Time Delay Estimation and Acoustic Source Localization

Javier Ribera Prat

***Abstract*—Time Delay Estimation (TDE) is a widely studied research topic which has many applications, specially in target localization and tracking. This article presents a MATLAB implementation and comparison of some algorithms used to do so, with emphasis on their impact on accuracy of the results in an underwater environment. A handy GUI was also developed to assist in visual comparison of algorithms and preprocessing filters. Finally, a simple crossing-lines method was used to try to localize the target. The considered case-study has been a recording from PMRF, an instrumented US Navy testing range located off the island of Kauai, Hawaii. The raw data was collected from seven bottom mounted hydrophones in deep water approximately 45 km northwest of Kauai. Targets were always minke whales.**

***Index Terms*—TDE, TDOA, localization, tracking, MATLAB, minke whale.**

I. Introduction

A. Motivation

THIS topic was proposed by Ludwig Houégnigan from the *Laboratori d'Aplicacions Bioacústiques* at *Universitat Politècnica de Catalunya* so it can be useful for their research areas.

The goal is to obtain a solid knowledge of the convenience of some methods used to track the position of cetaceans underwater. Human activities that disrupt marine animal environments such as ships often end up with collisions with cetaceans [1]. This leads to injuries and even death for the animals and a danger for sea navigation [2]. Not only vessels in the high seas but also coast cruises face these issues, especially where both whale and ship densities are concentrated. Unfortunately, many incidents of ship strike around the coast go unnoticed or unreported, and this makes it difficult to understand the scope of the problem [3].

Thus, a whale anti-collision system to warn ships in an area about whale locations is required. Moreover, spotting whales positions is also useful for scientific applications such as animal census, behaviour studies, environmental conservation, etc.

B. Scope of the project

The following algorithms to compute the TDE have been implemented in MATLAB[1]:

- Cross-correlation (CC)
- Generalized cross-correlation Phase Transform (GCC-PHAT)
- GCC Smoothed Coherence Transform (GCC-SCOT)
- Adaptive Least Mean Squares (LMS)
- Adaptive Eigenvalue Decomposition (AED)

Although, direct use of these algorithms do not yield good results. Hence, some preconditioning methods must be applied previously to the hydrophones recordings. The following filters have also been implemented:

- Pass-band filter
- Percentile noise removal
- Spectral substraction
- Time gain normalization
- Teager-Kaiser

In addition, two handy straight-forward GUIs haven been built in order to rapidly compare the algorithms and assess its pros and cons on each kind of signal.

The first GUI, named "Minke whales sounds TDE" [4] allows to visually:

- Import a pair of raw *.wav* signals
- Select a time segment of this signals in minutes and seconds
- Plot their time evolution or spectrogram
- Apply any combination of the preprocessing methods
- Visualize the frequency and phase response of the pass-band filter
- Choose the desired TDE algorithms to assess
- Plot the resulting (G)CC or in case of LMS, the estimated filter or its error
- Compute the absolute or relative TDE error in both samples and seconds
- Visualize the 3D disposition of the underwater sensors

The second GUI, named "Localization" [5] allows to visually:

- Import precomputed delays from a *.mat* file
- Switch between the recorded events inside this file
- Select one of the seven microphones as reference
- Load the delays from the imported file regarding event and reference selected, or insert custom delays
- Fix the desired speed of sound
- Choose the desired TDE algorithms to assess
- Plot the TDOA hyperbolas for all the microphones

II. Theoretical background

A. Chosen method

To estimate the localization of sources, beamforming technique is dismissed in favour of time-delay of arrival (TDOA) method. This is the only feasible method for this case, as

[1]Except simple cross-correlation, for which the simple built-in MATLAB function *xcorr* has been used.

beamforming cannot be used when distance between sensors is large. In addition, it is computationally cheaper and uses time signal processing and estimation theory.

TDOA should not be confused with TDE, although here both terms will be used with no distintion. TDE is in general the estimation of the delay of two signals, and can be split in two categories: TOA and TDOA. The former deals with the time difference from the emission of a known signal from the transmitter of a device until the reflection of this signal on the target comes back at the receiver. On the other hand, TDOA tries to estimate the delay of an incoming signal in two separated receivers. This signals is solely produced by the target. As the receivers (in our case hydrophones) are at different positions, the signal that impinges on one sensor would be a shifted version of the one at the other sensor. In this article we will focus on the second case: TDOA. Passive (just listening) is preferable in front of active approaches such as SONAR or beamforming techniques, because of ecological and technical aspects: the microphones used in the available data were separated many kilometers away from each other.

B. Cons to face

The passive TDOA approach and the underwater environment imply some inconvenients that must be taken into consideration:

1) The incoming signal is totally unknown, in contrast of in active (TOA) approach, where a simple matched filter would be optimum when the reference signal is produced by us. In our case, TDOA can do no presumption of the waveform, excluding the frequency range. Minke whales usally emit sounds from 1 to 12 kHz.
2) Signal to noise ratio (SNR) is very low. This will be the main limitation and that's the reason of the emphasis on the prefiltering methods for noise reduction.
3) Correlated noise, as the main contribution to it is not white noise, but strong interferences from other animals, devices and water currents.
4) Reflections on the water surface, mainly. This can make the algorithm to get confused by them and could return a wrong result. Reflections on the seafloor are negligible, as hydrophones are usually placed very near to the ground.
5) The recordings will not be just a shifted version of the other ones. Noise, reflections and interferences will introduce huge bias and variance in the estimation if not treated correctly.
6) Sources (whales) will not be still, so the delay will not be a constant in the long-run.

C. Signal model

As stated before, the overall results should take into consideration far reflections on the ocean surface. A simple scheme is shown in Figure 1, where flat ground is assumed.

