OCT DOPPLER - DLS TUTORIAL

BRENDAN HUANG

1. Introduction

Optical coherence tomography (OCT) is a non-invasive, microscopic medical imaging technique that can give information about microfluidic flow, including blood and mucociliary flow [1]. In particular, OCT can make use of the Doppler effect, a shift in the frequency of returning light, in order to estimate flow along the direction of light propagation. Dynamic light scattering (DLS) is a signal processing technique that has been proposed to be used to recover the dynamics of flow in all three-dimensions [2,3].

This tutorial gives a short overview of basic OCT signal processing. In it, you will learn what type of information we recover in an OCT measurement, and how we can process that measurement over time to get an estimate of fluid flow velocity. Additionally, OCT is often considered the light-analogue of ultrasound. All of the signal processing machinery presented here can equally be applied towards ultrasound signals as well.

In this tutorial, you will:

(1) Work with complex-valued OCT data in standard imaging

(2) Calculate Doppler measurements using a couple different algorithms

(3) Calculate the power spectrum of the signal

(4) Calculate transverse flow speed using DLS-OCT

1.1. Overview of Files.

In order to work on this tutorial, copy the four files to your working directory. They should be:

(1) tadpolcpx.mat, a Matlab matrix containing the complex-valued B-scan (image) demonstrating standard structural imaging with OCT

Date: Created 21 April 2014. Updated 14 May 2017.

(2) tubecomplex.mat, a Matlab matrix containing the complex-valued, m-mode acquisition of flowing liquid

(3) tubeimage.tif, an OCT image showing a larger view of the tube

(4) tubepicture.tif, a light microscopy image of the same flow phantom

2. Optical Coherence Tomography

2.1. **What is OCT?.** In optical coherence tomography (OCT), pulses of light are sent towards a tissue [4]. By detecting reflected light from the tissue and combining the light with that from a reference arm of a given length, the depth of the reflected light pulses can be determined. By scanning across a tissue and building up numerous depth reflectivity profiles, an image can be generated.

OCT is often said to be the analogue of ultrasound. In ultrasound, sound pulses are sent towards tissue, and an image is built up by calculating the depth and lateral location of each returning pulse. In this manner, OCT is very similar to ultrasound. The difference is that OCT typically uses light of approximately 800 - 1300 nm, while ultrasound uses sound with wavelength on the order of about 1 mm. As a result, OCT has significantly higher resolution (1 μm versus 1 mm). The tradeoff is that OCT only penetrates about 1 mm into tissue, while ultrasound can penetrate many centimeters into tissue.

2.2. **The OCT Image.** Load the tadpolecpx.mat file.

```
load('tadpolecpx.mat')
```

This file is an OCT image of an embryonic tadpole submerged in a solution with polystyrene microspheres. Notice that the dimensions are 512 x 1024. The OCT image was built up by taking a single line scan of 512x1 pixels, and then scanning that at 512 different points to build a picture.

Notice that the OCT data is complex. In fact, this is similar to ultrasound. The light has both a magnitude and phase associated with it. The phase will be important when we talk about doppler. For now, let's look a magnitude image.

```
depth_tadpole=linspace(1,size(tadpolecpx,1)-1,size(tadpolecpx,1))*2/512;
width_tadpole=linspace(1,size(tadpolecpx,2)-1,size(tadpolecpx,2))*2/1024;
imagesc(width_tadpole,depth_tadpole,abs(tadpolecpx))
xlabel('Length [mm]');ylabel('Depth [mm]');
colormap gray;
```

The typical way that this magnitude is displayed is in logarithmic scaling:

```
imagesc(width_tadpole,depth_tadpole,abs(tadpolecpx))
xlabel('Length [mm]');ylabel('Depth [mm]');
colormap gray;
```

Can you see the tadpole at the bottom of the image, and the small punctate microspheres floating in the fluid above? In summary, standard OCT imaging yields a 2D image that is a reflectometry profile. This type of image is called B-mode. In the next section, we will use m-mode to get velocity information.

