Worcester Polytechnic Institute
Digital WPI

Masters Theses (All Theses, All Years) Electronic Theses and Dissertations

2004-04-30

Multi-Layered Oxygen Tension Maps of the Retina

Adam Stuart Norige
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-theses

Repository Citation
Norige, Adam Stuart, "*Multi-Layered Oxygen Tension Maps of the Retina*" (2004). *Masters Theses (All Theses, All Years)*. 572.
https://digitalcommons.wpi.edu/etd-theses/572

This thesis is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Masters Theses (All Theses, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

المنارة للاستشارات

www.manaraa.com

Multi-Layered Oxygen Tension Maps of the Retina

By

Adam S. Norige

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHINIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

In

Biomedical Engineering

By

____

April 2004

APPROVED:

____

Ross D. Shonat, Ph.D., Major Advisor

____

Karl G. Helmer, Ph.D., Committee Member

____

Michael A. Gennert, Sc.D., Committee Member

www.manaraa.com

Acknowledgements

This work was supported in part by a biomedical engineering research grant from the Whitaker Foundation, Rosslyn VA.

I would like to express my gratitude to my family and friends for the support and guidance that they have provided me with throughout this work and to:

My fiancé *Jennifer Bachand* for always encouraging me to strive for success and for her continuous support;

David Mason and *Walker League-Pike* for their development of and maintenance of the diabetic rats examined in this study;

Kevin Cornwell for his interest in this project and for his valued scientific opinions regarding this research;

Karl Helmer and *Michael Gennert*, my thesis committee, for their interest in this project and their assistance in allowing me to complete this research;

Ross Shonat, my research advisor, for enabling me to fully appreciate science and for his generous guidance throughout my academic career.

المنارة للاستشارات

www.manaraa.com

Abstract

Retinal hypoxia is associated with many retinal diseases, such as diabetic retinopathy. Current retinal research suggests that retinal hypoxia appears prior to the onset of diabetic retinopathy. The preliminary association of retinal hypoxia to the early stages of diabetic retinopathy is stimulating the development of new technologies to measure the oxygen content of retinal tissue.

Frequency domain phosphoresence lifetime imaging (PLI) is a promising technology that enables the mapping of the oxygen content across the entire retina in the form of two-dimensional images. The two-dimensional images generated from the PLI process are a spatial mapping of the retinal tissue's oxygen tension. Currently, the phosphorescent based oxygen tension PLI measurements contain contaminating auto-fluorescent signals in addition to the desired phosphorescent signals. These auto-fluorescent signals artificially inflate the oxygen tension readings due to the nature of fluorescent signals in phosphorescent imaging. Additionally, the maps generated through PLI appear to contain oxygen tension information from both the retinal vasculature and the choroidal vasculature. The choroidal vasculature is situated directly behind the retina and can have a different oxygen tension value than the retinal vasculature.

This research enhanced the PLI system by mathematically eliminating the contaminating auto-fluorescent signals and investigated the methods aimed at separating the PO_2s of the retinal and choroidal vasculature beds. In addition, the application of the enhanced PLI technology to the investigation of retinal oxygen changes in a rat model of type I diabetes yielded results that suggest a hyperoxic to hypoxic trend prior to the onset of diabetic retinopathy.

المنارة للاستشارات
www.manaraa.com

Table of Contents

www.manaraa.com

www.manaraa.com

List of Tables and Figures

www.manaraa.com

Introduction

Retinal hypoxia is associated with many retinal diseases, such as diabetic retinopathy. Current retinal research suggests that retinal hypoxia appears prior to the onset of diabetic retinopathy. The preliminary association of retinal hypoxia to the early stages of diabetic retinopathy is stimulating the development of new technologies to measure the oxygen content of retinal tissue.

Frequency domain lifetime phosphorescent imaging (PLI) is a promising technology that enables the mapping of the oxygen content across the entire retina in the form of two-dimensional images. The two-dimensional images generated from the PLI process are a spatial mapping of the retinal tissue's oxygen tension (PO_2). Currently, it is believed that the maps generated through PLI contain PO_2 information from both the retinal and choroidal vasculature. The choroidal vasculature is situated directly behind the retina and may have a considerably different oxygen tension value than the retinal vasculature. In addition, the excitation signals of PLI cause the generation of auto-fluorescent by the retinal tissue. These auto-fluorescent signals artificially inflate the oxygen tension readings due to the nature of fluorescent signals in phosphorescent imaging.

This research investigates the functionality of the PLI imaging process, addresses the problematic fluorescence contamination and investigates the feasibility of generating multi-layered oxygen tension maps of the retina. The removal of the auto-fluorescent contamination will facilitate the separation of the choroidal and retinal oxygen tension maps. Methods utilizing post-processing algorithms, dual excitation wavelengths and optical sectioning were investigated as possible solutions for isolating the choroidal and retinal oxygen tension maps. The ultimate goal of research was to develop the PLI

المنارة للاستشارات
www.manaraa.com

process to a point were the instrument can effectively be applied to diabetic retinopathy research and determine the feasibility of generating of multi-layered oxygen tension maps of the retina, that express the choroidal and retinal vasculature in separate layers of an oxygen tension map.

Specific Aims

The specific aims of this research were to:

1. Characterize and model the effects of fluorescent signals in a typical PLI measurement.
2. Eliminate the auto-fluorescence contamination to generate oxygen tension maps that better reflect the actual oxygen tension of the tissue.
3. Investigate the use of algorithms, optical sectioning, and multiple excitation wavelengths on the isolation of the retinal and choroidal vasculatures.
4. If feasible, apply the above mentioned PLI enhancements to the study of oxygen tension levels in diabetic retinas.

Significance

Oxygen delivery to metabolically active tissues is a critical parameter for tissue health and functionality. Of all mammalian tissues, the retina is the most metabolically active resulting in a high rate of oxygen consumption.[17] Certain diseases that affect the health and function of the retina, such as diabetic retinopathy, have been associated with retinal hypoxia. The investigation into oxygen's roles during the onset of this disease has become a popular topic within biomedical research.

Diabetic retinopathy is a retinal disease, which affects more than 700,000 Americans per year and is responsible for over 5,000 new cases of legal blindness per

المنارة للاستشارات
www.manaraa.com

year.[1] Diabetic retinopathy is a disease induced by diabetes that affects the retinal vasculature by degrading retinal vessels. Diabetic retinopathy is clinically diagnosed once there is evidence of micro-aneurysms and hemorrhages of the retinal vasculature. As the disease progresses, the retinal aneurysms and hemorrhages intensify and significant retinal capillary loss occurs. In the late stages of the disease, fragile new vessels proliferate across the retina in an attempt to replace ischemic vessels, but this proliferation ultimately results in the complete loss of vision. [1]

Due to the significant physiological and societal impacts of diabetic retinopathy, much research has been conducted with the intent of understanding the biological mechanisms that lead to the onset of diabetic retinopathy.[2,8,12,13,20,25] Currently these mechanisms are not fully understood but research has shown that the retina undergoes certain physiologic changes before diabetic retinopathy reaches a stage where the disease can be clinically diagnosed. In particular, Berkowitz et al. and Alder et al. have shown that the retina appears to become hypoxic before the onset retinopathy.[6,2] Additionally, Amin et al. detected the presence of vascular endothelial growth factor in the retina during the early stages of diabetic retinopathy.[4] Most importantly, the presence of hypoxia in the retina before any clinical signs of diabetic retinopathy offers insight into the development of the disease. Although the role of oxygen in the retina with respect to diabetic retinopathy is still not completely understood, retinal hypoxia is believed to correlate with the progression of the disease, either as a direct cause or as a by-product of diabetic related changes in metabolism and vasculature physiology.[2]

The need to accurately assess the oxygen levels across the retina has become significant as research continues to strengthen the correlation between retinal oxygen

www.manaraa.com

tension and various diseases, specifically diabetic retinopathy. Traditionally, oxygen tension measurements of the retina have been conducted with microelectrodes that are highly invasive and measure the PO_2 of the retina at a single location.[25] As PO_2 detection technologies continue to advance, new techniques are enabling alternative methods for specifically measuring the retina's PO_2. Techniques such as MRI based ΔPO_2 detection, spectral imaging, and fluorescence / phosphorescence imaging, are the key technologies that are revolutionizing retinal PO_2 measurements.[6, 5,9,10,19] In particular, frequency domain phosphorescence lifetime imaging is a developing technology that possess great potential for determining the oxygen content of the retina in a non-invasive and accurate manner.

Frequency domain phosphorescent lifetime imaging will likely revolutionize oxygen measurements in the retina because the oxygen levels across the entire retina can be accurately recorded and displayed in a two dimensional oxygen map through a single measurement. The application of PLI to diabetic retinopathy and other oxygen related retinal diseases is two fold. PLI can be applied towards researching the progression of certain retinal diseases and their relation to retinal oxygenation. Through examination of the progression of retinal diseases using a robust oxygen mapping system, such as PLI, the link between the disease stages and changes in retinal physiology, such as hypoxia, can be solidified. In addition, the development of genetic knockout animal models for diabetes, which are good models of human diabetes, will allow for the oxygen differences between normal retinas and diabetic retinas to be quantified. Once a relation between the diseased and non-diseased states is established, the PLI system can be used as a diagnostic tool for detecting diabetic retinopathy. When the technology becomes

المنارة للاستشارات
www.manaraa.com

appropriate for use in humans, PLI can be used in a clinical manner to identify retinal changes before the clinical onset of diabetic retinopathy and allow adequate time for proper treatment to delay the disease progression.

Although the current capabilities of frequency domain phosphorescence lifetime imaging technologies, such a Shonat et al.'s PLI system, are functional and are currently being used for diabetic retinopathy research, further development and refinement of these systems is required for the consistent generation of highly accurate retinal oxygen maps. Important to the understanding of this research is an overview of the fundamental theory behind Shonat et al.'s PLI system, current limitations and problematic areas, and methods for investigation solutions to the current limitations of the system and enhancing the PLI system through the generation of multi-layered oxygen tension maps. [19]

Theory / Background

Phosphorescence and Fluorescence

Critical to the understanding of the PLI system are the physics of phosphorescence and fluorescence light emissions. Light is emitted from a substance in two different ways, fluorescence and phosphorescence. Both fluorescence and phosphorescence result from photon emission from an electronically excited molecular state. The difference between fluorescence and phosphorescence exists primarily from the different orbital transitions of the excited electron. Fluorescence emission is a product of the electron's return to the ground state (S_0) from the excited S_1 electron state, while phosphorescence is created by the transition of the excited electron to the first

www.manaraa.com

triplet state (T_1), termed intersystem crossing followed by decay from T_1 back to the ground state. These orbital transitions are diagramed in Figure 1

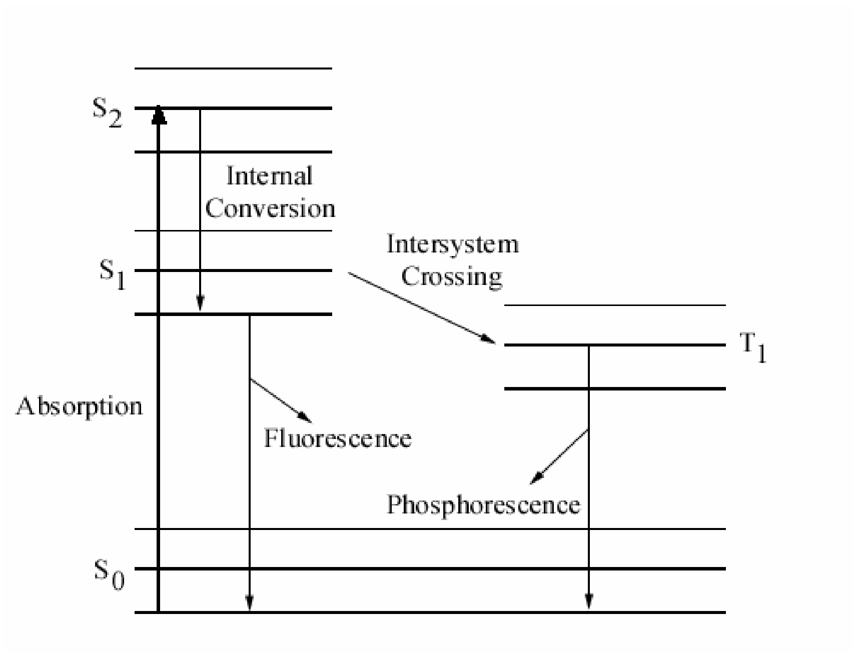


Figure 1: Fluorescence and Phosphorescence Emission

The orbital transitions of the excited electron dictate the lifetimes of the emitted light. The most important difference between fluorescence and phosphorescence in terms of PLI, are the lifetimes of each type of emission. Fluorescence emission lifetimes are on the order of nanoseconds (10^{-9}) while phosphorescence lifetimes are on the order of milliseconds (10^{-3}).[11]

Phosphorescence Lifetime Imaging

The theory behind the PLI system is based on oxygen dependent quenching of phosphorescence. Central to the functionality of this technology is a palladium porphyrin probe, Pd-*meso*-tetra [4-carboxylphenyl] porphine, which is bound to albumin and injected into the blood stream to generate a phosphorescence signal.[14] The phosphorescent signal is generated when the probe is excited with 412 nm and 524 nm

المنارة للاستشارات

www.manaraa.com

light and emits a ~700 nm phosphorescent signal. Oxygen levels in the blood stream are inversely related to the intensity of the phosphorescent signal generated by the probe. Oxygen quenches the phosphorescence of the probe, thus lower oxygen levels produce a more intense phosphorescent signal.[14] It is important to note that oxygen is the only significant quencher for the probe in the blood stream, making the phosphorescence intensity solely dependent upon the surrounding PO_2. The phosphorescence emission is an exponential decay, described by:

$$I(t) = I_0 e^{\frac{-t}{\tau}} \qquad [1]$$

where *I(t)* is the phosphorescence intensity, I_0 is the initial intensity, and τ is the lifetime of the phosphorescent decay. Increased oxygen levels in the blood stream decrease the lifetime of the phosphorescence intensity (τ). The lifetime of phosphorescence emitted by the probe is related to the oxygen tension surrounding the probe through the Stern-Volmer equation.

$$\frac{\tau_0}{\tau} = 1 + K_q \tau_0 PO_2 \qquad [2]$$

τ_0 is the initial phosphorescence in the absence of oxygen, K_q is the bimolecular-quenching constant for the probe, and PO_2 is the concentration of the quenching agent. From these two relations the PO_2 of a tissue can be determined using the value of τ which is experimentally derived.

In frequency domain phosphorescence imaging, τ is determined by exciting the probe using sinusoidally modulated light. The sinusoidal excitation light will cause a phosphorescence emission that is also sinusoidal but shifted by a phase angle θ and modulated by a factor *m*. Figure 2 displays the relationships between the PLI excitation

www.manaraa.com

signal and emission. Theta (θ) is the phase delay of the emission signal while modulation is simply:

$$m = \frac{B/A}{b/a} \qquad [3]$$

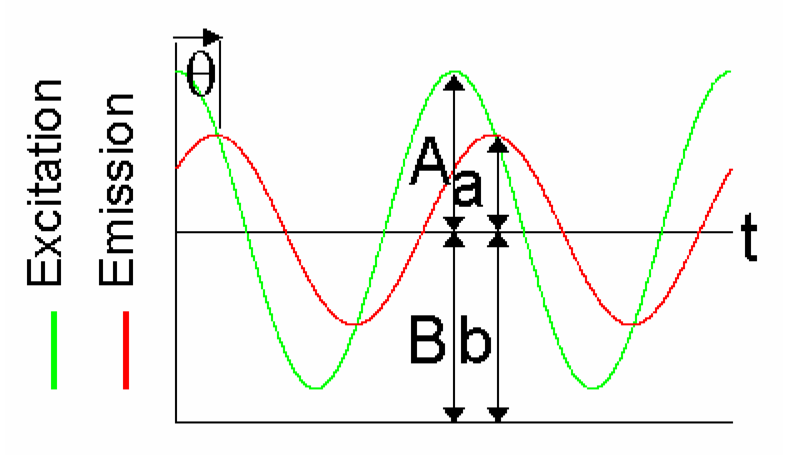


Figure 2: PLI Phase and Modulation

PLI uses a CCD to acquire the intensity of the phosphorescence emission in an image format. The intensifier of the CCD is used to modulate the sensitivity of the CCD enabling detection of the emission signal's phase and modulation. The intensity at each pixel of the CCD is represented by:

$$I(\theta_D) = k[Pd](1 + \frac{1}{2} m_D m \cos(\theta - \theta_D)) \qquad [4]$$

where θ_D is the phase shift of the CCD intensifier, m_D is the modulation profile of the CCD intensifier, *k* is a constant and [Pd] is the concentration of the probe. By acquiring an image set across an intensifier phase shift from 0 to 2π, where each image represents an incremental phase delay of the CCD intensifier's gating, τ can be determined from both the phase shift and modulation of the phosphorescence signal. The lifetime of the phosphorescent decay is related to phase shift by:

$$\tan\theta = \omega \cdot \tau_\theta \qquad [5]$$

المنارة للاستشارات
www.manaraa.com

where ω represents the modulation frequency and τ_θ represents the apparent phase calculated lifetime. The apparent lifetime of the phosphorescent signal is related to the modulation (*m*) as well.

$$m = (1 + \omega^2 \cdot \tau_m^2)^{-1/2} \qquad [6]$$

where τ_m is the apparent modulation calculated lifetime.

In order to enable efficient use of computer algorithms for processing the intensity data recorded by the CCD, a linear form of the equation relating the phase and modulation of the phosphorescent signal to the intensity of the phosphorescent signal must be used. Lakowicz *et al* developed a linear form of the equation, which requires three fitting parameters (a_0, a_1, and b_1) to be determined from the recorded intensity.[10] The relationship shown in equation 7 describes the measured intensity data in terms of phase, which is acquired by cross correlating the CCD's intensifier sensitivity with the phosphorescence emission. The data described by this relationship is referred to as the "cross correlation signal".

$$I(\theta_D) = a_0 + a_1 \cdot \cos(\theta_D) + b_1 \cdot \sin(\theta_D) \qquad [7]$$

Once the three fitting parameters are determined from the phosphorescent intensity signal, the phase and modulation of the phosphorescent signal can be determined. The phase is represented by:

$$\theta = \tan(\frac{b_1}{a_1})^{-1} \qquad [8]$$

and the modulation is represented by:

$$m = \frac{\sqrt{a_1^2 + b_1^2}}{a_0} \qquad [9]$$

المنارة للاستشارات
www.manaraa.com

In practice, conducting a retinal oxygen measurement consists of exciting the phosphorescent probe *in vivo* by exposing the retinal tissue to a sinusodially modulated excitation light and recording the phosphorescent emission using a CCD with an intensifier modulated at the same frequency as the excitation light. The phase and the modulation of the phosphorescent emission are determined through acquisition of a series of images with the CCD intensifier signal phase delayed with respect to the excitation signal. A minimum of three images, each of different phase delays, is required to determine the fitting parameters of equation [8], (a_0, a_1, and b_1) for a given measurement. The average intensity from a region of interest of the images is then used to determine the fitting parameters of equation [8]. As mentioned above, once the fitting parameters are determined, the phase and the modulation of the phosphorescent signal can be determined, ultimately allowing the calculation of the oxygen tension for that specific region.

Figure 3 shows the current PLI instrumentation setup. The primary components of the PLI system include a xenon arc lamp and monochromator based light source (Photon Technology International SID-101) coupled with an optical chopper (Photon Technology International OC-4000), a traditional fluorescence microscope (Nikon E600) with appropriate excitation and emission chromatic filters, and a digital CCD camera (Princeton Instruments IMAX CCD with a Gen III intensifier) with sensitivity gating performed by a precision delay generator (Stanford Research Systems DG535). Within the PLI instrumentation diagram (Figure 3), the green path represents the excitation light at 524 nm and the red path represents the phosphorescence emission at 700 nm. Further

www.manaraa.com

information describing the instrumentation set up and optimization techniques for Shonat's PLI system are documented elsewhere.[19]

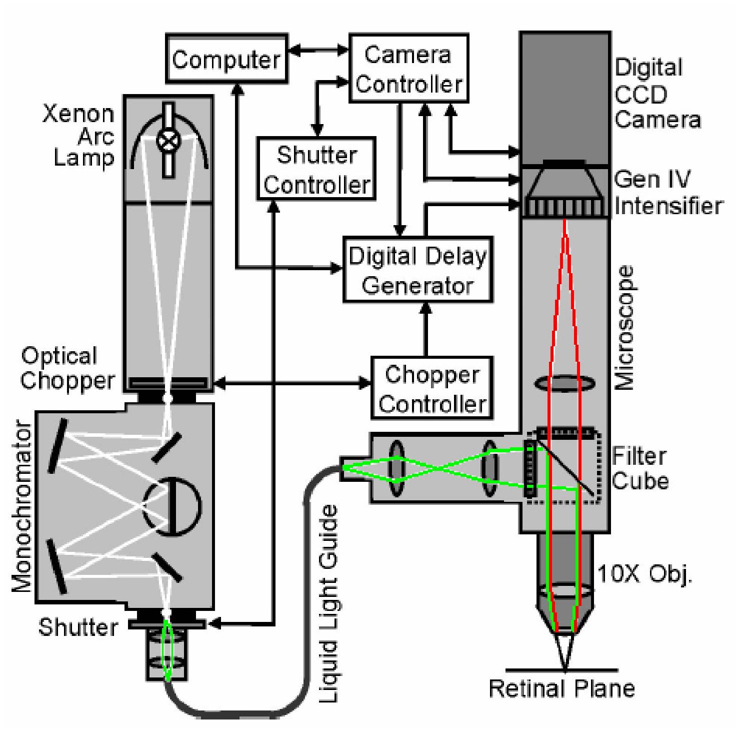


Figure 3: PLI Instrumentation

In addition to point measurements of specific regions of interest, two-dimensional maps representing the oxygen content across the retina can be generated. The two dimensional (2-D) oxygen tension maps are generated in a similar fashion as the point measurement, except the described calculations are performed on a pixel-by-pixel basis, allowing the derivation of a PO_2 value for each pixel of the image. The resultant PO_2 maps serve as a representation of the PO_2 information across the entire retina. Figure 4 portrays typical phase and PO_2 maps generated from a PLI measurement. These maps are a spatial representation of the phase and PO_2 across the retina. The phase and PO_2 information is encoded in the color of the image. In the phase shift map, the color ranges from black to light yellow where black represents 0 degrees of phase shift while light

www.manaraa.com

yellow represents 40 degrees of phase shift. Similarly, in the retinal PO_2 map the color also ranges from black to light yellow where black represents a PO_2 of 0 mmHg while a light yellow represents a PO_2 of 140 mmHg.

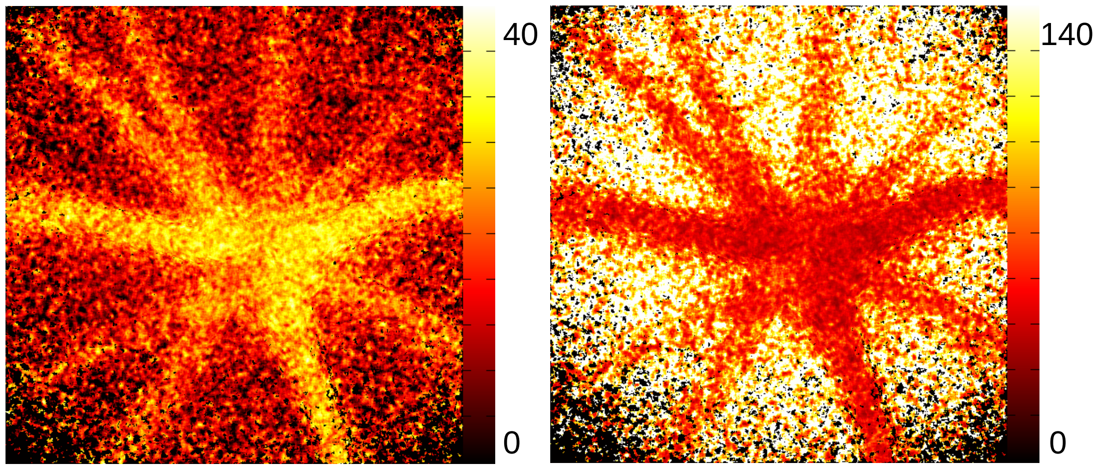


Figure 4: Phase shift (left) and PO_2 (right) maps of the retina

Although the current PLI system is able to generate oxygen tension maps of the retina, the PLI instrument has a difficult time producing highly accurate PO_2 values in the retina. Two of the primary contaminating factors that prevent accurate calculations of PO_2 are fluorescence and reflected excitation signals.

Fluorescence and Reflected Excitation Signals

Fluorescent signals and reflected excitation light reduce the accuracy of oxygen tension images generated through frequency domain phosphorescence lifetime imaging techniques. It is believed that current retinal oxygen maps suffer from an over abundance of induced fluorescence signals and reflected excitation light. Although these signals are generated from the surrounding retinal tissue and not by the phosphorescent probe, the current PLI system cannot discern between phosphorescence and fluorescence emissions. As previously discussed, fluorescent signals naturally have much shorter emission lifetimes when compared to phosphorescence signals, due to different orbital transitions.

www.manaraa.com

The extremely short lifetime of fluorescent signals, with respect to the PLI system, cause fluorescent signals to experimentally have a measured phase shift of zero. Currently, the PLI system does not have the resolution to measure fluorescence emission phase shifts due to the excitation frequencies used. Excitation frequencies would have to be in the MHz range to measure a fluorescence phase shift different from zero. This means that the PLI system is sensitive only to phosphorescence emissions and all fluorescence signals appear the same and have a measured phase shift of zero. Reflected excitation light from the PLI system's light source also produces a zero-phase signal because it is, in all respects, the same as the original excitation signal. Although the fluorescence microscope has optical filters in place to reduce excitation light from reaching the CCD camera, these filters are not efficient enough to prevent all of the excitation light from reaching the CCD camera. Additionally, fluorescence signals generated by the retinal tissue can have the same wavelength as the desired phosphorescence emission (~700 nm), preventing the use of optical filters as a means to reduce the effect of zero-phase contamination.

Zero-phase signals differ from the probe's phosphorescent signal in two general ways. First, the zero-phase signal has no phase delay with respect to the excitation signal, hence the name "zero-phase" signal. Secondly, the modulation of the zero-phase signal is equal to one, meaning that the ratio of the signal's baseline offset to half of the signals amplitude is the same as the ratio for the excitation signal, essentially they are the same signal (refer to equations [5-9]). These two properties of the zero-phase signal become a problem when the zero-phase signal combines with the phosphorescent signal. The combination of zero-phase signals with a legitimate phosphorescence signal causes

المنارة للاستشارات

www.manaraa.com

the phosphorescence signal to become contaminated, by altering its phase and modulation. The details of how zero-phase signals contaminate a phosphorescence signal are described in detail in the section of this document titled “Zero-Phase Signal Model”. Throughout this document, recorded fluorescence emissions and reflected excitation light are referred to as “zero-phase signals” or as “zero-phase contamination”.

PO_2 Mapping of the Retina using PLI

PLI is a technique that can be used to investigate the oxygen content of many different types of tissues. Some applications of PLI include using PLI to determine the PO_2 of subcutaneous tissue.[21] This research focuses on the application of PLI to develop PO_2 maps exclusively of the retina. Specific to the application of PLI to the retina, are imaging problems related specifically to the vasculature structure of the retina. In particular, it is theorized that the PLI process captures PO_2 information from both retinal vasculature and choroid in unknown proportions.

Retinal Vasculatures

The eye is a fluid filled spherical structure that captures light from the outside environment and passes it through the layers of the retina to photosensitive cells. Through visual processing within the retina, light is transformed into a complex series of signals that allow an image to be perceived by the brain via the optic nerve.

Anatomically, the retina can be divided into three cellular layers and two synaptic layers as seen in Figure 5. The three cellular layers consist of the outer nuclear layer (ONL), the inner nuclear layer (INL) and the ganglion cell layer (GCL).[7] The blood supply for the retina is delivered through two separate mechanisms, the retinal vessels and the choroid.[3] The retinal vessels are situated primarily in the inner nuclear layer while

المنارة للاستشارات
www.manaraa.com

the choroid is situated behind the photoreceptors. The retinal vessels consist of low flow rate arteries, veins and capillary beds, which nourish the ganglion and inner nuclear layer. In contrast, the choroid is a high flow rate vasculature system, which nourishes the outer nuclear layer, primarily the highly metabolically active photoreceptors. Traditionally, the retinal vessels experience a low venous saturation while the choroid experiences high venous oxygen saturation, due to the differences in flow rate.[3]

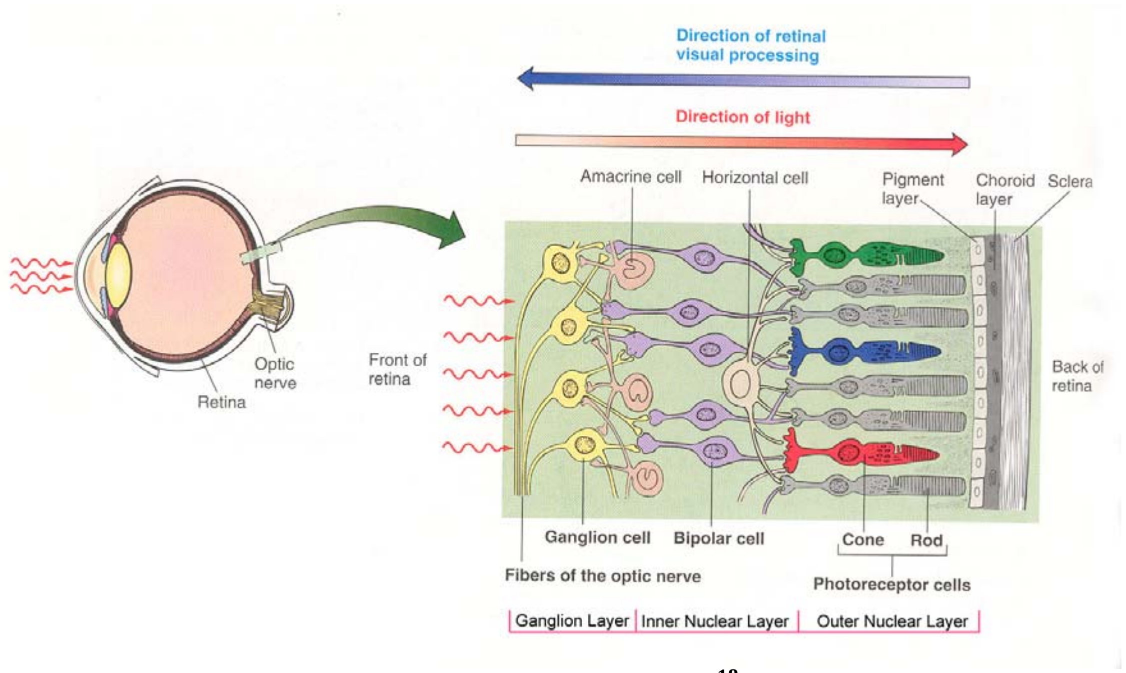


Figure 5: Retinal Anatomy[18]

The differences in physiology, specifically flow rate, between the two major vasculature systems of the retina establish the potential to complicate retinal PO_2 mapping.

Retinal and Choroidal PO_2 Mapping

Although Shonat et al's current PLI system[19] generates 2-D PO_2 maps of the retinal surface it is believed that phosphorescence signals produced in the systemic vasculature behind the retina, the choroid, are combining with the phosphorescent signals

المنارة للاستشارات
www.manaraa.com

from the retinal vasculatures. This combination of phosphorescent signals is problematic. The choroid naturally has a higher PO_2 than the retinal vasculature because it is the primary source of oxygen for most retinal cells.[3] In addition, the choroidal blood flow is very similar to systemic blood flow while the retinal vessels maintain a very low flow rate, allowing the retinal vasculature to typically have a PO_2 much lower than the PO_2 of the choroid. The combination of phosphorescent signals from retinal vasculature and the choroidal vasculature may cause the apparent PLI measured retinal PO_2 to be artificially shifted towards the PO_2 of the choroid.

In order to enhance the accuracy and increase the reported information of a PLI measurement of the retina, this research investigates the extraction and isolation of choroidal and retinal phosphorescent signals for the generation of multi-layered PO_2 maps of the retinal and choroidal vasculature. If feasible, the multi-layered PO_2 maps will contain the PO_2 values for the choroid and the retina but in separate layers of the image. The construction of these multi-layered PO_2 maps is based on the hypothesis that the choroidal and retinal vasculatures express fundamentally different oxygen tension values that ultimately lead to the generation of different phosphorescent signals during frequency domain phosphorescent lifetime imaging. One intent of this research is to investigate the feasibility of the separation of the PLI derived PO_2 information from the retinal and choroidal vasculatures and express the separated information in a multi-layered PO_2 map of the retina.

Methods and Results: Zero –Phase Signal Removal

The current PLI system does not produce accurate PO_2 maps because of two distinct problems. The first problem is caused by zero-phase signals created by auto

المنارة للاستشارات
www.manaraa.com

fluorescence generated in the retinal tissue and by scattered excitation light. The second problem is the combination of phosphorescence emissions from the retinal and choroidal vasculatures of the eye. In order to investigate the development of multi-layered retinal PO_2 maps the zero-phase signals must first be removed from the PLI measurement.

PLI Instrument Simulator

In order to assist the elimination of zero-phase signals and the investigation of the isolation of retinal and choroidal oxygen maps, the PLI instrument was mathematically simulated in Matlab (Mathworks, Inc.). The PLI simulator was created to produce theoretical simulations of the interactions of zero-phase signals with a phosphorescent signal, the interactions of multiple phosphorescent signals (retinal and choroidal), and to serve as a test bed for new PLI data processing algorithms. In addition, refinement of the PLI simulator extended its capabilities to allow mathematical modeling of the PLI instrument using alternative hardware setups, such as LED based excitation instead of arc lamp and optical chopper based excitation.

المنارة للاستشارات
www.manaraa.com

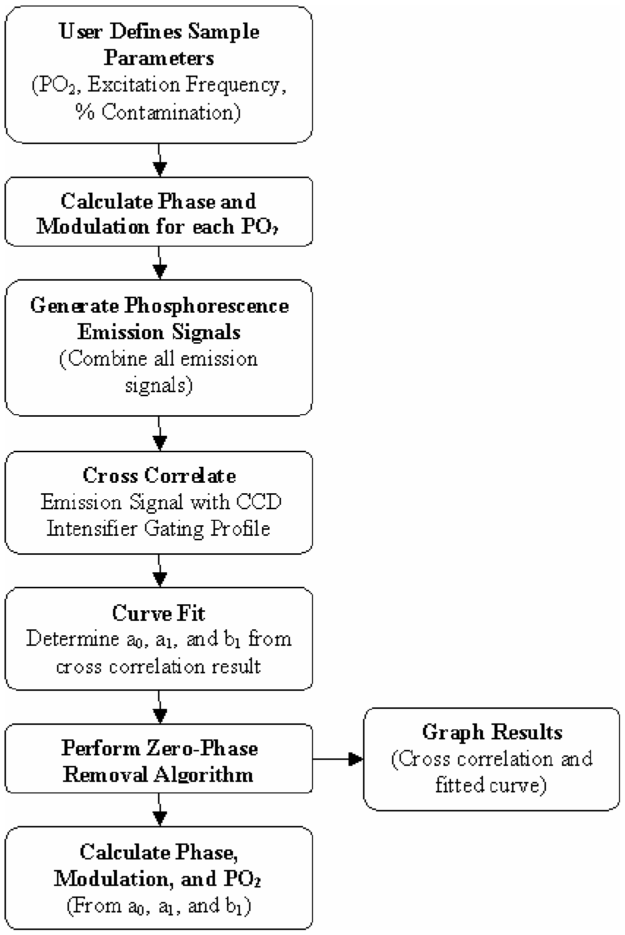

User Defines Sample Parameters
(PO2, Excitation Frequency, % Contamination)
Calculate Phase and Modulation for each PO2
Generate Phosphorescence Emission Signals
(Combine all emission signals)
Cross Correlate
Emission Signal with CCD Intensifier Gating Profile
Curve Fit
Determine a0, a1, and b1 from cross correlation result
Perform Zero-Phase Removal Algorithm
Graph Results
(Cross correlation and fitted curve)
Calculate Phase, Modulation, and PO2
(From a0, a1, and b1)

Figure 6: PLI Simulator Flow Chart

Figure 6 shows the sequence of steps that the PLI simulator uses to mathematically replicate the function of the actual PLI instrument. The simulator allows the user to enter two distinct target PO_2's and their corresponding mixing ratio (i.e. 40% PO_2 I and 60% of PO_2 II). The incorporation of two target PO_2s was intended to simulate the PO_2 contributions from the retinal vasculature and the choroidal vasculature. The simulator calculates the individual emission signals for both of the target PO_2 values and then combines these individual emission signals into one emission signal prior to cross correlation with the CCD intensifier profile. Following cross correlation, the fitting parameters a0, a1, and b1 are determined from the cross correlation data and a single PO_2

المنارة للاستشارات
www.manaraa.com

value is returned. This mathematical model represents the way the actual PLI instrument would handle measurement of a target consisting of two distinct PO_2 values. The PLI simulator allows the theoretical investigation of how two different PO_2 emission signals combine in the PLI process to produce a single PO_2 and serves as a test platform for algorithms that are aimed to enhance the post-processing of phosphorescence image data.

In addition to multiple PO_2 samples, zero-phase contamination capabilities were incorporated into the PLI simulator once an appropriate model of zero-phase contamination was derived. The capability to contaminate a phosphorescence emission signal with a zero phase signal allowed for the theoretical investigation of how zero-phase signals effect the calculation of a target's PO_2. Also, zero-phase removal algorithms could be theoretically tested using the PLI simulator. The source code for the PLI simulators is documented in appendix A.

Once the PLI simulator was constructed, it served as a highly effective tool for learning intricate mathematics of the system and provided informative visual information (charts and graphs) describing the dynamics of the instrument. Throughout this research the PLI simulator was used to answer many questions about the nature of the instrument.

Zero-Phase Contamination Model

In order to generate an effective algorithm directed at eliminating the zero-phase contamination, a valid model of the zero-phase contamination was developed. The zero-phase signal model was based on observations gathered from imaging white paper and a cuvette filled with distilled H_2O. The white paper essentially reflected much of the excitation light back into the microscope objective and without the any optical filters in place the reflected light was recorded by the CCD. Analysis of the reflected light data

المنارة للاستشارات
www.manaraa.com

allowed the development of a good understanding of how reflected light was interpreted in a PLI measurement.

In the time domain a zero-phase signal is simply:

$$I(t) = \cos(t) \qquad [10]$$

Where I is the intensity of the signal and t is time. Once the zero-phase signal is cross-correlated with the sinusoidally modulated sensitivity of the CCD's image intensifier, the zero-phase signal is expressed in the phase domain and is represented as follows:

$$I(\theta_D) = a_{0s} + a_{1s}\cos(\theta_d) + b_{1s}\sin(\theta_D), \qquad [11]$$

where θ_D is the intensifier's phase delay, and a_{0s}, a_{1s}, and b_{1s} are fitting parameters for the zero-phase cross correlated signal. Analysis of this relationship shows that $b_{1s} = 0$ and $a_{0s} = a_{1s}$ because characteristically a zero-phase signal has the following properties:

$$\theta_s = \tan^{-1}(\frac{b_{1s}}{a_{1s}}) = 0 \text{ and } m_s = \frac{\sqrt{a_{1s}{}^2 + b_{1s}{}^2}}{a_{0s}} = 1 \qquad [12,13]$$

These relationships allow the cross correlation signal for the zero-phase signal to be expressed as:

$$I(\theta_D) = S + S\cos(\theta_d)\text{, where } a_{0s} = a_{1s} = S \qquad [14]$$

During a PLI measurement, the zero-phase signal is combined with the phosphorescence emission signal. This combination can be mathematically represented as addition. So when the two signals are combined and then cross-correlated with the image intensifier, the calculated fitting parameters a_{0c}, a_{1c}, and b_{1c}, become a superposition of the fitting parameters of the zero phase signal ($a_0 = S$, $a_1 = S$, and $b_1 = 0$) and the fitting parameters of the phosphorescent signal (a_0, a_1, and b_1). The cross correlated combination of the zero-phase signal with the true phosphorescent signal is expressed as:

www.manaraa.com

$$I(\theta_D) = [a_0 + S] + [a_1 + S]\cos(\theta_d) + b_1 \sin(\theta_D) \qquad [15]$$

As mentioned above, a_0, a_1, and b_1 represent the fitting parameters for the true phosphorescent signal without zero-phase contamination.