Received signal could be expressed in general as

$$r_1[n] = \mathbf{h_1}[n]^T \mathbf{s}[n] + w_1[n] \tag{1}$$

$$r_2[n] = \mathbf{h_2}[n]^T \mathbf{s}[n] + w_2[n] \tag{2}$$

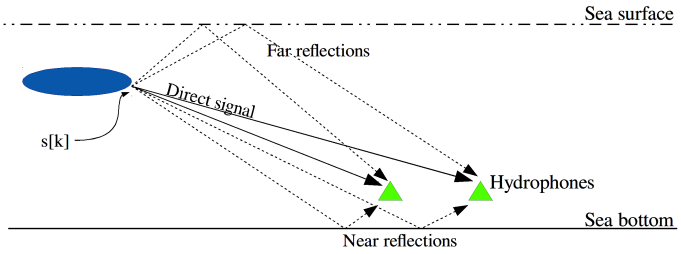

Fig. 1. Reflections in an underwater environment

In an ideal case, the impulse response of the channel $\mathbf{h_j}[n]$ would be a Kronecker delta at the delay position, so $\mathbf{h_j}[n] = [0\ 0\ ...\ \alpha\ ...\ 0\ 0]$, but in general, as indicated in the fifth con to face, distorsion and reflections can make the impulse response of the channel to be any kind [6].

$$\mathbf{h_j}[n] = [h_0\ \ h_1\ \ ...\ \ h_{L-1}] \tag{3}$$

III. TDE ALGORITHMS

In order to get a TDE, some algorithms were developed and a brief explanation of them is shown below:

A. Cross-correlation

The cross-correlation function is very simple, so the built-in MATLAB function *xcorr* was used only in this case. For a pair of finite sequences $x_1[n]$ and $x_2[k]$, the maximum of its cross-correlation function $R_{x_1[n]x_2[n]}[m]$ gives the CC TDE:

$$\tau = \arg\max_m R_{x_1[n]x_2[n]}[m] \tag{4}$$

As MATLAB function *xcorr* returns the result shifted so all values are at positive indexes, the TDE is the difference in samples from the maximum to the middle of the sequence.

This method is the simplest and most straight-forward, but in our scenario it gives the poorest estimations. The code implementation can be seen at file *delay_xcorr.m* [7].

B. Generalized cross-correlation

The generalized cross-correlation (GCC) is a generalization of the cross-correlation in the frequency domain. It is computed as follows:

$$R_{GCC_{x_1[n]x_2[n]}}[m] = \text{IFFT}_M\{\Phi[k] S_{x_1x_2}[k]\}[m] \tag{5}$$

Where $S_{x_1x_2}$ is the cross-spectrum of the input signals given by $E\{X_1[k]X_2^*[k]\}$, but for a finite-length sequence it is approximated as $X_1[k]X_2^*[k]$, where $X_j[k]$ denotes the discrete Fourier transform (DFT) of the input signal $x_j[n]$ and $(\cdot)^*$ the complex conjugate operator.

The function $\Phi[k]$ is the weighting filter. Depending of the chosen weighting filter, the GCC behaves diferent. As seen, the idea is to compute a cross-correlation with some frequency filtering. With the weighting filter $\Phi[k] = 1$, GCC returns the classical cross-correlation.

The two implemented weighting filters have been Phase Transform (GCC-PHAT) and Smoothed Coherence Transform (GCC-SCOT). The former makes use of only the phase of the

signals by using as a weighting filter $\Phi_{\text{PHAT}}[k] = \frac{1}{|S_{x_1x_2}[k]|}$. This method, though, has not yield good results in this case-study. The latter, GCC-SCOT, uses $\Phi_{\text{SCOT}}[k] = \frac{1}{\sqrt{X_1[k]X_2^*[k]}}$. The SCOT method returned much better results than PHAT. Their MATLAB implementations are in the file *gcc.m* [8].

The TDE with GCC is computed the same way as in the classical CC: taking the maximum value, as can bee seen in the file *delay_gcc.m* [9].

C. Least Mean Squares

The idea of the Least Mean Squares method is to try to synthesize a FIR filter that would represent the channel response between a pair of signals. Without taking into consideration the nonideality of the channel, the TDE would be computed as the maximum lag of such estimated filter. It is built by adaptively trying to minimize the Mean Squared Error (MSE) $E\{|\mathbf{e}[n]|^2\}$ between one input signal and the output of the filter, whose input is the other signal. Figure 2 shows the overall scheme. As we have a finite-length sequence available, the MSE is approximated as the instantaneous squared of the error, $|e[n]|^2$. Minimizing $|\mathbf{e}[n]|^2$ with respect to the filter yields to the expression

$$\mathbf{h}'[n+1] = \mathbf{h}[n] + 2\mu\mathbf{e}[n]\mathbf{x_1}[n] \tag{6}$$

where μ is the stepsize of the algorithm, which determines the speed of the convergence to a null error (in mean), but also the stability of the filter. It can be shown that an upper stability bound is $\frac{2}{\lambda_{max}}$, where λ_{max} is the maximum eigenvalue of the cross-correlation function between both signals. Instead of using a value near this bound, an adaptive solution for the stepsize μ has been used. This solution depends on the input power and β, a smoothing parameter. However, this should not be a problem if we use Time Gain Normalization preconditioning. The smoothing parameter is updated at each time instant as

$$\mu[n] = \frac{1-\beta}{\sigma^2[n]} \tag{7}$$

The instantaneous input power is estimated as

$$\sigma^2[n] = \beta\sigma^2[n-1] + (1-\beta)|x_2[n]|^2 \tag{8}$$

In addition, at each step the filter is normalized to an unitary filter to avoid numerical instability:

$$\mathbf{h}[n] = \frac{\mathbf{h}'[n]}{||\mathbf{h}'[n]||} \tag{9}$$

The overall code can be seen in the file *delay_lms.m* [10].

D. Adaptive Eigenvalue Decomposition

Adaptive eigenvalue decomposition (AED) algorithm was proposed to deal with TDE in reverberant environments. Unlike the crosscorrelation stated above, this algorithm first identifies the channel impulse responses from the source to the two sensors. The delay estimate is then determined by finding the direct paths from the two measured impulse responses. To do so, we observe that

$$\begin{aligned} x_2[k] * g_1 &= s[k] * g_2 * g_1 \\ &= x_1[k] * g_2 \end{aligned} \tag{10}$$

