Adaptation of an acoustic propagation model to the parallel architecture of a graphics processor

Emanuel Ey

Dissertation to obtain the
Master Degree in Electrical and Electronics Engineering
Specialization Area: Information and Telecommunication Technologies

2013

Adaptation of an acoustic propagation model to the parallel architecture of a graphics processor

Emanuel Ey
Nr. 23338

Dissertation to obtain the
Master Degree in Electrical and Electronics Engineering
Specialization Area: Information and Telecommunication Technologies

Supervisors: Prof. Dr. Orlando C. Rodríguez
Prof. Dr. Paulo Felisberto

2013

Adaptation of an acoustic propagation model to the parallel architecture of a graphics processor.

Adaptação de um modelo de propagação acústica à arquitectura paralela de um processador gráfico.

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências bibliográficas incluída.

Emanuel Ey Vaz Vieira

Acknowledgement

This work was funded by National Funds through FCT- Foundation for Science and Technology under project SENSOCEAN (PTDC/EEA_ELC/104561/2008).

Contents

Abstract

High performance underwater acoustic models are of great importance for enabling real-time acoustic source tracking, geoacoustic inversion, environmental monitoring and high-frequency underwater communications. Given the parallelizable nature of raytracing, in general, and of the ray superposition algorithm in particular, use of multiple computing units for the development of real-time efficient applications based on ray tracing is becoming of extreme importance.

Alongside the development of multi-core CPUs in recent years, graphics processing units (or GPUs) have gained importance in scientific computing. Desktop GPUs provide vast amounts of processing power at reasonable costs; while their usage may require extensive re-engineering of existing software, they represent an attractive possibility for high performance low-cost underwater acoustic modeling.

OpenCL is a programming standard designed for writing multithreaded code for a variety of different devices, including CPUs and GPUs. It consists of extensions to the C language as well as an API and device driver extensions.

The cTraceo raytracing model was developed at SiPLAB, University of the Algarve[1], in order to predict acoustic pressure and particle velocity in complex waveguides while incorporating backscattering. Intensive testing through comparisons with other models and experimental data has by now shown that cTraceo does indeed produce accurate acoustic field predictions[1, 2, 3].

This work addresses the adaptation of the cTraceo model from a single threaded CPU implementation to an OpenCL parallelized application designed for GPUs. Since the GPU performance of double precision arithmetic was found to be disappointing when compared to single precision, an effort was made to make use of single precision arithmetic in the parallel version of cTraceo. It was found that, for all practical purposes, results are identical to those of the previous implementation, with the advantage of significantly reduced processing times. Performance gains between one and two orders of magnitude were obtained, depending on the configuration of the waveguide and number of traced rays.

Keywords: underwater acoustics, acoustic propagation modeling, high-performance computing, OpenCL, parallel computing.

[1] http://www.siplab.fct.ualg.pt

Resumo

Os modelos de propagação de alto desempenho para acústica submarina são de grande importância para possibilitar o seguimento de fontes acústicas em tempo real, inversão geoacústica, monitorização ambiental, e o desenvolvimento de sistemas de comunicação acústica submarina a alta frequência.

A par do aumento de popularidade dos CPUs de múltiplos núcleos computacionais durante os últimos anos, os processadores gráficos (GPUs) têm vindo a ganhar importância na computação científica. Os GPUs disponibilizam vastos recursos computacionais robustos a custos razoáveis, e embora o seu uso possa requerer alterações extensas ao código existente, representam uma alternativa aliciante para avançar a modelação acústica submarina de alto desempenho a baixo custo.

O OpenCL é um *framework* que permite a escrita de aplicações paralelizadas em sistemas heterogéneos. OpenCL é a abreviatura de "Open Computing Language" e é um *standard* relativamente recente, criado para facilitar a programação de sistemas que contenham uma combinação de CPUs *multicore*, GPUs e outros tipos de processadores. Tem como objectivo permitir escrever um mesmo programa capaz de utilizar os recursos computacionais de dispositivos como *smartphones*, PCs ou mesmo supercomputadores, bastando para tal que esteja presente no sistema uma instalação de OpenCL.

Os CPUs foram desenvolvidos para alternar rapidamente entre tarefas distintas, e como tal uma grande parte das suas estruturas é dedicada a tarefas administrativas, tais como gerir privilégios de processos, memória virtual, mudanças de contexto ou previsão de acções futuras (como *branch prediction* e *prefetching*). Para além destas estruturas administrativas, um CPU também contém uma ou mais Unidades Lógicas e Aritméticas (em Inglês, *Arithmetic Logic Unit* –ALU), nas quais são efectuadas as operações matemáticas que representam a maior parte do tempo de cálculo em algoritmos de aplicações cientificas. Os CPUs modernos têm pequenos números de núcleos (*cores*) complexos com uma frequência de relógio de alguns GHz, adequados a tarefas generalistas. Os GPUs por sua vez têm grandes números de núcleos simples práticamente sem lógica de controlo, sendo constituídos quase só por ALUs, e existindo actualmente no mercado GPUs com 4096 núcleos com frequências a rondar 1 GHz. Devido à simplificação extrema da lógica de controlo, este elevado número de núcleos está sujeito a algumas limitações. Em OpenCL, estes núcleos estão lógicamente associados em

grupos, geralmente de 16, 32 ou 64 núcleos. Estes grupos são designados de *compute units* e são a menor unidade lógica à qual o programador pode explicitamente atribuir tarefas. Dado que todos os núcleos pertencentes a uma compute unit partilham a mesma lógica de controlo, têm necessariamente de executar a mesma instrução em simultâneo. Isto implica que a utilização de um GPU é eficiente apenas se o algoritmo paralelo realizar simultaneamente a mesma operação sobre um grande número de valores de entrada. Felizmente, este é geralmente o caso em aplicações científicas.

Do ponto de vista de uma aplicação, um sistema com suporte OpenCL é constituído por um CPU a agir como anfitrião (*host*) e um ou mais dispositivos de computação (*compute devices*). O anfitrião é responsável por gerir a memória, as operações de entrada e saída, bem como controlar a execução de código nos dispositivos de computação. É nestes dispositivos que é realizado o trabalho da aplicação paralelizada, sendo que o código neles executado é designado por *kernel*.

O TRACEO é um modelo de propagação baseado no traçamento de feixes Gaussianos, desenvolvido no Laboratório de Processamento de Sinal da Universidade do Algarve para modelação de pressão acústica e velocidade de partículas em guias de onda complexos, incluindo efeitos de retro-propagação. O modelo TRACEO foi desenvolvido na linguagem Fortran 77, tendo como objectivo principal a precisão das previsões acústicas.

Dada a natureza paralelizável do traçamento de raios, focou-se o desenvolvimento de uma implementação de alto desempenho do modelo TRACEO adaptado a GPUs, recorrendo ao framework OpenCL.

Num primeiro passo, o modelo de propagação TRACEO foi reimplementado em linguagem C, resultando no modelo cTraceo. O cTraceo destaca-se por desempenho e usabilidade melhoradas, tendo o código sido disponibilizado em open-source[2]. Deste trabalho resultaram três publicações cientificas, incluídas no Apêndice C.

Devido ao facto dos GPUs proporcionarem muito melhor desempenho de virgula flutuante em precisão simples do que em precisão dupla, e tendo em conta que tanto o TRACEO como o cTraceo foram implementados em

[2]http://eynuel.github.io/cTraceo

precisão dupla, implementou-se seguidamente uma versão de precisão simples do cTraceo. Após verificar que a qualidade dos resultados do modelo gerados em precisão simples eram comparáveis aos resultados gerados em precisão dupla, constatou-se também que esta alteração resultou num aumento de desempenho a rondar os 30%.

Seguidamente reimplemtentou-se o modelo de propagação em OpenCL, dividindo a aplicação em duas componentes. Uma componente administrativa (*host*) que trata da leitura de ficheiros de entrada, gestão de memória, controlo do dispositivo computacional e escrita de ficheiros de resultados; bem como uma componente executada no GPU (i.e., no dispositivo computacional) constituída por vários *kernels* que computam os resultados. Designou-se o modelo recém implementado de *clTraceo*.

Implementada a versão OpenCL do modelo de propagação, foram realizados testes comparativos de desempenho, avaliando-se o desempenho do clTraceo em diferentes dispositivos computacionais, para vários tipos de guias de onda e com diferentes números de raios. Observou-se um aumento de desempenho substancial entre o cTraceo e o clTraceo, com ganhos crescentes com o número de raios.

Palavras chave: acústica submarina, modelação, computação de alta performance, OpenCL, computação paralela.

List of Figures

List of Tables

Chapter 1

Introduction

Ocean engineering and research covers a wide area of applications ranging from reliable acoustic communications, to seabed surveying and environmental monitoring. The last field of research is becoming of increasing importance under the European Union's directive to ensure that "underwater noise is at levels that do not adversely affect the marine environment" [4].

Environmental monitoring and prediction applications range from prediction of noise levels from aquaculture and shipping [5] to environmental impact studies due to the construction and operation of offshore windfarms, with new applications emerging. With the growing complexity and range of possible applications, the computational requirements have grown continuously over the decades.

Ray tracing is an efficient approach for the modeling of high frequency acoustic propagation. Of particular interest in this area is the cTraceo Gaussian beam model model, which was developed at the Signal Processing Laboratory of the University of the Algarve[1]; cTraceo can provide predictions of acoustic pressure and particle velocity in waveguides with complex boundaries featuring range dependent compressional and shear properties; backscattering and multiple objects are also supported. Currently cTraceo is a single-threaded application and for increased performance the development of a parallelized version is highly desirable.

[1] www.siplab.fct.ualg.pt

Since the debut of commercially available multicore processors in mid 2005[2][3] Moore's Law (which states that processor's computational power doubles roughly every eighteen months) has only remained true when factoring in the computational power of all cores within a processor. To make use of the increasingly parallel architectures of CPUs, new software development methods emerged and tools previously only employed in high performance computing and data centers entered into the consumer market. After less than a decade, PC systems featuring sixty four cores are now readily available, while developing software for these systems remains a challenge.

Alongside the development of multicore CPUs, a quieter (r)evolution with enormous potential for scientific computing has taken place in the form of graphics processing units (GPUs). Originally developed in the mid nineties as highly specialized add-on gaming "3D-accelerators" which complemented existing 2D graphics adapters , GPUs have since combined both functionalities and have evolved into high performance general purpose vector processors.

GPUs containing 2048 cores on a single chip are now readily available, requiring new programming paradigms for efficient usage. Currently, two frameworks for GPU programming share the market: CUDA and OpenCL. CUDA is a product developed by NVidia for use with their range GPUs, and is thus restricted to this vendor's hardware. OpenCL is a multi-vendor industry standard targeted at software development for so-called "heterogeneous" systems, i.e., it is intended not only for developing software for GPUs, but also for multicore CPUs, mobile devices and other systems. Thus, OpenCL offers an approach for simultaneous software development for CPUs and GPUs, eliminating the need to master several development toolchains.

Given the parallelizable nature of of raytracing in general, and of the ray superposition algorithm in particular, adaptation of cTraceo to multiple compute cores promises to advance the quest for realtime efficient applications.

In this work, the parallelization of the cTraceo raytracing model using the OpenCL framework is addressed. This report is organized as follows: in Section 2 some of the theoretical background of raytracing is presented,

[2]The Intel Pentium D Dual Core processor was launched on the 25th of May 2005.
[3]The AMD Athlon 64 X2 was introduced on the 5th of June 2005.

while Section 3 offers some insight into the OpenCL framework. The design considerations for the parallelization are addressed in Section 4, with performance comparison results shown in Section 5;Sections 6 and 7 present the conclusions and suggestions for future work, respectively.

Chapter 2

Theoretical Background

This chapter will give a brief overview of the raytracing theory on which the clTraceo model is based.

2.1 The Acoustic Wave Equation

For a medium with constant density the Acoustic Wave Equation is given by [6]:

$$\nabla^2 p(\vec{r}, t) - \frac{1}{c^2} \frac{\partial p(\vec{r}, t)}{\partial t^2} = 0, \tag{2.1}$$

where $p(\vec{r}, t)$ is the pressure of the acoustic wave, c is the sound speed and ∇ is the Nabla differential operator. Applying a Fourier Transform to both sides of Eq. (2.1), one can obtain the Helmholtz Equation:

$$\left[\nabla^2 + \frac{\omega^2}{c^2}\right] P(\vec{r}, \omega) = 0, \tag{2.2}$$

where ω is the angular frequency and P is the acoustic pressure in the frequency domain. Assuming a plane wave approximation to the solution of Eq. (2.2), the expression for acoustic pressure can be written as:

$$P(\vec{r}, \omega) = A(\vec{r}) e^{-i\omega\tau(\vec{r})}, \tag{2.3}$$

where $\tau(\vec{r})$ is a rapidly varying phase function known as the *Eikonal*, and $A(\vec{r})$ is a much more slowly varying envelope function incorporating the effects of geometrical spreading and various loss mechanisms [7]. By substituting Eq. (2.3) in Eq. (2.2) and by considering the following high frequency approximation [6, 8],

$$\frac{\nabla^2 A(\vec{r})}{A(\vec{r})} \ll \frac{\omega^2}{c^2}, \tag{2.4}$$

the *Eikonal Equation* follows from the real part of the resulting equation:

$$(\nabla\tau)^2 = \frac{1}{c^2}, \tag{2.5}$$

while the *Transport Equation* follows from the imaginary part:

$$2(\nabla A \cdot \nabla\tau) + A\nabla^2\tau = 0. \tag{2.6}$$

The solutions of Eq. (2.5) and Eq. (2.6) will be described in the following sections.

2.2 Solution of the Eikonal Equation

From the solution of the Eq. (2.5) one can obtain the surfaces of constant phase (wavefronts). Ray paths are orthogonal to the wavefronts and indicate the direction of energy flow [7]. The solution of the Eikonal Equation requires solving the set of equations given by [8]:

$$\begin{gathered}\frac{d}{ds}\left(\frac{1}{c(s)}\frac{dx}{ds}\right) = \frac{\partial}{\partial x}\left(\frac{1}{c(s)}\right), \qquad \frac{d}{ds}\left(\frac{1}{c(s)}\frac{dy}{ds}\right) = \frac{\partial}{\partial y}\left(\frac{1}{c(s)}\right), \\ \frac{d}{ds}\left(\frac{1}{c(s)}\frac{dz}{ds}\right) = \frac{\partial}{\partial z}\left(\frac{1}{c(s)}\right),\end{gathered} \tag{2.7}$$

where s stands for the distance traveled by the acoustic wave along the ray path, $c(s)$ is the sound speed along the ray trajectory and the derivatives $\frac{dx}{ds}$, $\frac{dy}{ds}$ and $\frac{dz}{ds}$ define the ray tangent $\vec{e}_s$.

By replacing the sound speed c with sound slowness $\sigma = 1/c$, and after simplifying the set of equations given by Eq. (2.7) for a waveguide with cylindrical symmetry, the following set of equations can be obtained:

$$\frac{dr}{ds} = c(s)\sigma_r(s), \qquad \frac{dz}{ds} = c(s)\sigma_z(s), \qquad \frac{d\sigma_r}{ds} = -\frac{1}{c^2}\frac{\partial c(s)}{\partial r}, \qquad \frac{d\sigma_z}{ds} = -\frac{1}{c^2}\frac{\partial c(s)}{\partial z}, \tag{2.8}$$

where σ_r, σ_z are the vector components of the sound slowness σ. The initial conditions for solving Eq. (2.8) are given by [8]:

$$r(0) = r_0, \qquad z(0) = z_0, \qquad \sigma_r(0) = \frac{\cos(\theta_0)}{c_0}, \qquad \sigma_z(0) = \frac{\sin(\theta_0)}{c_0}, \tag{2.9}$$

where θ_0 is the launching angle of the ray, $[r_0, z_0]$ is the source's position and c_0 is the sound speed at the source position. By solving Eq. (2.8) together with the initial conditions (2.9) the ray paths and travel times through the waveguide can thus be determined. The next step will be determining the amplitude of the rays by solving the transport equation.

2.3 Solution of the Transport Equation

The classical solution for the ray pressure $P(s, \omega)$ can be written as [6]:

$$P(s, \omega) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c_0}\frac{\cos(\theta_0)}{J}}e^{-i\omega\tau(s)}, \tag{2.10}$$

where J stands for the Jacobian, $\tau(s)$ is the travel time along the ray $c(s)$ is the sound speed along the ray trajectory and ω is the angular frequency.

Unfortunately, the classical solution breaks down in the vicinity of caus-

tics, due to the fact that the Jacobian tends to zero [6, 7, 8]. These singularities can be addressed using the Gaussian Beam Approximation, which is discussed in the following section.

2.4 The Gaussian Beam Approximation

In the Gaussian Beam Approximation the ray is considered to be the central axis of a beam, which features a Gaussian intensity distribution along the ray normal, as shown in Fig. 2.1.

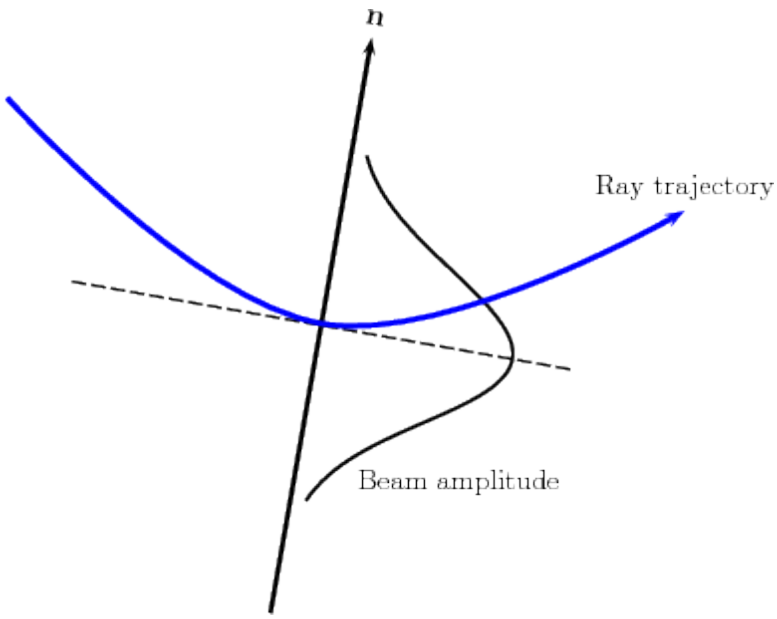


Figure 2.1: *Gaussian beams: amplitude decay along the normal.*

Thus a particular Gaussian beam solution for acoustic pressure, compatible with cylindrical spreading, can be written as [9]:

$$P(s,n) = \frac{1}{4\pi}\sqrt{\frac{\cos\theta_0}{c_0}\frac{c(s)}{rq(s)}}\; e^{-i\omega\left[\tau(s)+\frac{1}{2}\frac{p(s)}{q(s)}n^2\right]}, \tag{2.11}$$

where r is the range coordinate, n is the normal distance to the beam's central axis, and $p(s)$ and $q(s)$ are auxiliary functions, derived by solving a system of differential equations in the neighborhood of the ray axis [9]; $q(s)$

is also proportional to the Jacobian [6]. The approximation given by Eq. (2.11) solves the issues of the singularities at caustics, but it introduces other artifacts. Specifically, rays being returned to the source will correspond to focusing (instead of reflected) waves. And since the cylindrical approximation breaks down in the vicinity of the origin, the field of such backpropagating rays will increase as the ray aproaches the source, instead of decaying as raylength increases. A solution for the drawbacks of Eq. (2.11) is presented in the following section.

2.5 The Gaussian Beam Aproximation in Cartesian Coordinates

The typical Gaussian Beam Approximation for a waveguide with cylindrical symmetry is given by Eq. (2.11). After a carefull review of the beam expressions on the (x, z) plane, the following expression for a ray centered Gaussian Beam Approximation can be obtained [8, 1]:

$$P(s,n) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c_0}\frac{\cos(\theta_0)}{q_\perp(s)q(s)}}\; e^{-i\omega\left[\tau(s)+\frac{1}{2}\frac{p(s)}{q(s)}n^2\right]}, \tag{2.12}$$

where n is the normal distance to the central axis of the Gaussian beam, and $p(s)$, $q(s)$ and $q_\perp(s)$ are auxilliary parameters, which determine the beamwidth along the arclength s. In contrast to the solution for a waveguide with cylindrical simmetry as given by Eq. (2.11), $q_\perp(s)$ stands in the place of the radial coordinate r. Because the full Gaussian beam expression is solved on (x, z) plane instead of on the (r, z) plane, backscattering can now be accomodated. The approximation given by Eq. (2.12) is the basis for the calculation of ray amplitudes in cTraceo.

2.6 Attenuation

In Equation (2.12) the ray amplitude A is given by:

$$A = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c_0}\frac{\cos(\theta_0)}{q_\perp(s)q(s)}}. \tag{2.13}$$

The solution given by Eq. (2.13) does not incorporate the losses introduced by volume attenuation and reflections at media interfaces. To include these loss mechanisms, A is replaced by a corrected amplitude a, given by:

$$a = A\phi_r\phi_V, \tag{2.14}$$

where ϕ_r represents the total decay due to interface reflections, and ϕ_V represents the volumetric attenuation.

2.6.1 Volumetric Attenuation

The volumetric attenuation factor ϕ_V is given by:

$$\phi_V = e^{-\alpha_T s}, \tag{2.15}$$

where s is the ray arclength and α_T is the frequency dependent Thorpe attenuation in dB/m, as given by [10]:

$$\alpha_T = \frac{40f^2}{4100 + f^2} + \frac{0.1f^2}{1 + f^2}, \tag{2.16}$$

where the frequency f is given in kHz.

2.6.2 Boundary Reflection

The total amplitude decay due to reflections at media interfaces is given by:

$$\phi_r = \begin{cases} 1 & n_r = 0 \\ \\ \prod_{i=1}^{n_r} R_i & n_r > 0 \end{cases}, \tag{2.17}$$

where n_r is the total number of reflections at interfaces, and R_i is the complex reflection coefficient at the ith reflection. cTraceo considers four types of interfaces:

- Absorbent: the wave energy is transmitted completely to the medium beyond the interface, so $R = 0$, thus terminating the ray propagation.
- Rigid: the wave energy is reflected completely, with no phase change, so $R = 1$.
- Vacuum: the wave energy is reflected completely, with a phase change of π radians, so $R = -1$.
- Elastic: the wave energy is partially reflected, with R being a complex value and $|R| < 1$.

The reflection coefficient for an elastic medium is calculated through the following set of equations [11]:

$$R(\theta) = \frac{D(\theta)\cos\theta - 1}{D(\theta)\cos\theta + 1}, \qquad \text{and} \qquad D(\theta) = A_1\left(A_2\frac{1 - A_7}{\sqrt{1 - A_6^2}} + A_3\frac{A_7}{\sqrt{1 - A_5/2}}\right), \tag{2.18}$$

where θ is the ray's angle relative to the boundary's tangent, and with

$$A_1 = \frac{\rho_2}{\rho_1}, \qquad A_2 = \frac{\widetilde{c}_{p_2}}{c_{p_1}}, \qquad A_3 = \frac{\widetilde{c}_{s_2}}{c_{p_1}}, \tag{2.19}$$

$$A_4 = A_3\sin\theta, \qquad A_5 = 2A_4^2, \qquad A_6 = A_2\sin\theta, \quad A_7 = 2A_5 - A_5^2,$$

where

$$\widetilde{c}_{p_2} = \frac{c_{p_2}}{1 + \widetilde{\alpha}^2_{c_p}} \left(1 - i\widetilde{\alpha}_{c_p}\right), \qquad \widetilde{c}_{s_2} = \frac{c_{s_2}}{1 + \widetilde{\alpha}^2_{c_s}} \left(1 - i\widetilde{\alpha}_{c_s}\right), \tag{2.20}$$

$$\widetilde{\alpha}_{c_p} = \frac{\alpha_{c_p}}{40\pi \log e}, \qquad \widetilde{\alpha}_{c_s} = \frac{\alpha_{c_s}}{40\pi \log e},$$

where the attenuation values are given in dB/λ. The reflection coefficient is real when there is no attenuation and the angle of incidence is less than the critical angle

$$\theta_c = \arcsin\left(\frac{c_{p_1}}{c_{p_2}}\right).$$

Chapter 3

OpenCL

OpenCL is an open standard for cross-platform parallel programming of heterogeneous systems developed by an industry consortium coordinated through the Khronos Group [12]. OpenCL allows for developing software on CPUs, GPUs, FPGAs and custom devices (e.g. hardware video decoders) and is currently supported on Windows, Linux, Mac OS, iOS (iPhones and iPads) as well as Android and with support for web browsers currently under development [13]. Supporting such a diverse range of target architectures is possible through an abstracted system representation as presented in Section 3.1. The actual computational work is done in *kernels*, written in a programming language derived from C99 (OpenCL C), while an Application Programming Interface (API) is used for device control.

An overview of the workflow of an OpenCL application is given in Section 3.2.

3.1 System Representation in OpenCL

An OpenCL system consists of one *host* and one or more *compute devices*, each having it's own distinct memory space.

The host executes *host code* and is responsible for work scheduling and initialization tasks like querying the system for available compute devices, configuring compute devices and taking care of all memory allocation and

memory transactions, as discussed brieflyl in Section 3.2. The host process also assigns computational work to a compute device, which performs it's tasks asynchronously from the host.

Functions running on a compute device are written in a derivative of the C language (OpenCL C) and are called *kernels*. Kernels may be offloaded to one or more compute devices, connected to the host by a generic data bus.

For a device to be available for use in OpenCL, it's corresponding OpenCL driver must be installed.

Furthermore, computational tasks which for one reason or another are deemed inefficient to run on a compute device may be executed in host code instead, in parallel to the device code.

In the case of a typical desktop PC system containing a CPU and a GPU, the host code will be run by a single CPU thread, with the host's memory corresponding to the system's random access memory (RAM). Kernels are then executed on the GPU with it's own dedicated memory, which is connected to the host through a PCI-Express bus as seen in Figure 3.1.

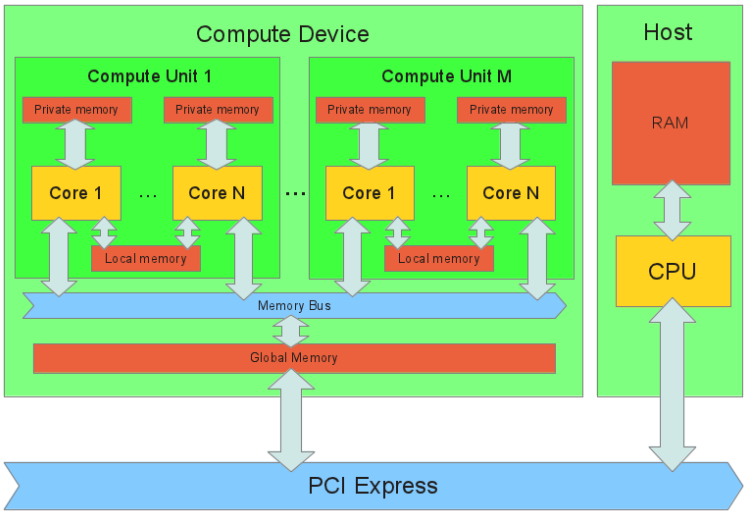


Figure 3.1: *Simplified OpenCL system representation of a typical Desktop PC containing a single compute device.*

It is important to note that as OpenCL can be used on many different systems, the distinction between host and device may become less clear than in this example. This is the case when running OpenCL kernels on a multi-core CPU, where the host code will be executed by one CPU thread, while each CPU core will also be running kernel thread(s) to perform computational work. In some cases, the same compute device may be simultaneously supported by more than one driver. This is the case for some Intel CPUs which are supported by both the Intel and AMD OpenCL drivers, in which case the software will be able to choose between two drivers for the same hardware. Throughout this text, unless otherwise noted, the compute device will be assumed to be a single GPU.

Compute devices are subdivided into one or more *Compute Units*, which are the smallest entities to which the host may explicitly assign work. As can be seen in Figure 3.1, each compute unit may contain one or more cores, or *Processing Elements* in OpenCL terminology. As the reader may be familiar with the term, in this text processing elements will be referred to as *cores*.

Compute units contain a limited amount of *private memory* which provides very low latency and high bandwidth for frequently used variables and which is flexibly split up among the threads assigned to it. A group of threads assigned to run on a compute unit is called a *workgroup*, within which each thread has access to it's own reserved private memory space. Although the number of threads per compute unit -i.e. the *workgroup size*- may be higher than the number of cores within the compute unit, it may still be limited by the total amount of private memory physically available to the compute unit.