The resultant combined signal (defined by equation 15) has a smaller phase delay and larger modulation than the true phosphorescent signal. The reduced phase delay and larger modulation, with respect to oxygen tension imaging, artificially increases the measured PO_2 of the sample. In addition, as the excitation frequency increases, the modulation of the phosphorescent signal decreases. This decrease in modulation of the phosphorescent signal allows the zero-phase signal to dominate the "combined" zero-phase and phosphorescent signals. The effects of phosphorescence emission contamination due to zero-phase signals were verified through mathematical simulation. A mathematical model was developed in Matlab to investigate the effects of zero-phase contamination across a range of excitation frequencies (PLI_BS_Spectrum.m, refer to Appendix B – PLI Contamination "Excitation Spectrum Analyzer" for the source code). Figure 7 displays the result of a simulation performed with PLI_BS_Spectrum. In this simulation, the target PO_2 was defined as 27 mmHg and zero-phase contamination was added to the emission signal. As seen in Figure 7 the "true PO_2" (uncontaminated) produces the same PO_2 value (27 mmHg) for all excitation frequencies, while the "contaminated PO_2" phosphorescence emission does not produce the correct PO_2 value under traditional analysis. In addition, the PO_2 value of a contaminated phosphorescence emission signal increases non-linearly with excitation with frequency.

المنارة للاستشارات
www.manaraa.com

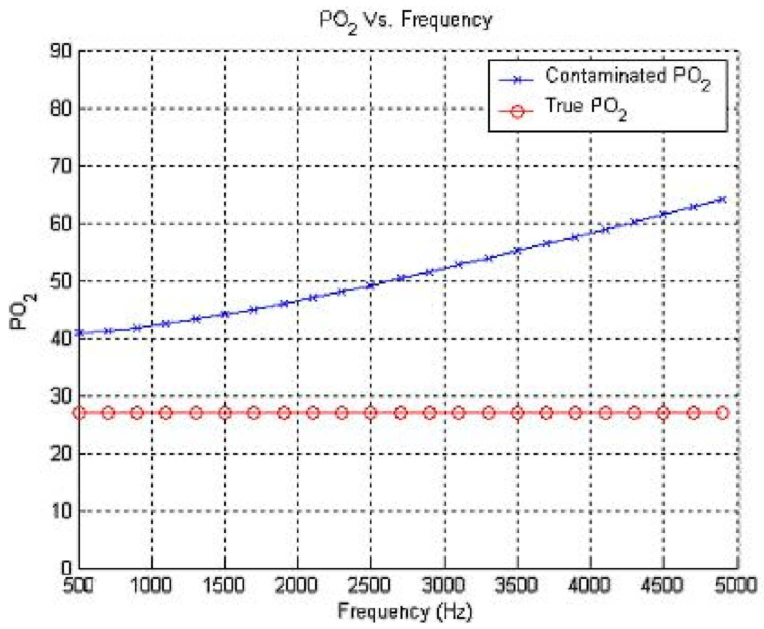


Figure 7: Frequency dependency of a zero-phase contaminated PO$_2$

Observations from non zero-phase corrected *in vivo* PLI studies show that calculated PO$_2$ values are dependent upon modulation frequency in the same manner depicted by the mathematical simulation (see Figure 8). This *in vivo* observation suggests that the simulation accurately models zero-phase contamination in PLI measurements and that the *in vivo* PLI measurements are zero-phase contaminated as theorized.

www.manaraa.com

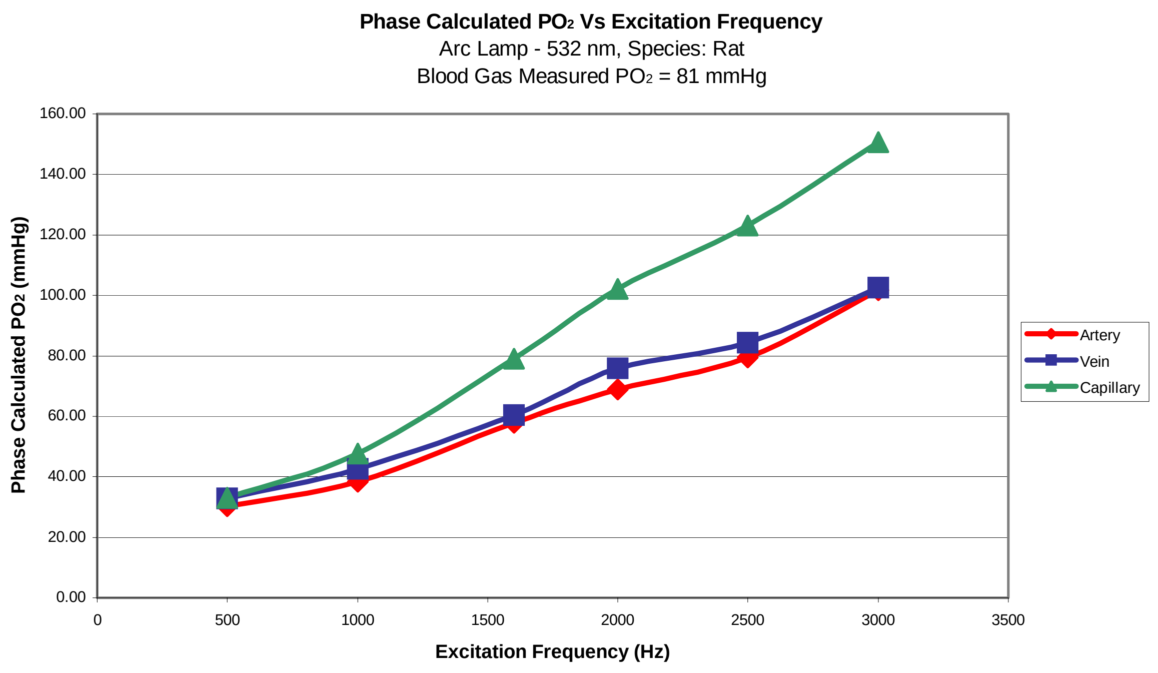


Figure 8: *In Vivo* PO$_2$ Dependency On Excitation Frequency

In vivo observations confirm that the simulation accurately models zero-phase contamination in PLI measurements and that the *in vivo* PLI measurements are zero-phase contaminated as theorized.

Zero-Phase Signal Removal

Current zero-phase reduction methods utilize optical filtering and shutter / intensifier timing techniques to reduce the effects of the zero-phase signal.[16] Analysis of the zero-phase signal mathematics and the usage of two excitation frequencies have led to the development of an algebraic removal of zero-phase signal contamination.

In order to eliminate the "S" term from relationship show in equation 15, image sets must be acquired at two different excitation frequencies (ω_1 and ω_2), producing:

$$I(\theta_D)_{\omega 1} = [a_{0_{\omega 1}} + S] + [a_{1\omega 1} + S]\cos(\theta_d) + b_{1\omega 1}\sin(\theta_D) \qquad [15]$$

$$I(\theta_D)_{\omega 2} = [a_{0\omega 2} + S] + [a_{1\omega 2} + S]\cos(\theta_d) + b_{1\omega 2}\sin(\theta_D) \qquad [16]$$

www.manaraa.com

where the notation ω_1 and ω_2 corresponds to the image sets acquired the different frequencies, ω_1 and ω_2. Through manipulation of equations 2-9, a relationship can be derived that relates the fitting parameters ω, a_1, and b_1 for both frequencies:

$$\frac{a_{1\omega 1}}{a_{1\omega 2}} = \frac{b_{1\omega 2} \times \omega_1}{b_{1\omega 1} \times \omega_2} = R \qquad [17]$$

This relationship can be solved for the ratio of $a_{1\omega 1}$ and $a_{1\omega 2}$ from known information. A new relationship can now be established that can be solved for S:

$$\frac{U_2 - S}{U_1 - S} = R \text{, where } U_2 = (a_{0_{\omega 2}} + S) \text{ and } U_1 = (a_{0_{\omega 1}} + S) \qquad [18]$$

$$S = \frac{RU_1 - U_2}{R - 1} \qquad [19]$$

Subtracting S from both a_1, and b_1 essentially removes the zero-phase contamination and produces the fitting parameters for the true phosphorescent signal.

$$I(\theta_D)_{\omega 1} = a_{0_{\omega 1}} + a_{1\omega 1}\cos(\theta_d) + b_{1\omega 1}\sin(\theta_D) \qquad [20]$$

$$I(\theta_D)_{\omega 2} = a_{0\omega 2} + a_{1\omega 2}\cos(\theta_d) + b_{1\omega 2}\sin(\theta_D) \qquad [21]$$

This mathematical elimination of the zero-phase signal only requires two image sets to be acquired at two different frequencies. Additional hardware, optical or electronic, is not required. The mathematics of this elimination process are also linear and rather simplistic so computation time required for algorithm analysis is minimal. A mathematical simulation of the elimination of zero-phase signals proved the ability of the algorithm to completely remove all contaminating zero-phase signals.

www.manaraa.com

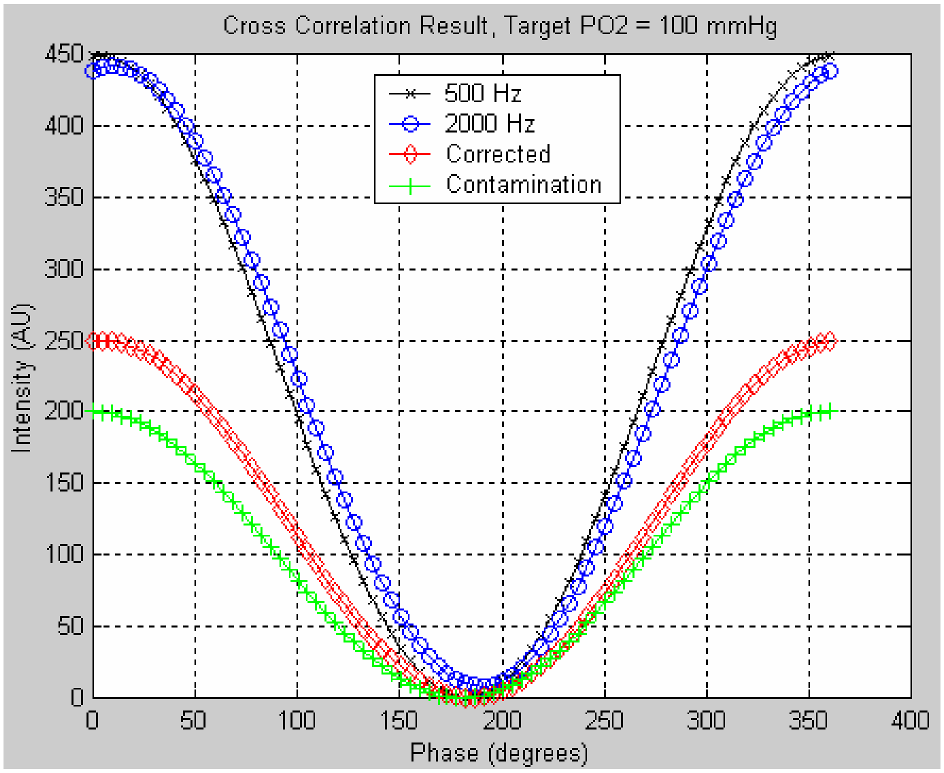


Figure 9: Implementation of Zero-Phase Removal Algorithm

Figure 9 shows the result generated by the PLI simulator when contamination was added to a phosphorescence emission signal (target PO_2 = 100 mmHg). The 500 Hz signal represents the "contaminated" cross correlation result for an excitation frequency of 500 Hz and the 2000 Hz signal represents the "contaminated" cross correlation result for an excitation of 2000 Hz. The "corrected" cross correlation signal represents the corrected phosphorescence emission signal produced by the zero-phase removal algorithm. The "contamination" cross correlation signal is the contaminating zero-phase signal determined by the zero-phase removal algorithm.

www.manaraa.com

	Measured PO_2	Target PO_2	Difference
500 Hz	183.8 mmHg	100 mmHg	83.8 mmHg
2000 Hz	191.7 mmHg	100 mmHg	91.7 mmHg
“Corrected”	100 mmHg	100 mmHg	0.0 mmHg
“Contamination”	∞	100 mmHg	∞

Table 1: Theoretical zero-phase removal algorithm performance

Table 1 displays the calculated PO_2 values for the PLI simulation conducted for the cross correlation data displayed in Figure 9. The data in Table 1 shows that under theoretical conditions, the zero-phase removal algorithm completely removes all zero phase contamination and allows for the calculation of the correct target PO_2 (compare “corrected” data to “500 Hz” and “2000 Hz ” data). The “contamination” data, determined by the zero-phase removal algorithm, produces a calculated PO_2 of infinity because its emission phase is zero and its modulation is one.

PLI Software Modification: Retina2

In order to use the zero-phase removal algorithm on with image sets acquired from the PLI instrument two image sets acquired from different excitation frequencies must be processed at the same time. The “Retina” software that was developed by Shonat et al. to process the raw intensity images was designed to handle only one image set at a time, corresponding to one excitation frequency. To make use of the zero-phase removal algorithm, the entire “Retina” analysis software package had to be redeveloped to accommodate two image sets of different frequencies. The enhanced version of the Retina analysis program was called Retina2.

المنارة للاستشارات

www.manaraa.com

Aside from incorporating the zero-phase removal algorithm and allowing the Retina analysis program to simultaneously process two image sets, the functionality of Retina2 would be the same as the Retina analysis program.

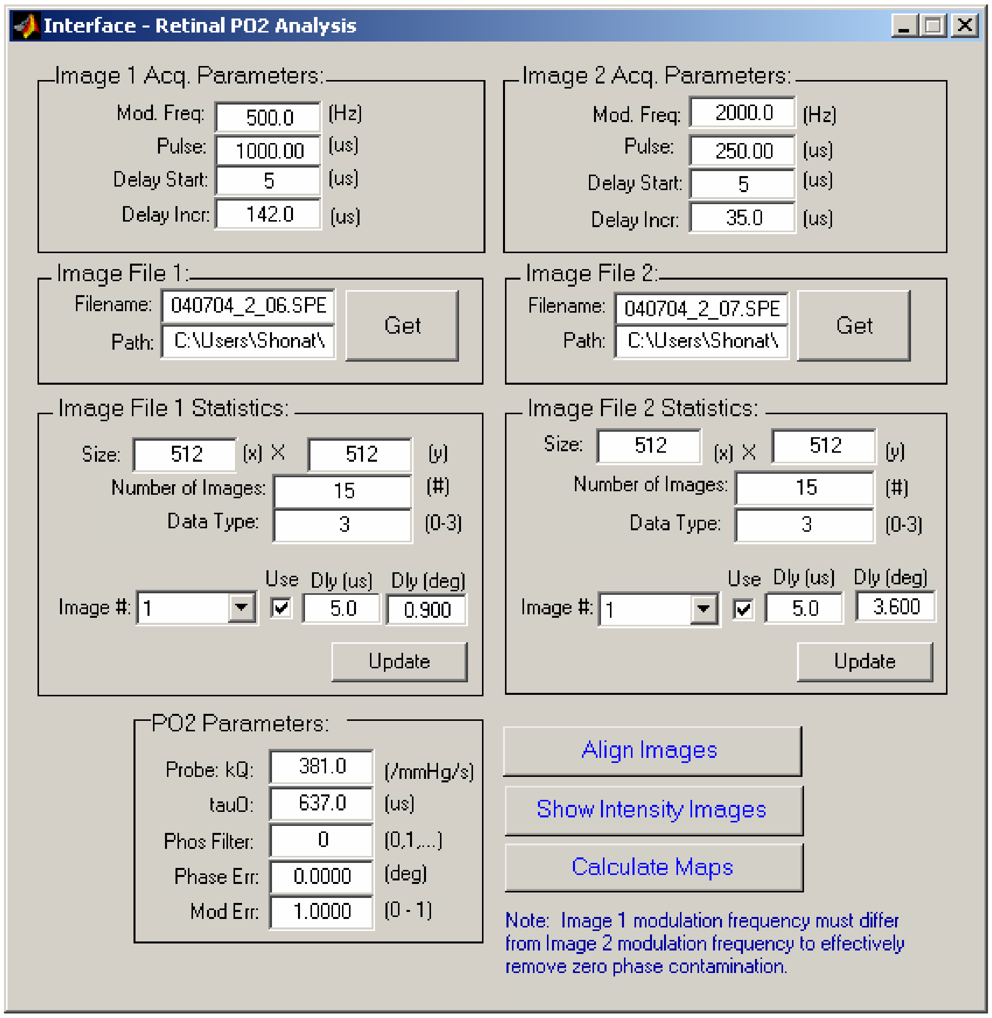

Interface - Retinal PO2 Analysis

Image 1 Acq. Parameters:
Mod. Freq: 500.0 (Hz)
Pulse: 1000.00 (us)
Delay Start: 5 (us)
Delay Incr: 142.0 (us)

Image 2 Acq. Parameters:
Mod. Freq: 2000.0 (Hz)
Pulse: 250.00 (us)
Delay Start: 5 (us)
Delay Incr: 35.0 (us)

Image File 1:
Filename: 040704_2_06.SPE
Path: C:\Users\Shonat\

Image File 2:
Filename: 040704_2_07.SPE
Path: C:\Users\Shonat\

Image File 1 Statistics:
Size: 512 (x) X 512 (y)
Number of Images: 15 (#)
Data Type: 3 (0-3)
Image #: 1 Use ✓ Dly (us) 5.0 Dly (deg) 0.900

Image File 2 Statistics:
Size: 512 (x) X 512 (y)
Number of Images: 15 (#)
Data Type: 3 (0-3)
Image #: 1 Use ✓ Dly (us) 5.0 Dly (deg) 3.600

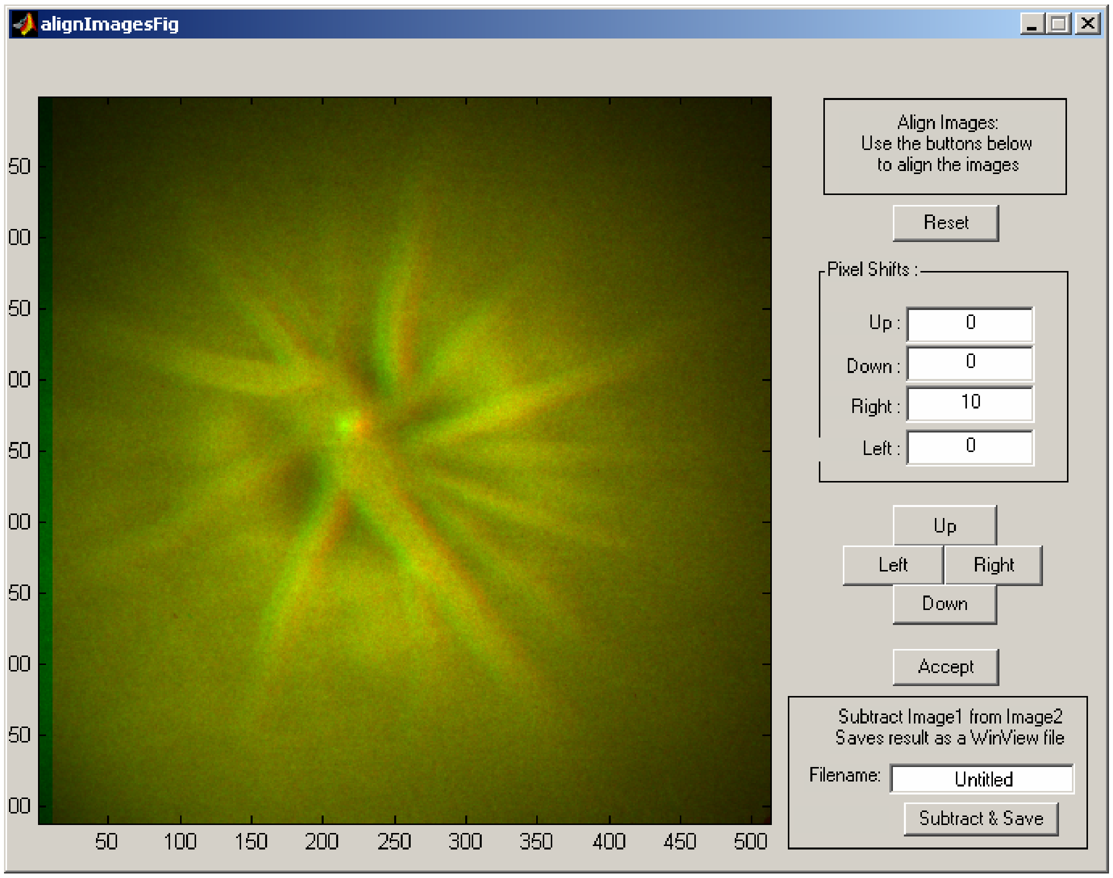
PO2 Parameters:
Probe: kQ: 381.0 (/mmHg/s)
tau0: 637.0 (us)
Phos Filter: 0 (0,1,...)
Phase Err: 0.0000 (deg)
Mod Err: 1.0000 (0 - 1)

Note: Image 1 modulation frequency must differ from Image 2 modulation frequency to effectively remove zero phase contamination.

Figure 10: Retina2 Main Screen

Figure 10 shows the main interface of the Retina2 program. From this interface, the user can select the two image sets, acquisition parameters for each image set and set global parameters for the PO_2 analysis (ie. kQ and τ_0). Also, this interface allows the user to select to align the two image sets, display the raw phosphorescence (intensity) images, or calculate the maps (PO_2, phase, modulation, etc…).

www.manaraa.com


alignImagesFig

Align Images:
Use the buttons below
to align the images

Pixel Shifts :
Up : 0
Down : 0
Right : 10
Left : 0

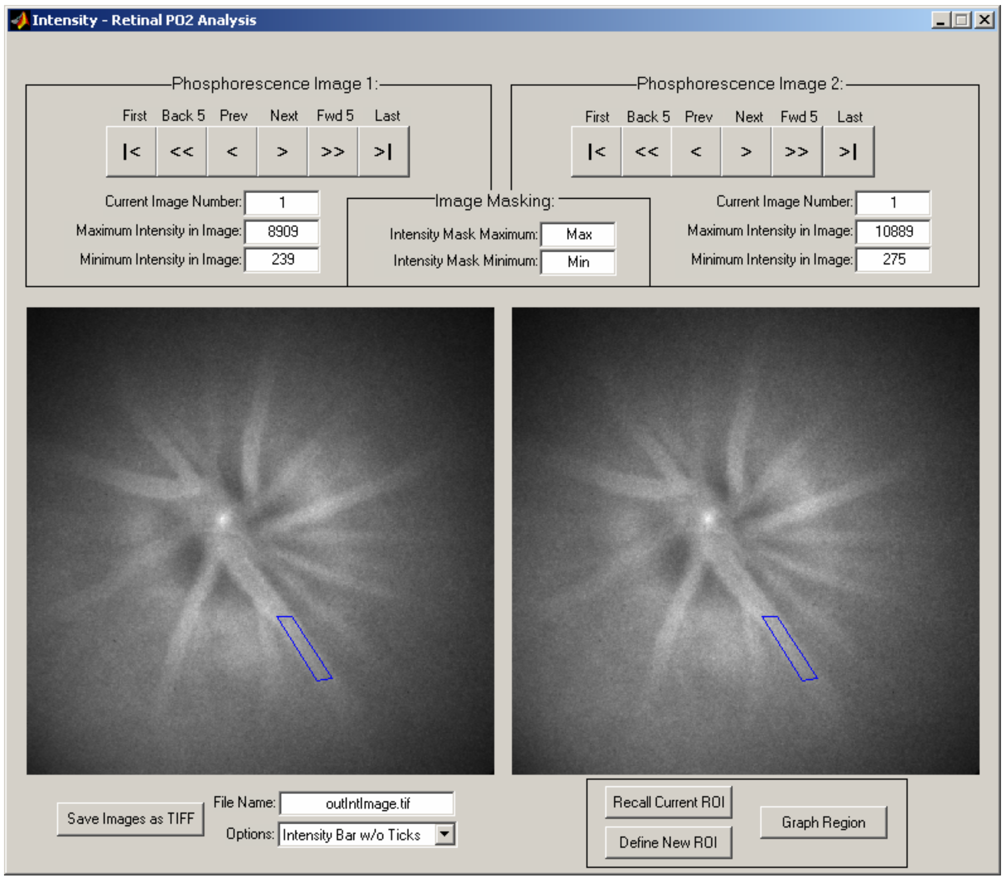
Subtract Image1 from Image2
Saves result as a WinView file
Filename: Untitled

Figure 11: Retina2's Align Images Interface

Figure 11 shows Retina2's "align images" interface. This interface was designed specifically for Retina2, to allow the user to align the phosphorescence images. This alignment accounts for any movement that might occur between the two image sets. Image alignment is a critical parameter for the performance of the zero-phase removal algorithm. This interface also allows the two image sets to be subtracted from each other and to be saved as a new image set. The image alignment process colors one image set completely red and the other green. When two regions of similar intensity are aligned, the resultant color will be yellow. So when the images are properly aligned, the image displayed in the "align image" interface window will be completely yellow, with no red or green displayed.

المنارة للاستشارات

www.manaraa.com



Phosphorescence Image 1:
Current Image Number: 1
Maximum Intensity in Image: 8909
Minimum Intensity in Image: 239

Image Masking:
Intensity Mask Maximum: Max
Intensity Mask Minimum: Min

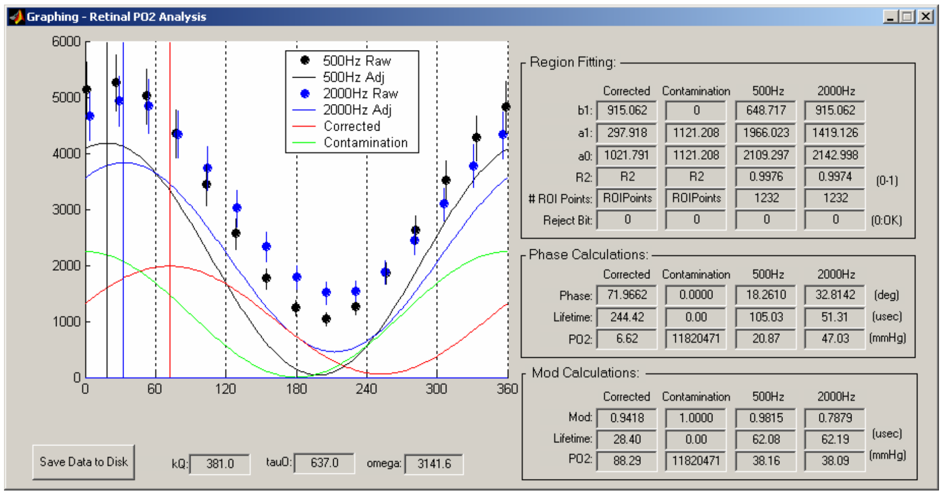
Phosphorescence Image 2:
Current Image Number: 1
Maximum Intensity in Image: 10889
Minimum Intensity in Image: 275

File Name: outIntImage.tif
Options: Intensity Bar w/o Ticks

Figure 12: Retina2's Phosphorescence Intensity Display

Figure 12 shows the Retina2’s phosphorescence intensity display interface. From this interface, the user can view the raw intensity images in each image stack and define regions of interest (ROI). The ROIs defined in this interface are used if the user decides to graph the region, which calculates a PO_2 for the region.

www.manaraa.com



kQ: 381.0 tau0: 637.0 omega: 3141.6

Region Fitting:

	Corrected	Contamination	500Hz	2000Hz	
b1:	915.062	0	648.717	915.062	
a1:	297.918	1121.208	1966.023	1419.126	
a0:	1021.791	1121.208	2109.297	2142.998	
R2:	R2	R2	0.9976	0.9974	(0-1)
# ROI Points:	ROIPoints	ROIPoints	1232	1232	
Reject Bit:	0	0	0	0	(0:OK)

Phase Calculations:

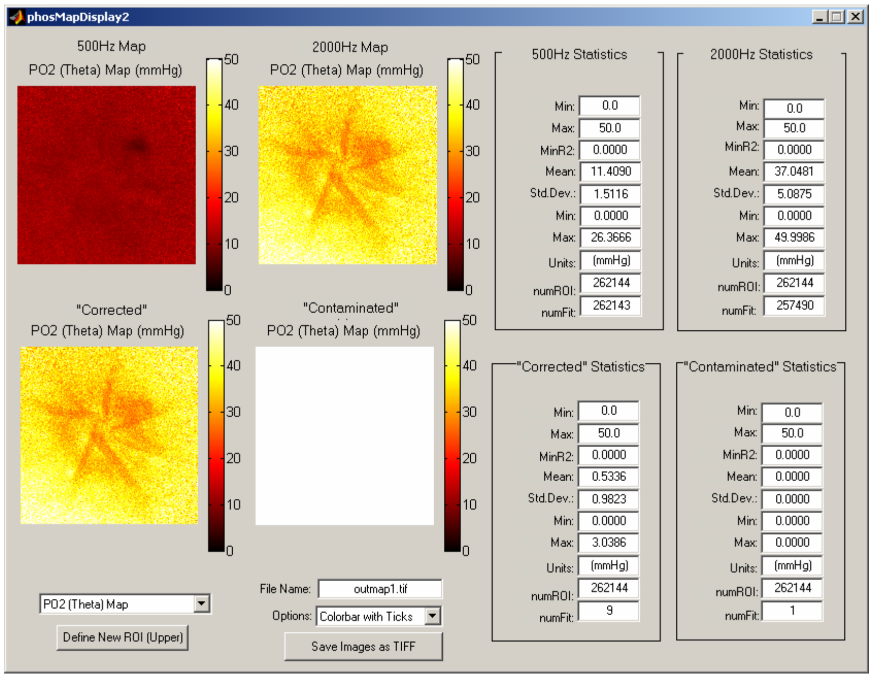
	Corrected	Contamination	500Hz	2000Hz	
Phase:	71.9662	0.0000	18.2610	32.8142	(deg)
Lifetime:	244.42	0.00	105.03	51.31	(usec)
PO2:	6.62	11820471	20.87	47.03	(mmHg)

Mod Calculations:

	Corrected	Contamination	500Hz	2000Hz	
Mod:	0.9418	1.0000	0.9815	0.7879	
Lifetime:	28.40	0.00	62.08	62.19	(usec)
PO2:	88.29	11820471	38.16	38.09	(mmHg)

Figure 13: Retina2's Regional PO$_2$ Calculator

Figure 13 displays the interface that is displayed when the user decides to calculate a PO$_2$ for a region of interest. This interface displays the cross correlation signal for the ROI at each excitation frequency, the “corrected” (zero-phase removed) cross correlation signal, and the removal algorithm derived “contamination” cross correlation signal. This interface also displays numerical values for various calculated parameters for each signal (PO$_2$, phase, modulation, fitting parameters, etc…).



PO2 (Theta) Map

File Name: outmap1.tif

Options: Colorbar with Ticks

500Hz Statistics

Min:	0.0
Max:	50.0
MinR2:	0.0000
Mean:	11.4090
Std.Dev.:	1.5116
Min:	0.0000
Max:	26.3666
Units:	(mmHg)
numROI:	262144
numFit:	262143

2000Hz Statistics

Min:	0.0
Max:	50.0
MinR2:	0.0000
Mean:	37.0481
Std.Dev.:	5.0875
Min:	0.0000
Max:	49.9986
Units:	(mmHg)
numROI:	262144
numFit:	257490

"Corrected" Statistics

Min:	0.0
Max:	50.0
MinR2:	0.0000
Mean:	0.5336
Std.Dev.:	0.9823
Min:	0.0000
Max:	3.0386
Units:	(mmHg)
numROI:	262144
numFit:	9

"Contaminated" Statistics

Min:	0.0
Max:	50.0
MinR2:	0.0000
Mean:	0.0000
Std.Dev.:	0.0000
Min:	0.0000
Max:	0.0000
Units:	(mmHg)
numROI:	262144
numFit:	1

Figure 14: Retina2's Map Display

www.manaraa.com

The Retina2 interface displayed in Figure 14 is essentially the same as the interface in Figure 13, but all of the calculations are performed on a pixel-by-pixel basis to generate maps of each parameter. It is through this interface that maps of each parameter are developed. As seen in Figure 14, a map for both excitation frequencies is shown as well as a “corrected” (zero-phase contamination removed) and a derived “contamination” map. The user is able to select which parameter maps are displayed in the interface.

Once the Retina2 analysis software was developed, it was used as the primary processing software for the phosphorescence images generated by the PLI instrument. Both the Retina and Retina2 analysis software were developed using Matlab. The source code for the Retina2 analysis software can be found in Appendix C – Retina2 Analysis Software Source Code.

Image Intensifier Frequency Dependence

During the process of developing the zero-phase removal algorithm, it was determined that the image intensifier (Princeton Instruments Gen III) was distorting the recorded phosphorescence signal with regards to phase and amplitude. While conducting tests at high excitation frequencies, it was found that the CCD camera or more specifically the image intensifier had a frequency dependent gain.

The emission phase recorded by the CCD camera may also be distorted for excitation frequencies over 1000 Hz due to the lifetime of the p43 phosphor within the intensifier. Current phosphorescence measurements acquired with the CCD do not adhere to the theoretical values at excitation frequencies over approximately 1000 Hz. One explanation for this phenomenon could be the low pass filtering effect of the p43

www.manaraa.com

phosphor used inside of the intensifier. The p43 phosphor takes approximately 1 ms to decay to 10% of its maximum value. This would suggest that signal distortion occurs for frequencies over 1000 Hz, resulting in a lower frequency signal, which alters the PO_2 analysis calculations. Even though the phosphorescence signal's phase appears distorted during PLI measurements and does not match the theoretical phase for each frequency, the calculated signal modulation does closely track the theoretical signal modulation for each excitation frequency. The distortion in calculated phase alone, suggest that the distortion occurring within the CCD/intensifier is primarily affecting the signal phase and frequency rather than the signal average and amplitude. Experiments that have isolated the intensifier have shown that the gain of the intensifier is specific to each excitation frequency. Other experiments have been conducted that have allowed profiling of the image intensifier and curve fitting has enabled the frequency dependent gain to be expressed mathematically.

Figure 15 shows the CCD intensifier gain across multiple gating frequencies. As seen in Figure 15, the gain of the intensifier is non-linear and increases with the gating frequency. This non-linear gain caused PLI measurements acquired at higher excitation frequencies to have a larger intensity than measurements taken at lower excitation frequencies. This was not a problem for single excitation frequency PLI measurements (ie not using the zero-phase removal algorithm), but for PLI measurements that required multiple excitation frequencies, such as with the zero-phase contamination removal, PO_2 calculations became highly problematic and error prone. The manufacturer of this CCD camera and intensifier verified that the intensifier does have a frequency dependent gain

www.manaraa.com

and was not operating according to its specifications. The manufacturer agreed to investigate the problem.

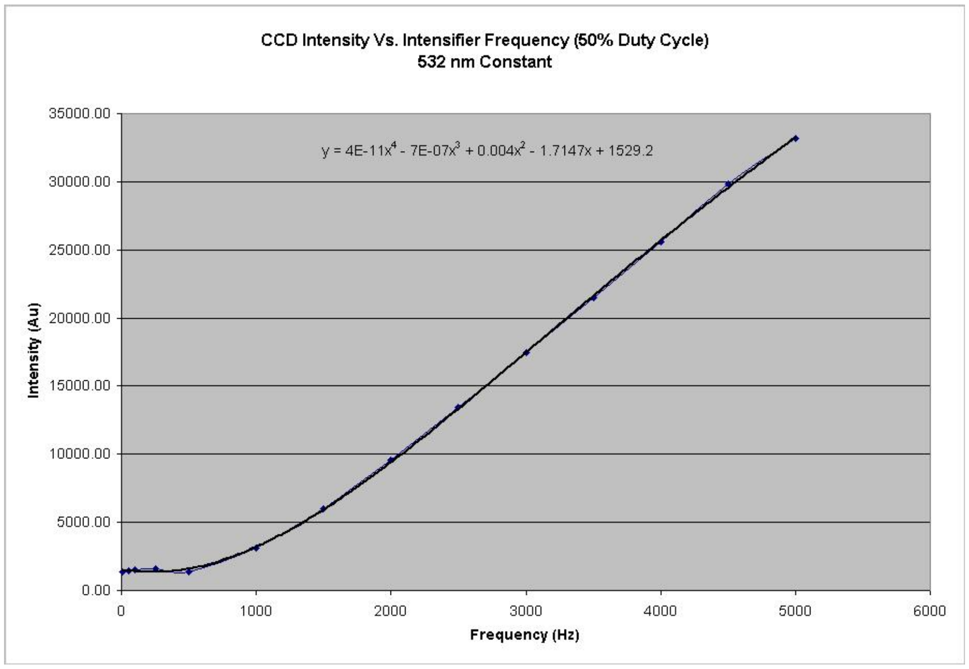


Figure 15: Gen III Intensifier's Frequency Dependent Gain

Analysis of the intensifier’s frequency dependent gain led to the generation of correction algorithms, which could normalize the intensities of emission signals, acquired using multiple excitation frequencies. Although multiple methods of intensity normalization were developed, a method that equilibrated the means of the cross correlation signals proved to be the best. Figure 16 displays the effect of the intensifier’s frequency dependent gain on the acquired cross correlation signals. The data displayed in Figure 16 was acquired by using the PLI instrument to image a sealed cuvette of phosphorescent probe. Glucose and glucose oxidase were added to the cuvette of probe solution in order to consume all of the available oxygen. The addition of this enzyme and the glucose allow for the PO_2 of the cuvette to fall to zero. As the excitation frequency

www.manaraa.com

was increased, the mean intensity of the corresponding cross correlation signal also increased. The effect of the intensifier's gain is most evident on the 2000 Hz data.

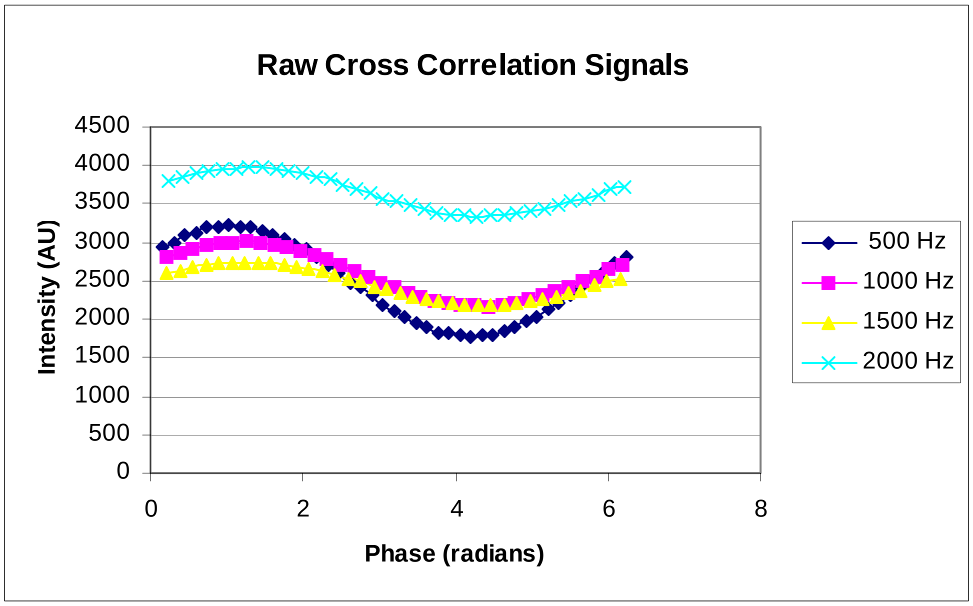


Figure 16: Raw Cross Correlation Signals

By equilibrating all of the cross correlation signal means to each other, the intensifier's frequency dependent gain was eliminated. This equilibration is performed by the Matlab M-file called intensifiercorrect.m, which is called by the Retina2 program. The correction occurs as the cross correlation signals are being determined from the phosphorescence images within the Retina2 program. The "intensifiercorrect" file determines the mean of each frequency's cross correlation signal, determines the ratio of frequency 1 (ω_1) and frequency 2 (ω_2)'s cross correlation means, and then multiplies each data point of ω_1's cross correlation mean by this ratio. This results in both cross correlation curves having the same mean value. When the data displayed in Figure 16 was processed with intensifiercorrect.m, cross correlation signals with equilibrated means were generated (see Figure 17).

www.manaraa.com

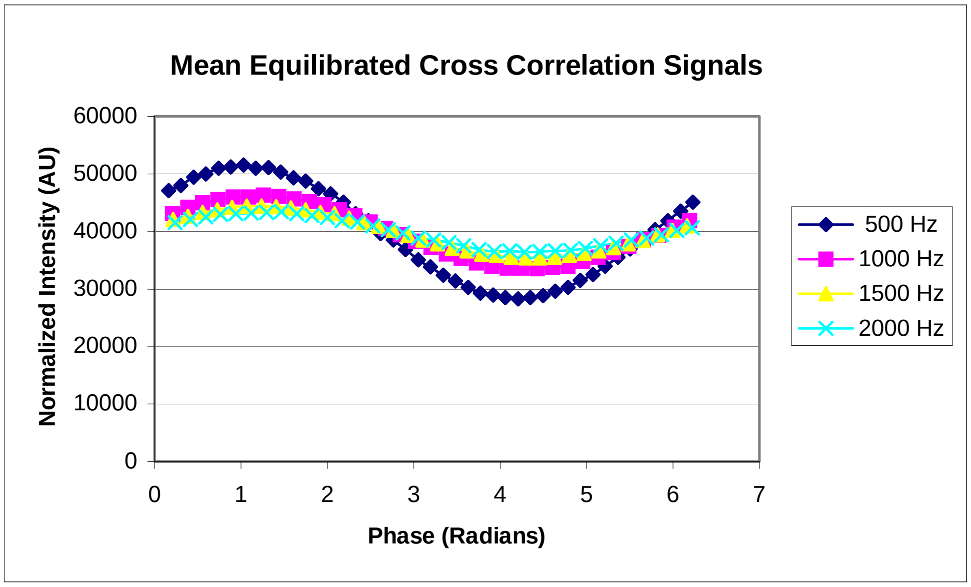


Figure 17: Mean Equilibrated Cross Correlation Signals

By equilibrating the means of the cross correlation signals, comparison of data acquired with different excitations can occur. This equilibration allows for the zero-phase removal algorithm to function properly.

