



PAMGUARD

PAMGUARD 4 Final Report

October 2008

Report Authors:

**Xiao Yan Deng
Douglas Gillespie
Jonathan Gordon
Ron McHugh
David McLaren
David Mellinger
Paul Redmond
Aaron Thode
Phil Trinder**

www.pamguard.org

Executive Summary

The PAMGUARD project began investigating next generation Passive Acoustic Monitoring (PAM) software in 2004. By PAMGUARD 3 (2006 - 2007) functionality analogous to that of the “RainbowClick” software was available and ruggedised by sea-trials and functionality analogous to the “Ishmael” software was under development. 2D localisation was provided using stereo hydrophones.

In this report we document PAMGUARD 4, the final phase of intensive PAMGUARD development. Beyond PAMGUARD 4 we anticipate that the PAMGUARD software will continue to evolve as part of the open-source community. Originally scheduled to run from January 2007 to December 2007 the loss of three key staff at OSU and Heriot-Watt during 2007 reduced available effort and caused the project to be rescheduled to complete in May 2008.

The three work packages in PAMGUARD 4 are the development of the Core Architecture (WP1), the Maintenance of software and user base (WP2) and the PAMGUARD Workshop (WP3). This document reports activities under (WP1), (WP2) and the PAMGUARD changes coming out of the CODA sea trial. The Workshop, WP3, was held at Heriot-Watt University in March 2007 and has previously been fully reported by Ecologic, and a summary is included in Section 2.5.

Key new functionalities added to the core PAMGUARD architecture in WP1 of the project are as follows. Tables of delivered functionality and module dependencies can be found in Appendix C.

- Adaptions to support offline analysis, e.g. a mode for reading stored data
- Advanced configuration utilities, e.g. storing and loading configurations
- Improved module management using a wizard
- A range of advanced displays
- An interface to SQL Databases, validated with MySQL and MS Access
- A standalone filter
- Improved sound acquisition capabilities
- An interface to industry standard ASIO sound cards with multiple channel selection
- A simulation capability
- NMEA data acquisition
- 3D Localisation capabilities

Key PAMGUARD maintenance activities undertaken in WP2 of the project are as follows:

- Users have been supported by email and phone
- Code has been maintained with activities including bug fixes, splitting development into a robust Core and a development (Beta) version, ports to multiple operating systems etc,
- Documentation has been developed in the form of on-line help and developer (JavaDoc) documentation
- The Web presence has been maintained with a major upgrade early in 2007
- Training has included the development of User Tutorials, a Developer Tutorial, and the delivery of training Workshops at the PAMGUARD Workshop and as part of Seiche training
- Regular releases of both Core and Beta versions have been made
- Existing Core functionality has been continually been improved.

PAMGUARD phase 4 has met almost all the objectives and exceeded them in many parts. All of WP2 Maintenance has been delivered as detailed in section 2. The Workshop, WP3, was delivered and is reported separately. Almost all of WP1 Core Architecture has been delivered.

The Future of PAMGUARD

PAMGUARD is a complex, evolving, and multidisciplinary software system and has the potential to be useful in a broad range of applications for many years to come.

To reach its full potential, and indeed to remain usable, PAMGUARD requires *guardianship*. In essence this entails both continual maintenance to track the evolution of components e.g. porting PAMGUARD to MS Vista, and integrating new technologies developed by the PAMGUARD community into the PAMGUARD core. Without effective guardianship it is likely that, although PAMGUARD will continue to be used, it will never reach its full audience or potential.

To ensure that the PAMGUARD community enjoys unbroken support, guardianship is required to continue in 2008 and beyond. The specific guardianship activities that are required are: bug fixes; integration of components developed by other groups into the PAMGUARD repository; maintenance of the PAMGUARD web site and repositories; annual and minor releases to enable standardisation.

Related Reports

This report relates to other PAMGUARD reports in the 2007/08 period as follows.

- A separate report has been provided on the SMRU PAMGUARD 4 activities.
- A separate report has been provided on the full CODA sea trial.
- A CD documenting the PAMGUARD Workshop (WP3) has been distributed.

This report incorporates much of the SMRU and CODA material, and summarises the WP3 work in Section 2.5.

Contents

| | |
|--|-----------|
| Executive Summary | 0 |
| The Future of PAMGUARD | 1 |
| Related Reports | 2 |
| 1. Introduction | 5 |
| 2. Maintenance activities | 6 |
| 2.1. Support | 6 |
| 2.2. Code maintenance | 6 |
| 2.3. Documentation | 7 |
| 2.4. Website | 8 |
| 2.5. Training | 9 |
| 2.5.1. Development of Tutorials | 9 |
| 2.5.2. Dissemination and Training | 10 |
| 2.6. Releases | 12 |
| 2.7. Improved functionality | 12 |
| 2.7.1. Click Detector | 12 |
| 2.7.2. Whistle Detector | 15 |
| 2.7.3. Map Display | 16 |
| 3. Core functionality work | 18 |
| 3.1. Adaptations to support offline re-analysis | 18 |
| 3.1.1. Viewer and Mixed mode operation | 18 |
| 3.1.2. Bulk file analysis | 18 |
| 3.2. Advanced Configuration utilities | 19 |
| 3.2.1. Save Settings options | 19 |
| 3.2.2. Hydrophone configuration, location and display | 19 |
| 3.2.3. New module creation and inter module dependencies | 19 |
| 3.3. Module management | 20 |
| 3.4. Advanced Displays | 21 |
| 3.4.1. Display Software Restructuring | 21 |
| 3.4.2. Vessel Display | 21 |
| 3.4.3. Airgun Display | 22 |
| 3.4.4. Pop-up hints | 24 |
| 3.4.5. Symbol keys on displays. | 24 |
| 3.4.6. Map Comments | 25 |
| 3.4.7. Depth Information | 26 |
| 3.5. Database interface | 27 |
| 3.6. Filter module | 27 |
| 3.7. Sound Acquisition | 28 |

| | |
|---|------------|
| 3.7.1. Sound Play back | 28 |
| 3.7.2. Amplifier | 29 |
| 3.7.3. Sound acquisition | 29 |
| 3.8. ASIO sound card selection | 32 |
| 3.9. Simulator | 34 |
| 3.10. NMEA Data Acquisition | 35 |
| 3.11. 3D Tracker | 36 |
| 3.12. Ishmael detectors | 40 |
| 3.13. Other features | 41 |
| | |
| 4. CODA/PAMGUARD Trial | 43 |
| 4.1. Trial plan | 43 |
| 4.2. Leg 1 | 43 |
| 4.3. Leg 2 | 51 |
| | |
| 5. Summary | 52 |
| | |
| Appendix A PAMGUARD User Tutorial (HWU/OSU) | 53 |
| | |
| Appendix B. Developer Tutorial (DG) | 99 |
| | |
| Appendix C Tables of Functionality | 109 |
| | |
| Appendix D Developer Guide Branched CVS Repository (HWU) | 113 |

1. Introduction

Phase 4 of the PAMGUARD project incorporates both maintenance and development aspects relating to the software. Much of this work has been carried out collaboratively between the partners involved. In addition, a conference and workshop was held at Heriot-Watt University in March 2007 on passive acoustics for monitoring marine mammals at sea. At the workshop, participants were introduced to PAMGUARD and provided with hands on experience with the software. PAMGUARD has also been subjected to one month of intensive de-bugging and user evaluation during the CODA trials in July 2007.

This document reports the activities and accomplishments in phase 4 of the PAMGUARD project. Section 2 describes work done under Work Package 1 (WP1) of the PAMGUARD 4 proposal. Section 3 describes work done under Work Package 2 (WP2) of the PAMGUARD 4 proposal. Section 4 summarises the CODA Trial work. The Workshop, Work Package 3 (WP3), was held at Heriot-Watt University in March 2007 and has previously been fully reported by Ecologic, and a summary is included in Section 2.5.