All cores within a compute unit, and thus all threads within a workgroup, share *Local Memory*, which provides higher latency and lower bandwidth than private memory and may be used for sharing data between threads or to store less frequently used data.

Global memory is generally a compute device's largest memory space and is accessible to threads from all workgroups. However, it generally also has the highest latency and lowest bandwidth.

Table 3.1 shows a comparison of the bandwidths and latencies for the

various memory regions available on a NVidia GeForce GTX 580 graphics card.

	Private	Local	Global
Size	16×128 KiB[1]	16×64 KiB[2]	3 GiB
Bandwidth	2.9 TiB/s	1 - 2 TiB/s	179 GiB/s
Latency	< 1 ns	3 - 4 ns	100 - 150 ns

Table 3.1: *Overview of memory bandwidths and latencies of NVidia GTX 580.*

It should be noted that a GPU's global memory, although being it's "lowest bandwidth" memory, may still be an order of magnitude faster than a PC's RAM.

A GPU's processing cores, private and local memory are generally contained within a single integrated circuit, while the much larger global memory is located off-chip on the printed circuit board of the graphics adapter.

In a CPU architecture, private, local and global memory all map to the system's RAM. While some performance differences between the memory spaces may exist due to optimization of cache usage, this will be implementation dependent and highly variable between architectures.

[1] In each of the GPU's 16 compute units, 128 KiB are flexibly divided up among 32 cores.

[2] Each compute unit contans 64 KiB of local memory.

3.2 OpenCL Workflow

Although most (if not all) of an application's computational work is offloaded to compute devices, the host process retains all control tasks. OpenCL provides the host with a large and complex API for these control tasks, the complete description of which would be beyond the scope of this text (for a more detailed background the reader may refer to [14]).

In a nutshell, the workflow of an OpenCL application consists of:

1. Probe the system for available OpenCL platforms (i.e., installed drivers).
2. Probe for compute devices available through each driver.
3. Load files containing the kernel source code[3] and compile it for the selected compute device(s).
4. Allocate required memory on the compute device(s) and copy ("upload") any required data to the device. Since the OpenCL API allows for asynchronous operation, this may be done in parallel to the kernel compilation.
5. Queue the kernel for execution on the device and either wait for it to finish or perform other tasks while waiting.
6. Copy ("download) the results from the compute device and perform any required post-processing.
7. Optionally repeat steps 4 to 6 to perform further work.

The host process is also responsible for explicitly freeing any memory allocated on the compute device. The results of failing to do so may vary greatly, depending on device vendor, operating system and driver version. Between having no consequence whatsoever to resulting in a complete system crash, a wide range of consequences can be possible.

[3] An intermediate representation for kernel code has been included as an optional extension in the OpenCL 1.2 standard.

Chapter 4

Implementation

This chapter will address several design decisions made before the actual implementation of the parallelized version of cTraceo. The structure of this chapter is as follows: Section 4.1 a set of test cases, which are used for various benchmarking purposes during the development; Section 4.2 investigates the viability of using single precision floating point arithmetic, while Section 4.3 addresses issues of local memory organization. Finally, Section 4.4 provides an overview of the structure adopted for the parallelized raytracing kernels.

4.1 Test Cases

In order to perform detailed comparisons, several waveguide configurations with varying complexity were chosen for benchmarking purposes:

- The `Pekeris` test case is a well known short range shallow water waveguide chosen for its simplicity. It consists of flat rigid bottom, flat surface with a vacuum above it, and an isovelocity sound speed profile, as shown in Figure 4.1a.
- The `Munk` test case is a long range deep water waveguide, with a flat surface with a vacuum above it, a flat rigid bottom and an interpolated user provided Munk sound speed profile, as shown in Figure 4.1b.

- The `Sletvik` waveguide is based on a real world bathymetry and measured sound speed profile, which is in contrast to the other test cases, which are based on synthetic data. This short range shallow water waveguide is modeled with a vacuum above the surface and elastic bottom properties as shown in Figure 4.1c.

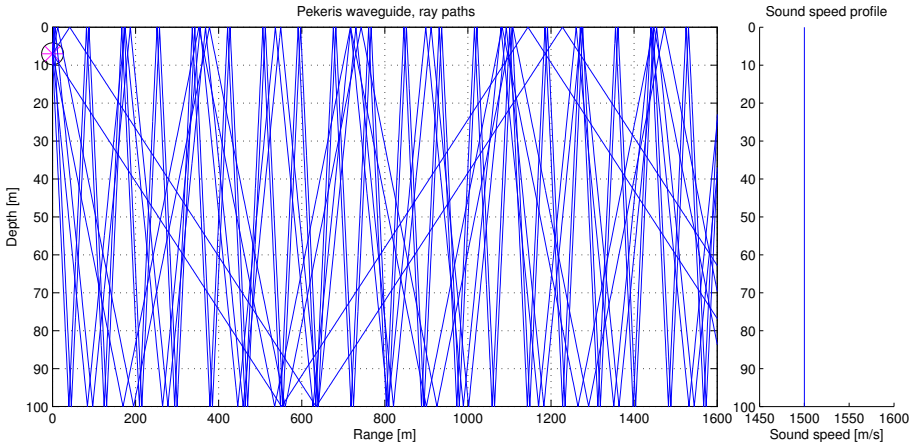
Table 4.1 shows an overview of the test cases' properties.

	Test Case		
	Pekeris	Munk	Sletvik
Max. Range	1 km	100 km	120 m
Max. Depth	100 m	5000 m	25 m
Source Depth	25 m	1000 m	3m
Surface Geometry	Flat	Flat	Flat
Surface Interface	Vacuum	Vacuum	Vacuum
Sound Speed Profile	Isovelocity	Interpolated	Interpolated
Bottom Geometry	Flat	Flat	Interpolated
Bottom Interface	Rigid	Rigid	Elastic

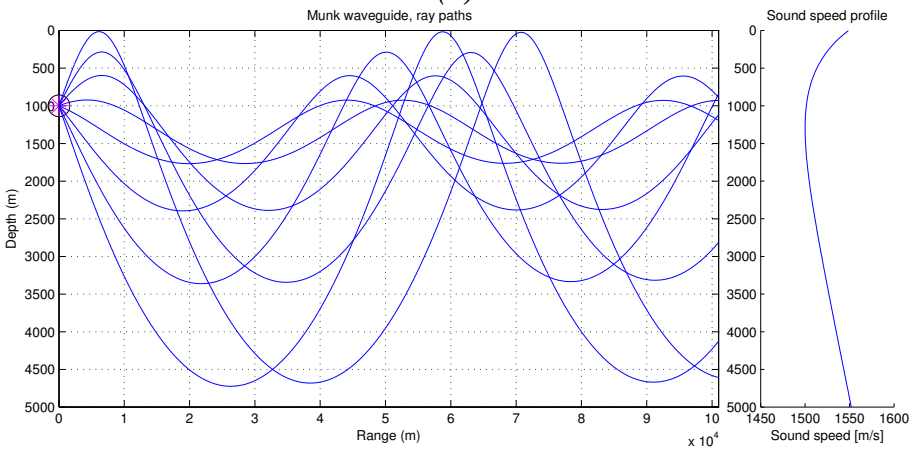
Table 4.1: *Overview of the waveguide geometry for the test cases.*

4.2 Single Precision

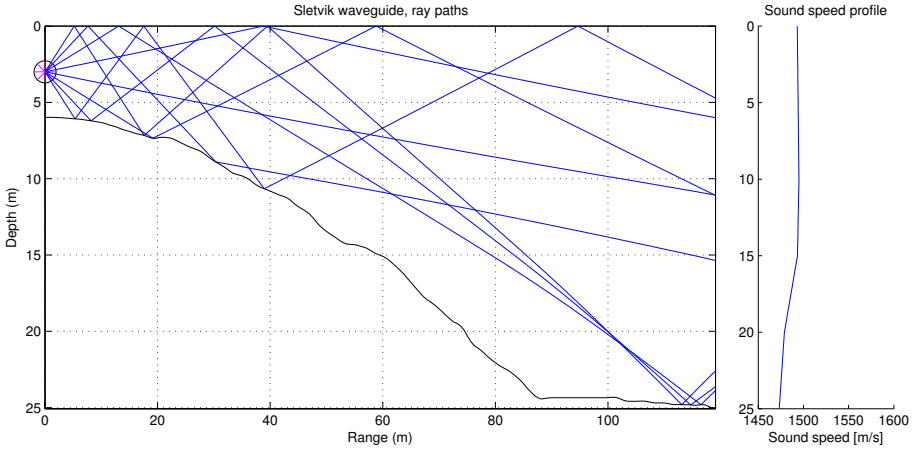
Given the fact that GPUs provide far better single precision than double precision floating point performance, it is highly desirable to perform all computations in single precision. Since the baseline cTraceo model is implemented in double precision, this raises the question of whether a conversion to single precision will introduce numerical stability issues. In order to verify if such issues occur, a single precision version of cTraceo was implemented and it's output was compared against the double precision implementation. A stage-wise approach to comparing the results was devised. At first the output of the numerical Eikonal and Transport Equation solvers was validated, while in a second stage the computation of acoustic pressure was addressed. The Eikonal and Transport Equation solvers produce several output vectors per traced ray, namely:



(a)



(b)



(c)

Figure 4.1: *Ray trace for test case (a) Pekeris, (b) Munk, (c) Sletvik.*

x - The ray's x coordinate for each point of it's trajectory,

y - The ray's y coordinate for each point of it's trajectory,

τ - The propagation time along the ray,

A - The complex beam amplitude along the ray.

As a metric of similarity to evaluate the quality of the single precision results, a cross correlation at zero lag was computed, using Matlab's `xcorr(a, b, 0, 'coeff')` command. The output of this metric is in the form of a coefficient in the range [-1, 1] where '1' implies that the vectors are identical, '-1' indicates the vectors are symmetric, while at '0' the vectors are entirely uncorrelated.

Each of the test waveguides was modeled at 256 rays, with a similarity coefficient computed for each ray. The mean of these coefficients was then computed for each waveguide. Results are summarized in Table 4.2. Due to the fact that many of the resulting values are close to unity, it was chosen to show the difference between the computed coefficients and unity.

	Test Case		
	Pekeris	Munk	Sletvik
x	8×10^{-5}	1×10^{-5}	1×10^{-5}
y	2×10^{-5}	1×10^{-5}	1×10^{-5}
τ	8×10^{-5}	1×10^{-5}	1×10^{-5}
A	1×10^{-5}	1.706×10^{-2}	1×10^{-5}

Table 4.2: *Difference between unity and the mean similarity coefficients for ray variables between single and double precision implementations of cTraceo.*

The results seen in Table 4.2 lead to the conclusion that for all practical purposes, the results of the single precision implementation are identical to those of the double precision version.

Having verified the equivalence of the single precision and double precision results of the Eikonal and Transport Equation solvers, the same test waveguides were used for comparing acoustic pressure results. For this, acoustic pressure for each waveguide was computed first along a vertically centered horizontal array and then along a horizontally centered vertical array, each with sixteen hydrophones. Comparison results are shown in Table 4.3.

	Horizontal Array		Vertical Array	
Waveguide Name	$Re\{P\}$	$Im\{P\}$	$Re\{P\}$	$Im\{P\}$
Pekeris	1×10^{-5}	1×10^{-5}	5×10^{-5}	6×10^{-5}
Munk	2×10^{-5}	1×10^{-5}	5×10^{-5}	1.2×10^{-4}
Sletvik	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}

Table 4.3: *Difference between unity and the similarity coefficients of pressure along sixteen-element hydrophone arrays between single and double precision implementations of cTraceo.*

With the result shown in Table 4.3 showing that the coherent pressure as computed by the single precision implementation and the double precision version are close to identical, concerns about the numerical stability when using single precision floating point were put aside.

A side effect of implementing a single precision version of cTraceo was an observed reduction of processing times of around 30% across all output options.

4.3 Local Memory Considerations

While each ray's solution is mathematically independent from the others', computationally this is not necessarily true. As any numerical implementation of raytracing requires an interpolation of environmental information, it is hard to predict the memory access patter to this information for the calculation of each ray, not to speak of the memory access pattern of several hundred simultaneous threads. GPU memory controllers are built in such a way that a memory read blocks the entire memory bus, even when reading less memory than the bus width. This means that best performance is obtained when accessing contiguous blocks of memory, with a total size which is an integer multiple of the memory bus width. In the case of the Nvidia Geforce GTX 580 which has a memory bus width of 384 bit, or 12 floats, this implies that reading a single float from global memory will waste 11/12 or 92% of the available memory bandwidth. With high numbers of concurrent threads wanting to access global memory simultaneously, memory bandwidth quickly becomes a performance bottleneck.

With computational performance of GPUs being highly dependent on an efficient global memory access pattern, a first step to attain good global memory performance is to reduce access to this "slow" memory by as much as possible. By simply keeping frequently accessed data in local memory, the need to optimize the global memory access pattern can be circumvented.

The waveguide's environmental information is a prime candidate to be kept in local memory since it is frequently accessed, and relatively small. The question then becomes how "small" this data is and whether the GPU's local memory is large enough to contain it.

In cTraceo the environmental information is stored in a data structure named `env_t`, which is defined as

```
typedef struct{
    float             top, bottom;    //2 * 4 Byte
    float             left, right;    //2 * 4 Byte
    interface_t       altimetry;
    soundSpeed_t      soundSpeed;
    interface_t       bathymetry;
}env_t;
```

where:

`top`, `bottom`, `left`, `right`
: are the waveguide's *hard* limits, i.e., the "box" within which rays are computed,

`interface_t`
: is a data structure which contains an interface's information, as shown below, and

`soundSpeed_t`
: is a data structure which contains the waveguide's sound speed information, also shown below.

The size of the `env_t` struct in bytes is thus given by:

$$N_{env} = 16 + N_{alt} + N_{ssp} + N_{bat}, \tag{4.1}$$

where N_{alt}, N_{ssp} and N_{bat} are the number of bytes required for storing the altimetry, sound speed profile and bathymetry respectively.

The `interface_t` struct is used to contain both the altimetry and bathymetry information, and for an interface with elastic properties is defined as:

```
typedef struct{
  uchar   interfaceType;           // 1 Byte
  uchar   interfacePropertyType;   // 1 Byte
  uchar   interfaceInterpolation;  // 1 Byte
  uchar   interfaceAttenUnits;     // 1 Byte
  int     nCoords;                 // 4 Byte
  float*  x;                       // nCoords * 4 Byte
  float*  y;                       // nCoords * 4 Byte
  float*  cp;                      // m * 4 Byte
  float*  cs;                      // m * 4 Byte
  float*  rho;                     // m * 4 Byte
  float*  ap;                      // m * 4 Byte
  float*  as;                      // m * 4 Byte
}interface_t;
```

where:

`interfaceType`, `interfacePropertyType`, `interfaceInterpolation`, `interfaceAttenUnits`
: are user specified options related to the interface's properties, who's allowable values are listed in Appendix B,

`nCoords`
: specifies the number of coordinates for the interface's geometry,

`x`, `y`
: are variably sized vectors which contain the interface's geometry, and

`cp`, `cs`, `rho`, `ap`, `as`
: are variably sized vectors containing the interface's elastic properties (compressional wave speed, shear wave speed, density, compressional wave attenuation and shear wave attenuation respectively). For an interface with constant elastic properties along it's geometry one set of values must be stored, i.e. $m = 1$. For non-homogeneous interfaces, one set of elastic properties must be stored for each point of the interface geometry, thus $m = nCoords$.

The total size in bytes required for storing an interface $N_{interface}$ is then:

$$N_{interface} = 8 + 4 \times (2 \times n_{interface} + 5 \times m)\,, \tag{4.2}$$

where $n_{interface}$ is the number of coordinates of the interface's geometry.

The `soundSpeed_t` struct contains the waveguide's sound speed information and in the case of a profile is defined as:

```
typedef struct{
  uchar   cDist;      // 1 Byte
  uchar   cClass;     // 1 Byte
  int     nCoords;    // 4 Byte
  float*  y;          // nCoords * 4 Byte
  float*  c;          // nCoords * 4 Byte
}soundSpeed_t;
```

where:

`cDist, cClass`
: are user specified options, who's alowable options are listed in Appendix B,

`nCoords`
: specifies the sound speed profile's number of points,

`y, c`
: are variably sized vectors containing the depths and corresponding sound speeds.

The number of bytes required for storing a sound speed profile is then:

$$N_{soundSpeed} = 6 + 8 \times n_{ssp}, \tag{4.3}$$

where n_{ssp} is the number of points in the ssp.

It then follows that the total number of bytes required for storing a waveguide environment in memory, N_{env} is given by:

$$N_{env} = 38 + 4 \times (2 \times (n_{alt} + n_{ssp} + n_{bat}) + 5 \times (m_{alt} + m_{bat})) \tag{4.4}$$

where n_{alt} is the number of points defining the surface geometry, n_{ssp} is the number of points in the sound speed profile, and n_{bat} is the number of points defining the bathymetry geometry.

According to Equation (4.4), a moderately complex waveguide with a flat homogeneous altimetry (i.e., defined over 2 points), a 128 point sound speed profile and a 512 point bathymetry would require a total of 15434 bytes of memory. Considering that the local memory size of the available testing hardware[1] is 64 kB, it was considered acceptable to make use of this memory to store the environmental data. Still, the user must take some care in preparing the waveguide data to avoid exceeding the available local memory.

In practice, the `env_t` struct does not directly contain any data vectors, but instead contains pointers to dynamically allocated memory chunks as seen in Figure 4.2.

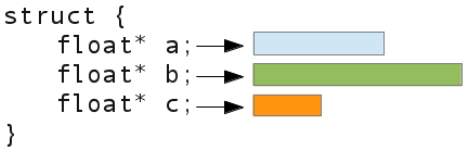


Figure 4.2: *Example of a C structure containing pointers to separately allocated memory chunks.*

This would result in having to allocate, copy and assign 17 separate arguments to a kernel for the environmental information alone. To reduce the number of dynamically allocated memory chunks, a single vector which is sufficiently large to contain all the environmental data is allocated.

The pointers contained in the `env_t` struct then point to the position corresponding to their associated data in the packed environmental information, as seen in Figure 4.3.

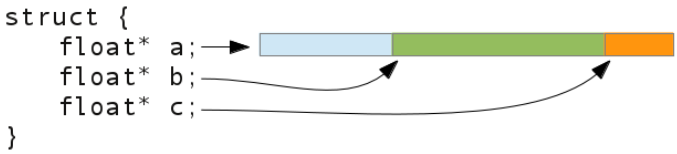


Figure 4.3: *Example of a C structure containing pointers to different positions of a packed vector allocated in a single memory chunk.*

This then simplifies further development by reducing the amount of API

[1] NVidia GeForce GTX 580

calls required to copy the environmental data from the host to the compute device.

4.4 OpenCL Kernel Structure

Since a high performance forward model implementation is mostly of interest when performing large amounts of model runs, it was decided to implement the output options which are relevant for such a purpose. Specifically, a focus was made on implementing computation of arrival patterns and acoustic pressure.

Since rays are mathematically independent from each other, it was decided to create one thread per launched ray. Each thread can then trace a single ray from beginning to end, without any need for inter-thread communication.

- In a first step, a kernel is launched which solves the Eikonal Equation for all rays simultaneously. This includes detection of ray-interface intersections and computation of reflection coefficients.
- In a next step, the data generated by the Eikonal solver is passed to the Transport Equation solver. This kernel again runs with one ray per thread and computes the beam amplitude along each ray as given by Equation 2.14.
- Following the computation of all data along the ray path, the next step in the process is finding each ray's bracketing indexes of all hydrophone positions. In other words, with a ray path consisting of a sequence of points in space, it is necessary to find the index of the ray coordinate closest to each hydrophone. A simple bisection method is employed for this purpose; with the corresponding kernel again running at one thread per launched ray.

At this point the paths diverge, depending on the chosen output option.

- If computing arrival patterns by proximity, the final step consists in storing the information of those rays which pass within a user specifiable distance of a hydrophone.
- If computing acoustic pressure, parallelism is achieved through a kernel which computes each ray's pressure contribution to all hydrophones. This results in a pressure map for each ray.
- A final step is then required to sum the intermediate per-ray pressure into a final result. Since realistic hydrophone arrays are of limited dimensions, it was decided to implement a reduction kernel in single threaded way, as no performance benefits are expectable at such small work sizes.

Chapter 5

Benchmark Results

To analyze the performance of the conversion to OpenCL, acoustic pressure over an eight element vertical array was computed for the test cases introduced in Section 4.1. Model run times for powers of 2 between 2^4 and 2^{12} rays were obtained. For every number of rays, thirteen runs were performed per test case. The maximum and minimum values were then discarded and the average run time was computed from the remaining eleven runs.

Benchmarks were performed on a desktop PC with an Intel CPU[1] and an NVidia GPU[2]. The same CPU was used to run the baseline cTraceo model as well as comparing OpenCL performance with the Intel and the AMD OpenCL drivers.

Results for all waveguides are shown in Figure 5.1, in which for each waveguide an overview is shown on the left, while on the right side only run times within the range of 1500 ms are shown in order to allow for easier visualization of small values.

[1] Intel i7-3930k with 6 cores at 3.5 GHz.
[2] NVidia GeForce GTX580 with 512 cores at 1.5 GHz.

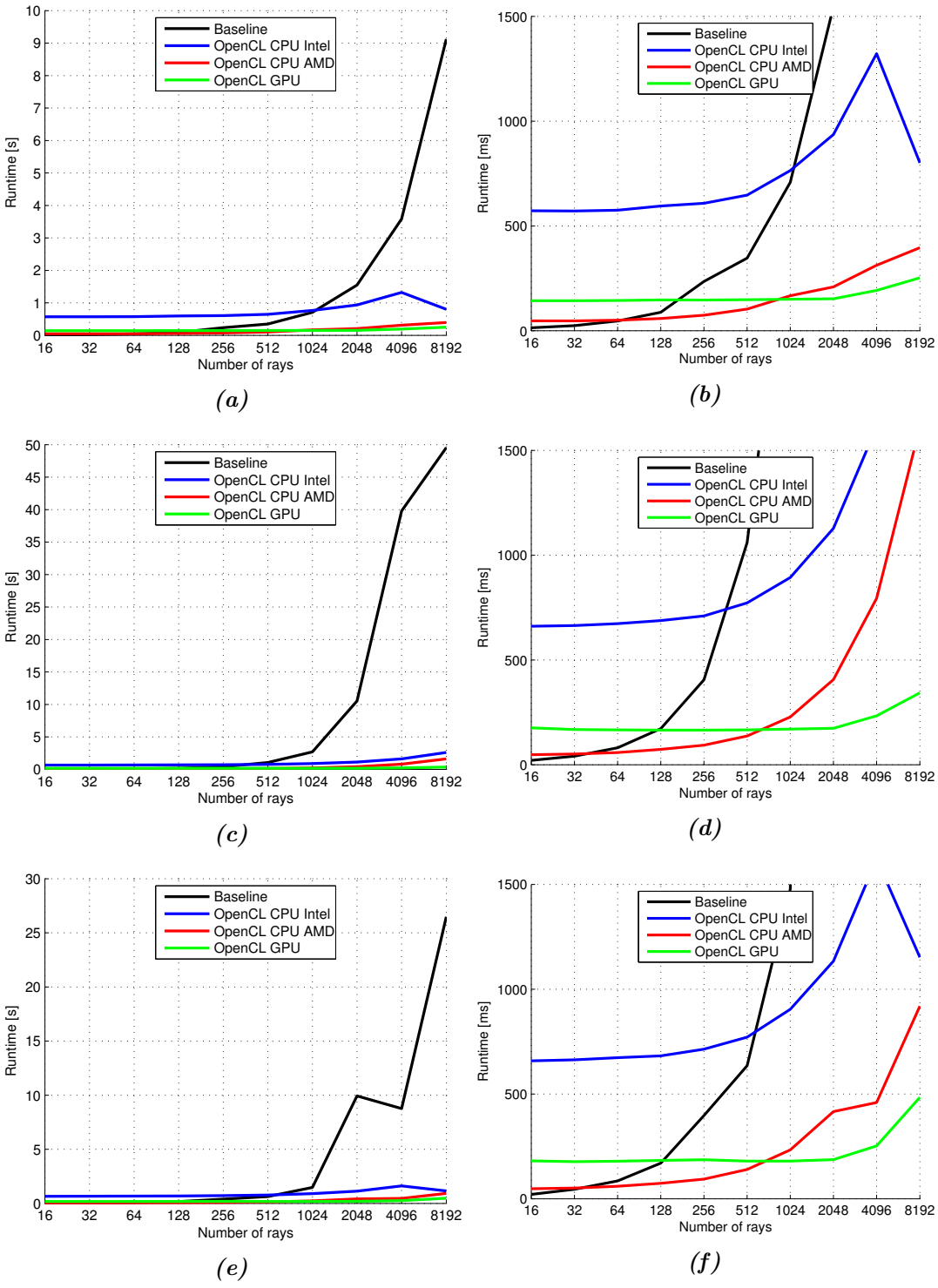


Figure 5.1: *Average model run times for the analyzed test waveguides; Pekeris (a) and (b), Munk (c) and (d), Sletvik (e) and (f).*

The average run times are also shown in detail in Tables 5.1, 5.2 and 5.3.

	Number of Rays									
	16	32	64	128	256	512	1024	2048	4096	8192
Baseline	14	24	46	88	235	346	708	1547	3584	9121
OpenCL CPU (Intel)	572	571	575	595	608	647	764	936	1322	801
OpenCL CPU (AMD)	47	47	50	58	74	103	167	209	312	396
OpenCL GPU	143	143	144	147	146	148	150	152	192	252

Table 5.1: *Average run times in milliseconds for Pekeris test case.*

	Number of Rays									
	16	32	64	128	256	512	1024	2048	4096	8192
Baseline	21	41	81	172	405	1060	2685	10536	39797	49536
OpenCL CPU (Intel)	661	664	673	688	710	772	893	1128	1612	2605
OpenCL CPU (AMD)	48	51	58	73	93	137	228	406	794	1610
OpenCL GPU	176	168	166	165	165	166	170	174	233	343

Table 5.2: *Average run times in milliseconds for Munk test case.*

	Number of Rays									
	16	32	64	128	256	512	1024	2048	4096	8192
Baseline	21	45	85	170	397	635	1477	9929	8757	26483
OpenCL CPU (Intel)	658	663	673	682	714	771	904	1134	1621	1152
OpenCL CPU (AMD)	48	51	60	74	94	140	233	416	459	919
OpenCL GPU	181	177	179	183	186	180	180	187	252	484

Table 5.3: *Average run times in milliseconds for Sletvik test case.*

It can be seen that the average run time for the baseline model scales approximately linearly with number of rays for the Pekeris case, i.e., with the doubling of the number of rays, processing time approximately doubles as well. This behavior is only partially applicable to the Munk and Sletvik

test cases, where the run times for high numbers of rays increase at higher rates.

For low numbers of rays, the OpenCL version running on a CPU with the Intel driver is slower than the baseline, only gaining the edge at or above 1024 rays, depending on the test case.

For any number of rays and over all test cases, using the AMD OpenCL driver on the same CPU results in a reduction of processing time between ≈550 and ≈650 milliseconds compared to the Intel driver. Also, it performs better than baseline at 64 rays and above.

Across all test cases, run times on the GPU stay almost constant with only minimal fluctuations up around to 1024 rays, only very slightly increasing at higher numbers of rays.

To give a measure of performance gained in relation to the baseline implementation, Figure 5.2 shows ratios between the baseline run time and the OpenCL run times.

In Figure 5.2 it can be seen that the baseline implementation is fastest when modeling the test cases at less than 64 rays. For up to 512 rays, best performance was obtained when running the test cases on the CPU with AMD driver. From 1024 rays onward, best performance is consistently provided by the GPU. The Intel driver consistently provided the lowest performance of the OpenCL devices.