3. Measuring speed with OCT

One advantage of OCT is that the signal also yields an estimate of movement (i.e. a velocimetry estimate). By taking advantage of the Doppler effect, we can use the shift in frequency to estimate the speed of a fluid or an object in the direction along the beam of light.

The most common controlled setting to test OCT-based velocity measurements is called a flow phantom. A flow phantom is an experimental setup where we know in advance what the flow profile of a fluid is. Here, we pump a fluid full of beads (beads used to scatter light) through a small capillary tube. Because we can control the flow rate, we also know the flow profile in advance.

Take a look at the two tiff files. tubepicture.tif shows the rectangular capillary tube that is being imaged. It is an en face plane. The red line shows the spatial extent of scanning of tubeimage.tif, the OCT image through that cross section. Notice that in the OCT image, you can tell that the tube is significantly tilted. This tilt will allow us to easily calculated a Doppler shift, as a significant portion of flow is in the axial (z-axis) direction. From tubeimage.tif, you can also calculate an angle of flow.

3.1. **Working with complex m-mode data.**

Let's load the complex OCT data of the flow phantom:

```
load('tubecomplex.mat');
```

Note that in contrast to our previous image, this OCT data is acquired not by scanning a beam over an object, but by keeping the beam in place and letting fluid flow by it. It is exactly analogous to m-mode in Ultrasound, and it carries the same name in OCT: m-mode.

Let?s look at the image to double check that it is what we want:

```
imagesc(abs(tubecomplex));
colormap gray;
```

Notice that the tube extends from approximately point 100-260 (in depth). Also notice that the first 100 or so lines (on the left of the image) look a little different than the rest of the image. The image distortion is caused because the galvo scanner moves away from the location of the beam at the beginning of each m-mode frame to get a background scan, which is necessary for reconstructing the image. Thus, the leftmost portion of the image is not actually sampling the same physical region as the rest of the m-mode acquisition. For this reason, we will simply drop the first 96 points, to be left with an even 4000 time points.

```
tubecomplex=tubecomplex(:,97:end);
```

The next thing we want to do is make sure that our axes have the proper labeling. The scan rate is 28000 Hz, and the total depth is 2 mm in air. Now when we display our image, it will have proper units of time (x-axis) and depth (z-axis).

```
linerate=28000;
time=linspace(1,size(tubecomplex,2)-1,size(tubecomplex,2))/linerate;
depth=linspace(1,size(tubecomplex,1)-1,size(tubecomplex,1))*2.49/512;
imagesc(time,depth,abs(tubecomplex))
xlabel('Time [s]');ylabel('Depth [mm]');
colormap gray;
```

3.2. **The naive approach to Doppler.** The first thing we will do to estimate velocity is to calculate a Doppler signal the simplest way, by looking at our change in phase ($d\Theta/dt$) over time. Notice that the complex nature of our signal, i.e. that the signal can be represented as $A(t)e^{i\Theta(t)}$, is what allows us to calculate a Doppler shift at all.

```
dopp_angle=diff(angle(tubecomplex),[],2);
imagesc(dopp_angle);colorbar
```

Although we can clearly see the tube, the flow profile is not totally obvious. This is because taking the derivative of the angle is the noisiest way to calculate Doppler, as it takes an instantaneous measurement. Can you use this data to calculate a Doppler velocity?

4. Calculating Power Spectrum and Estimating Doppler Shift

Now that we have done all we can with the simple complex data, we will focus on a time analysis of a single point in time. Point 180 (in depth) is about in the center, so we will focus on that. For simplicity, we will define a new variable that is just the time trace over that line:

```
line=180;
timeseries=tubecomplex(line,:);
plot(time,abs(timeseries))
```

Notice that the signal fluctuates up and down in a somewhat random fashion. Next, we will look at the complex signal. Because it can be difficult to digest, we will only look at a short portion of the complex data.

```
plot(timeseries(100:150))
```

What are the axes of this plot? Adjacent points in time are connected to one another. Can you see any trends amongst these points?

