

# Time of Flight Imaging of Tissue

A thesis submitted in partial fulfillment of the requirement  
for the degree of Bachelor of Science in  
Physics from the College of William and Mary in Virginia,

by

Edward C. Leland

---

Advisor: Prof. William Cooke

---

Prof. Gina Hoatson

Williamsburg, Virginia  
May 2017

## Abstract

Diffuse Optical Tomography (DOT) using Near Infrared Spectroscopy (NIRS) can be used to image tissue. In the past DOT setups have been large and inefficient, with bulky fiber optic bundles clustered around small numbers of light sensors. Time of Flight (TOF) cameras offer a solution to this. TOF cameras use IR light to produce detailed 3D images. The large resolution of TOF images is due to each pixel in a TOF camera being an individual sensor. TOF cameras can have more than 200,000 pixels. By using a TOF camera as a sensor array for DOT we can not only increase the number of sensors by two to three orders of magnitude compared to previous setups but also decrease the size of the apparatus. We attached four fiber optic cables to the laser diodes of a TOF camera and ran them behind a sample of artificial fat. We placed the camera on the other side and by using custom software we took images and analyzed them by constructing amplitude and phase models and comparing them to the original data. We found that we could accurately determine the total thickness of our artificial fat samples. We measured 11.73mm +/-0.01 mm which is within error of the 11.96+/-1.2 mm we calculated. We found that our setup had a systematic error that caused large uncertainties in our phase data. However we performed a trial calculation and determined the scattering coefficient  $\mu_s$  and the absorption coefficient  $\mu_a$  for the fat. Our values were We calculated .00365+/-0.02 counts/pixel for  $k_0$ , 1.12+/-6mm for  $\mu_s'$ , and .0077+/-6mm for  $\mu_a$ . We compared these to the parameters used in NIRFAST, an optical simulation software. Their values are  $\mu_s = .1/\text{mm}$  and  $\mu_a = 0.01/\text{mm}$ . These values are within the uncertainty but the systematic error requires better data before a conclusion can be made. However our accurate measurement of the fat's thickness shows that our technique for DOT using TOF deserves further study.

## 1. Introduction

NIRS is an imaging technique that uses light with a wavelength of approximately 700nm-2500nm to image human tissue. The light is placed close to the area of investigation. The photons from the light source travel through the tissue medium, where some are reflected, many scatter and are absorbed, and some transmit through. This is determined by the absorption and scattering coefficients  $\mu_a, \mu_s$ . Knowing these values it is possible to model the paths of the photons and to reconstruct an image of the medium they travelled through.

DOT uses many NIRS sensors and detectors to create a network. This network can image from many angles meaning that it can be used to reconstruct three dimensional representations of an object. However modern DOT setups are often static and space inefficient meaning they are difficult to use quickly and dynamically. However TOF cameras operate using the same technology as DOT. They use near IR light to construct 3D pictures. TOF cameras are portable, inexpensive, and use typically use a "smart pixel" system that gives them thousands of sensors in a small area.

TOF cameras consist of three basic components. Firstly, there is an IR emitter usually in the form of an IR laser diode. Secondly, there is an IR sensor. These two components are connected and managed by the third component the controller.

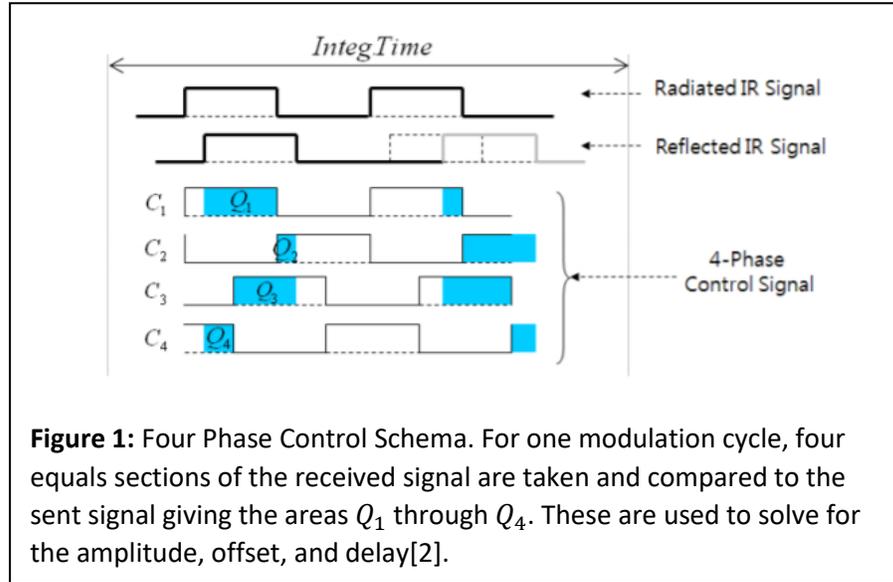
To collect an image, an emitter modulates a train of IR photons at a set frequency  $f$ . We write the sent signal as:

$$s(t) = a \cos(2\pi f t) [1] \quad (1)$$

And the received signal as:

$$s(t) = A \cos(2\pi f (t - \tau)) + B [1] \quad (2)$$

Where A is the amplitude, B is an offset, and  $\tau$  is the delay between the signals. In order to find these values we use a four phase control schema. Below is a visual description of this process:



The Amplitude is calculated as:

$$A = \frac{\sqrt{(Q_1 - Q_2)^2 + (Q_3 - Q_4)^2}}{2} [1] \quad (3)$$

The offset is calculated as:

$$B = \frac{Q_1 + Q_2 + Q_3 + Q_4}{4} [1] \quad (4)$$

The delay is calculated as:

$$2\pi f\tau = \arctan\left(\frac{Q_3 - Q_4}{Q_1 - Q_2}\right) [1] \quad (5)$$

For light propagating through air, the distance the light has traveled is proportional to this delay as:

$$d = c\tau [1] \quad (6)$$

The maximum distance that the light can travel before having the signal wrap around on itself is described by:

$$d_{max} = \frac{c}{2f} = \frac{\lambda}{2} \quad [1] \quad (7)$$

These equations control how a TOF camera functions. For us these are taken care up by the TOF camera which produces both phase and amplitude data. We use the following equations to model photons within the tissue medium:

$$\left( -D\nabla^2 + v\mu_a + \frac{\partial}{\partial t} \right) \Phi(\mathbf{r}, t) = vS(\mathbf{r}, t) \quad (8)$$

$$D = \frac{v}{3\mu'_s} = \frac{v}{3\mu_s(1-g)}$$

$$S = 0 \text{ inside the medium} \quad [12]$$

$$\left( -\frac{\nabla^2}{3\mu'_s} + \mu_a + \frac{\partial}{v\partial t} \right) \Phi(\mathbf{r}, t) = 0$$

$$\left( -\nabla^2 + 3\mu'_s\mu_a + \frac{3\mu'_s}{v} \frac{\partial}{\partial t} \right) \Phi(\mathbf{r}, t) = 0$$

where  $S$  is the photon density at the source,  $\mu_s$  is the scattering coefficient,  $\mu_a$  is the absorption coefficient,  $v$  is the velocity of the light in the medium, and  $\mu'_s$  is the effective scattering coefficient.  $\mu'_s$  is used due to the issue of forward scattering. In a medium where forward scattering can occur the photons need to scatter a number of times before their scattering becomes random. Their effective scattering length is longer than that of a photon that takes only one scattering interaction to scatter randomly. Since  $\frac{1}{\mu_s}$  is equal to the scattering length we adjust  $\mu_s$  by a factor  $(1 - g)$  in order to have a longer scattering length.