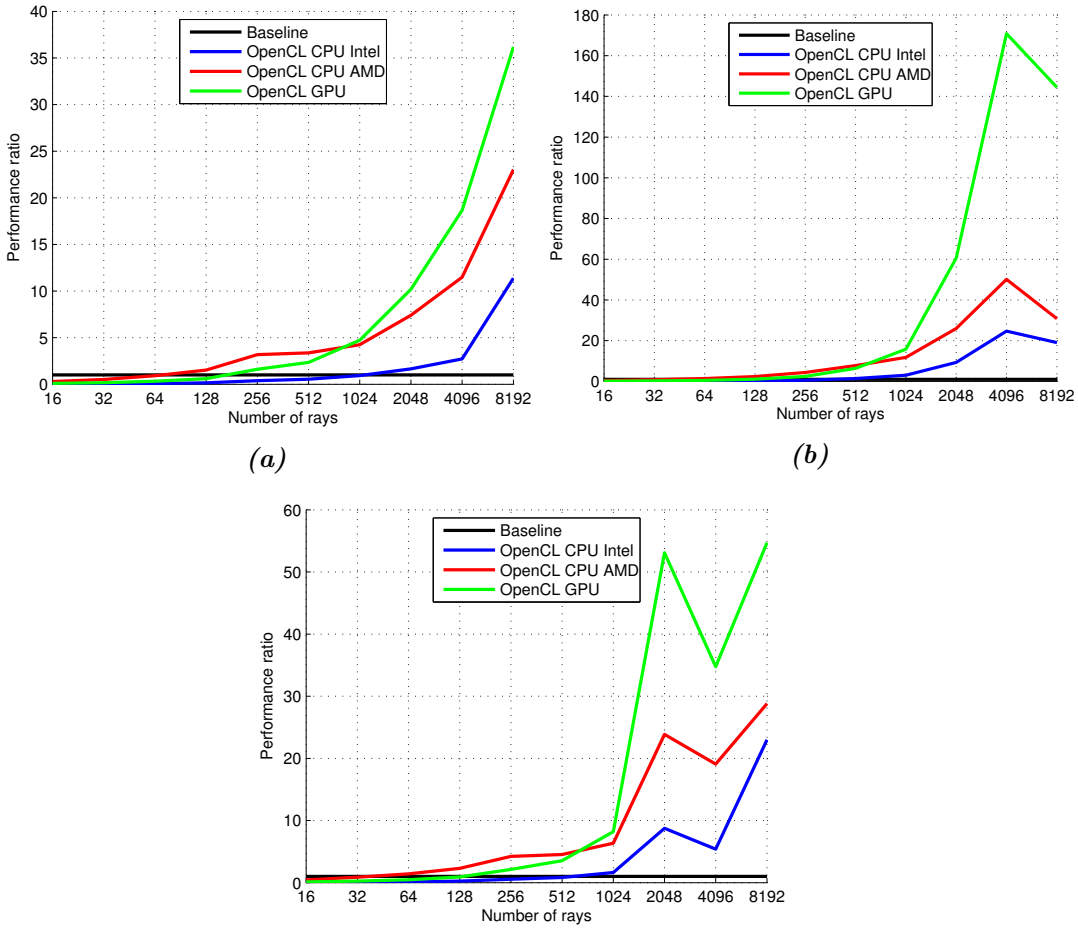


Figure 5.2: *Ration between run times of baseline cTraceo implementation and on different OpenCL platforms; Pekeris (a), Munk (b) Sletvik (c).*

Chapter 6

Conclusions

From the discussions presented in the previous chapters the following conclusions can be drawn:

- Algorithmically, the amount of computational work is linearly dependent on the number of traced rays which would imply that the processing time should increase at the same rate as the amount of rays. It was seen that in fact this does not always hold true for the baseline cTraceo model when running at high numbers of rays.

- The OpenCL framework is complex and comes with a steep learning curve, but it offers potentially large performance benefits. Particularly, it's explicit management of memory spaces (through global, local and private memory) forces the programmer to take the hardware structure into account which can help in improving performance.

- The linear offset in processing times on the same CPU between the Intel and AMD OpenCL drivers, with otherwise similar trend leads to the conclusion that the Intel driver incurs a large initialization overhead of around $\sim$550 to $\sim$650 milliseconds when compared to the AMD driver. Due to this and due to the model's otherwise short run times, the Intel driver is currently ill suited for use with clTraceo.

- With the good performance provided by the AMD OpenCL CPU driver, this seems to be a good option for real world cases.

- Due to the GPU's high number of cores (512), and due to the fact that each ray trajectory is mapped to one thread, the GPU is in effect being underutilized when computing less than 512 rays. This explains why up to this number of rays the model run times remain approximately constant and only start increasing at higher numbers of rays.
- The start-up and initialization time incurred due to OpenCL reduces the performance gains for simple waveguides and low numbers of rays, but this is more than made up for when modeling at high numbers of rays.

Chapter 7

Future Work

Overall, the performance gain obtained by the parallelization can be considered a success. In order to improve on this work, the following future work is proposed:

- With model run time already dominated by start-up and initialization tasks instead of actual computations, any algorithmic optimization will only result in minor performance improvements. It would thus be desirable to alter the model so as to model multiple waveguides without reinitializing.
- Several global variables which are computed for every step along a ray trajectory are computed in the `solveEikonal` kernel and later used in the `solveDynamicEq` kernel, as seen in Table 7.1. These variables are in fact temporary variables and storing them in global memory produces an unnecessary performance hit. By merging the code from these two kernels into a single one, and by keeping said temporary variables in private memory, the amount of allocated global memory could be reduced by almost half. More significant than the reduction in global memory usage would be the increase in performance gained from eliminating this low bandwidth bottleneck.
- Currently, the model relies on the user to provide a workgroup size, thus defining how many threads are assigned to each compute unit.

Variable name	Type	Used in Kernel `solveEikonalEq`	`solveDynamicEq`	`calcPressure`
`x`	float	✓	✓	✓
`y`	float	✓	✓	✓
`tau`	float	✓	-	✓
`amp`	complex	-	✓	✓
`decay`	complex	✓	✓	-
`phase`	float	✓	-	✓
`s`	float	✓	✓	-
`ic`	float	✓	✓	-
`caustc`	float	-	✓	✓
`c`	float	✓	✓	✓[1]
`p`	float	-	✓	-
`q`	float	-	✓	✓
`boundaryTg`	vector	✓	✓	-
`interfaceID`	char	✓	✓	-

Table 7.1: *Usage of global variables in kernels; highlighted variables are temporary variables which don't require global storage.*

Choosing this value has a big impact on performance, specially on GPUs, and will vary between devices. It may be desirable to automate this in order to simplify usage of the model by removing the need for the user to have an understanding of the GPU hardware.

- Currently, the complexity of modeled waveguides is limited by the size of environmental information, in other words, by the amount of available local memory. In order to model larger or more complex waveguides, it would be desirable to automatically subdivide the environment into smaller parts which will fit into the available memory. While ray-tracing, rays which leave one "subenvironment" must then be passed to the adjacent subenvironment and their propagation be resumed.

  Besides enabling larger or more complex waveguides, this would also be a step in implementing support for multiple bottom layers and transmissive objects.

- Finally, the model might be expanded to add support for more output

options, like transmission loss or particle velocity.

[1]The interpolated sound speed at source position, $\texttt{c}_0$, is used only once at the beginning of `calcPressure`. Since no other indices of this vector are accessed, this kernel doesn't require full access to this vector; passing $\texttt{c}_0$ as a scalar would suffice.

Bibliography

[1] Orlando Camargo Rodríguez, Jon M. Collis, Harry J. Simpson, Emanuel Ey, Joseph Schneiderwind, and Paulo Felisberto. Seismo-acoustic ray model benchmarking against experimental tank data. *The Journal of the Acoustical Society of America*, 132(2):709–717, 2012.

[2] Orlando Rodríguez, Paulo Felisberto, Emanuel Ey, Joseph Schneiderwind, and Sergio M. Jesus. Vector sensor geoacoustic estimation with standard arrays. volume 17, page 070087. ASA, 2013.

[3] Paulo Felisberto, Orlando C. Rodríguez, Paulo Santos, Emanuel Ey, and Sergio M. Jesus. Experimental results of underwater cooperative source localization using a single acoustic vector sensor. *Sensors*, 13(7):8856–8878, 2013.

[4] European Commission. Commission decision on criteria and methodological standards on good environmental status of marine waters. Official Journal of the European Union, September 2010.

[5] Marsensing, Lda. Shipping noise underwater acoustic map. http://www.shippingnoise.com. Retrieved 26 September 2013.

[6] Finn B. Jensen, William A. Kuperman, Michael B. Porter, and Henrik Schmidt. *Computational Ocean Acoustics*. American institute of Physics, Woodbury, New York, 1994.

[7] Michael J. Buckingham. Ocean-acoustic propagation models. technical report EUR 13810. Technical report, Comission of the European Comunities, 1991.

[8] Orlando C. Rodríguez. *The TRACEO ray tracing program.* Universidade do Algarve - Signal Processing Laboratory, 2010.

[9] Porter M.B. and Bucker H.P. Gaussian beam tracing for computing ocean acoustic fields. *J. Acoust. Soc. America*, 82(4):1349–1359, 1987.

[10] Michael Porter. The kraken normal mode program. Technical report, SACLANT UNDERSEA RESEARCH CENTRE, 1994.

[11] P.J. Papadakis, M.I. Taroudakis, and J.S. Papadakis. Recovery of the properties of an elastic bottom using reflection coefficient measurements. In *Proceedings of the 2nd European Conference on Underwater Acoustics*, volume II, pages 943–948, Copenhagen, Denmark, 1994.

[12] The Khronos Group. Homepage. http://www.khronos.org/. Retrieved 05 May 2013.

[13] The Khronos Group. *Heterogeneous Parallel Computing In HTML5 Web Browsers.* http://www.khronos.org/webcl/. Retrieved 25 September 2013.

[14] Aaftab Munshi, Benedict R. Gastner, Timothy G. Mattson, James Fung, and Dan Ginsburg. *OpenCL Programming Guide.* Addison-Wesley, Boston, US, December 2012.

Appendix A

Quick User Guide to clTraceo

A.1 Usage

When running clTraceo an OpenCL device has to be chosen. This is done by passing the command line switch '-d' followed by the device number, like so:

```
$> cltraceo -d 0 munk
```

To see the list of available opencl devices, the model should be run with the '–listDevices' option:

```
$> cltraceo --listDevices
```

Some devices (like GPUs) require choosing a custom workgroup size in order to get the best performance. This is done by using the '–wgSize' option:

```
$> cltraceo -d 0 --wgSize 16 munk
```

Some notes on workgroups:

- Workgroup sizes should be powers of 2.
- Default wgSize is 1.
- Changing workgroup sizes on CPUs does not seem to yield improvements.

- Best results on NVidia GTX 580 GPU are obtained with wgSize 16.
- In general -on the GPU- the larger the workgroup size, the better the performance (wgSize 16 results in 2-3× better performance than wgSize 1)
- If an 'out of resources' error occurs, then the chosen workgroup size is to big -try lowering it.

By default, the part of the model which is run on the OpenCL compute device has to be compiled every time the model is run, which causes a large run time overhead. To avoid this, the compiled device code can be stored in binary form for use in the next run by making use of the '–saveBinKernel' and '–loadBinKernel' options:

```
$> cltraceo -d 1 --saveBinKernel filename.bin munk
$> cltraceo -d 1 --loadBinKernel filename.bin munk
```

Note that changing some variables of the modeled environment requires recompiling the kernels. These are:

- the number of points in the altimetry;
- the altimetry's type, i.e., if it is homogeneous or non-homogeneous;
- the sound speed distribuition type and class;
- the number of points in the sound speed field/profile;
- the number of points in the bathymetry;
- the bathymetry's type, i.e., if it is homogeneous or non-homogeneous;
- the number of ray coordinates (which depend on the size of the rangge box and the the integration step size);
- the type of hydrophone array;
- the number of hydrophone's in the array;

Also:

- the workgroup size;
- the assumed hardware vector dimension;

In short, any change which would require the model to allocate a different amount of memory. In other words, between model runs, all values of the environment may change as long as it's basic configurations stays the same. This should not be a restriction for most inversion scenarios.

The run time information which was shown at the end of cTraceo has been removed as it's output is not correct when using OpenCL. For performance comparisons, the linux "time" command may be used to obtain run time measures:

```
$> time cltraceo -d 0 flat
```

A.2 Troubleshooting

Running clTraceo with the AMD APP SDK v2.8 the following message may be shown:

```
FATAL: Module fglrx not found.
Error! Fail to load fglrx kernel module! Maybe you can
   switch to root user to load kernel module directly
```

This error occurs on machines using AMD OpenCL SDK, which do not have an AMD GPU, i.e., when running on a CPU. This error is related to a driver bug and has no functional impact on the model; the code runs as expected.

Another possible error is:

```
Setting of real/effective user Id to 0/0 failed
WARNING: Deprecated config file /etc/modprobe.conf, all
   config files belong into /etc/modprobe.d/.
```

As in the previous case, this error is driver related and has no functional impact on the model; the code runs as expected.

```
Segmentation fault.
```

On machines accessed remotely via ssh, segmentation faults are likely to occur when using `ssh -X` or `ssh -Y`. This is due to the fact that the `-X` and `-Y` options connect the local system's X server session to the remote system's X server which in turn causes problems with OpenCL drivers.

```
Failed to find any OpenCL platform.
```

On some machines, calling clTraceo from within Matlab will sometimes cause clTraceo to return this error message. This is similar to the previous situation where calling clTraceo over an ssh connection caused an error. This problem can be circumvented by running Matlab without a graphical interface from a linux terminal:
`$> matlab -nojvm` It is currently unknown if this issue occurs on windows machines.

A.3 Compilation

To compile the clTraceo code the first requirement will be to install an OpenCL driver for the system.

For testing on a multicore CPU, the AMD OpenCL driver is recomended[1]. This will work on both Intel and AMD systems and actually delivers better results on Intel systems than Intel's own driver.

Next the makefile will need to be adapted along with the `SOURCE_PATH` definition in the `source/constants.h` file.

Compilation should then be straight forward.

[1]The AMD OpenCL's marketing name is *AMD Accelerated Parallel Processing SDK* (or *AMD APP SDK*) and it's available at http://developer.amd.com/tools-and-sdks/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk/downloads/

Appendix B

Constant Values

Throughout the model's source code, there are several variables which may only take a specific set of values. These values are defined in the `constants.h` file which is part of the source distribution and is reproduced in part below.

The `interfaceType` variable is part of the `interface_t` struct and it's possible values are:

```
#define INTERFACE_TYPE__ABSORVENT 1
#define INTERFACE_TYPE__ELASTIC   2
#define INTERFACE_TYPE__RIGID     3
#define INTERFACE_TYPE__VACUUM    4
```

The `interfacePropertyType` variable is part of the `interface_t` struct and it's possible values are:

```
#define INTERFACE_PROPERTY_TYPE__HOMOGENEOUS      5
#define INTERFACE_PROPERTY_TYPE__NON_HOMOGENEOUS  6
```

The `interfaceInterpolation` variable is part of the `interface_t` struct and it's possible values are:

```
#define INTERFACE_INTERPOLATION__FLAT     7
#define INTERFACE_INTERPOLATION__SLOPED   8
#define INTERFACE_INTERPOLATION__2P       9
#define INTERFACE_INTERPOLATION__3P       10
#define INTERFACE_INTERPOLATION__4P       11
```

The `interfaceAttenUnits` variable is part of the `interface_t` struct and it's possible values are:

```
#define INTERFACE_ATTEN_UNITS__dBperkHz       12
#define INTERFACE_ATTEN_UNITS__dBperMeter     13
#define INTERFACE_ATTEN_UNITS__dBperNeper     14
#define INTERFACE_ATTEN_UNITS__qFactor        15
#define INTERFACE_ATTEN_UNITS__dBperLambda    16
```

The `cDist` variable is part of the `soundSpeed_t` struct and it's possible values are:

```
#define C_DIST__PROFILE     17
#define C_DIST__FIELD       18
```

The `cClass` variable is part of the `soundSpeed_t` struct and it's possible values are:

```
#define C_CLASS__ISOVELOCITY        19
#define C_CLASS__LINEAR             20
#define C_CLASS__PARABOLIC          21
#define C_CLASS__EXPONENTIAL        22
#define C_CLASS__N2_LINEAR          23
#define C_CLASS__INV_SQUARE         24
#define C_CLASS__MUNK               25
#define C_CLASS__TABULATED          26
```

The `calcType` variable defines the model's chosen output option and it's possible values are:

```
#define CALC_TYPE__RAY_COORDS               27
#define CALC_TYPE__ALL_RAY_INFO             28
#define CALC_TYPE__EIGENRAYS_REG_FALSI      29
#define CALC_TYPE__EIGENRAYS_PROXIMITY      30
#define CALC_TYPE__AMP_DELAY_REG_FALSI      31
#define CALC_TYPE__AMP_DELAY_PROXIMITY      32
#define CALC_TYPE__COH_ACOUS_PRESS          33
#define CALC_TYPE__COH_TRANS_LOSS           34
#define CALC_TYPE__PART_VEL                 35
#define CALC_TYPE__COH_ACOUS_PRESS_PART_VEL 36
```

The `arrayType` variable sets the type of hydrophone array to be used in a given waveguide.

```
#define ARRAY_TYPE__RECTANGULAR     37
#define ARRAY_TYPE__HORIZONTAL      38
#define ARRAY_TYPE__VERTICAL        39
#define ARRAY_TYPE__LINEAR          40
```

Appendix C

Published Papers

C.1 June 2012

Seismo-acoustic ray model benchmarking against experimental tank data

Orlando Camargo Rodríguez[a)]
LARSyS, Campus de Gambelas, Universidade do Algarve, PT-8005-139 Faro, Portugal

Jon M. Collis
Department of Applied Mathematics and Statistics, Colorado School of Mines, 1500 Illinois Street, Golden, Colorado 80401

Harry J. Simpson
Physical Acoustic Branch Code 7136, Naval Research Laboratory, 4555 Overlook Avenue Southwest, Washington, DC 20375

Emanuel Ey, Joseph Schneiderwind, and Paulo Felisberto
LARSyS, Campus de Gambelas, Universidade do Algarve, PT-8005-139 Faro, Portugal

(Received 1 February 2012; revised 18 June 2012; accepted 21 June 2012)

Acoustic predictions of the recently developed TRACEO ray model, which accounts for bottom shear properties, are benchmarked against tank experimental data from the EPEE-1 and EPEE-2 (Elastic Parabolic Equation Experiment) experiments. Both experiments are representative of signal propagation in a Pekeris-like shallow-water waveguide over a non-flat isotropic elastic bottom, where significant interaction of the signal with the bottom can be expected. The benchmarks show, in particular, that the ray model can be as accurate as a parabolic approximation model benchmarked in similar conditions. The results of benchmarking are important, on one side, as a preliminary experimental validation of the model and, on the other side, demonstrates the reliability of the ray approach for seismo-acoustic applications.
 [http://dx.doi.org/10.1121/1.4734236]

PACS number(s): 43.30.Zk, 43.30.Ma, 43.20.Dk [AIT] Pages: 709–717

I. INTRODUCTION

Tank experiments constitute a fundamental reference for underwater acoustic modeling, by providing valuable data for model benchmarking. In particular these types of experiments are important because of the difficulties and costs involved with obtaining high-quality ocean acoustic data at sea. In contrast with benchmarking analytic solutions,[1–5] which are generally difficult to obtain for a broad set of geometries and boundary properties, benchmarking to tank experimental data imposes important constraints to numerical models. On one side, propagation conditions can be carefully controlled; on the other side, despite such control, mismatch will be always observed since the numerical model is an approximation of the theoretical solution. The importance of benchmarking to tank experimental data was shown, for example, by the EPEE-1 (Ref. 6) and EPEE-2 (Elastic Parabolic Equation Experiment) (Ref. 7) experiments, which demonstrated the excellent accuracy of the ROTVARS model, based on the variable rotated elastic parabolic equation.[8] The high quality of the data acquired during these tank experiments is extremely important because both experiments are representative of propagation in a shallow water waveguide, with an elastic bottom and range-dependent bathymetry involving sharp slope changes. Ray models[9–12] are also interesting candidates for benchmarking against the tank experimental data. The ray solution to the acoustic wave equation is an asymptotic approximation, which improves as frequency increases, and ray methods are computationally efficient in waveguides with complex characteristics, such as variable boundaries and range-independent or range-dependent sound speed distributions. Additionally, for a ray model to be accurate under such conditions, shear effects need to be included as well. Naturally, a question arises whether a ray model will be able to exhibit the same degree of accuracy as a parabolic equation solution, when benchmarked (in particular) to the data of the EPEE-1 and EPEE-2 tank experiments. The main purpose of the discussion presented here is to develop a systematic benchmarking of a ray model against such experimental data. To this end the tank experiments are briefly reviewed in Sec. II, while Sec. III describes the recently developed TRACEO ray model, which is benchmarked in detail in Sec. IV. The conclusions of benchmarking and future work are presented in Sec. V.

II. THE EPEE-1 AND EPEE-2 TANK EXPERIMENTS

The tank experiments are described in great detail in the literature;[6,7] therefore, a sufficiently compact description is presented in this section. Polyvinyl chloride (PVC) slabs with the elastic parameters given in Table I were suspended in a water tank by cables, that were attached to each slab corner at substantial distances from the sound source to avoid reflections. Source and receiver hydrophones were positioned over the slabs with a robotic arm, allowing for accurate positioning. Sound speed in the water is considered constant and corresponds to 1482 m/s. Acoustic

[a)]Author to whom correspondence should be addressed. Electronic mail: orodrig@ualg.pt

TABLE I. PVC elastic properties at 300 kHz (λ represents the acoustic wavelength).

Parameter	Units[a]	Value
Density	kg/m^3	1378
Compressional speed	m/s	2290
Shear speed	m/s	1050
Compressional attenuation	dB/λ	0.76
Shear attenuation	dB/λ	1.05

[a]A note of advice: Compressional and shear attenuations are given in Ref. 6 as 0.33 dB/m and 1.00 dB/m, respectively. Attenuation is given here in dB/λ, which are the units used by the ROTVARS model.

transmissions were performed for a wide set of frequencies, up to 1 MHz, but for the purposes of benchmarking only three frequencies are here considered, namely, 125, 200, and 275 kHz. Due to the nature of the source used in the experiment, data within the 100–300 kHz band are considered valid for comparison purposes.

Acoustic propagation calculations are performed at a scale of 1000:1; thus, for a proper modification of parameter values the following conversion of units is adopted: experimental frequencies in kHz become model frequencies in Hz and experimental lengths in mm become model lengths in m (for instance, an experimental frequency of 100 kHz becomes a model frequency of 100 Hz and an experimental distance of 10 mm becomes a model distance of 10 m). Sound speeds remain unchanged, as well as compressional and shear attenuations, which are given in dB/λ (where λ stands for the acoustic wavelength). In EPEE-1 the slab allowed both range-independent, "flat," and range-dependent waveguides (see Fig. 1). In EPEE-2 the slab geometry allowed three different types of range-dependent bottom bathymetries, namely, flat to downslope, upslope to flat, and upslope to downslope. Different configurations of the acoustic source and receiver were considered in both experiments, but the benchmarking presented here will be limited to a single position of both source and receiver; geometric parameters for the waveguides of both experiments are shown in Table II.

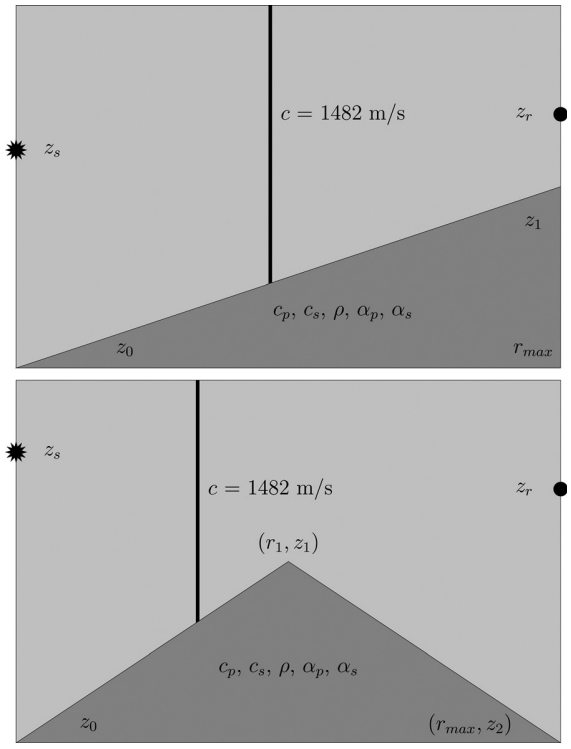


FIG. 1. EPEE-1, sloped case (top) and EPEE-2, upslope to downslope case (bottom).

III. THE RAY MODEL

The ray model benchmarked in the current work is the TRACEO Gaussian beam model, which is under current development at the SiPLAB of the University of Algarve.[13] The code TRACEO was developed in order to

(1) Predict acoustic pressure and particle velocity in environments with elaborate upper and lower boundaries, which can be characterized by range-dependent compressional and shear properties. Modeling particle velocity is important for vector sensor applications and can be used, in particular, for geoacoustic inversion with high frequency data.[14–16]
(2) Include one or more targets in the waveguide.
(3) Produce ray, eigenray, amplitude, and travel time information. In particular, eigenrays are to be calculated even if rays are reflected backwards on targets located beyond the current position of the hydrophone.

The following sections compactly describe the theory behind TRACEO calculations of acoustic pressure, shear inclusion and particle velocity calculations; a numerical example is presented as well.

A. Theoretical background

The starting point for the general description of a three-dimensional Gaussian beam is given by the expression[17]

$$P(s,\boldsymbol{n}) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{\det \boldsymbol{Q}}}\exp\left\{-i\omega\left[\tau(s)+\frac{1}{2}(\boldsymbol{M}n\cdot\boldsymbol{n})\right]\right\}, \tag{1}$$

where s stands for the ray arclength, $\boldsymbol{n}$ is the normal to the ray (such normal lies on a plane, which will be introduced later), $\boldsymbol{M}$ and $\boldsymbol{Q}$ are 2×2 matrices (whose meaning will be explained below), the center dot represents the inner vector

TABLE II. Geometric parameters for the waveguides of the EPEE-1 and EPEE-2 tank experiments.