Inherent PLI Instrument Phase Delay

In addition to complications induced by the intensifier's frequency dependent gains, the PLI instrument was shown to have a significant instrument phase delay. This phase delay would alter the phase of all measurements acquired with the PLI instrument. If unaccounted for, this instrument phase delay could significantly distort PLI calculations and prevent the accurate recovery of a sample's PO_2.

The first step towards accounting for the instrument's inherent phase delay was to classify the phase delay for different probe solutions (and solutions without probe) at various excitation frequencies.

www.manaraa.com

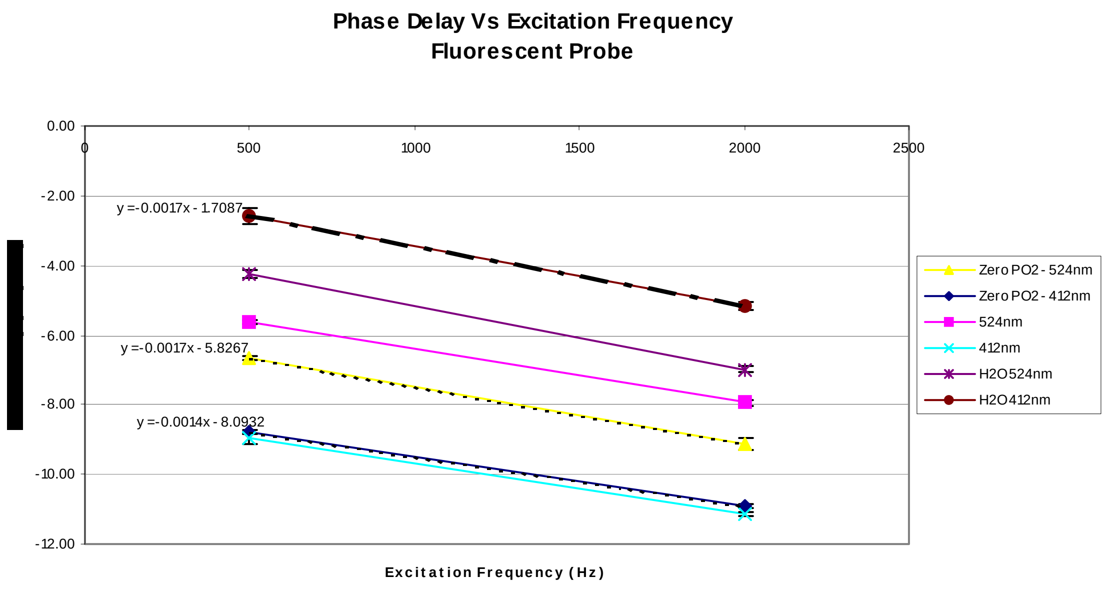


Figure 18: PLI Instrument Phase Delay

Figure 18 shows the classification of the system's instrument phase delay for a fluorescent probe at different excitation frequencies. A fluorescent sample was imaged, because the theoretical phase shift for a fluorescent sample is zero. So any phase shift different that zero could be attributed to the phase delay of the instrument.

Once the instrument's phase delay was classified, fitting a linear trend line to the appropriate phase delay trend enabled the development a phase correction. The equation for the line was used to derive phase correction values for the desired excitation frequency within the Retina2 program. Once obtained, the phase correction value was simply subtracted from the phase delay calculated from the target's cross correlation curve.

Although the source of the instrument phase delay is not fully understood, it is believed that the phase delay is created by the arc lamp and the monochromator. One current explanation suggests that the optical switch on the optical chopper which

www.manaraa.com

determines the status of the optical chopper (open or closed) might be out of alignment and would thus create a discrepancy between the frequency of the excitation light and the gating of the intensifier, resulting in a phase delay. Also, replacing the arc lamp bulb increased the magnitude of the phase delay, suggesting that the alignment of the bulb can alter the instrument's phase delay.

Cross Correlation Range Expansion

Previously, the cross correlation of the phosphorescence emission signal would only be carried out through 180 degrees of phase shift due to a physical limitation in the DG535 delay generator. Although 180 degrees of phase delay was all that was necessary to determine the phase shift of a phosphorescent emission, a larger range (ie 360 degrees) would assist the curve fitting process because the fitting algorithm would have more than a half period of cross correlation data to process. The limitation of the DG535 stemmed from the fact that it is driven by a TTL signal from the optical chopper to ensure that the two devices are synchronous in their timing. Due to the fact that the TTL signal had a 50% duty cycle, the DG535 could not produce a signal equal in frequency to the one received by the optical chopper and with a phase shift greater than 180 degrees with out being re-triggered by the optical chopper. Essentially, the DG535 must finish its delayed pulse before the next sync pulse from the optical chopper arrives, thus limiting the maximum phase delay to 180 degrees (using a 50% duty cycle).

www.manaraa.com

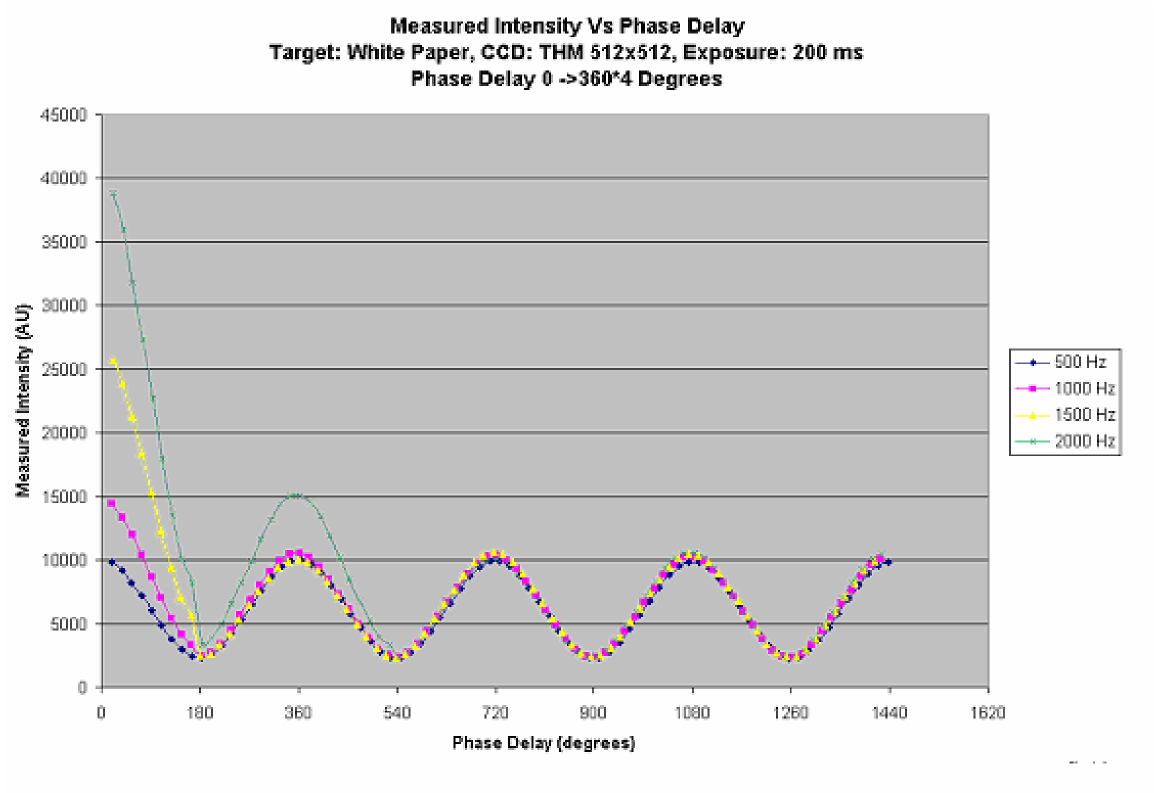


Figure 19: DG535 Induced Intensity Errors

Figure 19 displays what happens to the measured cross correlation intensity when the DG535 is forced to produce a phase delay greater than 180 degrees. As seen in Figure 19 the intensity is significantly reduced for each excitation frequency when the phase delay exceeds 180 degrees. When the phase delay of the DG535 surpasses 180 degrees, it starts to drop (ignores) every other pulse from the optical chopper. This dropping of pulses at 180 degrees creates a discontinuity in cross correlation signal intensity, which distorts the PO_2 calculation. The data in Figure 19 was acquired by imaging a white sheet of paper, which simulates a fluorescent (zero-phase signal).

A solution to this problem utilized the idea of forcing the DG535 to always ignore every other pulse from the optical, regardless of the DG535's current phase delay. This method eliminates the discontinuity at 180 degrees and allows the DG535 to acquire up to 540 degrees of phase with out any alterations to the intensity of the cross correlation

المنارة للاستشارات
www.manaraa.com

signal. By extending the phase delay range of the DG535, using this method, the overall intensity of the cross correlation curve was reduced by a factor of two, which does not alter any of the calculations used by the PLI system or zero-phase removal algorithm.

This method was implemented by using a ripple counter (74LS14) to divide the pulses produced by the optical chopper by two (essentially ignoring every other pulse).

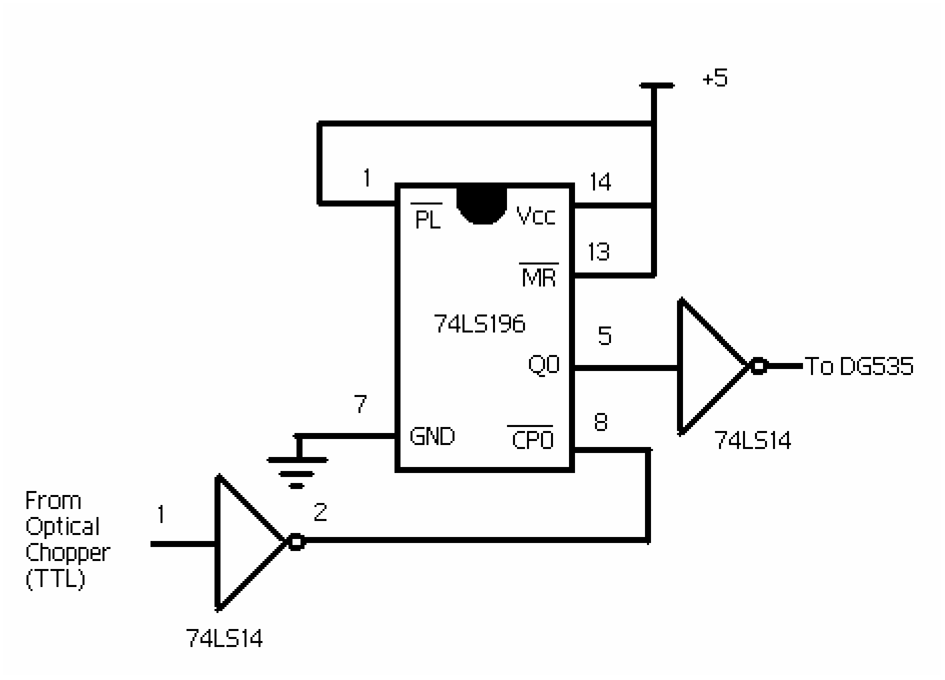


Figure 20: Ripple Counter Circuit

The ripple counter circuit shown in Figure 20 was placed in line with the optical chopper's TTL output and the DG535's input. With this circuit in place, the phase delay range can easily be run through 360 degrees of phase delay, which allows acquisition of an entire period of the cross correlation signal.

In Vitro Testing: Zero-Phase Removal

In order to verify that the zero-phase removal algorithm could isolate the zero-phase signals from actual PLI data recorded from the instrument, an *in vitro* test setup was configured to allow imaging of solutions with controlled PO_2 values. Figure 21

www.manaraa.com

shows the equipment setup for the *in vitro* testing. The equipment required for *in vitro* testing consists of a custom built gas mixing chamber made of silastic tubing, a capillary glass flow cell, a custom built bubble trap, a low flow rate peristaltic pump (Masterflex), and tygon tubing. The fluidic portion of the *in vitro* test setup consists of a closed system incorporating the flow cell and bubble trap. Probe solution was continuously pumped through the silasitic tubing of the gas exchange chamber (where the PO_2 of the probe solution is controlled), through the bubble trap and then through the flow cell where the probe solution is exposed for PLI analysis. The gas exchange chamber is connected to the laboratory gas mixers where the O_2 and N_2 gas ratio and flow rate could be controlled. As seen in Figure 21 the flow cell is placed on the microscope stage in the same location as where the retinal plane would be located in *in vivo* studies.

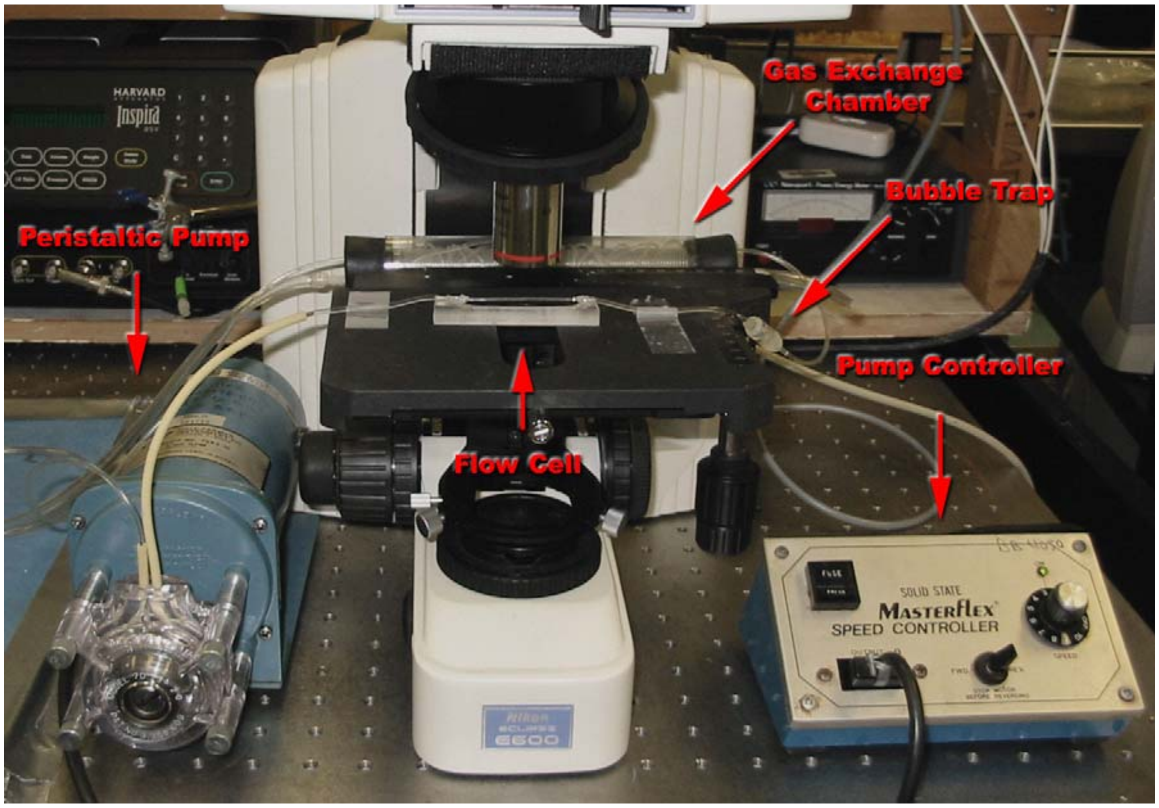


Figure 21: *In vitro* Test Setup

المنارة للاستشارات
www.manaraa.com

The zero-phase removal algorithm was tested by PLI analysis for different mixtures of phosphorescent and fluorescent probes under different oxygen tension values. The probe solutions consisted of concentrated phosphorescent probe (Pd-Meso-tetra[4-carboxyphenyl] porphyrin) or concentrated fluorescent probe (Meso-tetra[4-carboxyphenyl] porphyrin) mixed with 1.0x albumin stock solution.

Before fluorescent/phosphorescent probe solutions were imaged using the *in vitro* set up, solutions of pure fluorescent probe was imaged. Fluorescent probe solutions were imaged to check how well the algorithm could handle an *in vitro* derived signal completely composed of zero-phase contamination. Multiple (n = 6) *in vitro* tests, where a fluorescent probe was imaged, showed that the zero-phase removal algorithm reduced the intensity of a zero-phase signal by 98% *in vitro*. The *in vitro* fluorescence tests proved that the zero-phase removal algorithm could significantly reduce the contamination caused by a zero-phase signal. Figure 22 shows how the zero-phase removal algorithm reduced the contributions of zero-phase signals (fluorescence probe). The 500Hz and 2000Hz data in Figure 22 represents the cross correlation of the fluorescence emission at two separate excitation frequencies (500Hz and 2000Hz respectively). The "corrected" data shown in Figure 22 represents the true phosphorescence signal, which in the case of a fluorescence probe is essentially zero. In addition proving that the zero-phase removal algorithm can appropriately handle pure fluorescence emissions, the in vitro fluorescence tests confirmed that the zero-phase removal algorithm not only worked in the mathematical simulation of the PLI instrument but also performed well with the real PLI instrument. Once the algorithm was confirmed

المنارة للاستشارات
www.manaraa.com

to work on zero-phase signals alone, testing of zero-phase contaminated phosphorescence samples began.

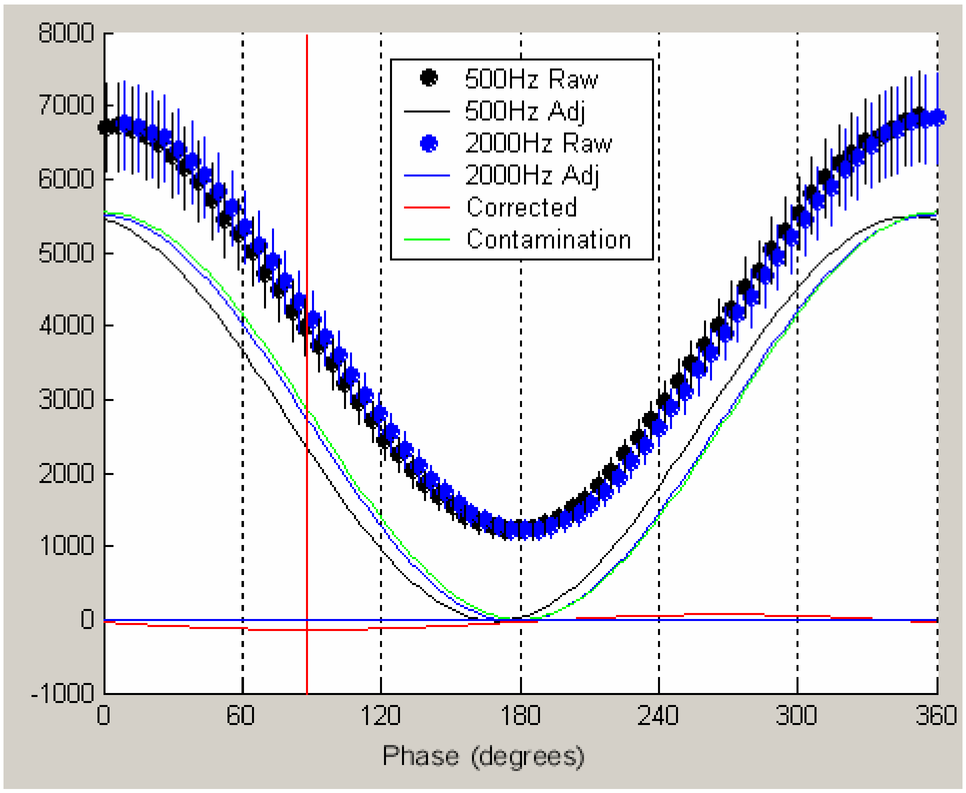
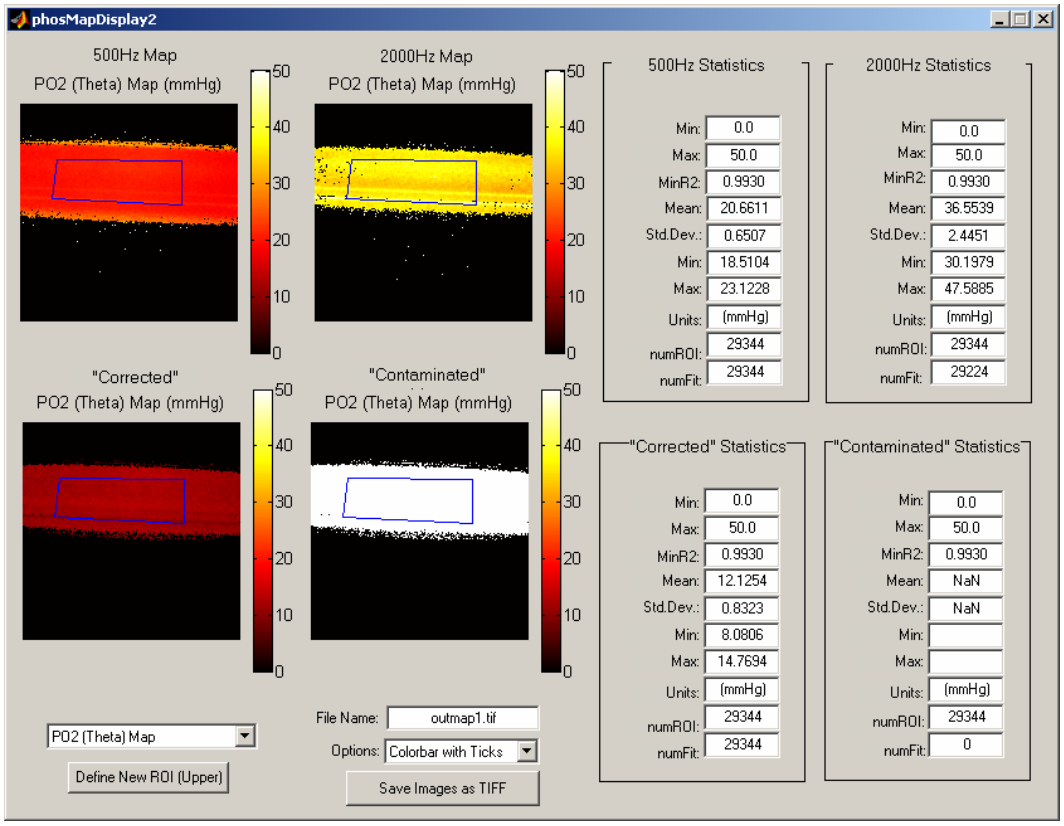


Figure 22: Zero-Phase Algorithm Performance

In order to test the algorithms ability to recover the correct PO_2 from a zero-phase contaminated sample, mixtures of fluorescent and phosphorescent probes were placed in the fluidics portion of the *in vitro* test system and were subject to various PO_2s by the gas mixers and gas exchange chamber. The fluorescent and phosphorescent probes solutions were mixed in 50-50 ratio and were placed in a 1.0x albumin solution. This probe mixture was imaged at four different PO2s (7.6mmHg, 15.06mmHg, 52.85mmHg and 75.5mm Hg).

www.manaraa.com



	500Hz Statistics	2000Hz Statistics
Min:	0.0	0.0
Max:	50.0	50.0
MinR2:	0.9930	0.9930
Mean:	20.6611	36.5539
Std.Dev.:	0.6507	2.4451
Min:	18.5104	30.1979
Max:	23.1228	47.5885
Units:	(mmHg)	(mmHg)
numROI:	29344	29344
numFit:	29344	29224

	"Corrected" Statistics	"Contaminated" Statistics
Min:	0.0	0.0
Max:	50.0	50.0
MinR2:	0.9930	0.9930
Mean:	12.1254	NaN
Std.Dev.:	0.8323	NaN
Min:	8.0806	
Max:	14.7694	
Units:	(mmHg)	(mmHg)
numROI:	29344	29344
numFit:	29344	0

PO2 (Theta) Map

File Name: outmap1.tif

Options: Colorbar with Ticks

Figure 23: Sample PO_2 Maps of *In vitro* Testing

Figure 23 displays sample PO_2 maps of the flow cell during *in vitro* tests. The controlled PO_2 for the results shown in Figure 23 was 12.0 mmHg. In this experiment the corrected PO_2 (12.13 mmHg) matched the controlled PO_2 value. As seen in Figure 23, the calculated PO_2s of the zero-phase contaminated 500 Hz and 2000 Hz signals produced values higher than the controlled PO_2 value (20.66 mmHg and 12.13 mmHg respectively). The data presented in Figure 23, shows that the zero-phase contamination model and removal algorithm hold true for low PO_2 values.

www.manaraa.com

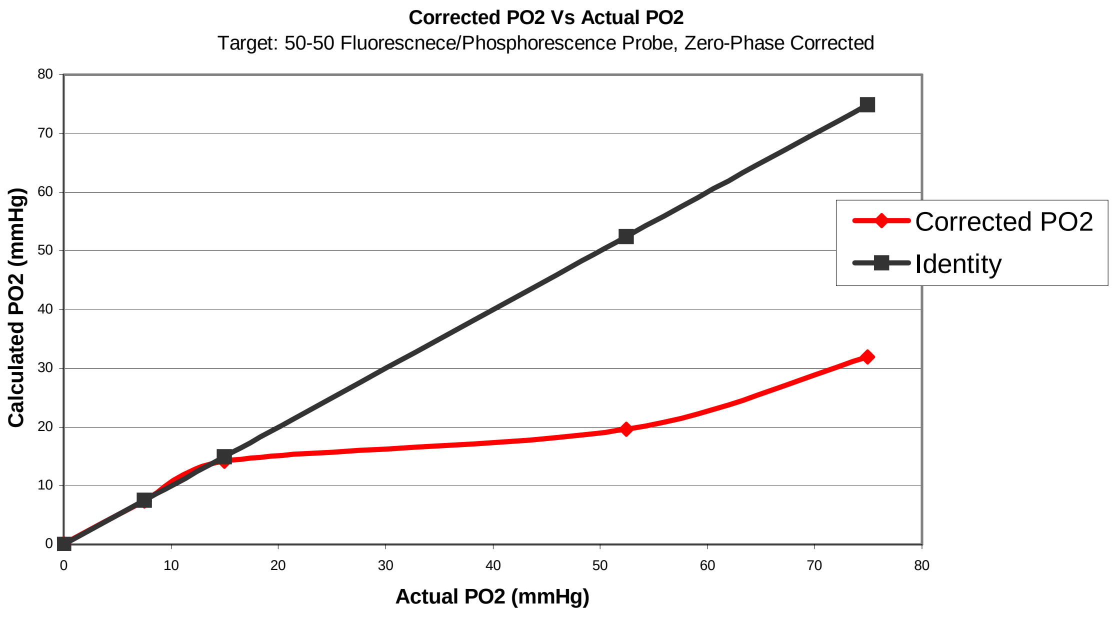


Figure 24: *In Vitro* Zero-Phase Removal Results

Figure 24 displays the results for the zero-phase removal *in vitro* tests. These results show that that algorithm functions properly for target samples with low PO_2 values but does not work well for target samples with PO_2 values greater than ~20 mmHg.

Although the degradation of the zero-phase removal algorithm's performance at high sample PO_2s is not completely understood, it is believed that the variability and instability produced by the arc lamp and optical chopper produce enough experimental variability to prevent proper algorithm performance at high PO_2s. In addition, these *in vitro* tests were conducted with two set excitation frequencies, 500 Hz and 2000 Hz. Experimentation with the PLI simulator proved that 500 Hz does not have very good sensitivity at high PO_2s while 2000 Hz has moderate sensitivity at frequencies ranging between 20 and 100 mmHg (see Figure 28). Also, at high PO2s, the phosphorescence intensity is significantly reduced due to the effects of phosphorescence modulation,

www.manaraa.com

making the actual phosphorescence signal difficult to detect when saturated with background fluorescence. The individual frequency's sensitivity at various PO_2s coupled with the variability of the optical chopper and the effects of phosphorescence modulation appear to be the leading cause of the algorithm's failure at high sample PO_2s.

LED Based Excitation Light Source

In order to provide a more stable excitation light source than the arc lamp and optical chopper, an LED based illumination system was developed for the PLI system. A high-powered LED (LumiLED, LXHL-MM1A) was used in conjunction with a custom built LED driver circuit to produce a stable LED based excitation system. The LED was a 1.0-watt LED that had a dominant emission wavelength of 530nm. In order to achieve maximum light output the LED was driven with a forward current of 350 mA. A function generator producing a typical TTL signal (0-5v square wave) provided the TTL drive signal to the LED. A custom driver circuit, Figure 25, was used to step the drive current of the TTL signal up to 350 mA. The function generator was used to control the on/off frequency of the LED with a 50% duty cycle.

المنارة للاستشارات
www.manaraa.com

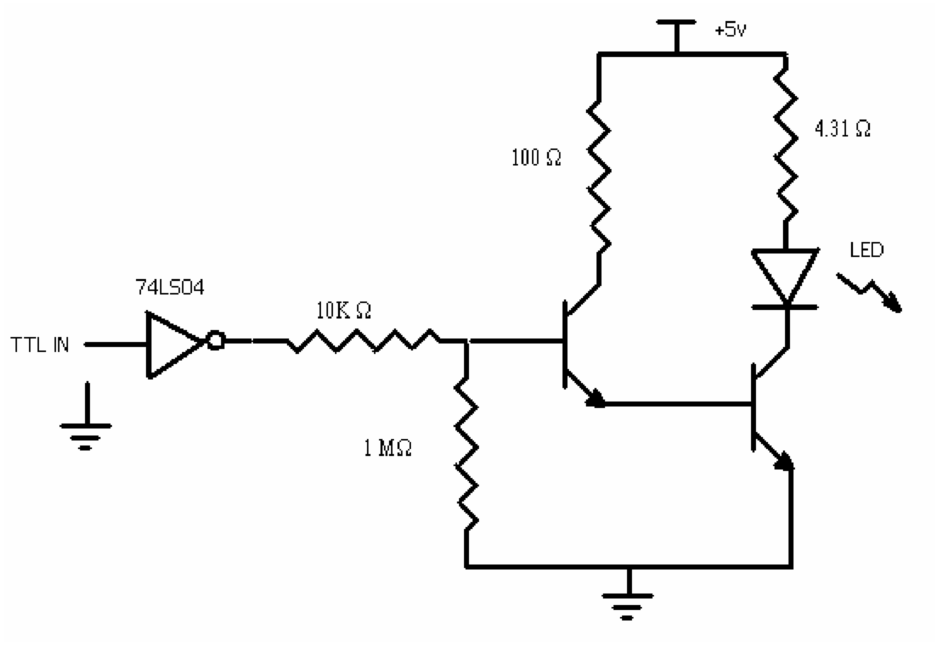


Figure 25: LED Driver Circuit

Numerous tests were conducted to determine the stability and accuracy of the LED profile at multiple drive frequencies. These profiling tests used the CCD camera with an intensifier gate delay of approximately 1 to 2 us to sample the LED cycling on and off. Results of two such tests are shown in Figure 26 and Figure 27. Figure 26 shows the LED profile at a drive frequency of 20,000 Hz. As seen in Figure 26, the square pulses of the LED are fairly crisp and accurately represent a 20,000 Hz square wave.

www.manaraa.com

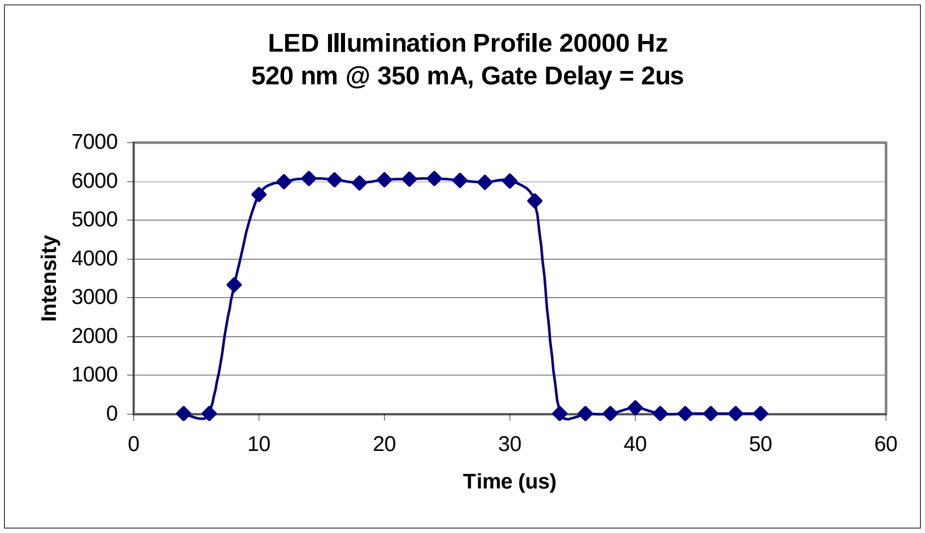


Figure 26: LED Illumination Profile, 20,000Hz

The data shown in Figure 27 shows that the LED when driven at frequencies around 40,000 Hz tend to break down. The edges of the square wave pulses become more rounded and less pronounced.

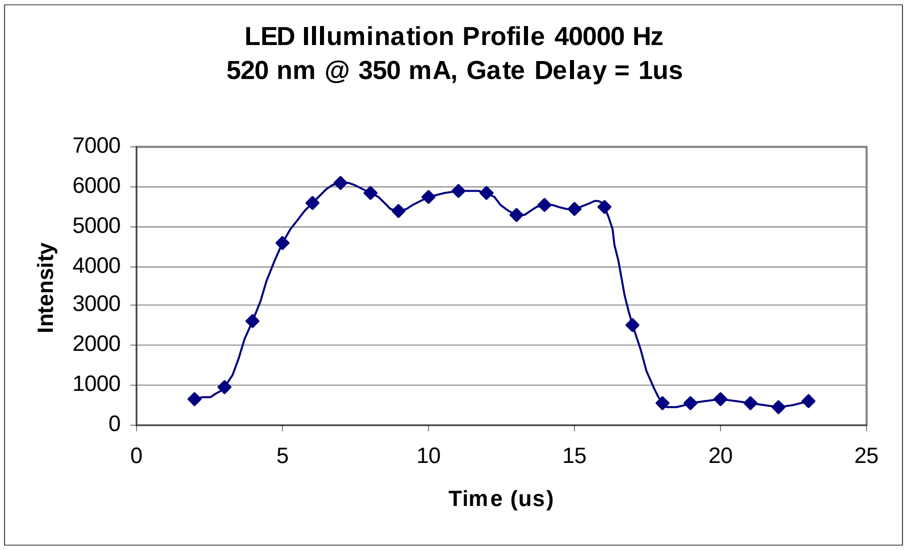


Figure 27: LED Illumination Profile, 40,000Hz

The LED profiling tests showed that the LED with a TTL driver circuit is highly stable and can achieve excitation frequencies orders of magnitude greater than the optical chopper. The one disadvantage to the LED as an excitation system is that it has a square wave profile, where as the optical chopper has a sinusoidal profile. The effects of square

المنارة للاستشارات

www.manaraa.com

wave excitation are not yet fully understood and require additional analysis and experimentation to learn how square wave excitation affects PLI parameters such as signal modulation.

Multi-layered Maps: Methods and Results

Once the suitable mathematics were developed for the removal of zero-phase signals, the isolation of phosphorescent signals from the retinal and choroidal vasculatures could be investigated. Principally, there were three applicable methods that had potential to assist the construction of the multi-layered PO_2 maps, given the nature of the PLI system. These methods consisted of mathematical algorithms that are applied to the phosphorescent images during post processing, tissue penetration depths of two wavelengths of excitation light, and methods of optical sectioning that eliminate out of focus light. The application of each of these methods to the PLI system and the ability to isolate the signals from both the choroidal and the retinal vasculatures was investigated.

Multi-layered PO_2 imaging is still in its early stages and there exist many problems, which still need to be fully understood and then solved in order to make significant progress toward the realization of these multi-layered PO_2 images.

Post Processing Algorithms

As described in previous discussion, the oxygen dependent phosphorescence signal emitted from the retinal and choroidal vascular beds are essentially combined together. The combination of these phosphorescence signals causes the apparent PO_2 of the tissue to contain a portion of the PO_2 information from each vascular bed. This combination of signals, with respect to the CCD, is essentially the summation of two sinusoids of equal frequency but of different phase shift and modulation. Breaking down

www.manaraa.com

the combined signal into its primary components is exceedingly difficult without precise knowledge of at least one of the individual PO_2s. Although the isolation of the phosphorescent lifetimes of the combined signal appears computationally problematic, methods for phosphorescence lifetime extraction, such as maximum entropy method and the multi-harmonic Fourier method do exist.[22] Adataptation and application of these methods for the PLI system might allow the separation of these two signals.

Similar to the extraction of zero-phase signals from the PLI measurement multi-excitation frequency analysis was investigated for the isolation of to unique phosphorescence emissions. Much of this investigation focused on determining how the phase and amplitude of the phosphorescence emission signals from each layer dictate the contribution of the individual signals to the combined signal. It was determined through use of the PLI simulator that the choroidal signal would dominate in terms of amplitude because the retinal phosphorescence signals (which are traditionally of lower PO_2) would have a smaller modulation value when compared to the choroidal phosphorescence signal. Figure 28 displays how the individual phosphorescence signals would differ in terms of phase and modulation for different sample POs. In addition to the reduced modulation value of the retinal vasculature, the retinal vasculature also has an increased phase delay when compared to an identical measurement of the choroidal

www.manaraa.com

phosphorescence.

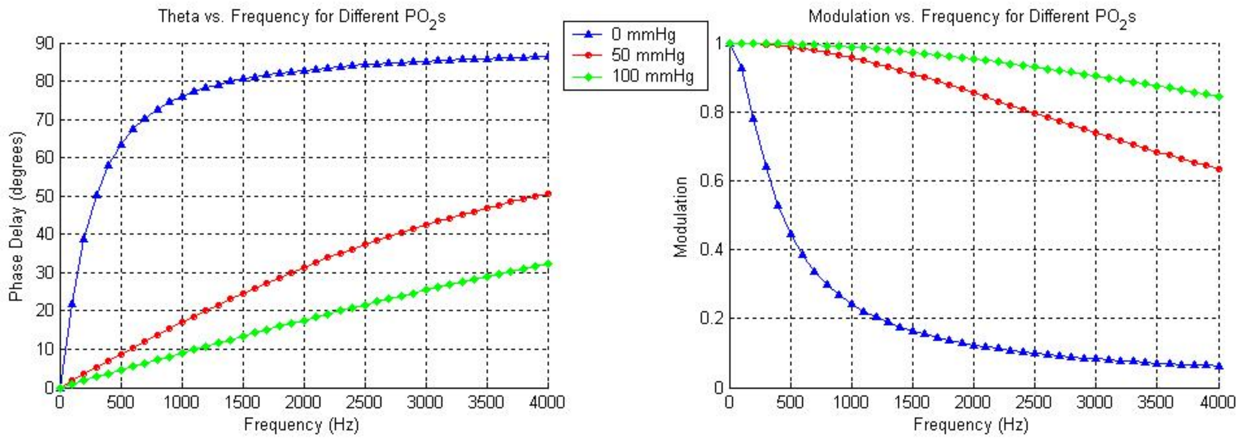


Figure 28: Frequency Response for Various PO_2 Values

Similar to the zero-phase removal algorithm, multi-frequency analysis consists of analyzing the phase and modulation of the phosphorescence signal over a wide range of frequencies. Analysis of the phosphorescence over a range of frequencies provides multiple “versions” of the “combined” PO_2 signal of which the mathematical phase extraction can be performed. Also, certain frequency ranges (primarily 1000 Hz to about 5,000Hz) magnify the phase and modulation differences of the phosphorescence signal for relatively close PO_2 levels. Analysis of the frequency response of the phase delay and the modulation over a broad range of excitation frequencies may permit decomposition of the combined signal. The development of the LED based light source for the PLI system permits multi-frequency analysis over 3000 Hz (the current optical chopper has a maximum frequency of 3000 Hz).