Considering the modulated, time-dependent case. Assuming the form of an isotropic point source spreading, we will use:

$$\Phi(\mathbf{r}, t) = A \frac{e^{-\mu r}}{r} e^{i(\omega t - kr)} \quad [12] \quad (9)$$

Where  $\omega = 2\pi f$  where  $f$  is the frequency of the photons. Solving for the real and imaginary parts gives:

$$k = \frac{3\omega \mu'_s}{2v \mu} \quad [12] \quad (10)$$

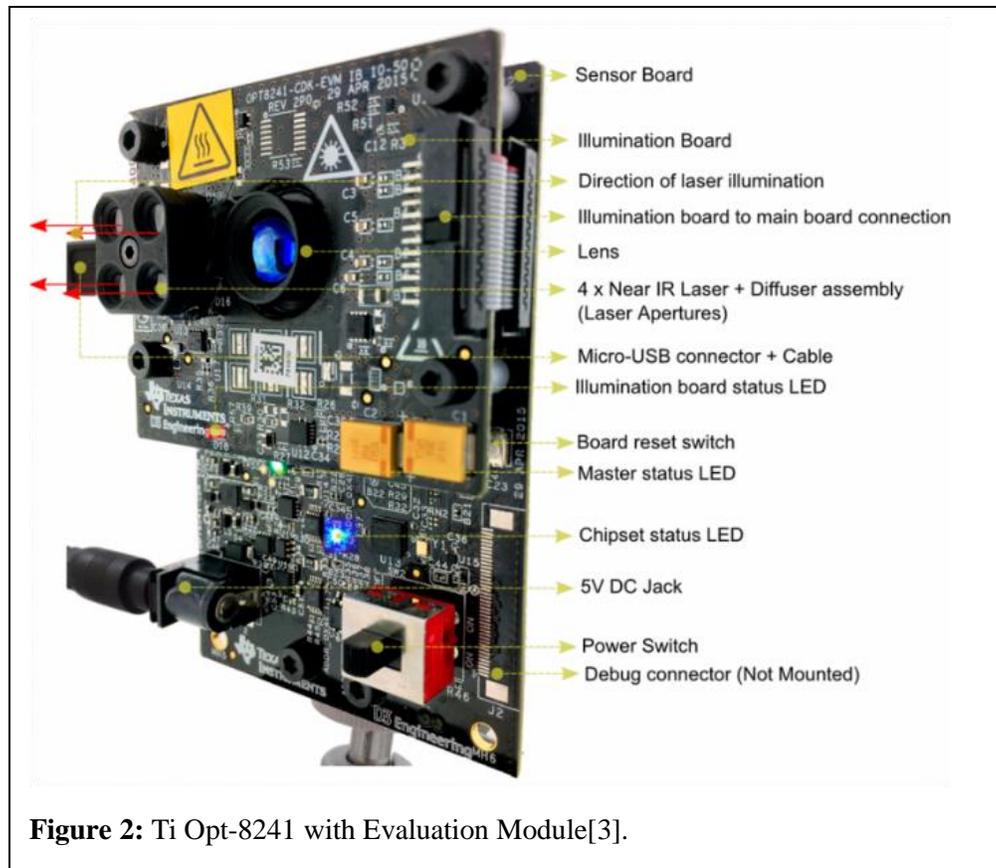
$$\mu \approx \sqrt{3\mu_a \mu'_s} \text{ for } \mu_a \gg \frac{\omega}{2v} \quad [12] \quad (12)$$

With these if we can measure  $k$  and  $\mu$  then we can solve for  $\mu'_s$  and  $\mu_a$ .

## 2. Setup and Procedure

Our goal was to measure the scattering coefficient of fat by using a TOF camera for DOT. To do this we needed to modify the camera's emitter so that we could control the intensity and position of the light source. We also needed to gain access to the controller's code to get the rawest possible data for analysis.

We investigated a few cameras; however, we eventually chose the Ti-OPT-8241 with project board for two reasons. First, Ti used an open source SDK to run the camera, making it possible for us to modify it for our uses. Second, the camera was components were easily accessible so that we could directly access the emitters and the sensor.



Although an open source SDK controlled the TI camera and board, we encountered a number of obstacles in our attempts to modify it. We only had access to Windows machines but we needed to code in C++. To solve this we used Microsoft's Visual Studio 15 (VS15) which allows C++ programming on

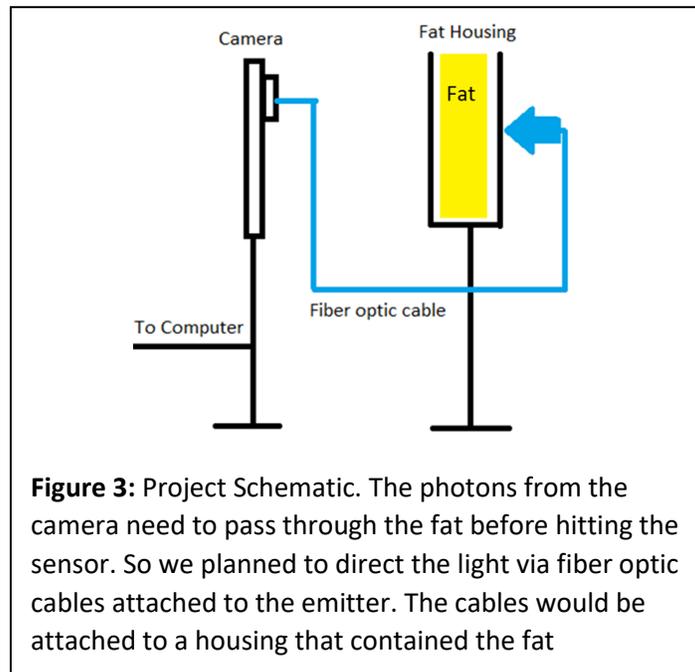
Windows OS [4]. The Ti's SDK needed a version of the Point Cloud Library (PCL) to run [5]. PCL had not updated their installation files in tandem with their third party dependencies. Luckily a Microsoft developer, Tsukasa Sugiura, had built binaries of their own and had released them for public use [6].

The SDK was able to fully install. We attempted to run sample code from the SDK. We found that the SDK libraries were not able to link with the sample code. We used a program called CMake to write a script that automatically linked the code and the libraries [7]. After running the sample code we inserted our own code into the sample code's file and re-compiled. We found that our insertions worked and so we modified the original code to write out phase and amplitude data in a format we could understand. The procedure for operating our program is described in appendix A.