	Flat	Upslope	Flat/down	Up/flat	Up/down
z_s (m)	69.1	63.4	74.2	73.5	72.1
z_r (m)	137.1	15.6	75.8	78.7	74.8
z_0 (m)	144.7	132.9	143.9	245.9	183.0
z_1 (m)	145.4	45.4	152.1	151.2	147.2
z_2 (m)	N/A	N/A	238.2	145.7	198.4
r_1 (m)	N/A	N/A	988.8	985.9	1000.3
$r_{\max}$ (m)	1200	1200	1898.0	1898.0	1898.0

product, $\theta(0)$ is the initial ray elevation (i.e., the angle relative to the horizontal plane, which is formed by the X and Y axes; the angle on the XY plane relative to the X axis will be called, hereafter, the azimuth), and $c(s)$ represents the sound speed along the ray trajectory:

$$c(s) = c(s,\,0)\,. \tag{2}$$

The travel time $\tau(s)$ in Eq. (1) is calculated solving the set of Eikonal equations[18]

$$\begin{aligned} &\frac{dx}{ds} = c(s)\sigma_x, \quad \frac{dy}{ds} = c(s)\sigma_y, \quad \frac{dz}{ds} = c(s)\sigma_z, \\ &\frac{d\sigma_x}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial x}, \quad \frac{d\sigma_y}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial y}, \quad \frac{d\sigma_z}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial z}, \end{aligned} \tag{3}$$

where σ_x, σ_y, and σ_z stand for the components of the vector of sound slowness. The derivatives dx/ds, dy/ds, and dz/ds define the ray tangent $\boldsymbol{e}_s$; the plane perpendicular to $\boldsymbol{e}_s$ defines the plane normal to the ray. Introducing on such plane a pair of unitary and orthogonal vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ one can write the ray normal as

$$\boldsymbol{n} = n_1\boldsymbol{e}_1 + n_2\boldsymbol{e}_2\,, \tag{4}$$

where n_1 and n_2 are arbitrary quantities. The matrices $\boldsymbol{M}$ and $\boldsymbol{Q}$ are required to be complex; therefore, the imaginary part of the product $\boldsymbol{M}n\cdot\boldsymbol{n}$ induces a Gaussian decay of beam amplitude along $\boldsymbol{n}$, while the real part introduces phase corrections to the travel time. As long as $\det\boldsymbol{Q} \neq 0$ the solution given by Eq. (1) does not exhibit singularities. Besides $\boldsymbol{Q}$ and $\boldsymbol{M}$ the Gaussian beam approximation involves two additional 2×2 matrices, represented generally as $\boldsymbol{P}$ and $\boldsymbol{C}$; all four matrices are related through the following relationships:[19]

$$\boldsymbol{M} = \boldsymbol{P}\boldsymbol{Q}^{-1}\,, \tag{5}$$

$$\frac{d}{ds}\boldsymbol{Q} = c(s)\boldsymbol{P}, \quad \frac{d}{ds}\boldsymbol{P} = -\frac{1}{c^2(s)}\boldsymbol{C}Q, \tag{6}$$

where

$$C_{ij} = \frac{\partial^2 c}{\partial n_i \partial n_j}\,, \tag{7}$$

i.e., the elements of $\boldsymbol{C}$ correspond to second order derivatives of sound speed along either $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, or both. Generally speaking $\boldsymbol{P}$ describes the beam slowness in the plane perpendicular to $\boldsymbol{e}_s$, while $\boldsymbol{Q}$ describes the beam spreading. The pair of expressions given by Eq. (6) is called the *dynamic equations* of the full Gaussian beam formulation. The expression given by Eq. (1) behaves near the source like an spherical wave emitted by a point source through the choice of initial conditions[19]

$$\boldsymbol{P}(0) = \begin{bmatrix} 1 & 0 \\ 0 & \cos\theta(0) \end{bmatrix} \Big/ c(0) \tag{8}$$

and[17]

$$\boldsymbol{Q}(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{9}$$

Generally speaking the full Gaussian beam approach is difficult to implement numerically, with the main difficulties being related to refraction effects (the problem of ray boundary reflection is in fact much easier to account for). In particular, when horizontal refraction is considered, rays with a common initial azimuth exhibit ray trajectories which do not lie on a common plane; besides, horizontal refraction also leads to the rotation of polarization vectors along a given ray trajectory, inducing a large variability of beam shapes within any group of rays (even if the initial orientation of polarization vectors was the same for all rays). Additionally, the calculation of beam influence (which requires a proper calculation of the matrix $\boldsymbol{C}$) for the arbitrary position of an hydrophone is cumbersome. Other issues related to the calculation of eigenrays, such as determining arrivals and respective amplitudes, are also difficult to implement. In order to develop a two-dimensional application of Eq. (1), TRACEO relies on the particular solution of dynamic and Eikonal equations when horizontal refraction is absent. For such case there is no rotation of the polarization vectors; thus, choosing $\boldsymbol{e}_2$ to lie on the horizontal plane fixes the positioning of $\boldsymbol{e}_1$ on a plane of constant azimuth (perpendicular to $\boldsymbol{e}_2$). Beam amplitude is then calculated by TRACEO on the plane of constant azimuth by considering the particular solution with $n_2 = 0$. Since the particular solution does not exhibit cylindrical symmetry the approximation used by TRACEO can be regarded as a Gaussian beam solution on the (x, z) plane (i.e., on the plane corresponding to azimuth zero).

Under the conditions above considered (i.e., absence of horizontal refraction and $\boldsymbol{e}_2$ placed initially on the horizontal plane) one can write that

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & 0 \\ 0 & 0 \end{bmatrix}. \tag{10}$$

Without loss of generality the matrix $\boldsymbol{Q}$ can be represented as

$$\boldsymbol{Q}(s) = \begin{bmatrix} q(s) & 0 \\ 0 & q_\perp(s) \end{bmatrix} \tag{11}$$

so $\det\boldsymbol{Q} = q(s)q_\perp(s)$. Combining the initial conditions for a point source with the given approximations one can substitute the pair Eq. (6) with the expressions

$$\frac{d}{ds}q = c(s)\,, \quad \frac{d}{ds}p = -\frac{c_{11}}{c(s)^2}\,q, \tag{12}$$

where $p = p_{11}$ [it can be shown that $p_{12} = p_{21} = 0$, $p_{22} = \cos\theta(0)/c(0)$]; thus, the particular solution of Eq. (1) can be written as

$$\begin{aligned} P(s, n_1, n_2) = &\frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{q_\perp(s)q(s)}} \\ &\times \exp\left[-i\omega\left(\tau(s) + \frac{1}{2}\frac{p(s)}{q(s)}n_1^2 + \frac{1}{2I_c(s)}n_2^2\right)\right], \end{aligned} \tag{13}$$

where

$$I_c(s) = \int_0^s c(s')\, ds' \tag{14}$$

and

$$q_\perp(s) = \frac{\cos\theta(0)}{c(0)} I_c(s). \tag{15}$$

Taking $n_2 = 0$ in Eq. (13) and representing n_1 simply as n one can write the solution on the plane of constant azimuth as

$$P(s,n) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{q_\perp(s)q(s)}} \times \exp\left[-i\omega\left(\tau(s) + \frac{1}{2}\frac{p(s)}{q(s)}n^2\right)\right]. \tag{16}$$

Equation (16) is similar to the Gaussian beam expression for a waveguide with cylindrical symmetry[18,20]

$$P(s,n) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{rq(s)}}\exp\left[-i\omega\left(\tau(s) + \frac{1}{2}\frac{p(s)}{q(s)}n^2\right)\right]. \tag{17}$$

The term $1/\sqrt{r}$ appears in Eq. (17) due to the cylindrical spreading of the pressure field, and the expression itself is an asymptotic solution of the wave equation, which breaks down at the source position. Therefore, a "blind" numerical application of Eq. (17) to rays propagating back to the source produces waves, which are focused back to the source and break down in its vicinity. When compared to Eq. (17) one can notice that Eq. (16) contains the parameter $q_\perp(s)$ instead of r; since $q_\perp(s)$ is proportional to the ray arclength, beam amplitudes given by Eq. (16) always decrease independently of rays propagating forwards or backwards. This feature of Eq. (16) is expected to be relevant for backscattering studies.

The field given by Eq. (16) is not sufficient for TRACEO to account properly for phase and amplitude corrections every time a ray hits a boundary; in such case the beam amplitude is multiplied by a boundary reflection coefficient, which takes into account shear speed and shear attenuation;[21] the full expression of such reflection coefficient is presented in Appendix A. Additional corrections to the ray amplitude are introduced using finite element ray tracing[22] and phase corrections induced by caustics (which are described in detail in Ref. 18).

To understand the method implemented in TRACEO for particle velocity calculations let us recall that particle velocity $\boldsymbol{v}$ is related to acoustic pressure P in the frequency domain through the relationship[23]

$$\boldsymbol{v} = -\frac{i}{\omega\rho}\nabla P, \tag{18}$$

where ρ represents the watercolumn density, and ω stands for the frequency of the propagating wave. The factors ρ and ω only affect the amplitude of $\boldsymbol{v}$, while the imaginary unit implies a phase shift of $\pi/2$ radians. Without these factors particle velocity can be viewed as the gradient of acoustic pressure. To obtain this gradient, TRACEO calculates the acoustic pressure on a star-shaped stencil, with the hydrophone located at the star's center; outer points are located at the coordinates $(r \pm \Delta, z \pm \Delta)$. The points aligned along the horizontal are used to calculate the coefficients of a parabolic interpolator (described in Appendix B), and those coefficients allow determination of the horizontal derivative at the center; a similar procedure is followed for the points aligned along the vertical. To avoid aliasing, the spacing between an outer point and the center is taken (arbitrarily) as corresponding to $\Delta = \lambda/10$, where λ represents the acoustic wavelength. Interpolation is preferred to analytic expressions derived from Eq. (16) or Eq. (17) because either of them is written in terms of ray coordinates (s, n), instead of horizontal and vertical coordinates. Such analytic expressions are elaborate and can be used only with a Gaussian beam model, while the interpolation approach is valid for any model and can easily be extended to three dimensions.

B. Numerical example

The capabilities of TRACEO require intense testing through comparisons with other models, a discussion of backscattering issues in more detail and comparison between experimental data and field predictions when targets are present in the water column (just to mention a few further directions of research). Such issues go far beyond the main goals of the discussion presented here and will be addressed in future studies. The numerical example in this section is limited to a comparison of TRACEO predictions of the horizontal and vertical components of particle velocity (hereafter represented as u and w, respectively), with the corresponding values found from analytic expressions for an elastic bottom Pekeris waveguide (shear included); the compressional and shear potentials of such a waveguide are well known in the literature,[24] and the particle velocity components are easily calculated from the analytic expressions for both potentials. It is worth remarking that, from the point of view of normal modes, the contribution of lower-order modes is enhanced in the field of u, while the field of w is enhanced by the contribution of higher-order modes.[25] The comparison between particle velocity calculations from the analytic solution and TRACEO predictions for the flat waveguide (200 Hz, $z_0 = z_1 = 145$ m) is shown in Fig. 2; in both cases the analytic solution is indicated by the solid curve, while the model prediction is indicated by the dashed curve. The interpolation approach is so accurate in the case of u that the two curves are difficult to distinguish [Fig. 2(a)]. The approach is less accurate for w, with the model exhibiting some overshooting of the analytic solution [Fig. 2(b)], although it does correctly reproduce the interference pattern in both phase and amplitude in range.

IV. BENCHMARKING

Seismo-acoustic benchmarking of the ray solution against tank data is discussed in this section through a

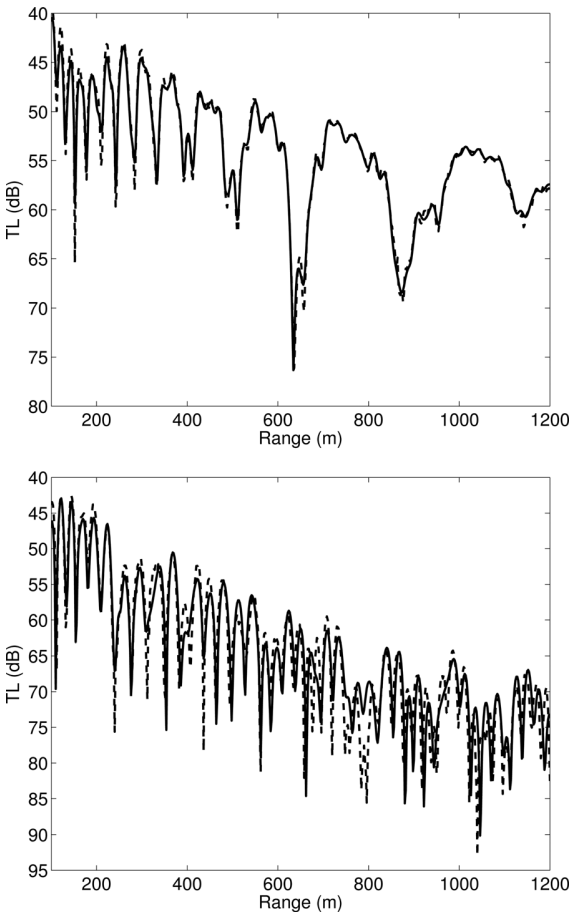


FIG. 2. Particle velocity calculations at 200 Hz for the flat waveguide ($z_0 = z_1 = 145$ m): u (top); w (bottom). Solid curve: analytic solution; dashed curve: TRACEO's prediction (compare with the middle plot of Fig. 3).

systematic set of comparisons with transmission loss curves calculated from tank experimental data at model frequencies of 125, 200, and 275 Hz. Comparisons against EPEE-1 data are discussed in Secs. IV A and IV B for the flat and upslope waveguides, while comparisons againts EPEE-2 data for the flat to downslope, upslope to flat, and upslope to downslope waveguides are discussed in Secs. IV C–IV E. Results from Secs. IV A and IV B can be related to Figs. 4 and 7 from Ref. 6, respectively; the results for the EPEE-2 data concern source-receiver configurations not discussed in Ref. 7. In all sections, tank data is shown in the figures as a solid curve, while the dashed curve indicates TRACEO predictions; additionally, the plots are arranged with frequency increasing from top to bottom; geometries (source depth, receiver depth, and source-receiver range) are all given in model values. It is worth remarking that the benchmarking was not limited to the mentioned set of frequencies. Additional comparisons were performed at model frequencies of 100 and 300 Hz, with no appreciable deviation from what was found at the chosen set of frequencies. Since no new information was provided by the benchmarking at such frequencies, comparisons are not included in the discussion. In order to produce TRACEO predictions as objectively as possible the following procedure was followed: the number of rays was taken, arbitrarily, as high as 201 rays, in order to ensure that field coherence was properly modeled at all frequencies. Source aperture corresponded to 55.25°; that value was obtained by minimizing the standard deviation over apertures in the interval [35°, 85°] of the difference between the experimental transmission loss and the model transmission loss for the flat waveguide at the "central" model frequency of 200 Hz. Thus, 201 rays between −55.25° and 55.25° were calculated by TRACEO at all frequencies, for all waveguides, and for all considered source/receiver configurations.

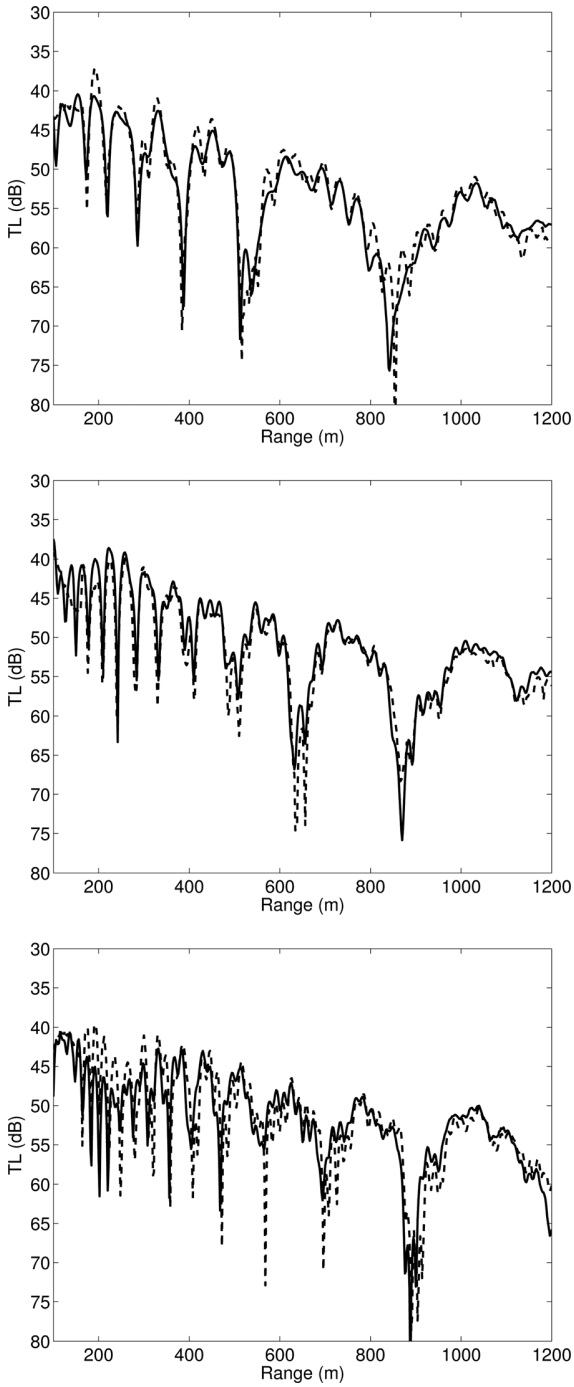


FIG. 3. Benchmarking for the flat waveguide, source at 69.1 m and receiver at 137.1 m: 125 Hz (top); 200 Hz (middle); 275 Hz (bottom). Solid curve: tank data; dashed curve: TRACEO's prediction (compare with Fig. 4 of Ref. 6).

A. Flat waveguide

Tank experiment and model curve comparisons for the flat waveguide are shown in Fig. 3. In all cases the ray model consistently and accurately reproduces the phase and amplitude behavior of experimental data, although occasional overshooting is observed. Despite the eventual limitations that one would expect from ray predictions at low frequencies the match between TRACEO and experimental data is remarkably accurate at 125 Hz. The results can be compared directly with Fig. 4 of Ref. 6 and indicate that for the given configuration the accuracy of both TRACEO and ROTVARS is nearly the same.

B. Upslope waveguide

Tank experiment data and model curve comparisons for the upslope waveguide are shown in Fig. 4. As one might

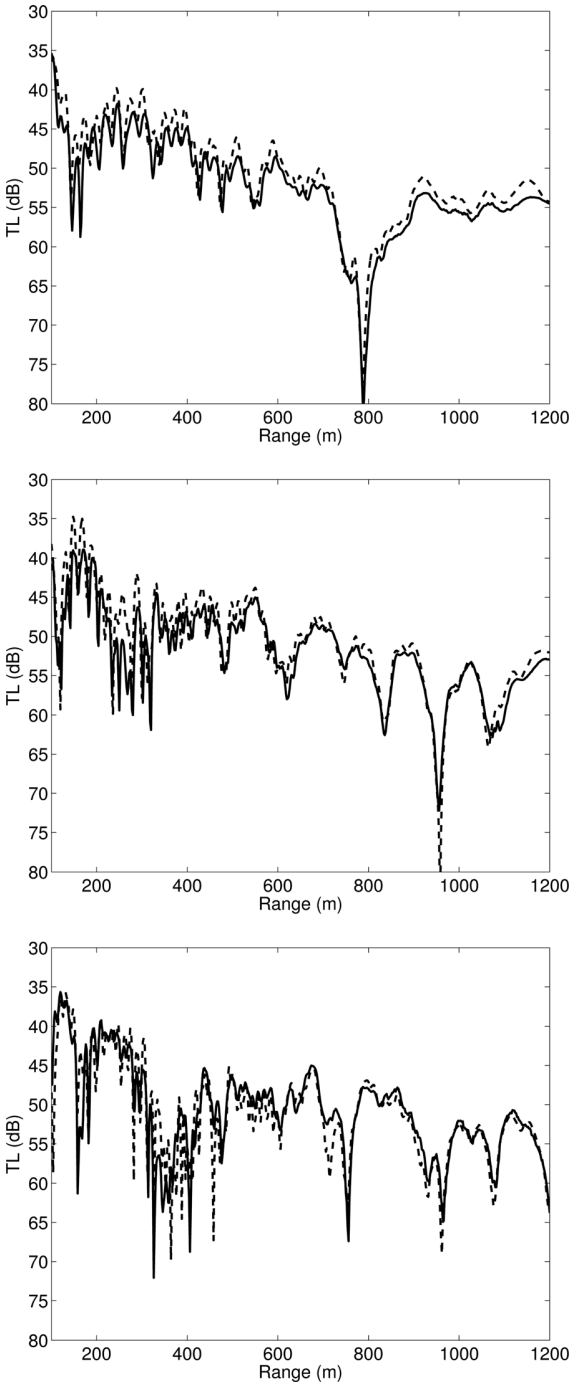


FIG. 4. Benchmarking for the upslope waveguide, source at 63.4 m and receiver at 15.6 m: 125 Hz (top); 200 Hz (middle); 275 Hz (bottom). Solid curve: tank data; dashed curve: TRACEO's prediction (compare with Fig. 7 of Ref. 6).

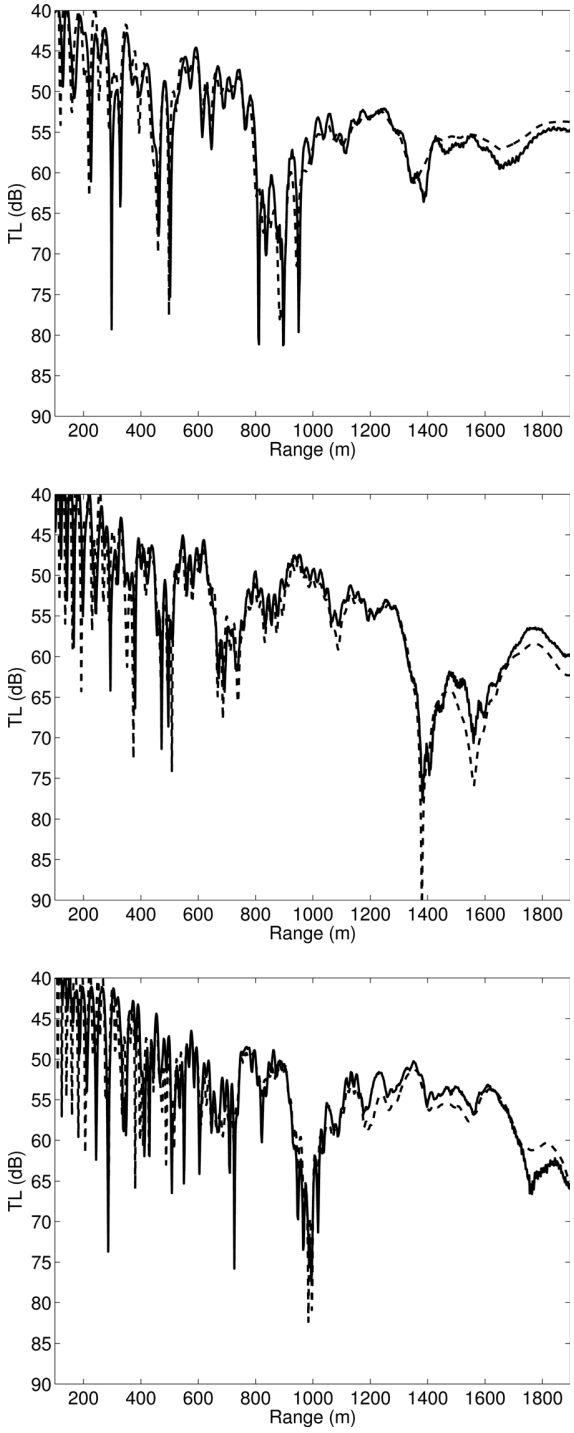


FIG. 5. Benchmarking for the flat to downslope waveguide, source at 74.2 m and receiver at 75.8 m: 125 Hz (top); 200 Hz (middle); 275 Hz (bottom). Solid curve: tank data; dashed curve: TRACEO's prediction (compare with Figs. 2 and 3 of Ref. 7).

expect ray results improve as frequency increases and TRACEO produces accurate predictions in all cases (with matches that at first glance appear even more accurate than those of the flat waveguide). The match is surprisingly good in many cases near the final ranges if one takes into account the existing bottom gap beyond the end of the PVC slab. The results can be compared directly with Fig. 7 of Ref. 6 and indicate, one more time, that for the given configuration there are no significant differences in the predictions produced by either TRACEO or ROTVARS.

C. Flat to downslope waveguide

Tank experiment data and model curve comparisons for the flat to dowslope waveguide are shown in Fig. 5. The figures reveal some undershooting or overshooting of the

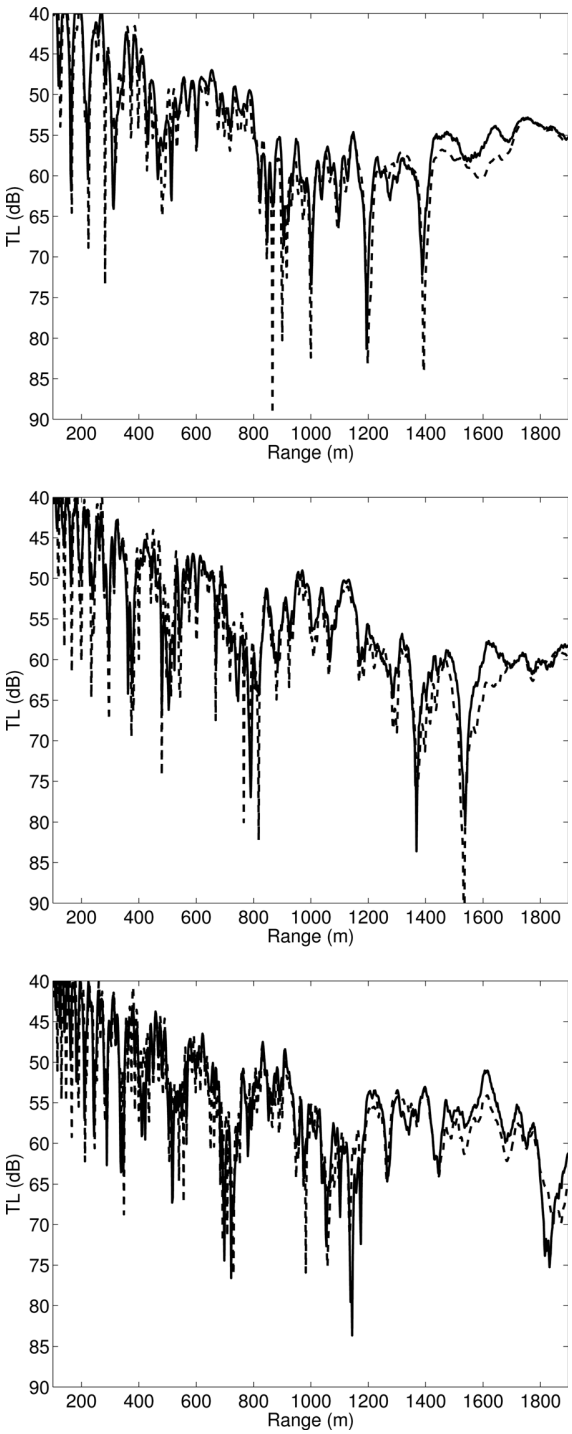


FIG. 6. Benchmarking for the upslope to flat waveguide, source at 73.5 m and receiver at 78.7 m: 125 Hz (top); 200 Hz (middle); 275 Hz (bottom). Solid curve: tank data; dashed curve: TRACEO's prediction (compare with Figs. 4 and 5 of Ref. 7).

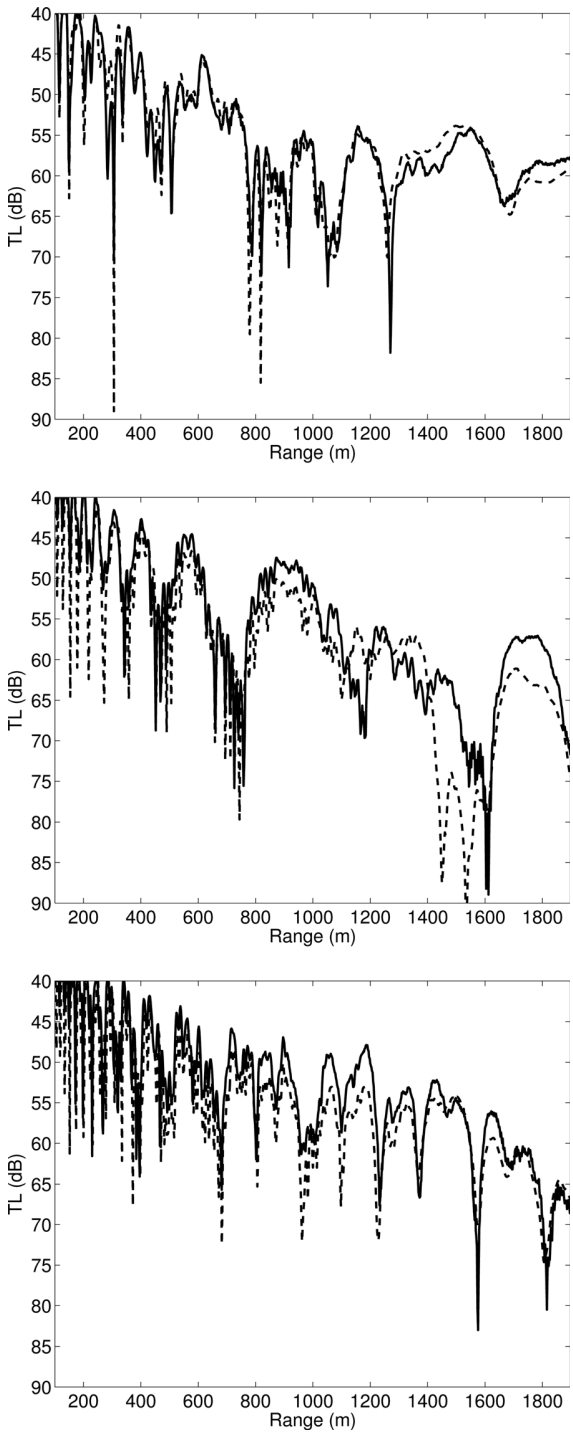


FIG. 7. Benchmarking for the upslope to downslope waveguide, source at 72.1 m and receiver at 74.8 m: 125 Hz (top); 200 Hz (middle); 275 Hz (bottom). Solid curve: tank data; dashed curve: TRACEO's prediction (compare with Fig. 6 of Ref. 7).

solution near the end of the slab, although the behavior is not consistent over frequency. Curiously the slight deviation of TRACEO's prediction from experimental data does not start at the range where the slab ends, but slightly beyond that range. Despite the slight mismatch, TRACEO properly reproduces the phase variations of the experimental data. A partial comparison can be done with Figs. 2 and 3 from Ref. 7 (the data is from the EPEE-2 experiment, but for different source and receiver depths), which indicates that TRACEO exhibits nearly the same accuracy as ROTVARS.

