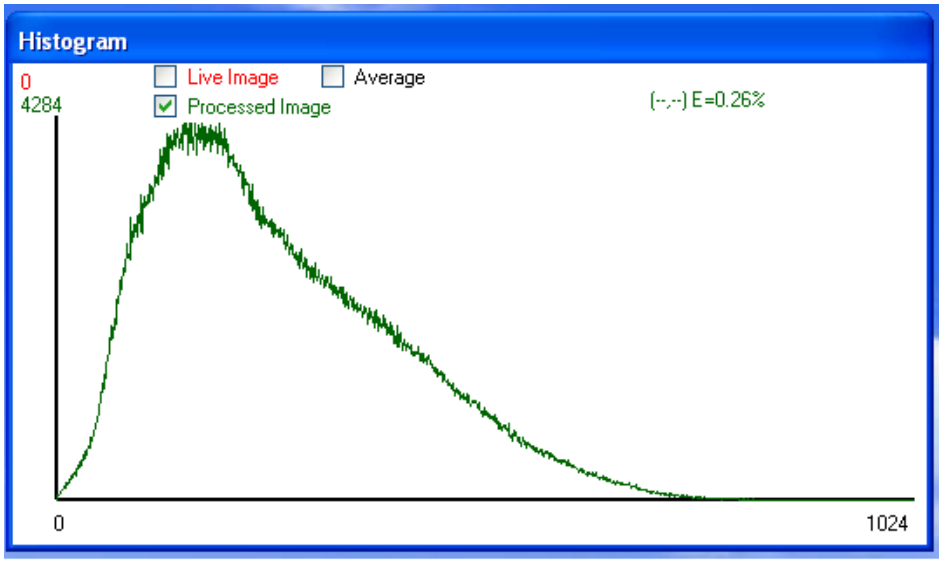


Fig. 19. Histogram window, showing the intensity distribution of a processed image.

5.4.11. EKG Synchronization

When experimentation is done on live animals, the heartbeat of the animal can interfere with the measurement of the object of interest. To reduce this problem, an external trigger is supported. This is implemented by using an analog input from the DAQ card, which triggers a capture when a certain user-defined threshold is reached. This can be configured to trigger a capture after a specified delay, and can ignore triggers for a specified period of time.

The InputTrigger class handles this by continuously monitoring the analog input channel `TRIGGER_AI`, specified in settings.h, at a rate of 100 samples per second. The Input Trigger window

displays a real-time visualization of the input, and allows the user to indicate a value that will trigger a capture with an arbitrary delay time. When a value received from the analog input exceeds the threshold, InputTrigger's `sendTrigger()` method is called, activating the capture procedure in the control window.

6. Experimental Setups

6.1. Preliminary Tests

The complete holography system contains many components, both software and hardware, requiring synchronization and interoperation. Therefore, extensive testing was necessary before deploying the system. Early experiments were performed on synthetic materials, allowing for long-term testing for optical components and the LaserView software.

6.2. Massachusetts Eye & Ear Infirmary

An experimental version of the system has been installed at Mass. Eye and Ear Infirmary (MEEI), as seen in Fig. 20 [Furlong et al., 2008].

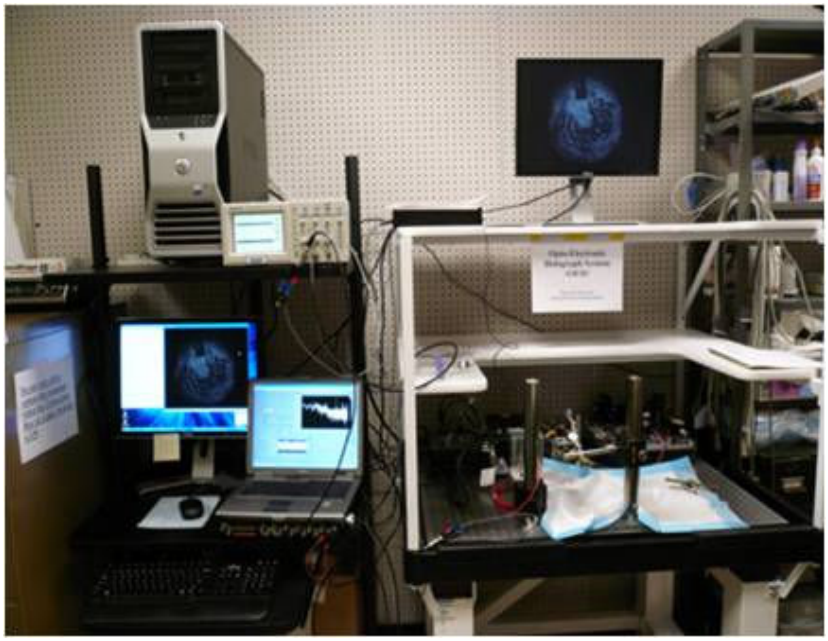


Fig. 20. Experimental version of the system installed at MEEI, Boston MA.

Experiments have been carried out on a cadaveric human, a cadaveric chinchilla, a live chinchilla, and a cadaveric cat. These experiments have yielded interesting results, and show a series of vibration patterns on the TM dependant on the frequency and amplitude of the stimulus, as shown in Fig. 21.