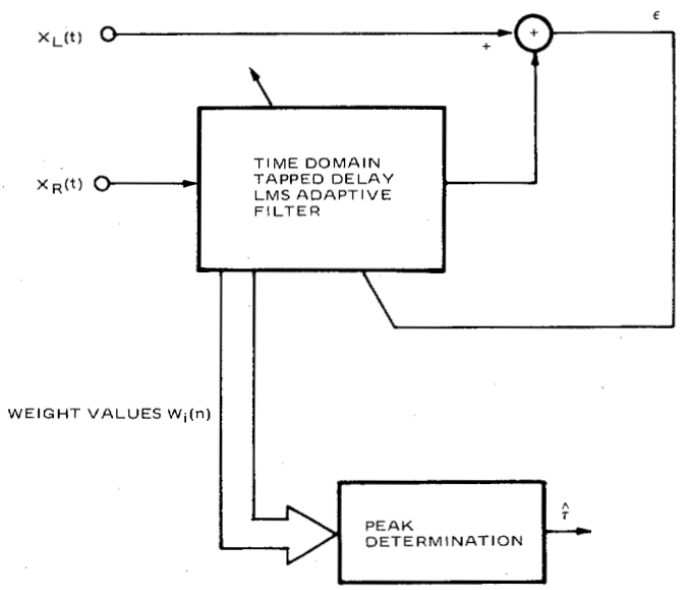

Fig. 2. TDE using an LMS filter

Then, we can state that $R_u = 0$. This implies that vector u which consists of two impulse responses is in the null space of R. More specifically, u is the eigenvector of R corresponding to the eigenvalue 0 ($u[k] = g_2[k] - g_1[k]$). We observe that u can be uniquely determined if the following two conditions hold [6], [11]:

- g_1 and g_2 do not share any common zeros.
- The covariance matrix R is full rank.

In case of noise, the covariance matrix will regularize. That will provoke that the algorithm stated above become inoperable because the lack of a 0 eigenvalue. To solve that we have implemented an adaptive way to calculate the eigenvector with minimum eigenvalue:

$$u[k+1] = \frac{u[k] - \mu e[k]x[k]}{||u[k]||} \tag{11}$$

where μ is the positive adaptation step.

With the identified impulse responses g_1 and g_2, the time delay estimate is determined as the difference between two direct paths. That is,

$$T_{\text{AED}} = \arg\max_T |g_2| - |g_1| \tag{12}$$

Finally, the noise will not be a problem with the time delay estimation because we only want to find the two direct path and not the whole impulse response. Then, AED is the best of the stated algorithms in front of noise and reverberation.

IV. Preconditioning filters

Prior to the delay detection, it is not only useful, but also necessary, to preprocess the filter in some way. Usually, TDE block gets confused by noise, reflections and variance in the power throughout time. Each preconditioning method tries to solve some of these inconvenience.

A. Time Gain Normalization

Time Gain Normalization algorithm tries to fit the average amplitude to a fixed level, by calculating the long-term average level $\overline{x}[n]$ using a smoothing time averaging parameter α and a L^P norm in the following way:

$$\overline{x}[n] = (\alpha \overline{x}^P[n-1] + (1-\alpha)|\overline{x}[n]|^P)^{\frac{1}{P}} \quad (13)$$

Then, the output signal with a normalized gain of r^2 can be computed as

$$x_{norm}[n] = r\frac{x[n]}{\overline{x}[n-1]} \quad (14)$$

The implementation of Time Gain Normalization can be seen in the file *time_gain.m* [12].

B. Spectral Substraction

Spectral subtraction is a method for restoration of the power spectrum or the magnitude spectrum of a signal observed in additive noise, through subtraction of an estimate of the average noise spectrum from the noisy signal spectrum. The noise spectrum is usually estimated, and then updated from the periods when the signal is absent and only the noise is present. We did the assumption that the noise is a stationary or a slowly varying process, and that the noise spectrum does not change significantly between the update periods [13].

For restoration of time-domain signals, an estimate of the instantaneous magnitude spectrum is combined with the phase of the noisy signal, and then transformed via an inverse discrete Fourier transform to the time domain.

In terms of computational complexity, spectral subtraction is relatively inexpensive. The magnitude and power spectrum are non-negative variables, and any negative estimates of these variables should be mapped into non-negative values. This nonlinear rectification process provoke a distortion in the cleaned signal. The processing distortion becomes more noticeable as the signal-to-noise ratio decreases. To diminish the distortion due to the harder spectrum attenuation, we have added the spectral floor (β) to limit the subtraction and force some residual noise at the output.

The algorithms computes the clean signal as follows:

$$|X[k]|^2 = \begin{cases} |Y[k]|^2 - \alpha|N[k]|^2 & \text{if } |X[k]|^2 > \beta|N[k]|^2 \\ \beta|N[k]|^2 & \text{otherwise} \end{cases} \quad (15)$$

Where $X[k]$ is the spectrum of the clean signal, $Y[k]$ is the noisy signal, $N[k]$ is the estimated noise spectrum, α is the oversubtraction parameter and β is the spectral floor.

The implementation of Spectral Substraction can be seen in the file *spectralsubstraction.m* [14].

C. Percentile Noise Removal

Percentile Noise Removal (PNR) is a method for restoration of the power spectrum through substraction of an estimate of the percentile nth of the noise spectrum from the noisy signal spectrum. The noise spectrum is usually estimated like in the spectral subtraction method. Then, we apply the percentile function to this noise spectrum. All of this steps are updated from the periods when the signal is absent and only the noise is present. Finally, we subtract the product of the percentile estimation with a constant to the Spectrogram of the signal. The chosen value in many of our simulations was 5 after trial and error.

We did the assumption that the noise is a stationary or a slowly varying process, and that the noise spectrum does not change significantly between the update periods. The magnitude and power spectrum are non-negative variables, and any negative estimates of these variables should be mapped into 0.

The proposed percentile value is 90. It has been elected because it is the optimum value in our case, where the curve of spectrogram-percentile is flattest. Note that if we take nth=50, the Percentile Noise Removal is very similar to Spectral Subtraction, since the nth=50 means taking the median of the estimated noise spectrum and the Spectral Subtraction uses the average of the noise spectrum.

The PNR algorithm is comptued as follows:

$$S'(t,f) = max\{S(t,f) - cN, 0\} \quad (16)$$

where $S'(t,f)$ is the spectrogram of the clean signal, $S(t,f)$ is the spectrogram of the noisy signal, c is the oversubtraction parameter and N is the chosen percentile of the estimated noise spectrogram.

The implementation of Percentile Noise Removal can be seen in the file *percentile.m* [15].