D. Upslope to flat waveguide

Tank experiment data and model curve comparisons for the upslope to flat waveguide are shown in Fig. 6. In this case the general trend of the ray model is to slightly undershoot the experimental data near the final ranges, although phase variations are again accurately reproduced by the numerical solution. A partial comparison can be done with Figs. 4 and 5 from Ref. 7 (the data are from the EPEE-2 experiment, but for different source and receiver depths), which shows no significant differences in the accuracy of either TRACEO or ROTVARS.

E. Upslope to downslope waveguide

Tank experiment and model curves for the upslope to downslope waveguide are shown in Fig. 7. This case could be considered to be the most difficult to simulate because of the significant change in bottom slope and it reveals in fact a set of less satisfactory matches to the experimental data. The comparisons are somehow intriguing for two reasons: first, one can notice that the mismatch is much more severe at the "central" model frequency of 200 Hz; second, the mismatch at 275 Hz is more severe near the middle of the waveguide than near the end. Additional comparisons with the RAMS model[26] (not shown here) produced a similar set of results. Curiously, Fig. 6 in Ref. 7 (which is related also to the EPEE-2 experiment, but for different source and receiver depths) exhibits a similar pattern.

V. CONCLUSIONS AND FUTURE WORK

The discussion presented in the previous sections demonstrated the feasibility of using a ray approach for seismo-acoustic studies related to acoustic propagation over isotropic elastic bottoms. Systematic benchmarking of the TRACEO ray model to experimental tank data, representative of propagation over elastic bottoms with sharp slope transitions exhibited a high degree of accuracy, comparable to the one already found with ROTVARS. As an experimental validation of the TRACEO model the results are extremely encouraging for further model applications, given the unique model features. Such applications can be oriented, for instance, to seismo-acoustic inversion of both compressional and shear bottom properties, using either standard hydrophone or vector sensor arrays, as long as the bottom does not exhibit a complex layered structure. Benchmarking results shown in Sec. IV indicate that such applications do not require necessarily to deal with high frequency propagation, since TRACEO exhibited a high accuracy at relatively low frequencies. A preliminary comparison of computational times between TRACEO and RAMS (which is widely available) on a typical laptop produced average values of 0.9 s vs 1.8 s, respectively, for the configurations of the EPEE-1 experiment, and of 1.8 s vs 5.2 s, respectively, for the configurations of the EPEE-2 experiment. Thus this particular trend shows TRACEO being faster than RAMS, although model parameters in the two cases were not optimized to minimize computational time without compromising accuracy. At high frequencies differences in computational times can be expected to become more relevant. Future directions of research will necessarily include detailed comparisons with field data (where mismatch can be expected to become more relevant), studies of backscattering issues (based on benchmarking against analytic solutions, backscattering-capable models, and/or available experimental data), accounting for ray tracing in elastic layered systems and three-dimensional field predictions where a ray approach looks like an attractive alternative for efficient and fast field computations.

ACKNOWLEDGMENTS

The authors are deeply thankful to the Naval Research Laboratory (NRL) for allowing the use of the experimental data discussed in this work. This work was supported by the Portuguese Foundation for Science and Technology through the SENSOCEAN project (FCT Contract No. PTDC/EEA-ELC/104561/2008).

APPENDIX A: RAY MODEL REFLECTION COEFFICIENT FOR AN ELASTIC BOTTOM

The calculation of the reflection coefficient for the elastic bottom is given by the following expression:[21]

$$R(\theta_1) = \frac{D(\theta_1)\cos\theta_1 - 1}{D(\theta_1)\cos\theta_1 + 1}, \tag{A1}$$

where

$$D(\theta_1) = A_1\left(A_2(1-A_7)\Big/\sqrt{1-A_6^2} + A_3A_7\Big/\sqrt{1-A_5/2}\right),$$

$$A_1 = \frac{\rho_2}{\rho_1}, \quad A_2 = \frac{\tilde{c}_{p2}}{c_{p1}}, \quad A_3 = \frac{\tilde{c}_{s2}}{c_{p1}},$$

$$A_4 = A_3\sin\theta_1, \quad A_5 = 2A_4^2, \quad A_6 = A_2\sin\theta_1,$$
$$A_7 = 2A_5 - A_5^2,$$

$$\tilde{c}_{p2} = c_{p2}\frac{1 - i\tilde{\alpha}_{cp}}{1 + \tilde{\alpha}_{cp}^2}, \quad \tilde{c}_{s2} = c_{s2}\frac{1 - i\tilde{\alpha}_{cs}}{1 + \tilde{\alpha}_{cs}^2},$$

$$\tilde{\alpha}_{cp} = \frac{\alpha_{cp}}{40\pi\log e}, \quad \tilde{\alpha}_{cs} = \frac{\alpha_{cs}}{40\pi\log e};$$

the units of attenuation should be given in dB/λ and the angle θ_1 is given relative to normal to the bottom (see Fig. 8). In general the reflection coefficient is real when $\alpha_{cp} = \alpha_{cs} = 0$, and the angle of incidence θ_1 is less than the critical angle θ_{cr}, with θ_{cr} given by the expression

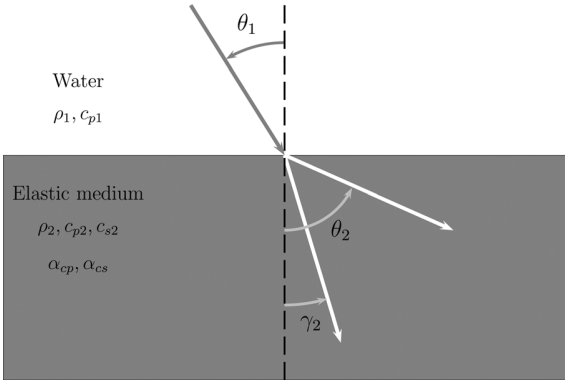


FIG. 8. Ray reflection at an elastic media interface.

$$\theta_{cr} = \arcsin\left(\frac{c_{p1}}{c_{p2}}\right). \tag{A2}$$

Attenuation can be expected to be negligible when $\theta_1 < \theta_{cr}$, and for small θ_1 the energy transferred to shear waves in the elastic medium is only a small fraction of the total energy transferred.

APPENDIX B: BARYCENTRIC PARABOLIC INTERPOLATION

The barycentric parabolic interpolator can be described as follows: let us consider a set of three points x_1, x_2, and x_3 aligned along the X axis, and the corresponding function values $f(x_1), f(x_2)$, and $f(x_3)$. At a given point x between x_1 and x_3 the interpolant can be written as

$$f(x) = f(x_1) + a_2(x - x_1)(x - x_3) + a_3(x - x_1)(x - x_2). \tag{B1}$$

It follows from this expression that

$$a_2 = \frac{f(x_2) - f(x_1)}{(x_2 - x_1)(x_2 - x_3)} \text{ and } a_3 = \frac{f(x_3) - f(x_1)}{(x_3 - x_1)(x_3 - x_2)}.$$

The approximations for the derivatives become

$$\frac{df}{dx} = a_2(2x - x_1 - x_3) + a_3(2x - x_1 - x_2)$$

and

$$\frac{d^2f}{dx^2} = 2(a_2 + a_3).$$

[1]F. Jensen and C. Ferla, "Numerical solution of range-dependent benchmark problems in ocean acoustics," J. Acoust. Soc. Am. **87**, 1499–1510 (1990).

[2]R. Stephen, "Solutions to range-dependent benchmark problems by the finite-difference method," J. Acoust. Soc. Am. **87**, 1527–1534 (1990).

[3]D. Thomson and G. Brooke, "Numerical implementation of a modal solution to a range-dependent benchmark problem," J. Acoust. Soc. Am. **87**, 1521–1526 (1990).

[4]E. Westwood, "Ray model solutions to the benchmark wedge problems," J. Acoust. Soc. Am. **87**, 1539–1545 (1990).

[5]M. Buckingham and A. Tolstoy, "An analytical solution for benchmark problem 1: The ideal wedge," J. Acoust. Soc. Am. **87**, 1511–1513 (1990).

[6]J. Collis, W. Siegmann, M. Collins, H. Simpson, and R. Soukup, "Comparison of simulations and data from a seismo-acoustic tank experiment," J. Acoust. Soc. Am. **122**, 1987–1993 (2007).

[7]H. Simpson, J. Collis, R. Soukup, M. Collins, and W. Siegmann, "Experimental testing of the variable rotated elastic parabolic equation," J. Acoust. Soc. Am. **130**, 2681–2686 (2011).

[8]D. Outing, W. Siegmann, M. Collins, and E. Westwood, "Generalization of the rotated parabolic equation to variable slopes," J. Acoust. Soc. Am. **6**, 3534–3538 (2006).

[9]M. Porter, *The BELLHOP Manual and User's Guide* (HLS Research, La Jolla, CA, 2011).

[10]O. Rodríguez, "General description of the BELLHOP ray tracing model," technical report, 2008 (Last viewed 06/15/2012).

[11]E. Westwood and P. Vidmar, "Eigenray finding and time series simulation in a layered-bottom ocean," J. Acoust. Soc. Am. **81**, 912–924 (1987).

[12]H. Weinberg and R. Keenan, "Gaussian ray bundles for modeling high-frequency propagation loss under shallow-water conditions," J. Acoust. Soc. Am. **100**, 1421–1431 (1996).

[13]http://www.siplab.fct.ualg.pt (Last viewed 06/15/2012).

[14]P. Santos, P. Felisberto, O. Rodríguez, and S. Jesus, "Geoacoustic matched-field inversion using a vector sensor array," in *Conference on Underwater Acoustic Measurements*, Nafplion, Greece (2009), pp. 29–34.

[15]P. Santos, O. Rodríguez, P. Felisberto, and S. Jesus, "Seabed geoacoustic characterization with a vector sensor array," J. Acoust. Soc. Am. **128**, 2652–2663 (2010).

[16]P. Santos, P. Felisberto, O. Rodríguez, J. Joao, and S. Jesus, "Geometric and Seabed parameter estimation using a Vector Sensor Array—Experimental results from Makai experiment 2005," in *OCEANS2011*, Santander, Spain (2011), pp. 1–10.

[17]M. Popov and I. Peník, "Computation of ray amplitudes in homogeneous media with curved interfaces," Studia Geoph. Geod. **22**, 248–258 (1978).

[18]F. Jensen, W. Kuperman, M. Porter, and H. Schmidt, *Computational Ocean Acoustics, AIP Series in Modern Acoustics and Signal Processing* (AIP, New York, 1994), Chap. 3.

[19]M. Popov, "Ray theory and Gaussian beam method for geophysicists," technical report EDUFBA, Universidade Federal da Bahia (2002).

[20]M. Porter and H. Bucker, "Gaussian beam tracing for computing ocean acoustic fields," J. Acoust. Soc. Am. **82**, 1349–1359 (1987).

[21]P. Papadakis, M. Taroudakis, and J. Papadakis, "Recovery of the properties of an elastic bottom using reflection coefficient measurements," in *Proceedings of the 2nd European Conference on Underwater Acoustics*, Copenhagen, Denmark (1994), Vol. II, pp. 943–948.

[22]M. Porter and Y.-C. Liu, "Finite-Element Ray Tracing," in *Theoretical and Computational Acoustics* (World Scientific Publishing, Singapore, 1994), Vol. 2, pp. 947–956.

[23]H. Schmidt, "SAFARI, seismo-acoustic fast field algorithm for range-independent enviroments. User's guide," technical report, Saclant Undersea Research Centre (SM-113), La Spezia, Italy (1987).

[24]F. Press and M. Ewing, "Propagation of explosive sound in a liquid layer overlying a semi-infinite elastic solid," Geophysics **15**, 426–446 (1950).

[25]P. Felisberto, O. Rodríguez, P. Santos, and S. Jesus, "On the usage of the particle velocity field for bottom characterization," in *International Conference on Underwater Acoustic Measurements*, Kos, Greece (2011), pp. 19–24.

[26]M. Collins, *User's Guide for RAM Versions 1.0 and 1.0p* (Naval Research Laboratory, Washington, DC, 2006).

C.2 July 2012

Proceedings of Meetings on Acoustics

Volume 17, 2012 http://acousticalsociety.org/

ECUA 2012 11th European Conference on Underwater Acoustics
Edinburgh, Scotland
2 - 6 July 2012
Session UW: Underwater Acoustics

UW170. Vector sensor geoacoustic estimation with standard arrays

Orlando Rodríguez*, Paulo Felisberto, Emanuel Ey, Joseph Schneiderwind and S. M. Jesus

***Corresponding author's address: Laboratório de Robótica e Sistemas em Engenharia e Ciência (LARSyS), Campus de Gambelas, Universidade do Algarve, Faro, 8005-139, N/A, Portugal, orodrig@ualg.pt**

Vector Sensor Arrays (hereafter VSAs) are progressively becoming more and more attractive among the underwater acoustics community due to the advantages of VSAs over standard arrays of acoustic hydrophones. While the later record only acoustic pressure, VSAs record also particle velocity; such technical feature increases by a factor of four the amount of information that can be used for the processing ofacoustic data, leading to a substantial increase in performance. Since VSA sensor technology is relatively recent, and thus not yet fully available, one can consider the usage of closely located pairs of standard hydrophones, which can be used to estimate the vertical component of particle velocity as a difference of acoustic pressure, measured at each pair of hydrophones. The present discussion introduces a theoretical review of particle velocity calculations using different acoustic models, and tests the performance of estimators for geoacoustic inversion using acoustic pressure, particle velocity components and direct and approximated values of the vertical component only.

Published by the Acoustical Society of America through the American Institute of Physics

 [DOI: 10.1121/1.4792228]
Received 3 Sep 2012; published 31 Jan 2013

1 INTRODUCTION

Vector Sensor Arrays (hereafter VSAs) are progressively becoming more and more attractive among the underwater acoustics community, mainly due to the advantages of VSAs over arrays of standard hydrophones. While the later measure only acoustic pressure, VSAs measure particle velocity as well; such feature increases by a factor of four the amount of information that can be used for the processing of acoustic data, leading to a substantial improvement of array performance. Since VSA sensor technology is relatively recent, and thus not yet fully available, one can consider using closely located pairs of standard hydrophones to estimate the vertical component of particle velocity; the estimation uses the difference of acoustic pressure, measured at each pair. The present discussion introduces a review of particle velocity calculations based on the theoretical background provided by different acoustic models; additionally, the discussion tests the performance of geoacoustic estimation using acoustic pressure, particle velocity components, and direct and approximated values of the vertical component only. The conclusions are presented at the end of the discussion.

2 PARTICLE VELOCITY ESTIMATION

Particle velocity $\mathbf{v}$ is related to acoustic pressure P in the frequency domain through the relationship [1]

$$\mathbf{v} = -\frac{i}{\omega\rho}\nabla P \,, \tag{1}$$

where ρ represents the watercolumn density, and ω stands for the frequency of the propagating wave. The factors ρ and ω only affect the amplitude of particle velocity, while the imaginary unit implies a phase shift of $\pi/2$ radians. Without such factors particle velocity can be viewed as the gradient of acoustic pressure; therefore, the gradient of acoustic pressure will be identified as the particle velocity in the discussion that follows.

2.1 NORMAL MODE APPROACH

The typical normal mode approach is based on the solution of the wave equation for a waveguide with cylindrical symmetry, and further representation of acoustic pressure as the product of two functions; the first function is range dependent, while the second function is depth dependent. The differential equation for the depth dependent function provides a set of orthogonal solutions, called normal modes. The analytical expression for the normal mode expansion can then be written as [2]

$$P(r,z) = S(\omega)\frac{e^{i\pi/4}}{\rho(z_s)\sqrt{8\pi}}\sum_{m=1}^{M} u_m(z_s)\frac{e^{ik_m r}}{\sqrt{k_m r}}\; u_m(z) \tag{2}$$

where $S(\omega)$ stands for the source spectrum, z_s is the source depth and ρ stands for water density. It follows from Eq.(2) that the horizontal derivative of acoustic pressure corresponds, approximately, to

$$u \approx iS(\omega)\frac{e^{i\pi/4}}{\rho(z_s)\sqrt{8\pi}}\sum_{m=1}^{M} u_m(z_s)k_m\frac{e^{ik_m r}}{\sqrt{k_m r}}\ u_m(z) \tag{3}$$

while the vertical derivative corresponds (exactly) to:

$$w = S(\omega)\frac{e^{i\pi/4}}{\rho(z_s)\sqrt{8\pi}}\sum_{m=1}^{M} u_m(z_s)\frac{e^{ik_m r}}{\sqrt{k_m r}}\ \frac{du_m}{dz}\ . \tag{4}$$

Following [3] one can notice that both p and u are expanded on the basis of modes $u_m(z)$, while w is expanded on the basis of du_m/dz. In a general sense one can write that

$$\frac{du_m}{dz} \approx i\gamma_m u_m(z) \quad \text{where} \quad \gamma_m = \sqrt{\omega^2/c^2 - k_m^2} \quad \text{stands for the vertical wavenumber.} \tag{5}$$

Additionally, since γ_m increases with increasing mode number the net effect of the derivative is to enhance the contribution of higher modes in the field of w, relative to the stronger weighting of low order modes in the fields of p and u. Thus, Eq.(2) provides a clear framework to distinguish the different enhancement of modes in the fields of u and w. Additional approaches for the calculation of particle velocity from acoustic pressure will follow in the next sections. To this end and without loss of generality it will be considered that $S(\omega)$ = 1 and $\rho(z)$ = constant.

2.2 GAUSSIAN BEAM APPROACH

Within the Gaussian beam approach the influence of a given ray to the field of acoustic pressure can be written as [4]

$$P(s,n) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{q_\perp(s)q(s)}} \times \exp\left[-i\omega\left(\tau(s) + \frac{1}{2}\frac{p(s)}{q(s)}n^2\right)\right], \tag{6}$$

where s and n stand for the ray arclength and the ray normal, respectively, $c(s)$ stands for sound speed along the ray trajectory, $\theta(0)$ is the launching angle, and p and q are beam parameters, obtained from the solution of dynamic equations. The acoustic pressure at a given point is obtained summing the influences of different rays passing near the hydrophone. The pressure gradient can be written in terms of ray arclength and ray normal as

$$\nabla P = \frac{\partial P}{\partial n}\mathbf{e}_n + \frac{\partial P}{\partial s}\mathbf{e}_s\ , \tag{7}$$

where $\mathbf{e}_s$ and $\mathbf{e}_n$ are unitary vectors, oriented along s and n, respectively. The two vectors are related to the ray elevation θ through the relationships

$$\mathbf{e}_s = [\cos\theta, \sin\theta] \quad \text{and} \quad \mathbf{e}_n = [-\sin\theta, \cos\theta] \tag{8}$$

(see Fig.2.2). In order to obtain analytical expressions for the derivatives along n and s let us rewrite Eq.(6) as

$$P(s,n) = P_0(s)\exp\left[-i\omega\left(\frac{s}{c(s)} + \frac{1}{2}\gamma(s)n^2\right)\right]\ , \tag{9}$$

where $s/c(s) = \tau(s)$ and $\gamma(s) = p(s)/q(s)$. Therefore, the first derivative corresponds exactly to

$$\frac{\partial P}{\partial n} = -i\omega\gamma(s)nP\ . \tag{10}$$

The exact derivative along s produces a cumbersome expression due to the dependence of the different factors on s. Since the exponential factor has the largest impact on the derivative one can write approximately that

$$\frac{\partial P}{\partial s} = -i\frac{\omega}{c(s)}P\ . \tag{11}$$

Further, the velocity components (u, w) can be calculated projecting the gradient components $\partial P/\partial n$ and $\partial P/\partial s$ onto the (r, z) axes as

$$u = (\nabla P \cdot \mathbf{e}_r) = -\frac{\partial P}{\partial n}\sin\theta + \frac{\partial P}{\partial s}\cos\theta \tag{12}$$

and

$$w = (\nabla P \cdot \mathbf{e}_z) = \frac{\partial P}{\partial n}\cos\theta + \frac{\partial P}{\partial s}\sin\theta \tag{13}$$

where $\mathbf{e}_r$ and $\mathbf{e}_z$ stand for the unitary vectors along r and z, respectively.

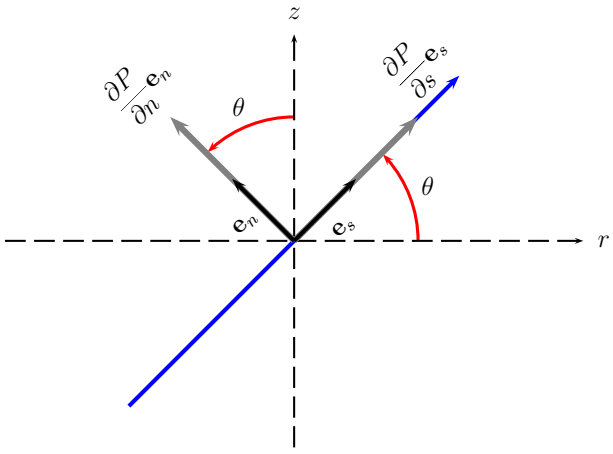


Figure 1: Ray polarization vectors $\mathbf{e}_s$ and $\mathbf{e}_n$ and pressure gradient components along the n and s directions; θ stands for the ray elevation (i.e. the ray angle relative to the r axis).

2.3 PARABOLIC EQUATION APPROACH

The parabolic approach is based on the substitution of the wave equation with a parabolic equation (hereafter PE), which contains a single derivative along range; following [5, 6] one can write that

$$P(r, z) = P_0 \frac{1}{\sqrt{r}} \Psi(r, z) \exp(ik_0 r) \tag{14}$$

where

$$\frac{\partial \Psi}{\partial r} = -ik_0 \hat{H} \Psi \ , \tag{15}$$

$k_0 = \omega/c_0$, c_0 is a reference sound speed and $\hat{H}$ represents a total energy-like differential operator. Given $\Psi(0, z)$ a marching solution can be obtained by integrating Eq.(15) progressively along range through a split-step technique. PE models can be divided into two categories depending on how $\hat{H}$ is approximated, namely Padé approximations and spectral techniques. Padé approximations replace $\hat{H}$ with rational polynomials, which reduce Eq.(15) to a set of tridiagonal linear equations, which are solved at each range step. Spectral techniques rely on the properties of Fourier transforms to solve Eq.(15) in the wavenumber domain, providing the following marching solution [5]

$$\Psi(r + \Delta r, z) = \exp\left(-ik_0 \Delta r \hat{U}\right) \times \mathsf{FFT}\left[\exp\left(-ik_0 \Delta r \hat{T}\right) \times \hat{\Psi}(r, k)\right] \tag{16}$$

where $\hat{\Psi}(r, k)$ is the inverse Fourier transform of $\Psi(r, z)$ and $\hat{T} + \hat{U} = \hat{H}$. The spectral technique allows to obtain closed-form analytical expressions for particle velocity; in particular (and without loss of generality) the operators $\hat{T}$ and $\hat{U}$ for the standard parabolic equation correspond to

$$\hat{T} = -\frac{1}{2k_0^2}\frac{\partial^2}{\partial z^2} \quad \text{and} \quad \hat{U} = -\frac{1}{2}\left(n^2 - 1\right) \ ,$$

where $n = c/c_0$. The corresponding expressions for u and w become

$$u = u_0 \exp(ik_0 r) \times (u_1 + u_2 - u_3) \ , \tag{17}$$

$$w = w_0 \exp(ik_0 r) \times \mathsf{FFT}\left[ik\hat{\Psi}(r,k)\right] \ , \tag{18}$$

where

$$u_1 = i\frac{\Psi(r,z)}{2k_0 r} \quad , \quad u_2 = \frac{1}{2}(n^2+1)\Psi(r,z)$$

and

$$u_3 = \mathsf{FFT}\left[\frac{1}{2}\left(\frac{k}{k_0}\right)^2 \hat{\Psi}(r,k)\right] \ .$$

2.4 INTERPOLATION (MODEL-INDEPENDENT) APPROACH

The previous approaches are all model-based in the sense that the calculation of u and w is based on derivatives of an analytical expression for acoustic pressure; such expressions are specific to the model considered. The approach used in the acoustic model TRACEO relies on a barycentric parabolic interpolator, which is used to estimate both horizontal and vertical derivatives. The interpolator is constructed using a star-like stencil of five points; the center of the star coincides with the position of the hydrophone, while the horizontal and vertical points are located at the left and right sides of the hydrophone, and also above and below it. Acoustic pressure is calculated at the five points of the stencil; the points aligned along the horizontal are used to calculate the horizontal derivative at the star's center; the vertical derivative at the same position is calculated using the points aligned along the vertical. The spacing between an outer point and the star's center corresponds to $\lambda/10$; such choice is expected to avoid frequency aliasing. The interpolator itself can be described as follows: let us consider a set of three points x_1, x_2 and x_3, and the corresponding function values $f(x_1)$, $f(x_2)$ and $f(x_3)$.

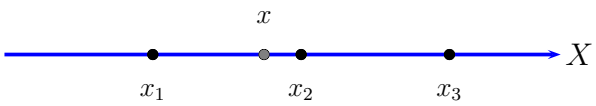


Figure 2: Barycentric parabolic interpolation.

At a given point x (see Fig.2) the interpolant can be written as

$$f(x) = f(x_1) + a_2 (x - x_1)(x - x_3) + a_3 (x - x_1)(x - x_2) \ . \tag{19}$$

It follows from this expression that

$$a_2 = \frac{f(x_2) - f(x_1)}{(x_2 - x_1)(x_2 - x_3)} \quad \text{and} \quad a_3 = \frac{f(x_3) - f(x_1)}{(x_3 - x_1)(x_3 - x_2)} \ .$$

The approximations to the derivatives become

$$\frac{df}{dx} = a_2 (2x - x_1 - x_3) + a_3 (2x - x_1 - x_2)$$

and

$$\frac{d^2 f}{dx^2} = 2(a_2 + a_3) \ .$$

The validity of the interpolation approach will be discussed in the following section.

3 TRACEO'S DESCRIPTION AND PRELIMINARY VALIDATION

TRACEO is a Gaussian beam model, which is under current development at the SiPLAB of the University of Algarve.[1] The model was developed in order to:

- Predict acoustic pressure and particle velocity in environments with elaborate upper and lower boundaries, which can be characterized by range-dependent compressional and shear properties. Modeling particle velocity is important for vector sensor applications and can be used, in particular, for geoacoustic inversion with high frequency data [7–9].
- Include one or more targets in the waveguide.
- Produce ray, eigenray, amplitude and travel time information. In particular, eigenrays are to be calculated even if rays are reflected backwards on targets located beyond the current position of the hydrophone.

A preliminary validation of TRACEO's accuracy for particle velocity calculations is presented here through the comparison between TRACEO's predictions and the exact results of particle velocity calculations for a Pekeris waveguide (shear included); the compressional and shear potentials of such a waveguide are well know in the literature, [10] and the particle velocity components are easily calculated from the analytic expressions for both potentials. The comparison is shown in Fig. 3 for the set of parameters f = 200 Hz, z_s = 69.1 m, z_r = 137.1 m, D = 145 m, c_p = 2290 m/s, c_s = 1050 m/s, ρ_b = 1.378 g/cm^3, α_p = 0.76 dB/λ and α_s = 1.05 dB/λ; in both cases the analytic solution is indicated by a solid curve, while TRACEO's prediction is indicated by a dashed curve. The interpolation approach is so accurate in the case of u that the two curves are difficult to distinguish (Fig. 3(a)). The approach is less accurate for w, with TRACEO exhibiting some overshooting of the analytic solution (Fig. 3(b)), but still reproducing the interference pattern in both phase and amplitude along range.