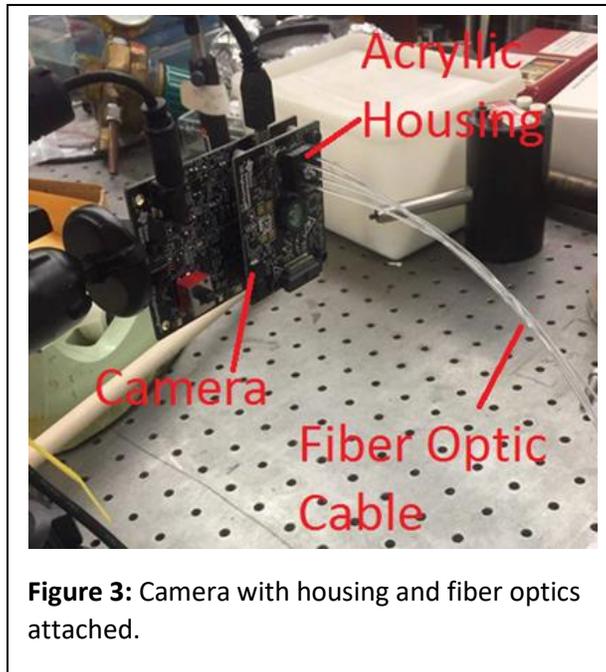
For a TOF camera the difference in phase between two objects in front of the camera should not change when the camera moves toward or away from them. We needed to test this to ensure our camera was functioning as it should. We placed a large board a one end of a table. Then we placed a smaller board 11.8 $\pm$ .05cm in front of it. Then the camera was positioned .5 $\pm$ .0005 meters from the smaller board. Images were taken at .5 meter intervals starting at .5m and going to 2.5m. This was repeated once more. The result was two sets of phase and amplitude data for each distance interval. By averaging over the areas that contained the smaller and the large board we could calculate the average phase value on the board. Subtracting the average over the smaller board from the larger board gave the phase difference between the two board for a given image.

With this complete we moved towards analysis. We decided that we would use MATLAB to analyze the data since it could model 3D sets of data quickly. Since our files were binary data all we had to do was use the simple MATLAB file I/O and we could immediately image a surface.

To allow the camera's photons to pass through fat before reaching the camera's sensor we needed to make some modifications. We created the following simple schematic as a roadmap of what we needed to do:

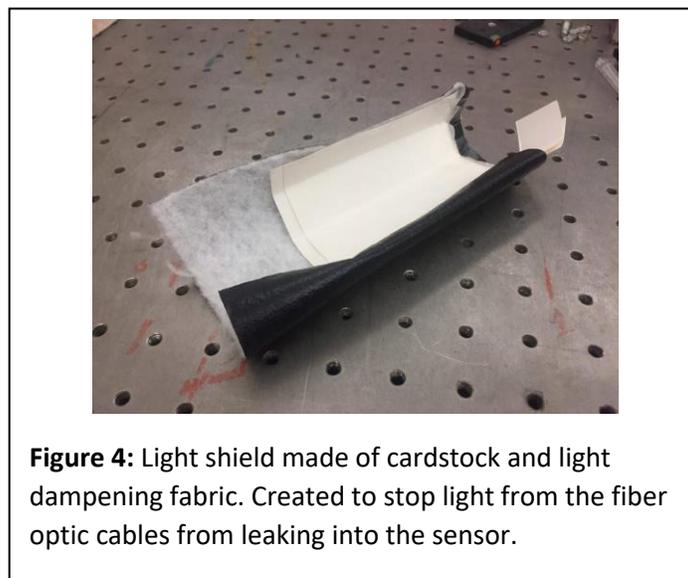


We modified the emitter by redirecting the camera's laser diode emitter. We removed the filters covering the emitter and added a small 3D printed housing with four holes. Into these holes we inserted four 1mm diameter fiber optic cables. This way we could position the IR source independently in relation to the rest of the camera.



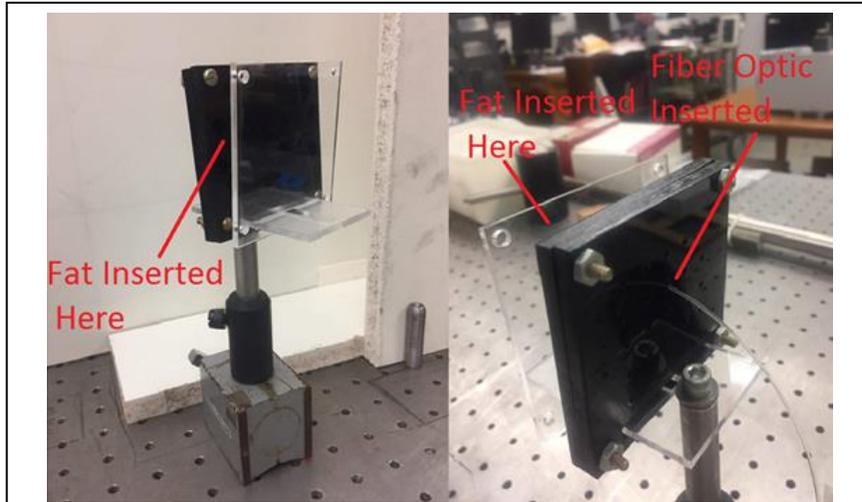
**Figure 3:** Camera with housing and fiber optics attached.

After some testing we saw that the camera was picking up light leaking from the fiber optic cables. This light was strong enough to saturate the camera sensor so we needed to shield the camera. We created a simple light shield out of cardstock and light dampening fabric.



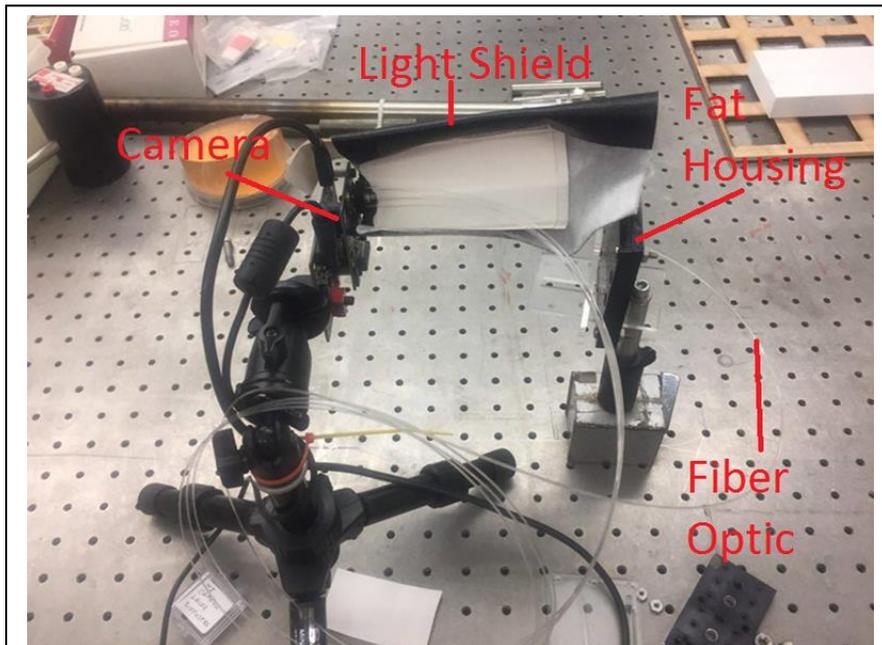
**Figure 4:** Light shield made of cardstock and light dampening fabric. Created to stop light from the fiber optic cables from leaking into the sensor.