TABLE I
POSITION OF HYDROPHONES

hydrophone #	X[m]	Y[m]	Z[m]	filename
7 #	-6129	9784	-4750	27Apr09_074921_NN_p7.wav
6 #	-6183	-4874	-4650	27Apr09_074921_NN_p6.wav
5 #	-6163	-12402	-4600	27Apr09_074921_NN_p5.wav
4 #	6865	12844	-4750	27Apr09_074921_NN_p4.wav
3 #	6520	5240	-4650	27Apr09_074921_NN_p3.wav
2 #	6635	-2132	-4500	27Apr09_074921_NN_p2.wav
1 #	6566	-9617	-4500	27Apr09_074921_NN_p1.wav

V. EXPERIMENTAL SETUP

A. Origin of the data

The data and information is coming from PMRF, an instrumented US Navy testing range located off the island of Kauai, Hawaii. Data were collected from seven bottom mounted hydrophones (4 to 5 m off the seafloor) in deep water (nominally 4,600 meters) approximately 45 km northwest of Kauai. The relative location of the hydrophones, their designations and filenames for the data files are as shown in Table I. Our targets were always minke whales.

The dataset is from the 27th of April 2009 at approximately 12:00 Hawaiian standard time. Thirty minutes of data from each hydrophone are provided as three files (approx. 10 minutes per file). The sampling rate of the recordings (Windows PCM .wav format) is $Fs = 96$ kHz with 16 bits resolution. The data are sampled simultaneously, so sample N from one file is

the same relative time as sample N of a second file with the same filename. The file format is .wav in little-endian format.

The test signals used to assess the proper implementation of the algorithms were two chirps, two pure tones and white noise delayed a desired number of samples. The implementation of their creation can be seen in file *init_test_signals.m* [16].

B. Procedure

First, two signals are loaded and cut to focus at an interesting event. We've used Audacity to visually discover the minke whale sounds. The length of the window to cut the signals must be at least $TDOA_{expected} + L_{signal}$.

One one hand, the worst case for $TDOA_{expected}$ is when the source is aligned with both microphones. Then, the maximum possible delay between to pair of microphones will be $\frac{\text{distance}}{c}$, where c is the speed of sound underwater. c is always considered constant, altough better approximations could have been used, taking into consideration temperature, salinity and pressure (depth) [17]. On the other hand, a minke whale sound (L_{signal}) lasts around 6 seconds. Hence, our window length will be very large due to the huge distance between sensors. In our case, $TDOA_{max}$, corresponds to the delay between the furthest microphones. Such value has been computed using the code in the file *maxdelay.m* [18]. The huge distance between sensors (many kilometers) leads us to a window length of 20-25 s. This will a problem when we run the adaptive algorithms.

Then, after having cut the signal, we pre-processed it with a noise removal algorithm. Finally, we proceed to the TDE between the two signals. Knowing the true delay, we only have to do a ground truth assess. The true delay is easily visualized using Audacity. Then, we record the results as relative and absolute error in samples.

The comparison between TDE algorithms and preprocessing methods has been carried out using sensor 1 and 2 and event #1 (described below). The last step, localization, which is discussed later, have been tried between all combination of sensors and event #1.

C. Some events

We focused on two events:

- First event: 2:20-2:45. Only noise, no other interference than the constant tone at much higher frequencies.
- Second event: 8:40-9:00. It has interferences from other animals (dolphins and whales). For this reason, it's more critical and more important to reduce strongly the noise and interference. The results of the simulation depend heavily on the algorithms of noise reduction and TDOA used.

VI. Results

In this section, the results of the successfuly implemented methods that perform better will be shown. That means the combination of preconditioning filters and the TDE algorithm and their parameters.

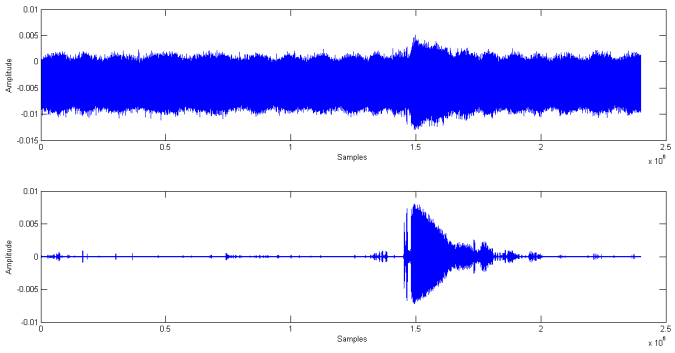

Fig. 3. Above: Waveform of sensor 1 recording without pre-processing. Below: Waveform of sensor 1 recording after filter + PNR

A. Prefiltering methods

First, as it is a crucial step, the prefiltering methods are assessed individually.

1) PNR: In the Figure 3, at the top we can see the noisy signal and at the bottom the signal after being processed with Percentile Noise removal. We can observe that the algorithm has deleted almost all of the noise. However, the signal has decreased its level.

After assessing the improvement, we can state that the SNR before the pre-processing was 3.06 dB and after applying the band pass filter and PNR, it increased up to 39.50 dB. This leads to a good noise reduction algorithm. Computing the entropy using the built-in MATLAB funcion *entropy()*, the result is that after the pre-processing the entropy value increases. This is not what we would have expected a priori. The reason is that we have subtracted the correlated noise and pre-whitened the signal. Therefore, we have added a little randomness in the final signal. That is why entropy is not always a good assessment method. However, visually it enhances a lot the scenario and the TDE performs with much more accuracy.

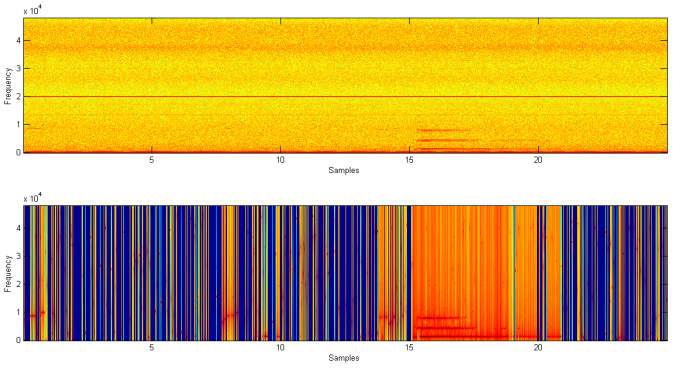

Fig. 4. Above: Spectrogram of sensor 1 recording without pre-processing. Below: Spectrogram of sensor 1 recording after filter + PNR

At Figure 4 we can see the noisy spectrogram and the "clean" spectrogram after being processed with PNR. We can observe that the algorithm has modified completely the Spectrogram. After a lot of work done in the implementation of the

algorithm, we could not be able to improve the spectrogram. A missing issue about this algorithm is that there was not enough time to find the optimal parameters: window length and window length shift. This might have improved the visual aspect of the spectrogram. Furthermore, the distance between sensors and the computational complexity of the algorithm made the work harder. However, after simple visual inspection, it's easier to identify the whale sound in the spectrogram, as the SNR of the clean signal has increased.