Partner's activities were coordinated by email, phone and conference calls. Monthly reports were collected and from all the partners and edited by Heriot Watt University for the JIP.

1.1.1. Related Reports

This report relates to other PAMGUARD reports in the 2007/08 period as follows.

- A separate report has been provided on the SMRU PAMGUARD 4 activities.
- A separate report has been provided on the full CODA sea trial.
- A CD documenting the PAMGUARD Workshop (WP3) has been distributed.

This report incorporates much of the SMRU and CODA material, and summarises the WP3 work in Section 2.5.

2. Maintenance activities

2.1.Support

Support for PAMGUARD users has been provided by telephone and email, using the info@pamguard.org and support@pamguard.org email addresses.

There have been over 300 downloads of the software this year, and almost 600 to date. Figure 1 shows the download statistics taken from the PAMGUARD SourceForge site (sourceforge.net/projects/pamguard/).

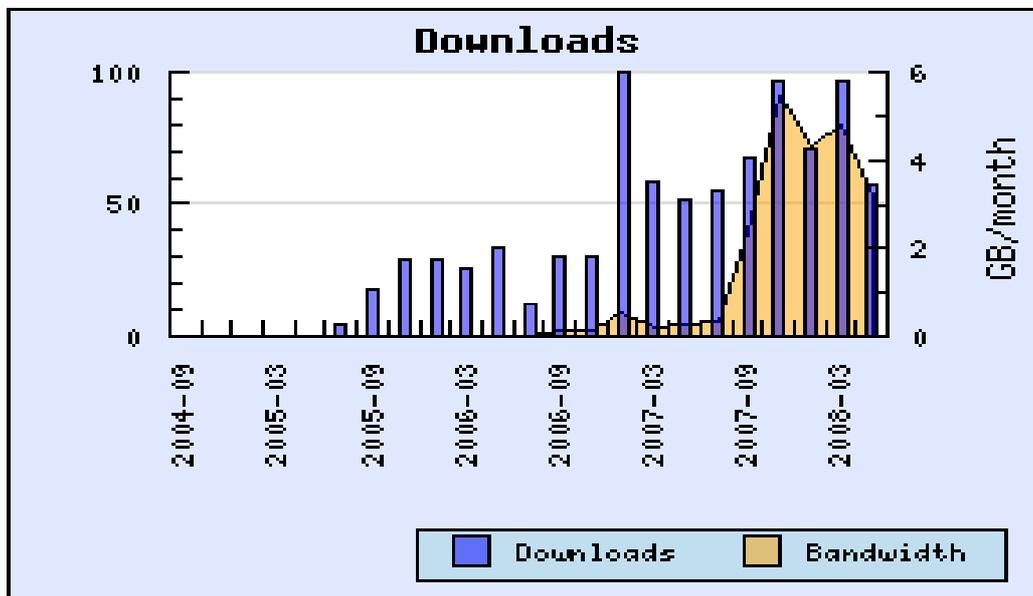


Figure 1 PAMGUARD download history

At the time of this report 30 individuals have contacted the PAMGUARD team for technical support. The majority of these requests have related to hardware/software setups as opposed to PAMGUARD specifically, and have been solved by giving advice over email or telephone conversations.

(HWU)

2.2.Code maintenance

Debugging is an on-going activity carried out by all partners. However, a considerable bugs list was generated during the CODA trial in July and these bugs are mainly the

concern of HWU/DG. Some of these bugs were fixed during the trial and some have subsequently been fixed.

Bugs have ranged from detectors crashing to small annoyances, such as settings not reloading correctly or incorrect distance being given in the GPS text area of the map. Most bugs were fixed in less time than it would have taken to write them down and there are no outstanding major bugs in the current PAMGUARD core. Section 4 details the code maintenance induced by the Coda trial in more detail.

(All)

Branching CVS Repository. The PamGuard CVS repository has been split into two branches, the “pambranch” and the “Head” branch. pambranch contains all the new functionalities and developing modules; Head contains the fully tested functionalities. A developer tutorial for how to access these two branches was written. See Appendix D.

(HWU)

Multiple Operating System Support: While PamGuard is primarily developed and tested on Microsoft Windows XP, it has been demonstrated to run satisfactorily on Microsoft Vista.

(HWU)

The most common alternative operating systems other than MS Windows XP or Vista would be versions of UNIX e.g. Leopard for Macintoshes or LINUX for PCs. As a Java application PAMGUARD is designed to execute on these platforms but must have appropriate drivers for the peripherals e.g. sound cards etc.

LINUX has always been seen as an integral part of the PAMGUARD development philosophy, indeed from the very conception of the project the use of Java was seen as the only way to ensure portability.

(All)

2.3.Documentation

As the code base grows, allied documentation is continually added using the JavaDoc tool from Sun Microsystems. This documentation may be viewed as comments within the source code, may appear as pop-up help from within some development environments, and a recent version is provided on the PAMGUARD web page.

The developer API documentation is also available on the PAMGUARD web site by selecting the DevZone menu, which was updated in May 2008.

Help documentation continues to grow in line with the expanding functionality of PAMGUARD. The release in May 2008 includes help files for the new modules. e.g. ASIO channel selection, 3D localization.

(HWU)

2.4. Website

A major overhaul of the PAMGUARD website (www.pamguard.org/) was carried out in February 2007.

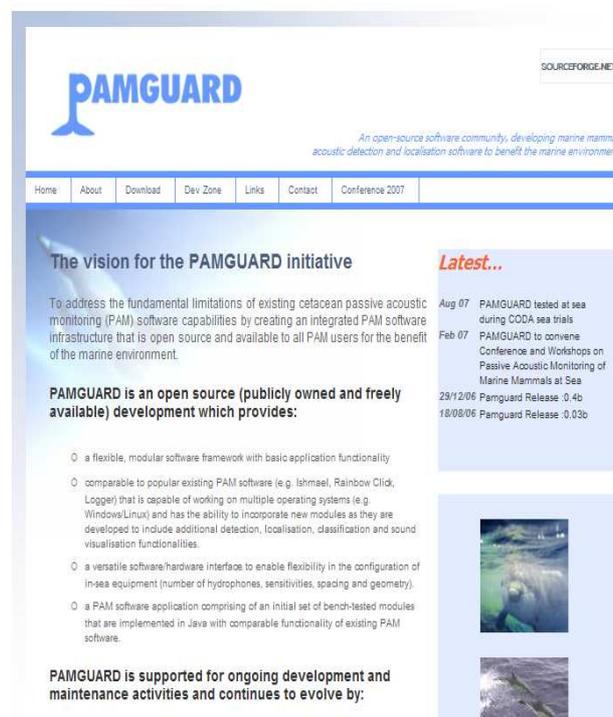


Figure 2 Revised PAMGUARD website

The site is now constructed using Dreamweaver. As well as updating the visual aspect of the site, the new site is much faster to load, and easier to maintain. The new site makes significant use of SSIs (server side includes) which in this case effectively means that regions within pages which are to be updated most often are sourced from individual files. This allows for editing of these regions with a minimal understanding of HTML. HWU kept publishing new activities of Pamguard project on the website.

(HWU)

Ecologic has also been working on some aspects of the web sites, for example providing more information on marine mammal acoustics and developing animated how-tos or

tutorials. These will be available on a private test site for comment and approval before being integrated with the existing web site.

(Ecologic)

2.5. Training

2.5.1. Development of Tutorials

User Tutorial

A training tutorial for PAMGUARD was written up and used for the PAMGUARD workshop sessions in March 07. The tutorial provides an introduction to PAMGUARD from a user's perspective, involving exercises in setting up and running the software with a variety of configurations of plug-in modules. 39 of 43 participants expressed that the tutorial was enjoyable in the workshop survey, and some constructive comments were made.