Next we needed to build a housing for the artificial fat samples. Another researcher Charles Soulen created a simple vice system out of acrylic. We had a piece of opaque black acrylic as the backing. This piece had 1mm holes in it so that we could insert the fiber optic cables from the camera. To hold the fat in place we placed a clear piece of plastic facing the camera on an adjustable slide.



**Figure 5:** Fat Housing. The fat is inserted between the clear and opaque acrylic. The fiber optic cable is inserted into one of several holes in the opaque acrylic. The array of holes is for use in future experiments when we have more fiber optic cables.

With this the setup was complete:



**Figure 6:** Complete setup.

Before we gathered any fat data we first calculated the pixel to mm conversion factor. When we first tried to take data we saw that using all four fiber optic cables created too much light and saturated the camera's sensor. We decided to use only one and found that it still provided too much light. So we introduced a neutral density 4 filter. The camera no longer saturated. We placed the camera at a fixed distance of 60cm from the fat holder. Then we placed a thin piece of cardstock measuring 17.8mm in width at the front of the fat holder. We took a picture and displayed the data in MATLAB. By comparing how many pixel data points constituted the width in the camera's picture vs real life mm we found that for each pixel there were .26 +/- .02mm. With this information we were ready to take fat data.

We took three runs of data at 50MHz each consisting of ten pictures. The first was with 1 layer of fat and a neutral density 1 filter. The second was with just two layers of fat. The third was with three layers of fat. For each data run we read in all ten frames and averaged them together. Next we imaged the data in a surface plot and isolated the section that contained the light from our emitter. These sections have a Gaussian shape with the maximum in the center. We recorded this center and the amplitude at that point.

Using the center point and the maximum we constructed a mathematical model of the data. With the center as the maximum the data values decayed like a Gaussian exponential the further they were radially from the center. For an individual pixel data point the model's formula looked like:

$$A_{i,j} = \frac{A_0 z_c e^{\frac{z_c}{\omega}}}{e^{\sqrt{\frac{((j-x_c)^2 + z_c^2 + (i-y_c)^2)}{\omega}}}} \cdot \frac{1}{((j-x_c)^2 + z_c^2 + (i-y_c)^2)} \quad (13)$$

The values represented are  $A_{i,j}$ , the amplitude value of the model's pixel data point,  $i$  and  $j$  the position of the pixel,  $x_c$  and  $y_c$ , the coordinates of the center maximum,  $A_0$ , the maximum amplitude of the center point,  $z_c$ , the thickness of the fat sample in the  $z$  direction, and  $\omega$  which is the decay parameter. Our goal is to compare the calculated  $z_c$  with the measured width of the fat. We want an  $\omega$  that fits the data well in order to find  $\mu_a, \mu_s$  through the relation  $\omega = \frac{1}{\mu}$  where  $\mu$  is approximately equivalent to  $\sqrt{3\mu_a\mu_s'}$  [12].

Everything is accounted for except for  $z_c$  and  $\omega$ . To accurately find these we set these arbitrary values and subtracted the results of the model from our original data. From there it was simple manipulation of the variables to reduce that difference. When we were satisfied with our model we recorded our values for the variables and used our conversion factor from earlier to calculate real world values from the pixel data. We also averaged the result of the subtraction of our model from the original data getting the average error per pixel data point.

In addition we created a model for phase:

$$\varphi_{i,j} = \varphi_0 + k_0 \sqrt{((j-x_c)^2 + z_c^2 + (i-y_c)^2)} \quad (14)$$

Where  $\varphi_{i,j}$  is the phase count value of a pixel data point,  $k_0$  is the absorption decay parameter,  $\varphi_0$  is an offset, and the remaining variables are the same as for the amplitude model. Similar to the amplitude model we vary the unknowns  $k_0$  and  $\varphi_0$  to create  $\varphi_{i,j}$  values that are close to the  $\varphi$  values we collected with the TOF camera. We do this because of the relation between  $k_0$  and  $\mu_a, \mu_s$ :

$$k_0 = \frac{2\pi}{\lambda} = \frac{\lambda_{vac}}{n} \sqrt{\frac{4\mu_a}{3\mu_s}} \quad [12] \quad (15)$$

Where  $\lambda$  is the wavelength of the photons in the fat and  $\lambda_{vac}$  is the wavelength of the photons in a vacuum. By comparing the result for  $\mu$  from the amplitude model with our result for  $k_0$  from the phase model we can calculate  $\mu_s'$ . And with  $\mu_s'$  we can calculate  $\mu_a$ .

We can compare these values against known values for tissue. We used the default values from a simulation program called NIRFAST. Their values are  $\mu_s' = .1/\text{mm}$  and  $\mu_a = 0.01/\text{mm}$  [8][9].

### 3. Results

We created the following tables for our phase calibration:

50MHz	$\Delta\varphi$ Picture 1 (counts)	$\Delta\varphi$ Picture 2 (counts)	Conversion Factor (degrees/count)
.5m	192	192	.37
1m	184+/-4.0	189+/-4.0	.38+/-0.001
1.5m	159+/-3.0	163+/-3.0	.44+/-0.001

**Table 1:** 50MHz Phase Calibration

25MHz	$\Delta\varphi$ Picture 1 (counts)	$\Delta\varphi$ Picture 2 (counts)	Conversion Factor (degrees/count)
1m	77+/-0.7	78+/-0.7	.0457+/-0.0004
2m	84	84	.0421
2.5m	72+/-4.0	67+/-3.0	.0509+/-0.003

**Table 2:** 25MHz Phase Calibration

We created the following table for our  $z_c$  and  $\omega$  values calculated from our amplitude model:

	$z_c$ (mm)	$\omega$ (mm)
1 Layer + 1D Filter	1.89+/-0.3	3.25+/-0.7
2 Layers	6.16+/-0.7	5.82+/-0.9
3 Layers	11.96+/-1.2	6.24+/-0.8