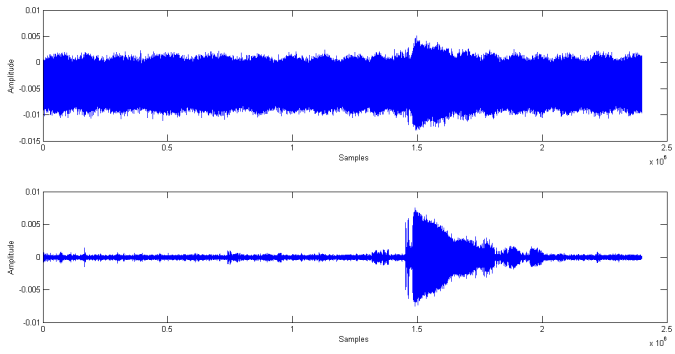

Fig. 5. Above: Waveform of sensor 1 recording without pre-processing. Below: Waveform of sensor 1 recording after filter + Spectral Substraction

2) Spectral Substraction: Figure 5 shows the noisy signal and the signal after being processed with Spectral Subtraction. We can observe that the algorithm has deleted most of the noise although the signal has decreased it level a little bit. It doesn't decrease all the noise because of the parameter β, which fixes a noise floor. After assessing the improvement, we can state that the SNR before the pre-processing was $3.0583\,\text{dB}$ and after applying the band pass filter + SS, it increased up to $33.88\,\text{dB}$. The same reasoning as in PNR case can be made. The enhancement is a little bit lower but still very high. This is because SS is a particularization of PNR when the percentile value is 50.

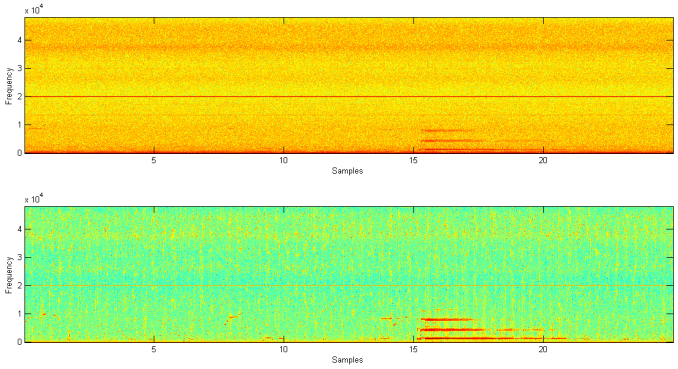

Fig. 6. Above: Spectrogram of sensor 1 recording without pre-processing. Below: Spectrogram of sensor 1 recording after filter + Spectral Substraction

Figure 6 shows the noisy signal and the signal after being processed with Spectral Subtraction. We can observe that the algorithm has a much more good-looking Spectrogram than in PNR method, but the SNR is lower. This could be due to the parameter β, that allows a little noise level in the clean signal. In TDE, the PNR performs better because, although the clean signal has more distortion, it is looking at the peak to find the correlation between sensors. Then, for our application, we will have better results pre-processing with the PNR rather than with Spectral Subtraction Algorithm.

B. TDE

With Time Gain Normalization algorithm, we observe quite good results in Time Delay Estimation. All the situations have a relative error below 1% despite one simulation that has a relative error of 500% (GCC-SCOT between sensors 1-2. at 2nd Event). With Percentile Noise Removal denoising algorithm, the results in Time Delay Estimation outperform. Indeed, all the simulations have a relative error below 0.7%. If we apply the band-pass filter, we keep obtaining the same results. Finally, with Spectral Subtraction algorithm we observe bad results in Time Delay Estimation. Indeed, all the simulations with GCC-PHAT and almost all of them with GCC-SCOT get a relative error of 100%. If we apply the band pass filter, we keep obtaining good results with the cross-correlation (below 1.5% of relative error) and we have improved the simulations with PHAT and SCOT. However, PHAT algorithm keep getting a relative error of 50%. As stated before, the cross-correlation algorithm presents good results in all the possible simulations. The interpretation of this results can be understood as the problem of Time Gain Normalization. This is that the high distance between the sensors leads to a small coherence between the received signals, so it makes the PHAT algorithm weak. About the failure of the SCOT without applying the filter can be explained as the Scot algorithm has taken into account the coherence between the strong interference at $20\,\text{kHz}$ in sensors 3 and 4. Then, applying the filter we eliminate that interference. We sum up saying that PNR is the best noise reduction algorithm we have explained. We had expected this result when getting $39.5\,\text{dB}$ of SNR stated previously.

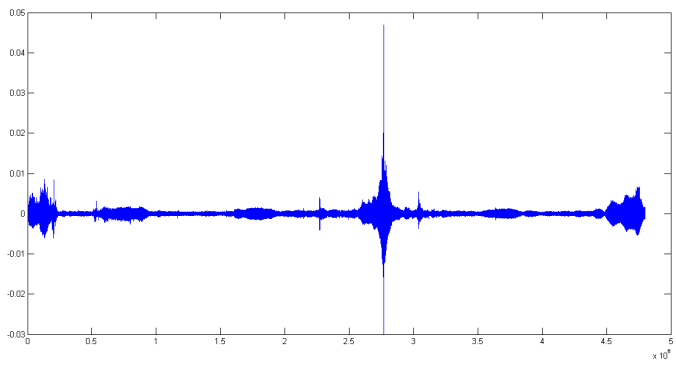

Fig. 7. GCC-SCOT between sensors 1 and 2 at the first event (2:20-2:45) after PNR

1) GCC-SCOT with PNR: Figure 7 shows GCC-SCOT between sensors 1 and 2 at the first minke whale event. As we were expecting after looking at the SNR stated above, the Time Delay Estimation is very accurate. Indeed, we have only a 0.26% of relative error. GCC-SCOT regularly performs

that better. Simulations between sensors 1 and 2, 3, 4 and 5 at the two events and pre-processing with the Percentile Noise Removal algorithm, all of them obtained less than 0.7% of relative error in all the simulations. We have tried the simulations with these algorithms: Cross-correlation, SCOT Generalised Cross-correlation and PHAT Generalised Cross-correlation.