The tutorial is available from the PAMGUARD website and was also distributed on the CD from the workshop, along with the necessary sound files and PAMGUARD settings files. This tutorial is available on the PAMGUARD website and was also used during a training session at Seiche's Underwater Acoustics 2007 training sessions in Teddington, London in September

The tutorial has been updated to reflect these comments and recent changes and additions to PAMGUARD, which includes 3D localization and ASIO sound card selection. A copy of the most recent tutorial can be found in Appendix A.

(HWU)

Developer Tutorial

Pamguard is an open source project, and as such, developers are welcome to look at any part of the code, comment on it, modify it, and hopefully improve it! A developer tutorial has been produced with a focus on the development of new detector modules using the existing Pamguard infrastructure. A copy of the tutorial can be found in Appendix B.

(DG)

2.5.2. Dissemination and Training

Conference and Workshops

A conference and workshop event was held at Herriot Watt University on 28th and 29th of March 2007. The event had several objectives

- To introduce passive acoustic monitoring (PAM) and the PAMGUARD project to a broad audience of stakeholders, including marine mammal observers (MMOs), PAM providers, regulators, and developers.
- To provide initial training and an introduction to the PAMGUARD software for users – PAM operators, MMOs (and others) and developers – and receive, collate, and eventually incorporate their feedback.
- To provide an introduction and workshop session for potential PAMGUARD developers
- To initiate and guide discussion on the strengths and weaknesses of PAM with different species and situations.
- To discuss areas for development and contribution from other PAM software developers within the open source framework of PAMGUARD.

The first day was taken up with a series of lectures from the Pamguard team and other contributors. In all, 18 lectures or short talks were presented along with four posters.

- Why PAM? An introduction to marine mammal acoustics and passive acoustic monitoring for mitigation and research. **Jonathan Gordon, EcologicUK, UK.**
- Principles and approaches to automatic detection and classification: a non-technical overview. **David K. Mellinger, Oregon State University, USA.**
- Principles and approaches to localisation: a non-technical overview. **Aaron Thode, Scripps Institution of Oceanography, USA.**
- The PAMGUARD Concept. Why it is necessary, goals, organisation, structure, future plans
- **Mike Jenkerson, International Association of Oil and Gas Producers, E&P Sound and Marine Life.**
- Research perspective; Distribution and abundance surveys. **Oliver Boiseau, International Fund for Animal Welfare.**
- Seismic Operations perspective. **David Hedgeland, PGS.**
- Bioacoustic applications and analysis requirements for biological research. **Harold Figueroa, Bioacoustic Research Program, Cornell University, USA.**
- Passive Acoustic Monitoring Using DASARs to Assess Impacts of Offshore Oil Production on Bowhead Whales at BP's Northstar Facility, Alaskan Beaufort Sea. **Bill Streever, BP, USA.**
- Construction/other Energy related activities, (piling, decommissioning.) **Roy Wyatt, Seiche, UK.**
- Military perspective. **Ed Harland, Chickerell Bioacoustics, UK**
- Regulatory perspective. **Zoe Crutchfield, JNCC, UK?**
- MMO perspective. **Alison Gill, Marine Team**
- PAM equipment and service provider's perspective. **Scott Carr, JASCO, Canada**
- A non-technical introduction to the PAMGUARD software environment. **Douglas Gillespie, University of St. Andrews, UK.**
- Future capabilities and requirements for mitigation and monitoring using PAMGUARD. **David K. Mellinger, Oregon State University, USA and Aaron Thode, Scripps Institution of Oceanography, USA.**
- Wider uses for PAMGUARD and the PAMGUARD programming environment. **Aaron Thode, Scripps Institution of Oceanography, USA.**

- Strategies for long term sustainability of PAMGUARD. **Phil Trinder, Heriot-Watt University.**

The second day consisted of six workshops on various aspects of PAM.

- Testing, validating and quantifying the performance of Pamguard. How can the software best be thoroughly tested in real world conditions so that relevant aspects of performance, such as detection probability and localisation accuracy, be quantified.
- Wider applications for PAM within the PAMGUARD framework. Discussion of other uses, research, survey, non-marine mammal acoustics.
- Practical experience of using PAM, problems and opportunities. This provided an opportunity to provide guidance and feedback to the development team.
- Training and support. Discussion of strategies and mechanisms for providing training in use of PAMGUARD and support to users.
- Regulation. A discussion of realistic capabilities and constraints of PAM as part of regulation and mitigation.
- Development of a strategy for sustainability and long term support for the core software.

Each session was run by one or more conveners and structured to allow broad and open discussion amongst all the participants. These sessions ran concurrently with two hands-on tutorial sessions in the Department's Computer Classroom. The first of these was a general supervised tutorial session designed as an introduction for new users which was run twice to allow all participants to complete it. The other workshop consisted of a tutorial and discussion session for those who might be interested in developing new modules in PAMGUARD in the future.

The workshop finished with a meeting to discuss ASA standards on acoustic monitoring for marine mammals convened by Aaron Thode.

Attendance was good with around 90 people attending. Most were from the UK but participants also came from USA, Canada, France, Italy, Norway, Greenland, Mexico, Denmark and Australia, and represented a good mix of the different interest groups and stake holders. We feel that through this mix of presentations and workshops we achieved the event's rather wide objectives. Feedback received both directly and via questionnaires was very positive.

Workshop planning was greatly helped by a steering committee which convened several times by phone.

A report of the workshop was prepared as an HTML based CD which included all of the presentations as both pdf documents and flash animations as well as reports, from each of the conveners, for the workshops held on the second day. In addition, all of the tutorial materials, including example sound and map files and software were included on the CD. CDs were distributed to all of the participants and to other stake holders and user groups. The report will likely be published online at a later date.

(Ecologic,All)

Seiche's Underwater Acoustics September 2007

A PamGuard training session was given at Seiche's Underwater Acoustics 2007 training sessions in Teddington, London in September. A PamGuard demo was prepared for the tutorial. CDs contains user tutorial, PamGuard demo, sound files, and map files were prepared. A Pamguard lecture was given at the and tutorial was set up and run at Seiche course.

(HWU)

2.6.Releases

Several major releases have been made in the course of this project e.g. in August 2007, January 2008, and in May 2008. Since January 2008 both core and beta releases have been made concurrently.

Several code enhancements were required for these releases. Some of the new modules and functionalities in PAMGUARD require Dynamic Link Libraries (DLLs) to be loaded before they can operate. These libraries (ASIO, serial port and system timing DLLs) typically serve as interfaces to "native" operations. These libraries must be visible in the java library path which is specific to individual machines. Since PAMGUARD is downloaded by the user as an executable Jar file, the path to the jar contents is not included in the system path. Methods were therefore required to update the java library path and extract the DLLs to that location.

(HWU)

2.7.Improved functionality

The following section outline improvements to existing core functionality.

2.7.1. Click Detector

As one of the key components, and one which has now been extensively used at SMRU for analysis of survey data, a number of improvements have been made to the click detector module.

Better channel grouping in the click detector for multi channel analysis and better displays for multi pair channel configurations have been added. To further assist set-up by the user, trigger information can be displayed as a plug-in in the bottom of the spectrogram window.

Click detector performance has been improved by speeding up both the writing of clicks to file and the click detector graphics.

Improved time delay in correlation functions used to measure angles

A new algorithm has been implemented which uses parabolic interpolation between the integer sample numbers to improve the accuracy of delay (and therefore bearing) calculations in the click detector. . One practical implication of this is that useful bearing accuracy can now be obtained from hydrophone pairs that are an order of magnitude closer together.

Improved manual tracking

A pop-up menu has been added so that when an operator clicks on a detected click, tracked clicks can be labelled / coloured (Figure 3).