In order to better characterize the periodicities in the system, we will look at the power spectrum. The power spectrum will display frequencies between $-f/2$ and $f/2$, where f is determined by the linerate of the camera.

```
frequencybin=linspace(-linerate/2,linerate/2,length(timeseries));
plot(frequencybin,fftshift(10*log10(abs(fft(timeseries)))))
xlabel('Frequency [Hz]');ylabel('Power [dB]');
```

Notice that the power spectrum is centered at a non-zero location. What is the reason for this? The center frequency appears to be about 1690 Hz. We will use this a second method of calculating our Doppler velocity:

```
lambda=1.3*10^-3;
fd=1690;
dopp_ps=fd*lambda/2;
```

Our flow tube should have a midline velocity of about 2.3 mm/s. Is this consistent with our Doppler measurement? What other information do we need to calculate the total speed? Can you get this information from any of the other two files you were provided?

We took the power spectrum using the original complex signal. How would it differ if we took the power spectrum of our intensity signal?

5. Dynamic Light Scattering and Autocorrelation Analysis

We have looked at our power spectrum, and we will now look at extracting similar information from the autocorrelation of the signal. We will use the xcorr function in Matlab. It is worthwhile to look the documentation for this function, as there are some subtleties to using it.

Recall that the autocorrelation can be defined in multiple ways. Here, we will use the definition given by Lee et al., that is: $g(\tau) = \langle S(t)S^*(t+\tau) > / < |S(t)|^2 \rangle$. (Note that often times, the deviation of the signal from its mean, i.e. $\delta S(t) = S(t) - \langle S \rangle$, is used instead).

```
ac=xcorr(timeseries)/(sum(timeseries.*conj(timeseries)));
ac=ac(length(timeseries):end);
plot(time,abs(ac))
```

Here, we have plotted $|g(\tau)|$, because the autocorrelation may not always be purely real. Here we have truncated half of the values of our autocorrelation, but if we plotted all of them we would see that the function is symmetric about the point $\tau = 0$. This is because for a causal signal, shifting the signal forwards or backwards in time is equivalent. Additionally, note that it is maximal about $\tau = 0$ and then decays towards zero. We have defined our signal, in fact, to be unity at zero time lag.

Zoom into the section for $\tau < 0.002$ s. Quantifying the magnitude of decay in this area will give us our total diffusivity / speed. Before we head there, though, let?s take a look at the fully complex autocorrelation function, again just for a short segment:

```
plot(ac(1:30))
```

Note here again that the correlation function rotates in complex space. Does this remind you of the original signal? How are the two related? Notice, also, that by definition the signal is completely real at $\tau = 0$. That means in order to estimate the rate of rotation, we just need to calculate the angle of the first non-zero time lag:

```
dopp_kasai=angle(ac(2))/(4*pi/lambda/linerate);
disp(dopp_kasai)
```

This is just the Kasai estimator for velocity. A couple asides: First, Kasai is typically computed on just a few time points, just 8 or so, not the 4000 that we have done here. Additionally, sometimes the difference between the first and second non-zero time lag can be dominated by noise, so often times the $\tau = 0$ data point is dropped. Thus,

more robustly we would use $d\Theta/dt$, where Θ is now the angle of our autocorrelation function, not our original signal.

We will now proceed to quantify the rate of decorrelation of our signal. Although we could fit our entire autocorrelation set to the full function defined by Lee et al., instead we will assume that our flow is fast compared to our diffusion and that our flow dominates decorrelation [5]. This will simplify fitting for this toy problem.