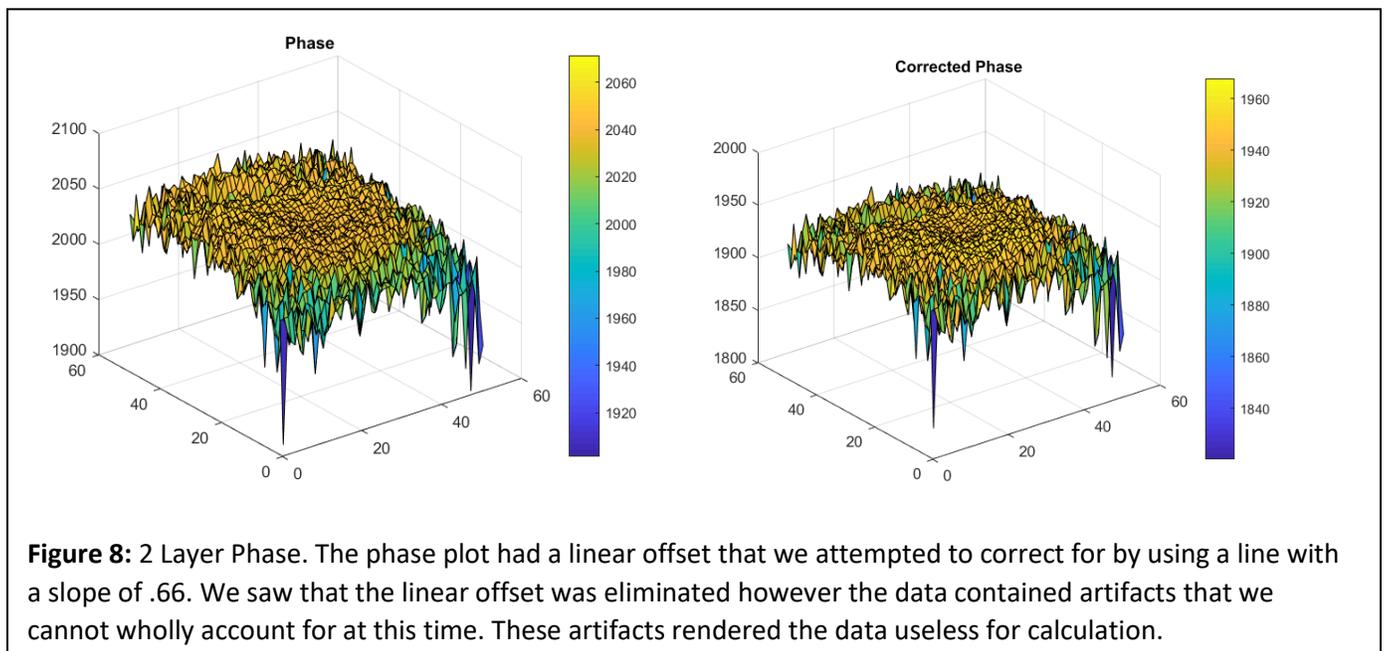
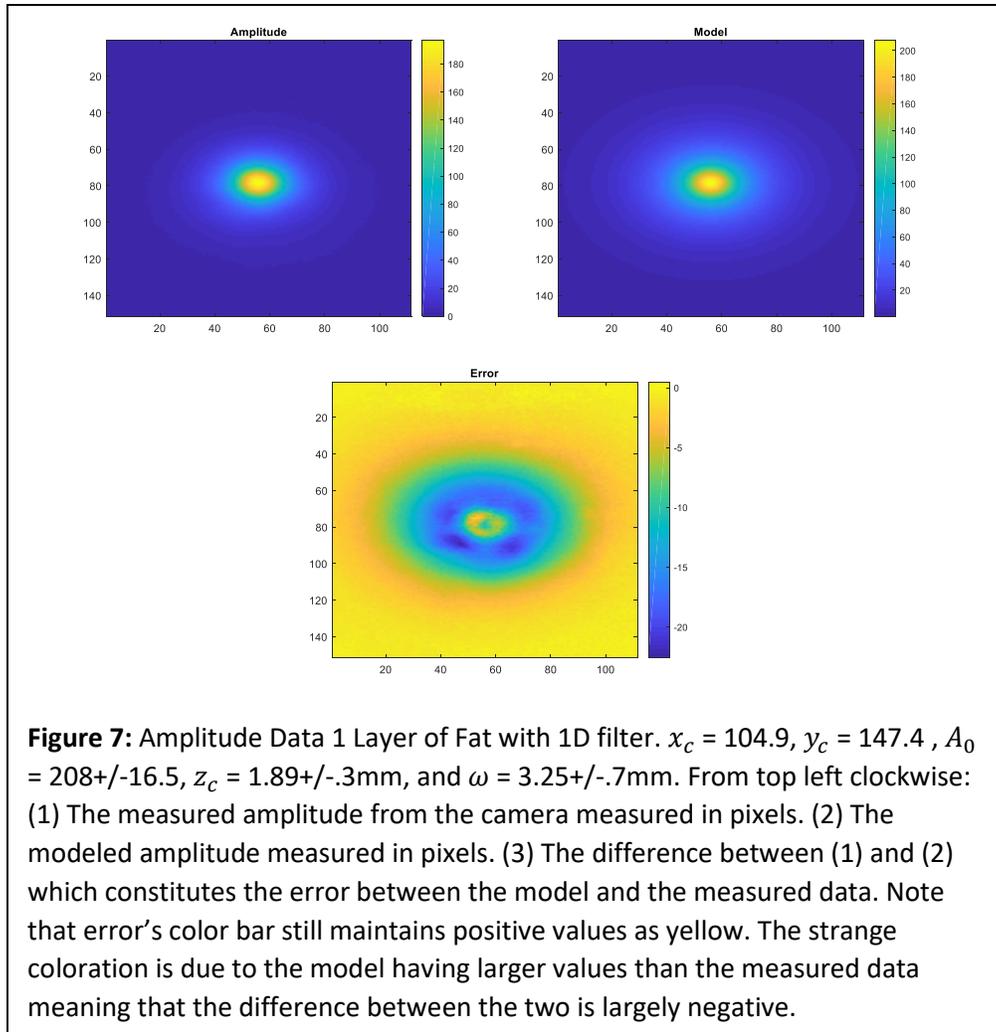
**Table 3:**  $z_c$  and  $\omega$  for 1, 2, and 3 layers of fat. Calculated from the amplitude data.