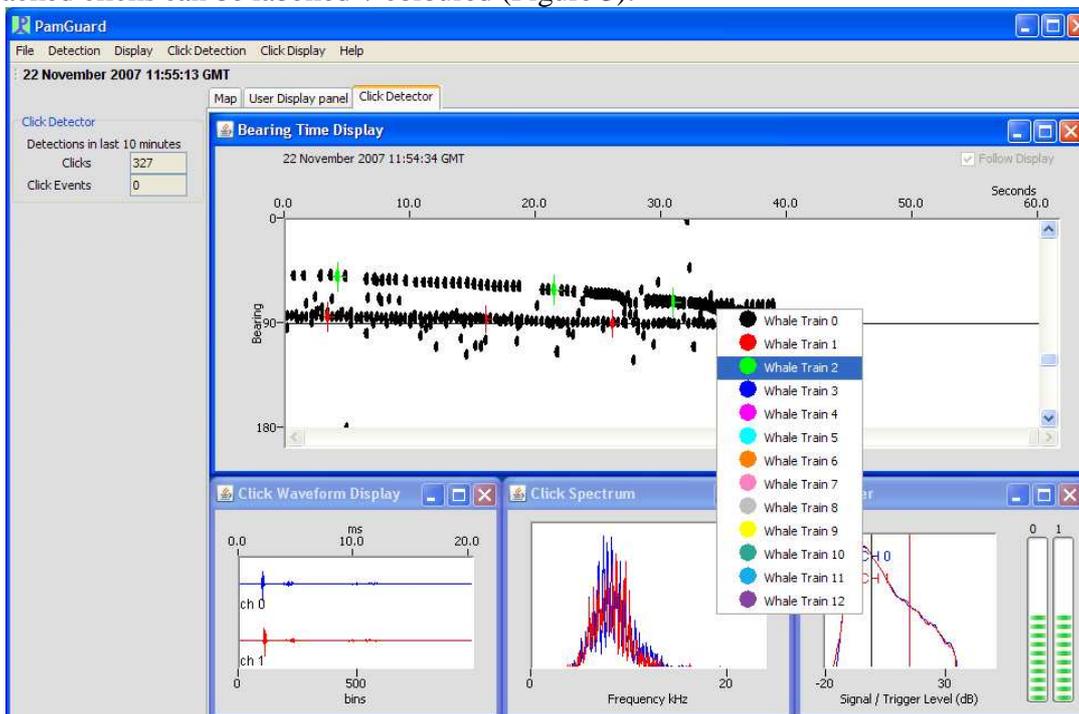


Figure 3. Manual tracking of multiple sperm whales. When the right mouse button is pressed, the display freezes, if the mouse is released over a click, a menu pop-s up and a train can be assigned a colour / whale number. The example shows tracks of two sperm whales.

Clicks which have been labelled in this way are added to grouped detections and a localisation calculated using target motion analysis (crossing bearings from multiple points along the ships track). This can work either with a single hydrophone pair or with pairs of hydrophones some distance apart. In the latter case, it is possible to get a location from a single click since bearings from two widely separated points can be crossed accurately. If only a single hydrophone pair is used, then locations can only be accurately calculated once the vessel has progressed some distance along the track. These locations are only accurate if the animal movement is small compared to that of the vessel.

It has proven to be a reliable method for estimating positions of sperm whales, but is not useful with fast moving animals close to the vessel such as dolphins. The calculation computes a position along the vessel track line and the perpendicular distance from the

track line. Errors are also calculated on both coordinates. Positions and errors are shown on the map (Figure 4).

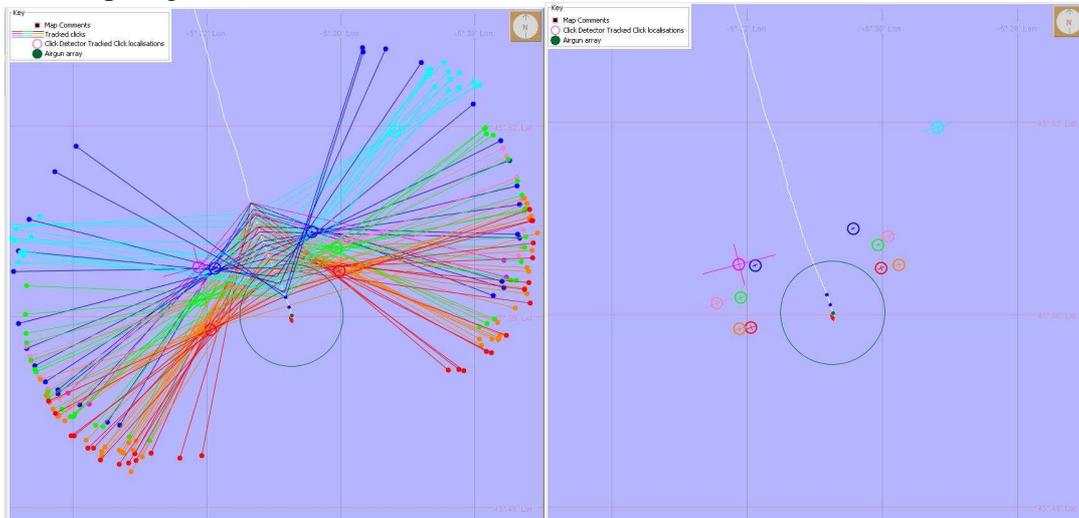


Figure 4 Plots of locations of multiple sperm whales. The left hand plot shows bearing lines to manually selected clicks. This right plot shows locations and errors on locations. The circle is a 1km circle around the airgun positions (see section 3.4.3).

Improved click classification

Improvements to the IFAW RainbowClick click classifier, which were developed for the detection of beaked whale clicks in 2007 have now been fully implemented in PAMGUARD (Figure 5). The click classifier contains functions for inserting default parameters for some species (small pop-up menu in lower right hand corner of Figure 5) as well using a new set of parameters. This list can easily be added to by developers.

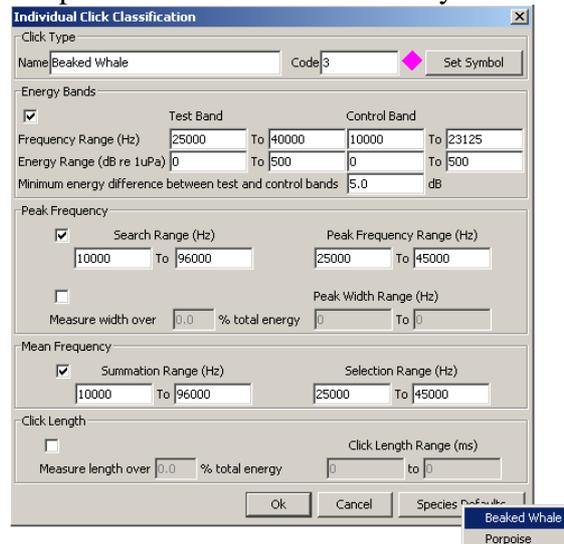


Figure 5. Click classification options dialog

The click detector displays have also been improved giving options to only display clicks classified to a particular species if required. Although this classifier has proven useful and is still being used at SMRU for the detection of beaked whales and porpoise, it may be superseded later in 2008 by work being undertaken under the JIP funded Odontocete Classification project.

2.7.2. Whistle Detector

Another core detection component is the whistle detector which has also undergone considerable bug fixing and improvement during the past year. Additional options have been added to the peak detection stage of the whistle detector to allow for more flexible configurations. In particular, options to limit the frequency band for whistle searching have been added and two alternative peak detection methods are now available.

Improved multi channel performance and configuration

The whistle detector has been altered so that multiple, independent detectors will run on pairs of hydrophone channels, each pair detecting whistles and calculating bearings independently. Figure 6 shows the new data source configuration dialog.