Although the mathematics of the PLI system have been studied at length using the PLI simulator, no mathematics could be generated, under the constraints of the current PLI instrument, that would enable accurate separation of retinal and choroidal phosphorescence signals. The difficulty in deriving mathematics to isolate the

www.manaraa.com

phosphorescence signals from these two vasculatures, stems from the fact that both signals have variable phase and modulation values. Once the signals are combined in the imaging process, there are an infinite number of signals with different phases and modulations that be combined to represent the measured phosphorescence signal. In addition, this method depends on the fact that the PO_2s of the retinal and choroidal vasculatures are in fact different. If there is no difference in the PO_2 between the retinal and choroidal vasculatures, the two phosphorescence signals can not be mathematically isolated.

Dual Excitation Wavelengths

The fundamental difference between the penetration depths of green (λ = 532 nm) and blue light (λ = 412 nm) was exploited to obtain specific data about the PO_2 of the retinal vasculature. Due to the shorter wavelength of the blue light, the light does not penetrate as deep into the retinal tissue as the green light[24]. Theoretically, the images generated with a blue excitation light are dominated by a phosphorescence signal that was developed by the retinal vasculature rather than the choroidal vasculature. Experimentation shows that an overall lower PO_2 is achieved using a blue excitation light, which would suggest the phosphorescence signal is generated primarily from the retina.

Figure 29 shows two phosphorescence images acquired using two different excitation wavelengths, acquired sequentially from the same animal. As seen in Figure 29, the image acquired with green excitation light (524 nm) produces a much more diffuse image due to a strong choroidal presence. The image acquired with blue excitation (412 nm) produces a sharper image where the retinal vessels are emphasized.

المنارة للاستشارات
www.manaraa.com

In vivo experimentation showed that the 524nm derived phosphorescence images were almost completely dominated by the choroid in the capillary regions. The 412nm-derived images are a fairly good representation of the retinal phosphorescence signals.

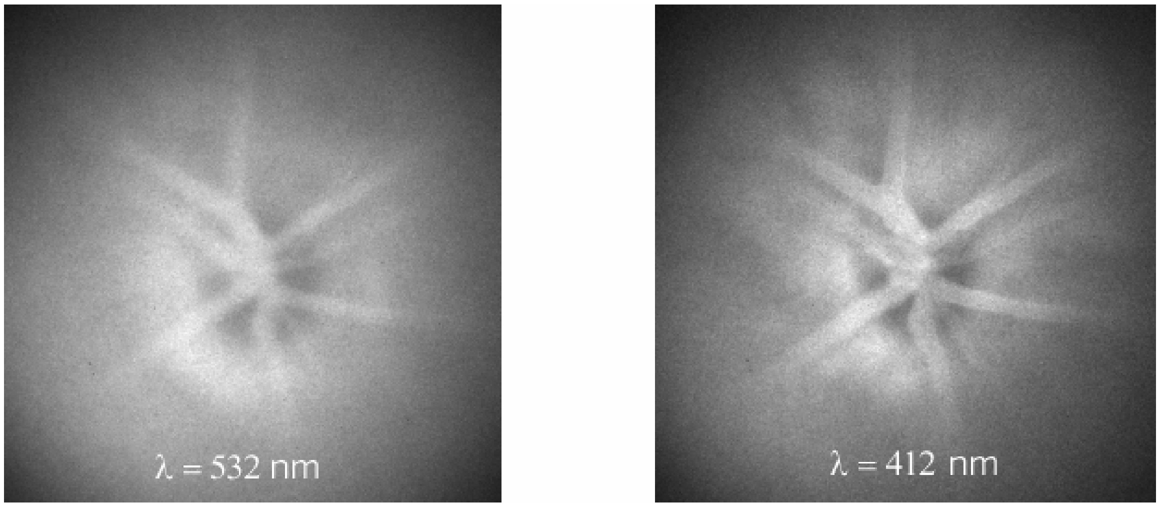


Figure 29: Dual Excitation Wavelength Phosphorescence Images

Due to fact that the zero-phase removal algorithm could not successfully remove the zero-phase contamination at high PO_2s (due to the optical chopper), the phosphorescence contributions of each vasculature could not be quantified enough to produce multi-layered maps of the retina and choroid. Although complete multi-layered images cannot yet be generated using this method, multi-excitation wavelength analysis appears to be highly promising. With the current instrumentation setup, partial application of multi-excitation wavelength analysis was used. When conducting *in vivo* studies, blue and green (excitation light) image sets were acquired. The capillary regions of the "green excitation light" image sets were analyzed to obtain PO_2 values for the choroidal vasculature, while "blue excitation light" image sets were analyzed to obtain PO_2 values that better approximated retinal the retina PO_2.

www.manaraa.com

Optical Sectioning

Optical sectioning is another method that has potential to contribute to the construction of a multi-layered PO_2 map. Optical sectioning allows the removal of out of focus light through various methods, such as, multi-photon and confocal imaging. By removing out of focus light from the phosphorescence images, tissue depths that are not directly in the focal plane of the microscope can be removed. This method would allow the generation of multi-layered PO_2 maps by acquiring the PO_2 map for the retina and the choroid independently and then combining the two layers to form the "multi-layered" image. Currently, optical sectioning is a popular field of research with a strong focus on multi-photon and confocal microscopy. In order to apply optical sectioning to the PLI instrument and alternative optical sectioning technique would have to be implemented because the current microscope is a research grade fluorescence microscope, not a confocal or multi-photon microscope. One such alternative was to use a Ronchi grating system (Optem Inc's optigrid system) to eliminate the out of focus light. The Ronchi grating system projects a Ronchi grid onto the microscopes optical target and through software analysis determines which areas of the image are out of focus.

Although this method has the potential to enable the construction of multi-layered images, tests performed using the Ronchi grating system proved to be indiscernible from measurements without the Ronchi grating. The decrease in effectiveness of the Ronchi system might be caused by the limited ability of the microscope to clearly focus in on a specific vascular bed. Experimentation proved that it is fairly difficult to focus the microscope on either the retinal vasculature or the choroidal vasculature. Optimization of the focusing abilities of the microscope may permit increased effectiveness of optical sectioning using the Ronchi grating.

المنارة للاستشارات
www.manaraa.com

Despite the fact that the current PLI instrument requires more improvement, before multi-layered PO_2 maps are feasible, this research was able to verify the best method for isolating the retinal and choroidal PO_2s. The most applicable method for generating multi-layered oxygen tension maps is to the use of blue excitation light to obtain a phosphorescence lifetime image dominated by the retina vasculature and then to use mathematical extraction procedures to isolate the two different phosphorescence signals. The dual excitation wavelengths may separate the differences between the choroid and retina's phosphorescent signal enough to all mathematical algorithms to distinguish between the two phosphorescence signals. Once the PLI instrument is able to remove zero-phase contamination at high PO_2s, development of mathematics to separate the retinal and choroidal layers will become feasible. The difficulty in this process arises from the fact that there is currently no way to mathematically model the dynamics of multi-wavelength excitation. All studies must be conducted *in vivo*, where the current application of the zero-phase removal algorithm breaks down. The development of multi-layered PO_2 maps would be greatly accelerated if a mathematical model of multi-wavelength excitation existed or if the zero-phase removal algorithm functioned properly *in vivo*.

PLI Investigation of Type I Diabetes

Retinal oxygen tension changes have been theorized to be associated with the pre-clinical stages of diabetic retinopathy. An initial study utilized PLI oxygen mapping to investigate the retinal PO_2 content of naturally diabetic (type I) BBDP/Wor rats and age matched non-diabetic control rats (BBDR/Wor). Rats were anesthetized and kept under tight physiologic control throughout the imaging process. Analysis of the BBDR and

المنارة للاستشارات
www.manaraa.com

BBDP PO_2 maps suggested that retinal hypoxia was present in the diabetic retinas. Average PO_2 in the arterial, venous, and capillary regions analyzed were, respectively, 17 ± 7 %, 11 ± 10 %, and 18 ± 13 % lower in the diabetic rats (n = 3, average age = 115 ± 8 days, duration of diabetes = 25, 29, and 48 days) when compared to controls (n = 3, average age = 130 ± 50 days). Systemic arterial PO_2 was not statistically different between groups and no overt visual indications of diabetic retinopathy were present. While not statistically significant ($P > 0.20$), preliminary results support the hypothesis that retinal PO_2 changes occur in type I diabetic rats

The preliminary BBDR/BBDP retinal PO_2 research was expanded upon by performing an extensive retinal PO_2 mapping study on poly I:C induced diabetic LEW1 rats and age matched non-diabetic control LEW1 rats. Physiologic monitoring and PO_2 mapping was conducted in the same fashion as with the BBDR/BBDP study, with the addition of blood glucose and HbA1c monitoring. In addition conduction velocities were recorded to gauge diabetic neuropathies.

Rat Model of Type I Diabetes

The LEW.1WR1 rat was chosen as an appropriate model of type I diabetes because of its ability to develop type I diabetes following Polyinosine-polycytidylic acid (poly I:C) treatment. poly I:C is a synthetic analogue to double stranded DNA. When injected into the blood stream, the poly I:C acts like a viral infection and stimulates an immune response.

The LEW.1WR1 rat spontaneously develops diabetes in 4-5% of the population. Delivery of the TRL antagonist, poly I:C, further increases the development of type I diabetes in the LEW.1WR1 rat to nearly 100%. Poly I:C was delivered at a dose of 1 mg

www.manaraa.com

/ g of body weight three times a week until diabetes was induced (~17 days). Once diabetes was detected, the animal's blood glucose levels were monitored on a daily basis and insulin was delivered appropriately.

Blood glucose levels in diabetic and control rats were monitored to ensure the diabetic state of the animal. As show in Figure 30, there was a significant difference between the blood glucose levels of the control and diabetic rats at the time in which retinal oxygen mapping was performed.

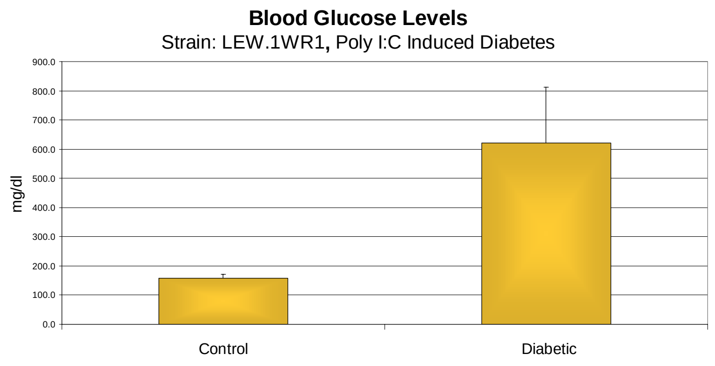


Figure 30: Blood Glucose Levels

In addition, the percent glycosylation of hemoglobin (HbA1c) was measured in both diabetic and control rats as a measure of the long term blood glucose levels. The HbA1c test is a blood test that measures the levels of hemoglobin glycosolation. Over time, high blood glucose levels cause a glycosolation of hemoglobin's surface. The HbA1c test provides information about the animal's history of blood glucose levels weeks. This test further confirms the presence of diabetes by proving that the data displayed in Figure 30 is not simply due to random chance.

www.manaraa.com

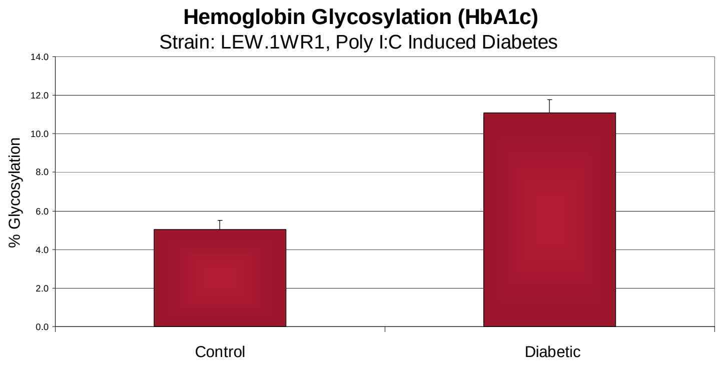


Figure 31: HbA1c Levels

In addition to the blood glucose level data (Figure 30) and the HbA1c data (Figure 31) nerve conduction tests confirmed the presence of diabetes. These nerve conduction studies were performed by League-Pike, et al. on the LEW.1WR1 rats after retinal oxygen mapping was performed. [15] The results of the nerve conduction test proved that many of diabetic rats were experiencing diabetes induced neuropathies (Figure 32). This data indicates that these rats had entered a stage of diabetes were tissue damage was beginning to occur because of abnormal blood glucose levels.

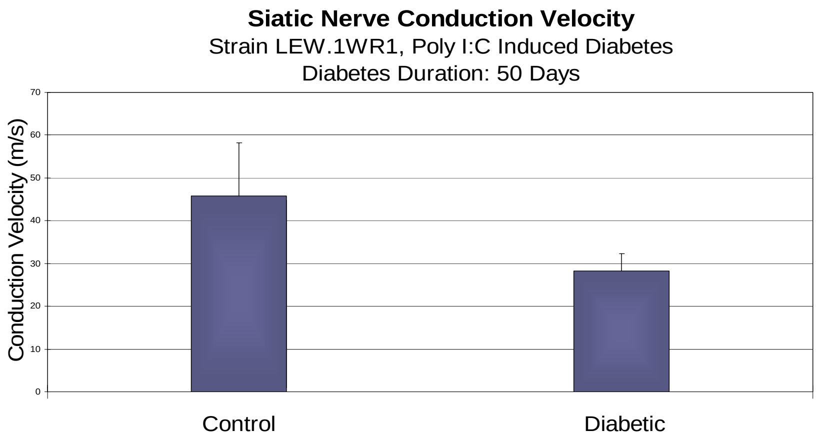


Figure 32: Nerve Conduction Data[15]

www.manaraa.com

PLI Imaging of Diabetic Rats

All experiments were performed in accordance with WPI's Institutional Animal Care and Use Committee (IACUC) protocols and the Association for Research in Vision and Ophthalmology statement for the Use of Animals in Ophthalmic and Vision Research. Blood pressure, respiration rate, tidal CO_2, and core body temperature were monitored and recorded continuously. Systemic blood gas was measured intermittently. PO_2 maps were generated using two excitation wavelengths (l_1 = 524nm and l_2 = 412nm) and multiple modulation frequencies (500Hz and 2000Hz). The acquired phosphorescence images were processed using the Retina2 analysis software to generate PO_2 maps of the retinas.

As shown in Figure 33 the results of a regional tissue analysis of PO_2 maps generated from 412nm excitation light showed the presence of retinal hyperoxia in diabetic rats, when compared to age matched non-diabetic rats.

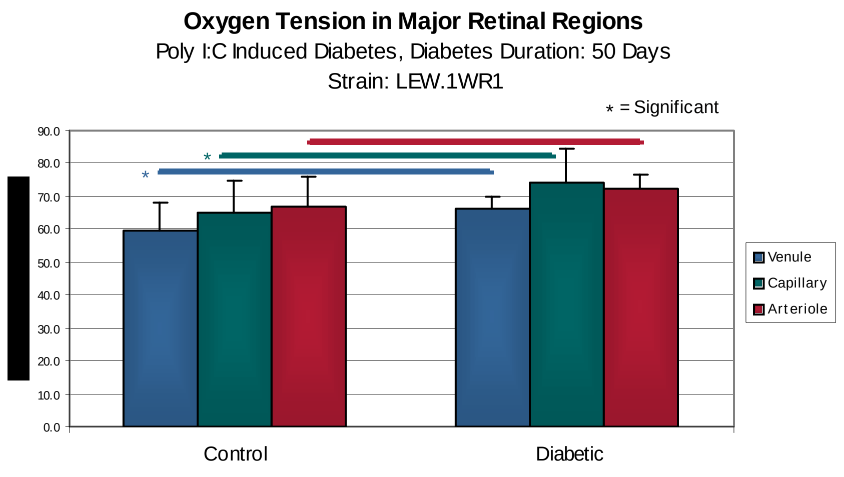


Figure 33: Regional Analysis of Diabetic and Non-Diabetic Retinas

The results shown in Figure 33 are contrary to what was expected. Retinal hypoxia was expected to exist in the diabetic rats, but this data suggests that retinal hyperoxia was present in this stage of diabetes. A time course analysis (Figure 34)

www.manaraa.com

suggests that retinal hyperoxia exists in the early stages of diabetes and proceeds towards hypoxia as the duration of diabetes is extended. No correlation was found between the age of the rat and the oxygen state of the retina.

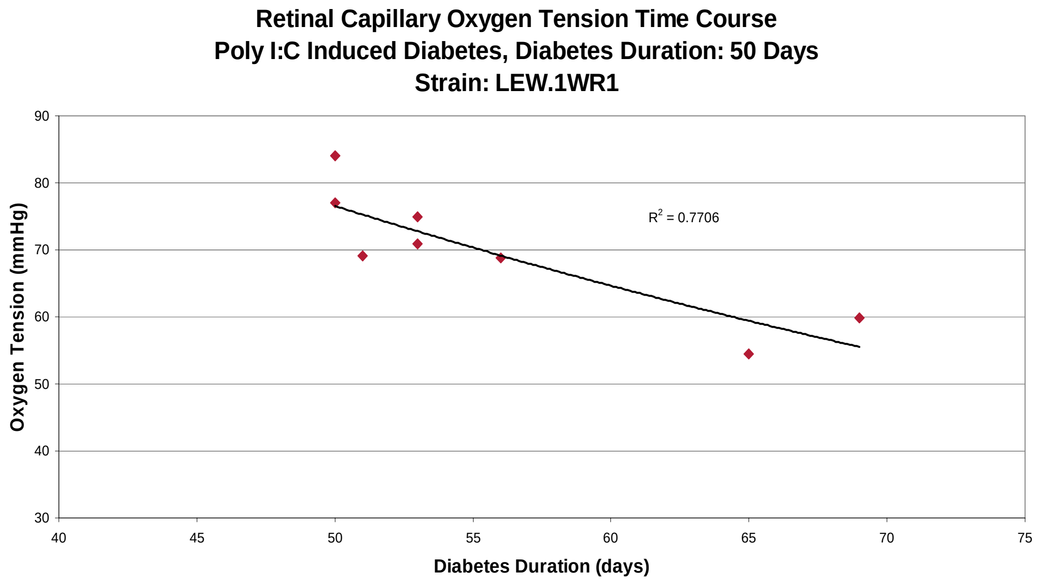


Figure 34: Time Progression of the retinal Oxygen State in Diabetic Rats

Currently, it is hypothesized that the changes in retinal oxygenation are associated with the blood flow to the retina. These results suggest that prior to the onset of diabetic retinopathy, the blood flow briefly increases (resulting in hyperoxia) and then proceeds to decrease until the retina becomes hypoxic. Although, the results produced from this study indicate that a retinal hyperoxia to hypoxia trend prior to diabetic retinopathy, more animals specifically with a longer diabetes duration (>65 days) must be examined to strengthen these findings.

Discussion

This research investigated the current state of PLI for use in the retina. Within this investigation, the PLI instrument was mathematically modeled, current short comings of PLI were identified and modeled, correction algorithms were developed and tested,

www.manaraa.com

new hardware configurations were constructed, image enhancements were studied, and the PLI technology was applied to diabetic research. Additionally, each of the initial specific aims was addressed.

The contaminating nature of fluorescence and reflected excitation light was studied and an accurate mathematical model for this contamination was developed. The characterization and modeling of these zero-phase signals was critical to their removal. The mathematical model for zero-phase contamination greatly facilitated the discovery of zero-phase signals in the current PLI system. The model also provided information into the mechanics of how the zero-phase contamination affected a typical PLI measurement.

Once the zero-phase signals were fully understood, algorithms were developed to remove these signals from PLI measurements. Without the elimination of the zero-phase signals, the PLI instrument would continue to produce results that do not accurately represent the actual PO_2 of the tissue. The process of eliminating the zero-phase signals consisted searching for mathematical relationships between PLI measurements acquired using two separate excitation frequencies. Through mathematical analysis, it was determined that the zero-phase contamination appears the same (mathematically) on both excitation frequencies. With this knowledge an algorithm was developed that searched for non-phosphorescence information that was identical on both excitation frequencies. Once this proved feasible a mathematical removal algorithm was developed. Mathematical modeling proved the effectiveness of the removal algorithm, but *in vitro* testing exposed instrumentation limitations that prohibited the algorithm from functioning properly at high PO_2s.

المنارة للاستشارات
www.manaraa.com

Although, multi-layered oxygen tension maps proved unfeasible under with the current PLI instrumentation, investigations proved that multi-wavelength excitations do in fact have future value. By exciting the probe with two separate wavelengths of light, each with different tissue penetration depths, the different layers of the retina could be emphasized. Once the PLI system becomes capable to properly run the zero-phase removal algorithm, dual-excitation wavelengths can further be classified *in vivo.*

In addition to the enhancements made to the PLI instrument, the technology was applied to *in vivo* diabetic studies. These studies examined the retinas of type I diabetic rats and age-matched controls. The results of this study suggested that the retinas of diabetic rats become hyperoxic in the diabetic rats.

Future Work

The initial step towards continuing this research is to upgrade the current PLI system to remove current components that were discovered in this research to cause measurement error. Primarily the optical chopper and monochromator based excitation system should be replaced with an LED based excitation system. The LED system will remove the excitation frequency variability and inherent instrument phase delay experienced with the current system. Addition of an LED based illumination system might expand the PO_2 range in which the zero-phase removal algorithm performs. Also, the LED based excitation system will enable the use of high frequency excitation (> 5000 Hz), which currently cannot be achieved with the optical chopper system.

In the short term, the performance of the zero-phase removal algorithm might be enhanced by using excitation frequencies closer together such as 2000 Hz and 2050 Hz

المنارة للاستشارات
www.manaraa.com

(span of 500 Hz) rather than 500 Hz and 2000 Hz (span of 1500 Hz). A frequency range too large seems to cause problems because of the drastic difference in measurement sensitivities for the corresponding excitation frequencies. Data can also be acquired in pairs for different frequency ranges. For example, data can be acquired at 500 and 550 Hz, then at 1500 and 1550 Hz and then at 3000 and 3050 Hz. Acquisition in this format would allow a large range of the excitation frequency's measurement sensitivity to be sampled for the unknown PO_2, and a small excitation frequency range would be acquired at each interval (500, 1500, and 3000 Hz) to allow accurate application of the zero-phase removal algorithm.

In addition to replacing the LED system, alternative excitation waveforms (i.e.: square wave, impulse, and triangle wave, etc…) should be investigated. These methods might allow measurements utilizing multiple excitation frequencies to be acquired using one measurement and post processing with a form of the Fourier transform. If this proves to be a viable method for performing PLI measurements, imaging time would be significantly reduced because data could be acquired for multiple excitation frequencies using one measurement.

www.manaraa.com

References

1. Aiello, L.P. Diabetic Retinopathy. *Diabetes Care* 21:153-147, 1998

2. Alder, V.A., Er-Nign, S., Yu, D.Y., Cringle, S. J., Yu, P. Diabetic retinopathy: Early functional changes. *Clin.Exp. Pharmacol. Physiol.* 24:785-788, 1997.

3. Alm , A., A. Bill. "Ocular Circulation." In *Adler's Physiology of the Eye, Clinical Application,* edited by R.A. Moses & W.M. Heart Jr. St. Louis: The C.V. Mosby Company, 1987

4. Amin, R.H., Frank, R.N., Kennedy, A., Elliot, D., Pulkin, J.E. & Abrams, G.W. Vascular endothelial growth factor is present in glial cells of the retina and optic nerve of human subjects with nonproliferative diabetic retinopathy. *Invest. Ophthalmol. Vis. Sci*. 38:2729-2741,1997

5. Beach, J.M, Schwenzer, K.J., Srinivas, S., Kim, D. & Tiedeman, J.S. Oximetry of retinal vessels by dual-wavelength imaging: Calibration and influence of pigmentation. *J. Appl. Physiol.* 86:748-758,1999.

6. Berkowitz, B.A., Kowluru, R.A., Frank, R.N., Kern, T., Hohman, T., & Prakash, M. Subnormal Retinal Oxygenation Response Preceded Diabetic-like Retinopathy. *Invest. Ophthalmol. Vis Sci 40:2100-2105, 199*3.

7. Dowling, John E. Neurons and Networks: An Introduction to Neuroscience. Cambridge, MA: Belknap Press, 1992.

8. Ernest, J.T., T.K. Goldstick, R.L Engerman. Hyperglycemia Impairs Retinal Oxygen Autoregulation in Normal and Diabetic Dogs. *Assn for Research in Vis Ophthalmol 24:985-989, 1983.*

9. Hartmann, P., Ziegler, W., Holst, G., Lübbers, D.W. Oxygen flux fluorescence lifetime imaging. *Sensors and Actuators B* 38-39:110-115, 1997.

10. Lakowicz, J.R., Szmacinski, H., Nowaczyk, K., Berndt, K.W., Johnson, M. Fluorescence Lifetime Imaging. *Ann Biochem* 202, 316-330, 1992.

11. Lakcwicz, J.R., Principles of Fluorescence Spectroscopy: Second Edition. Kluwer Academic / Plenum Publishers: New York, 1999.

12. Linsenmeier, R.A., C.M Yancey. Effects of Hyperoxia on the Oxygen Distribution in the Intact Cat Retina. *Invest. Ophthalmol. Vis Sci 30:612-618, 1989.*

13. Linsenmeier, R.A., R.D. Braun, M.A. McRipley, L.B. Padnick, J. Ahmed, D.L. Hatchell, D.S., McLeod, G.A. Lutty. Retinal Hypoxia in Long-Term Diabetic Cats. *Invest Ophtalmol Vis Sci* 39:1647–1657,1998.

14. Lo, L.-W., Koch, C.J. & Wilson, D.F. Calibration of oxygen-dependent quenching of the phosphorescence of Pd-meso-tetra [4-carboxylphenyl] porphine: A phosphor with general application for measuring oxygen concentration in biological systems. *Anal. Biochem.* 236:153-160,1996.

15. League-Pike, W. F., D.L. Mason, R. D. Shonat., Characterizing Neuropathy in the Sciatic Nerve of a Poly I:C – Induced Rat Modle of Type I Diabetes. Proceedings, 30th Annual Northeast Bioengineering Conference, Western New England College, Springfield, MA., 2004.

المنارة للاستشارات
www.manaraa.com

16. Rowe, H.E., Sing Po Chan, J.N. Demas, B.A. Degraff. Elimination of Fluorescence and Scattering Backgrounds in Luminescence Lifetime Measurements Using Gated-Phase Fluorometry. *Anal Chem.* 74:4821-4827, 2002.

17. Saari, J.C. "Metabolism and photochemistry in the retina." In *Adler's Physiology of the Eye, Clinical Application,* edited by R.A. Moses & W.M. Heart Jr. St. Louis: The C.V. Mosby Company, 1987.

18. Sherwood, L.. Human Physiology-From Cells to Systems. Pacific Grove, CA.: Brooks/Cole. 2001.

19. Shonat, R.D, Kight, A.C. Oxygen Tension Imaging in the Mouse Retina. *Ann. Biomed. Eng.* 2003.

20. Stefánsson, E. Oxygen and diabetic eye disease. *Graefe's Arch.Clin.Exp.Ophthalmol*.228:120-123,1990.

21. Vinogradov, S.A., M.A. Fernandez-Searra, B.W. Dupan, D.F. Wilson. A method for measuring oxygen distributions in tissue using frequency domain phosphorometry. *Comp Biochemistry and Physiology.* 72:148-152, 2002.

22. Vinogradov, S.A., M.A. Fernandez-Searra, B.W. Dupan, D.F. Wilson. Frequency domain instrument for measuring phosphorescence lifetime distributions in heterogeneous samples. *Rev. of Sci Inst.* 72:3396-3406, 2001.

23. Webb, S. E. D., Y. Gu, S. Le´veˆque-Fort, J. Siegel, M. J. Cole, K. Dowling, R. Jones, P. M. W. French, M. A. A. Neil, R. Jusˇkaitis, L. O. D. Sucharov, T. Wilson, and M. J. Lever. A wide-field time-domain fluorescence lifetime imaging microscope with optical sectioning. *Rev of Sci Inst.* 73:1898-1907, 2002

24. Wilson, DF, S.M. Evans, W.T.Jenkins, S.A. Vinogradov, E. Ong, M.W. Dewhirst. Oxygen distributions within R3230Ac tumors growing in dorsal flap window chambers in rats. *Adv Exp Med Biol.* 454:603-9, 1998.

25. Yu D-Y., S.J. Cringle. Oxygen Distribution and Consumption within the Retina in Vascularised and Avascular Retinas and in Animal Models of Retinal Disease. *Progress in Retinal and Eye Research*, 20:175-208, 2001.

المنارة للاستشارات
www.manaraa.com

Appendix A – PLI Simulator Source Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Frequency Domain Imaging Simulator with Zero Phase Contamination
%    and Contamination Removal
% Created by: Adam Norige 8/30/03
%
%   This version of the frequency domain imaging (PLI) simulator
%    contaminates the phosphorescence signal with backscattered light
%    The back scattered light is incorporated by cross corralating the
%    excitation signal and adding the two x-corr results together
%    (phosp and excite).
%   This simulator implements a method to remove the imposed in-phase
%    contamination and then reconstructs the true sample PO2.
%    This simulator allows for 2 fully selectable excitation frequencies
%    to be chosen for in-phase contamination removal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function PLI_simulator ()

global Kq T0
%Probe = pd-meso-tetra[4carboxyohenyl] porphine
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% CONSTANTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kq = 381;            %Quenching constant (1/torr*1/s) @ pH=7.4 & 38 C
T0 = .000637;        %Lifetime w/o quenching (s) @ pH=7.4 & 38 C
I0 = 100;              %Excitation Intensity w/o quenching (arbitrary value)
%-------------------------------------------------
Po2_1= 100;          %PO2 value 1
Po2_2= 100;          %P02 value 2
%-------------------------------------------------
Po2_wt1 =1;         %Percent Weight for PO2 1 (less than 1)
                    %Percent weight for PO2 2 is  Calculated
%-------------------------------------------------
pc1     =1;        %Percent Weight True Phosp signal(less than 1)
%                   %Percent weight for in-phase contamination
%                       is Calculated
%-------------------------------------------------
freq1 = 500;        %Modulation frequency 1 (Hz)
freq2 = 2000;       %Modulation frequency 2 (Hz)
%-------------------------------------------------
a = .5;             %DC offset of excitation signal
b = .5;             %Half the amplitude of excitation signal
%                       Note a is ALWAYS larger than b

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% VARIABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tau_1   = 0;           %Actual Lifetime value 1
Tau_2   = 0;           %Actual Lifetime value 2
theta_1 = 0;           %Phosphorescence lifetime phase 1
theta_2 = 0;           %Phosphorescence lifetime phase 2
mag_1   = 0;           %Phosphorescence magnitude 1
mag_2   = 0;           %Phosphorescence magnitude 2
omega   = freq1*2*pi;  %Initial Modulation Frequency

A_1 = 0;               %DC offset of emission signal 1
B_1 = 0;               %Half the amplitude of the excitation signal 1
```

www.manaraa.com

```
A_2 = 0;                %DC offset of emission signal 2
B_2 = 0;                %Half the amplitude of the excitation signal 2
AB_ratio = 0;           %Ratio of A and B ... B/A *temp variable*

n = 1;                  %Counter for array Storage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    LIFETIME CALCULATION - Stern Volmer Equation
Tau_1 = T0/(1+Kq*T0*Po2_1); %Determine Lifetime
Tau_2 = T0/(1+Kq*T0*Po2_2);
I_1  =  1;%I0/(1+Kq*T0*Po2_1); %Determine Intensity
I_2  =  1;%I0/(1+Kq*T0*Po2_2);

%Main Loop to cycle through frequencies
while (omega == freq1*2*pi | omega == freq2*2*pi)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    EMISSION PHASE
    theta_1 = atan(omega*Tau_1)              %Calculate Theta
    theta_2 = atan(omega*Tau_2);              %tan(theta) = Omega*Tau
    temp = theta_1*(180/pi);

    %Generate Instrument Phase delay
    frq = omega/(2*pi);                      %Excitation Frequency in Hz
    thetaCorr = (-2.2351*log(frq) + 6.7637); %Freq dependant phase delay
    thetaCorr = thetaCorr*(pi/180);          %Convert to radians
    theta_1 = theta_1 + thetaCorr;           %Put in inst phase delay
    theta_2 = theta_2 + thetaCorr;
    temp = theta_1*(180/pi);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    EMISSION MAGNITUDE
    mag_1   = (1+omega^2*Tau_1^2)^(-1/2);    %Calculate Mw for PO2_1
    AB_ratio = mag_1*(b/a);                  %Determine A & B for PO2_1
    A_1 = 0.5;                               %Just a guess!
    B_1 = AB_ratio*A_1;                      %Here's B

    mag_2   = (1+omega^2*Tau_2^2)^(-1/2);    %Calculate Mw for PO2_2
    AB_ratio = mag_2*(b/a);                  %Determine A & B for PO2_1
    A_2 = A_1;                               %Just a guess!
    B_2 = AB_ratio*A_2;                      %Here's B

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    GENERATE THE TIME ARRAY
    dps    = 1000;                           %Number of data points
    t_step = (2*pi)/(omega*dps);             %Time step for time domain
    t      = 0:t_step:((2*pi)/omega)-t_step; %Time array

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    GENERATION of SINUSOIDS
    Excite =a+b*cos(omega*t);                              %Excitation Signal
    Fluorescence=(a)+(b)*cos(omega*t + thetaCorr);         %Zero phase signal

    Emission_1 = (I_1*A_1)+(I_1*B_1)*cos(omega*t+theta_1); %PO2_1 Emission
    Emission_2 = (I_2*A_2)+(I_2*B_2)*cos(omega*t+theta_2); %PO2_2
    Emission_Sum = (Po2_wt1*Emission_1+(1-Po2_wt1)*Emission_2);  %Combined Emission

%     figure(1)
%     hold on;
```

www.manaraa.com

```
%     if omega == freq1*2*pi
%         plot(t,Emission_Sum, '-r');
%     else
%         plot(t,Emission_Sum, '-b');
%         grid on;
%         xlabel('Time');
%         ylabel('Intensity');
%         title(sprintf('Emission Signals, %d & %d Hz', freq1, freq2));
%         legend(sprintf('%d Hz', freq1), sprintf('%d Hz', freq2));
%     end
%     hold off;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    PERFORM CROSS CORRELATION
    [CorrS,phase]=Cross_Corr(Fluorescence,t,omega);   %Cross Correlate the back scatter
    [Corr,phase]=Cross_Corr(Emission_Sum, t, omega);  %Cross Correlate the real signal
    %Corr = pc1*Corr+(1-pc1)*CorrS;                    %Create the contaminated signal
    Corr = Corr+.8*CorrS;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    CROSS CORRELATION PLOT -- Figure 2
    figure(2);
    hold on;
    if omega == freq1*2*pi
        %subplot(2,1,1);
        phase_degree = phase.*(180/pi);
        plot(phase_degree, Corr, 'k');
        grid on
        title(sprintf('Cross-Correlation, %d & %d Hz', freq1, freq2));

    elseif omega==freq2*2*pi
        %subplot(2,1,2);
        phase_degree = phase.*(180/pi);
        plot(phase_degree, Corr, 'b');
        grid on
        xlabel('Phase (degrees)');
        ylabel('Intensity');
        title(sprintf('Cross-Correlation, %d & %d Hz', freq1, freq2));
    end
    hold off;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    CURVE FIT
    [theta_fit, mod_fit, a0, a1, b1]=Curve_bs_fit(Corr,phase);

    % Uncorrected Phase Calculated PO2
    Tau_p_fit = (tan(theta_fit))/omega;
    PO2_p_fit(n) = ((T0/Tau_p_fit)-1)/(Kq*T0)

    % Uncorrected Modulation Calculated PO2
    Tau_m_fit = sqrt(((omega^2*mod_fit^2)^-1)-(omega^-2));
    P02_m_mod(n) = ((Tau_m_fit*Kq)^-1)-((T0*Kq)^-1)

    %Correct Instrument phase delay
    frq = omega/(2*pi);                             %Get frequency in Hz
    thetaCorrection = -2.2351*log(frq) + 6.7637; %Calculate Phase Correction
    thetaCorrection = thetaCorrection*(pi/180);  %Convert to radians
    theta_p(n) = theta_fit - thetaCorrection;    %Correct Theta
```

www.manaraa.com

```
    ratio = tan(theta_p(n));    %Determine new a1/b1 ratio
    const = sqrt(a1^2+b1^2);

    a1 = sqrt(const^2/(1+ratio^2)); %New b1 term
    b1 = a1*ratio;                  %New a1 term

    theta = atan(b1/a1);
    Tau_uc(n) = (tan(theta))/omega;
    PO2_uc(n) = ((T0/Tau_uc(n))-1)/(Kq*T0)

    %This is the a0 correction for a cos intensifier profile
    a0_array(n) = a0/(2*a/b);%Linear Fit Constant a0
    %This is the approx a0 correction for a square intensifier profile
    %a0_array(n) = a0/1.5708;  %Linear Fit Constant a0
    a1_array(n) = a1;  %Linear Fit Constant a1
    b1_array(n) = b1;  %Linear Fit Constant b1

    n=n+1;             % Next element in the array
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    UPDATE OMEGA  - Loop Control
    if omega == freq2*2*pi
        omega=0;                 %Just to exit the loop
    elseif omega == freq1*2*pi
        omega=freq2*2*pi;        %Last iteration of loop
    end

end     % End of While loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    REMOVE ZERO PHASE CONTAMINATION
Omega_1 = 2*pi*freq1;
Omega_2 = 2*pi*freq2;

% fit_curve = a0+a1*cos(phase)+b1*sin(phase);
% delta_a0 = min(fit_curve)
% divisor = a0_array(1)/ delta_a0
% a0_array(1) = a0_array(1)/divisor
% a0_array(2) = a0_array(2)/divisor

% figure(2)
% hold on
% fitted_curve = a0_array(1)+a1_array(1)*cos(phase)+b1_array(1)*sin(phase);
% plot(phase_degree, fitted_curve ,'--r');
% %ylim([0 450])
% hold off;

R = (b1_array(2)*Omega_1)/(b1_array(1)*Omega_2);  %a1-frequency ratio
X = (R*a1_array(1)-a1_array(2))/(R-1)     %Determine a1 contamination

% Freq1 Reconstruction
a0 = a0_array(1) - X;
a1 = a1_array(1) - X;
b1 = b1_array(1);

%Plot the Corrected Signal and the Contamination
figure(2)
hold on
```

www.manaraa.com

```
fitted_curve = a0+a1*cos(phase)+b1*sin(phase);
plot(phase_degree, fitted_curve ,'-r');
contamination = X+X*cos(phase);
plot(phase_degree, contamination, '-g');
legend(sprintf('%d Hz', freq1), sprintf('%d Hz', freq2), 'Corrected', 'Contamination');
hold off

% check for zeroes in a0 and a1
if a0 == 0.0          % reject fits with a0 = 0
    a0 = eps;         % make a0 a very, very small number
end

if a1 == 0.0          % reject fits with a1 = 0
    a1 = eps;         % make a1 a very, very small number
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    GENERATE PHASE SHIFT FROM NEW FIT DATA
thetaCorrection = 0;        %phase correction for inst. delay (assumed zero for now)
theta = atan(b1/a1);
theta*(180/pi);
theta_est = phase(find(Corr == max(Corr)));
theta = theta - thetaCorrection;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    GENERATE MODULATION FROM NEW FIT DATA
modCorrection = 1;          %modulation correction (excitation modulation - assumed 1)
mod = sqrt(a1^2+ b1^2);
mod = mod/a0;
mod = modCorrection * mod;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    CALCULATE PO2 FROM PHASE DATA
%       NOTE: LOW FREQUENCY RECONSTUCTION (freq1)
Tau_phase = (tan(theta))/Omega_1;
PO2_phase = ((T0/Tau_phase)-1)/(Kq*T0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    CALCULATE PO2 FROM MODULATION DATA
%       NOTE: LOW FREQUENCY RECONSTUCTION (freq1)
Tau_mod = sqrt(((Omega_1^2*mod^2)^-1)-(Omega_1^-2));
P02_mod = ((Tau_mod*Kq)^-1)-((T0*Kq)^-1)