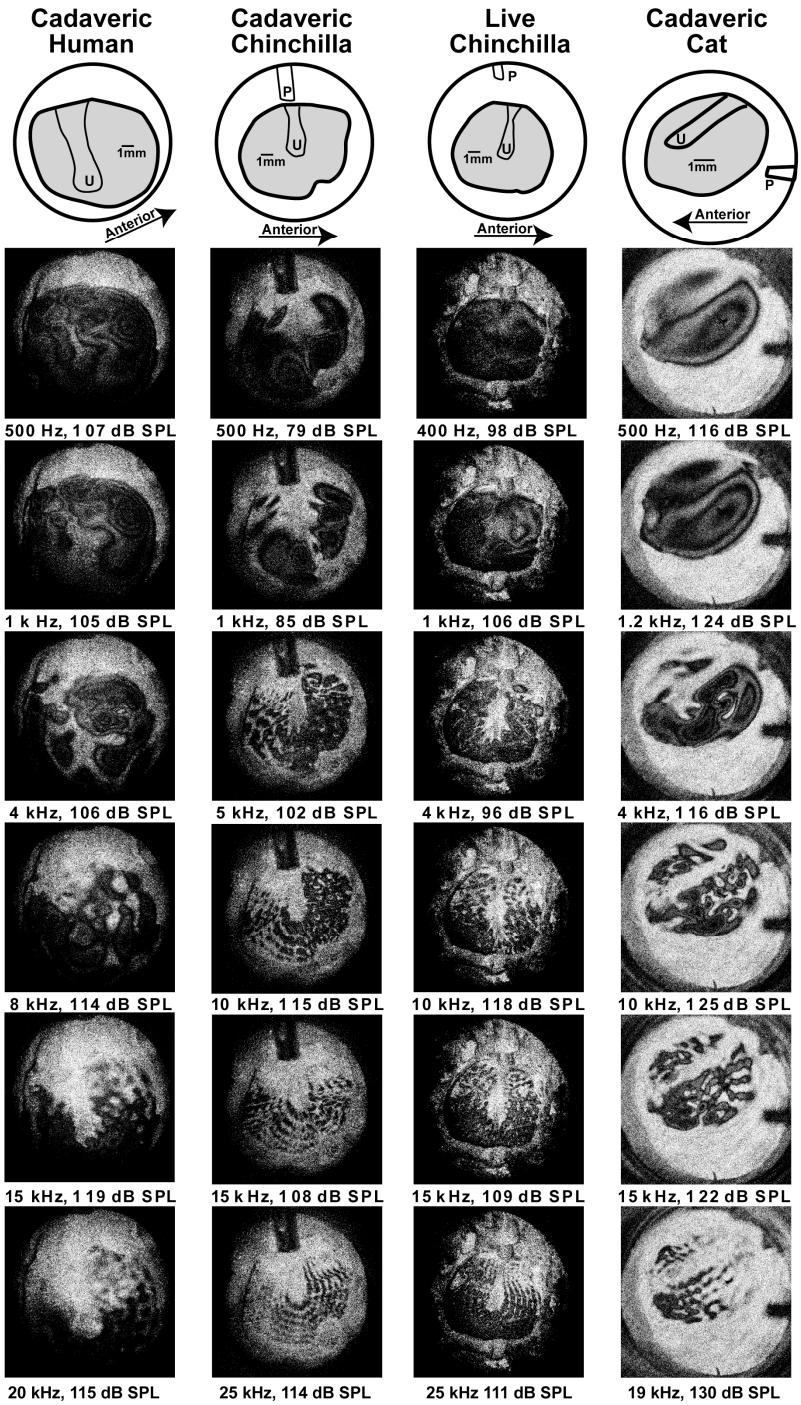


Fig. 21. Experimental Results from four specimens. [Furlong et al., 2008; Furlong, Rosowski, 2008; Hernandez-Montes et al., 2008]

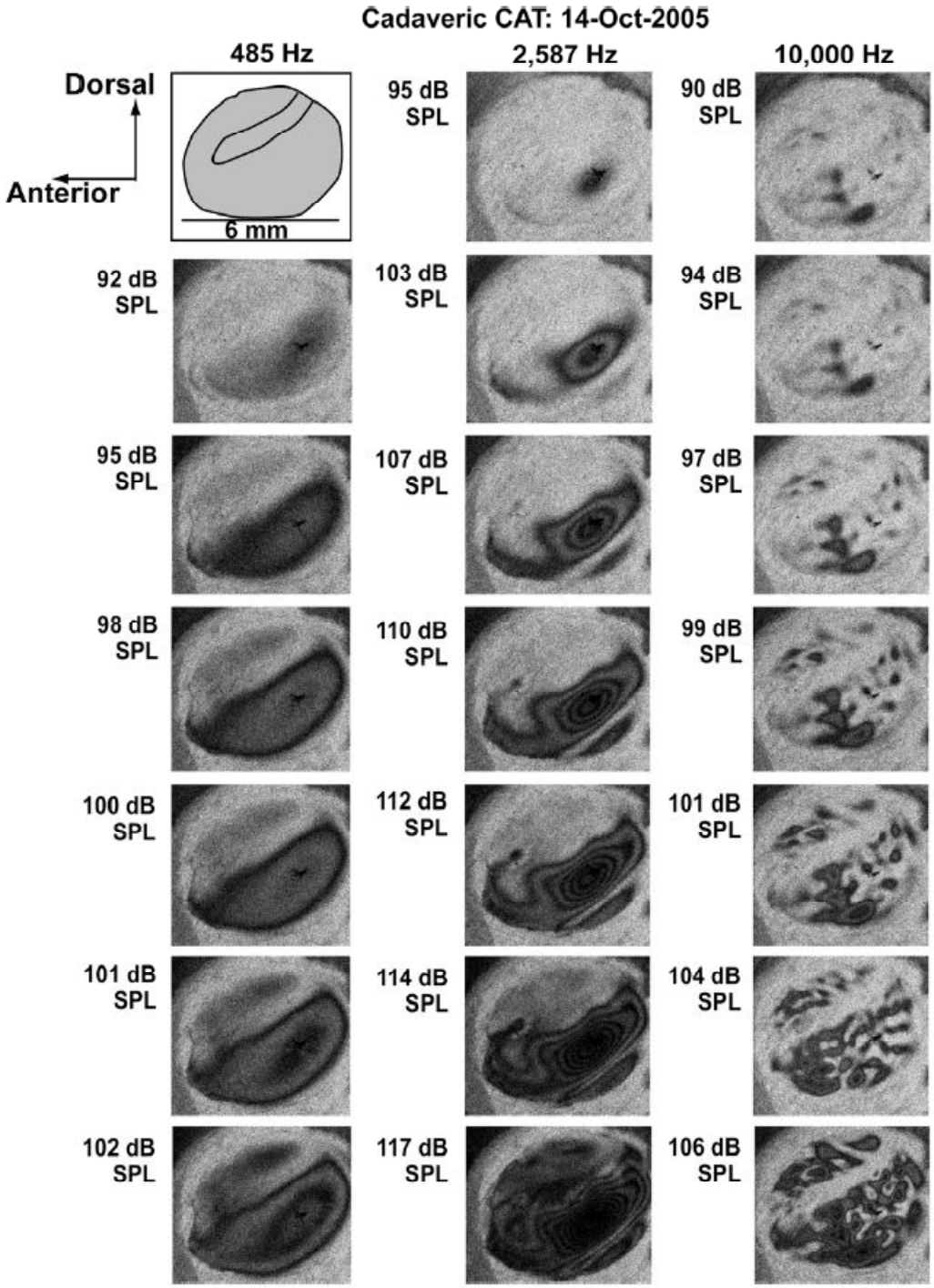


Fig. 22. Time-average holograms of a cadaveric cat at varying frequencies and levels [Rosowski, et al., 2009].