Figure 6. Whistle data source configuration. In the example, two pairs of hydrophones are used. Whistles will be detected independently on each pair, and a location, based on crossed bearings, will be calculated if whistles appear simultaneously on both hydrophone pairs.

If detections on multiple pairs of hydrophones are detected simultaneously and overlap in both time and frequency, then the detection is assumed to be of the same whistle and the crossed bearing localisation is computed, stored in the database and displayed on the map.

Further improvements in the whistle detector are expected to arise from the JIP funded species discrimination project.

2.7.3. Map Display

The third key PAMGUARD component which has undergone considerable revision in 2007 is the map.

As well as the map comments, keys and detection hints described above, the map graphics have been sped up. The map used to redraw every second and take approximately half a second to redraw each time, thereby rendering PAMGUARD almost inoperable. Sensible buffering of underlying map images and improved code now means that most of the map does not redraw at all (unless zoom settings are adjusted) and when it does completely redraw, it only takes about 1 millisecond to do so.

Coastline images can currently be loaded as ASCII files from the Gebco digital atlas. Names of Gebco files are now correctly held in the PAMGUARD configuration options between runs (Figure 7).

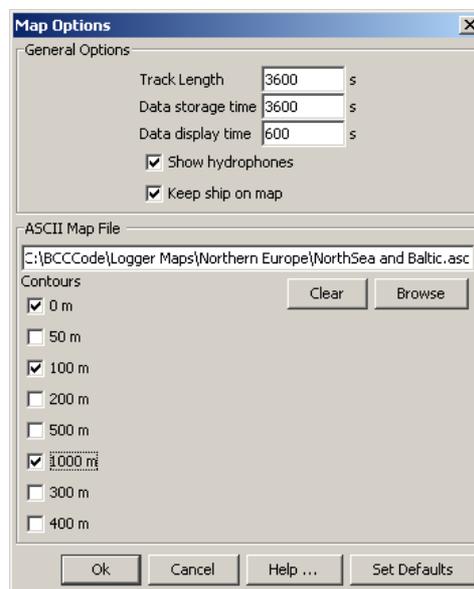


Figure 7 Map options dialog

The GPS text area in the bottom right of the display has been debugged so that it now shows the correct distance from the vessel to the cursor (this was not the case in earlier releases).

The pan/zoom buttons and the GPS text area have also been relocated slightly and there are options to remove them from the map altogether in order to increase the overall viewable map area (Figure 8).

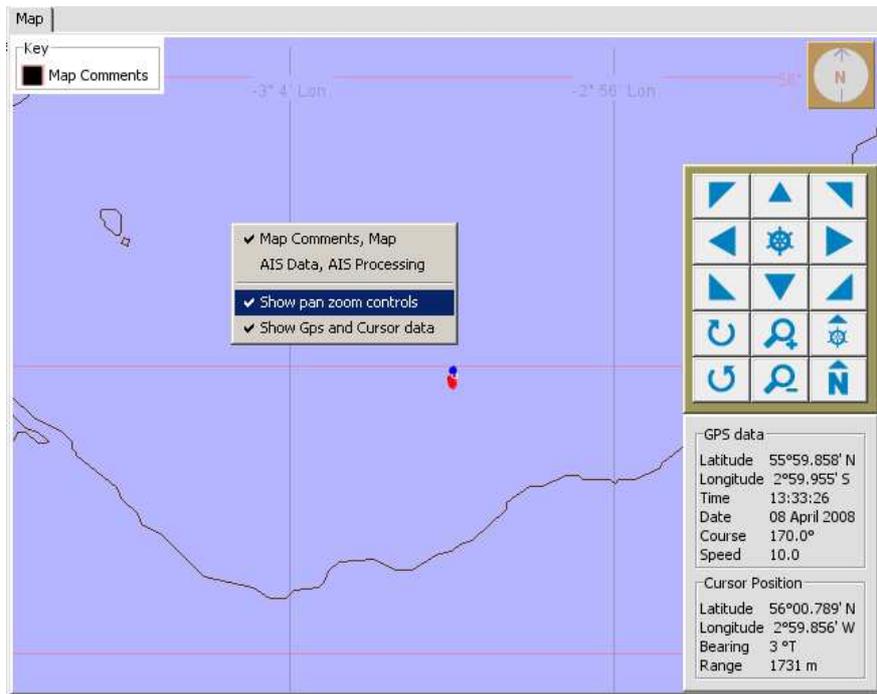


Figure 8. New position of Pan zoom and GPS text area and menu commands to remove them from the map if desired

(DG)

3. Core functionality work

3.1. Adaptations to support offline re-analysis

3.1.1. Viewer and Mixed mode operation

PAMGUARD now has three main modes of operation, Normal mode, Viewer mode and Mixed mode. During normal operation, PAMGUARD collects data from the outside world (through sound cards, GPS's etc.) processes those data and stores summary information in a database.

In Viewer mode, PAMGUARD reads data back from the database and re-displays it on the map. This enables operators to review data following a cruise, prepare maps for reports, etc. In viewer mode, the 'start' and 'stop' menu commands have been replaced with a dialog where the user sets the times for which they wish to display data.

In mixed mode, data are travelling in two directions at once – both into the database and out of it. GPS data and other data which does not derive directly from sound (e.g. depth data) are read back from the database. Sound data are reprocessed, generally from file. The output of detectors analysing that data is written back into the database. A main motivator behind mixed mode is the re-analysis of CODA / PAMGUARD field trial data during which the operator is re-analysing data as though in the field. During mixed mode, from the operators perspective, PAMGUARD operation looks and feels (apart from the lack of motion sickness) exactly as it does during real time operation. Mixed mode will also be useful in any other re-processing of data where detections and localisations are being reprocessed and need to be correctly linked with GPS and other non-sound data.

3.1.2. Bulk file analysis

File folder analysis functionality for bulk processing of audio files. The user can now select a single file for analysis, a complete folder of many files (including those in sub-folders if desired) or a selection of any number of files from within a single folder. There are options to either analyse all sound as though it were one long recording not stopping between files, or to stop and restart detectors at the end of each file.

For bulk file analysis to work well, PAMGUARD must be able to detect the time at which a sound file begins – its timestamp. PAMGUARD can now read a variety of date/time formats encoded in file names and set the corresponding time for the start of the file.

3.2. Advanced Configuration utilities

3.2.1. Save Settings options

PAMGUARD now has the facility to save configuration settings in settings files. When PAMGUARD starts, the user is presented with a drop down list of recently loaded configurations (Figure 9), or they can browse their computer for settings files. Settings files can thus be developed for different applications and can be readily shared between users.

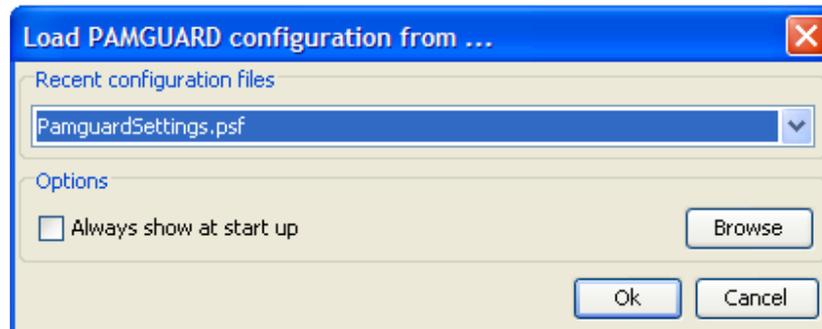


Figure 9 : Save settings options dialog presented to the user at start-up

3.2.2. Hydrophone configuration, location and display

Hydrophone configuration options have been modified to cater for static (e.g. bottom mounted) hydrophones. If a static array is selected, hydrophone positions are referenced to a latitude and longitude entered by the user.