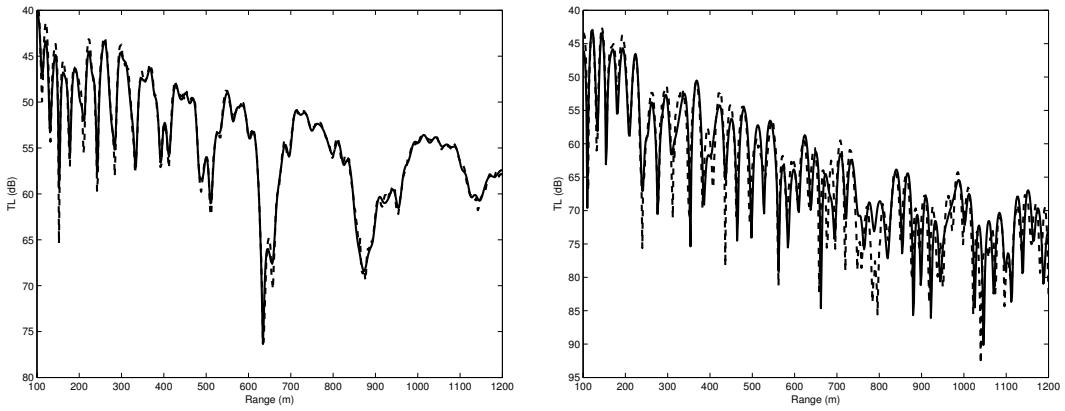


Figure 3: Particle velocity calculations for the Pekeris waveguide: u (left); w (right). In both cases the solid curve represents the analytic solution, while the dashed curve represents TRACEO's prediction.

4 SIMULATIONS

The performance of geoacoustic estimation using acoustic pressure or particle velocity components is discussed in this section through simulations of the Makai waveguide [8]. The discussion is divided in three parts, namely:

- TRACEO predictions will adress the performance of geoacoustic estimation when the Bartlett estimator is calculated using either p, u or w; the similarities and differences between the estimators will be outlined.

[1] http://www.siplab.fct.ualg.pt.

- TRACEO predictions for the best estimator will be validated through comparisons with the predictions provided by other models.
- TRACEO geoacoustic estimation using Δp will be compared with the estimation obtained with model predictions of w.

4.1 ESTIMATION USING P, V AND W

The geoacoustic estimation performed with TRACEO considers an idealized Makai waveguide, in which the true value of sediment compressional speed corresponds to 1570 m/s; during the experiment it was used a VSA with 4 hydrophones between depths 79.6 to 79.9 m, and signals were transmitted to a range of 1830 m. The Bartlett estimator was calculated at 13078 Hz in the interval from 1500 m/s to 1800 m/s using either p, u or w; the three estimators are shown in Fig.4. The figure

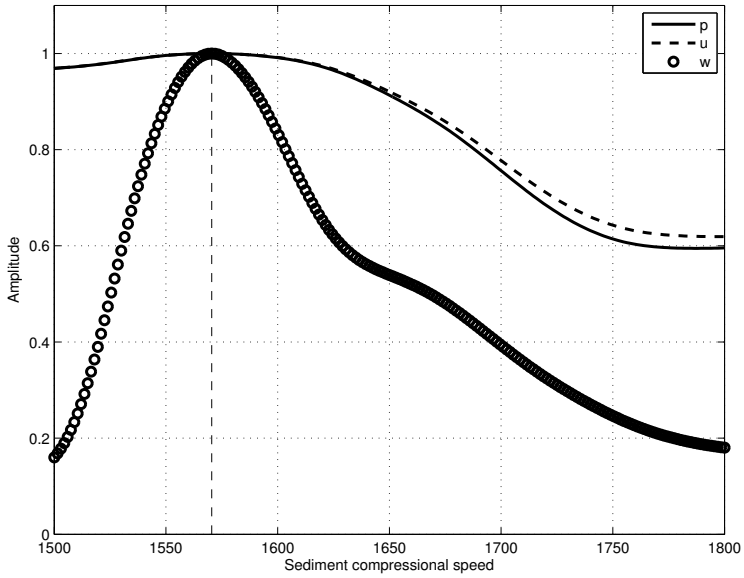


Figure 4: Bartlett estimators of compressional sound speed using acoustic pressure p, horizontal component of particle velocity u and vertical component of particle velocity w, calculated with TRACEO. The true value of compressional speed is indicated by the vertical dashed line.

reveals significant differences between the estimators using p or u, and the estimator using w; in fact, the first two estimators practically coincide, and exhibit a weak variation over the interval of considered compressional speeds; on the other side the estimator using w is significantly narrower over the same interval. Therefore, the results clearly indicate a clear advantage for geoacoustic estimation of using w over the use of either p or u.

4.2 ESTIMATION USING W WITH DIFFERENT MODELS

To further validate the geoacoustic estimation based on w from the previous section a new set of calculations was considered using three more acoustic models, namely KRAKEN (a normal mode model [2]), Bellhop (a Gaussian beam ray tracing model [4]) and MMPE (a parabolic equation model [6]). However, running KRAKEN and MMPE at 13078 Hz required such a fine discretization in depth and range which implied extremely long calculations, besides occupying significant disk space. To deal with both issues (and since the main purpose of the simulation is to validate geoacoustic estimation using w) it was decided to calculate the Bartlett estimator at 500 Hz for an idealized array with 6 hydrophones, between depths 75 to 80 m. The corresponding curves are shown in Fig.5; although the curves do not exhibit the same behaviour the general trend is very similar, with a narrow peak

around the true value of compressional sound speed. Remarkably, there is an excellent agreement between TRACEO and KRAKEN predictions. Such results confirm not only the advantage of relying on w for geoacoustic estimation, but also the validity of the interpolation approach for the calculation of particle velocity components.

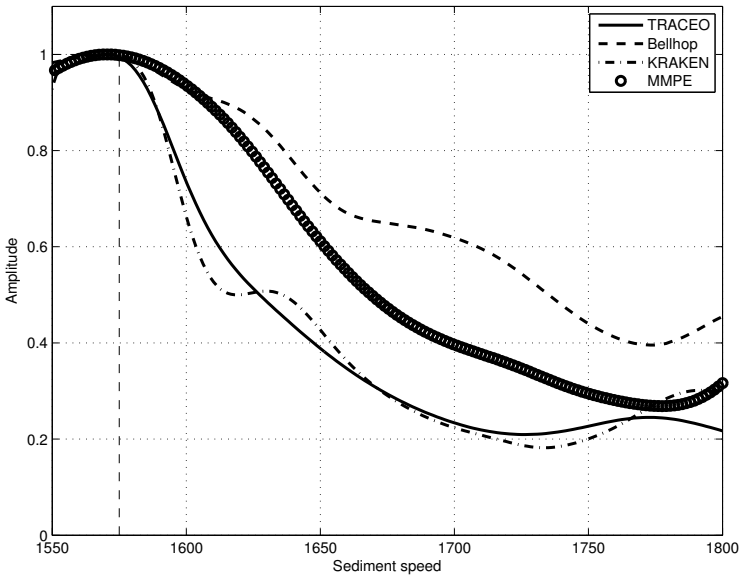


Figure 5: Bartlett estimators of compressional sound speed using w, calculated with TRACEO, Bellhop, KRAKEN and MMPE. The true value of compressional speed is indicated by the vertical dashed line.

4.3 ESTIMATION USING ΔP INSTEAD OF W

Following the results from the previous section the Makai waveguide was again considered as described in section 4.1, but with an hypothetical array of five standard hydrophones, located between depths 79.55 to 79.95 m. Acoustic pressure for such array was calculated again with TRACEO, and the "true" data was calculated as consecutive differences of acoustic pressure (2nd hyd minus 1st hyd, 3rd hyd minus 2nd. hyd, and so on); thus, the covariance matrix was calculated using such true data. Further, when calculating the Bartlett estimator, TRACEO predicted the values of w for an array of four hydrophones between depths 79.6 to 79.9 m. The results of geoacoustic estimation at 13078 Hz using both w and Δp are shown in Fig.6, which shows that the two curves coincide almost perfectly. However, data for the second curve would require an array of standard hydrophones, while the data for the first curve would require a VSA.

5 CONCLUSIONS AND FUTURE WORK

The discussion presented here is extremely encouraging for the development of VSAs with pairs of standard hydrophones, which would measure not only acoustic pressure but also the vertical component of particle velocity. The applications for geoacoustic estimation at high frequencies were positively outlined, and ray tracing models (despite apparent shortcomings) were shown to perform quite efficiently and accurately for estimation. There are, however, important issues that remain to be considered such as, for instance, if the accuracy of estimation is going to hold over arbitrary distances. Additionally, the sensitivity of estimation to noise corruption and to the type of ocean bottom (which implies significant differences in the values of compressional sound speed) will require detailed discussion in the future.

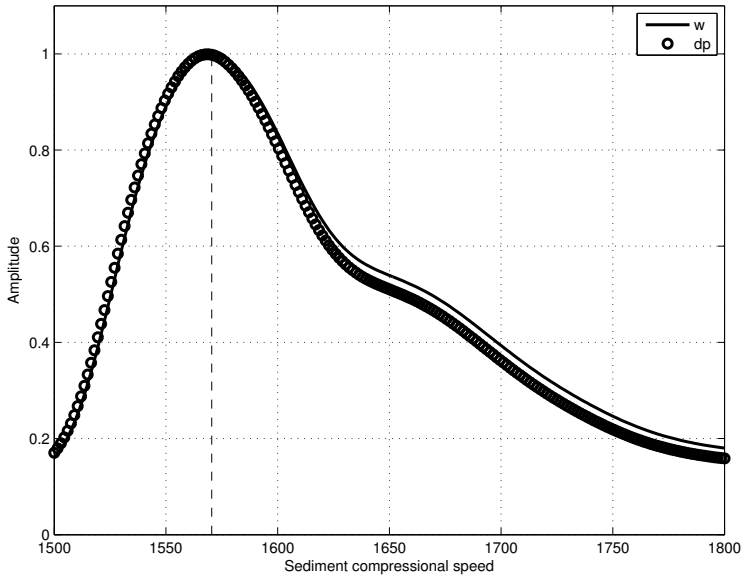


Figure 6: Bartlett estimators of compressional sound speed using w and Δp, calculated with TRACEO. The true value of compressional speed is indicated by the vertical dashed line.

ACKNOWLEDGEMENTS

This work was funded by National Funds through FCT - Foundation for Science and Technology under project SENSOCEAN (PTDC/EEA-ELC/104561/2008).

REFERENCES

1. H. Schmidt. *SAFARI, Seismo–Acoustic Fast field Algorithm for Range – Independent enviroments. User's Guide*. Tech. report, SACLANT UNDERSEA RESEARCH CENTRE (SM-113), La Spezia, Italy, pp. 9-17, 1987.

2. M. Porter. *The KRAKEN normal mode program*. Tech. report, SACLANT UNDERSEA RESEARCH (memorandum), San Bartolomeo, Italy, page 8, 1991.

3. P. Felisberto, O. C. Rodríguez, P. Santos, and S. M. Jesus. On the usage of the particle velocity field for bottom characterization. In *Int. Conf. on Underwater Acoustic Measurements*, pp. 19–24, Kos, Greece, June 2011.

4. M. Porter and H. Bucker. Gaussian beam tracing for computing ocean acoustic fields. *J. Acoust. Soc. Am.*, 82(4): pp. 1349–1359, 1987.

5. K. Smith and F. Tappert. *UMPE: The University of Miami Parabolic Equation Model*. Tech. report, University of Miami (MPL Technical Memorandum 432), Miami, USA, pp. 45-48, 1994.

6. K. Smith. Convergence, stability, and variability of shallow water acoustic predictions using a split-step Fourier parabolic equation model. *J. Acoust. Soc. Am.*, 9(1): pp. 243–285, 2001.

7. P. Santos, P. Felisberto, O. C. Rodríguez, and S. M. Jesus. Geoacoustic Matched-field Inversion using a Vector Sensor Array. In *Conf. on Underwater Acoustic Measurements*, pp. 29–34, Nafplion, Greece, 21-26 June 2009.

8. P. Santos, O. C. Rodríguez, P. Felisberto, and S. M. Jesus. Seabed geoacoustic characterization with a vector sensor array. *J. Acoust. Soc. Am*, 5(128): pp. 2652–2663, November 2010.

9. P. Santos, P. Felisberto, O. C. Rodríguez, J. J. ao, and S. M. Jesus. Geometric and Seabed parameter estimation using a Vector Sensor Array - Experimental results from Makai experiment 2005. In *OCEANS2011*, pp. 1–10, Santander, Spain, June 2011.

10. F. Press and M. Ewing. Propagation of explosive sound in a liquid layer overlying a semi-infinite elastic solid. *Geophysics*, 15(3): pp. 426–446, 1950.

C.3 July 2013

Sensors **2013**, *13*, 8856-8878; doi:10.3390/s130708856

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Experimental Results of Underwater Cooperative Source Localization Using a Single Acoustic Vector Sensor

Paulo Felisberto *, Orlando Rodriguez, Paulo Santos, Emanuel Ey and Sérgio M. Jesus

Laboratory of Robotics and Systems in Engineering and Science (LARSyS), University of Algarve, Campus de Gambelas, 8005-139 Faro, Portugal; E-Mails: orodrig@ualg.pt (O.R.); pjsantos@ualg.pt (P.S.); emanuel.ey@gmail.com (E.E.); sjesus@ualg.pt (S.M.J.)

* Author to whom correspondence should be addressed; E-Mail: pfelis@ualg.pt; Tel.: +35-128-980-0100 (ext. 6575); Fax: +35-128-986-4258.

Received: 7 May 2013; in revised form: 29 June 2013 / Accepted: 1 July 2013 / Published: 12 July 2013

Abstract: This paper aims at estimating the azimuth, range and depth of a cooperative broadband acoustic source with a single vector sensor in a multipath underwater environment, where the received signal is assumed to be a linear combination of echoes of the source emitted waveform. A vector sensor is a device that measures the scalar acoustic pressure field and the vectorial acoustic particle velocity field at a single location in space. The amplitudes of the echoes in the vector sensor components allow one to determine their azimuth and elevation. Assuming that the environmental conditions of the channel are known, source range and depth are obtained from the estimates of elevation and relative time delays of the different echoes using a ray-based backpropagation algorithm. The proposed method is tested using simulated data and is further applied to experimental data from the Makai'05 experiment, where 8–14 kHz chirp signals were acquired by a vector sensor array. It is shown that for short ranges, the position of the source is estimated in agreement with the geometry of the experiment. The method is low computational demanding, thus well-suited to be used in mobile and light platforms, where space and power requirements are limited.

Keywords: vector sensors; source localization; ray backpropagation

---

1. Introduction

This paper proposes a single sensor based three-dimensional localization method that, by taking advantage of the spatial filtering capabilities of a vector sensor, allows for a low computational demanding implementation, suitable for light real-time systems. An acoustic vector sensor (VS) is a device that measures the three orthogonal components of the particle velocity simultaneously with the pressure field at a single position in space. Vector sensors have been long used for target localization by the US Navy, due to their inherent spatial filtering capabilities [1]. In the early 1990s, in a paper that received considerable attention, D'Spain *et al.* [2] presented results for single-element and full array beamformed data acquired by an array of 16 vector sensors, the directional frequency analysis and recording (DIFAR) array. During the last two decades, several authors have conducted research on the signal processing theory of vector sensors (see, for instance, [3–5] and the references therein). Although the majority of that work is related to direction of arrival estimation, in the last decade, vector sensors have been proposed in other fields, like port and waterway security [6], underwater communications [7], geoacoustic inversion [8–10] and geophysics [11].

Taking advantage of the intrinsic spatial filtering capability of a vector sensor (a typical VS presents a figure of height directivity pattern and a directivity index of 4.8 dB [12]), the usage of a single vector sensor for the direction of arrival estimation (azimuth and elevation) was considered by several theoretical and simulation studies. Due to the collocation of the pressure and the orthogonal particle velocity sensing elements in a single vector sensor device, the direction of arrival algorithms can be frequency invariant, thus computationally simple direction of arrival (DOA) algorithms can be used for *a priori* unknown and time-varying broadband signals in the presence of spatially distributed broadband interferences [13]. Azimuth and elevation algorithms for tracking of a passive source using a single vector sensor were proposed by Liu *et al.* [14] based in Kalman filters and Awad and Wong [15] based in a recursive least-squares. The performance of both methods were compared in [15] considering a simulation scenario.

Due to multipath, in shallow water environments, the waveform impinging on a receiver is a sum of different echoes. Rahamim *et al.* [16] proposed various vector sensor array (VSA)-based direction of arrival estimators for multipath environments and evaluated their performance using simulations. Arunkumar and Anand [17] proposed a method for three-dimensional (3D) source localization of a narrowband source using a vector sensor array. Their method is based in a normal mode representation of a range-independent shallow ocean. It is shown that the azimuth of the source can be estimated directly from the horizontal components measured at a vector sensor array, the range is obtained by closed form, and the depth is estimated by a matched-field approach. Hurtado and Nehorai [18] analyzed the performance of a passive direction of arrival and a range estimation method of a source in the air above the ocean based on the interference between the direct and sea-surface reflected field impinging on polarization-sensitive (electromagnetic) sensors.

Thanks to technological advances and small size, low noise underwater acoustic vector sensors with improved dynamic range and bandwidth are becoming available [19]. Those compact sensors are well-suited to be used in light systems, where space and computational resources are limited and energy consumption is of concern, as, for example, in autonomous underwater vehicles (AUV) and similar

mobile platforms. Hawkes and Nehorai [20] proposed a fast broadband intensity-based algorithm for determining three-dimensional localization of a source using distributed vector sensors situated on a reflecting boundary. The method considers that each vector sensor is impinged on by a direct echo, which determines the elevation of the source, and by a reflected echo. The reflected echo does not affect the azimuth estimate, as long as the environment is considered homogeneous, but it introduces errors in elevation estimation. The authors proposed a method that filters out the reflected echo, thus achieving a more accurate elevation estimate of the source. The performance of the method was shown for a simulated environment, where the three-dimensional localization of the source was obtained from a set of azimuth and elevation estimates obtained from distributed vector sensors.

The present paper shows that the azimuth, range and depth of a high frequency broadband cooperative source, slowly moving ($<$0.3 m/s) in a shallow water environment, can be tracked in the presence of multipath using a single vector sensor. The azimuth and elevation of the echoes impinging on the vector sensor are estimated from the amplitude of the particle velocity components using a least squares-based algorithm. Then, a ray backpropagation method [21] is applied to estimate source range and depth, where ray trajectories are launched from the receiver at the elevation angles estimated from the various echoes. Afterwards, the range and depth estimates are obtained by least squares minimization of an objective function that combines the ray trajectories and the relative travel times estimated in the previous stage. The range and depth estimation method can be implemented with a single forward ray tracing model run. Additionally, when only the direct and the surface-reflected echoes are considered, source range and depth can be estimated using the source image method. Although the method requires *a priori* a complete record of the source signal, it is very simple to implement even in light platforms, thus suitable for real-time localization and tracking of cooperative sources. The proposed method is tested with simulated data and applied to a data set acquired during the Makai Experiment (Makai'05) held in September 2005, off the coast of Kauai Island (Hawaii, HI, USA) using a Wilcoxon TV-001 vector sensor device [22]. The orientation of the x- and y-axes of the vector sensor, initially unknown, is determined from the ship self noise using an intensity method [23], based on the inner product between the sample pressure and the various particle velocity components. The results obtained from a 8–14 kHz chirp signal transmitted from a cooperative source are in agreement with the known geometry of the experiment, showing that the 3D localization of the source is achieved for ranges until 500 m (the azimuth alone was tracked along a 2 km transect [24]). The method presented herein is very fast when compared with single hydrophone methods [25–28], which require a large number of time-consuming forward model runs associated with complex optimization procedures.

This paper is organized as follows: Section 2 presents the theoretical framework considered in the data processing and analysis. In Section 3, the proposed method is tested in a simulated scenario. Section 4 shows and discusses the results obtained on a real data set, and Section 5 summarizes the paper. Preliminary results of this work were presented in [29].

2. Theoretical Framework

2.1. The Measurement Model

In the following, a vector sensor is considered that measures the pressure, $p(t)$, and the three orthogonal components of the particle velocity, $v_x(t)$, $v_y(t)$ and $v_z(t)$, along the x-, y- and z-axes, respectively. The vector sensor is located at the origin of the Cartesian coordinate system, x–y being the horizontal plane and x–z the vertical plane. The azimuth, $\Theta\,(-180^\circ \leq \Theta \leq 180^\circ)$, and elevation, $\Phi\,(-90^\circ \leq \Phi \leq 90^\circ)$, are defined in a conventional manner.

Without loss of generality, it is assumed that the signal impinging on the vector sensor is in the far-field and band-limited; thus, pressure, $p(t)$, is related to particle velocity, $\mathbf{v}$, by:

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho_0}\nabla p \tag{1}$$

where ρ_0 is the medium density and ∇ is the gradient operator.

Assuming a monochromatic signal of frequency, ω, one can write that:

$$\mathbf{v} = -\frac{1}{j\omega\rho_0}\nabla p \tag{2}$$

where j is the imaginary unit. In the far-field of a free-space environment with sound speed, c_0, the wavefront is planar; thus:

$$\mathbf{v} = \frac{1}{\rho_0 c_0} p\mathbf{u} \tag{3}$$

where $\mathbf{u} = [u_x, u_y, u_z]$ is the unit vector pointing to the source (thus, in the opposite direction of the wavefront propagation).

When a field due to a point source in the far-field is sampled by a vector sensor with small dimension compared to the signal wavelength, then the wavefront can by considered planar. Thus, from Equation (3), the measured pressure and particle velocity components are related by:

$$\begin{bmatrix} p(t) \\ v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} = \begin{bmatrix} s(t) \\ \alpha s(t) u_x \\ \alpha s(t) u_y \\ \alpha s(t) u_z \end{bmatrix} + \begin{bmatrix} n(t) \\ n_x(t) \\ n_y(t) \\ n_z(t) \end{bmatrix} \tag{4}$$

where $s(t)$ is the source waveform, $u_x = \cos(\Phi_s)\cos(\Theta_s), u_y = \cos(\Phi_s)\sin(\Theta_s), u_z = \sin(\Phi_s)$, Φ_s is the source elevation and Θ_s is the source azimuth. The proportionality factor, α, arises directly from Equation (3), but in a more general setting, it can also account for any existing proportionality in the output streams of a vector sensor device, due to the various electro-mechanical principles used to measure pressure and particle velocity. In Equation (4), $n(t)$ represents additive pressure noise, and $n_x(t)$, $n_y(t)$, $n_z(t)$ its particle velocity counterparts. A common assumption is that signal and noise are uncorrelated both in time and space. The cross-correlation between the four vector sensor components have been studied by several authors [30,31]. It was demonstrated that in the presence of azimuthally isotropic noise, the horizontal particle velocity and the pressure components are mutually uncorrelated. Moreover, if the noise is spherically symmetric, the vertical particle velocity term is also uncorrelated with the other

noise terms. Furthermore, the noise power at the pressure channel is equal to the sum of noise power at the so-called pressure equivalent of particle velocity [31], which is obtained by the product of the particle velocity by $-\rho_0 c$ [23].

2.2. Intensity-Based Azimuth Estimation

Intensity-based source direction estimation was considered in the pioneering work of D'Spain *et al*. [2]. Later on, Nehorai and Paldi [23] revisited the method and analyzed its statistical performance in terms of the Cramér-Rao bound and mean square angular error. The method is based on the cross-correlation between the pressure measurements and the various components of the particle velocity, which allow one to estimate the factors, αu_x, αu_y and αu_z, and, subsequently, the direction of the impinging wavefront. Taking into account that the signal and the noise are zero mean uncorrelated processes and the pressure and the x component of the particle velocity in Equation (4), one can write the cross-correlation at lag 0 between these two vector sensor components as:

$$E\left[v_x(t)p(t)\right] = \alpha u_x E\left[s^2(t)\right] + E\left[n_x(t)n(t)\right] \tag{5}$$

where $E[\,]$ is the expectation operator and $E\left[s^2(t)\right]$ represents the energy of the signal as seen by the vector sensor. The term, $E\left[n_x(t)n(t)\right]$, represents the cross-correlation (at lag 0) between the pressure and the x component of the particle velocity noise. For a number of cases, in the presence of isotropic noise, this term can be assumed to be zero (see [30,31]) or, in practice, a small fraction of the signal power term (high signal to noise ratios (SNR) situation). Thus, for high SNR, αu_x can be estimated directly from the cross-correlation (at lag 0) between the pressure and the x component of particle velocity. Similar analysis holds for the cross-correlation between the pressure and the y component of the particle velocity.

Assuming that $s(t)$ and the noise components are ergodic processes, a possible estimator for the azimuthal direction of the source signal, Θ_s, is given by:

$$\hat{\Theta}_s = \arctan \frac{\langle v_y(t)p(t)\rangle}{\langle v_x(t)p(t)\rangle} \approx \arctan \frac{u_y}{u_x} \tag{6}$$

where $\langle\,\rangle$ stands for time averaging, and the approximate expression was obtained using Equation (5). The full 360° azimuth is resolved, taking into account the sign of the numerator and denominator of Equation (6).

If the following assumptions hold: (1) the source is in the far field; (2) 3D propagation effects can be neglected; (3) the frequency of the signal is high compared with the cutoff frequency of the acoustic channel—therefore it acts as a waveguide—and (4) the receiver is far from the boundaries—the method above can be used even in a multipath environment [20]. However, due to multipath, a similar approach cannot be, in general, used to estimate elevation, since the energy generated by the source impinges on the vector sensor in multiple arrivals (echoes) from different elevation angles; thus, the elevation seen by the vector sensor is in some sense only a "mean" elevation [20].

2.3. Amplitude-Delay-Angle Estimation in a Multipath Environment

In an underwater environment, it is a common assumption to consider that the multipath structure received on a sensor well away from the channel boundaries is a sum of plane waves. Thus, the ocean acts as a linear system, and neglecting the Doppler, the waveform sampled by the pressure sensor is:

$$p(t) = \sum_{m=1}^{M} a_m s(t - \tau_m) + n(t) \tag{7}$$

where M is the number of echoes, a_m and τ_m are, respectively, the strength (amplitude) and time delay of the m-th echo and $n(t)$ represents the additive noise. In the far-field, the pressure and the particle velocity components are in phase [32]; therefore model Equation (7) can be extended to the particle velocity field by:

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} = \begin{bmatrix} \sum_{m=1}^{M} a_m^x s(t - \tau_m) \\ \sum_{m=1}^{M} a_m^y s(t - \tau_m) \\ \sum_{m=1}^{M} a_m^z s(t - \tau_m) \end{bmatrix} + \begin{bmatrix} n_x(t) \\ n_y(t) \\ n_z(t) \end{bmatrix} \tag{8}$$

where the coefficients, a_m^x, a_m^y, a_m^z, are the attenuation along the m-th path for the three components of the particle velocity. The noise sequences, $n_x(t), n_y(t), n_z(t)$, are additive zero mean, mutually uncorrelated and uncorrelated with the signal, which is a fair assumption when the sensor self-noise is the most relevant noise component. Making the further assumption that the signal, $s(t)$, is known and has a narrow autocorrelation, a least-squares or maximum likelihood approach for time delay and amplitude estimation can be applied [26]. Given the estimates of a_m^x, a_m^y, a_m^z, the elevation (and azimuth) of the different echoes can be obtained by simple relations.

Considering a snapshot of N samples acquired at a sampling interval, Δt, System Equation (8) can be written as:

$$\mathbf{Y} = \mathbf{S}(\tau)\mathbf{A} + \mathbf{N} \tag{9}$$

where $\mathbf{Y}$ is a matrix of dimension, $N \times 3$, whose columns, $\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z$, represent the components of the vector sensor ($\mathbf{Y} = [\mathbf{v}_x|\mathbf{v}_y|\mathbf{v}_z]$), amplitude matrix, $\mathbf{A}$, is of dimension, $M \times 3$, $\mathbf{A} = [\mathbf{a}_x|\mathbf{a}_y|\mathbf{a}_z]$, $\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z$ are the vectors of amplitudes of individual components and matrix $\mathbf{S}(\tau)$ has dimension $N \times M$ ($\tau = [\tau_1, ...\tau_m, ..., \tau_M]$), where the m-th column is given by $\mathbf{s}(\tau_m) = [s(-\tau_m), ..., s((N-1)\Delta t - \tau_m)]^T$. Matrix $\mathbf{N}$ of dimension ($N \times 3$) represents the noise components ($\mathbf{N} = [\mathbf{n}_x|\mathbf{n}_y|\mathbf{n}_z]$).