The patterns from these experiments are described as simple, complex, or ordered depending on the appearance of the patterns [Hulli, 2008]. As shown in Fig. 22, simple patterns are observed at lower frequencies, complex at medium, and ordered at high frequencies.

To enhance definition of fringes, TMs may be painted with a white solution of TiO_2 powder and saline to give better reflectivity. Fig. 23 shows the improvement of contrast from painting a TM [Rosowski et al., 2009]. Painting an eardrum is not currently a viable option for living humans, but is helpful for research in obtaining the best possible fringe patterns.

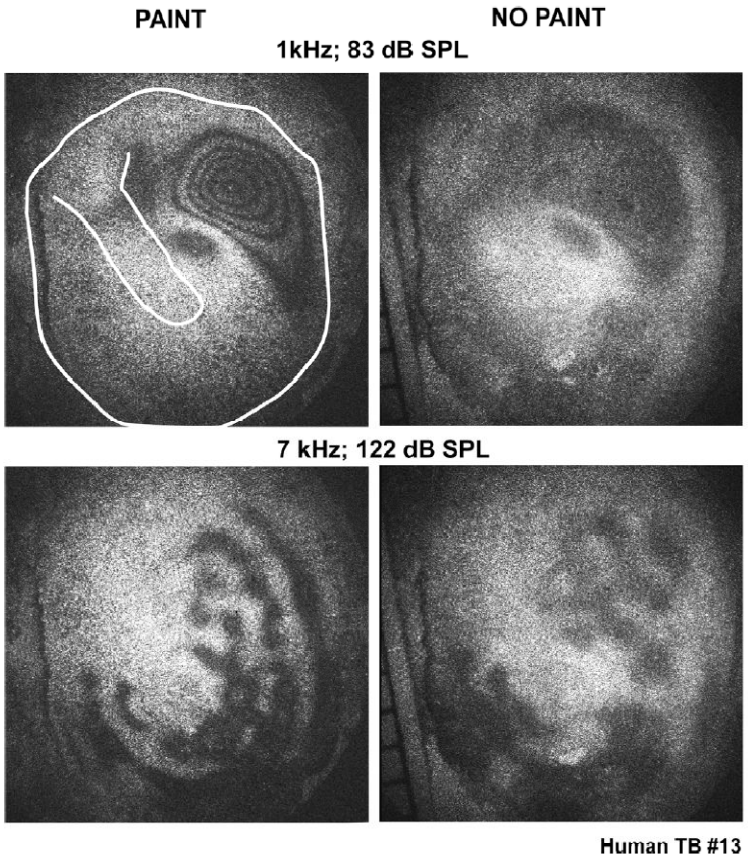


Fig. 23. Comparison of a painted and a non-painted tympanic membrane in time-average mode. [Rosowski et al., 2009]

Using the LaserView software in conjunction with other analysis software, 3D imagery can be created showing a detailed image of an object, as shown in Fig. 22 and Fig. 23.

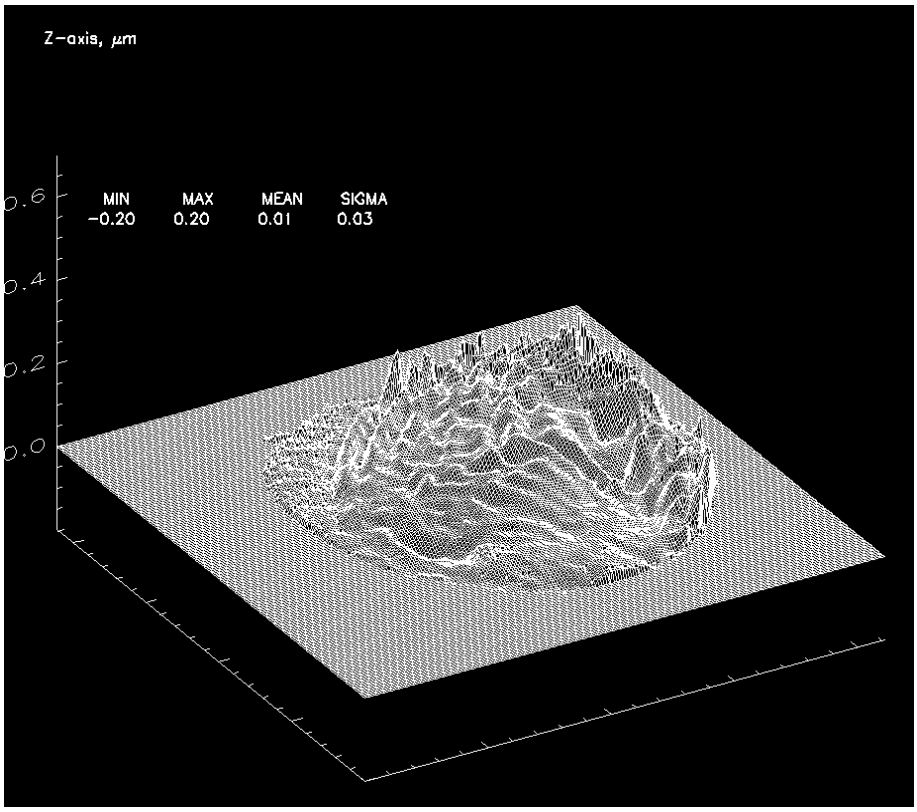
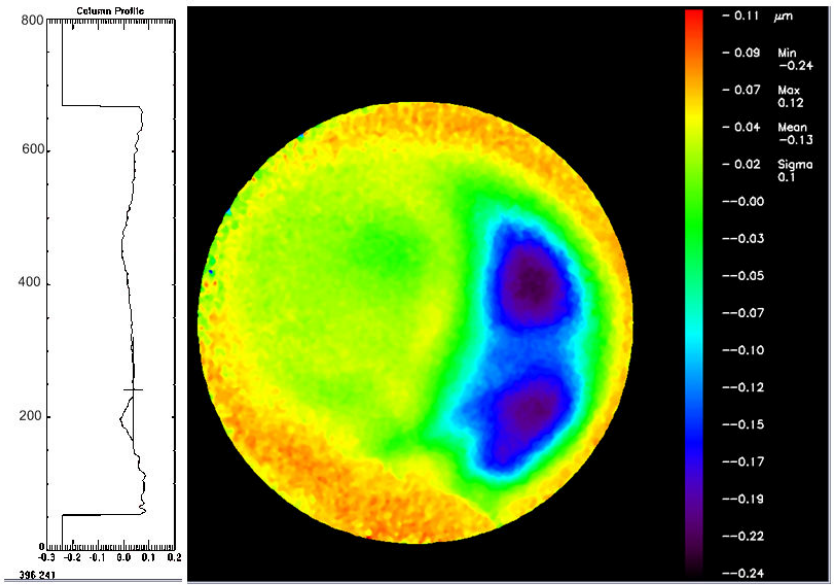
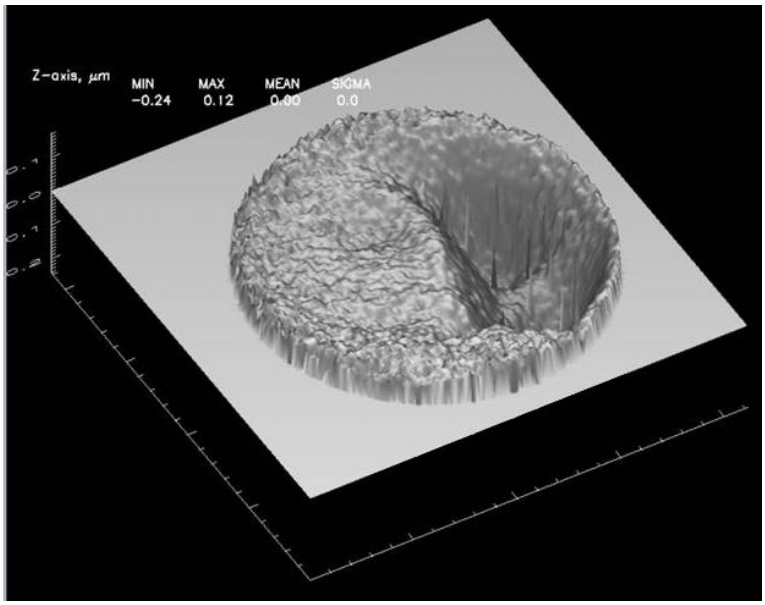