In Matlab, we need to define our fitting function, which here will be a simple Gaussian of the form $Ae^{-\gamma^2\tau^2}$. Once we define our function, we need to put in initial guesses for each of the parameters, and then run non-linear least squares, typically using the Levenberg-Marquardt algorithm.

```
modelFun = @(b,time) b(1).*exp(-b(2)^2*time.^2);
start = [1 1000];
beta=nlinfit(time(1:50),abs(ac(2:51)),modelFun,start);
```

We get back a vector, here called beta, which contains the optimized parameters. Now let?s plot it with our model function to see if it looks at all like our data:

```
fittime=linspace(time(1),time(100),1000);
fitmodel=beta(1)*exp(-beta(2)^2*fittime.^2);
plot(time(1:100),abs(ac(2:101)),fittime,fitmodel)
```

Good, now that we have an estimate for $\gamma = v/w$, with an estimate for w (6.7 μm), we can calculate our total speed profile.

```
wx=6.7*10^-3;
speed_DLS=beta(2)*wx;
disp(speed_DLS)
```

Does this make sense based on our prior estimates? What might be different? Let's repeat this procedure for the entire tube plus some additional bounds, from lines 50-300.

```
lstart=50;
lend=300;
speed=zeros(lend-lstart+1,1);
speeddop=zeros(lend-lstart+1,1);
for line1=lstart:lend
     ac1=xcorr(tubecomplex(line1,:))/(sum(tubecomplex(line1,:).*conj(tubecomplex(line1,:
    beta1=nlinfit(time(1:50),abs(ac1(4001:4050)),modelFun,start);
    speed(line1-lstart+1)=beta1(2);
    speeddop(line1-lstart+1)=angle(ac1(4001));
```

```
end
speed=speed*wx;
speeddop=speeddop*lambda/(4*pi/linerate);
figure(1);plot(speed)
title('DLS Measurement');xlabel('Position in Tube [pix]');ylabel('DLS Speed [mm/s]
figure(2);plot(speeddop)
title('Doppler Measurement');xlabel('Position in Tube [pix]');ylabel('Doppler Spee
```

Look at the flow profile as gathered by Doppler versus DLS. Does it agree qualitatively with what you expect? Quantitatively? What prior information did we use that could potentially be incorrect? Can you try to redo the fitting, this time incorporating a diffusive term? Does this data look better?

5.1. **Improving DLS.** In the analysis here, the flow profile that we calculate using DLS does not have great experimental agreement with what we know the flow in our phantom to be. We see that our measurement shows where there is flow, but not a nice parabolic flow profile that we expect. The reason is that our signal does not obey some of the implicit assumptions we made in deriving our expression for the autocorrelation function.

We worked on a few ways to improve flow estimation. The analysis is a bit more complicated and numerically intensive, but it works. If you'd like to see our results, please look up our papers [2,3].

6. Summary

In this tutorial, we have looked at a time varying, complex OCT signal. We calculated Doppler using three methods: (1) the change in the complex angle from scan to scan, (2) the center frequency of our Fourier power spectrum, and (3) by the Kasai method (autocorrelation). Lastly, we used DLS-OCT to estimate total speed of our signal, also by looking at the autocorrelation function.

There are many other subtleties to processing this data, but the hope is that this document gives an intuitive understanding of how the data is processed in its simplest form.

7. References

[1] Huang, BK and MA Choma. "Microscale imaging of cilia-driven fluid flow." Cell. and Mol. Life Sci. (epub 2014)

[2] Huang, BK and MA Choma. "Resolving directional ambiguity in dynamic light scattering-based transverse motion velocimetry in optical coherence tomography." Opt. Lett. 39, 3 (2014)

[3] Huang, BK, UA Gamm, V Bhandari, MK Khokha, and MA Choma, "Three-dimensional, three-vector-component velocimetry of cilia-driven fluid flow using correlation-based approaches in optical coherence tomography." Biomed. Opt. Exp. 6, 9 (2015)

[4] https://en.wikipedia.org/wiki/Optical_coherence_tomography

[5] Lee, Jonghwan, et al. "Dynamic light scattering optical coherence tomography." Optics express 20.20 (2012): 22262-22277.