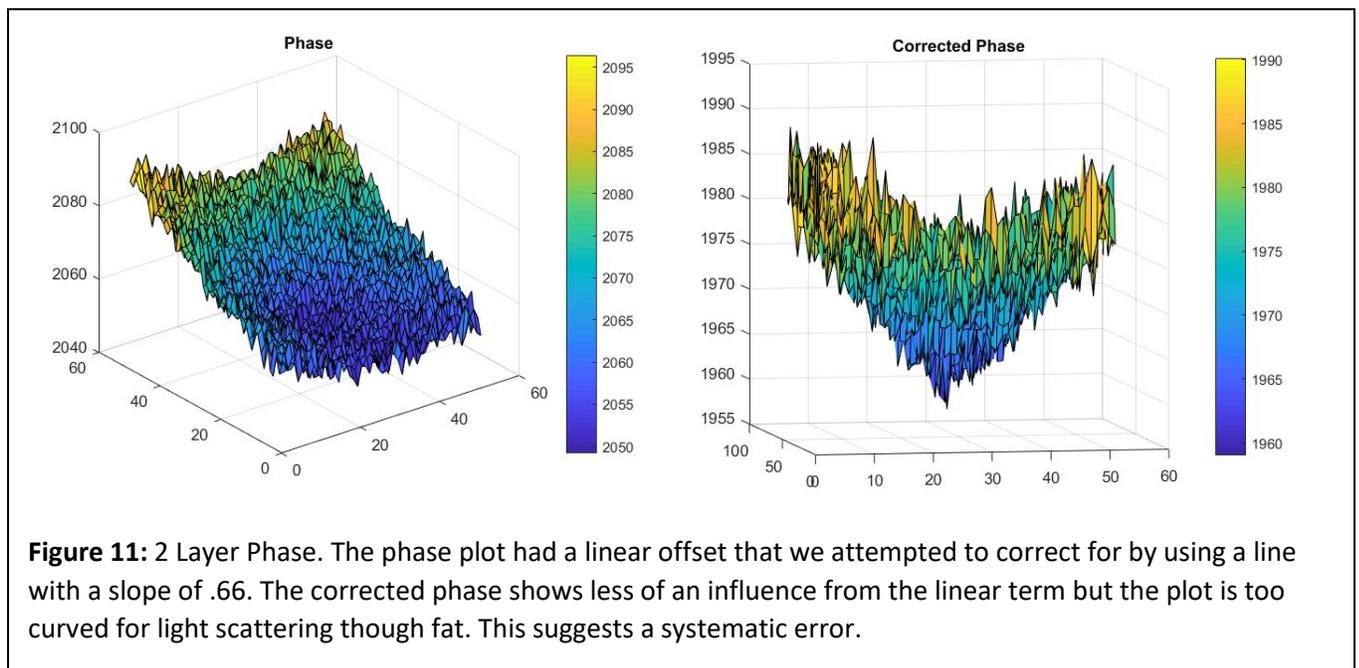
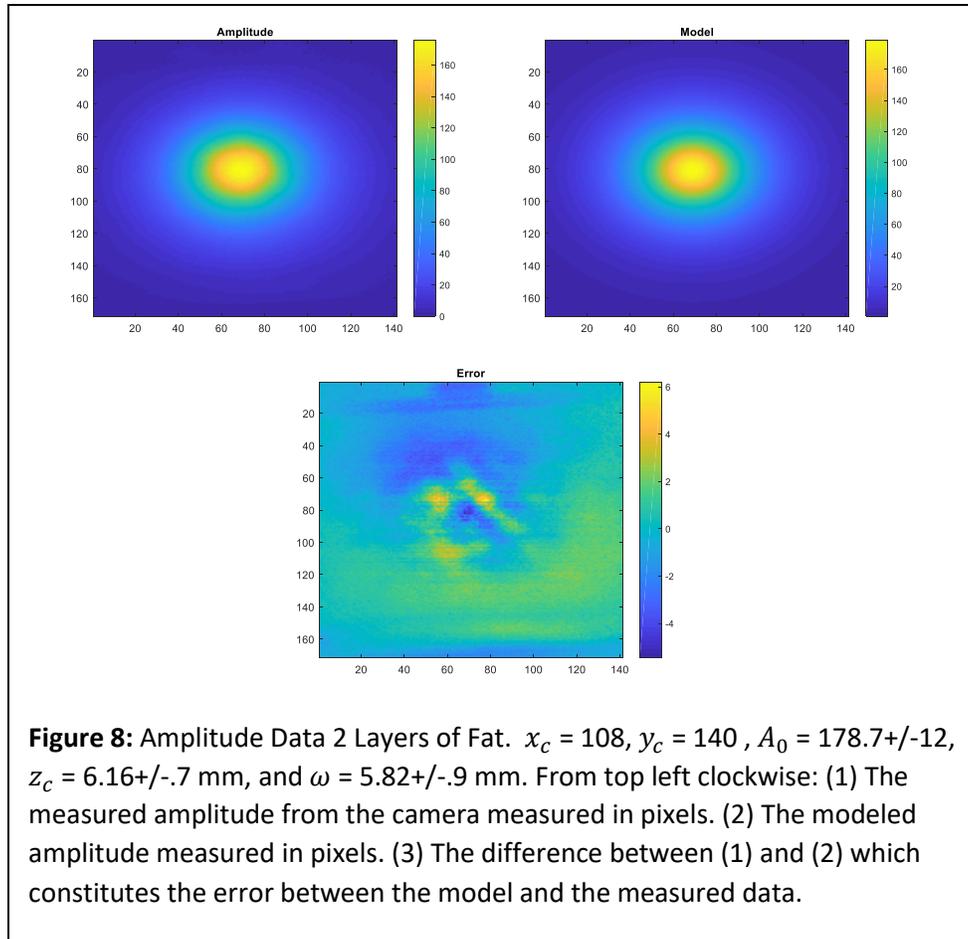
The artificial fat we used measured 2.99 mm, 3.69 mm, and 4.10 mm all with uncertainty +/-0.005 mm. Total = 10.79 +/- 0.01 mm. Our total  $z_c$  for three layers was 11.96+/-1.23 mm.

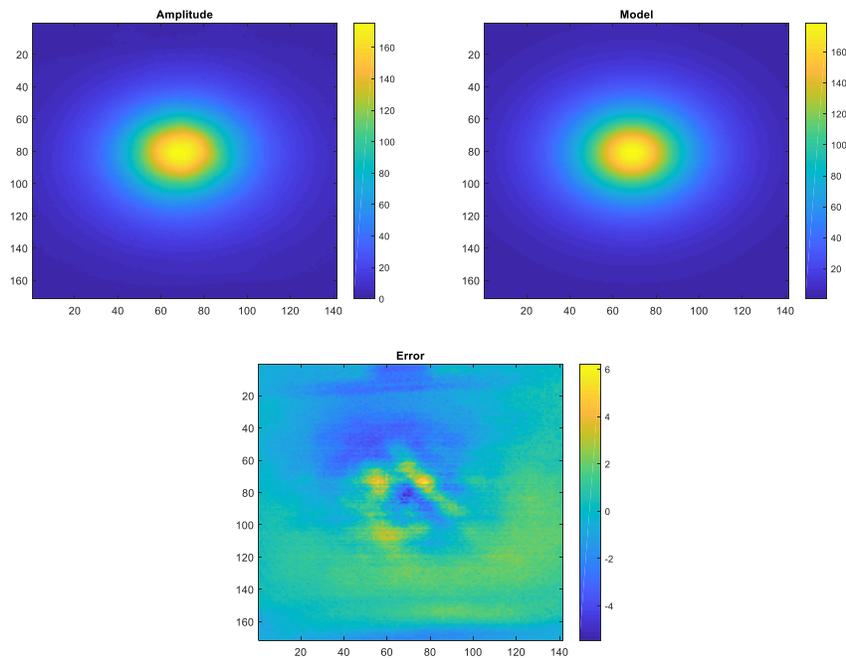
Using our phase model we calculated the following values for  $k_0$ ,  $\mu_s'$ ,  $\mu_a$ :

	$k_0$ (counts/pixel)	$\mu_s'$ (mm)	$\mu_a$ (mm)
3 Layer	.00365+/-0.02	1.12+/-6	.0077+/-0.042