If the amplitude matrix, $\mathbf{A}$, is deterministic, a least squares approach can be used to estimate the amplitudes and time delays [26]. Assuming that the delays are known, the amplitudes are estimated by minimizing the functional:

$$\hat{\mathbf{A}}(\tau) = \arg\{\min_{\mathbf{A}} \parallel \mathbf{Y} - \mathbf{S}(\tau)\mathbf{A} \parallel^2\} \tag{10}$$

whose solution is given by:

$$\hat{\mathbf{A}}(\tau) = \left[\mathbf{S}^{\mathbf{H}}(\tau)\mathbf{S}(\tau)\right]^{-1} \mathbf{S}^{\mathbf{H}}(\tau)\mathbf{Y} \tag{11}$$

where the superscript $\mathbf{H}$ represents complex conjugate transpose. Since the time delays are generally unknown, the amplitudes are estimated for each plausible time delay, giving rise to a delay-amplitude

curve. When the autocorrelation function of the source signal is sharp and the relative time delays are of smaller order than the observation time ($N\Delta t$), then the envelope of the absolute value of a delay-amplitude curve is known as the arrival pattern. The amplitude-delay estimates of the echoes are given by their M highest peaks (absolute value). When the noise is white and the different echoes suffer uncorrelated perturbations, the amplitude-delay estimation procedure can be equivalently obtained by a matched filter [26]; thus, independently for each vector sensor component. In the case of a stationary environment and when several transmissions are available, the amplitude estimates can be obtained by averaging.

Once the coefficients, $\hat{a}_m^x, \hat{a}_m^y, \hat{a}_m^z$, of the m-th echo are estimated, then the corresponding azimuth, $\hat{\theta}_m$, and elevation, $\hat{\phi}_m$, are given by:

$$\hat{\theta}_m = \arctan \frac{\hat{a}_m^y}{\hat{a}_m^x} \tag{12}$$

$$\hat{\phi}_m = \arctan \frac{\hat{a}_m^z}{\sqrt{(\hat{a}_m^x)^2 + (\hat{a}_m^y)^2}}. \tag{13}$$

The elevation estimates, along with the relative echo arrival times, form the basis for the source range-depth estimation algorithm.

2.4. Range-Depth Estimation by Backpropagation

The source range and depth backpropagation estimation procedure used in this work was introduced by Voltz and Lu [21] for a vertical hydrophone array. Let us assume an ideal noise-free scenario, where a source is transmitting a signal, and one is able to accurately determine the elevation and associated arrival times of the signal echoes impinging on a receiver. By the reciprocity principle, a ray launched from the receiver at a given angle has the same trajectory as an echo received at that elevation angle. One says that such a ray is backpropagated. One should note that backpropagation, like other model-based methods, requires *a priori* knowledge of the environment (e.g., sound speed profile, bathymetry and bottom parameters) [33].

Source localization is possible by tracing the trajectories of, at least, two echoes impinging on the receiver from different elevation angles and searching for range-depth points, where trajectories intersect each other. Several intersection points can occur along the trajectories; however, the source position can be uniquely determined by using the knowledge of relative time delays between echoes. This can be done by time aligning the different rays, *i.e.*, delaying rays by the estimated relative delays. Since the ray trajectories depend on the sound speed profile and bathymetry, the method is sensitive to uncertainties in those parameters. In practice, estimates of the elevation and travel time are also affected by noise. The estimate, $\hat{r}$, of the source range, r, and $\hat{z}$ of the depth, z, can be obtained by joint minimization of the mean square error:

$$\hat{r} = \arg\{\min_{r,\tau_a} \sum_{m=1}^{M} [r_m(\tau_a) - r]^2\} \tag{14}$$

$$\hat{z} = \arg\{\min_{z,\tau_a} \sum_{m=1}^{M} [z_m(\tau_a) - z]^2\} \tag{15}$$

where τ_a is the aligned time, $r_m(\tau_a)$ and $z_m(\tau_a)$ are, respectively, the range and depth of the m-th ($m = 1 \cdots M$) backpropagated ray trajectories at time, τ_a. The well-known solution for this optimization problem is the average range and depth given by:

$$\hat{r}(\tau_a) = \frac{1}{M} \sum_{m=1}^{M} r_m(\tau_a) \tag{16}$$

$$\hat{z}(\tau_a) = \frac{1}{M} \sum_{m=1}^{M} z_m(\tau_a) \tag{17}$$

at a time, τ_a, where the range variance, $\sigma_r^2(\tau_a)$, and the depth variance, $\sigma_z^2(\tau_a)$, are obtained when:

$$\sigma_r^2(\tau_a) = \frac{1}{M} \sum_{m=1}^{M} [r_m(\tau_a) - \hat{r}(\tau_a)]^2 \tag{18}$$

$$\sigma_z^2(\tau_a) = \frac{1}{M} \sum_{m=1}^{M} [z_m(\tau_a) - \hat{z}(\tau_a)]^2 \tag{19}$$

jointly attain the minimum. Thus, a possible objective function to be minimized is the sum of variances, *i.e.*:

$$\sigma^2(\tau_a) = \sigma_r^2(\tau_a) + \sigma_z^2(\tau_a) \tag{20}$$

or, equivalently, its square root, σ (aka standard deviation).

This method is numerically very efficient, since it only requires the computation of the trajectory and respective travel time of few rays and a one-dimensional search.

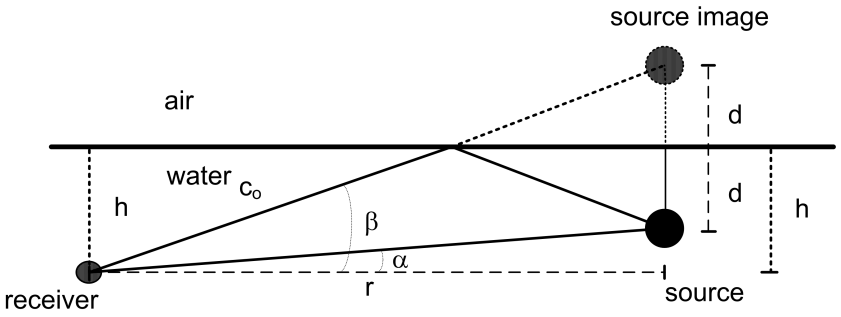
2.5. Range-Depth Estimation by the Image Method

Assuming that the sound speed profile is (approximately) isovelocity and that the geometry of the experiment allows for a direct and a surface-reflected echo between the source and the receiver, the source range and depth can be estimated by simple geometric relations based on the source image method (Figure 1). α being the elevation of the direct echo and β the elevation of the surface-reflected echo as seen by the receiver, the source range, r, and depth, d, are related by the following equations:

$$\begin{aligned} r \tan \alpha + d &= h \\ r \tan \beta - d &= h \end{aligned} \tag{21}$$

where h is the receiver depth. The solution of Equation (21) for r and d is straightforward. Although limited to a particular geometry, the source image method allows one to determine the range and depth of a source without the need for a ray tracing code, which can be advantageous to implement in light systems.

Figure 1. Geometry of the source image method.



3. Numerical Examples

For testing the methods presented in the previous section and anticipating their performance on real data, the environmental and geometry scenario used for simulation is based on that of the Makai Experiment [34]. The simulation scenario is shown in Figure 2, where a vector sensor is suspended from a research vessel at 40 m depth. The sound source was suspended at 10 m depth and moved along a straight line between 100 m and 1,000 m range from the receiver, transmitting a linear frequency-modulated (LFM) chirp with a duration of 50 ms and a frequency band of 8–14 kHz. The bathymetry is range-dependent with water depth 104 m at the receiver and 203 m at the source when at 1,000 m range. The sound speed profile at the vector sensor location is represented in Figure 2, showing a thick mixed layer with a thermocline starting at 60 m depth. The bottom was modeled as a half-space characterized by the values estimated in [10]: a bottom compressional speed of 1,575 m/s, a density of 1.5 g/cm^3 and a compressional attenuation of 0.6 dB/λ. In these simulations, it will be assumed that the azimuth is known; thus, only horizontal and vertical particle velocity components are considered. The channel pressure and particle velocity field frequency response were modeled by the cTraceo ray tracing model [35]. The time domain received waveforms were computed by Fourier synthesis. Figure 3 shows the eigenrays (paths of the echoes that impinge on the receiver) when the source is at a range of 900 m (Figure 3(a)) and the arrival patterns for the pressure, horizontal and vertical particle velocity (Figure 3(b–d), respectively). The arrival patterns for the pressure are normalized by the overall maximum, whereas the arrival patterns for the particle velocity components are normalized by the joint overall maximum. Note that the scale used for the particle velocity arrival patterns are different. In the eigenrays plot, one can notice a direct echo and a surface reflected echo, which correspond to the earliest peaks in the arrival patterns plots. These echoes have low angles relative to the horizontal plane containing the source, which decrease with an increasing source-receiver range. This behavior is observed in the particle velocity components, especially in the vertical component, where the amplitude of the peaks in the first cluster decreases as the source gets further way from the receiver. The latter echoes are bottom reflected. These echoes are also clustered in groups of two echoes depending on the number of surface reflections. Bottom reflected echoes have wider angles and lower amplitudes (especially pressure), mainly due to the attenuation in the bottom. Note that in the vertical particle velocity arrival patterns, the latter peaks have relatively higher amplitudes, since for wider angles, the

module of $\sin(\Phi)$ tends to unity. The amplitudes of the different echoes as seen by the vector sensor components illustrate, in the time domain, the spatial filtering capabilities of a single vector sensor.

Figure 2. Makai'05 scenario used for the simulation: the source deployed at 10 m depth moves between a 100 and 1,000 m range from the vector sensor deployed at 40 m. The sound speed profile shows a large mixed layer, characteristic of Hawaii, USA. The bottom parameters are those estimated in [10].

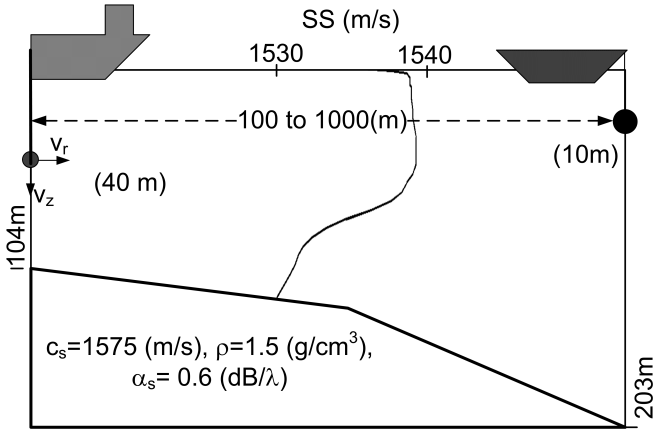
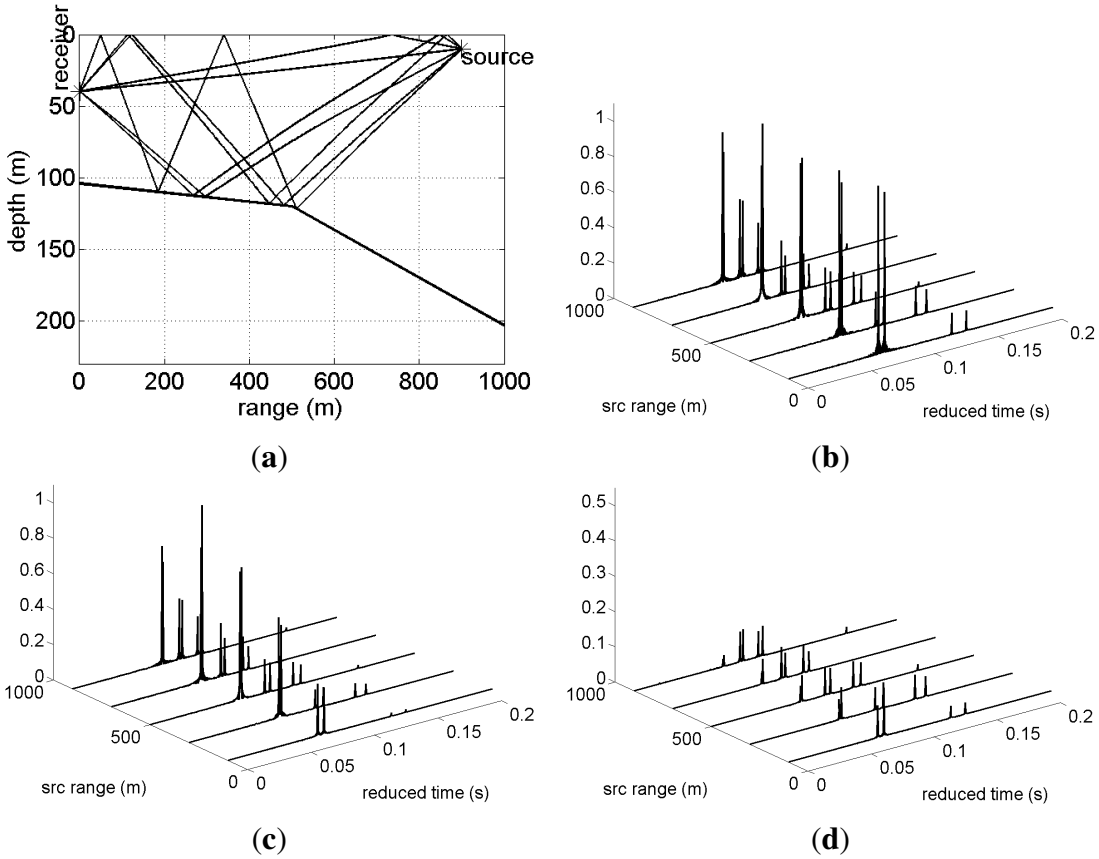


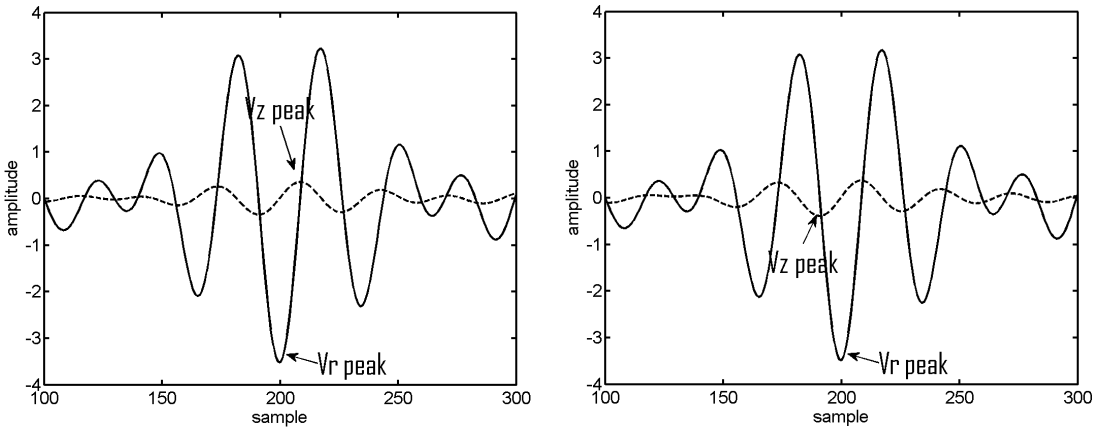
Figure 3. Makai'05 simulation scenario of Figure 2: eigenrays for a source at 900 m (**a**); and arrival patterns for various source ranges—pressure (**b**); horizontal particle velocity (**c**); and vertical particle velocity (**d**).



A number of 100 realizations were generated according to model Equation (8), but limited to a horizontal and a vertical vector sensor component and source ranges from 100 to 900 m. The additive noise was independent for each component and obeyed a Gaussian distribution. Two different signal to noise ratios (SNR), 5 and 20 dB, at the receiver were generated. Since, in the considered geometry, the received energy in the horizontal component is higher than in the vertical component, as can be seen from the arrival structure shown in Figure 3, the SNR is related to the horizontal component, thus representing the worst case scenario. The elevation angles of the four earlier echoes impinging on the vector sensor were estimated independently for the different realizations. Then, the mean and the standard deviation were computed for each echo from the realization ensemble. In order to reduce time and amplitude discretization-related errors, the sampling frequency of the received waveforms (48 kHz) was increased (interpolated) by a factor of 10. The results are summarized in Table 1, where the lines labeled *true* show the values predicted by the forward ray tracing model.

The values in brackets represent the standard deviation. The star mark appears when a sign error occurs at least once in the ensemble of realizations for the given echo. It can be seen that the absolute errors are always smaller then 1.2 degrees, and as expected, the standard deviation increases with decreasing SNR. Generally, the SNR at the receiver for distant signals decreases, which, in turn, also contributes to a higher variance of the estimates. Unsurprisingly, the sign error of the estimates increases significantly with decreasing SNR. The autocorrelation function of an LFM chirp is an oscillatory function; thus, due to the noise, the location of the absolute maximum that determines the sign of the elevation estimates can be shifted by an oscillation period, therefore resulting in a sign error. This situation is illustrated in Figure 4, showing the amplitude-delay curve (horizontal and vertical particle velocity) in the neighborhood of the first echo for two realizations at a source range of 300 m. One can observe that the sign of the peak of the vertical particle velocity changed among realizations. This sign ambiguity of the elevation estimates could be, in principle, minimized using a second vector sensor.

Figure 4. Zoom of the amplitude-delay curve in the neighborhood of the first echo for two signal realizations (5 dB SNR) at a source distance of 300 m showing the expected behavior (**a**) and a sign error (**b**). The horizontal particle velocity (Vr) is represented by the solid line and the vertical particle velocity (Vz) by the dashed line. The arrows indicate the peaks.



(**a**) (**b**)

Table 1. Estimated angles of the four earliest echoes impinging on the vector sensor at different source distances as given by the forward model (true) and estimated considering an signal to noise ratios (SNR) of 20 and 5 dB. The values in curved brackets represent the standard deviations. The star mark indicates that at least one sign error occurred in the ensemble of estimates for the given echo.

Source Depth 10 m		**Echo Number**				
Range [m]	**SNR [dB]**	**1 [°]**	**2 [°]**	**3 [°]**	**4 [°]**	
	True	16.3	26.1	−60	−63	model
100	20	17.3 (0.1)	27.3 (0.1)	−59.4 (0.5)	−62.5 (0.7)	estimate
	5	17.3 (0.4)	27.4 (0.4)	−59.4 (3.4) *	−62.1 (3.7) *	estimate
	True	5.6	9.3	−30.8	−33.8	model
300	20	5.9 (0.1)	9.9 (0.1)	−31.8 (0.4)	−34.4 (0.3)	estimate
	5	5.9 (0.4) *	10.1 (0.5)	−31.9 (2.3) *	−34.5 (2.0) *	estimate
	True	3.3	5.5	−20.7	−22.8	model
500	20	3.7 (0.1)	5.8 (0.1)	−21.8 (0.3)	−23.8 (0.3)	estimate
	5	3.7 (0.5) *	5.8 (0.5)	−21.9 (1.5) *	−24.0 (2.1) *	estimate
	True	2.3	3.9	−15.9	−17.6	model
700	20	2.5 (0.1)	4.1 (0.1)	−16.8 (0.2)	−18.8 (0.4)	estimate
	5	2.7 (0.6) *	4.1 (0.4) *	−16.8 (1.3) *	−18.7 (2.0) *	estimate
	True	1.8	2.9	−13.2	−14.5	model
900	20	1.9 (0.1)	3.1 (0.1)	−14.1(0.2)	−15.2 (0.2)	estimate
	5	2.1 (0.5) *	3.2 (0.5) *	−14.1 (1.0) *	−15.3 (1.0) *	estimate

Next, using the elevation angle estimates for the 5 dB SNR presented in Table 1 and respective arrival times (not shown), the source range and depth were estimated applying the ray backpropagation and source-image method. For the ray backpropagation method, the estimates were obtained considering three different sets of echoes (the results are summarized in Table 2): all four echoes, the direct and the surface reflected echoes only (column labeled *echoes 1&2*) and the bottom reflected echoes only (column labeled *echoes 3& 4*). For the image method only, the direct and surface-reflected echoes are considered. Figure 5 illustrates the backpropagation method when the source is at a 900 m range. Figure 5(a) presents the backpropagated rays, where one can notice that the rays do not intercept at a single point, due to angle and travel time estimation errors. The objective function dependence in the range is plotted for four echoes in Figure 5(b) and for two echoes in Figure 5(c). For each case, the estimated source range is given by the minimum of the objective function and corresponding depth (respectively, lower and upper plots of Figure 5(b,c)). Whereas the objective function for four backpropagated rays has a single sharp minimum (Figure 5(b)) when only two rays are backpropagated, the shape of the objective function is smooth, giving rise to an increased ambiguity (Figure 5(c)), especially in ranges close to the receiver, because of the very short relative time delay between echoes and the close shooting angles.

Generally, range and depth estimation errors increase with source range, since small angle perturbations from the nominal value give rise to larger range depth perturbations. However, the errors

are less than 2 m in depth and 75 m in range at the maximum range of 900 m, the worst case considered. The results obtained from the latter echoes, which are bottom reflected, present higher estimation errors than those obtained from direct and surface reflected echoes. Bottom reflection is frequency dependent, which is accounted for by the forward propagation model used to synthesize the received signal; however, the backpropagation uses only the echo path and travel time computed at a given frequency (in general, the middle frequency of the signal band). Moreover, bottom reflected echoes are more attenuated (depending on the bottom structure and the angle of incidence); thus, they are more affected by noise. One can also notice that the source image method gives reasonable estimates in the considered scenario, even at longer ranges, because the sound speed profile is almost isovelocity in the source-receiver layer.

Figure 5. Source localization results for the scenario of Figure 2 using the backpropagation method for source range and depth, 900 m and 10 m, respectively and SNR = 5 dB: true source and receiver position (represented by the star) and backpropagated rays (**a**), ambiguity curves (σ) and source-localization plot considering four rays (**b**) and two rays (**c**). The arrows in plots (**b**) and (**c**) indicate the estimated source position (upper plots) and the corresponding minimum of the ambiguity curve (lower plots).

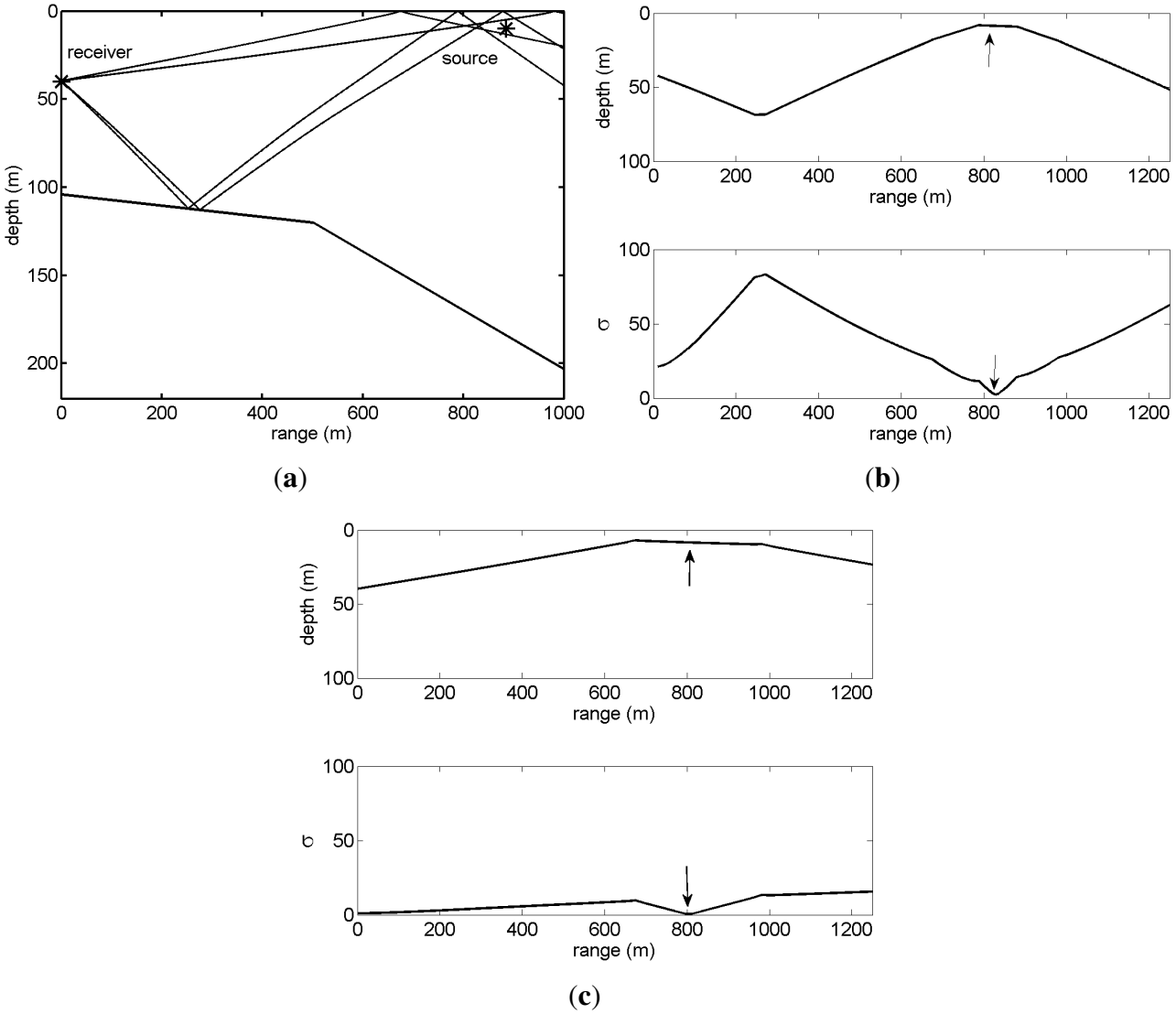


Table 2. Estimates of range and depth of a simulated source at 10 m depth, between a 100 and 900 m range from the receiver. The backpropagation method was applied to the four echoes, the earlier two echoes (1& 2) and the last two echoes (1& 2). The source image method was applied to the earlier two echoes. The range and depth estimates were obtained using estimates of the elevation angles from simulated data with 5 dB SNR.

Simulated	**Ray Backpropagation**						**Image Method**	
Depth 10 m	**4 Echoes**		**Echoes 1&2**		**Echoes 3 &4**		**Echoes 1&2**	
Range [m]	**Range [m]**	**Depth [m]**	**Range [m]**	**Depth [m]**	**Range [m]**	**Depth [m]**	**Range [m]**	**Depth [m]**
100	102.5	10.0	99.4	10.2	104.3	9.7	95.7	9.9
300	299.3	10.2	300.1	11.3	290.3	8.9	282.1	10.5
500	499.4	9.8	500.9	9.5	469.1	9.4	477.6	8.8
700	649.1	9.6	668.1	8.2	640.2	8.3	649.5	10.9
900	829.2	8.6	804.5	8.4	830.1	8.4	857.7	8.3

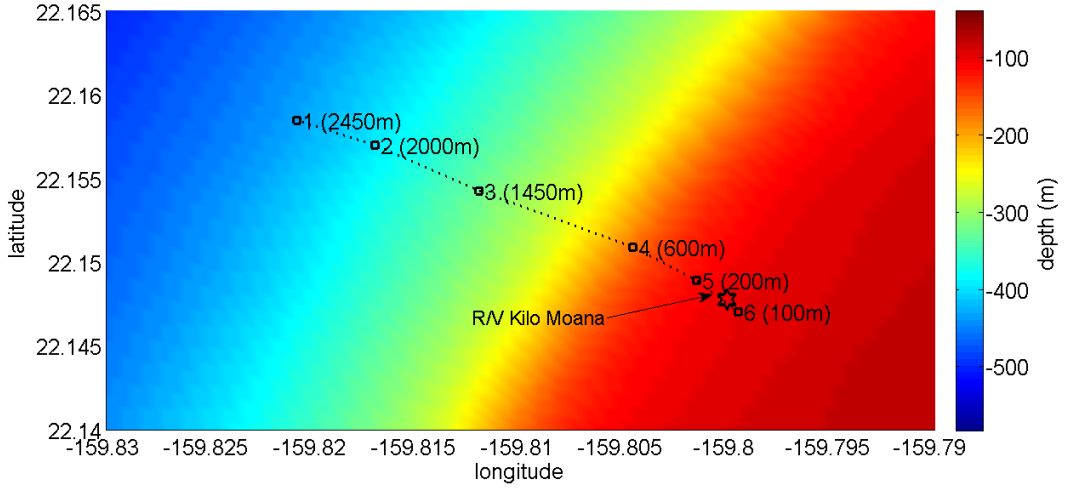
4. Experimental Data Results

4.1. Experimental Setup

The data set analyzed here was acquired during the Makai Experiment (Makai'05), which took place off the west coast of Kauai I. in September 2005. The Makai'05 experiment was devoted to high frequency acoustics, and details of the experimental setup are described in [34]. This paper is concerned with the data acquired during the field calibration event, whose geometry is identical to that used in the numerical example (Figure 2). The vector sensor acquisition system used in the experiment was composed by four Wilcoxon TV-001 vector sensors [1,6,22], configured in a vertical array with 10 cm element spacing. The system was suspended off the stern of the research vessel, Kilo Moana, with a 150 kg weight at the bottom, to ensure that the array stayed as close to the vertical as possible. The z-axis was vertically oriented downwards, with the deeper sensor at 40 m. In the present data analysis, only the vector sensor at 40 m is considered. In the field calibration event, a Lubell 916C sound source deployed at 10 m depth from a small rubber boat was towed during a period of one hour from a 2.5 km distant point towards the research vessel that was holding a fixed position. Figure 6 shows the rubber boat GPS fixes and the research vessel (R/V) Kilo Moana location superimposed on the bathymetry of the area. The rubber boat track starts at a deeper location, moving along the continental steep slope towards a smooth and uniform area with a water depth of approximately 104 m. The source signals used for localization in this work were transmitted at various locations between approximately 600 m northwest and 100 m southeast of the R/V Kilo Moana, respectively, GPS fixes 4 and 6 in Figure 6. Ground truth measurements were carried out in this area during previous experiments and showed that most of the bottom in the area is covered with coral sands over a basalt hard bottom. The acoustic parameters of the sediment were estimated by Santos *et al.* [10], for the bottom compressional speed, attenuation and density—values already used in Section 3 of this paper. The Lubell 916C sound source

transmitted 50 ms long LFM chirps spanning the 8–14 kHz band, transmitted in blocks of 30 chirps from various ranges along the track. The signals were acquired at a sampling frequency of 48 kHz. Due to a technical problem, the time stamp in the data is not synchronized with GPS; therefore, the precise position of the source for the various blocks of acquired data is not available. Figure 7 shows the time series (Figure 7(a)) and spectrograms (Figure 7(b)) of the received waveforms, when the source was at approximately 350 m range, where a strong multipath effect can be seen. The response of the vertical component should also be noted, which emphasizes the latter arrivals when compared with pressure or horizontal components: the relative amplitude of the latter arrival appearing approximately at 0.1 s is in the vertical component, higher than in the other components. This differentiating spatial response of the vertical component was explored in [10,36] to enhance the resolution of bottom parameter estimates. Technical problems with the gain of the pre-amplifiers explain why the amplitude of signals received from larger distances and those that suffer bottom reflections, which are more attenuated, is very low. In the next section, only the transmissions at a source-receiver range smaller than 500 m and the direct and surface reflected arrivals are considered.