Framework code has been added to locate hydrophones streamed behind a vessel using either predicted positions, based on vessels recent track, rate of turn, etc, or using data from sensors or using a combination of predicted and measured location. Java classes have been written for a simple locator which assumes the hydrophone is sticking straight out the back of the boat and a “threading” locator, which assumes that the hydrophone is following the exact track of the vessel as well as for static hydrophones. The type of locator can easily be selected by the user from a drop down menu. Developers can use the framework to easily implement alternative locators if required in the future.

3.2.3. New module creation and inter module dependencies

A key feature of PAMGUARD is the way in which the user can add and configure modules for different signal processing, detection and localisation tasks depending on the type of work being undertaken.

The PAMGUARD user creates new modules from the PAMGUARD file menu. Previously, the user always had to type in a name for each new module. Names are now assigned automatically, although the user still has to confirm. Module names are now also required to be unique, for instance if you have two instances of the click detector, it is not possible to name them both ‘click detector’, you’d have to use ‘click detector’ and ‘click detector 2’, or better, ‘sperm whale detector’ and ‘beaked whale detector’, etc. The new dialog for entering module names is shown in Figure 10.

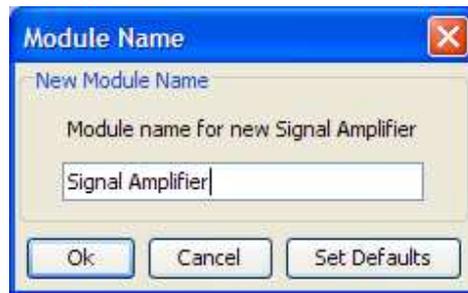


Figure 10. New module name dialog.

Many PAMGUARD modules require other PAMGUARD modules to be present before they can function correctly. For instance, the click detector requires some kind of sound input, the whistle detector requires a FFT (spectrogram) Engine, which in turn requires sound input, etc. When a module is added by the user, PAMGUARD checks that any dependencies are satisfied, and where necessary prompts the operator to create additional modules. Figure 11 shows a typical information panel which pops up if dependencies are not satisfied.

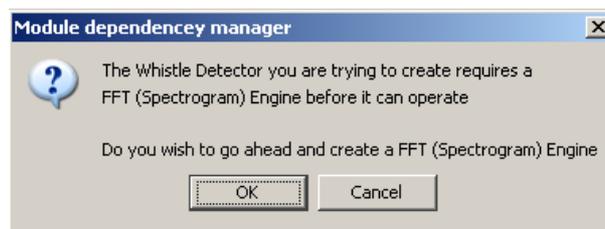


Figure 11. Message from the dependency manager indicating that additional modules need to be created.

(DG)

3.3. Module management

Users may wish to add a module without having the full knowledge of the data which that module requires to operate. To assist the user in this case, a module dependency “wizard” has been implemented. When a new module is created, the module runs checks to verify that the required data source(s) exist(s). If this is not the case, dialogues are presented to the user to add the appropriate module(s). For example, if a user wishes to display a spectrogram on a user display panel, and there is no acquisition source set up, a dialogue will first appear stating that there is no FFT data source and asking the user whether one should be created. The user selects OK and is then presented with another dialogue box stating that the FFT module requires a sound acquisition module, again asking if one should be created.

Previously in PAMGUARD, the user always had to type in a name for each new module. Names are now assigned automatically, although the user still has to confirm. Module names are now also required to be unique, for instance if you have two instances of the

click detector, it is not possible to name them both 'click detector'. Instead, the user must use unique names such as 'click detector' and 'click detector 2', or better, 'sperm whale detector' and 'beaked whale detector', etc.

(DG)

3.4. Advanced Displays

3.4.1. Display Software Restructuring

The PAMGUARD software has been restructured to be more generic. This facilitates the development of new core functionality and of plug-ins.

(HWU)

3.4.2. Vessel Display

A vessel dimensions dialog allows the user to enter the length and breadth of the vessel as well as the relative position of the GPS receiver on the vessel so that the vessel is correctly drawn on the map relative to the GPS receiver antenna (Figure 12). The 'predict ships position' options can be used to show the likely position of the vessel in the near future (based on current speed and heading info.)

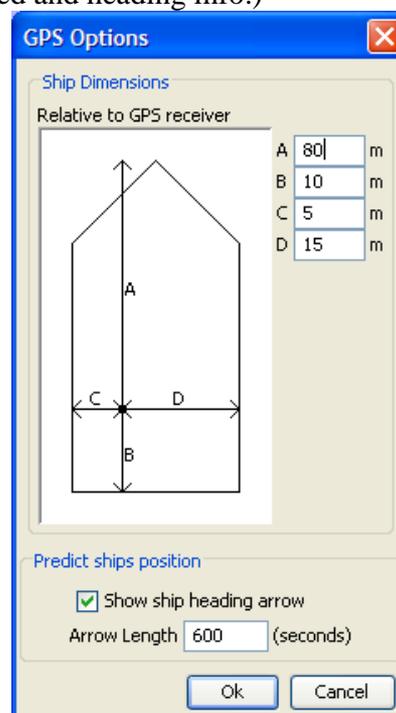


Figure 12: Vessel display options

(DG)

3.4.3. Airgun Display

Airgun display options (Figure 13) can be used to display the exact location of the airgun array. The module supports airguns deployed from the same vessel that PAMGUARD is running on as well as airguns deployed from different vessels (for example, if monitoring were taking place from a guard vessel). If guns are deployed from a different vessel, then AIS (Automatic Identification System) data are used to obtain the location and heading of the source vessel and these are mapped in PAMGUARD.

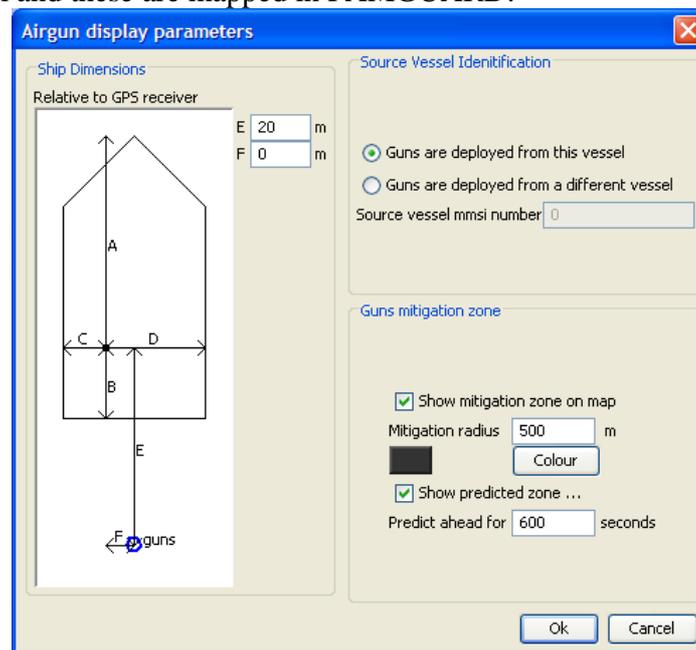


Figure 13 Airgun Display Options.

The airgun display shows the position of the guns, an optionally, both the mitigation zone (drawn as a circle centred on the guns) and a predicted swath of “mitigation area” ahead of the vessel which will be ensounded by the guns in the near future (set to 600s in the example in Figure 14).