Fig. 24. 3D wireframe plot of full-field-of-view stroboscopic holography measurements in human temporal bone vibrating at 20 kHz, showing a maximum displacement in the order of 390nm, with a resolution of 1-5 nm [Hulli, 2008].



(a)



(b)

Fig. 25. Full-field-of-view stroboscopic holography measurements in human temporal bone at 800 Hz, showing a peak-to-peak surface out of plane displacement on the order of 120nm: (a) unwrapped phase (2D plot); and (b) 3D plot. [Hulli, 2008; Furlong, Rosowski, 2008]

Among other experiments, investigation was performed into the speed of wave propagation on various species' middle ear delays based on the wave propagation on the TM.

Species	Estimate of middle-ear delay	Source	Reference
Cat	0.036 ms	Analysis of middle-ear input admittance measurements	Puria and Allen 1998
Gerbil	0.032 ms	Group delay in inner-ear to TM sound pressure ratio	Dong and Olson 2006
Human temporal bones	0.038–0.093 ms	Analysis of middle-ear functional measurements	O'Connor and Puria 2008
Gerbil	0.030 ms	Stapes velocity to TM sound pressure Group delay	Overstreet and Ruggero 2002
Gerbil	0.020–0.039 ms	Stapes velocity to TM sound pressure Group delay	Ravicz et al., 2008
Human temporal bone	0.083 ms	Group delay in inner-ear to TM sound pressure ratio	Nakajima et al. 2009
Human temporal bones	0.06–0.089 ms*	Estimated from computed wave propagation velocity and 4 mm TM radius	This study
Cat	0.075–0.100 ms*	Estimated from computed wave propagation velocity and 3 mm TM radius	This study
Chinchilla	0.060–0.160 ms*	Estimated from computed wave propagation velocity and 4 mm TM radius	This study

* Delay varies inversely with frequency.

Fig. 26. Estimated middle ear delays depending on species [Rosowski et al., 2009]

7. Other Applications

In addition to the application on researching and diagnosing the human tympanic membrane, LaserView can also be a useful tool for several other applications.

7.1. Microscopy

Using similar optics to the tympanic membrane application, a laser beam is modulated in the same fashion to produce phase-shifted images. This produces the same type of imagery and data that is acquired, but on a smaller scale, giving a higher spatial resolution. This method gives the ability to measure the displacement of MEMS devices in full-field-of-view with unprecedented accuracy [Balboa, Dobrev, Fossett, 2008].