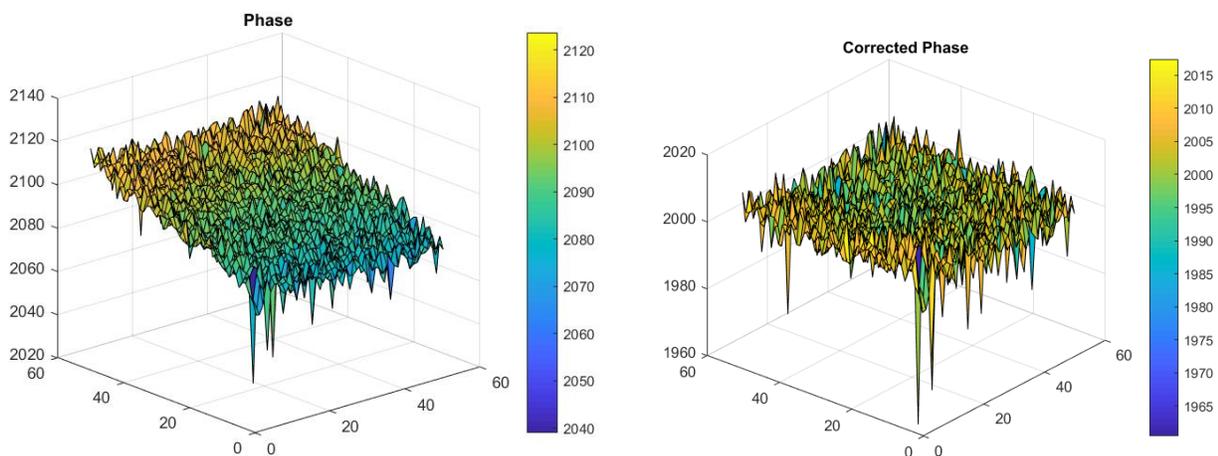
Uncertainties for these values had to be visually inspected due to artifacts within the data. This data is merely a trial calculation performed as a test and should not be considered definitive. Starting on the next page are MATLAB renderings of our amplitude and phase measurements.







**Figure 9:** Amplitude Data 3 Layers of Fat.  $x_c = 117.6$ ,  $y_c = 140.9$ ,  $A_0 = 54.5 \pm 3$ ,  $z_c = 11.96 \pm 1.23$  mm, and  $\omega = 6.24 \pm 0.78$  mm. From top left clockwise: (1) The measured amplitude from the camera measured in pixels. (2) The modeled amplitude measured in pixels. (3) The difference between (1) and (2) which constitutes the error between the model and the measured data.



**Figure 10:** 3 Layer Phase. The phase plot had a linear offset that we corrected for by using a line with a slope of .66. The corrected phase shows a small curvature which is what we would expect for light scattering through tissue. However upon closer inspection the three layer suffers from the same issues as the 2 and 1 layer data.

## 4. Conclusions

We found that our phase calibration results were subject a problematic error. The camera would sometimes generate garbage data for the first frame. This lead to a few of our trials not being viable. We learned that to avoid this we needed to stream multiple images and take the 2<sup>nd</sup> or 3<sup>rd</sup> frame. However the trials left without error for both 50MHz and 25MHz show a consistency in  $\Delta\phi$  from the first set of pictures to the second. This shows that our camera functions as intended and allows us to take phase and amplitude data without fear of systematic fluctuation in measured values. Also we were able to produce a conversion factor between phase counts and degrees. This can be used in future experiments to better investigate the phase images produced by the camera.

Our average errors amplitude errors were low, on the order of 1 pixel. This means our models were very close to the real world data. This means that we chose our  $z_c$  and  $\omega$  values well. Thus our analysis is not likely to be error prone.

$\omega$  for the 2 and 3 layer data only differed by .49mm. Since all of the data was run through synthetic fat all of these should have the similar  $\omega$  values. The 2 and 3 layer data are within error of each other which gives us confidence that we correctly measured the scattering coefficient for fat. However the 1 layer data is far removed from the other two with  $\omega = 3.25\pm.7$ . At first this was disconcerting but we realized that the 1mm diameter fiber optic cable we were using for a source was  $\frac{1}{2}$  the area of the spot in our data. This meant that the light would mostly go straight through the fat without having time to scatter due to the relatively large size and power of the source. This would reduce our measurement of the scattering coefficient. In addition the light coming from the cable is not a perfect point source. As such with only one layer of fat any internal structure the light has would interfere with scattering especially when the radius of the source is large compared to the radius of the scattering.

The artificial fat we used measured 2.99 mm, 3.69 mm, and 4.10 mm all with uncertainty  $\pm.005$  mm with the total = 10.79  $\pm.01$  mm. Our total  $z_c$  for three layers was 11.96 $\pm$ 1.23 mm. These two totals are within range of each other which shows we can accurately measure the width of fat samples with our technique. It is of not that the fat samples' thickness could be larger or smaller depending on how much pressure the measuring calipers put on the fat. When taking the second two measurements for the first time I put too much pressure and they measured 2.53mm  $\pm.005$  mm and 3.85mm  $\pm.005$  mm. This means that our fat samples could have been compressed or allowed to expand each time we put a new sample in the holder. By moderately squeezing the samples I obtained a combined thickness of 11.73mm  $\pm.01$  mm which is close to the 11.96 $\pm$ 1.23 mm value we found. Considering compression during measurement as a factor for error means that our value for  $z_c$  might be even closer to the real thickness than we though. This shows that we accurately modeled the thickness of our samples.

Our estimation of for  $k_0$ ,  $\mu_s'$ , and  $\mu_a$  were hampered by discrepancies in our data. The first layer phase data showed a curvature that had strange artifacts that caused the area around the maximum point to be deformed. We are unsure as to what caused this problem but one possibility is that the structure of the light coming form the fiber optic cable caused the sensor to register erroneous values for the phase. The two layer fat phase is much too curved to be a realistic representation of the system. As of now we are unable to offer a definite explanation as to why this occurred but we believe it might be a systematic error related to the camera. We thought that the three layer fat performed well however upon closer inspection we found that the data also contained artifacts. These bear more investigation but since the three layer data was the best set of the three we decided to run a trial calculation. We were able to calculate the curvature of the phase by using our model. We calculated .00365 $\pm$ .02 counts/pixel for  $k_0$ , 1.12 $\pm$ 6mm

for  $\mu_s'$ , and .0077+/-6mm for  $\mu_a$ . However our attempts to use the least squares method to calculate uncertainty were disturbed by artifacts within the data so we had to visually determine the error. This makes our findings unreliable and as such we need to perform more tests and gather more data before we can make any definitive claims about the scattering and absorption coefficients. It is worth noting that the values that we did find are within error of the NIRFAST values.

We found that it was possible to modify a TOF camera to get photon data through synthetic fat. We also saw that it was possible to construct accurate models of that data. And, though we require better data, our trial calculation of the scattering and absorption coefficients was promising. This shows that our concept for a TOF imaging system for NIRS is worth further study. We need to improve a few components before we continue. Firstly we need to separate the emitter/fiber optic cable system from the camera. Having this on a separate board will reduce the amount of unwanted light leaking into pictures as well as negating the need for a light shield. Secondly we need to reduce the power of the emitter/fiber optics as they can easily saturate the camera's sensor. We need a fat housing that is easier to add or take away samples from. This would help keep our experiments consistent which would eliminate the stretching/compression problems we saw in our  $z_c$  measurements. We also need to run more data gathering tests to identify and remove the artifacts that damaged our phase data.