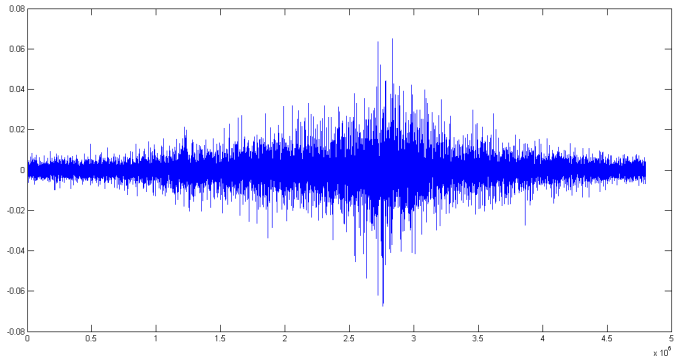

Fig. 8. GCC-PHAT between sensors 1 and 2 at the first event (2:20-2:45) after Spectral Substraction

2) GCC-PHAT with Spectral Substraction: Figure 8 shows GCC-PHAT between sensors 1 and 2 at the first minke whale event. Looking at the plot, we can see that the Time Delay Estimation is quite poor. Indeed, we got relative error of 18%. PHAT algorithm doesn't work well in all the simulations. Due to the high (12-28 km) distance between the sensors, the coherence of the received signals is very low. While SCOT takes that into account, the PHAT algorithm fails in some of these situations. The PNR is the only noise removal algorithm that leads to a good GCC-PHAT results in all the simulations.

VII. Localization

Given a time delay and the known position of the sensors, a loci of possible points of the source gets defined. For each pair of sensors, an hyperbola of possible points is defined. For a given estimated delay of τ_{ij} between two sensors located at (x_i, y_i, z_i) and (x_j, y_j, z_j) and for a constant speed of sound of c, the resulting hyperbola is represented at Figure 9 and its expression is

$$\tau_{ij} = \frac{1}{c}\left(\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} - \sqrt{(x-x_j)^2+(y-y_j)^2+(z-z_j)^2}\right) \quad (17)$$

So, later on, with another pair of microphones, the intersection yields to the source localization.

In our short experiment, for tracking a minke whale on a 2D map, we have developed a rudimentary multilateration system, on the basis of 6 TDOAs only. The localization by itself leads to a non-linear solution. In addition, non-idealities imply a no unique solution, so we get a zone where the source could be located. We have not gone so far, but the estimated position could have been estimated by algorithms such as Crossing Lines or Steered-response power.

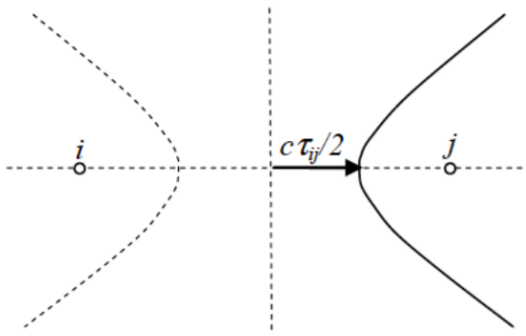

Fig. 9. 2D hyperbola loci defined for a TDOA of τ_{ij} between sensors i and j

We first show the position of the hydrophones on the map, and then plot the hyperbolas corresponding to each pair of TDOA, assuming the speed of sound in water is 1510 m/s [17]. We have computed the TDOA between the sensor 1 and all the other ones, after having pre-processed the signals with PNR and doing the estimation with the Generalised Cross-Correlation Smoothed Coherence Transform.

Then, looking at Figure 10, we have obtained 6 curves that ideally will intersect at one point. In our case, it is quite normal that the curves don't intersect all, but we can see where the whale is approximately, as the curves intersect in a little area. The first 5 curves have a Time Delay Estimation error smaller than 1%. The worst case corresponds to the furthest sensors, 1 and 7, and they are separated 28 km, so our resolution is 280 m. That is why the curves don't match in one point. Nevertheless, we do a rough visual estimation so the whale is at the position x=14000 y=5300 at event #1.

VIII. Graphical User Interface

The Graphical User Interface, built using the GUIDE MATLAB tool, has all the capabilities stated at the Introduction, section B, "Scope of the project".

An screenshot of the same can be seen at Figure 11 using the GCC-SCOT result with Time Gain Normalization. The two loaded signals come from sensors 1 and 2 and are focused at the minke whale sound #1 (a Minke whale sound).

The 3D distribution of the PMRF sensors can be seen and rotated using the option View 3D position of sensors. A screenshot is attached at Figure 12. The corresponding code is in file *plot3Dsensors.m* [19].

The code of the GUI is in file *interactive_TDE.m* [4] and *localization.m* [5].