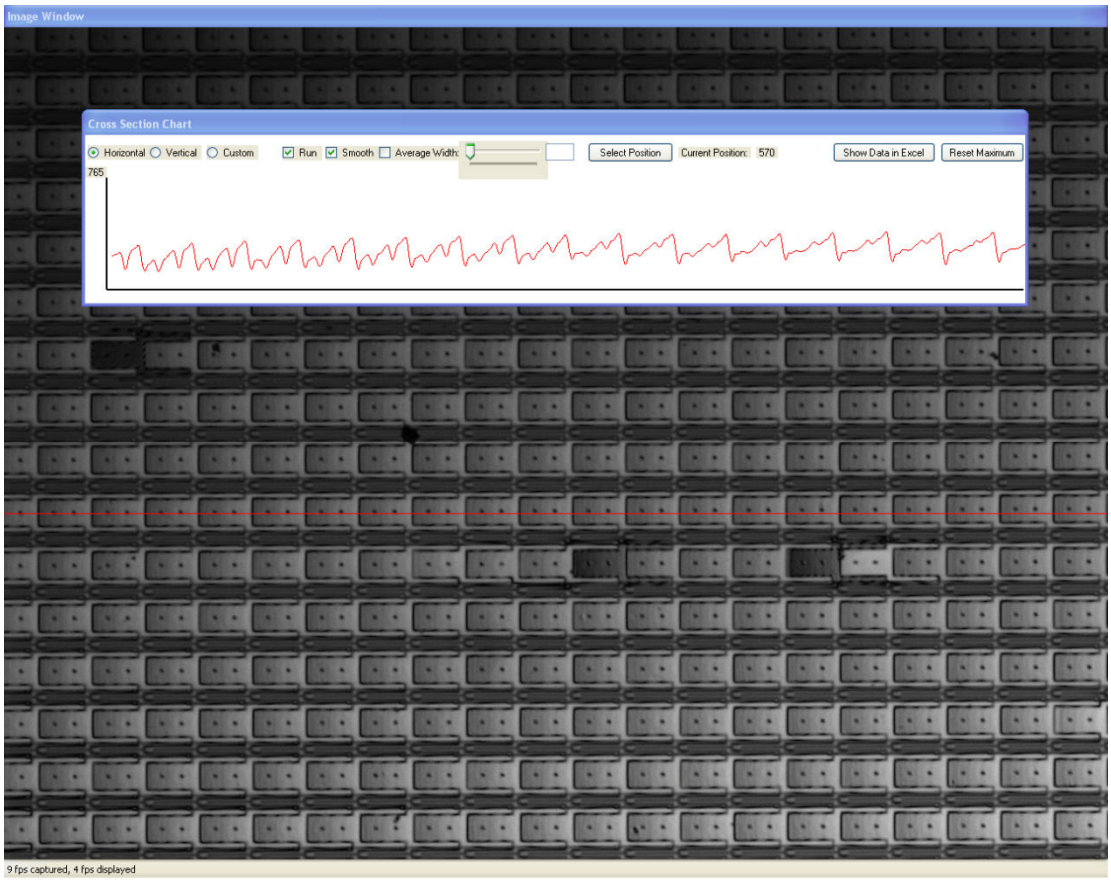


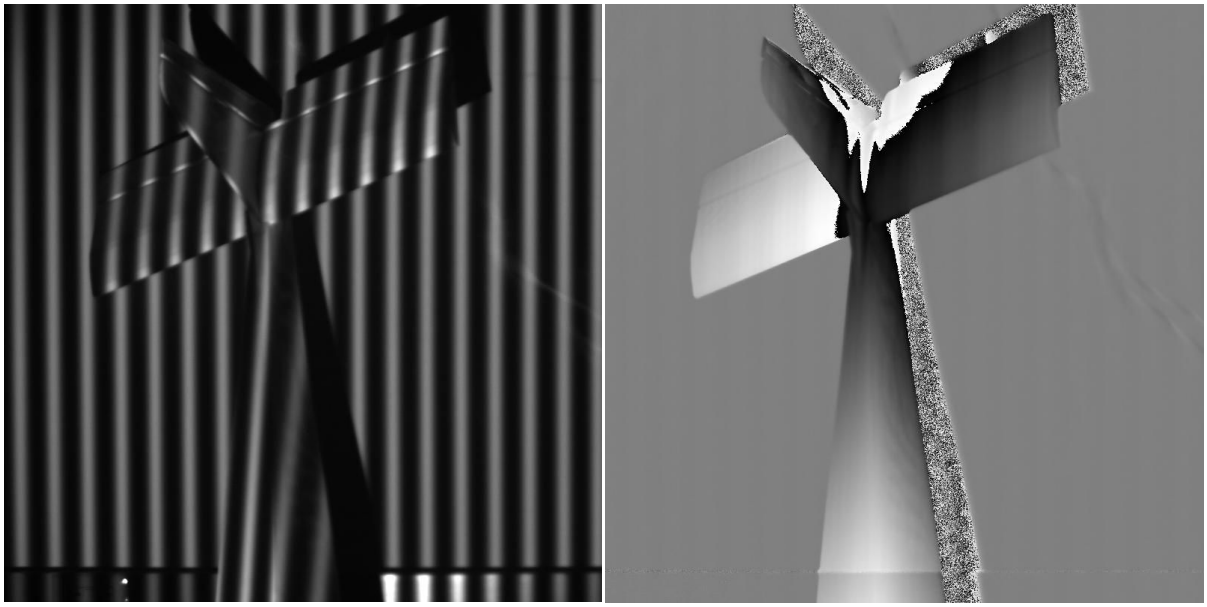
Fig. 27. Image of micromirror array using a cross section chart to examine displacement on the horizontal axis near the middle of the image.

Microscopic holography allows the observation of movements and deformations in MEMS devices, including precise heat detection using deformations of specific MEMS devices.

7.2. Fringe Projection

Instead of modulating a laser beam, fringe projection can be used for macro-scale objects. A sinusoidal image, with a varying phase synchronized with acquisition, is projected onto the object and observed from a camera near the projection source. Using the same algorithms as the tympanic membrane project, this imagery can be processed to extract the shape of the object. Currently, the

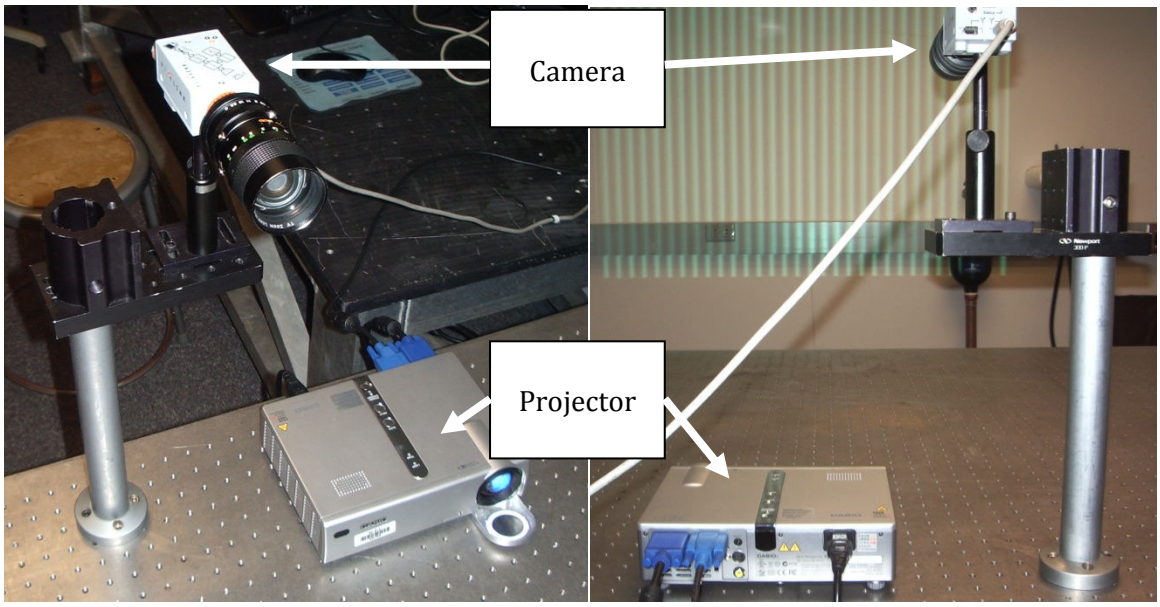
project has an experimental setup that uses a low-quality computer projector to project the sinusoidal images. This setup currently has limitations of a low frame rate due to synchronization and refresh-rate issues.