%  End PLI_simulator.m
```

www.manaraa.com

```
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
%     CROSS CORRELATION    (Cross_Corr)                                   +
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function [correlation, phase]=Cross_Corr(emission, time, omega)
%This function takes an emission waveform and correlates it with the
% intensifier signal.  The real intensifier signal is a square wave
% but to adhere to the true theory, a simple sinusoid is used.
%
% Input: There are three required input values,
%        Emission: The phosphorescence emission signal (array)
%        time: The time for which the emission signal is defined (array)
%        omega: Frequency in rad/sec of emission (int)
% Return: There are two return values for this function,
%        correlation: The correlated signal (array)
%        phase: Phase angles over which the correlation was conducted
%               (array)
%
% Written by: Adam Norige 6/24/03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    DEFINES
n=length(emission);           %Length of phosphorescence excitation buffer
phase_angle = 2*pi;           %Maximum phase angle of correlation
n_phase  = 1000;              %number of DPs in Phase array
phase_step = phase_angle/(n_phase-1);
temp = 0:phase_step:phase_angle;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    VARIABLES
phase = 0;                    %Phase shift
index = 1;                    %Index to Corrlation signal dps

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    PROCEEDURE
for phase = 0:phase_step:phase_angle;    %Perform over entire phase range

    %This intensifer profile mimics the theoretical sinusoid intensifer.
    intensifier=double(0.5+0.5*cos(((omega)*time)+(phase)));  %Theoretical intensifer
signal

    %This intensifier profile mimics the square wave intensifer
    %intensifier=double(0.5+0.5*square(((omega)*time)+(phase)+(pi/2)));  %Square
intensifer signal

    % This signal mimics the system when every other pulse is dropped.
    % intensifier=double(0.5+0.5*square(((omega/2)*time)+(phase/2)+(pi/4),25));  %Special
intensifer signal

    % The actual correlation shifts the intensifier signal by increasing
    % its phase.  For each phase shift, the emission signal and the
    % intensifier signals are multiplied together and the result is summed.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % This code is for diagnostic use only! Ensure
    %  a break point is set here to halt matlab
    %  or matlab will crash.
    %
    %  if phase >= (4*pi/2)
```

المنارة للاستشارات
www.manaraa.com

```
    %      figure (3);
    %      hold on
    % %     plot(time,(intensifier.*emission),'-rd')
    %      plot(time,(intensifier) , '-kx')
    %      plot(time,(emission) , '-bo')
    %      legend('Intensifier','Emission');
    %      hold off
    %  end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    correlation(index) = double(sum((intensifier.*emission))); %Record correlation dp
    index=index+1;                      %Next correlation data point
end

%Define an array of the phase shifts for return
phase = temp; %Make Phase an array

%Clear unneeded variables.
clear intensifier; phase_angle; phase_step; n; temp;

% End Cross_Corr.m
```

www.manaraa.com

```
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
%     Fit Curve to Cross Correlation    (Curve_fit)                         +
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function [theta, mod, a0, a1, b1]=Curve_bs_fit(corr,phase)

%This function takes the cross correlation result and determines
% a0, a1, and b1.  The phase shift and modulation are also calculated.
%
% Input: There are two required input values,
%        Corr: The cross correlation signal (array)
%        phase: Detector phase shift steps (array)
%
% Return: There are five return values for this function,
%        a0: fitting parameter
%        a1: fitting parameter
%        b1: fitting parameter
%        theta: calculated phase shift
%        mod: calculated modulation
%
% Written by: Adam Norige 6/24/03
numImagesToAnalyze = length(corr);        %Number of correlation data points
numPhaseDelaysToAnalyze = length(phase);  %Number of something or other

% Perform some preliminary calculations
sinThetaDArray = sin(phase);
cosThetaDArray = cos(phase);

% Build the A matrix (3 X 3)
matrix_A = [ sum(sinThetaDArray.*sinThetaDArray) sum(sinThetaDArray.*cosThetaDArray)
sum(sinThetaDArray);
             sum(sinThetaDArray.*cosThetaDArray) sum(cosThetaDArray.*cosThetaDArray)
sum(cosThetaDArray);
             sum(sinThetaDArray)                 sum(cosThetaDArray)
numImagesToAnalyze];

% Build the B matrix (3 X 1) by reading specified raw intensity images
matrix_B = zeros(3, 1);                % initialize to zero
imageIndex = 1;                        % start at first row of phase information matrix
%phosIntensityArray = zeros(numImagesToAnalyze);

for i = 1:numImagesToAnalyze
    phosIntensityArray(imageIndex) = corr(imageIndex);  % store mean value in first
column

    matrix_B = matrix_B + [sinThetaDArray(imageIndex) * phosIntensityArray(imageIndex);
                           cosThetaDArray(imageIndex) * phosIntensityArray(imageIndex);
                           phosIntensityArray(imageIndex)                             ];

    imageIndex = imageIndex + 1;                         % next row of matrix
end

% build the 3 unknown parameter images a0, a1, and b1 by solving AX = B for X
matrix_X = matrix_A \ matrix_B;
b1 = matrix_X(1,:);
a1 = matrix_X(2,:);
a0 = matrix_X(3,:);
```

www.manaraa.com

```
% check for zeroes in a0 and a1
if a0 == 0.0            % reject fits with a0 = 0
    a0 = eps;           % make a0 a very, very small number
end

if a1 == 0.0            % reject fits with a1 = 0
    a1 = eps;           % make a1 a very, very small number
end

% Generate phase shift from fit data
thetaCorrection = 0;          % correction phase for instrumentation delay (assumed zero
for now)
theta = atan(b1/a1);
theta = theta;

% Create amplitude modulation map (modMap) from fit data
modCorrection = 1;            % correction modulation (excitation modulation - assumed 1)
mod = sqrt(a1*a1 + b1*b1);
mod = mod/(a0/2);             %dividing a0 by 2 is a correction
mod = mod*modCorrection;
```

% End Curve_bs_fit.m

المنارة للاستشارات
www.manaraa.com

Appendix B – PLI Contamination "Excitation Spectrum Analyzer"

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Spectrum Analyzer for PDI with zero phase contamination.
%
%  Created by: Adam Norige 8/27/03
%
%   This version of the frequency domain imaging (PLI) simulator
%    contaminates the phosphorescence signal with backscattered light
%    The back scattered light is incorporated by cross corralating the
%    excitation signal and adding the two x-corr results together
%    (phosp and excite).  This simulator makes no attempt to remove
%    the back scattered light from the phosphorescence signal, this
%    function m-file is intended to display the behavior of the
%    combination of true phosphorescence and back scattered light.
%
%   This "version" of PLI_Spectrum.m allows calculations to be
%    conducted over a selectable range of frequencies.  Summary
%    plots are generated at the end of the simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function PLI_BS_Spectrum()

global Kq T0

%Probe = pd-meso-tetra[4carboxyohenyl] porphine
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% CONSTANTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kq = 381;             %Quenching constant (1/torr*1/s) @ pH=7.4 & 38 C
T0 = .000637;         %Lifetime w/o quenching (s) @ pH=7.4 & 38 C

%------------------------------------------------
Po2_1=27;             %PO2 value 1
Po2_2=27;             %P02 value 2
%------------------------------------------------

Po2_wt1 =0.5;         %Percent Weight for PO2 1 (less than 1)
%                     %Percent weight for PO2 2 is calculated

a = 5;                %DC offset of excitation signal
b = 5;                %Half the amplitude of excitation signal
%   Note a is ALWAYS larger than b
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% VARIABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tau_1 = 0;              %Actual Lifetime value 1
Tau_2 = 0;              %Actual Lifetime value 2
theta_1 = 0;            %Phosphorescence lifetime phase 1
theta_2 = 0;            %Phosphorescence lifetime phase 2
mag_1   = 0;            %Phosphorescence magnitude 1
mag_2   = 0;            %Phosphorescence magnitude 2
omega   = 0;            %Initial Modulation Frequency

A_1 = 0;                %DC offset of emission signal 1
B_1 = 0;                %Half the amplitude of the excitation signal 1
A_2 = 0;                %DC offset of emission signal 2
B_2 = 0;                %Half the amplitude of the excitation signal 2
AB_ratio = 0;           %Ratio of A and B ... B/A *temp variable*
```

www.manaraa.com

```
n = 1;                  %Counter for arrays to hold the PO2s for each freq

%-------------------------------------------------
freq = 500;             %Starting Frequency
index = 1;              %Index for recorded data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    LIFETIME CALCULATION
Tau_1 = T0/(1+Kq*T0*Po2_1);        %Stern Volmer Equation, PO2 = [q]
Tau_2 = T0/(1+Kq*T0*Po2_2);
jk=1;                              %Modulation Index

%Main Loop to cycle through frequencies
while (freq <= 5000)
    omega = freq*2*pi;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    EMISSION PHASE
    theta_1 = atan(omega*Tau_1);                %Calculate Theta
    theta_2 = atan(omega*Tau_2);                %tan(theta) = Omega*Tau
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    EMISSION MAGNITUDE
    mag_1 = 1/(sqrt(1+omega^2*Tau_1^2));     %Calculate Mw for PO2_1
    AB_ratio = mag_1*(b/a);                  %Determine A & B for PO2_1
    A_1=5;                                   %No baseline change
    B_1=5;%AB_ratio*A_1;                        %Here's B
    mag_1v(jk) = mag_1;

    mag_2 = 1/(sqrt(1+omega^2*Tau_2^2));     %Calculate Mw for PO2_2
    AB_ratio = mag_1*(b/a);                  %Determine A & B for PO2_1
    A_2=5;                                   %No baseline change
    B_2=5;%AB_ratio*A_2;                        %Here's B
    mag_2v(jk) = mag_2;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    GENERATE THE TIME ARRAY
    t_step = 1/(omega/(2*pi)*100);                       %Time step for time domain
    t=0:t_step:(3/(omega/(2*pi)))-t_step;                %Generate Time Array
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    GENERATION of SINUSOIDS
    Excite =a+b*cos(omega*t);                            %Zero Phase shift Signal
    Emission_1 = double(A_1+B_1*cos(omega*t+theta_1)); %PO2_1 Emission
    Emission_2 = A_2+B_2*cos(omega*t+theta_2);           %PO2_2 Emission
    Emission_Sum = (Po2_wt1*Emission_1+(1-Po2_wt1)*Emission_2);   %Combined Emission
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    PERFORM CROSS CORRELATION
    [Corr,phase]=Cross_Corr(Emission_Sum, t, omega);
    [CorrS,phase]=Cross_Corr(Excite,t,omega);           %Cross Correlate the back scatter
    Corr = .7*Corr+.3*CorrS;                                  %Create the contaminated
signal
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    CURVE FIT
    [theta_fit, mod_fit]=Curve_fit(Corr,phase);         %Fit Curve to x-corr data
    Tau_fit = (sin(theta_fit)/cos(theta_fit))/omega;    %Calculate Tau
    PO2_fit(n) = ((T0/Tau_fit)-1)/(Kq*T0);              %Calculate PO2

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %    UPDATE FREQUENCY and RECORD DATA
    phase_1p(index) = theta_1*(180/pi);      %Record original phase (When P02s are equal)
    phase_fit(index) = theta_fit*(180/pi);   %Record calculated phase, curve fit
```

www.manaraa.com

```
    mag_p (index) = mag_1;                   %Record original modulaton
    magc_p(index) = mod_fit;                 %Record calculated modulation, curve fit

    po2f_p(index) = PO2_fit(n);              %Record PO2 data from linear fit
    po2_real(index)= Po2_1;                   %What the PO2 should be

    freq_p(index) = freq;                    %Record frequency
    index=index+1;                           %Update index

    freq = freq+200;                         %Next frequency
    jk=jk+1;                                 %Next Index Value
    n=n+1;                                   %Next element in the array
end     % End of While loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    PLOT RECORD DATA
figure(1);
hold on;
plot(freq_p, phase_1p, '-r^', 'markersize', 3, 'MarkerFaceColor', 'r');
plot(freq_p, phase_fit, '-k+', 'markersize', 3, 'MarkerFaceColor', 'k');
xlabel('Frequency (Hz)');
ylabel('Phase (Degrees)');
legend('Original', 'Calculated (fit)');
title('Phase Vs. Frequency');
grid on;
hold off;

figure(2);
hold on;
plot(freq_p, mag_p, '-r^', 'markersize', 3, 'MarkerFaceColor', 'b');
plot(freq_p, magc_p, '-k+', 'markersize', 3, 'MarkerFaceColor', 'r');
xlabel('Frequency (Hz)');
ylabel('Modulation');
legend('Original','Calculated (fit)');
title('Modulation Vs. Frequency');
grid on;
hold off;

figure(3);
hold on;
plot(freq_p,po2f_p, '-bx');
plot(freq_p,po2_real, '-ro');
legend('Contaminated PO_2', 'True PO_2');
xlabel('Frequency (Hz)');
ylim([0,90]);
ylabel('PO_2');
title('PO_2 Vs. Frequency');
grid on
hold off;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %    CALCULATE DERIVATIVES
 slope_po2_fit = diff(po2_fit);    %Determine the Derivative

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %    PLOT DERIVATIVES
```

www.manaraa.com

```
figure (4);
hold on
title('1st Derivative of the Phase-Frequency Trend for a 50% Contaminated Signal');
plot(slope_po2_fit, '-rx');
legend('Linear Fit');
grid on;
hold off

figure (5);
hold on;
title('2nd Derivative of the Phase-Frequency Trend for a 50% Contaminated Signal');
plot(diff(slope_po2_fit), '-rx');
legend('Linear Fit');
grid on;
hold off;


%end PLI_BS_Spectrum.m
```

المنارة للاستشارات
www.manaraa.com

Appendix C – Retina2 Analysis Software Source Code

Only new source files or source files modified from the original “Retina” program are included in this index. Source files that are unmodified from the “Retina” program but are called by the “Retina2” program are not included in this appendix.

Retina2.m

```
function varargout = retina2(varargin)
% RETINA2 Application M-file for retina.fig
%    FIG = RETINA2 launch retina GUI.
%
%    RETINA2 Is used to evaluate two phase delay intensity maps at once
%       with the intent of removing all zero-phase signals from the
%       images.  Two images acquired at different modulation frequencies
%       are required to successfully remove the zero-phase signals.
%       RETINA2 allows analysis of the individual images, de-contaminated
%       image, and zero-phase contamination image.  RETINA2 also allows,
%       P02 maps to be generated for the de-contaminated image.
%
%      H = RETINA2 returns the handle to a new RETINA2 or the handle to
%      the existing singleton*.
%
%
% RETINA2 (Created by Adam Norige) was developed from RETINA (Created by Ross Shonat)
% Created by Adam Norige 22-Sep-2003
% Last Modified by GUIDE v2.0 03-Mar-2004 10:32:31

% declare global variables
global kQ tauO filterPhos bckGnd thetaCorrection modCorrection
global xDimension yDimension
global modFreq1 omega1 pulseWidth1 delayStart1 delayIncr1
global modFreq2 omega2 pulseWidth2 delayStart2 delayIncr2
global regionMap1 xPolyCoordinates1 yPolyCoordinates1 imageFullMatrix1
global regionMap2 xPolyCoordinates2 yPolyCoordinates2 imageFullMatrix2
global mapNames filterParams map1Value map2Value

if nargin == 0  % LAUNCH GUI

    % define initial values for important GLOBAL parameters
    kQ = 381.0;                 % quenching constant
    tauO = 637.0;               % probe lifetime in the absence of oxygen
    filterPhos = 0;             % spatial filter for phosphorescence intensity maps (0:
no spatial filtering)
    modFreq1 = 500.0;          % modulation frequency (in Hz)
    modFreq2 = 2000.0;          % modulation frequency (in Hz)
    delayStart1 = 5;            % starting delay (us)
    delayStart2 = 5;            % starting delay (us)
    delayIncr1 = 142;           % incremental delay (us)
    delayIncr2 = 35;            % incremental delay (us)
    bckGnd = 141.0;             % average camera background noise level (AU)
```

www.manaraa.com

```
    thetaCorrection = 0;   % correction phase for instrumentation delay (radians from in
vitro tests)0.0005166
    modCorrection =1;      % correction modulation (excitation modulation - from in vitro
tests)  0.636

    % perform some initial calculations
    omega1 = modFreq1 * 2 * pi;   % calculated modulation frequency (in rad/sec)
    omega2 = modFreq2 * 2 * pi;   % calculated modulation frequency (in rad/sec)
    pulseWidth1 = 1000000.0 / (modFreq1 * 2);     % pulse width = 1/2 of sine wave period
    pulseWidth2 = 1000000.0 / (modFreq2 * 2);     % pulse width = 1/2 of sine wave period

    % create array of map names for mapping display
    mapNames = strvcat('Phase Shift (Theta) Map','Modulation Map','DC-Level Map','R2
Map',...
        'Lifetime (Theta) Map','PO2 (Theta) Map','Rejection (Theta) Map','Lifetime (Mod)
Map',...
        'PO2 (Mod) Map','Rejection (Mod) Map','a1Map','b1Map');
    map1Value = 1;                      % default to PO2 (Theta) Map

    % set original region for analysis (all of image)
    xDimension = 512;            % assume for the moment that the x-axis dimension is 512
    yDimension = 512;            % assume for the moment that the y-axis dimension is 512
    regionMap1 = ones([yDimension xDimension]);
    xPolyCoordinates1 = [1; xDimension; xDimension; xDimension ; xDimension; 1; 1; 1];
    yPolyCoordinates1 = [1; 1; 1; yDimension; yDimension; yDimension; 1; 1];
    regionMap2 = ones([yDimension xDimension]);
    xPolyCoordinates2 = [1; xDimension; xDimension; xDimension ; xDimension; 1; 1; 1];
    yPolyCoordinates2 = [1; 1; 1; yDimension; yDimension; yDimension; 1; 1];

    % create the filterParams matrix (col 1: minValue, col 2: maxValue, col 3: minR2)
    filterParams = [0.0     180.0   0.0;             % thetaMap (1)
                    0.0       1.0   0.0;             % modMap (2)
                    0.0   10000.0   0.0;             % DCMap (3)
                    0.0       1.0   0.0;             % R2Map (4)
                    0.0     800.0   0.0;             % tauThetaMap (5)
                    0.0     200.0   0.0;             % PO2ThetaMap (6)
                    0.0       1.0   0.0;             % rejectThetaMap (7)
                    0.0     800.0   0.0;             % tauModMap (8)
                    0.0     200.0   0.0;             % PO2ModMap (9)
                    0.0       1.0   0.0;             % rejectModMap (10)
                    0.0   20000.0   0.0;             % Case 11 Map (11)
                    0.0   20000.0   0.0];            % Case 12 Map (12)

    % open the interface control window
    interfaceFig = openfig(mfilename,'reuse');
       %**** Note interfaceFig is the same as interfaceHandles.retina2Tag
        set(interfaceFig,'Color',get(0,'DefaultUicontrolBackgroundColor'));

   % Generate a structure of handles to pass to callbacks, and store it.
   interfaceHandles = guihandles(interfaceFig);

    % Write handle structure back
    guidata(interfaceFig, interfaceHandles);

    %Populate the interface control screen with parameters
    set(interfaceHandles.modFreqTag1,'String',num2str(modFreq1, '%6.1f'));
    set(interfaceHandles.pulseWidthTag1,'String',num2str(pulseWidth1, '%6.2f'));
```

www.manaraa.com

```
    set(interfaceHandles.delayStartTag1,'String',num2str(delayStart1, '%3.0f'));
    set(interfaceHandles.delayIncrTag1,'String',num2str(delayIncr1, '%3.0f'));
    set(interfaceHandles.modFreqTag2,'String',num2str(modFreq2, '%6.1f'));
    set(interfaceHandles.pulseWidthTag2,'String',num2str(pulseWidth2, '%6.2f'));
    set(interfaceHandles.delayStartTag2,'String',num2str(delayStart2, '%3.0f'));
    set(interfaceHandles.delayIncrTag2,'String',num2str(delayIncr2, '%3.0f'));
    set(interfaceHandles.probeKqTag,'String',num2str(kQ, '%4.1f'));
    set(interfaceHandles.tauOTag,'String',num2str(tauO, '%4.1f'));
    set(interfaceHandles.filterPhosTag,'String',num2str(filterPhos, '%2.0f'));
    set(interfaceHandles.phaseErrTag,'String',num2str((thetaCorrection * 180 / pi),
'%6.4f'));
    set(interfaceHandles.modErrTag,'String',num2str(modCorrection, '%6.4f'));

    % Query for WinX Data file to start with (Image Set 1)
    getFile1_Callback(interfaceFig, [], interfaceHandles);

    if nargout > 0
      varargout{1} = interfaceFig;
   end

    %Obtain the updated "interfaceHandles" from the figure
    interfaceHandles = guidata(interfaceFig);

    % Query for WinX Data file to start with (Image Set 2)
    getFile2_Callback(interfaceFig, [], interfaceHandles);

    if nargout > 0
      varargout{1} = interfaceFig;
   end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
      [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
   catch
      disp(lasterr);
   end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- Image Set Number 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------------
% --- Executes on button press in getFile1.
function getFile1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global fileName1 filePath1
global rightshift leftshift upshift downshift

% Get the WinX image set and return information about the data file
```

www.manaraa.com

```
[filePath, fileName, xDimension, yDimension, numImages, dataType] = GetWinXDataSetInfo;

if (~isequal(filePath,-1) & ~isequal(fileName,-1))
    filePath1 = filePath;
    fileName1 = fileName;
    cd(filePath1);          % make data set file path current

    % Store WinX data file information into handle structure
    interfaceHandles.xDimension1 = xDimension;
    interfaceHandles.yDimension1 = yDimension;
    interfaceHandles.numImages1  = numImages;
    interfaceHandles.dataType1   = dataType;

    % Put WinX data file information onto figure screen
    set(interfaceHandles.fileNameTag1,'String',fileName1);
    set(interfaceHandles.filePathTag1,'String',filePath1);
    set(interfaceHandles.xDimensionTag1,'String',num2Str(xDimension,'%3.0f'));
    set(interfaceHandles.yDimensionTag1,'String',num2str(yDimension,'%3.0f'));
    set(interfaceHandles.numImagesTag1,'String',num2str(numImages,'%3.0f'));
    set(interfaceHandles.dataTypeTag1,'String',num2str(dataType,'%1.0f'));

    %Reset the Image Shifts
    rightshift = 0;
    leftshift  = 0;
    upshift    = 0;
    downshift  = 0;

    % calculate and update image delay info on screen
    updateImageDelayInformation(hObject, interfaceHandles,1);
end

%----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%----------------------------------------------------------------------------
% --- Executes on button press of useTag1.
function useTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix1

imageNumber = get(interfaceHandles.imageNumPopupTag1,'Value');
imageFullMatrix1(imageNumber,2) = get(interfaceHandles.useTag1,'Value');

%----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%----------------------------------------------------------------------------
% --- Executes on selection change in imageNumPopupTag1.
function imageNumPopupTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix1

imageNumber = get(interfaceHandles.imageNumPopupTag1,'Value');
%Update Display information (use,delay us,delay deg) accordingly
set(interfaceHandles.imageNumPopupTag1,'Value',imageNumber);
```

المنارة للاستشارات
www.manaraa.com

```
set(interfaceHandles.useTag1,'Value',imageFullMatrix1(imageNumber,2));
set(interfaceHandles.delayTag1,'String',num2str(imageFullMatrix1(imageNumber,3),
'%5.1f'));
set(interfaceHandles.degreeTag1,'String',num2str(imageFullMatrix1(imageNumber,4),
'%5.3f'));

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
% --- Executes on button press in updateTag1.
function updateTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix1 imagePhaseMatrix1

% image phase matrix is a subset of the full image matrix
imagePhaseMatrix1 = imageFullMatrix1;

%determine number of rows to read
[nrows, ncols] = size(imageFullMatrix1);

phaseMatrixRow = 1;  % start at first row
for i = 1:nrows
    if imageFullMatrix1(i,2) == 0.0           % eliminate rows where "use" flag is zero
(unchecked)
        imagePhaseMatrix1(phaseMatrixRow,:) = [];
    else
        phaseMatrixRow = phaseMatrixRow + 1;  % increment row counter only if row not
deleted
    end
end

% generate imagePhaseMatrix (column 1: image numbers to use, column 2: phase shift in
radians)
imagePhaseMatrix1(:,2) = []; % eliminate column for storing the "use" flag
imagePhaseMatrix1(:,2) = []; % eliminate column for storing phase shift in microseconds
imagePhaseMatrix1(:,2) = []; % eliminate column for storing phase shift in degrees

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function modFreqTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global modFreq1 omega1

modFreq1 = str2num(get(interfaceHandles.modFreqTag1,'String'));   % get modulation
frequency (in Hz)
omega1 = modFreq1 * 2 * pi;      % store also in radian format (rad/sec)
% Re-write the freq1 field to have the correct format
set(interfaceHandles.modFreqTag1,'String',num2str(modFreq1, '%5.1f'));

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
```

المنارة للاستشارات

www.manaraa.com

```
function pulseWidthTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global pulseWidth1

pulseWidth1 = str2num(get(interfaceHandles.pulseWidthTag1,'String'));   % get pulse width
(in usec)
set(interfaceHandles.pulseWidthTag1,'String',num2str(pulseWidth1, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,1)

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function delayStartTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global delayStart1

delayStart1 = str2num(get(interfaceHandles.delayStartTag1,'String'));   % get intensifier
delay start time
set(interfaceHandles.delayStartTag1,'String',num2str(delayStart1, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,1)   % calculate and update image
delay info on screen

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function delayIncrTag1_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global delayIncr1

delayIncr1 = str2num(get(interfaceHandles.delayIncrTag1,'String'));   % get intensifier
delay increment time
set(interfaceHandles.delayIncrTag1,'String',num2str(delayIncr1, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,1)     % calculate and update image
delay info on screen

%--------------------------------------------------------------------------



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- Image Set Number 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------------
% --- Executes on button press in getFile2.
function getFile2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global fileName2 filePath2
```

www.manaraa.com

```
global rightshift leftshift upshift downshift

% Get the WinX image set and return information about the data file
[filePath, fileName, xDimension, yDimension, numImages, dataType] = GetWinXDataSetInfo;
if (~isequal(filePath,-1) & ~isequal(fileName,-1))
    filePath2 = filePath;
    fileName2 = fileName;
    cd(filePath2);                                   % make data set file path current

    % Store WinX data file information into handle structure
    interfaceHandles.xDimension2 = xDimension;
    interfaceHandles.yDimension2 = yDimension;
    interfaceHandles.numImages2 = numImages;
    interfaceHandles.dataType2 = dataType;

    % Put WinX data file information onto figure screen
    set(interfaceHandles.fileNameTag2,'String',fileName2);
    set(interfaceHandles.filePathTag2,'String',filePath2);
    set(interfaceHandles.xDimensionTag2,'String',num2Str(xDimension,'%3.0f'));
    set(interfaceHandles.yDimensionTag2,'String',num2str(yDimension,'%3.0f'));
    set(interfaceHandles.numImagesTag2,'String',num2str(numImages,'%3.0f'));
    set(interfaceHandles.dataTypeTag2,'String',num2str(dataType,'%1.0f'));

    %Reset the Image Shifts
    rightshift = 0;
    leftshift  = 0;
    upshift    = 0;
    downshift  = 0;

    updateImageDelayInformation(hObject, interfaceHandles,2)   % calculate and update
image delay info on screen
end
%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
% --- Executes on button press in useTag2.
function useTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix2

imageNumber = get(interfaceHandles.imageNumPopupTag2,'Value');
imageFullMatrix2(imageNumber,2) = get(interfaceHandles.useTag2,'Value');

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
% --- Executes on selection change in imageNumPopupTag2.
function imageNumPopupTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix2

imageNumber = get(interfaceHandles.imageNumPopupTag2,'Value');
%Update Display information (use,delay us,delay deg) accordingly
```

المنارة للاستشارات
www.manaraa.com

```
set(interfaceHandles.imageNumPopupTag2,'Value',imageNumber);
set(interfaceHandles.useTag2,'Value',imageFullMatrix2(imageNumber,2));
set(interfaceHandles.delayTag2,'String',num2str(imageFullMatrix2(imageNumber,3),
'%5.1f'));
set(interfaceHandles.degreeTag2,'String',num2str(imageFullMatrix2(imageNumber,4),
'%5.3f'));
%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
% --- Executes on button press in updateTag2.
function updateTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global imageFullMatrix2 imagePhaseMatrix2

% image phase matrix is a subset of the full image matrix
imagePhaseMatrix2 = imageFullMatrix2;

%determine number of rows to read
[nrows, ncols] = size(imageFullMatrix2);

phaseMatrixRow2 = 1; % start at first row
for i = 1:nrows
    if imageFullMatrix2(i,2) == 0.0                  % eliminate rows where "use" flag is
zero (unchecked)
        imagePhaseMatrix2(phaseMatrixRow2,:) = [];
    else
        phaseMatrixRow2 = phaseMatrixRow2 + 1;        % increment row counter only if row
not deleted
    end
end

% generate imagePhaseMatrix (column 1: image numbers to use, column 2: phase shift in
radians)
imagePhaseMatrix2(:,2) = []; % eliminate column for storing the "use" flag
imagePhaseMatrix2(:,2) = []; % eliminate column for storing phase shift in microseconds
imagePhaseMatrix2(:,2) = []; % eliminate column for storing phase shift in degrees

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function modFreqTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global modFreq2 omega2

modFreq2 = str2num(get(interfaceHandles.modFreqTag2,'String'));   % get modulation
frequency (in Hz)
omega2 = modFreq2 * 2 * pi;      % store also in radian format (rad/sec)
set(interfaceHandles.modFreqTag2,'String',num2str(modFreq2, '%5.1f'));

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function pulseWidthTag2_Callback(hObject, eventdata, interfaceHandles)
```

المنارة للاستشارات
www.manaraa.com

```
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global pulseWidth2

pulseWidth2 = str2num(get(interfaceHandles.pulseWidthTag2,'String'));   % get pulse width
(in usec)
set(interfaceHandles.pulseWidthTag2,'String',num2str(pulseWidth2, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,2)   % calculate and update image
delay info on screen

%-------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-------------------------------------------------------------------------
function delayStartTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global delayStart2

delayStart2 = str2num(get(interfaceHandles.delayStartTag2,'String'));   % get intensifier
delay start time
set(interfaceHandles.delayStartTag2,'String',num2str(delayStart2, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,2)   % calculate and update image
delay info on screen

%-------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-------------------------------------------------------------------------
function delayIncrTag2_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global delayIncr2

delayIncr2 = str2num(get(interfaceHandles.delayIncrTag2,'String'));   % get intensifier
delay increment time
set(interfaceHandles.delayIncrTag2,'String',num2str(delayIncr2, '%5.1f'));
updateImageDelayInformation(hObject, interfaceHandles,2)   % calculate and update image
delay info on screen

%-------------------------------------------------------------------------


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- General
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------------------------
function probeKqTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global kQ

kQ = str2num(get(interfaceHandles.probeKqTag,'String'));
```

www.manaraa.com

```
%Check the validity of kQ value
if kQ >= 2000.0
    kQ = 2000.0;
elseif kQ <= 1.0
    kQ = 1.0;
end
set(interfaceHandles.probeKqTag,'String',num2str(kQ,'%4.1f'));

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function tauOTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global tauO

tauO = str2num(get(interfaceHandles.tauOTag,'String'));
%Check the validity of tau0 value
if tauO >= 999.0
    tauO = 999.0;
elseif tauO <= 1.0
    tauO = 1.0;
end
set(interfaceHandles.tauOTag,'String',num2str(tauO,'%4.1f'));

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function filterPhosTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterPhos

filterPhos = str2num(get(interfaceHandles.filterPhosTag,'String'));
%Check the validity of filterPhos value
if filterPhos >= 10
    filterPhos = 10;
elseif filterPhos <= 0
    filterPhos = 0;
end
set(interfaceHandles.filterPhosTag,'String',num2str(filterPhos,'%2.0f'));

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function phaseErrTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global thetaCorrection

degreeCorrection = str2num(get(interfaceHandles.phaseErrTag,'String'));
thetaCorrection = degreeCorrection * pi / 180;
set(interfaceHandles.phaseErrTag,'String',num2str(degreeCorrection,'%3.1f'));
```

المنارة للاستشارات
www.manaraa.com

```
%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function modErrTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global modCorrection

modCorrection = str2num(get(interfaceHandles.modErrTag,'String'));
%Check the validity of modCorrection value
if modCorrection >= 1
    modCorrection = 1;
elseif modCorrection <= 0
    modCorrection = 0;
end
set(interfaceHandles.modErrTag,'String',num2str(modCorrection, '%6.4f'));

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function alignImgTag_Callback(hObject, eventdata, handles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
alignImages;

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function calculateTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosMapDisplay2;

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function ShowIntensImgTag_Callback(hObject, eventdata, interfaceHandles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplay2;

%--------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   FUNCTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------------
function updateImageDelayInformation(hObject, interfaceHandles, imageset)

global pulseWidth1 delayStart1 delayIncr1 imageFullMatrix1
global pulseWidth2 delayStart2 delayIncr2 imageFullMatrix2
```

المنارة للاستشارات
www.manaraa.com

```
% create the imageFullMatrix (matrix of image numbers, use stats, and phase delays to use
in analysis)
if (imageset == 1)
    imageNumbers = [1:1:interfaceHandles.numImages1]';  % column of image numbers
    imageUse = ones(interfaceHandles.numImages1,1);     % assume that all of these images
will be used (= 1)
    % specify phase delay for these images (in usec)
    phaseDelayMicroSeconds = [delayStart1:delayIncr1:(delayStart1 + delayIncr1 *
(interfaceHandles.numImages1-1))]';
    imageUse(1) = 0;                                    % except first image, which is
rejected by default
    phaseDelayDegrees = phaseDelayMicroSeconds*180.0/pulseWidth1;  % convert to degrees
    phaseDelayRadians = phaseDelayDegrees*pi/180;       % convert to radians
    % store phase information in full 5 column matrix
    imageFullMatrix1 = [imageNumbers imageUse phaseDelayMicroSeconds phaseDelayDegrees
phaseDelayRadians];
elseif (imageset == 2)
    imageNumbers = [1:1:interfaceHandles.numImages2]';  % column of image numbers
    imageUse = ones(interfaceHandles.numImages2,1);     % assume that all of these images
will be used (= 1)
    % specify phase delay for these images (in usec)
    phaseDelayMicroSeconds = [delayStart2:delayIncr2:(delayStart2 + delayIncr2 *
(interfaceHandles.numImages2-1))]';
    imageUse(1) = 0;                                   % except first image, which is
rejected by default
    phaseDelayDegrees = phaseDelayMicroSeconds*180.0/pulseWidth2;  % convert to degrees
    phaseDelayRadians = phaseDelayDegrees*pi/180;      % convert to radians
    % store phase information in full 5 column matrix
    imageFullMatrix2 = [imageNumbers imageUse phaseDelayMicroSeconds phaseDelayDegrees
phaseDelayRadians];
end

imageNames = '';
if (imageset == 1)
    for i=1:interfaceHandles.numImages1
        imageNames = strvcat(imageNames,num2str(i,'%2.0f'));
    end
elseif (imageset == 2)
    for i=1:interfaceHandles.numImages2
        imageNames = strvcat(imageNames,num2str(i,'%2.0f'));
    end
end

% load values into image identification windows
imageNumber = 1;   % default to first image
if (imageset ==1)
    set(interfaceHandles.imageNumPopupTag1,'String',imageNames);
    set(interfaceHandles.imageNumPopupTag1,'Value',imageNumber);
    set(interfaceHandles.useTag1,'Value',imageFullMatrix1(imageNumber,2));
    set(interfaceHandles.delayTag1,'String',num2str(imageFullMatrix1(imageNumber,3),
'%5.1f'));
    set(interfaceHandles.degreeTag1,'String',num2str(imageFullMatrix1(imageNumber,4),
'%5.3f'));
    % create the image phase matrix used for calculations
    updateTag1_Callback(hObject,[], interfaceHandles);
elseif (imageset == 2)
    set(interfaceHandles.imageNumPopupTag2,'String',imageNames);
```

المنارة للاستشارات
www.manaraa.com

```
    set(interfaceHandles.imageNumPopupTag2,'Value',imageNumber);
    set(interfaceHandles.useTag2,'Value',imageFullMatrix2(imageNumber,2));
    set(interfaceHandles.delayTag2,'String',num2str(imageFullMatrix2(imageNumber,3),
'%5.1f'));
    set(interfaceHandles.degreeTag2,'String',num2str(imageFullMatrix2(imageNumber,4),
'%5.3f'));
    % create the image phase matrix used for calculations
    updateTag2_Callback(hObject,[], interfaceHandles);
end

%Store all changes in the "interfaceHandles" structure
guidata(hObject, interfaceHandles);
%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


% --------------------------------------------------------------------
function varargout = retina2Tag_CloseRequestFcn(hObject, eventdata, interfaceHandles)

selection = questdlg('Remove All Global Variables?',...
                     'Close Request Function',...
                     'Yes','No','Yes');
switch selection,
   case 'Yes',
     delete(hObject)
     clear global
   case 'No'
     delete(hObject)
 end

%end retina2.m
```

alignImages.m

```
function varargout = alignImagesFig(varargin)
% ALIGNIMAGESFIG M-file for alignImagesFig.fig
%      ALIGNIMAGESFIG, by itself, creates a new ALIGNIMAGESFIG or raises the existing
%      singleton*.
%
%      H = ALIGNIMAGESFIG returns the handle to a new ALIGNIMAGESFIG or the handle to
%      the existing singleton*.
%
%      ALIGNIMAGESFIG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ALIGNIMAGESFIG.M with the given input
%      arguments.
%
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Created By: Adam Norige 10-14-2003

% Last Modified by GUIDE v2.0 24-Mar-2004 09:27:53



if nargin == 0  % LAUNCH GUI

    global xDimension yDimension
    global fileName1 filePath1 fileName2 filePath2
    global imagePhaseMatrix1 imagePhaseMatrix2

    %open the phosphorescence intensity display window
    alignImagesFig = openfig(mfilename,'reuse');
    set(alignImagesFig,'Color',get(0,'defaultUicontrolBackgroundColor'));

    % Generate a structure of handles to pass to callbacks, and store it.
   alignImagesHandles = guihandles(alignImagesFig);

    [outImage1, numImages, minPixelIntensity1, maxPixelIntensity1] =
GetWinXIntensityImage(filePath1, fileName1, 1);
    [outImage2, numImages, minPixelIntensity2, maxPixelIntensity2] =
GetWinXIntensityImage(filePath2, fileName2, 2);

    %Normalize the intensity values of the images
    y = IntensifierCorrect(ones([yDimension xDimension]), 0, 141);
    outImage2 = outImage2*y;

    min_outImage1 = min(min(outImage1));
    min_outImage2 = min(min(outImage2));

    if min_outImage1 < 0
        outImage1 = outImage1-min_outImage1;
    end

    if min_outImage2 < 0
        outImage2 = outImage2-min_outImage2;
    end
```

www.manaraa.com

```
    alignImagesHandles.redImg   = outImage1./max(max(outImage1));
    alignImagesHandles.greenImg = outImage2./max(max(outImage2));

    %alignImagesHandles.redImg = ones(size(outImage1));

    %Generate buffer Matrix to hold shifted Images
    alignImagesHandles.upBuffer    = zeros(floor(yDimension/2),xDimension);
    alignImagesHandles.downBuffer  = zeros(floor(yDimension/2),xDimension);
    alignImagesHandles.leftBuffer  = zeros(yDimension,floor(xDimension/2));
    alignImagesHandles.rightBuffer = zeros(yDimension,floor(xDimension/2));

    %Shift Counts
    alignImagesHandles.rshift = 0;  %Right Shift
    alignImagesHandles.lshift = 0;  %Left Shift
    alignImagesHandles.ushift = 0;  %Up Shift
    alignImagesHandles.dshift = 0;  %Down Shift

    %Display the composite Image
    DisplayComposite(alignImagesFig, alignImagesHandles);

    if nargout > 0
        varargout{1} = phosIntensityDisplayFig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
    catch
        disp(lasterr);
    end

end

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on button press in UpButtonTag.
function UpButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ShiftUp(hObject, 1, alignImagesHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on button press in LeftButtonTag.
function LeftButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ShiftLeft(hObject, 1, alignImagesHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
```

www.manaraa.com

```
% --- Executes on button press in RightButtonTag.
function RightButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ShiftRight(hObject, 1, alignImagesHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on button press in DownButtonTag.
function DownButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ShiftDown(hObject, 1, alignImagesHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on button press in ResetButtonTag.
function ResetButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global xDimension yDimension
global fileName1 filePath1 fileName2 filePath2

[outImage1, numImages, minPixelIntensity1, maxPixelIntensity1] =
GetWinXIntensityImage(filePath1, fileName1, 1);
[outImage2, numImages, minPixelIntensity2, maxPixelIntensity2] =
GetWinXIntensityImage(filePath2, fileName2, 2);

alignImagesHandles.redImg   = outImage1./max(max(outImage1));
alignImagesHandles.greenImg = outImage2./max(max(outImage2));

%Generate buffer Matrix to hold shifted Images
alignImagesHandles.upBuffer    = zeros(floor(yDimension/2),xDimension);
alignImagesHandles.downBuffer  = zeros(floor(yDimension/2),xDimension);
alignImagesHandles.leftBuffer  = zeros(yDimension,floor(xDimension/2));
alignImagesHandles.rightBuffer = zeros(yDimension,floor(xDimension/2));

%Reset Shift Counts
alignImagesHandles.rshift = 0;  %Right Shift
alignImagesHandles.lshift = 0;  %Left Shift
alignImagesHandles.ushift = 0;  %Up Shift
alignImagesHandles.dshift = 0;  %Down Shift

%Display the composite Image
DisplayComposite(hObject, alignImagesHandles);
%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on key press in alignImagesFig fig.
function KeyPress_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to alignImageFig (see GCF)
```

www.manaraa.com

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
key = get(hObject, 'CurrentCharacter');
if key == '8'
    ShiftUp(hObject, 1, alignImagesHandles);
elseif key == '2'
    ShiftDown(hObject, 1, alignImagesHandles);
elseif key == '6'
    ShiftRight(hObject, 1, alignImagesHandles);
elseif key == '4'
    ShiftLeft(hObject, 1, alignImagesHandles);
end