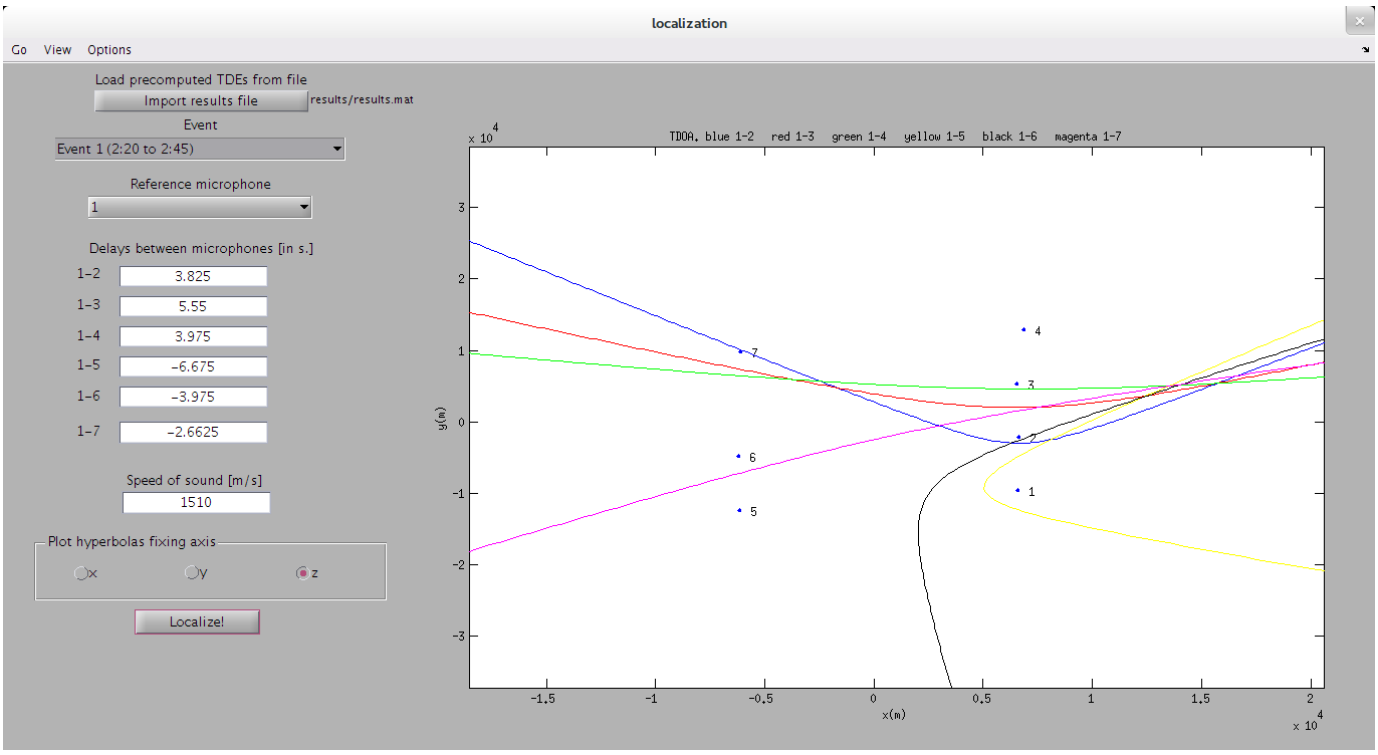

Fig. 10. Localization of the minke whale through multilateratization between sensor 1 and the rest.

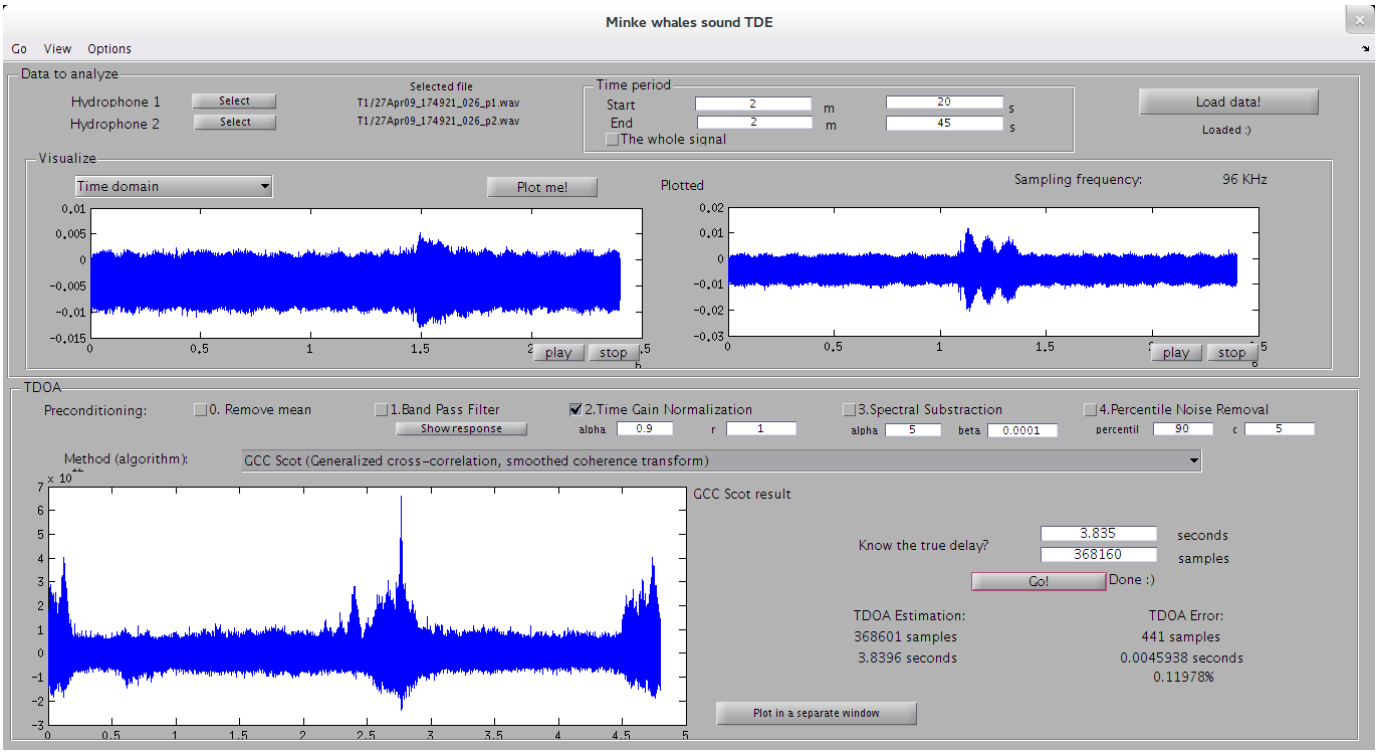

Fig. 11. Graphical User Interface screenshot with a GCC-SCOT estimation using Time Gain Normalization over a Minke Whale sound

IX. Code

All the code of the project is publicy available in a git repository on Bitbucket within the next URL: https://bitbucket.org/javiribera/dsap/src A brief high-level explanation of some files is done below:

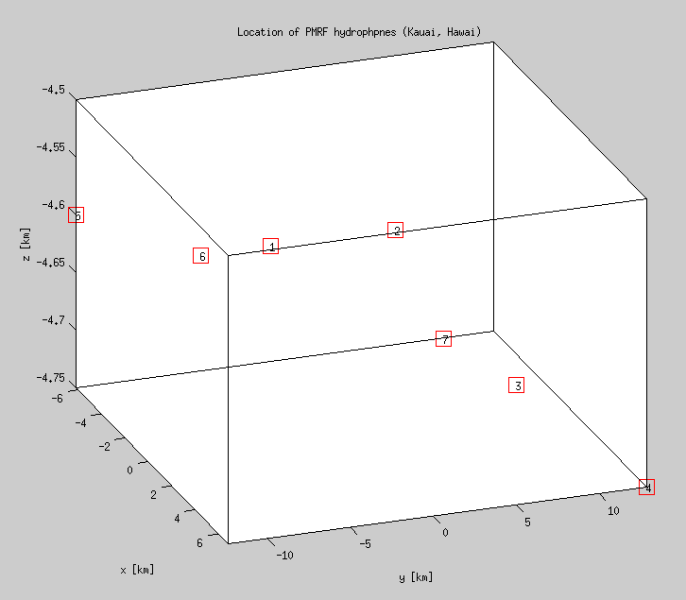

Fig. 12. Underwater distribution of the PMRF hydrophones in kilometers

- *algorithms_TDE* folder
  - *gcc.m* [8] computes the Generalized Cross-Correlation between parameter 1 and 2 using the weighting algorithm passed as third parameter ("cc","phat" or "scot").
  - *delay_gcc.m* [9] bypasses its parameters to *gcc.m* and takes the last maximum index value.
  - *delay_xcorr.m* [7] computes the classical cross-correlation and also takes the last maximum index value.
  - *delay_lms.m* [10] returns the delay between first and second parameter using LMS adaptive method using the fifth parameter (beta) as smoothing parameter. The third (max_expected) and forth (length_signal) parameters are used to know the minimum order or the filter. The last parameter (handles) is used only to be able to plot the results to the GUI.
- *preconditioning* folder
  - *build_filter.m* [20] returns a very high-order (50) filter designed to bandpass filter minke whale sounds (1 and 12 kHz).
  - *filter_passband.m* [21] outputs the input after being filtered by the filter returned by *build_filter.m*.
  - *percentile.m* [15] outputs the input after being processed by the percentile noise removal algorithm.
  - *spectralsubstraction.m* [14] outputs the input processed by the spectral substraction algorithm.
  - *time_gain.m* [12] outputs the input after being processed by the Time Gain Normalization algorithm.
- *GUI* folder
  - *main_GUI.m* [22] is a simple 2-button GUI to choose between the next 2 GUI:
  - *interactive_TDE.m* [4] displays the GUI shown at Figure 11 that has all the capabilities explained at Introduction, section B, "Scope of the project".
  - *localization.m* [5] displays the GUI shown at Figure 9 that has all the capabilities explained at Introduction, section B, "Scope of the project".
- *clean_signal.m* [23] accepts as the second parameter ("preprocessing_methods") a string cell array containing the names of the preconditioning methods to be applied to the signal in the first parameter ("input"). Available options are: "remove_mean", "band_pass", "time_gain", "spectral_substraction", "percentile" and "tk". It applies the selected method in a hardcoded order and outputs the clean signal.
- *clean_signal.m* [23] is a script to create the 7x3 matrix containing the [X,Y,Z] position of the hydrophones in meters.
- *test_signals/init_test_signals.m* [16] is a script that creates three pairs of artificial signals (two tones, two chirps and 2 random noise signals) identical but delayed a known number of seconds, given a sample frequency. It then saves them inside folder *test_signals*as a *.mat* file and as *.wav* files.
- *utils* folder
  - *delay_between_sensors.m* [24] returns the delay in seconds between the sensors provided by the indexes at first and second parameter.
  - *plot3Dsensors.m* [19] shows the 3D disposition of the PMRF hydrophones.
  - *sensors_delay_max.m.m* [18] computes the maximum possible delay in seconds between the furthest sensors.

X. Conclusions

Seen the results, the feasibility of a simple localization has been proved, as long as the proper noise reduction algorithm is chosen. The limited available amount of time didn't allow us to assess more sophisticated algorithms such as LMS and AED. Additionaly, we would have also liked to be able to implement localization algorithms (Crossing Lines and Steered-response power) and to be able to track the evolution of the whale.

Acknowledgment

I would like to thank Ludwig Houégnigan for the proposal of this topic for the project, for providing the data and for deeply assisting in theorical and implementation aspects. Also, for kindly reviewing this report.

References

[1] D. W. Laist, A. R. Knowlton, J. G. Mead, A. S. Collet, and M. Podesta, "Collisions between ships and whales," *Marine mammal Science*, vol. 17(1), pp. 35–75, 2001.

[2] "Cruise ship in whale collision," *TV news*, 2009. [Online]. Available: https://www.youtube.com/watch?v=JuxGpcvZYp0

[3] B. C. S. Network, "Collisions between vessels and whales," *TV news*. [Online]. Available: http://wildwhales.org/conservation/threats/collisions-between-vessels-and-whales/

[4] "interactive_tde.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/GUI/interactive_TDE.m?at=master

[5] "localization.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/GUI/localization.m?at=master
[6] J. Chen, J. Benesty, and Y. A. Huang, "Time delay estimation in room acoustic environments: An overview," *EURASIP Journal on Applied Signal Processing*, vol. 2006, p. 1–19, 2006.
[7] "delay_xcorr.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/algorithms_TDE/delay_xcorr.m?at=master
[8] "gcc.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/algorithms_TDE/gcc.m?at=master
[9] "delay_gcc.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/algorithms_TDE/delay_gcc.m?at=master
[10] "delay_lms.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/algorithms_TDE/delay_lms.m?at=master
[11] J. Benesty, "Adaptive eigenvalue decomposition algorithm for passive acoustic source localization," vol. 107, 2000.
[12] "time_gain.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/preconditioning/time_gain.m?at=master
[13] P. C. Loizou, "Speech enhancement," 2013.
[14] "spectralsubstraction.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/preconditioning/spectralsubstraction.m?at=master
[15] "percentile.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/preconditioning/percentile.m?at=master
[16] "init_test_signals.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/test_signals/init_test_signals.m?at=master
[17] "Speed of sound in sea-water." [Online]. Available: http://resource.npl.co.uk/acoustics/techguides/soundseawater/
[18] "sensors_delay_max.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/utils/sensors_delay_max.m?at=master
[19] "plot3dsensors.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/utils/plot3Dsensors.m?at=master
[20] "build_filter.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/preconditioning/build_filter.m?at=master
[21] "filter_passband.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/preconditioning/filter_passband.m?at=master
[22] "main_gui.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/GUI/main_GUI.m?at=master
[23] "clean_signal.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/clean_signal.m?at=master
[24] "delay_between_sensors.m." [Online]. Available: https://bitbucket.org/javiribera/tde-and-whale-localization/src/c6c367b546486840b8e5f66d51159c7a2b2b81ba/utils/delay_between_sensors.m?at=master