(a) (b)

Fig. 28. Fringe projection holography of a model airplane. (a) One of four phase-shifted images with fringes projected onto object of interest. (b) The wrapped phase of this object.

The projector, shown in Fig. 29, is controlled by LaserView and illuminates the object with a sinusoidal fringe pattern sequentially with a varying phase offset. This gives the appearance of a series of fringes continuously travelling on the object, as shown in Fig. 28 (b). The camera, as seen in Fig. 29, is focused on the object of interest. Using the same double exposure algorithms as it uses for laser interferometry, LaserView generates a live wrapped phase image, as shown in Fig. 28 (b).



(a) (b)

Fig. 29. Fringe projection system. A computer projector and digital camera aimed at the same area are connected to a computer and synchronized similar to the piezo and camera's synchronization.

Several major improvements can be made to this system. Replacing the standard computer with micromirror projection would allow for a higher quality projections, higher frame rate, and the possibility of projecting infrared light, eliminating visible fringes on the subject [Texas Instruments, 2009]. Adding additional algorithms to the software will also allow for processing imagery with fewer shifted frames, thus allowing for near-instantaneous display of the image's phase.

Possible applications using fringe projection include facial recognition and 3-dimensional object scanning for modeling, virtual reality, or video games.

8. Conclusions

Full-field-of-view laser holography serves as an extremely useful tool for the measurement of object shape and deformations for many applications. In measurement of the tympanic membrane, it is the most useful technique for detailed measurement of the response to sound because it can offer a live view of the entire TM, with the ability to measure nanometer-scale deformations.

9. Future Work

While many aspects of this project have been resolved, there are still many more areas to be researched and further developed. In software, further improvements, including EKG synchronization, should be tested and implemented. Enhancements and additional image processing algorithms are planned to enhance the system further. In hardware, additional miniaturization and further engineering are necessary to make this technology viable for common medical usage.

One possible hardware improvement is to eliminate the piezo and remove one of the two fibers. This would require additional algorithms to be implemented in the software to perform the phase calculation.

Digital holography, a lens-less configuration of the optical system allows for focusing through the software at an arbitrary distance from the camera through the use of Fourier transforms combined with other algorithms.

The LaserView program and the OEHO system have already been heavily utilized in experimentation of various species' tympanic membranes. Further research into the middle ear is planned, and after further developments, the system may be a useful clinical diagnostic tool.

To allow for clinical measurement of human TMs, specialized hardware is under development to control the movement of an otoscope to allow it to be positioned it in view of the tympanic membrane, while also keeping the patient's head stable.

A future improvement that would require both hardware and software modifications is to add the ability to automatically adjust laser beams to improve the image quality. This would require servo motors to be operated by the computer to change the behavior of the optics. A software-only improvement is the addition of image stability controls, which would show a stabilized image, adjusted based on a reference point that is maintaining its position. Several other improvements are prepared for implementation and deployment as well, as requested by the operators using the Mass. Eye and Ear Infirmary configuration.

LaserView will be further developed to provide functionality as needed for research of human TMs, and newly developed image processing algorithms will be added.

Appendix A: Selected code from LaserView

Implementation of optical phase mode

```
for(int i = 0; i < numPixels; i++){
       if (imRefIn == NULL){ // not in reference mode
            imag = *(imGroup[3]++)-*(imGroup[1]++);
            real = *(imGroup[0]++)-*(imGroup[2]++);
       }
       else{
             // pixel values are not incremented here because they are
used again
            imag = ( *(imGroup[0]) - *(imGroup[2])) * (*(imRef[1]) -
*(imRef[3])) - (*(imGroup[1]) - *(imGroup[3])) * (*(imRef[0]) -
*(imRef[2]));
            real = ( *(imGroup[0]++) - *(imGroup[2]++)) *
(*(imRef[0]++) - *(imRef[2]++)) + ( *(imGroup[1]++) - *(imGroup[3]++))
* (*(imRef[1]++) - *(imRef[3]++));
      }
      if(imag||real)
      {
            tmp = atan2(imag,real);
            if(tmp < 0)
                  //tmp +=2*Math::PI;
                  // using predefined literal for efficiency:
                  tmp +=6.28;
            //*(imageOut++)=int(tmp/(2*Math::PI)*camMax);
            // using predefined value for efficiency:
            *(imageOut++)=(int)(tmp*mul);
      }
      else
            *(imageOut++)=0;
}
```

Implementation of image averaging

```
if (averageNumSets > 1){
      int rawImageSize = cam->getRawImageSize();
      // tmpImage must be inited the first time.
       // After that, the same buffer will be reused.
      if (tmpImage == nullptr)
            tmpImage = (RawImage)malloc(rawImageSize);
      // check if we need to allocate space for the images to be
averaged
      if (averageNumSets != averageNumSetsMallocd){
            // clear any previous buffer
            free(averagedImages);
            averagedImages =
                  (RawImage)malloc((averageNumSets-1) * ImageSize);
            averageNumSetsMallocd = averageNumSets;
      }
      // go over each pixel
      int numPixels = imageWidth*imageHeight;
      for(int i = 0; i < numPixels; i++){
            // start with the new image
            tmpImage[i] = args->outputImage[i];
            // average over all the stored images
            for(int c = 2; c <= averageNumSets; c++)
                  tmpImage[i] +=
                        averagedImages[imageWidth*imageHeight * (c-2)
+ i];
            tmpImage[i] /= averageNumSets;
            }
      lastAverageImageSaved++;
      if (lastAverageImageSaved >= (averageNumSets-1))
            lastAverageImageSaved = 0;
      // copy the most recent live image to the average buffer
      memcpy((averagedImages + rawImageSize * lastAverageImageSaved),
             args->outputImage, rawImageSize);
```

```
      // copy the averaged image into the output image buffer,
       // which will be drawn
      memcpy(args->outputImage, tmpImage, rawImageSize);
}
```