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
% --- Executes on button press in acceptButtonTag.
function acceptButtonTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to acceptButtonTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rightshift leftshift upshift downshift

%store the shifts as global variables
rightshift = alignImagesHandles.rshift;
leftshift  = alignImagesHandles.lshift;
upshift    = alignImagesHandles.ushift;
downshift  = alignImagesHandles.dshift;

close(hObject);
%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function pixelsUpTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to pixelsUpTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global yDimension
limit =  floor(yDimension/2);

ushift = str2num(get(alignImagesHandles.pixelsUpTag,'String'));
ushift = floor(ushift);  %Ensure value is a whole number
if (ushift < 0)
    ushift = 0;          %Value cant be negative
elseif (ushift >= limit)
    ushift = limit-1;   %Value cant be greater than image bounds
end

%Determine the Shift amounts and the direction
if (alignImagesHandles.ushift == 0 & alignImagesHandles.dshift == 0 )
    ShiftUp(hObject, ushift, alignImagesHandles);
elseif (alignImagesHandles.ushift ~= 0 & alignImagesHandles.dshift == 0 )
    if (ushift >= alignImagesHandles.ushift)
        ushift = ushift-alignImagesHandles.ushift;
        ShiftUp(hObject, ushift, alignImagesHandles);
    else
```

المنارة للاستشارات
www.manaraa.com

```
        ushift = alignImagesHandles.ushift-ushift;
        ShiftDown(hObject, ushift, alignImagesHandles);
    end
elseif (alignImagesHandles.ushift == 0 & alignImagesHandles.dshift ~= 0 )
    ushift = ushift+alignImagesHandles.dshift;
    ShiftUp(hObject, ushift, alignImagesHandles);
elseif (alignImagesHandles.ushift == 0 & alignImagesHandles.dshift ~= 0 )
    %This should never happen -- just reset for now
    ResetButtonTag_Callback(hObject, [], alignImagesHandles);
end

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function pixelsDownTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to pixelsDownTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global yDimension
limit =  floor(yDimension/2);

dshift = str2num(get(alignImagesHandles.pixelsDownTag,'String'));
dshift = floor(dshift);  %Ensure value is a whole number
if (dshift < 0)
    dshift = 0;          %Value cant be negative
elseif (dshift >= limit)
    dshift = limit -1;   %Value cant be greater than image bounds
end

%Determine the Shift amounts and the direction
if (alignImagesHandles.dshift == 0 & alignImagesHandles.ushift == 0 )
    ShiftDown(hObject, dshift, alignImagesHandles);
elseif (alignImagesHandles.dshift ~= 0 & alignImagesHandles.ushift == 0 )
    if (dshift >= alignImagesHandles.dshift)
        dshift = dshift-alignImagesHandles.dshift;
        ShiftDown(hObject, dshift, alignImagesHandles);
    else
        dshift = alignImagesHandles.dshift-dshift;
        ShiftUp(hObject, dshift, alignImagesHandles);
    end
elseif (alignImagesHandles.dshift == 0 & alignImagesHandles.ushift ~= 0 )
    dshift = dshift+alignImagesHandles.ushift;
    ShiftDown(hObject, dshift, alignImagesHandles);
elseif (alignImagesHandles.dshift == 0 & alignImagesHandles.ushift ~= 0 )
    %This should never happen -- just reset for now
    ResetButtonTag_Callback(hObject, [], alignImagesHandles);
end

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function pixelsRightTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to pixelsRightTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

المنارة للاستشارات
www.manaraa.com

```
global xDimension
limit =  floor(xDimension/2);

rshift = str2num(get(alignImagesHandles.pixelsRightTag,'String'));
rshift = floor(rshift);  %Ensure value is a whole number
if (rshift < 0)
    rshift = 0;          %Value cant be negative
elseif (rshift >= limit)
    rshift = limit -1;   %Value cant be greater than image bounds
end

%Determine the Shift amounts and the direction
if (alignImagesHandles.rshift == 0 & alignImagesHandles.lshift == 0 )
    ShiftRight(hObject, rshift, alignImagesHandles);
elseif (alignImagesHandles.rshift ~= 0 & alignImagesHandles.lshift == 0 )
    if (rshift >= alignImagesHandles.rshift)
        rshift = rshift-alignImagesHandles.rshift;
        ShiftRight(hObject, rshift, alignImagesHandles);
    else
        rshift = alignImagesHandles.rshift-rshift;
        ShiftLeft(hObject, rshift, alignImagesHandles);
    end
elseif (alignImagesHandles.rshift == 0 & alignImagesHandles.lshift ~= 0 )
    rshift = rshift+alignImagesHandles.lshift;
    ShiftRight(hObject, rshift, alignImagesHandles);
elseif (alignImagesHandles.rshift == 0 & alignImagesHandles.lshift ~= 0 )
    %This should never happen -- just reset for now
    ResetButtonTag_Callback(hObject, [], alignImagesHandles);
end

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function pixelsLeftTag_Callback(hObject, eventdata, alignImagesHandles)
% hObject    handle to pixelsLeftTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global xDimension
limit =  floor(xDimension/2);

lshift = str2num(get(alignImagesHandles.pixelsLeftTag,'String'));
lshift = floor(lshift);  %Ensure value is a whole number
if (lshift < 0)
    lshift = 0;          %Value cant be negative
elseif (lshift >= limit)
    lshift = limit -1;   %Value cant be greater than image bounds
end

%Determine the Shift amounts and the direction
if (alignImagesHandles.lshift == 0 & alignImagesHandles.rshift == 0 )
    ShiftLeft(hObject, lshift, alignImagesHandles);
elseif (alignImagesHandles.lshift ~= 0 & alignImagesHandles.rshift == 0 )
    if (lshift >= alignImagesHandles.lshift)
        lshift = lshift-alignImagesHandles.lshift;
        ShiftLeft(hObject, lshift, alignImagesHandles);
    else
```

المنارة للاستشارات
www.manaraa.com

```
        lshift = alignImagesHandles.lshift-lshift;
        ShiftRight(hObject, lshift, alignImagesHandles);
    end
elseif (alignImagesHandles.lshift == 0 & alignImagesHandles.rshift ~= 0 )
    lshift = lshift+alignImagesHandles.rshift;
    ShiftLeft(hObject, lshift, alignImagesHandles);
elseif (alignImagesHandles.lshift == 0 & alignImagesHandles.rshift ~= 0 )
    %This should never happen -- just reset for now
    ResetButtonTag_Callback(hObject, [], alignImagesHandles);
end
%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
% --- Displays the Composite Image in the figure
function DisplayComposite(hObject, alignImagesHandles)
%hObject    handle to alignImageFig (see GCF)
%alignImageHandles  structure containing handles and user data

%Create the composite
composite = cat(3,alignImagesHandles.redImg, alignImagesHandles.greenImg,
zeros(size(alignImagesHandles.redImg)));

%Display the Image
image(composite,'parent',alignImagesHandles.imageTag);

%Update counts
set(alignImagesHandles.pixelsRightTag,'String',
num2str(alignImagesHandles.rshift,'%4.0f'));
set(alignImagesHandles.pixelsLeftTag, 'String',
num2str(alignImagesHandles.lshift,'%4.0f'));
set(alignImagesHandles.pixelsUpTag,   'String',
num2str(alignImagesHandles.ushift,'%4.0f'));
set(alignImagesHandles.pixelsDownTag, 'String',
num2str(alignImagesHandles.dshift,'%4.0f'));

%Store all of the data in the figure's handles
guidata(hObject, alignImagesHandles);


%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function ShiftUp(hObject, ushift, alignImagesHandles)
%This function does the "up" shifting of the image.
% hObject =  The alignImages figure (alignImagesFig)
% ushift  =  Whole number representing up shift amount (pixels)
% alignImageHandles = alignImagesHandles structure
%This function automatically plots the shifted image and returns nothing.

global yDimension
limit =  floor(yDimension/2);

bufferBlock = limit-ushift;      %Calculation Parameter
imgBlock    = yDimension-ushift; %Calculation Parameter

if (sum(alignImagesHandles.upBuffer(1,:))==0 | ushift == 0); %Check if buffer is full of
real data
```

المنارة للاستشارات
www.manaraa.com

```
    %Shift "upBuffer" up

alignImagesHandles.upBuffer([1:bufferBlock],:)=alignImagesHandles.upBuffer([1+ushift:limi
t],:);

    %put top rows in "upbuffer" for storage

alignImagesHandles.upBuffer([bufferBlock+1:limit],:)=alignImagesHandles.redImg([1:ushift]
,:);

    %Shift redImg up one row

alignImagesHandles.redImg([1:imgBlock],:)=alignImagesHandles.redImg([1+ushift:yDimension]
,:);

    %put top row of "downbuffer" in redImg

alignImagesHandles.redImg([imgBlock+1:yDimension],:)=alignImagesHandles.downBuffer([1:ush
ift],:);

    %Shift "downbuffer" rows up

alignImagesHandles.downBuffer([1:bufferBlock],:)=alignImagesHandles.downBuffer(1+ushift:l
imit,:);
    alignImagesHandles.downBuffer([bufferBlock+1:limit],:)=0; %Clear these rows out

    % Keep the shift counts
    if (alignImagesHandles.dshift == 0)
        alignImagesHandles.ushift = alignImagesHandles.ushift+ushift;
    else
        if (ushift>=alignImagesHandles.dshift)
            alignImagesHandles.ushift = ushift-alignImagesHandles.dshift;
            alignImagesHandles.dshift=0;
        else
            alignImagesHandles.dshift=alignImagesHandles.dshift-ushift;
        end
    end

%     temp = cat(3,alignImagesHandles.upBuffer, zeros(size(alignImagesHandles.upBuffer)),
zeros(size(alignImagesHandles.upBuffer)));
%     temp1 = cat(3,alignImagesHandles.downBuffer,
zeros(size(alignImagesHandles.upBuffer)), zeros(size(alignImagesHandles.upBuffer)));
%     figure (1);
%     image(temp);
%     figure(2)
%     image(temp1);

    %Display the Images
    DisplayComposite(hObject, alignImagesHandles);
end
%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function ShiftDown(hObject, dshift, alignImagesHandles)
%This function does the "up" shifting of the image.
% hObject =  The alignImages figure (alignImagesFig)
```

المنارة للاستشارات
www.manaraa.com

```
% ushift  =  Whole number representing up shift amount (pixels)
% alignImageHandles = alignImagesHandles structure
%This function automatically plots the shifted image and returns nothing.

global yDimension
limit =  floor(yDimension/2);

bufferBlock = limit-dshift;      %Calculation Parameter
imgBlock    = yDimension-dshift; %Calculation Parameter

if (sum(alignImagesHandles.downBuffer(limit,:))==0 | dshift == 0); %Check if buffer is
full of real data

    %Shift "downBuffer" down

alignImagesHandles.downBuffer([1+dshift:limit],:)=alignImagesHandles.downBuffer([1:buffer
Block],:);

    %put bottom rows in "downbuffer" for storage

alignImagesHandles.downBuffer([1:dshift],:)=alignImagesHandles.redImg([imgBlock+1:yDimens
ion],:);

    %Shift redImg down

alignImagesHandles.redImg([dshift+1:yDimension],:)=alignImagesHandles.redImg([1:imgBlock]
,:);

    %put bottom row of "upbuffer" in redImg

alignImagesHandles.redImg([1:dshift],:)=alignImagesHandles.upBuffer([bufferBlock+1:limit]
,:);

    %Shift "upbuffer" rows down

alignImagesHandles.upBuffer([1+dshift:limit],:)=alignImagesHandles.upBuffer(1:bufferBlock
,:);
    alignImagesHandles.upBuffer([1:dshift],:)=0; %Clear these rows out

    % Keep the shift counts
    if (alignImagesHandles.ushift == 0)
        alignImagesHandles.dshift = alignImagesHandles.dshift+dshift;
    else
        if (dshift>=alignImagesHandles.ushift)
            alignImagesHandles.dshift = dshift-alignImagesHandles.ushift;
            alignImagesHandles.ushift=0;
        else
            alignImagesHandles.ushift=alignImagesHandles.ushift-dshift;
        end
    end
%     temp = cat(3,alignImagesHandles.upBuffer, zeros(size(alignImagesHandles.upBuffer)),
zeros(size(alignImagesHandles.upBuffer)));
%     temp1 = cat(3,alignImagesHandles.downBuffer,
zeros(size(alignImagesHandles.upBuffer)), zeros(size(alignImagesHandles.upBuffer)));
%     figure (1);
%     image(temp);
%     figure(2)
```

المنارة للاستشارات
www.manaraa.com

```
%     image(temp1);

    %Display the Images
    DisplayComposite(hObject, alignImagesHandles);
end

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function ShiftRight(hObject, rshift, alignImagesHandles)
%This function does the "right" shifting of the image.
% hObject =  The alignImages figure (alignImagesFig)
% ushift  =  Whole number representing right shift amount (pixels)
% alignImageHandles = alignImagesHandles structure
%This function automatically plots the shifted image and returns nothing.

global xDimension
limit =  floor(xDimension/2);

bufferBlock = limit-rshift;      %Calculation Parameter
imgBlock    = xDimension-rshift; %Calculation Parameter

if (sum(alignImagesHandles.rightBuffer(:,limit))==0 | rshift == 0); %Check if buffer is
full of real data

    %Shift "rightBuffer" right

alignImagesHandles.rightBuffer(:,[1+rshift:limit])=alignImagesHandles.rightBuffer(:,[1:bu
fferBlock]);

    %put right rows in "rightbuffer" for storage

alignImagesHandles.rightBuffer(:,[1:rshift])=alignImagesHandles.redImg(:,[imgBlock+1:xDim
ension]);

    %Shift redImg right

alignImagesHandles.redImg(:,[rshift+1:xDimension])=alignImagesHandles.redImg(:,[1:imgBloc
k]);

    %put right row of "leftbuffer" in redImg

alignImagesHandles.redImg(:,[1:rshift])=alignImagesHandles.leftBuffer(:,[bufferBlock+1:li
mit]);

    %Shift "leftbuffer" rows right

alignImagesHandles.leftBuffer(:,[1+rshift:limit])=alignImagesHandles.leftBuffer(:,[1:buff
erBlock]);
    alignImagesHandles.leftBuffer(:,[1:rshift])=0; %Clear these rows out

   % Keep the shift counts
    if (alignImagesHandles.lshift == 0)
        alignImagesHandles.rshift = alignImagesHandles.rshift+rshift;
    else
        if (rshift>=alignImagesHandles.lshift)
            alignImagesHandles.rshift = rshift-alignImagesHandles.lshift;
```

www.manaraa.com

```
            alignImagesHandles.lshift=0;
        else
            alignImagesHandles.lshift=alignImagesHandles.lshift-rshift;
        end
    end

%     temp = cat(3,alignImagesHandles.rightBuffer,
zeros(size(alignImagesHandles.leftBuffer)), zeros(size(alignImagesHandles.leftBuffer)));
%     temp1 = cat(3,alignImagesHandles.leftBuffer,
zeros(size(alignImagesHandles.leftBuffer)), zeros(size(alignImagesHandles.leftBuffer)));
%     figure (3);
%     image(temp);
%     figure(4)
%     image(temp1);

    %Display the Images
    DisplayComposite(hObject, alignImagesHandles);
end
%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function ShiftLeft(hObject, lshift, alignImagesHandles)
%This function does the "left" shifting of the image.
% hObject =  The alignImages figure (alignImagesFig)
% ushift  =  Whole number representing left shift amount (pixels)
% alignImageHandles = alignImagesHandles structure
%This function automatically plots the shifted image and returns nothing.

global xDimension
limit =  floor(xDimension/2);

bufferBlock = limit-lshift;      %Calculation Parameter
imgBlock    = xDimension-lshift; %Calculation Parameter

if (sum(alignImagesHandles.leftBuffer(:,1))==0 | lshift == 0); %Check if buffer is full
of real data

    %Shift "leftBuffer" up

alignImagesHandles.leftBuffer(:,[1:bufferBlock])=alignImagesHandles.leftBuffer(:,[1+lshif
t:limit]);

    %put left rows in "leftbuffer" for storage

alignImagesHandles.leftBuffer(:,[bufferBlock+1:limit])=alignImagesHandles.redImg(:,[1:lsh
ift]);

    %Shift redImg left one row

alignImagesHandles.redImg(:,[1:imgBlock])=alignImagesHandles.redImg(:,[1+lshift:xDimensio
n]);

    %put left row of "rightbuffer" in redImg

alignImagesHandles.redImg(:,[imgBlock+1:xDimension])=alignImagesHandles.rightBuffer(:,[1:
lshift]);
```

```
    %Shift "rightbuffer" rows left

alignImagesHandles.rightBuffer(:,[1:bufferBlock])=alignImagesHandles.rightBuffer(:,[1+lsh
ift:limit]);
    alignImagesHandles.rightBuffer(:,[bufferBlock+1:limit])=0; %Clear these rows out

    % Keep the shift counts
    if (alignImagesHandles.rshift == 0)
        alignImagesHandles.lshift = alignImagesHandles.lshift+lshift;
    else
        if (lshift>=alignImagesHandles.rshift)
            alignImagesHandles.lshift = lshift-alignImagesHandles.rshift;
            alignImagesHandles.rshift=0;
        else
            alignImagesHandles.rshift=alignImagesHandles.rshift-lshift;
        end
    end

%     temp = cat(3,alignImagesHandles.rightBuffer,
zeros(size(alignImagesHandles.leftBuffer)), zeros(size(alignImagesHandles.leftBuffer)));
%     temp1 = cat(3,alignImagesHandles.leftBuffer,
zeros(size(alignImagesHandles.leftBuffer)), zeros(size(alignImagesHandles.leftBuffer)));
%     figure (3);
%     image(temp);
%     figure(4)
%     image(temp1);

    %Display the Images
    DisplayComposite(hObject, alignImagesHandles);
end


% --------------------------------------------------------------------
function varargout = filenameTag_Callback(hObject, eventdata, alignImagesHandles)
file_name = str2num(get(alignImagesHandles.filenameTag,'String'));
% alignImageHandles.file_name = file_name;
% guidata(hObject, alignImageHandles);

% --------------------------------------------------------------------
function varargout = subnSaveTag_Callback(hObject, eventdata, alignImagesHandles)

global fileName1 filePath1 fileName2 filePath2
global rightshift leftshift upshift downshift

%Notify user that subtraction is prcessing
set(alignImagesHandles.statustxtTag,'String', 'Processing...');

%store the shifts as global variables
rightshift = alignImagesHandles.rshift;
leftshift  = alignImagesHandles.lshift;
upshift    = alignImagesHandles.ushift;
downshift  = alignImagesHandles.dshift;

for imgSet = 1:2
    % Set the appropriate info for the file...
    if imgSet == 1
        % open the 1st raw image file
```

المنارة للاستشارات
www.manaraa.com

```
        [fid, errorMsg] = fopen([filePath1 fileName1], 'rb');
        if fid == -1
            disp(errorMsg)
        end
    else
        % open the 2nd raw image file
        [fid, errorMsg] = fopen([filePath2 fileName2], 'rb');
        if fid == -1
            disp(errorMsg)
        end
    end
    % WinX binary data files begin with a 4100 byte header containing the necessary image
acquisition
    % parameters, such as image dimension (xDimension, yDimension) and the number of
images (numImages). The
    % format of the data is also encoded in the header. Here, header is read in unsigned
16-bit
    % integer format (of length 2050) to obtain the x-axis dimension "xDimension", y-axis
dimension
    % "yDimension", the number of images "numImages", and the image data type "dataType".
    header = fread(fid, 2050, 'uint16');
    xDimension = header(22);        % actual # of pixels on x axis
    dataType(imgSet) = header(55);       % experimental data type (0:float, 1: long int,
2: int, 3:short)
    yDimension = header(329);       % actual # of pixels on y axis
    numImages(imgSet) = header(724);     % number of images in data file
    if dataType(imgSet) == 3
        format = 'uint16';
        byteMultiplier = 2;
    elseif dataType(imgSet) == 2
        format = 'int';
        byteMultiplier = 4;
    elseif dataType(imgSet) == 1
        format = 'int';
        byteMultiplier = 4;
    else
        format = 'float';  % dataType = 0
        byteMultiplier = 4;
    end

    % Determine how far to jump into image stack (from beginning of file, after header)
    % Jump over header (4100 bytes) and all images before the one wanted.
    imageNumber = 1;

    fseek(fid,(4100 + xDimension * yDimension * byteMultiplier * (imageNumber - 1)),
'bof');

    for i = 1:numImages
        outImage = fread(fid, [xDimension, yDimension], format);    % read images one by
one
        outImage = outImage';   % rotate to be compatible with regionMap

        if (imgSet == 1)
            %Shift the Image (if needed)
            if rightshift ~= 0         %Shift Right
                outImage(:,[rightshift+1:xDimension])=outImage(:,[1:xDimension-
rightshift]);
```

www.manaraa.com

```
                outImage(:,[1:1+rightshift])=0;
            elseif leftshift ~= 0       %Shift Left
                outImage(:,[1:xDimension-
leftshift])=outImage(:,[1+leftshift:xDimension]);
                outImage(:,[xDimension-leftshift:xDimension])=0;
            end

            if downshift ~= 0        %Shift Down
                outImage([downshift+1:yDimension],:)=outImage([1:yDimension-
downshift],:);
                outImage([1:1+downshift],:)=0;
            elseif upshift ~= 0          %Shift Up
                outImage([1:yDimension-upshift],:)=outImage([1+upshift:yDimension],:);
                outImage([yDimension-upshift+1:yDimension],:)=0;
            end
        end

        %Store the image stacks in temporary arrays
        if imgSet == 1
            imgStack1(:, :, i)  = outImage;
        else
            imgStack2(:, :, i)  = outImage;
        end
    end
    fclose(fid);
end

%Determine number of images in stack to subtract
if numImages(1) ~= numImages(2)
    if numImages(2) > numImages(1)
        imgs = numImages(1);
        header(724) = imgs;    %Update the file header
    end
else
    imgs = numImages(2);
end

for j = 1:imgs
%Subtract imgStack1 (Background) from imgStack2
resultStack(:,:,j) = imgStack2(:,:,j)-imgStack1(:,:,j);
%Transpose Images Back for WinView format
resultStack(:,:,j) = resultStack(:,:,j)';
end

% figure(1)
% imshow((resultStack(:,:,1)),[]);

%Obtain the file name
file_name = get(alignImagesHandles.filenameTag,'String');

%No extenstions on filename
str_index = strfind(file_name,'.');
if ~isempty(str_index)
    file_name = file_name(1:str_index-1);
end

%Determine the data type and
```

المنارة للاستشارات
www.manaraa.com

```
%  choose most precise type
if dataType(1) ~= dataType(2)
    if dataType(2)>dataType(1)
        dataType = dataType(1);
        header(55) = dataType;   %Update the file header
    end
else
    dataType = dataType(2);
end

%Interpert the data type
if dataType == 3
    format = 'uint16';
elseif dataType == 2
    format = 'int';
elseif dataType == 1
    format = 'int';
else
    format = 'float';  % dataType = 0;
end

%Write Image back in WinView Format
fid = fopen([file_name '.SPE'],'wb');
fwrite(fid,header,'uint16');
fwrite(fid,resultStack, format);
fclose(fid);

%Notify user that subtraction is done
set(alignImagesHandles.statustxtTag,'String', 'Done.');

%end alignImages.m
```

المنارة للاستشارات

www.manaraa.com

intensifierCorrect.m

```
function [correction] = IntensifierCorrect(regionMap, filterPhos, bckGnd)
%Created by: Adam Norige

global filePath1 fileName1 filePath2 fileName2
global imagePhaseMatrix1 imagePhaseMatrix2

%Created by: Adam Norige
%Read in Both files and extract Cross Correlation Mean then
% calculate the
for imgSet = 1:2
    % Set the appropriate info for the file...
    if imgSet == 1
        % open the 1st raw image file
        [fid, errorMsg] = fopen([filePath1 fileName1], 'rb');
        if fid == -1
            disp(errorMsg)
        end
        imagePhaseMatrix = imagePhaseMatrix1;
    else
        % open the 2nd raw image file
        [fid, errorMsg] = fopen([filePath2 fileName2], 'rb');
        if fid == -1
            disp(errorMsg)
        end
        imagePhaseMatrix = imagePhaseMatrix2;
    end

    % WinX binary data files begin with a 4100 byte header containing the necessary image
acquisition
    % parameters, such as image dimension (xDimension, yDimension) and the number of
images (numImages). The
    % format of the data is also encoded in the header. Here, header is read in unsigned
16-bit
    % integer format (of length 2050) to obtain the x-axis dimension "xDimension", y-axis
dimension
    % "yDimension", the number of images "numImages", and the image data type "dataType".
    header = fread(fid, 2050, 'uint16');
    xDimension = header(22);        % actual # of pixels on x axis
    dataType = header(55);          % experimental data type (0:float, 1: long int, 2:
int, 3:short)
    yDimension = header(329);       % actual # of pixels on y axis
    numImages = header(724);        % number of images in data file
    if dataType == 3
        format = 'uint16';
    elseif dataType == 2
        format = 'int';
    elseif dataType == 1
        format = 'int';
    else
        format = 'float';  % dataType = 0
    end

    [numImagesToAnalyze, numPhaseDelaysToAnalyze] = size(imagePhaseMatrix);
```

www.manaraa.com

```
    imageIndex = 1;                % Start at first row of phase information matrix
    j=find(regionMap == 1);        % Coordinates of Analysis region (same for freq1 & 2)

    for i = 1:numImages
        phosIntImage = fread(fid, [xDimension, yDimension], format);    % read images one
by one
        phosIntImage = phosIntImage';   % rotate to be compatible with regionMap

        % correct for background
        phosIntImage = phosIntImage - bckGnd;   % subtract out background value

        if imageIndex <= numImagesToAnalyze
            if i == imagePhaseMatrix1(imageIndex, 1)
                if filterPhos > 0
                    phosIntImage = medfilt2(phosIntImage, [filterPhos filterPhos]);
                end
                phosIntensityArray(imageIndex) = mean(phosIntImage(j));    % store mean
value in 1st col
                imageIndex = imageIndex + 1;    % point to next row of matrix
            end
        end
    end

    freqMean(imgSet) = mean(phosIntensityArray);
    fclose(fid);
end
%Calculate the correction factor
correction = freqMean(1)/freqMean(2);

%End intensifierCorrect.m
```

المنارة للاستشارات

www.manaraa.com

MaskMap2.m

```
function [outputMap, outputMaskMap] = MaskMap(inputMap, R2Map, minR2, inputMaskMap,
minVal, maxVal)

% Function to take a raw "inputMap" and map to black (set pixel to zero) all pixels that
do not meet the
%   appropriate criteria (R2 too low, previously rejected pixel, not between minimum and
maximum).

% Input Parameters:
%   inputMap = unprocessed input map
%   R2Map = coefficient of determination map
%   minR2 = minimum acceptable R2 value (reject below minimum)
%   inputMaskMap = previously defined masking image (reject all masked pixels)
%   minVal = minimum acceptable value (reject below minimum)
%   maxVal = maximum acceptable value (reject above maximum)
%
% Output Parameters:
%   outputMap = processed output map
%   outputMaskMap = output masking image
%       Pixel value         Interpretation
%           0               Accept pixel, no errors
%           0.1             Reject pixel,
%           0.2             Reject pixel, a0, a1, or b1 = 0       (determined in
PhosMapCalculate2)
%           0.3             Reject pixel, XMap < 0                (determined in
phosMapDisplay2)
%           0.4             Reject pixel, Tau (theta) > tolerance (determined in
PO2MapCalculate2)
%           0.5             Reject pixel, < minVal
%           0.6             Reject pixel, > maxVal
%           0.7             Reject pixel,
%           0.8             Reject pixel, Tau (theta) < 0         (determined in
PO2MapCalculate2)
%           0.9             Reject pixel, image alignment         (determined in
phosMapDisplay2)
%           1.0             Reject pixel, R2 < minR2
%
% Created February 7, 2002 Ross D. Shonat, PhD
% Modified October 22, 2002 Adam Norige

outputMap = inputMap;        % copy the input map to the output map
outputMaskMap = inputMaskMap;

% reject pixels where the R2 is less than the specified value (R2Min) by setting map
pixel to zero
i = find(R2Map < minR2);
outputMap(i) = minVal;       %Set pixel to Black
outputMaskMap(i) = 1;

% reject map pixels for pixels already rejected in the mask map
i = find(inputMaskMap > 0);  %Accounts for:
outputMap(i) = minVal;       %Set pixel to Black

% reject map pixels where the value is less than the minimum allowed
```

www.manaraa.com

```
i = find(outputMap < minVal);
outputMap(i) = minVal;
outputMaskMap(i) = 0.5;
outputMap(1,1) = minVal;     %Make sure minVal appears in current map

% reject map pixels where the value is greater than the maximum allowed (set to maximum)
i = find(outputMap > maxVal);
outputMap(i) = maxVal;
outputMaskMap(i) = 0.6;
outputMap(1,2) = maxVal;     %Make sure maxVal appears in current map

% insure that outputMaskMap displays in range 0-1
outputMaskMap(1,1) = 0.0;
outputMaskMap(1,2) = 1.0;

%End MaskMap2.M
```

المنارة للاستشارات
www.manaraa.com

phosGrapDisplay2.m

```
function varargout = phosGraphDisplay2(varargin)
% PHOSGRAPHDISPLAY2 M-file for phosGraphDisplay2.fig
%
%      H = PHOSGRAPHDISPLAY2 returns the handle to a new PHOSGRAPHDISPLAY2 or the handle
%      given property value pairs. Unrecognized properties are passed via
%      varargin to phosGraphDisplay2_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      PHOSGRAPHDISPLAY2('CALLBACK') and PHOSGRAPHDISPLAY2('CALLBACK',hObject,...)

% Last Modified by Adam Norige (2003)

% global variables defined mostly in interface figure
global kQ tauO omega1 omega2  imagePhaseMatrix1 imagePhaseMatrix2 bckGnd thetaCorrection
modCorrection
global regionMap1 regionMap2 filterPhos1 filterPhos2
global fileName1 filePath1 fileName2 filePath2

if nargin == 0  % LAUNCH GUI

    phosGraphDisplayFig = openfig(mfilename,'reuse');

   % Use system color scheme for figure:
   set(phosGraphDisplayFig,'Color',get(0,'defaultUicontrolBackgroundColor'));

   % Generate a structure of handles to pass to callbacks, and store it.
   phosGraphDisplayHandles = guihandles(phosGraphDisplayFig);

    %Set the frequency label
    freq = round(omega1/(2*pi));
    txtlabel = strcat(num2str(freq), 'Hz');
    set(phosGraphDisplayHandles.freq11Tag, 'String', txtlabel);
    set(phosGraphDisplayHandles.freq12Tag, 'String',txtlabel);
    set(phosGraphDisplayHandles.freq13Tag, 'String',txtlabel);

    freq = round(omega2/(2*pi));
    txtlabel = strcat(num2str(freq), 'Hz Statistics');
    set(phosGraphDisplayHandles.freq21Tag, 'String', txtlabel);
    set(phosGraphDisplayHandles.freq22Tag, 'String',txtlabel);
    set(phosGraphDisplayHandles.freq23Tag, 'String',txtlabel);
    clear freq txtlabel;

    % ---> FREQUENCY 1
    % determine phosphorescence intensity and fitting parameters in the region specified
by "regionMap"
    [phosIntensityArray1, b11, a11, a01, theta1, mod1, R21, numROIPoints1, reject1] = ...
        PhosRegionCalculate2(regionMap1, filterPhos1, bckGnd, thetaCorrection,
modCorrection, 1);

    % ---> FREQUENCY 2
    % determine phosphorescence intensity and fitting parameters in the region specified
by "regionMap"
    [phosIntensityArray2, b12, a12, a02, theta2, mod2, R22, numROIPoints2, reject2] = ...
```

www.manaraa.com

```
        PhosRegionCalculate2(regionMap2, filterPhos2, bckGnd, thetaCorrection,
modCorrection, 2);

    % store fitting parameters into phosphorescence graph display handle structure
    phosGraphDisplayHandles.phosIntensityArray1 = phosIntensityArray1;
   phosGraphDisplayHandles.phosIntensityArray2 = phosIntensityArray2;

    % REMOVE ZERO-PHASE SIGNAL
    R = (b12*(omega1))/(b11*(omega2));  %a1-frequency ratio
    X = ((R*a11)-a12)/(R-1);            %Determine a1 contamination

    if (X < 0)   %Dont allow a negative X
        X = 0;
    end


    %Scale X
    %X=X*0.4;


    % Frequency 1 (freq1) Reconstruction
    a0Corr = a02 - X;   %x=S(r) or x=a0Cont
    a1Corr = a12 - X;   %x=S(r) or x=a1Cont
    b1Corr = b12;       %b11=b1Corr=b1Cont

    rejectCorr = 0;
    rejectCont = 0;
    %%  CALCULATIONS FOR CORRECTED SIGNAL -- This is what we want
    % check for zeroes in a0 and a1 and reject if true
    if a0Corr == 0.0          % reject fits with a0 = 0
        a0Corr = eps;         % make a0 a very, very small number
        rejectCorr = 1;
    end
    if a1Corr == 0.0          % reject fits with a1 = 0
        a1Corr = eps;         % make a1 a very, very small number
        rejectCorr = 1;
    end

    % Calculate phase shift from fit data
    thetaCorr = atan(b1Corr/a1Corr);

    % Calculate modulation from fit data
    modCorr = sqrt((a1Corr*a1Corr)+(b1Corr*b1Corr))/(a0Corr);

    %%  CALCULATIONS FOR CONTAMINATION -- Just for fun
    % check for zeroes in a0 and a1 and reject if true
    if X == 0.0          % check if X = 0
        X = eps;         % make a0 a very, very small number
        rejectCont = 0.2;
    end

    % Calculate phase shift from fit data
    thetaCont = atan(0/X);
    thetaCont = thetaCont - thetaCorrection;

    % Calculate modulation from fit data
    modCont = sqrt(X*X);
```