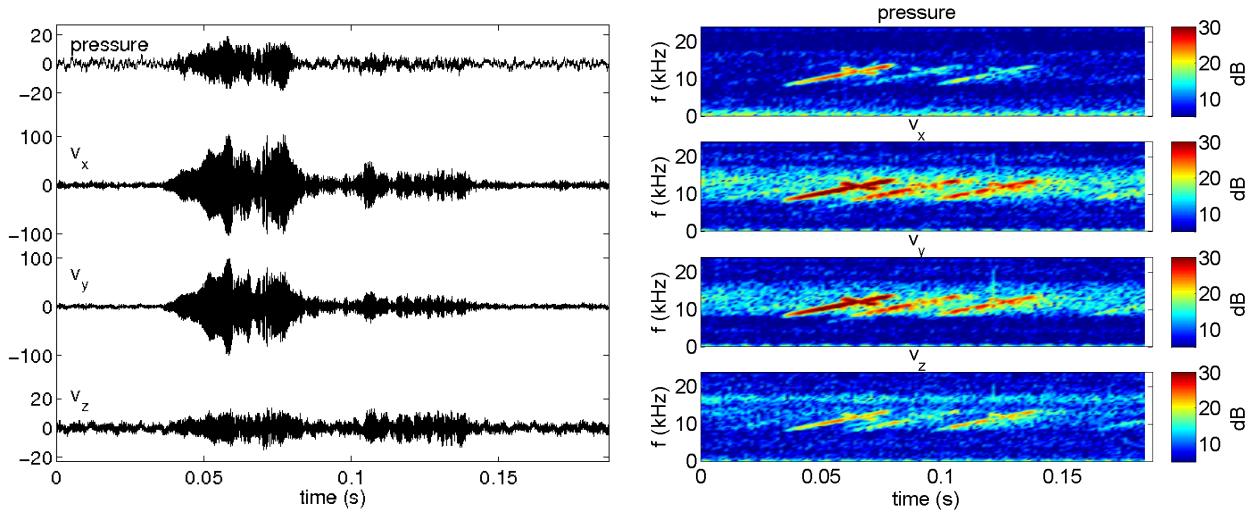
Figure 6. Bathymetry of the Makai'05 experimental area for the field calibration event with the superimposed research vessel R/V Kilo Moana location (pentagon) and GPS fixes of the source rubber boat (square). The values in brackets represent the distance to the R/V Kilo Moana.



4.2. Azimuth, Elevation and Travel Time Estimates

The received signal was filtered for a ship noise band (90–350 Hz) and an acoustic source band (8–14 kHz) by linear phase bandpass filtering. The ship noise was used to determine the orientation of the x-axis relative to the ship. For this purpose, the azimuth of the ship noise was estimated using the intensity-based method. The estimates were obtained by applying Equation (6) to ship noise received simultaneously with LFM chirps. The azimuth estimates are presented in a reference frame, where the x-axis is aligned west-east and the y-axis is aligned south-north. The elevation angles are considered positive towards the surface. The azimuth estimates obtained from the ship noise are very stable along the whole experiment [24,37].

Figure 7. Waveforms received from a 350 m distant source for the various vector sensor components (**a**) and respective spectrograms (**b**).

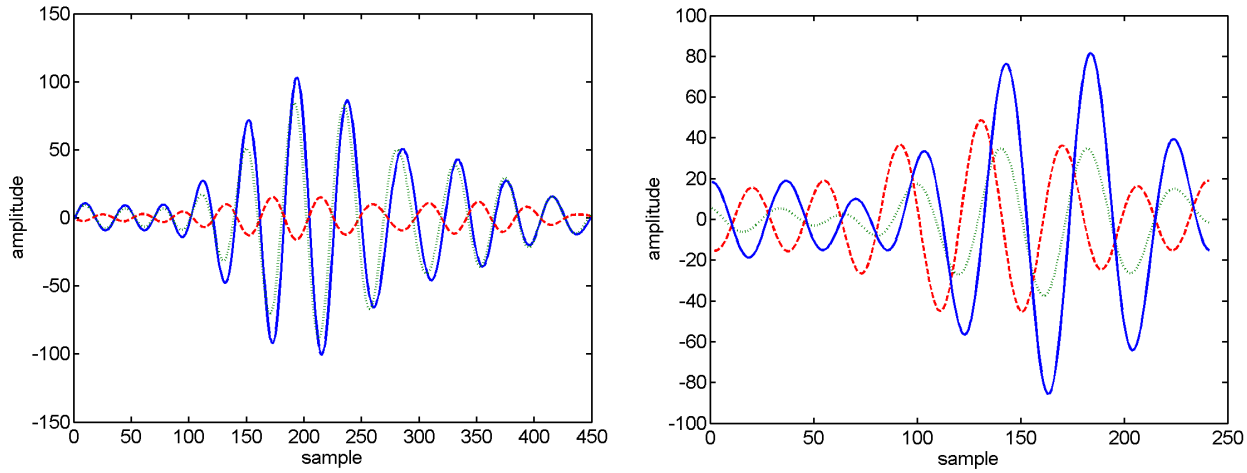


(**a**) (**b**)

As a first step to localize the source, the arrival times and amplitudes of the various echoes impinging on the vector sensor from each transmitted chirp were estimated from the arrival patterns using Equation (11). Since the transfer function of the transmission chain was unknown, a simulated undistorted LFM chirp was used in the amplitude arrival time estimation procedure. In order to increase the travel time and amplitude resolution, the acquired signal was up sampled by a factor of 10, becoming the virtual sampling frequency of 480 kHz. Figure 8 shows the delay-amplitude curve in the neighborhood of the first echo for the three vector sensor components at two different source ranges, illustrating that the travel time of an echo varies significantly among vector sensor components. While in Figure 8(a), the various components are nearly in phase, in Figure 8(b), a significant deviation occurs. Data inspection revealed that the perturbations varied among distances and among echoes, but were stable for the same echo number among transmissions at a given distance. Thus, for azimuth and elevation estimates, using Equations (12) and (13), respectively, the different components were treated independently. The amplitude of a given component was considered, where its absolute maximum occurs, whereas the associated travel time to be used in the backpropagation algorithm was that of the z-component. For the reasons explained above, only the direct and surface-reflected echoes were used for source range-depth estimation. For each distance, six chirps in the signal block were processed; azimuth and elevation were estimated. For the azimuth estimation only, the direct echo was considered. Table 3 presents the mean values and the standard deviations of those estimates, and for the sake of clarity, the range estimates are discussed in next section. As discussed in the numerical examples, the sign of the estimates suffer from a large ambiguity; thus, the sign of the elevation angle of the chirp echoes was assigned using the previous knowledge of the geometry. However, the sign of the azimuth of the ship noise was determined directly from the data. The orientation of the vector sensor relative to the R/V Kilo Moana given by the azimuth estimates from the ship noise was stable along time. This behavior was also observed for the other Makai'05 events for a full vector sensor array [37] and a single vector sensor [24]. One should note that the standard deviations of the elevation angles are lower for

direct than for surface reflected echoes, which have a smaller amplitude and are subject to perturbations induced by the ocean surface. The high standard deviation of azimuth estimates at position min 54 and min 57 can be explained by the shorter range. A horizontal displacement at a shorter range gives rise to a higher azimuth perturbation when compared with similar displacement at a longer range.

Figure 8. Zoom of the amplitude-delay curve in the neighborhood of the first echo for the x-component (solid line), y-component (dotted line) and z-component (dashed line) at approximate source-receiver distances of 250 m (**a**) and 350 (**b**).



(**a**) (**b**)

Table 3. Mean azimuth and elevation estimates obtained at various instants of the Makai'05 field calibration event for the ship noise (azimuth only) and for the broadband sound source at 10 m depth and a range between 100 and 400 m (estimated from acoustic data, Figure 10). The values in brackets represent the estimated standard deviation.

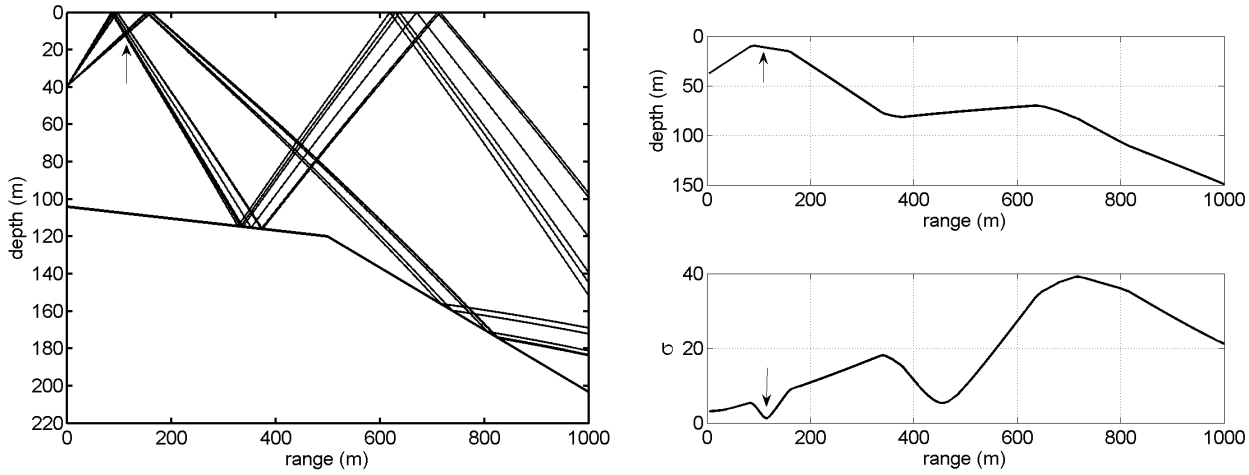
Time [min]	Source Range [m]	Azimuth [°]		Elevation [°]	
		Ship Noise 90–350 Hz	Source 8–14 kHz	Dir.Echo	Surf.Ref.Echo
35	386.3	132.4	136.1 (0.2)	4.1 (0.1)	7.5 (0.1)
48	357.5	132.2	137.3 (0.8)	4.9 (0.1)	7.5 (0.7)
50	279.8	134.4	140.9 (0.4)	5.5 (0.3)	9.8 (1.3)
54	100.8	133.1	135.1 (2.3)	16.7 (0.2)	25.7 (2.0)
57	114.2	131.7	−10.2 (1.0)	14.0 (0.4)	23.8 (1.0)
60	145.4	132.2	−12.6 (0.6)	12.4 (0.2)	17.8 (1.1)

4.3. Range-Depth Estimates

The range and depth of the source were estimated with the ray backpropagation method using the elevation angles of the direct and surface reflected echoes and respective relative arrival times. In order to obtain an estimate of the standard deviation of the estimates, the objective function is an average of

the objective functions computed for each single realization of the transmitted chirp signal. Figure 9(a) shows the direct and surface reflected backpropagated rays with elevation angles estimated from six chirps transmitted at min 57, where the source was at a range of 114 m. The overall objective function (ambiguity curve) dependence in the range computed from these rays (and travel time estimates) is shown in the lower plot of Figure 9(b), whereas the estimated source range is given by the minimum of the objective function and the corresponding depth in the upper plot. The range and depth estimates at various source distances obtained by backpropagation are presented in Table 4. Column σ represents the minimum of the square root of the objective function, where the smaller values (smaller variance), are attained at closer ranges. Model-based source localization methods are, in general, not considered for real-time implementations, because of the time needed to compute the optimization procedure, which requires a large number of forward model runs, but a non-optimized Matlab implementation of the ray backpropagation method took less than 4 s in a current laptop. It is expected that further code optimization would allow for real-time application.

Figure 9. Makai'05 source range-depth estimation at min 57 of the field calibration event considering six sets of backpropagated rays (12 rays) (**a**). The ambiguity curve (**b**) has the minimum (indicated by the arrow) at the 114 m range, lower plot, corresponding to 11.5 m depth in the upper plot.



(**a**) (**b**)

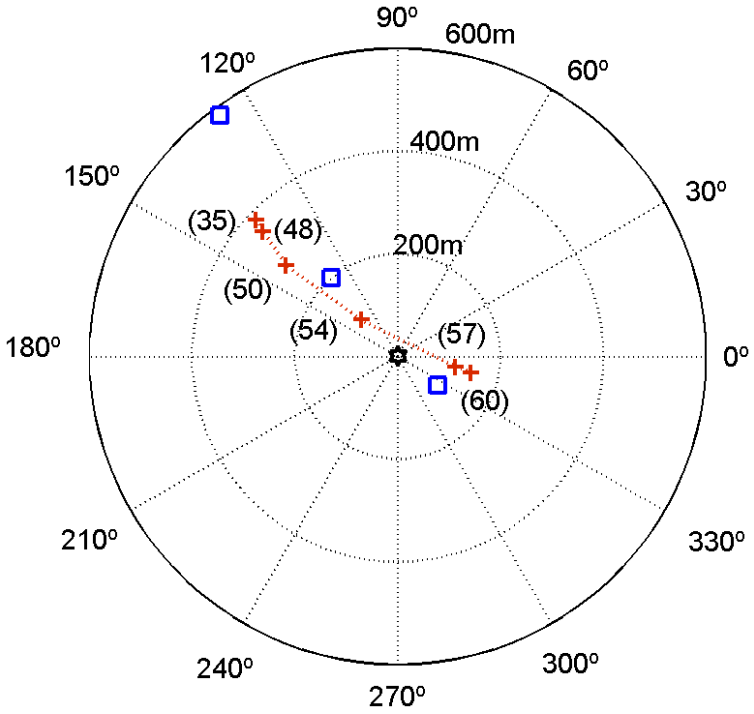
For comparison purposes the results obtained using the source image method are also shown in Table 4. At longer distances, the difference between the estimates obtained by both methods increases, due to the cumulative effect of considering a constant sound speed with the image method. One should remark that the source depth estimates are in close agreement with the nominal depth of the source of 10 m. Figure 10 shows a polar plot with the location of the source using the source range and azimuth estimates, which are represented by stars. The squares represent the positioning of the source relative to the R/V Kilo Moana (at the origin) estimated from the ship's GPS and a handheld GPS device on the source's boat. Unfortunately, the handheld GPS device had no recording capabilities; thus, the source path between GPS fixes is uncertain. Since the range estimates by GPS and by acoustics are affected by some offset, because of the different location of the vector sensor and GPS on board the R/V Kilo

Moana, and there is no time stamp in the acoustic data, this did not allow for synchronization between acoustic and GPS data. The source track derived from acoustic data are in relatively close agreement with GPS fixes.

Table 4. Source range and depth estimates at various instants of the Makai'05 field calibration event using the ray backpropagation method and the image method. The column marked, σ, represents the minimum of the square root value of the objective function used with the backpropagation method. The true source depth is 10 m. The estimated range values compare with the GPS fixes in Figure 10.

Time [min]	Ray Backpropagation			Image Method	
	Range [m]	Depth [m]	σ [m]	Range [m]	Depth [m]
35	386.3	10.54	6.7	391.1	11.7
48	357.5	10.55	9.4	367.1	8.3
50	279.8	11.1	1.8	298.8	11.0
54	100.8	10.1	1.5	101.9	9.2
57	114.2	11.5	1.0	115.0	11.0
60	145.4	7.9	1.2	146.7	7.3

Figure 10. Estimated source location (cross marks) relative to the R/V Kilo Moana (located at the origin) of the Makai'05 field calibration event. The square marks indicate GPS fixes obtained with a handheld GPS on board the source rubber boat. The values in brackets represent the time (in min) of Tables 3 and 4.



5. Conclusions

This paper illustrates the spatial filtering capabilities of a vector sensor applied to source localization of a known broadband signal in a multipath environment. It was shown that the estimation of the angle of arrival (elevation) of a single echo was possible. Given the estimates of the amplitudes of the echoes in the v_x,v_y and v_z vector sensor components, a method to estimate the source azimuth and the elevation was presented. The elevation angles of the direct and surface reflected echoes were used to estimate source range and depth localization by a ray backpropagation algorithm. The method was discussed in the context of simulated data and for a real data set acquired during the Makai Experiment. It was shown that for ranges below 500 m, it was possible to estimate the source range and depth in agreement with the known geometry of the experiment. To the best of our knowledge, this is the first work in open literature that reports 3D localization results with a single vector sensor in a shallow water environment. In comparison with other model-based methods discussed in the literature for source localization using a single device (hydrophone), the present method explores the spatial filtering capabilities of a single vector sensor to significantly reduce the number of forward model runs; thus, it can be potentially implemented in low end or light real-time systems.

Acknowledgments

The authors would like to thank Michael Porter, chief scientist for the Makai Experiment, Jerry Tarasek for the loan of the vector sensor array used in the experiment and Paul Hursky, Martin Siderius and Bruce Abraham for providing assistance with the data acquisition. The Makai Experiment was supported by the US Office of Naval Research. This work was funded by National Funds through Foundation for Science and Technology (FCT) under project SENSOCEAN (PTDC/EEA-ELC/104561/2008).

Conflict of Interest

The authors declare no conflict of interest.

References

1. Silvia, M.T.; Richards, R.T. A Theoretical and Experimental Investigation of Low-frequency Acoustic Vector Sensors. In Proceedings of the Oceans 2002 Conference (OCEANS'02), Biloxi, MI, USA, 29–31 October 2002.
2. D'Spain, G.; Hogkiss, W.S.; Edmonds, G.L. Initial Analysis of the Data from the Vertical DIFAR Array. In Proceedings of the Oceans 1992 Conference (OCEANS'92), Newport, USA, 26–29 October 1992.
3. Miron, S.; Le Bihan, N.; Mars, J.I. Vector-sensor MUSIC for polarized seismic sources localization. *J. Appl. Signal Process.* **2005**, *2005*, 74–84.
4. Miron, S.; Le Bihan, N.; Mars, J. Quaternion-MUSIC for vector-sensor array processing. *Signal Process. IEEE Trans.* **2006**, *54*, 1218–1229.

5. Tam, P.K.; Wong, K.T. Cramér-Rao bounds for direction finding by an acoustic vector sensor under nonideal gain-phase responses, noncollocation, or nonorthogonal orientation. *IEEE Sens. J.* **2009**, *9*, 969–982.
6. Shipps, J.C.; Abraham, B.M. The Use of Vector Sensors for Underwater Port and Waterway Security. In Proceedings of the ISA/IEEE Sensors for Industry Conference, New Orleans, LA, USA, 24 August 2004.
7. Song, A.; Badiey, M.; Hursky, P.; Abdi, A. Time Reversal Receivers for Underwater Acoustic Communication Using Vector Sensors. In Proceedings of the Oceans 2008 Conference (OCEANS'08), Quebec City, Canada, 15–18 September 2008.
8. Han-Shu, P.; Feng-Hua, L. Geoacoustic inversion based on a vector hydrophone array. *Chin. Phys. Lett.* **2007**, *24*, 1977–1980.
9. Santos, P.; Rodriguez, O.; Felisberto, P.; Jesus, S. Geoacoustic Matched-Field Inversion Using a Vertical Vector Sensor Array. In Proceedings of the 3rd International Conference and Exhibition on Underwater Acoustic Measurements: Technologies and Results, Nafplion, Greece, 21–26 June 2009; pp. 29–34.
10. Santos, P.; Rodriguez, O.C.; Felisberto, P.; Jesus, S.M. Seabed geoacoustic characterization with a vector sensor array. *J. Acoust. Soc. Am.* **2010**, *128*, 2652–2663.
11. Miron, S.; Le Bihan, N.; Mars, J. Joint Estimation of Direction of Arrival and Polarization Parameters for Multicomponent Sensor Array. In Proceedings of the 66th European Association of Geoscientists and Engineers Conference and Exhibition, Paris, France, 7–10 June 2004.
12. Sherman, C.H.; Butler, J.L. *Transducers and Arrays for Underwater Sound*; The Underwater Acoustic series, Springer: Berlin/Heidelberg, Germany, 2007.
13. Wu, Y.I.; Wong, K.T.; Yuan, X.; Lau, S.K.; keung Tang, S. A directionally tunable but frequency-invariant beamformer on an acoustic velocity-sensor triad to enhance speech perception. *J. Acoust. Soc. Am.* **2012**, *131*, 3891–3902.
14. Liu, X.; Xiang, J.; Zhou, Y. Passive tracking and size estimation of volume target based on acoustic vector intensity. *Chin. J. Acoust.* **2001**, *20*, 225–238.
15. Awad, M.; Wong, K. Recursive least-squares source tracking using one acoustic vector sensor. *Aerosp. Electron. Syst. IEEE Trans.* **2012**, *48*, 3073–3083.
16. Rahamim, D.; Tabrikian, J.; Shavit, R. Source localization using vector sensor array in a multipath environment. *Signal Process. IEEE Trans.* **2004**, *52*, 3096–3103.
17. Arunkumar, K.; Anand, G. Source Localisation in Shallow Ocean Using Vertical Array of Acoustic Vector Sensors. In Proceedings of the European Signal Processing Conference (EUSIPCO), Poznan, Poland, 3–7 September 2007; pp. 2454–2458.
18. Hurtado, M.; Nehorai, A. Performance analysis of passive low-grazing-angle source localization in maritime environments using vector sensors. *Aerosp. Electron. Syst. IEEE Trans.* **2007**, *43*, 780–789.
19. Abdi, A.; Guo, H.; Sutthiwan, P. A New Vector Sensor Receiver for Underwater Acoustic Communications. In Proceedings of the Oceans 2007 Conference (OCEANS'07), Vancouver, Canada, 29 September–4 October 2007.

20. Hawkes, M.; Nehorai, A. Wideband source localization using a distributed acoustic vector-sensor array. *IEEE Trans. Signal Process.* **2002**, *27*, 628–637.
21. Voltz, P.; Lu, I.T. A time-domain backpropagating ray technique for source localization. *J. Acoust. Soc. Am.* **1994**, *95*, 805–812.
22. Abraham, B.M. Ambient Noise Measurements with Vector Acoustic Hydrophones. In Proceedings of the Oceans 2006 Conference (OCEANS'06), Boston, MA, USA, 18–21 September 2006.
23. Nehorai, A.; Paldi, E. Acoustic vector-sensor array processing. *IEEE Trans. Signal Process.* **1994**, *42*, 2481–2491.
24. Felisberto, P.; Santos, P.; Jesus, S. Tracking Source Azimuth Using a Single Vector Sensor. In Proceedings of the Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), Venice, Italy, 18–25 July 2010; pp. 416–421.
25. Hermand, J.P. Broad-band inversion in shallow water from waveguide impulse response measurements on a single hydrophone: Theory and experimental results. *IEEE J. Ocean. Eng.* **1999**, *24*, 41–66.
26. Jesus, S.; Porter, M.; Stéphan, Y.; Démoulin, X.; Rodriguez, O.; Coelho, E. Single hydrophone source localization. *IEEE J. Ocean. Eng.* **2000**, *25*, 337–346.
27. Gac, J.C.L.; Asch, M.; Stéphan, Y.; Démoulin, X. Geoacoustic inversion of broadband acoustic data in shallow water on a single hydrophone. *IEEE J. Ocean. Eng.* **2003**, *28*, 479–493.
28. Felisberto, P.; Jesus, S.M.; Stephan, Y.; Demoulin, X. Shallow Water Tomography with a Sparse Array during the INTIMATE'98 Sea Trial. In Proceedings of the MTS/IEEE Oceans 2003 Conference (OCEANS'03), San Diego, USA, 22–26 September 2003; pp. 571–575.
29. Felisberto, P.; Rodriguez, O.; Jesus, S.M. Estimating the Multipath Structure of an Underwater Channel Using a Single Vector Sensor. In Proceedings of the ECUA 2012 11th European Conference on Underwater Acoustics, Edinburgh, Scotland, 2–6 July 2013; Volume 17, pp. 1–9.
30. Hawkes, M.; Nehorai, A. Acoustic vector-sensor correlations in ambient noise. *IEEE J. Ocean. Eng.* **2001**, *26*, 337–347.
31. Abdi, A.; Guo, H. Signal correlation modeling in acoustic vector sensor arrays. *IEEE Trans. Signal Process.* **2009**, *57*, 892–903.
32. Wu, Y.; Wong, K.; Lau, S.K. The acoustic vector-sensor's near-field array-manifold. *IEEE Trans. Signal Process.* **2010**, *58*, 3946–3951.
33. Meyer, M.; Hermand, J.P. Backpropagation Techniques in Ocean Acoustic Inversion: Time Reversal, Retrogation and Adjoint Modelling. In *Acoustic Sensing Techniques for the Shallow Water Environment*; Caiti, A., Chapman, N., Hermand, J.P., Jesus, S.M., Eds.; Springer: Dordrecht, The Netherlands, 2006; pp. 29–46.
34. Porter, M.; Abraham, B.; Badiey, M.; Buckingham, M.; Folegot, T.; Hursky, P.; Jesus, S.; Kim, K.; Kraft, B.; McDonald, V. *et al*. The Makai experiment: High frequency acoustics. Proceedings of the 8th ECUA European Conferece on Underwater Acoustic, Carvoeiro, Portugal, 12–15 June 2006; Volume 1, pp. 9–18.
35. Rodriguez, O.; Collis, J.; Simpson, H.; Ey, E.; Schneiderwind, J.; Felisberto, P. Seismo-acoustic ray model benchmarking against experimental tank data. *J. Acoust. Soc. Am.* **2012**, *132*, 709–717.

36. Felisberto, P.; Rodriguez, O.; Santos, P.; Jesus, S.M. On the Usage of the Particle Velocity Field for Bottom Characterization. In Proceedings of the International Conference on Underwater Acoustic Measurements, Kos, Greece, 19–24 June 2011; pp. 19–24.
37. Santos, P.; Felisberto, P.; Hursky, P. Source Localization with Vector Sensor Array during the Makai Experiment. In Proceedings of the 2rd International Conference and Exhibition on Underwater Acoustic Measurements: Technologies and Results, Heraklion, Greece, 25–29 June 2007; pp. 895–900.