Implementation of shifter voltage calculation

```
// stepUp is true when incrementing, false when decrementing
bool stepUp = true;
// loop over each value to be tested
for(double v = minV; v <= maxV; v += stepV){
      // if going up, go down; if going down, go up.
      // remove this line to disable step-up, step-down
      stepUp = !stepUp;
      // loop through the four phases
for(int ph2 = 0; ph2 <= 4; ph2++){
            int ph = ph2;
            if (!stepUp)
                  ph = 4 - ph;
            double shifterVoltage = v * ((double)ph / 4);
            nicard->SetValueAnalogChannel(SHIFTER_AO,shifterVoltage);
            // [error-handling code removed]
            cam->reorderBits((RawImage)images[ph]);
      }
      // alpha = (I5 - I1) / (2 (I4 - I2))
      // start with I5
      memcpy(alpha, images[5-1], imageSizeBytes);
      // subtract I1
      for(int i = 0; i < imageNumPixels; i++){
            alpha[i] -= images[1-1][i];
      }
      //denominator starts with I4
      memcpy(tmpImage, images[4-1], imageSizeBytes);
      // subtract I2 and multiply difference by 2
      for(int i = 0; i < imageNumPixels; i++){
            tmpImage[i] -= images[2-1][i];
            tmpImage[i] *= 2;
```

```
        }
        // do the division
        for(int i = 0; i < imageNumPixels; i++){
              if (tmpImage[i] == 0)
                    alpha[i] = 0;
              else
                    alpha[i] /= tmpImage[i];
        }
        // now we have alpha.
        // calculate the mean value.
        int imageSum = 0;
        for(int i = 0; i < imageNumPixels; i++){
              imageSum += alpha[i];
        }
        double imageMean = (double)imageSum / imageNumPixels;
        double alphaVal = Math::Acos(imageMean);
        // add this to the results.
        textBoxPhaseCalculationResults->Text +=
            v.ToString() + "\t" + alphaVal.ToString() + "\r\n";
```

Implementation of multiplier for modulation mode

```
if (scaleValueModulation && scaleValueModulation != 1.0){
      // Final value is square root of sum of squares
*imageOut = min(camMaxValue-1,
                  (int)(scaleValueModulation *
                        Math::Sqrt(real*real + imag*imag)));
}
```

Implementation of modulation mode

```
// iterate over each pixel in the image.
 for(int i = 0; i < numPixels; i++){
       // calculation with no Reference images
       if (imRefIn == NULL){
            // difference between opposite images
            real = (*(imGroup[0]++) - *(imGroup[2]++));
            imag = (*(imGroup[3]++) - *(imGroup[1]++));
       }
       else{
            // reference mode on
            real = *(imGroup[0]++) - *(imGroup[2]++) + *(imRef[0]++) -
(imRef[2]++);
            imag = *(imGroup[3]++) - *(imGroup[1]++) + *(imRef[3]++) -
*(imRef[1]++);
       }
       if (scaleValueModulation && scaleValueModulation != 1.0){
             // Final value is square root of sum of squares
             *imageOut = min(camMaxValue-1, (int)(scaleValueModulation
* Math::Sqrt(real*real + imag*imag)));
      }
      else{
             // Final value is square root of sum of squares
            *imageOut = qSqu->doQuickSqrt(real*real + imag*imag);
      }
      imageOut++;
 }
```

Implementation of picture-in-picture

```
// NB: output may be equal to primary or pip, so the inputs should be copied
//first if they are still needed after this call.
private: System::Void createPIP(RawImage output, RawImage primary, RawImage
pip){
       RawImage tmpImage = NULL;
       if (pip == output){
            // we don't want to lose the pip,
             // so copy it if it's the same pointer
            tmpImage = (RawImage)malloc(imageWidth * imageHeight * 2);
            copyImage(tmpImage, pip, imageWidth * imageHeight);
            pip = tmpImage;
       }
       if (output != primary){
            // copy the primary image to output
             copyImage(output, primary, imageWidth * imageHeight);
       }
       // now copy the pip to the output image
       for (int y = 0; y < imageHeight/2; y++){
            for (int x = 0; x < imageWidth/2; x++){
                  (*output) = (*pip);
                  pip+=2; // pip incremented by 2 pixels on the x axis
                  output++;// output incremented by 1 pixel on x-axis
            }
            pip += imageWidth;// pip incremented by 2 pixels on the y axis
            output += imageWidth/2;// output incremented by 1 pixel on y-axis
       }
       if (tmpImage != NULL){ // if we created an image, deallocate it
             free(tmpImage);
       }
 }
```

Appendix B: Hardware and Software Used for Running LaserView

Computers

Virtually any PC running Windows XP or higher with sufficient memory and processor speed.

Sample configuration:

Dell Precision Workstation T7400, 32-bit

Dual Core Intel® Xeon® Processor X5260 (3.33GHz,6M L2,1333)

Microsoft Windows XP Professional

4GB DDR2 SDRAM

256MB PCIe x16 nVidia NVS 290, Dual Monitor Capable

http://www.dell.com/us/en/business/desktops/precn_t7400/pd.aspx?refid=precn_t7400&cs=04&s=bsd

Cameras

Silicon Imaging SI-1280FM-CL

Maximum Resolution: 1280x1024

CameraLink interface (requires EPIX CameraLink board)

Monochromatic

12 bits per pixel

http://www.siliconimaging.com/SI-1280CL%20spec.htm

PixeLink PL-A741 / PL-A741E / PL-B741F / PL-B741EF

Maximum Resolution: 1280x1024

Optional near-infrared signified by “E” in model number

Firewire (IEEE 1394) interface

Monochromatic

10 bits per pixel

http://www.pixelink.com/products_info.asp?productcode=PL-B741F

PixeLink PL-A781 / PL-B781F

Maximum Resolution: 2208 x 3000

Firewire (IEEE 1394) interface

Monochromatic

10 bits per pixel

http://www.pixelink.com/products_info.asp?productcode=PL-B741F

Capture Cards (for Silicon Imaging Cameras)

Most EPIX CameraLink boards

Tested boards:

PIXCI CL1 (PCI)

PIXCI CL2 (PCI)

PIXCI E4DB (PCI Express x4)

Function Generator

Tektronix AFG3102

http://www2.tek.com/cmswpt/psdetails.lotr?ct=PS&lc=EN&ci=13567&cs=psu

Data Acquisition Cards

Virtually any National Instruments DAQ card supported by NI-DAQmx with enough input and output channels for the application.