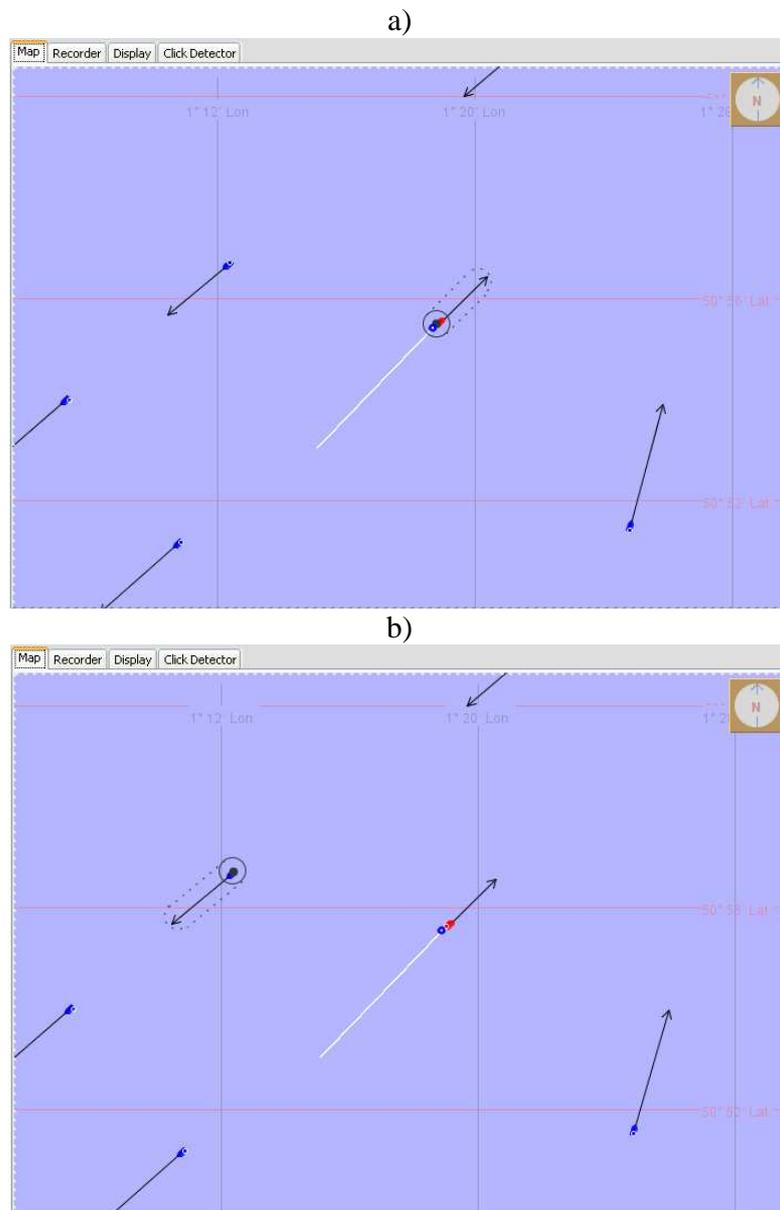


Figure 14 Vessel and airgun display for situation in which monitoring is occurring on the source vessel (a) and on an independent monitoring vessel (b). In which case location and heading for the source vessel would be provided by AIS. The blue circle just astern of the monitoring vessel is the position of the monitoring array hydrophones and the black blob is the airgun array, the black circle is the mitigation zone and the area enclosed by dotted lines is the predicted 10 minute ensounded swath.

The ship and airgun displays on the map are shown in figures 14a and 14b for airguns displayed on the monitor vessel and on a different vessel. Figure 14 uses archived AIS data (collected in the English Channel by Richard McLanaghan of IFAW) and simulated GPS data. International Maritime Organisation (IMO) regulations require AIS to be fitted aboard all ships of 300 gross tons or more, engaged on international voyages, cargo ships of 500 gross tonnage and upwards not engaged on international voyages and all passenger

ships irrespective of size. The requirement became effective for all ships by 31 December 2004. Presumably, most seismic source vessels would therefore transmit AIS. AIS receivers can be purchased for a few hundred dollars.

(DG)

3.4.4. Pop-up hints

Hints providing information on detection and other objects displayed on the map and other displays (Figure 15) have been added. This is an addition to the core architecture of the graphic overlay functionalities within PAMGUARD and hence can now easily be utilised by all future detectors.

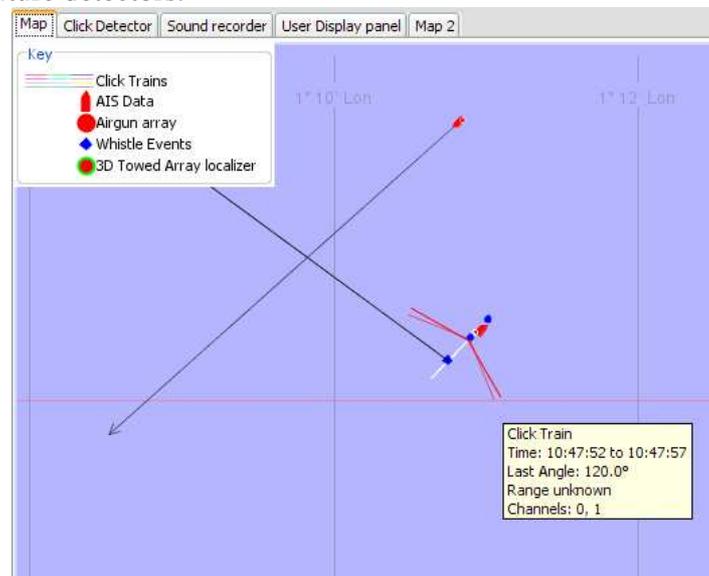


Figure 15 Example of a key in the corner of the map display and a hint which popped up when the mouse was hovered over a click train detection.

(DG)

3.4.5. Symbol keys on displays.

Keys have been added to maps and other displays (Figure 15). This required infrastructure changes so that each graphics overlay can provide correct information for each display. Users can also now select different symbols for each graphics overlay using standard dialogs. This has become necessary due to the large number of detection and graphics options now available within PAMGUARD.

(DG)

3.4.6. Map Comments

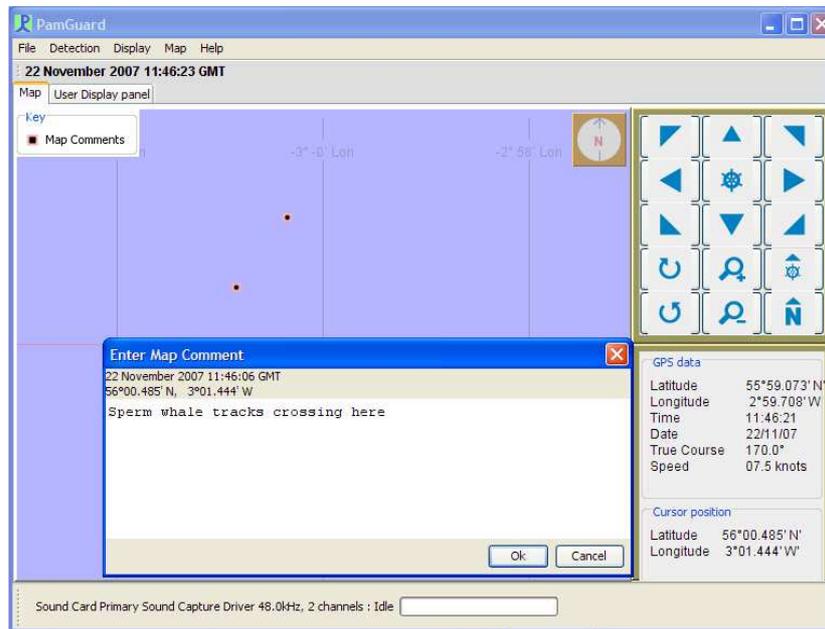


Figure 16 Dialog for entry of a map comment.

The user can double click anywhere on the map and a dialog pops up (Figure 16) with the time and lat / long of the cursor's location. You may enter a comment or cancel. The date, time, lat, long and comment text are written to the database. Locations of entered comments are displayed on the map and hovering the mouse will display the date, time, lat, long and comment information (Figure 17). This feature is useful for registering the results of manual tracking using crossing bearings from the click detector.