المنارة للاستشارات
www.manaraa.com

```
    modCont = modCont / (X * modCorrection);


    % store handle structure for phosphorescence graph display figure
    guidata(phosGraphDisplayFig, phosGraphDisplayHandles);

%     xlabel('Phase Shift (Degrees)', 'HorizontalAlignment', 'center');
%     ylabel('Phosphorescence Intensity (AU)', 'HorizontalAlignment', 'center');

    hold on;
    axes(phosGraphDisplayHandles.IntensityGraphTag);
    xlim([0 360]);
    set(phosGraphDisplayHandles.IntensityGraphTag,'XTick',0:60:360);
    set(phosGraphDisplayHandles.IntensityGraphTag,'XGrid','on');
    set(phosGraphDisplayHandles.IntensityGraphTag,'Color','w');

    % plot data to graph
    phosDelayX1 = imagePhaseMatrix1(:, 2);        % phase delay array in radians
    phosDelayX2 = imagePhaseMatrix2(:, 2);        % phase delay array in radians
    phosDelayXDegree1 = phosDelayX1 * 180.0 / pi; % covert to degrees for plotting
    phosDelayXDegree2 = phosDelayX2 * 180.0 / pi; % covert to degrees for plotting

    phosIntensityY1 = phosIntensityArray1(:, 1);
    phosIntensityY2 = phosIntensityArray2(:, 1);

    fitX = 0:pi/100.0:2*pi;
    fitXDegree = fitX * 180.0 / pi;               % convert to degrees for plotting

    %Develop fitted Curves
    fitYCorr = a0Corr + a1Corr * cos(fitX) + b1Corr * sin(fitX);
    fitYCont = X + (X * cos(fitX));
    fitY1 = a01 + a11 * cos(fitX) + b11 * sin(fitX);
    fitY2 = a02 + a12 * cos(fitX) + b12 * sin(fitX);


plot(phosDelayXDegree1,phosIntensityY1,'ok',fitXDegree,fitY1,'k','MarkerFaceColor','k');

plot(phosDelayXDegree2,phosIntensityY2,'ob',fitXDegree,fitY2,'b','MarkerFaceColor','b');
%     plot(phosDelayXDegree1,phosIntensityY1,'ok','MarkerFaceColor','k');
%     plot(phosDelayXDegree2,phosIntensityY2,'ob','MarkerFaceColor','b');
    plot(fitXDegree,fitYCorr,'r');
    plot(fitXDegree,fitYCont,'g');

    %Legend Titles
    freq = round(omega1/(2*pi));
    tmp1 = strcat(num2str(freq), 'Hz');
    freq = round(omega2/(2*pi));
    tmp2 = strcat(num2str(freq), 'Hz');
    legend([tmp1 ' Raw'], [tmp1 ' Adj'], [tmp2 ' Raw'], [tmp2 ' Adj'], 'Corrected',
'Contamination');

    for i = 1:length(phosDelayX1)
        errorBarX = [phosDelayXDegree1(i),phosDelayXDegree1(i)];
        errorBarY = [phosIntensityY1(i)- phosIntensityArray1(i,2), phosIntensityY1(i) +
phosIntensityArray1(i,2)];
        plot(errorBarX, errorBarY, '-k');
    end
```

www.manaraa.com

```
    for i = 1:length(phosDelayX2)
        errorBarX = [phosDelayXDegree2(i),phosDelayXDegree2(i)];
        errorBarY = [phosIntensityY2(i) - phosIntensityArray2(i,2),
phosIntensityY2(i)+phosIntensityArray2(i,2)];
        plot(errorBarX, errorBarY, '-b');
    end

    % Calculate phase, modulation, lifetime, and PO2 from fitting parameters
    thetaDegree1 = theta1 * 180.0 / pi;
    tauTheta1 = tan(theta1) * 1000000.0 / omega1;
    PO2Theta1 = 1/tauTheta1;
    PO2Theta1 = 1000000.0 * (1 / kQ) * (PO2Theta1 - (1/tau0) );

    tauMod1 = 1000000.0 * sqrt((1.0/(mod1*mod1)-1.0) / (omega1*omega1));
    PO2Mod1 = 1/tauMod1;
    PO2Mod1 = 1000000.0 * (1 / kQ) * (PO2Mod1 - (1/tau0));

    % put fitting parameters into box
    set(phosGraphDisplayHandles.b1Freq1Tag,'String',num2str(b11,'%8.3f'));
    set(phosGraphDisplayHandles.a1Freq1Tag,'String',num2str(a11,'%8.3f'));
    set(phosGraphDisplayHandles.a0Freq1Tag,'String',num2str(a01,'%8.3f'));
    set(phosGraphDisplayHandles.R2Freq1Tag,'String',num2str(R21,'%6.4f'));

set(phosGraphDisplayHandles.numROIPointsFreq1Tag,'String',num2str(numROIPoints1,'%5.0f'))
;
    set(phosGraphDisplayHandles.rejectFreq1Tag,'String',num2str(reject1,'%1.0f'));
    set(phosGraphDisplayHandles.thetaFreq1Tag,'String',num2str(thetaDegree1,'%8.4f'));

set(phosGraphDisplayHandles.lifetimeThetaFreq1Tag,'String',num2str(tauTheta1,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ThetaFreq1Tag,'String',num2str(PO2Theta1,'%6.2f'));
    set(phosGraphDisplayHandles.modFreq1Tag,'String',num2str(mod1,'%6.4f'));
    set(phosGraphDisplayHandles.lifetimeModFreq1Tag,'String',num2str(tauMod1,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ModFreq1Tag,'String',num2str(PO2Mod1,'%6.2f'));

    % Calculate phase, modulation, lifetime, and PO2 from fitting parameters
    thetaDegree2 = theta2* 180.0 / pi;
    tauTheta2 = tan(theta2) * 1000000.0 / omega2;
    PO2Theta2 = 1/tauTheta2;
    PO2Theta2 = 1000000.0 * (1 / kQ) * (PO2Theta2 - (1/tau0) );

    tauMod2 = 1000000.0 * sqrt((1.0/(mod2*mod2)-1.0) / (omega2*omega2));
    PO2Mod2 = 1/tauMod2;
    PO2Mod2 = 1000000.0 * (1 / kQ) * (PO2Mod2 - (1/tau0));

    % put fitting parameters into box
    set(phosGraphDisplayHandles.b1Freq2Tag,'String',num2str(b12,'%8.3f'));
    set(phosGraphDisplayHandles.a1Freq2Tag,'String',num2str(a12,'%8.3f'));
    set(phosGraphDisplayHandles.a0Freq2Tag,'String',num2str(a02,'%8.3f'));
    set(phosGraphDisplayHandles.R2Freq2Tag,'String',num2str(R22,'%6.4f'));

set(phosGraphDisplayHandles.numROIPointsFreq2Tag,'String',num2str(numROIPoints2,'%5.0f'))
;
    set(phosGraphDisplayHandles.rejectFreq2Tag,'String',num2str(reject2,'%1.0f'));
    set(phosGraphDisplayHandles.thetaFreq2Tag,'String',num2str(thetaDegree2,'%8.4f'));

set(phosGraphDisplayHandles.lifetimeThetaFreq2Tag,'String',num2str(tauTheta2,'%6.2f'));
```

www.manaraa.com

```
    set(phosGraphDisplayHandles.PO2ThetaFreq2Tag,'String',num2str(PO2Theta2,'%6.2f'));
    set(phosGraphDisplayHandles.modFreq2Tag,'String',num2str(mod2,'%6.4f'));
    set(phosGraphDisplayHandles.lifetimeModFreq2Tag,'String',num2str(tauMod2,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ModFreq2Tag,'String',num2str(PO2Mod2,'%6.2f'));

    % Calculate phase, modulation, lifetime, and PO2 from fitting parameters
    thetaDegreeCorr = thetaCorr * 180.0 / pi;
    tauThetaCorr = tan(thetaCorr) * 1000000.0 / omega2;
    PO2ThetaCorr = 1/tauThetaCorr;
    PO2ThetaCorr = 1000000.0 * (1 / kQ) * (PO2ThetaCorr - (1/tau0) );

    tauModCorr = 1000000.0 * sqrt((1.0/(modCorr*modCorr)-1.0) / (omega2*omega2));
    PO2ModCorr = 1/tauModCorr;
    PO2ModCorr = 1000000.0 * (1 / kQ) * (PO2ModCorr - (1/tau0));

    % put fitting parameters into box
    set(phosGraphDisplayHandles.b1CorrTag,'String',num2str(b1Corr,'%8.3f'));
    set(phosGraphDisplayHandles.a1CorrTag,'String',num2str(a1Corr,'%8.3f'));
    set(phosGraphDisplayHandles.a0CorrTag,'String',num2str(a0Corr,'%8.3f'));
    set(phosGraphDisplayHandles.rejectCorrTag,'String',num2str(rejectCorr,'%1.0f'));
    set(phosGraphDisplayHandles.thetaCorrTag,'String',num2str(thetaDegreeCorr,'%8.4f'));

set(phosGraphDisplayHandles.lifetimeThetaCorrTag,'String',num2str(tauThetaCorr,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ThetaCorrTag,'String',num2str(PO2ThetaCorr,'%6.2f'));
    set(phosGraphDisplayHandles.modCorrTag,'String',num2str(modCorr,'%6.4f'));
    set(phosGraphDisplayHandles.lifetimeModCorrTag,'String',num2str(tauModCorr,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ModCorrTag,'String',num2str(PO2ModCorr,'%6.2f'));

    % Calculate phase, modulation, lifetime, and PO2 from fitting parameters
    thetaDegreeCont = thetaCont * 180.0 / pi;
    tauThetaCont = tan(thetaCont) * 1000000.0 / omega2;
    if tauThetaCont == 0
        tauThetaCont = eps;  %Really small number
    end
    PO2ThetaCont = 1/tauThetaCont;
    PO2ThetaCont = 1000000.0 * (1 / kQ) * (PO2ThetaCont - (1/tau0) );

    tauModCont = 1000000.0 * sqrt((1.0/(modCont*modCont)-1.0) / (omega2*omega2));
    if tauModCont == 0
        tauModCont = eps;  %Really small number
    end
    PO2ModCont = 1/tauModCont;
    PO2ModCont = 1000000.0 * (1 / kQ) * (PO2ModCont - (1/tau0));

    % put fitting parameters into box
    set(phosGraphDisplayHandles.b1ContTag,'String',num2str(0.0));
    set(phosGraphDisplayHandles.a1ContTag,'String',num2str(X,'%8.3f'));
    set(phosGraphDisplayHandles.a0ContTag,'String',num2str(X,'%8.3f'));
    set(phosGraphDisplayHandles.rejectContTag,'String',num2str(rejectCont,'%1.0f'));
    set(phosGraphDisplayHandles.thetaContTag,'String',num2str(thetaDegreeCont,'%8.4f'));

set(phosGraphDisplayHandles.lifetimeThetaContTag,'String',num2str(tauThetaCont,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ThetaContTag,'String',num2str(PO2ThetaCont,'%6.2f'));
    set(phosGraphDisplayHandles.modContTag,'String',num2str(modCont,'%6.4f'));
    set(phosGraphDisplayHandles.lifetimeModContTag,'String',num2str(tauModCont,'%6.2f'));
    set(phosGraphDisplayHandles.PO2ModContTag,'String',num2str(PO2ModCont,'%6.2f'));
```

www.manaraa.com

```
    % put vertical line to indicate phase shift
    %axes(phosGraphDisplayHandles.IntensityGraphTag);
    yLimits = get(phosGraphDisplayHandles.IntensityGraphTag,'YLim');
    xLimits = get(phosGraphDisplayHandles.IntensityGraphTag,'XLim');
    line([thetaDegreeCorr; thetaDegreeCorr], [yLimits(1); yLimits(2)],'color','r');
    line([thetaDegree1; thetaDegree1], [yLimits(1);yLimits(2)], 'color', 'k');
    line([thetaDegree2; thetaDegree2], [yLimits(1);yLimits(2)], 'color', 'b');
    line([xLimits(1); xLimits(2)], [0;0], 'color', 'b');
    hold off;

    set(phosGraphDisplayHandles.kQTag,'String',num2str(kQ, '%4.1f'));
    set(phosGraphDisplayHandles.tauOTag,'String',num2str(tauO, '%4.1f'));
    set(phosGraphDisplayHandles.omegaTag,'String',num2str(omega1, '%6.1f'));

   if nargout > 0
      varargout{1} = phosGraphDisplayFig;
   end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

   try
      if (nargout)
         [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
      else
         feval(varargin{:}); % FEVAL switchyard
      end
   catch
      disp(lasterr);
   end

end

function saveTag_Callback(hObject, eventdata, handles)
% hObject -  handle to figure (interfaceHandles.retina2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data)

%end phosGraphDisplay2.m
```

www.manaraa.com

phosIntensityDisplay2.m

```
function varargout = phosIntensityDisplay2(varargin)
% PHOSINTENSITYDISPLAY2 Application M-file for phosIntensityDisplay.fig
%    FIG = PHOSINTENSITYDISPLAY2 launch phosIntensityDisplay GUI.
%    PHOSINTENSITYDISPLAY2('callback_name', ...) invoke the named callback.

% Last Modified by Adam Norige (2003)

global fileName1 filePath1 outImage1
global fileName2 filePath2 outImage2
global xDimension yDimension
global rightshift leftshift upshift downshift

if nargin == 0  % LAUNCH GUI

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% IMAGE 1 %%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % read first image to establish masking terms
    imageNumber = 1;        % start with first image in stack
    [outImage1, numImages, minPixelIntensity1, maxPixelIntensity1] =
GetWinXIntensityImage(filePath1, fileName1, 1);

    %Shift the Image (if needed)
    if rightshift ~= 0          %Shift Right
        outImage1(:,[rightshift+1:xDimension])=outImage1(:,[1:xDimension-rightshift]);
        outImage1(:,[1:1+rightshift])=0;
    elseif leftshift ~= 0       %Shift Left
        outImage1(:,[1:xDimension-leftshift])=outImage1(:,[1+leftshift:xDimension]);
        outImage1(:,[xDimension-leftshift:xDimension])=0;
    end

    if downshift ~= 0        %Shift Down
        outImage1([downshift+1:yDimension],:)=outImage1([1:yDimension-downshift],:);
        outImage1([1:1+downshift],:)=0;
    elseif upshift ~= 0          %Shift Up
        outImage1([1:yDimension-upshift],:)=outImage1([1+upshift:yDimension],:);
        outImage1([yDimension-upshift+1:yDimension],:)=0;
    end

    %open the phosphorescence intensity display window
   phosIntensityDisplayFig2 = openfig(mfilename,'reuse');
    set(phosIntensityDisplayFig2,'Color',get(0,'defaultUicontrolBackgroundColor'));

   % Generate a structure of handles to pass to callbacks, and store it.
   phosIntensityDisplayHandles = guihandles(phosIntensityDisplayFig2);

    % store parameters into intensity display figure handle structure
    phosIntensityDisplayHandles.imageNumber1 = imageNumber;
    phosIntensityDisplayHandles.numImages1 = numImages;

    % open the intensity figure window and display image and image statistics
    set(phosIntensityDisplayHandles.imageNumTag1,'String',num2str(imageNumber,'%3.0f'));
```

www.manaraa.com

```
set(phosIntensityDisplayHandles.maxPixelIntTag1,'String',num2Str(maxPixelIntensity1,'%5.0
f'));

set(phosIntensityDisplayHandles.minPixelIntTag1,'String',num2Str(minPixelIntensity1,'%5.0
f'));

    % display the image
    axes(phosIntensityDisplayHandles.intensityImageTag1);
    imshow(outImage1,[minPixelIntensity1 maxPixelIntensity1]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% IMAGE 2 %%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % read first image to establish masking terms
    imageNumber = 1;        % start with first image in stack
    [outImage2, numImages, minPixelIntensity2, maxPixelIntensity2] =
GetWinXIntensityImage(filePath2, fileName2, 1);

    set(phosIntensityDisplayHandles.minMaskIntTag,'String','Min');
    phosIntensityDisplayHandles.minMaskIntensity = -1;

    set(phosIntensityDisplayHandles.maxMaskIntTag,'String','Max');
    phosIntensityDisplayHandles.maxMaskIntensity = -1;

    % store parameters into intensity display figure handle structure
    phosIntensityDisplayHandles.imageNumber2 = imageNumber;
    phosIntensityDisplayHandles.numImages2 = numImages;

    % open the intensity figure window and display image and image statistics
    set(phosIntensityDisplayHandles.imageNumTag2,'String',num2str(imageNumber,'%3.0f'));

set(phosIntensityDisplayHandles.maxPixelIntTag2,'String',num2Str(maxPixelIntensity2,'%5.0
f'));

set(phosIntensityDisplayHandles.minPixelIntTag2,'String',num2Str(minPixelIntensity2,'%5.0
f'));

    % display the image
    axes(phosIntensityDisplayHandles.intensityImageTag2);
    imshow(outImage2,[minPixelIntensity2 maxPixelIntensity2]);

    % load values into file output section of window
    set(phosIntensityDisplayHandles.outputStyleTag,'String',strvcat('No Intensity
Bar',...
            'Intensity Bar w/o Ticks','Intensity Bar with Ticks'));
    set(phosIntensityDisplayHandles.outputStyleTag,'Value',2);
    set(phosIntensityDisplayHandles.fileNameTag,'String','outIntImage.tif');

    % store handle structure for phosphorescence intensity display figure
    guidata(phosIntensityDisplayFig2, phosIntensityDisplayHandles);

    if nargout > 0
      varargout{1} = phosIntensityDisplayFig;
   end
```

www.manaraa.com

```
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

   try
      [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
   catch
      disp(lasterr);
   end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- Image Set Number 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function leftBeginningTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = 1;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function leftFiveTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.imageNumber1 - 5;
if phosIntensityDisplayHandles.imageNumber1 < 1
    phosIntensityDisplayHandles.imageNumber1 = 1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function leftOneTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.imageNumber1 - 1;
if phosIntensityDisplayHandles.imageNumber1 < 1
    phosIntensityDisplayHandles.imageNumber1 = 1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function rightOneTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.imageNumber1 + 1;
```

المنارة للاستشارات
www.manaraa.com

```
if phosIntensityDisplayHandles.imageNumber1 > phosIntensityDisplayHandles.numImages1
    phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.numImages1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function rightFiveTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.imageNumber1 + 5;
if phosIntensityDisplayHandles.imageNumber1 > phosIntensityDisplayHandles.numImages1
    phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.numImages1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function rightEndTag1_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber1 = phosIntensityDisplayHandles.numImages1;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- Image Set Number 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function leftBeginningTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = 1;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2)

%---------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%---------------------------------------------------------------------------
function leftFiveTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.imageNumber2 - 5;
if phosIntensityDisplayHandles.imageNumber2 < 1
    phosIntensityDisplayHandles.imageNumber2 = 1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);
```

المنارة للاستشارات

www.manaraa.com

```
%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function leftOneTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.imageNumber2 - 1;
if phosIntensityDisplayHandles.imageNumber2 < 1
    phosIntensityDisplayHandles.imageNumber2 = 1;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function rightOneTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.imageNumber2 + 1;
if phosIntensityDisplayHandles.imageNumber2 > phosIntensityDisplayHandles.numImages2
    phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.numImages2;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function rightFiveTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.imageNumber2 + 5;
if phosIntensityDisplayHandles.imageNumber2 > phosIntensityDisplayHandles.numImages2
    phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.numImages2;
end
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function rightEndTag2_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
phosIntensityDisplayHandles.imageNumber2 = phosIntensityDisplayHandles.numImages2;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%-----------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS -- General
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function graphRegionTag_Callback(hObject, eventdata, handles)
```

www.manaraa.com

```
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data

% pass control to the phosphorescence graphing display window
phosGraphDisplay2;

%-------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-------------------------------------------------------------------------
function defineROITag_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global regionMap1 xPolyCoordinates1 yPolyCoordinates1
global regionMap2 xPolyCoordinates2 yPolyCoordinates2

displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);
[regionMap2, xPolyCoordinates2, yPolyCoordinates2] = roipoly;
line(xPolyCoordinates2, yPolyCoordinates2);

xPolyCoordinates1 = xPolyCoordinates2;
yPolyCoordinates1 = yPolyCoordinates2;
regionMap1 = regionMap2;
axes(phosIntensityDisplayHandles.intensityImageTag1);
line(xPolyCoordinates1, yPolyCoordinates1);

%-------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-------------------------------------------------------------------------
function recallTag_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global regionMap1 xPolyCoordinates1 yPolyCoordinates1
global regionMap2 xPolyCoordinates2 yPolyCoordinates2

displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);
line(xPolyCoordinates1, yPolyCoordinates1);
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);
line(xPolyCoordinates2, yPolyCoordinates2);
%-------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-------------------------------------------------------------------------
function imageSaveTag_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global outImage1

% reformat the "double" precision maps into 0-255 format needed for TIFF output.
saveMap = ((outImage1 - phosIntensityDisplayHandles.minMaskIntensity)/...
          (phosIntensityDisplayHandles.maxMaskIntensity -
phosIntensityDisplayHandles.minMaskIntensity)) ...
          * 255.0;
```

www.manaraa.com

```
optionValue = get(phosIntensityDisplayHandles.outputStyleTag,'Value');
% create the intensity bar (if selected)
if (optionValue == 2) | (optionValue == 3)
    [xDim yDim] = size(outImage);
    verticalBar = ones(xDim, round(yDim/15.0));
    verticalBar(1,1:round(yDim/60.0)) = 0.0;
    verticalBar(1,3*round(yDim/60.0):4*round(yDim/60)) = 0.0;
    for i = 1:xDim
        verticalBar(i,:) = ((xDim - (i-1)) / xDim) * 255.0;
        if (optionValue == 3) & mod(i,round(xDim/10)) == 0
            verticalBar(i,1:round(yDim/60.0)) = 0.0;
            verticalBar(i,3*round(yDim/60.0):4*round(yDim/60)) = 0.0;
        end
    end
    verticalBar(xDim,1:round(yDim/60.0)) = 0.0;
    verticalBar(xDim,3*round(yDim/60.0):4*round(yDim/60)) = 0.0;
    saveMap = [saveMap verticalBar];
end

saveFileName = get(phosIntensityDisplayHandles.fileNameTag1,'String');   % get file name
to save
imwrite(saveMap, colormap, saveFileName, 'tif');


%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function minMaskIntTag_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
if (strcmp((lower(get(phosIntensityDisplayHandles.minMaskIntTag,'String'))),'min'))
        minMask = -1;
        tempout = 'Min';
else
    minMask = str2num(get(phosIntensityDisplayHandles.minMaskIntTag,'String'));
    if minMask >= 99999
        minMask = 99999;
    elseif minMask <= 1
        minMask = 1;
    end
    tempout = num2str(minMask,'%5.0f');
end
set(phosIntensityDisplayHandles.minMaskIntTag,'String',tempout);
phosIntensityDisplayHandles.minMaskIntensity = minMask;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function maxMaskIntTag_Callback(hObject, eventdata, phosIntensityDisplayHandles)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
if (strcmp((lower(get(phosIntensityDisplayHandles.maxMaskIntTag,'String'))),'max'))
    maxMask = -1;
```

www.manaraa.com

```
    tempout = 'Max';
else
    maxMask = str2num(get(phosIntensityDisplayHandles.maxMaskIntTag,'String'));
    if maxMask >= 99999
        maxMask = 99999;
    elseif maxMask <= 1
        maxMask = 1;
    end
    tempout = num2str(maxMask,'%5.0f');
end
set(phosIntensityDisplayHandles.maxMaskIntTag,'String',tempout);
phosIntensityDisplayHandles.maxMaskIntensity = maxMask;
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,1);
displayWinXIntensityImage(hObject, phosIntensityDisplayHandles,2);

%---------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   FUNCTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function displayWinXIntensityImage(hObject, phosIntensityDisplayHandles, imageset)
% hObject -  handle to figure (interfaceHandles.phosIntensityDisplayTag)
% interfaceHandles - structure with interfaceHandles and user data
% imageset - the image set selected for processing (1 or 2)
global fileName1 filePath1 outImage1
global fileName2 filePath2 outImage2
global rightshift leftshift upshift downshift
global xDimension yDimension

[outImage1, numImages1, minPixelIntensity1, maxPixelIntensity1] =
GetWinXIntensityImage(filePath1, fileName1, ...
    phosIntensityDisplayHandles.imageNumber1);

%Shift the Image (if needed)
if rightshift ~= 0          %Shift Right
    outImage1(:,[rightshift+1:xDimension])=outImage1(:,[1:xDimension-rightshift]);
    outImage1(:,[1:1+rightshift])=0;
elseif leftshift ~= 0       %Shift Left
    outImage1(:,[1:xDimension-leftshift])=outImage1(:,[1+leftshift:xDimension]);
    outImage1(:,[xDimension-leftshift:xDimension])=0;
end

if downshift ~= 0        %Shift Down
    outImage1([downshift+1:yDimension],:)=outImage1([1:yDimension-downshift],:);
    outImage1([1:1+downshift],:)=0;
elseif upshift ~= 0          %Shift Up
    outImage1([1:yDimension-upshift],:)=outImage1([1+upshift:yDimension],:);
    outImage1([yDimension-upshift+1:yDimension],:)=0;
end

[outImage2, numImages2, minPixelIntensity2, maxPixelIntensity2] =
GetWinXIntensityImage(filePath2, fileName2, ...
    phosIntensityDisplayHandles.imageNumber2);

if (imageset == 1)
    % open the intensity figure window and display image and image statistics
```

www.manaraa.com

```
set(phosIntensityDisplayHandles.imageNumTag1,'String',num2str(phosIntensityDisplayHandles
.imageNumber1,'%3.0f'));

set(phosIntensityDisplayHandles.maxPixelIntTag1,'String',num2Str(maxPixelIntensity1,'%5.0
f'));

set(phosIntensityDisplayHandles.minPixelIntTag1,'String',num2Str(minPixelIntensity1,'%5.0
f'));

    % display the image
    axes(phosIntensityDisplayHandles.intensityImageTag1);

    if (phosIntensityDisplayHandles.minMaskIntensity == -1 &
phosIntensityDisplayHandles.maxMaskIntensity == -1)
        imshow(outImage1,[minPixelIntensity1 maxPixelIntensity1]);
    elseif (phosIntensityDisplayHandles.minMaskIntensity == -1 &
phosIntensityDisplayHandles.maxMaskIntensity ~= -1)
        imshow(outImage1,[minPixelIntensity1
phosIntensityDisplayHandles.maxMaskIntensity]);
    elseif (phosIntensityDisplayHandles.minMaskIntensity ~= -1 &
phosIntensityDisplayHandles.maxMaskIntensity == -1)
        imshow(outImage1,[phosIntensityDisplayHandles.minMaskIntensity
maxPixelIntensity1]);
    else
        imshow(outImage1,[phosIntensityDisplayHandles.minMaskIntensity
phosIntensityDisplayHandles.maxMaskIntensity]);
    end

elseif (imageset == 2)

    % open the intensity figure window and display image and image statistics

set(phosIntensityDisplayHandles.imageNumTag2,'String',num2str(phosIntensityDisplayHandles
.imageNumber2,'%3.0f'));

set(phosIntensityDisplayHandles.maxPixelIntTag2,'String',num2Str(maxPixelIntensity2,'%5.0
f'));

set(phosIntensityDisplayHandles.minPixelIntTag2,'String',num2Str(minPixelIntensity2,'%5.0
f'));

    % display the image
    axes(phosIntensityDisplayHandles.intensityImageTag2);

    if (phosIntensityDisplayHandles.minMaskIntensity == -1 &
phosIntensityDisplayHandles.maxMaskIntensity == -1)
        imshow(outImage2,[minPixelIntensity2 maxPixelIntensity2]);
    elseif (phosIntensityDisplayHandles.minMaskIntensity == -1 &
phosIntensityDisplayHandles.maxMaskIntensity ~= -1)
        imshow(outImage2,[minPixelIntensity2
phosIntensityDisplayHandles.maxMaskIntensity]);
    elseif (phosIntensityDisplayHandles.minMaskIntensity ~= -1 &
phosIntensityDisplayHandles.maxMaskIntensity == -1)
        imshow(outImage2,[phosIntensityDisplayHandles.minMaskIntensity
maxPixelIntensity2]);
    else
```

www.manaraa.com

```
        imshow(outImage2,[phosIntensityDisplayHandles.minMaskIntensity
phosIntensityDisplayHandles.maxMaskIntensity]);
    end
end

guidata(hObject, phosIntensityDisplayHandles);


%end phosIntensityDisplay2.m
```

www.manaraa.com

phosMapCalculate.m

```
function [thetaMap, modMap, a0Map, a1Map, b1Map, R2Map, rejectMap] =
PhosMapCalculate(filterPhos,...
    bckGnd, thetaCorrection, modCorrection, imgset)

% PHOSMAPCALCULATE m-file is intended for use with
% PhosMapDisplay.m  Under normal use PhosMapCalculate should
% only be called from PhosMapDisplay2.fig
%
% Reference: Lakowicz et al (1992). Fluorescence lifetime imaging. Anal. Biochem.
202:316-330.
% Define the following equation: I(thetaD) = a0 + a1*cos(thetaD) + b1*sin(thetaD)
%   where I(thetaD) = phosphorescence intensity at a pixel for a given phase shift
"thetaD"
%          a0, a1, and b1 = unknowns to solve by least squares
% Solve for a0, a1, and b1 by solving the following equation: Ax=B
%   where A = 3 x 3 matrix
%         B = 3 x numPixels matrix
%         x = 3 x numPixels maxtrix = [b1, a1, a0] x numPixels
% Then
%   phosphorescence phase = theta = atan(b1/a1)
%   modulation amplitude = mod = sqrt(a1^2+b1^2)/a0
%   DC signal amplitude = a0
%
% Input Parameters:
%   filepath = path where the raw intensity image file is stored
%   filename = name of the raw intensity image file
%   imagePhaseMatrix = matrix holding the image numbers and phase delays to analyze
%       Column 1: imageNumbers       Column 2: phaseDelay (radians) (max rows: numImages)
%   filterPhos = spatial filtering of phosphorescence intensity image (filtering > 1)
%   bckGnd = camera background value (AU)
%   thetaCorrection = correction term for instrumentation phase shift
%   modCorrection = correction term for instrumentation modulation
%
% Output Parameters:
%   thetaMap = phosphorescence phase shift map (in radians)
%   modMap = modulated amplitude map (unitless)
%   DCMap = steady-state (or DC) component of the phosphorescence (units of intensity)
%   a1Map = map of a1 values
%   b1Map = map of b1 values
%   R2Map = map of correlation coefficient (0-1)
%   rejectMap = map of rejected pixels
%       Pixel value         Interpretation
%           0               Accept pixel, no errors
%           0.7             Reject pixel, a0, a1, or b1 = 0
%
% PHOSMAPCALCULATE2 (Created by Adam Norige) was developed from
%                  CALCULATE (Created by Ross Shonat)
% Created by Adam Norige 22-Sep-2003
% Last Modified by GUIDE v2.5 22-Oct-2003 13:36:57

global rightshift leftshift upshift downshift
global xDimension yDimension
global omega1 omega2
```

www.manaraa.com

```
global filePath1 fileName1 filePath2 fileName2
global imagePhaseMatrix1 imagePhaseMatrix2


%Select the correct frequency
if imgset == 1
    omega = omega1;
    imagePhaseMatrix = imagePhaseMatrix1;
    % open the 1st raw image file
    [fid, errorMsg] = fopen([filePath1 fileName1], 'rb');
    if fid == -1
        disp(errorMsg)
    end
elseif imgset == 2
    %Obtain the intensifer correction value
    y = IntensifierCorrect(ones([yDimension xDimension]), filterPhos, bckGnd);
    omega = omega2;
    imagePhaseMatrix = imagePhaseMatrix2;
    % open the 1st raw image file
    [fid errorMsg] = fopen([filePath2 fileName2], 'rb');
    if fid == -1
        disp(errorMsg)
    end
end


% WinX binary data files begin with a 4100 byte header containing the necessary image
acquisition
% parameters, such as image dimension (xDimension, yDimension) and the number of images
(numImages). The
% format of the data is also encoded in the header. Here, header is read in unsigned 16-
bit
% integer format (of length 2050) to obtain the x-axis dimension "xDimension", y-axis
dimension
% "yDimension", the number of images "numImages", and the image data type "dataType".
header = fread(fid, 2050, 'uint16');
xDimension = header(22);        % actual # of pixels on x axis
dataType = header(55);          % experimental data type (0:float, 1: long int, 2: int,
3:short)
yDimension = header(329);       % actual # of pixels on y axis
numImages = header(724);        % number of images in data file
if dataType == 3
    format = 'uint16';
elseif dataType == 2
    format = 'int';
elseif dataType == 1
    format = 'int';
else
    format = 'float';  % dataType = 0
end

% Perform some preliminary calculations
rejectMap = zeros(xDimension,yDimension);   % begin by assuming no rejection of pixels (=
0)
[numImagesToAnalyze, numPhaseDelaysToAnalyze] = size(imagePhaseMatrix);
phaseDelayArray = imagePhaseMatrix(:,2);
sinThetaDArray = sin(phaseDelayArray);
```

www.manaraa.com

```
cosThetaDArray = cos(phaseDelayArray);

% Build the A matrix (3 X 3)
A = [ sum(sinThetaDArray.*sinThetaDArray) sum(sinThetaDArray.*cosThetaDArray)
sum(sinThetaDArray);
      sum(sinThetaDArray.*cosThetaDArray) sum(cosThetaDArray.*cosThetaDArray)
sum(cosThetaDArray);
      sum(sinThetaDArray)                 sum(cosThetaDArray)
numImagesToAnalyze];

% Build the B matrix (3 X 1) by reading specified raw intensity images
numImagePixels = xDimension * yDimension;          % calculate total number of pixels
B = zeros(3, numImagePixels);                      % initialize to zero
phosIntImageSquared = zeros(1, numImagePixels);    % holds the value of sum(intensity^2)
imageIndex = 1;  % start at first row of phase information matrix
for i = 1:numImages
    phosIntImage = fread(fid, [xDimension, yDimension], format);    % read images one by
one
    phosIntImage = phosIntImage - bckGnd;
    if imageIndex <= numImagesToAnalyze
        if i == imagePhaseMatrix(imageIndex, 1)
            if filterPhos > 0
                phosIntImage = medfilt2(phosIntImage, [filterPhos, filterPhos]);
            end
            %-------------------------------------------------------------------------
----------------------
            % INTENSIFIER CORRECTION
            if imgset == 2
                phosIntImage = phosIntImage.*y;
            else
                phosIntImage = phosIntImage;
            end
%             freq = 0;   %Max intensity of fitted curve
%             y = 4*10^-11*(freq^4)-7*10^-7*(freq^3)+0.004*(freq^2)-1.7147*(freq)+1000;
%             freq = omega/(2*pi);
%             y=y/(4*10^-11*(freq^4)-7*10^-7*(freq^3)+0.004*(freq^2)-1.7147*(freq)+1000);
%             phosIntImage = phosIntImage.*y;

            %SHIFT FREQ1 IMAGE according to alignImages.fig (if needed)
            if rightshift ~= 0  & imgset == 1          %Shift Right
                phosIntImage(:,[rightshift+1:xDimension])=phosIntImage(:,[1:xDimension-
rightshift]);
                phosIntImage(:,[1:rightshift])=0;
                rejectMap(:,[1:rightshift])= 0.9;      %Note image elimination in reject
map
            elseif leftshift ~= 0  & imgset == 1      %Shift Left
                phosIntImage(:,[1:xDimension-
leftshift])=phosIntImage(:,[1+leftshift:xDimension]);
                phosIntImage(:,[xDimension-leftshift:xDimension])=0;
                rejectMap(:,[xDimension-leftshift:xDimension])= 0.9; %Note image
elimination in reject map
            end

            if downshift ~= 0  & imgset == 1           %Shift Down
                phosIntImage([downshift+1:yDimension],:)=phosIntImage([1:yDimension-
downshift],:);
                phosIntImage([1:downshift],:)=0;
```

المنارة للاستشارات
www.manaraa.com

```
                rejectMap([1:downshift],:)= 0.9;       %Note image elimination in reject
map
            elseif upshift ~= 0  & imgset == 1         %Shift Up
                phosIntImage([1:yDimension-
upshift],:)=phosIntImage([1+upshift:yDimension],:);
                phosIntImage([yDimension-upshift+1:yDimension],:)=0;
                rejectMap([yDimension-upshift+1:yDimension],:)= 0.9; %Note image
elimination in reject map
            end
            %------------------------------------------------------------------------
----------------------
            B = B + [sinThetaDArray(imageIndex) * phosIntImage(1:numImagePixels);
                 cosThetaDArray(imageIndex) * phosIntImage(1:numImagePixels);
                 phosIntImage(1:numImagePixels)                     ];
            phosIntImageSquared = phosIntImageSquared +
phosIntImage(1:numImagePixels).^2;  % for R2 calculation
            imageIndex = imageIndex + 1;    % point to next row of matrix
        end
    end
end
fclose(fid);

% build the 3 unknown parameter images a0, a1, and b1 by solving AX = B for X
X = A \ B;
b1 = X(1,:);
a1 = X(2,:);
a0 = X(3,:);

% calculate the R2 map
%  For reasons unknown, the correct a0 is achieved when a0 is unmodified
%    (reduced to its smallest form).  So Calculate R2 before modifing the
%    a0 term.
R2Map = R2FitMatrix3(A, B, b1, a1, a0, phosIntImageSquared, xDimension, yDimension);
R2Map = R2Map';
R2Map(1,1) = 1.0;  %Force the Full Scale (Max=1)
R2Map(1,2) = 0.0;  %Force the Full Scale (Min=0)

%Correct The a0 term for zero-phase removal algorithm
a0 = X(3,:)/1.47;%1.412;

% Create DC map
a0Map = a0;
a0Map = reshape(a0Map, xDimension, yDimension);
a0Map = a0Map';

% Create a1 map
a1Map = a1;
a1Map = reshape(a1Map, xDimension, yDimension);
a1Map = a1Map';

% Create b1 map
b1Map = b1;
b1Map = reshape(b1Map, xDimension, yDimension);
b1Map = b1Map';


% Check for zeros in the a0 map
```

www.manaraa.com

```
i = find (a0Map <= 0);         % reject pixels with a0 = 0 by setting rejectMap pixel to
1
a0Map(i) = eps;                % make a0 a very, very small number
%i = find (a0 == 0);
rejectMap(i) = 0.7;

%Check for zeros in the a1 Map
i = find (a1Map <= 0);         % reject pixels with a1 = 0 by setting rejectMap pixel to
1
a1Map(i) = eps;                % make a1 a very, very small number
%i = find (a1 == 0);
rejectMap(i) = 0.7;

%Check for zeros in the b1 Map
i = find (b1Map <= 0);         % reject pixels with b1 = 0 by setting rejectMap pixel to
1
b1Map(i) = eps;                % make a1 a very, very small number
%i = find (b1 == 0);
rejectMap(i) = 0.7;

% Create phase shift map (thetaMap) from fit data
thetaMap = atan(b1Map./a1Map);
thetaMap = thetaMap - thetaCorrection;

% Create amplitude modulation map (modMap) from fit data
modMap = sqrt(a1Map.*a1Map + b1Map.*b1Map);
modMap = modMap ./ a0Map;
modMap = modMap ./ modCorrection;


%Correct Instrument phase delay
frq = omega/(2*pi);                    %Get frequency in Hz
%-----------------------------------------------------------------------
%%%% --> Two Point Correction factors for different samples <-- %%%%%
thetaFix = 0;                          %No Correction
% %thetaFix = -2.2351*log(frq) + 6.7637;
% %thetaFix = -0.0017*frq - 5.8267;
% %thetaFix = -0.0014*frq - 8.0932;
% %thetaFix = -0.0017*frq - 1.7087;        %Water Cuvette *Old Arc Lamp*
% %thetaFix = -0.003*frq + 2.3747;
% %thetaFix = -0.0014*frq - 5.2001;        %031604_1 Blue
% %thetaFix = -0.0013*frq - 3.6466;        %031604_1 Green
% %thetaFix = -0.0014*frq + 4.1101;        %032204_1 Green
% %thetaFix = -0.0009*frq - 2.318;         %032204_1 Blue
% %thetaFix = -0.0013*frq + 0.1125;        %032204_2 Green
% %thetaFix = -0.0002*frq - 3.3026;        %032204_2 Blue
% %thetaFix = -0.0019*frq + 9.6427;        %032204_3_Green
% %thetaFix = -0.0022*frq + 9.361;         %032204_3_Blue
% %thetaFix = -0.002*frq + 9.7419;         %032304_1_Green
% %thetaFix = -0.0012*frq + 7.1563;        %032304_1_Blue
% %thetaFix = -0.0022*frq + 16.0;%83;      %032604_1_Blue
% %thetaFix = -0.0013*frq + 13.96;         %032604_2_Blue
% %thetaFix = -0.0015*frq + 13.463;        %032904_1 Green
% %thetaFix = -0.002*frq + 13.744;         %032904_1 Blue
% %thetaFix = -0.0017*frq + 13.604;        %032904_2 Green
% %thetaFix = -0.0013*frq + 14.539;        %032904_2 Blue
% %thetaFix = -0.0025*frq + 15.608;        %032904_3 Green
```

www.manaraa.com

```
% %thetaFix = -0.0015*frq + 13.38;        %032904_3 Blue
% %thetaFix = -0.0028*frq + 15.196;       %032904_4 Green
% %thetaFix = -0.0024*frq + 15.654;       %032904_4 Blue
% %thetaFix =  (7*10^-05)*frq + 0.6833;   %033004_1 Green
% %thetaFix = -0.0013*frq + 14.443;       %033004_1 Blue
% %thetaFix = -0.0018*frq + 13.327;       %033004_2 Green
% %thetaFix = -0.0015*frq + 13.471;       %033004_2 Blue
% %thetaFix = -0.0015*frq + 12.688;       %033004_3 Green
% %thetaFix = -0.0005*frq + 12.99;        %033004_3 Blue
% %thetaFix = -0.0015*frq + 13.894;       %040504_5 Blue
% %thetaFix = -0.0022*frq + 13.835;       %040504_5 Green
% %thetaFix = -0.0026*frq + 16.104;       %040704_1 Green
% %thetaFix = -0.0017*frq + 16.557;       %040704_5 Blue
% %thetaFix = -0.0015*frq + 13.893;       %040804_2 Green
% %thetaFix = -0.0015*frq + 16.574;       %040804_2 Blue
% %thetaFix = -0.0012*frq + 16.134;       %040904_1 Blue
% %thetaFix = -0.0017*frq + 14.081;       %040904_1 Green
% %thetaFix = -0.0017*frq + 17.138;       %041204_1 Blue
%thetaFix = -0.0018*frq + 12.873;       %Water *New Arc Lamp* -- 524nm
%thetaFix = -0.0018*frq + 14.87;        %Water *New Arc Lamp* -- 412nm
%----------------------------------------------------------------------

thetaFix = thetaFix*(pi/180);           %Convert to radians
thetaMap = thetaMap - thetaFix;         %Correct Theta

ratio = tan(thetaMap);                  %Determine new a1/b1 ratio
const = sqrt(a1Map.^2+b1Map.^2);

a1Map = sqrt((const.^2)./(1+ratio.^2)); %New b1 term
b1Map = a1Map.*ratio;                   %New a1 term

%end phosMapCalculate2.m
```