## References

1. Yuan, Wenjia , Richard E. Howard, Kristen J. Dana, Ramesh Raskar, Ashwin Ashok, Marco Gruteser, and Narayan Mandayam. "Phase Messaging Method for Time-of-flight Cameras." *Rutgers University, WINLAB*: 3-4. Accessed May 1, 2017. doi:10.1075/ps.5.3.02chi.audio.2f.
2. Miles Hansard, Seungkyu Lee, Ouk Choi, Radu Horaud. *Time of Flight Cameras: Principles, Methods, and Applications*. Springer, pp.95, 2012, SpringerBriefs in Computer Science, ISBN 978-1-4471-4658-2.
3. Design Center. "Evaluation Module for OPT8241 3D Time-of-Flight (ToF) Sensor." Evaluation Module for OPT8241 3D TimeofFlight ToF Sensor Version History. June 28, 2016. Accessed April 18, 2017. <https://www.element14.com/community/docs/DOC-82660/1/evaluation-module-for-opt8241-3d-time-of-flight-tof-sensor>.
4. Welcome to Visual Studio 2015. Accessed April 18, 2017. <https://msdn.microsoft.com/en-us/library/dd831853.aspx>
5. "Documentation - Point Cloud Library (PCL)." Accessed January 10, 2017. <http://pointclouds.org/documentation/>.
6. Sugiura, Tsukasa. "Point Cloud Library 1.8.0 Has Been Released |." June 18, 2016. Accessed January 10, 2017. <http://unanancyowen.com/en/pcl18/>
7. "Build, Test and Package Your Software with CMake." Accessed January 10, 2017. <https://cmake.org/>.
8. M. Jermyn, H. Ghadyani, M.A. Mastanduno, W. Turner, S.C. Davis, H. Dehghani, and B.W. Pogue, "Fast segmentation and high-quality three-dimensional volume mesh creation from medical images for diffuse optical tomography," *J. Biomed. Opt.* 18 (8), 086007 (August 12, 2013), doi: 10.1117/1.JBO.18.8.086007
9. H. Dehghani, M.E. Eames, P.K. Yalavarthy, S.C. Davis, S. Srinivasan, C.M. Carpenter, B.W. Pogue, and K.D. Paulsen, "Near infrared optical tomography using NIRFAST: Algorithm for numerical model and image reconstruction," *Communications in Numerical Methods in Engineering*, vol. 25, 711-732 (2009)
10. Texas Instruments. "3D Time of Flight (ToF) Sensors." 2015. Accessed January 10, 2017. <http://www.ti.com/product/OPT8241/datasheet>.
11. "About NIRS (Principle of Operation and How It Works)." About NIRS (Principle of Operation and How It Works) | SHIMADZU AUSTRIA. 2017. Accessed April 17, 2017. <https://www.shimadzu.eu.com/about-nirs-principle-operation-and-how-it-works>.
12. W. E. Cooke, Private Communication, 2017

# APPENDIX A

## TOF Camera and Program, Setup and Operating Procedures

### 1. Downloads (In Order) For a Windows XP Machine

- 1.1. Download VS2015 (This link changes with each update try to find a legacy download link. If you can't, try to find the closest version to 2015)
- 1.2. Download CMake from: <https://cmake.org/>
- 1.3. Download Voxel Viewer from the Ti website: <http://www.ti.com/tool/opt8241-cdk-evm>
- 1.4. Download PCL 1.8 binaries from: <http://unanancyowen.com/en/pcl18/>
- 1.5. Download Qt 5.4 from: <https://download.qt.io/archive/qt/5.4/5.4.1/>
- 1.6. Download Voxel SDK from: <https://github.com/3dtof/pcl/releases/tag/1.7.2>
- 1.7. Download example program from: <https://github.com/3dtof/voxelsdk-examples/releases>
- 1.8. Connect the camera to the computer and run Voxel Viewer to see if everything is set up.

### 2. Building and Running an Example Program

- 2.1. Extract the DepthCapture folder from the example program folder
  - 2.1.1. In this create two folders src and build
  - 2.1.2. Move CMakeLists.txt and DepthCapture.cpp into the src folder
- 2.2. Open CMake and set the source directory to the src folder and the build directory to the build folder
  - 2.2.1. Click configure and choose VS2015 native compilers
  - 2.2.2. Click generate
- 2.3. Navigate to the build folder and now there should be a new directory system inside it
  - 2.3.1. Double click on the sln file, this will open VS2015
- 2.4. In VS2015 use the directory on the right to navigate to the DepthCapture.cpp file
  - 2.4.1. Make sure the pull down tab in the toolbar at the top of the screen is set to release instead of debug.
  - 2.4.2. Go to the build tab and click build solution
  - 2.4.3. This will create a new directory in the build folder named Release
- 2.5. Navigate to the Program Files folder in OS (C:)
  - 2.5.1. Find the Voxel SDK folder and open it
  - 2.5.2. (Optional, depending on the OS you can move the release program over to a new machine after doing this) Enter the lib folder and copy and paste every .dll file into the new Release folder
  - 2.5.3. (Optional) In lib there is a directory named voxel copy and paste every dll file here into the new Release folder
- 2.6. Navigate to the Release folder
  - 2.6.1. Open a cmd window

- 2.6.2. Drag and drop the .exe file in the Release folder into the cmd window and hit enter
  - 2.6.3. This should make the program tell you that it needs some arguments to run. It describes each in detail
  - 2.6.4. Find the Vendor ID, Product ID, and Serial Number by using the device manager. Open device manager and locate the camera. Open its properties menu and navigate to the details tab. Switch to Hardware IDs
  - 2.6.5. Drag and drop again this time with the command line arguments and hit enter.
- 2.7. The program should run. The program's code is under your control now modify it as you see fit.

### 3. Using the Current Program

#### 3.1. Writing Files

- 3.1.1. In the Students directory navigate to the desktop
- 3.1.2. Enter the Ti\_Camera\_app folder
  - 3.1.2.1. Source Code is in the Source\_Code folder
  - 3.1.2.2. Built code is in the Built\_Code folder
- 3.1.3. Enter the Built\_Code folder then the Ti\_Writing folder
- 3.1.4. Enter Release
- 3.1.5. Open a cmd window and navigate to a directory where you want the files to be written out
- 3.1.6. Drag and drop the .exe file from Release into the cmd window and hit enter
- 3.1.7. The program will ask for a file name, give it one (filenames will have \_Amplitude and \_Phase appended to them)
- 3.1.8. The files are now written

#### 3.2. Imaging files

- 3.2.1. Navigate to the Ti\_Imaging folder in Built\_Code
- 3.2.2. Enter Release
- 3.2.3. Open a cmd window and navigate to where the files you want to image are stored
- 3.2.4. Drag and drop the .exe from Release into the cmd window and hit enter
- 3.2.5. It will ask for a file name. Only enter the part without the \_Amplitude or \_Phase
- 3.2.6. Hit enter and two images named Phase and Amplitude should show up
- 3.2.7. To exit hit any key