Sample of tested boards:

NI PCI-6713

8 AO (analog output) channels, 12-bit

No AI (analog input)

http://sine.ni.com/nips/cds/view/p/lang/en/nid/10702

NI PCI-6259

4 AO channels, 16-bit

32 AI channels, 16-bit

http://sine.ni.com/nips/cds/view/p/lang/en/nid/10702

Software

Microsoft Visual C++ Express Edition

Compilation of LaserView

EPIX XCAP Lite

Testing and configuration of Silison Imaging cameras

http://www.epixinc.com/products/xcap.htm

EPIX XCLIB

Libraries for communication with EPIX PIXCI boards and Silicon Imaging cameras

http://www.epixinc.com/products/xclib.htm

IDL

Programming Language optimized for graphics operations

http://www.ittvis.com/ProductServices/IDL.aspx

PixeLink Software Developers Kit v4

Libraries for communication with PixeLink cameras

http://www.pixelink.com/uploads/PL-SDK-VERSION-4_datasheet.pdf

National Instruments NI-DAQmx

Software and libraries for communication with National Instruments DAQs and to the VISA interface for the communication with the function generator.

http://sine.ni.com/nips/cds/view/p/lang/en/nid/10181

References

Balboa, M., Dobrev, I., Fossett, R., 2008, *MEMS for Real-Time Imaging Applications*, Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA.

Bushman, T., 1993, *Automated Fringe Unwrapping by Energy Minimization*, Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA.

Creath, K., 1985, *Phase-shifting speckle interferometry*, Appl. Opt. 24, 3053-3058

Csillag, A., 2005, *Atlas of the sensory organs: functional and clinical anatomy*, 1st ed., Human press, Totowa, New Jersey.

Dwyer, B. W., Maccaferri, M. E., Wester, C. R., 2008, *Design and realization of a laser holographic otoscope*, Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA.

Furlong, C., 2000, *The RTI Image File Format*, http://users.wpi.edu/~cfurlong/me-593n/rtiimage.html (last accessed: 16 November 2009).

Furlong, C., Pryputniewicz, R. J., 1998, *Electro-optic holography method for determination of surface shape and deformation*, SPIE, 3478:86-97.

Furlong C., Hernández-Montes, M. S., Hulli, N., Cheng, J. T., Ravicz M. E., and Rosowski J. J., 2008, *Development of an optoelectronic holographic otoscope for characterization of sound-induced displacements in tympanic membranes*, 31st Midwinter Meeting of the Association for Research in Otolaryngology, Phoenix, Az.

Grundman, J., and Wigton, R., *Tympanic membrane evaluation*, in *Ear Examination*, http://webmedia.unmc.edu/intmed/general/eye&ear/introtm.htm, (last accessed: 10 August 2009).

Hernández-Montes MdS, Furlong C, Rosowski JJ, Hulli N, Harrington E, Cheng JT, Ravicz ME, Santoyo FM, 2008, *Optoelectronic holographic otoscope for measurement of nano-displacements in tympanic membrane*, J. Biomed. Optics., 14: 034023

Huber, A. M., Schwab, C., Linder, T., Stoeckli, S. J., Ferrazzini, M., Diller N., and Fisch U., 2001, *Evaluation of eardrum laser Doppler interferometry as a diagnostic tool*, Laryngoscope, 111(3):501-507.

Hulli, N., Franco, R. A., Rodriguez-Vera, R., and Furlong, C., 2007, *Development of an optoelectronic, high-speed, 3D shape measurement system for medical applications*, *Proc. SEM.*

Hulli, N., 2008, *Development of an optoelectronic holographic otoscope system for characterization of sound-induced displacements in tympanic membranes*, Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA.

Karasic, G., Largesse, N., Lincoln, L. S., 2008, *Miniturization of an Optoelectronic Holographic Otoscope for Measurement of Nanodisplacements in Tympanic Membranes*, Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA.

Pryputniewicz, R. J., 1985, *Time average holography in vibration analysis Opt. Eng.*, 24(5):843-848.

Pryputniewicz, R. J., 1987, *Quantitative interpretation of time-average holograms in vibration analysis*, *Optical metrology*, in NATO Advanced Science Institute (ASI) Series, Porto, Portugal, pp. 296-371.

Pryputniewicz, R. J., 1989, *Measurement of vibration patterns using electro-optic holography*, *SPIE*, 1162:456-467

Pryputniewicz, R. J., 1996, *Holographic numerical analysis*, ME/CHSLT-NEST, Worcester Polytechnic Institute, Worcester, MA.

Kuypers, L. C., Decraemer, W. F., and Dirckx J. J. J, 2006, *Thickness distribution of fresh and preserved human eardrums measured with confocal microscopy*, Otol Neurotol, 27(2): 256-264.

Rosowski, J. J., Cheng, J. T., Ravicz, M. E., Hulli, N., Hernandez-Montes, M., Harrington, E., Furlong, C., 2009, *Computer-assisted time-averaged holograms of the motion of the surface of the mammalian tympanic membrane with sound stimuli of 0.4–25 kHz*, Hearing Research 253 83–96.

Silicon Imaging, Inc., *SI-1280F- M & RGB MegaCamera™*, http://www.siliconimaging.com/SI-1280CL%20spec.htm, (last accessed: 14 September, 2009)

Stetson, K. A., Brohinsky, W. R., 1985, *Electrooptic holography and its application to hologram interferometry*, Appl. Opt. 24, 3631-3637

Stetson, K. A., Brohinsky, W. R., 1987, *An electro-optic holography system for vibration analysis and nondestructive testing*, Proc. SPIE, 746:44-51.

Sundberg, M.,2008, *Optical methods for tympanic membrane characterization towards objective otoscopy in otitis media*, Department of Biomedical Engineering, Linköpings universitet, Linköping Dissertation No. 1173, Linköping, Sweden, pp.19-21.

Tektronix, Inc., *Arbitrary/Function Generators*, http://www2.tek.com/cmswpt/psdetails.lotr?ct=PS&lc=EN&ci=13567&cs=psu, (last accessed: 10 August, 2009)

Texas Instruments. Inc., *Discovery 3000 Kit*, http://www.dlp.com/regional/dlp_discovery/discovery3000.aspx (last accessed: 10 November, 2009)

Tonndorf, J., Khanna, S.M., 1970, *The role of the tympanic membrane in middle ear transmission*, Ann. Otol. 79, 743–753.

Tonndorf J, Khanna, S.M., 1972, *Tympanic-Membrane Vibrations in Human Cadaver Ears Studied by Time-Averaged Holography*, J. Acoust. Soc. Am. 52, 1221, DOI:10.1121/1.1913236

Wikipedia, http://upload.wikimedia.org/wikipedia/commons/7/7c/HumanEar.jpg, (last accessed: 19 August 2008).

Whittemore, K. R., Merchant, S. N., Poon B. B., and Rosowski, J. J., 2004, *A normative study of tympanic membrane motion in humans using a laser Doppler vibrometer (LDV)*, Hearing Research, 187(1-2):85-104.