المنارة للاستشارات
www.manaraa.com

phosMapDisplay2.m

```
function varargout = phosMapDisplay(varargin)
% PhosMapDisplay2  Application M-file for PhosMapDisplay.fig
%    FIG = PhosMapDisplay launch retina GUI.
%
%       PHOSMAPDISPLAY is the application M-file for PhosMapDisplay.fig.
%       The PhosMapDisplay m-file and fig-file are intended for use with
%       retina2.m (retina2.fig).  Under normal use PhosMapDisplay should
%       only be called from retina2.fig because certain values utilized
%       by PhosMapDisplay are set in retina2.
%
%       PHOSMAPDISPLAY displays specific Maps (PO2, Phase, Modulation, r^2,
%       etc.) for the retina2 analysis program.  In general PhosMapDisplay
%       generates four sets of maps.  A set of maps is generated for the
%       following four parameters: modulation frequency 1, modulation
%       frequency 2, corrected, and contamination.  Modulation frequency 1
%       represents a phase dependent-phosphorescent image set taken at one modulation
%       frequency and Modulation frequency 2 represents a phophorescent
%       image set taken at a *different* modulation frequency.  Best results
%       are achieved when modulation frequency 1 is less than modulation
%       frequency 2.  The "corrected" set represents the set of maps with
%       the zero-phase contamination (present in mod freq1 & 2 image sets)
%       removed from the maps.  The "contamination" map is a map specifically
%       displaying information generated from the zero-phase or contaminated
%       signal.
%
% PHOSMAPDISPLAY2 (Created by Adam Norige) was developed from
%                   PHOSMAPDISPLAY (Created by Ross Shonat)
% Created by Adam Norige 22-Sep-2003
% Last Modified by GUIDE v2.5 22-Oct-2003 13:36:57

global mapNames filterParams map1Value map2Value
global kQ tauO omega1 omega2 filterPhos
global imagePhaseMatrix1 imagePhaseMatrix2 bckGnd thetaCorrection modCorrection
global filePath1 fileName1 filePath2 fileName2
global outputMap1 xDimension yDimension
global regionMap xPolyCoordinates yPolyCoordinates
%Specific Maps
global thetaMap1 modMap1 a0Map1 a1Map1 b1Map1 R2Map1 rejectMap1
global thetaMap2 modMap2 a0Map2 a1Map2 b1Map2 R2Map2 rejectMap2
global thetaMapCorr modMapCorr a0MapCorr a1MapCorr b1MapCorr R2MapCorr rejectMapCorr
global thetaMapCont modMapCont XMap
global tauThetaMap1 PO2ThetaMap1 rejectThetaMap1 tauModMap1 PO2ModMap1 rejectModMap1
global tauThetaMap2 PO2ThetaMap2 rejectThetaMap2 tauModMap2 PO2ModMap2 rejectModMap2
global tauThetaMapCorr PO2ThetaMapCorr rejectThetaMapCorr tauModMapCorr PO2ModMapCorr
rejectModMapCorr
global tauThetaMapCont PO2ThetaMapCont rejectThetaMapCont tauModMapCont PO2ModMapCont
rejectModMapCont


if nargin == 0  % LAUNCH GUI

    % Open the window
   phosMapDisplayFig = openfig(mfilename,'reuse');
```

www.manaraa.com

```
   set(phosMapDisplayFig,'Color',get(0,'defaultUicontrolBackgroundColor'));

   % Generate a structure of handles to pass to callbacks, and store it in the figure.
   phosMapDisplayHandles = guihandles(phosMapDisplayFig);
   guidata(phosMapDisplayFig, phosMapDisplayHandles);

    % load values into phosphorescence mapping window
    set(phosMapDisplayHandles.selectMapTypeTag,'String',mapNames);
    set(phosMapDisplayHandles.selectMapTypeTag,'Value',map1Value);

    % load values into file output section of mapping window
    set(phosMapDisplayHandles.selectOptionTag,'String',strvcat('No Colorbar','Colorbar
w/o Ticks',...
            'Colorbar with Ticks'));
    set(phosMapDisplayHandles.selectOptionTag,'Value',3);
    set(phosMapDisplayHandles.fileNameTag,'String','outmap1.tif');

    % GENERATE MAPS
    % --> Frequency 1 Prelim Maps
    [thetaMap1, modMap1, a0Map1, a1Map1, b1Map1, R2Map1, rejectMap1] =
PhosMapCalculate2(...
        filterPhos, bckGnd, thetaCorrection, modCorrection, 1);

    %Generate Corrected and Contamination Reject Maps
    rejectMapCorr = rejectMap1;
    rejectMapCont = rejectMap1;

    % --> Frequency 1 Refined Maps
    [tauThetaMap1, PO2ThetaMap1, rejectThetaMap1, tauModMap1, PO2ModMap1, rejectModMap1]
= ...
        PO2MapCalculate2(thetaMap1, modMap1, rejectMap1, omega1, tau0, kQ);

    % --> Frequency 2 Prelim Maps
    [thetaMap2, modMap2, a0Map2, a1Map2, b1Map2, R2Map2, rejectMap2] =
PhosMapCalculate2(...
        filterPhos, bckGnd, thetaCorrection, modCorrection, 2);
    % --> Frequency 2 Refined Maps
    [tauThetaMap2, PO2ThetaMap2, rejectThetaMap2, tauModMap2, PO2ModMap2, rejectModMap2]
= ...
        PO2MapCalculate2(thetaMap2, modMap2, rejectMap2, omega2, tau0, kQ);

    % REMOVE ZERO-PHASE SIGNAL
    % --> Generate "Corrected" Maps and "Contamination" Maps
    RMap = (b1Map2.*omega1)./(b1Map1.*omega2);       %a1-frequency ratio
    XMap = ((RMap.*a1Map1)-a1Map2)./(RMap-1);      %Determine a1 contamination


    %Scale X Map
    temp = find(XMap >= max(max(a0Map2))*0.5);
    XMap(temp) = max(max(a0Map2))*0.5;
    temp = find(XMap <= 0);
    XMap(temp) = 0;
    %XMap = XMap.*0.4;
%     figure(1);
%     imshow(XMap,[]),colorbar,colormap(hot);

    % --> Build Contamination Reject Map
```

www.manaraa.com

```
    i = find(XMap <= 0);      % find pixels <= 0
    XMap(i) = eps;            % make zero values a small number
    i = find(XMap <  0);      % find XMap values < 0
    rejectMapCorr(i) = 0.3;   % Note them in reject map
    rejectMapCont(i) = 0.3;   % Note them in reject map

    % GENERATE CORRECTED MAP
    % --> Frequency 1 (freq1) Reconstruction
    a0MapCorr = a0Map2 - XMap;   %XMap=S(r) or XMap=a0Cont
    a1MapCorr = a1Map2 - XMap;   %XMap=S(r) or XMap=a1Cont
    b1MapCorr = b1Map2;          %b11=b1Corr=b1Cont

    % --> Check for zeros in Corrected a0 Map
    i = find (a0MapCorr <= 0);    % find pixels <= 0
    a0MapCorr(i) = eps;           % make a0 a very, very small number
    i = find(a0MapCorr <  0);     % reject a0MapCorr values < 0
    rejectMapCorr(i) = 0.7;       % Note them in reject map

    % --> Check for zeros in Corrected a1 Map
    i = find (a1MapCorr <= 0);    % find pixels <= 0
    a1MapCorr(i) = eps;           % make a1 a very, very small number
    i = find(a1MapCorr <  0);     % reject a1MapCorr values < 0
    rejectMapCorr(i) = 0.7;       % Note them in reject map

    % --> Build Corrected and Contamination R2 Map
    R2MapCorr = (R2Map1+R2Map2)/2;

    % --> Calculate phase shift from fit data
    thetaMapCorr = atan(b1MapCorr./a1MapCorr);

    % --> Calculate modulation from fit data
    modMapCorr = sqrt(a1MapCorr.*a1MapCorr+b1MapCorr.*b1MapCorr)./a0MapCorr;

    % CALCULATIONS FOR CONTAMINATION -- Just for fun
    % --> Calculate phase shift from fit data
    thetaMapCont = atan(0./XMap);

    % --> Calculate modulation from fit data
    modMapCont = sqrt(XMap.*XMap);

    % REFINED CALCULATIONS FOR CORRECTED AND CONTAMINATION
    % --> Calculate PO2 and Such for Corrected Signal
    [tauThetaMapCorr, PO2ThetaMapCorr, rejectThetaMapCorr, tauModMapCorr, PO2ModMapCorr,
rejectModMapCorr] = ...
        PO2MapCalculate2(thetaMapCorr, modMapCorr, rejectMapCorr, omega1, tau0, kQ);

    % --> Calculate PO2 and Such for Contamination Signal
    [tauThetaMapCont, PO2ThetaMapCont, rejectThetaMapCont, tauModMapCont, PO2ModMapCont,
rejectModMapCont] = ...
        PO2MapCalculate2(thetaMapCont, modMapCont, rejectMapCorr, omega1, tau0, kQ);

    % --> Initially Setup the Region Map (ROI)
    regionMap = ones([yDimension xDimension]);

    % --> Display the 1st Maps!
    selectMapTypeTag_Callback(phosMapDisplayFig, [], phosMapDisplayHandles);
```

www.manaraa.com

```
    if nargout > 0
      varargout{1} = phosMapDisplayFig;
   end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

   try
      if (nargout)
         [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
      else
         feval(varargin{:}); % FEVAL switchyard
      end
   catch
      disp(lasterr);
   end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CALLBACKS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------------
function selectMapTypeTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams regionMap xPolyCoordinates yPolyCoordinates
global thetaMap1 modMap1 a0Map1 a1Map1 b1Map1 R2Map1 rejectMap1
global thetaMap2 modMap2 a0Map2 a1Map2 b1Map2 R2Map2 rejectMap2
global thetaMapCorr modMapCorr a0MapCorr a1MapCorr b1MapCorr R2MapCorr rejectMapCorr
global thetaMapCont modMapCont XMap

global tauThetaMap1 PO2ThetaMap1 rejectThetaMap1 tauModMap1 PO2ModMap1 rejectModMap1
global tauThetaMap2 PO2ThetaMap2 rejectThetaMap2 tauModMap2 PO2ModMap2 rejectModMap2
global tauThetaMapCorr PO2ThetaMapCorr rejectThetaMapCorr tauModMapCorr PO2ModMapCorr
rejectModMapCorr
global tauThetaMapCont PO2ThetaMapCont rejectThetaMapCont tauModMapCont PO2ModMapCont
rejectModMapCont
global omega1 omega2

% determine which map to use and set filtering parameters
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');

%Set the frequency label
freq = round(omega1/(2*pi));
txtlabel = strcat(num2str(freq), 'Hz Statistics');
set(phosMapDisplayHandles.freq1TextTag, 'String', txtlabel);
txtlabel = strcat(num2str(freq), 'Hz Map');
set(phosMapDisplayHandles.freq1MapTextTag, 'String',txtlabel);

freq = round(omega2/(2*pi));
txtlabel = strcat(num2str(freq), 'Hz Statistics');
set(phosMapDisplayHandles.freq2TextTag, 'String', txtlabel);
txtlabel = strcat(num2str(freq), 'Hz Map');
set(phosMapDisplayHandles.freq2MapTextTag, 'String',txtlabel);
```

www.manaraa.com

```
clear freq txtlabel;

% Display Some Information: MAP Ranges (min, max, minR2) --> From FilterParams
% -> Frequency 1 <-
set(phosMapDisplayHandles.freq1MinValueTag,'String',num2str(filterParams(map1Value,1),
'%4.1f'));
set(phosMapDisplayHandles.freq1MaxValueTag,'String',num2str(filterParams(map1Value,2),
'%4.1f'));
set(phosMapDisplayHandles.freq1MinR2Tag,'String',num2str(filterParams(map1Value,3),
'%6.4f'));
% -> Frequency 2 <-
set(phosMapDisplayHandles.freq2MinValueTag,'String',num2str(filterParams(map1Value,1),
'%4.1f'));
set(phosMapDisplayHandles.freq2MaxValueTag,'String',num2str(filterParams(map1Value,2),
'%4.1f'));
set(phosMapDisplayHandles.freq2MinR2Tag,'String',num2str(filterParams(map1Value,3),
'%6.4f'));
% -> Corrected <-
set(phosMapDisplayHandles.corrMinValueTag,'String',num2str(filterParams(map1Value,1),
'%4.1f'));
set(phosMapDisplayHandles.corrMaxValueTag,'String',num2str(filterParams(map1Value,2),
'%4.1f'));
set(phosMapDisplayHandles.corrMinR2Tag,'String',num2str(filterParams(map1Value,3),
'%6.4f'));
% -> Contamination <-
set(phosMapDisplayHandles.contMinValueTag,'String',num2str(filterParams(map1Value,1),
'%4.1f'));
set(phosMapDisplayHandles.contMaxValueTag,'String',num2str(filterParams(map1Value,2),
'%4.1f'));
set(phosMapDisplayHandles.contMinR2Tag,'String',num2str(filterParams(map1Value,3),
'%6.4f'));

% Just Get the size of the Maps
[m,n] = size(rejectThetaMap1);

% switch to appropriate map depending on value in popup object
switch map1Value
case 1                  % thetaMap
    thetaDegreeMap1 = (180.0/pi) * thetaMap1;        % convert to degrees for display
    thetaDegreeMap2 = (180.0/pi) * thetaMap2;        % convert to degrees for display
    thetaDegreeMapCorr = (180.0/pi) * thetaMapCorr;  % convert to degrees for display
    thetaDegreeMapCont = (180.0/pi) * thetaMapCont;  % convert to degrees for display
    [outputMap1, outputMaskMap1] = MaskMap2(thetaDegreeMap1, R2Map1,
filterParams(map1Value,3), rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(thetaDegreeMap2, R2Map2,
filterParams(map1Value,3), rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(thetaDegreeMapCorr, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(thetaDegreeMapCont, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(Deg)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(Deg)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(Deg)');
```

المنارة للاستشارات
www.manaraa.com

```
    set(phosMapDisplayHandles.contUnitsTag,'String','(Deg)');
    mapTitle = 'Phase Shift (Theta) Map (Deg)';
case 2                  % modMap
    [outputMap1, outputMaskMap1] = MaskMap2(modMap1, R2Map1, filterParams(map1Value,3), 
rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(modMap2, R2Map2, filterParams(map1Value,3), 
rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(modMapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(modMapCont, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(0 - 1)');
    mapTitle = 'Modulation Map (AU)';
case 3                  % A0Map
    [outputMap1, outputMaskMap1] = MaskMap2(a0Map1, R2Map1, filterParams(map1Value,3), 
rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(a0Map2, R2Map2, filterParams(map1Value,3), 
rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(a0MapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(XMap, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(AU)');
    mapTitle = 'DC Intensity Map (AU)';
case 4                  % R2Map
    [outputMap1, outputMaskMap1] = MaskMap2(R2Map1, R2Map1, filterParams(map1Value,3), 
rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(R2Map2, R2Map2, filterParams(map1Value,3), 
rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(R2MapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(R2MapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(0 - 1)');
    mapTitle = 'R2 Map (0 - 1)';
case 5                  % tauThetaMap
```

www.manaraa.com

```
    [outputMap1, outputMaskMap1] = MaskMap2(tauThetaMap1, R2Map1,
filterParams(map1Value,3), rejectThetaMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(tauThetaMap2, R2Map2,
filterParams(map1Value,3), rejectThetaMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(tauThetaMapCorr, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(tauThetaMapCont, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(usec)');
    mapTitle = 'Lifetime (Theta) Map (usec)';
case 6                  % PO2ThetaMap
    [outputMap1, outputMaskMap1] = MaskMap2(PO2ThetaMap1, R2Map1,
filterParams(map1Value,3), rejectThetaMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(PO2ThetaMap2, R2Map2,
filterParams(map1Value,3), rejectThetaMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(PO2ThetaMapCorr, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(PO2ThetaMapCont, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(mmHg)');
    mapTitle = 'PO2 (Theta) Map (mmHg)';
case 7                  % rejectThetaMap
    [outputMap1, outputMaskMap1] = MaskMap2(rejectThetaMap1, R2Map1,
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMap2, outputMaskMap2] = MaskMap2(rejectThetaMap2, R2Map2,
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(rejectThetaMapCorr, R2MapCorr,
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMapCont, outputMaskMapCont] = MaskMap2(rejectThetaMapCont, R2MapCorr,
filterParams(map1Value,3), zeros(m,n), 0, 1);
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(0-1)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(0-1)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(0-1)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(0-1)');
    mapTitle = 'Rejection (Theta) Map (0-1)';
case 8                  % tauModMap
    [outputMap1, outputMaskMap1] = MaskMap2(tauModMap1, R2Map1,
filterParams(map1Value,3), rejectModMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(tauModMap2, R2Map2,
filterParams(map1Value,3), rejectModMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
```

www.manaraa.com

```
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(tauModMapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(tauModMapCont, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(usec)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(usec)');
    mapTitle = 'Lifetime (Mod) Map (usec)';
case 9                  % PO2ModMap
    [outputMap1, outputMaskMap1] = MaskMap2(PO2ModMap1, R2Map1, 
filterParams(map1Value,3), rejectModMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(PO2ModMap2, R2Map2, 
filterParams(map1Value,3), rejectModMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(PO2ModMapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(PO2ModMapCont, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(mmHg)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(mmHg)');
    mapTitle = 'PO2 (Mod) Map (mmHg)';
case 10                 % Reject modMap
    [outputMap1, outputMaskMap1] = MaskMap2(rejectModMap1, R2Map1, 
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMap2, outputMaskMap2] = MaskMap2(rejectModMap2, R2Map2, 
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(rejectModMapCorr, R2MapCorr, 
filterParams(map1Value,3), zeros(m,n), 0, 1);
    [outputMapCont, outputMaskMapCont] = MaskMap2(rejectModMapCont, R2MapCorr, 
filterParams(map1Value,3), zeros(m,n), 0, 1);
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(0 - 1)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(0 - 1)');
    mapTitle = 'Rejection (Mod) Map (0 - 1)';
case 11                  % A1Map
    [outputMap1, outputMaskMap1] = MaskMap2(a1Map1, R2Map1, filterParams(map1Value,3), 
rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(a1Map2, R2Map2, filterParams(map1Value,3), 
rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(a1MapCorr, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(XMap, R2MapCorr, 
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(AU)');
```

المنارة للاستشارات
www.manaraa.com

```
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(AU)');
    mapTitle = 'a_1 Intensity Map (AU)';
case 12                     % B1Map
    [outputMap1, outputMaskMap1] = MaskMap2(b1Map1, R2Map1, filterParams(map1Value,3),
rejectMap1, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMap2, outputMaskMap2] = MaskMap2(b1Map2, R2Map2, filterParams(map1Value,3),
rejectMap2, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCorr, outputMaskMapCorr] = MaskMap2(b1MapCorr, R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    [outputMapCont, outputMaskMapCont] = MaskMap2(zeros(size(b1Map2)), R2MapCorr,
filterParams(map1Value,3), rejectMapCorr, ...
        filterParams(map1Value,1), filterParams(map1Value,2));
    set(phosMapDisplayHandles.freq1UnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.freq2UnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.corrUnitsTag,'String','(AU)');
    set(phosMapDisplayHandles.contUnitsTag,'String','(AU)');
    mapTitle = 'b_1 Intensity Map (AU)';
end


% put the output map to the window
axes(phosMapDisplayHandles.freq1AxesTag);
imshow(outputMap1,[]),colorbar,colormap(hot),title(mapTitle);
line(xPolyCoordinates, yPolyCoordinates);

axes(phosMapDisplayHandles.freq2AxesTag);
imshow(outputMap2,[]),colorbar,colormap(hot),title(mapTitle);
line(xPolyCoordinates, yPolyCoordinates);

axes(phosMapDisplayHandles.corrAxesTag);
imshow(outputMapCorr,[]),colorbar,colormap(hot),title(mapTitle);
line(xPolyCoordinates, yPolyCoordinates);

axes(phosMapDisplayHandles.contAxesTag);
imshow(outputMapCont,[]),colorbar,colormap(hot),title(mapTitle);
line(xPolyCoordinates, yPolyCoordinates);

% figure (1);
% imshow(outputMapCont,[]),colorbar,colormap(gray),title(mapTitle);

% perform statistical analysis in the region defined by regionMap and outlined by
PolyCoordinates
i=find(regionMap == 1);
j1=find((regionMap == 1) & (outputMaskMap1 == 0));
j2=find((regionMap == 1) & (outputMaskMap2 == 0));
k1 =find((regionMap == 1) & (outputMaskMapCorr == 0));
k2 =find((regionMap == 1) & (outputMaskMapCont == 0));
numROIPoints = length(i);

% Frequency 1
numFitPoints = length(j1);
Mean = mean(outputMap1(j1));
```

المنارة للاستشارات

www.manaraa.com

```
stdDev = std(outputMap1(j1));
minValue = min(min(outputMap1(j1)));
maxValue = max(max(outputMap1(j1)));
set(phosMapDisplayHandles.freq1MeanTag,'String',num2str(Mean, '%8.4f'));
set(phosMapDisplayHandles.freq1StdDevTag,'String',num2str(stdDev, '%8.4f'));
set(phosMapDisplayHandles.freq1MinTag,'String',num2str(minValue, '%8.4f'));
set(phosMapDisplayHandles.freq1MaxTag,'String',num2str(maxValue, '%8.4f'));
set(phosMapDisplayHandles.freq1NumROITag,'String',num2str(numROIPoints, '%6.0f'));
set(phosMapDisplayHandles.freq1NumFitTag,'String',num2str(numFitPoints, '%6.0f'));

% Frequency 2
numFitPoints = length(j2);
Mean = mean(outputMap2(j2));
stdDev = std(outputMap2(j2));
minValue = min(min(outputMap2(j2)));
maxValue = max(max(outputMap2(j2)));
set(phosMapDisplayHandles.freq2MeanTag,'String',num2str(Mean, '%8.4f'));
set(phosMapDisplayHandles.freq2StdDevTag,'String',num2str(stdDev, '%8.4f'));
set(phosMapDisplayHandles.freq2MinTag,'String',num2str(minValue, '%8.4f'));
set(phosMapDisplayHandles.freq2MaxTag,'String',num2str(maxValue, '%8.4f'));
set(phosMapDisplayHandles.freq2NumROITag,'String',num2str(numROIPoints, '%6.0f'));
set(phosMapDisplayHandles.freq2NumFitTag,'String',num2str(numFitPoints, '%6.0f'));

% Corrected
numFitPoints = length(k1);
Mean = mean(outputMapCorr(k1));
stdDev = std(outputMapCorr(k1));
minValue = min(min(outputMapCorr(k1)));
maxValue = max(max(outputMapCorr(k1)));
set(phosMapDisplayHandles.corrMeanTag,'String',num2str(Mean, '%8.4f'));
set(phosMapDisplayHandles.corrStdDevTag,'String',num2str(stdDev, '%8.4f'));
set(phosMapDisplayHandles.corrMinTag,'String',num2str(minValue, '%8.4f'));
set(phosMapDisplayHandles.corrMaxTag,'String',num2str(maxValue, '%8.4f'));
set(phosMapDisplayHandles.corrNumROITag,'String',num2str(numROIPoints, '%6.0f'));
set(phosMapDisplayHandles.corrNumFitTag,'String',num2str(numFitPoints, '%6.0f'));

% Contaminated
numFitPoints = length(k2);
Mean = mean(outputMapCont(k2));
stdDev = std(outputMapCont(k2));
minValue = min(min(outputMapCont(k2)));
maxValue = max(max(outputMapCont(k2)));
set(phosMapDisplayHandles.contMeanTag,'String',num2str(Mean, '%8.4f'));
set(phosMapDisplayHandles.contStdDevTag,'String',num2str(stdDev, '%8.4f'));
set(phosMapDisplayHandles.contMinTag,'String',num2str(minValue, '%8.4f'));
set(phosMapDisplayHandles.contMaxTag,'String',num2str(maxValue, '%8.4f'));
set(phosMapDisplayHandles.contNumROITag,'String',num2str(numROIPoints, '%6.0f'));
set(phosMapDisplayHandles.contNumFitTag,'String',num2str(numFitPoints, '%6.0f'));


%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function defineROITag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
```

www.manaraa.com

```
global filterParams regionMap xPolyCoordinates yPolyCoordinates xDimension yDimension
global thetaMap1 modMap1 a0Map1 a1Map1 b1Map1 R2Map1 rejectMap1
global thetaMap2 modMap2 a0Map2 a1Map2 b1Map2 R2Map2 rejectMap2
global thetaMapCorr modMapCorr a0MapCorr a1MapCorr b1MapCorr R2MapCorr rejectMapCorr
global thetaMapCont modMapCont XMap

global tauThetaMap1 PO2ThetaMap1 rejectThetaMap1 tauModMap1 PO2ModMap1 rejectModMap1
global tauThetaMap2 PO2ThetaMap2 rejectThetaMap2 tauModMap2 PO2ModMap2 rejectModMap2
global tauThetaMapCorr PO2ThetaMapCorr rejectThetaMapCorr tauModMapCorr PO2ModMapCorr
rejectModMapCorr
global tauThetaMapCont PO2ThetaMapCont rejectThetaMapCont tauModMapCont PO2ModMapCont
rejectModMapCont

% redraw appropriate map without poly coordinates
xPolyCoordinates1 = [1; xDimension; xDimension; xDimension ; xDimension; 1; 1; 1];
yPolyCoordinates1 = [1; 1; 1; yDimension; yDimension; yDimension; 1; 1];
% --> Initially Setup the Region Map (ROI)
regionMap = ones([yDimension xDimension]);
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

axes(phosMapDisplayHandles.freq1AxesTag);
[regionMap, xPolyCoordinates, yPolyCoordinates] = roipoly;

%Redraw Maps with poly coordinates
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%------------------------------------------------------------------------
function saveTIFFTAG_Callback(hObject, eventdata, handles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data


%------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%------------------------------------------------------------------------
function fileNameTag_Callback(hObject, eventdata, handles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data


%------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%------------------------------------------------------------------------
function selectOptionTag_Callback(hObject, eventdata, handles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data

%------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%------------------------------------------------------------------------
function varargout = freq1MinValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
```

المنارة للاستشارات
www.manaraa.com

```
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minValue = str2num(get(phosMapDisplayHandles.freq1MinValueTag,'String'));   % get minimum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 1) = minValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = freq1MaxValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
maxValue = str2num(get(phosMapDisplayHandles.freq1MaxValueTag,'String'));   % get maximum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 2) = maxValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = freq1MinR2Tag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minR2 = str2num(get(phosMapDisplayHandles.freq1MinR2Tag,'String'));   % get minimum R2
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 3) = minR2;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = freq2MinValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minValue = str2num(get(phosMapDisplayHandles.freq2MinValueTag,'String'));   % get minimum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 1) = minValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = freq2MaxValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
```

المنارة للاستشارات

www.manaraa.com

```
global filterParams
maxValue = str2num(get(phosMapDisplayHandles.freq2MaxValueTag,'String'));   % get maximum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 2) = maxValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = freq2MinR2Tag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minR2 = str2num(get(phosMapDisplayHandles.freq2MinR2Tag,'String'));   % get minimum R2
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 3) = minR2;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = corrMinValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minValue = str2num(get(phosMapDisplayHandles.corrMinValueTag,'String'));   % get minimum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 1) = minValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = corrMaxValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
maxValue = str2num(get(phosMapDisplayHandles.corrMaxValueTag,'String'));   % get maximum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 2) = maxValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%--------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%--------------------------------------------------------------------------
function varargout = corrMinR2Tag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minR2 = str2num(get(phosMapDisplayHandles.corrMinR2Tag,'String'));   % get minimum R2
```

المنارة للاستشارات
www.manaraa.com

```
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 3) = minR2;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function varargout = contMinValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minValue = str2num(get(phosMapDisplayHandles.contMinValueTag,'String'));   % get minimum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 1) = minValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function varargout = contMaxValueTag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
maxValue = str2num(get(phosMapDisplayHandles.contMaxValueTag,'String'));   % get maximum
value
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 2) = maxValue;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------
function varargout = contMinR2Tag_Callback(hObject, eventdata, phosMapDisplayHandles)
% hObject -  handle to figure (phosMapDisplayHandles.phosMapDisplay2Tag)
% eventdata - reserved for a future version of MATLAB
% interfaceHandles - structure with interfaceHandles and user data
global filterParams
minR2 = str2num(get(phosMapDisplayHandles.contMinR2Tag,'String'));   % get minimum R2
map1Value = get(phosMapDisplayHandles.selectMapTypeTag,'Value');
filterParams(map1Value, 3) = minR2;
selectMapTypeTag_Callback(hObject, [], phosMapDisplayHandles);

%-----------------------------------------------------------------------------
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%-----------------------------------------------------------------------------

%End phosMapDisplay2.m
```

المنارة للاستشارات

www.manaraa.com

phosRegionCalculate2.m

```
function [phosIntensityArray, b1, a1, a0, theta, mod, R2, numberROIPoints, reject] = ...
                PhosRegionCalculate(regionMap, filterPhos, bckGnd, thetaCorrection,
modCorrection, freqnum)

% Reference: Lakowicz et al (1992). Fluorescence lifetime imaging. Anal. Biochem.
202:316-330.
% Define the following equation: I(thetaD) = a0 + a1*cos(thetaD) + b1*sin(thetaD)
%   where I(thetaD) = phosphorescence intensity in a region for a given phase shift
"thetaD"
%          a0, a1, and b1 = unknowns to solve by least squares
% Solve for a0, a1, and b1 by solving the following equation: Ax=B
%   where A = 3 x 3 matrix
%         B = 3 x 1 vector
%         x = 3 x 1 vector = [b1, a1, a0]
% Then
%   phosphorescence phase = theta = atan(b1/a1)
%   modulation amplitude = mod = sqrt(a1^2+b1^2)/a0
%   DC signal amplitude = a0
%
% Input Parameters:
%   filepath1 = path where the raw intensity image file is stored
%   filename1 = name of the raw intensity image file
%   imagePhaseMatrix = matrix holding the image numbers and phase delays to analyze
%       Column 1: imageNumbers       Column 2: phaseDelay (radians) (max rows: numImages)
%   regionMap = region to analyze (means calculated in region)
%   filterPhos = spatial filtering of phosphorescence intensity image (when filtering >=
1)
%   bckGnd = camera background noise level
%   thetaCorrection = correction term for instrumentation phase delay
%   modCorrection = correction term for instrumentation modulation error
%
% Output Parameters:
%   phosIntensityArray = matrix holding the means and standard deviations in the
"regionMap"
%       Column 1: mean for image numbers analyzed   Column 2: standard deviations for
image numbers analyzed
%   b1, a1, a0 = best-fit linear estimates for I(thetaD)
%   R2 = coefficient of determination for fit
%   ROIPoints = number of points in the region analyzed
%   reject = rejection code for fit
%           Value       Interpretation
%           0               Accept, no errors
%           0.2             Reject fit, a0 or a1 = 0

%
% Created February 12, 2002 Ross D. Shonat, PhD
% Modified by Adam Norige (2003)

global omega1 omega2
global filePath1 fileName1 filePath2 fileName2
global imagePhaseMatrix1 imagePhaseMatrix2
global rightshift leftshift upshift downshift
```

www.manaraa.com

```
%Select the correct frequency
if freqnum == 1
    omega = omega1;
    imagePhaseMatrix = imagePhaseMatrix1;
    % open the 1st raw image file
    [fid, errorMsg] = fopen([filePath1 fileName1], 'rb');
    if fid == -1
        disp(errorMsg)
    end
elseif freqnum == 2
    %Obtain the intensifer correction value
    y = IntensifierCorrect(regionMap, filterPhos, bckGnd);
    omega = omega2;
    imagePhaseMatrix = imagePhaseMatrix2;
    % open the 1st raw image file
    [fid errorMsg] = fopen([filePath2 fileName2], 'rb');
    if fid == -1
        disp(errorMsg)
    end
end

% WinX binary data files begin with a 4100 byte header containing the necessary image
acquisition
% parameters, such as image dimension (xDimension, yDimension) and the number of images
(numImages). The
% format of the data is also encoded in the header. Here, header is read in unsigned 16-
bit
% integer format (of length 2050) to obtain the x-axis dimension "xDimension", y-axis
dimension
% "yDimension", the number of images "numImages", and the image data type "dataType".
header = fread(fid, 2050, 'uint16');
xDimension = header(22);        % actual # of pixels on x axis
dataType = header(55);          % experimental data type (0:float, 1: long int, 2: int,
3:short)
yDimension = header(329);       % actual # of pixels on y axis
numImages = header(724);        % number of images in data file
if dataType == 3
    format = 'uint16';
elseif dataType == 2
    format = 'int';
elseif dataType == 1
    format = 'int';
else
    format = 'float';  % dataType = 0
end

% Perform some preliminary calculations
[numImagesToAnalyze, numPhaseDelaysToAnalyze] = size(imagePhaseMatrix);
phaseDelayArray = imagePhaseMatrix(:, 2);
sinThetaDArray = sin(phaseDelayArray);
cosThetaDArray = cos(phaseDelayArray);


% Build the A matrix (3 X 3)
A = [ sum(sinThetaDArray.*sinThetaDArray) sum(sinThetaDArray.*cosThetaDArray)
sum(sinThetaDArray);
```

www.manaraa.com

```
      sum(sinThetaDArray.*cosThetaDArray) sum(cosThetaDArray.*cosThetaDArray)
sum(cosThetaDArray);
      sum(sinThetaDArray)                 sum(cosThetaDArray)
numImagesToAnalyze];

% Build the B matrix (3 X 1) by reading specified raw intensity images
B = zeros(3, 1);                        % initialize to zero
phosIntImageSquared = 0.0;              % holds the value of sum(intensity^2)
imageIndex = 1;                         % start at first row of phase information matrix
j=find(regionMap == 1);
numberROIPoints = length(j);
phosIntensityArray = zeros(numImagesToAnalyze, 2);

for i = 1:numImages
    phosIntImage = fread(fid, [xDimension, yDimension], format);    % read images one by
one
    phosIntImage = phosIntImage';   % rotate to be compatible with regionMap

    %Shift Image if necessary
    if (freqnum == 1)
        %Shift the Image (if needed)
        if rightshift ~= 0          %Shift Right
            phosIntImage(:,[rightshift+1:xDimension])=phosIntImage(:,[1:xDimension-
rightshift]);
            phosIntImage(:,[1:1+rightshift])=0;
        elseif leftshift ~= 0       %Shift Left
            phosIntImage(:,[1:xDimension-
leftshift])=phosIntImage(:,[1+leftshift:xDimension]);
            phosIntImage(:,[xDimension-leftshift:xDimension])=0;
        end

        if downshift ~= 0        %Shift Down
            phosIntImage([downshift+1:yDimension],:)=phosIntImage([1:yDimension-
downshift],:);
            phosIntImage([1:1+downshift],:)=0;
        elseif upshift ~= 0          %Shift Up
            phosIntImage([1:yDimension-
upshift],:)=phosIntImage([1+upshift:yDimension],:);
            phosIntImage([yDimension-upshift+1:yDimension],:)=0;
        end
    end

    % correct for background
    phosIntImage = phosIntImage - bckGnd;   % subtract out background value

    if imageIndex <= numImagesToAnalyze
        if i == imagePhaseMatrix(imageIndex, 1)
            if filterPhos > 0
                phosIntImage = medfilt2(phosIntImage, [filterPhos filterPhos]);
            end

            %-----------------------------------------------------------------------------
            % INTENSIFIER CORRECTION
            if freqnum == 2
                phosIntensityArray(imageIndex,1) = mean(phosIntImage(j)*y);    % store
mean value in first column
```

المنارة للاستشارات
www.manaraa.com

```
                phosIntensityArray(imageIndex,2) = std(phosIntImage(j)*y);     % store
standard deviation in second column
            else
                phosIntensityArray(imageIndex,1) = mean(phosIntImage(j));    % store mean
value in first column
                phosIntensityArray(imageIndex,2) = std(phosIntImage(j));     % store
standard deviation in second column
            end
%             freq = 250;   %min intensity of fitted curve
%             y = 4*10^-11*(freq^4)-7*10^-7*(freq^3)+0.004*(freq^2)-1.7147*(freq)+1529.2
%             freq = omega/(4*pi)
%             y=y/(4*10^-11*(freq^4)-7*10^-7*(freq^3)+0.004*(freq^2)-
1.7147*(freq)+1529.2)
%             phosIntensityArray(imageIndex,1)= phosIntensityArray(imageIndex,1)*y;

            %-----------------------------------------------------------------------------

            B = B + [sinThetaDArray(imageIndex) * phosIntensityArray(imageIndex,1);
                     cosThetaDArray(imageIndex) * phosIntensityArray(imageIndex,1);
                     phosIntensityArray(imageIndex,1)                              ];
            phosIntImageSquared = phosIntImageSquared +
phosIntensityArray(imageIndex,1)^2;  % for R2 calculation
            imageIndex = imageIndex + 1;    % point to next row of matrix
        end
    end
end

% build the 3 unknown parameter images a0, a1, and b1 by solving AX = B for X
X = A \ B;
b1 = X(1,:);
a1 = X(2,:);
a0 = X(3,:);

% calculate the R2 map
R2 = R2FitMatrix3(A, B, b1, a1, a0, phosIntImageSquared, 1, 1);

a0 = X(3,:)/1.47;%1.412;%1.35;

% check for zeroes in a0 and a1 and reject if true
reject = 0.0;         % begin by assuming no rejection of pixels (= 0)
if a0 == 0.0          % reject fits with a0 = 0
    a0 = eps;         % make a0 a very, very small number
    reject = 0.2;
end
if a1 == 0.0          % reject fits with a1 = 0
    a1 = eps;         % make a1 a very, very small number
    reject = 0.2;
end

% Calculate phase shift from fit data
theta = atan(b1/a1);

% Calculate modulation from fit data
mod = sqrt(a1*a1 + b1*b1);
mod = mod / (a0 * modCorrection);

% %Correct Instrument phase delay
```

المنارة للاستشارات
www.manaraa.com

```
frq = omega/(2*pi);                              %Get frequency in Hz
%----------------------------------------------------------------------
%%%% --> Two Point Correction factors for different samples <-- %%%%%
thetaFix = 0;                               %No Correction
% %thetaFix = -2.2351*log(frq) + 6.7637;
% %thetaFix = -0.0017*frq - 5.8267;
% %thetaFix = -0.0014*frq - 8.0932;
% %thetaFix = -0.0017*frq - 1.7087;          %Water Cuvette *Old Arc Lamp*
% %thetaFix = -0.003*frq + 2.3747;
thetaFix = -0.0021*frq + 7.6661;            %112604_2 Green

%thetaFix = -0.0018*frq + 12.873;         %Water *New Arc Lamp* -- 524nm
%thetaFix = -0.0018*frq + 14.87;          %Water *New Arc Lamp* -- 412nm
%----------------------------------------------------------------------
thetaFix = thetaFix*(pi/180);               %Convert to radians
theta = theta - thetaFix;                   %Correct Theta

ratio = tan(theta);                         %Determine new a1/b1 ratio
const = sqrt(a1^2+b1^2);

a1 = sqrt(const^2/(1+ratio^2));             %New b1 term
b1 = a1*ratio;                              %New a1 term

%end phosRegionCalculate2.m
```

المنارة للاستشارات

www.manaraa.com

PO2MapCalculate2.m

```
function [tauThetaMap, PO2ThetaMap, rejectThetaMap, tauModMap, PO2ModMap, rejectModMap] =
...
                PO2MapCalculate(thetaMap, modMap, rejectMap, omega, tau0, kQ)
% Input Parameters:
%   thetaMap = map of phase shifts (in radians)
%   modMap = map of amplitude modulation
%   rejectMap = map of rejected pixels (0: accept, 0.6 rejected in calculation of phos
intensity data)
%   omega = modulation frequency (in rad/sec)
%   tau0 = lifetime in a zero oxygen environment (in usec)
%   kQ = quenching constant (in /mmHg/sec)
%
% Output Parameters:
%   PO2ThetaMap = map of PO2 (in mmHg) calculated from phase shift map
%   rejectThetaMap = map of rejected pixels for theta map PO2 calculations
%      Pixel value     Interpretation
%           0               Accept pixel, no errors
%           0.2             Reject pixel, a0 or a1 = 0  (calculated in another routine)
%           0.4             Reject pixel, tau > (tau0 + tauTolerance)
%           0.6             Reject pixel, < minPO2      (calculated in another routine)
%           0.8             Reject pixel, > maxPO2
%           1.0             Reject pixel, R2 < minR2    (calculated in another routine)
%   PO2ModMap = map of PO2 (in mmHg) calculated from amplitude modulation map
%   rejectModMap = map of rejected pixels for mod map PO2 calculations
%       Pixel value     Interpretation
%           0               Accept pixel, no errors
%           0.2             Reject pixel, a0 or a1 = 0 (calculated in another routine)
%           0.4             Reject pixel, tau > (tau0 + tauTolerance)
%           0.6             Reject pixel, < minPO2      (calculated in another routine)
%           0.8             Reject pixel, > maxPO2      (or tau <= 0)
%           1.0             Reject pixel, R2 < minR2    (calculated in another routine)
%
% Created February 7, 2002 Ross D. Shonat, PhD
% Modified October 22, 2003 Adam S. Norige

rejectThetaMap = rejectMap;
rejectModMap =   rejectMap;

tauTolerance = 0;               % tolerance for tau rejection (usec)

% THETA CALCULATED
% generate lifetime map from phase shift map, rejecting all pixels with tau greater than
tau0 or <= zero
tauThetaMap = tan(thetaMap) * 1000000.0 / omega;    % express results in microseconds
% i = find (tauThetaMap > (tau0 + tauTolerance));
% tauThetaMap(i) = (tau0 + tauTolerance);
% rejectThetaMap(i) = 0.4;

%Check for rejected pixels
i = find (tauThetaMap < 0);     % Find Tau values < 0
tauThetaMap(i) = eps;           % Make them a small number
rejectThetaMap(i) = 0.8;        % Note these values in the reject map
i = find (tauThetaMap == 0);    % Find Tau values == 0
```

www.manaraa.com

```
tauThetaMap(i) = eps;          % Make them a small number

% Generate PO2 Map from phase shift map
PO2ThetaMap = 1./tauThetaMap;
PO2ThetaMap = (1000000.0 * (1 / kQ)) .* (PO2ThetaMap - (1/tau0));

% MODULATION CALCULATED
% generate lifetime map from amplitude modulation map, rejecting all pixels greater than
tauO
tauModMap = 1000000.0 * sqrt((1./(modMap.*modMap)-1) / (omega*omega));  % express results
in microseconds
% i = find (tauModMap > (tau0 + tauTolerance));
% tauModMap(i) = (tau0 + tauTolerance);
% rejectModMap(i) = 0.4;

%Check for rejected pixels
i = find (tauModMap < 0);      % Find Tau values < 0
tauModMap(i) = eps;            % Make them a small number
rejectModMap(i) = 0.8;         % Note these values in the reject map
i = find (tauModMap == 0);     % Find Tau values == 0
tauModMap(i) = eps;            % Make them a small number

% Generate PO2 Map from amplitude modulation map
PO2ModMap = 1./tauModMap;
PO2ModMap = (1000000.0 * (1 / kQ)) .* (PO2ModMap - (1/tau0));

% End PO2MapCalculate2.m
```

المنارة للاستشارات
www.manaraa.com