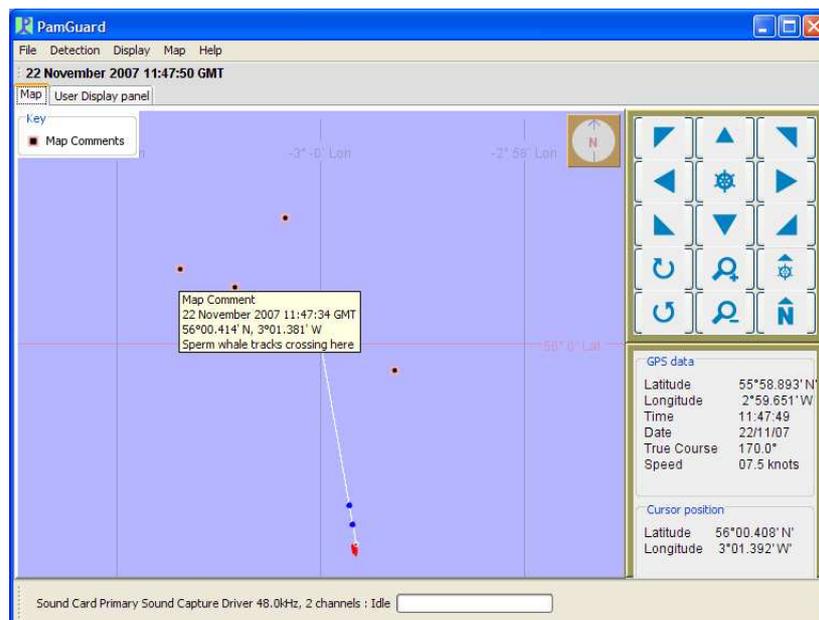


Figure 17 The map comment showing as a new symbol on the map, complete with pop-up 'hover' information

(DG)

3.4.7. Depth Information

PAMGUARD now performs 3D tracking, as outlined in Section 3.11, and the depth information is displayed on the map, as illustrated in Figure 18 below, and in the Figures in Section 3.11.

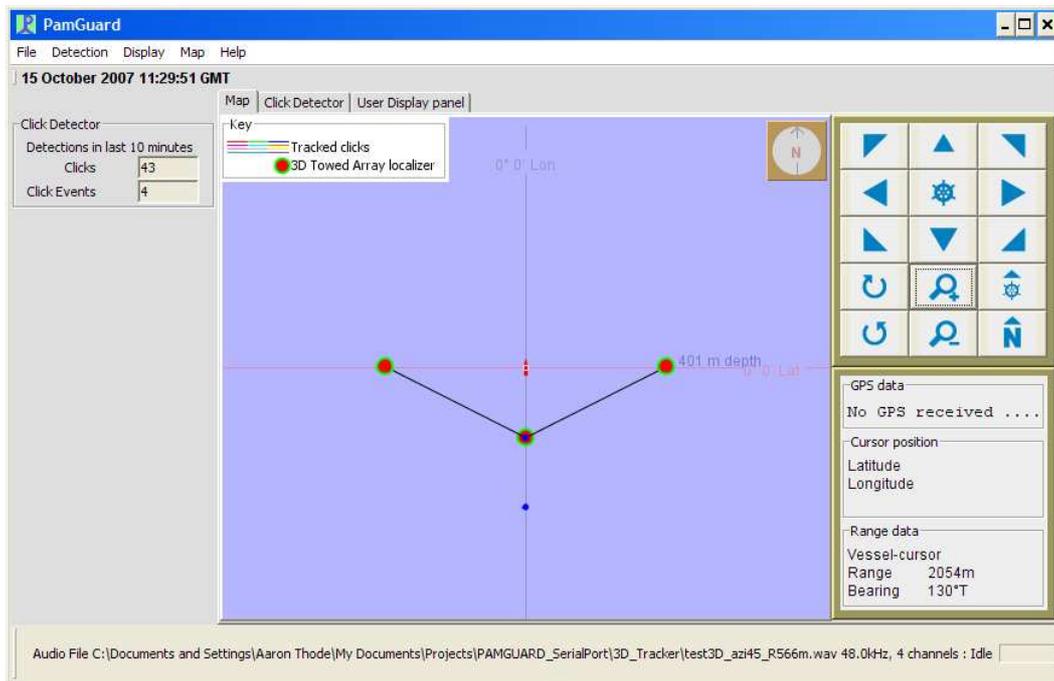
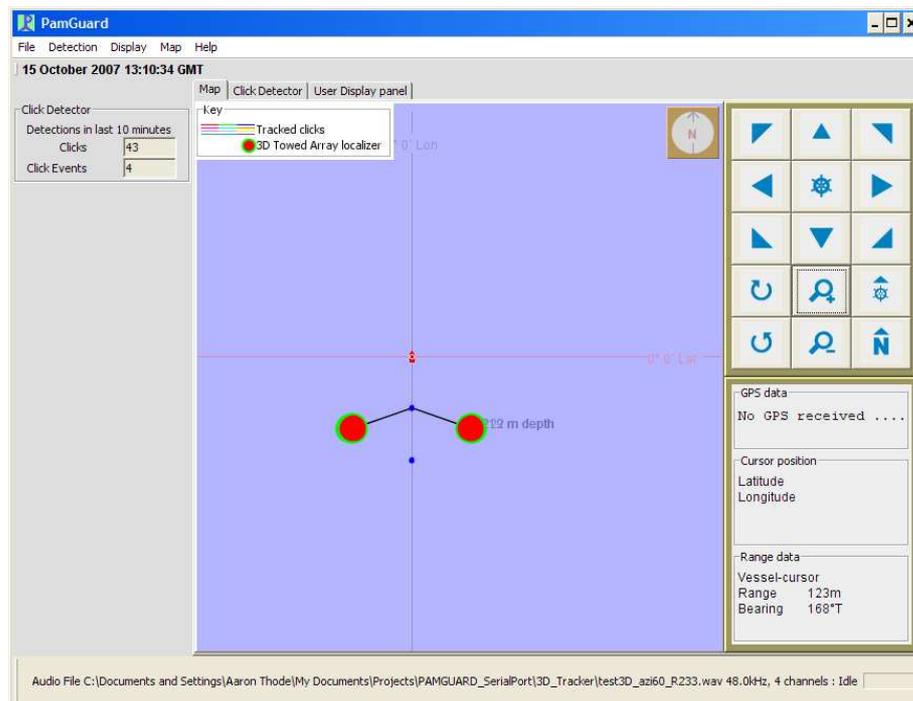


Figure 18 : Depth Information Display

(Scripps)



3.5.Database interface

A new database interface, which is easily extendable to any SQL based database system has been written. This currently supports MySQL and MS Access databases (tested with Access 2000 and Access 2003, but not Access 2007). The architecture is such that support for other database formats can easily be added if required. The database module will automatically configure a database so that it has the correct tables, with the correct columns for whichever PAMGUARD modules are in use at any particular time. Database tables have also been added which record which modules within PAMGUARD are installed and how they are configured in the database every time PAMGUARD starts. The database therefore forms a single document, containing not only detection information, but also the PAMGUARD setup. This has two uses 1) auditing PAM operations and 2) aids data analysis since the configuration 'travels' with the detection data in a single document. Code has been written to read these data back into PAMGUARD when running in 'viewer' mode, so that the PAMGUARD configuration matches the data in the database tables.

(DG)

3.6.Filter module

A stand alone filter which can be inserted into PAMGUARD data pathways. The user can set up filters with a Chebyshev or Butterworth response for high pass, low pass or band pass filtering (Figure 18).

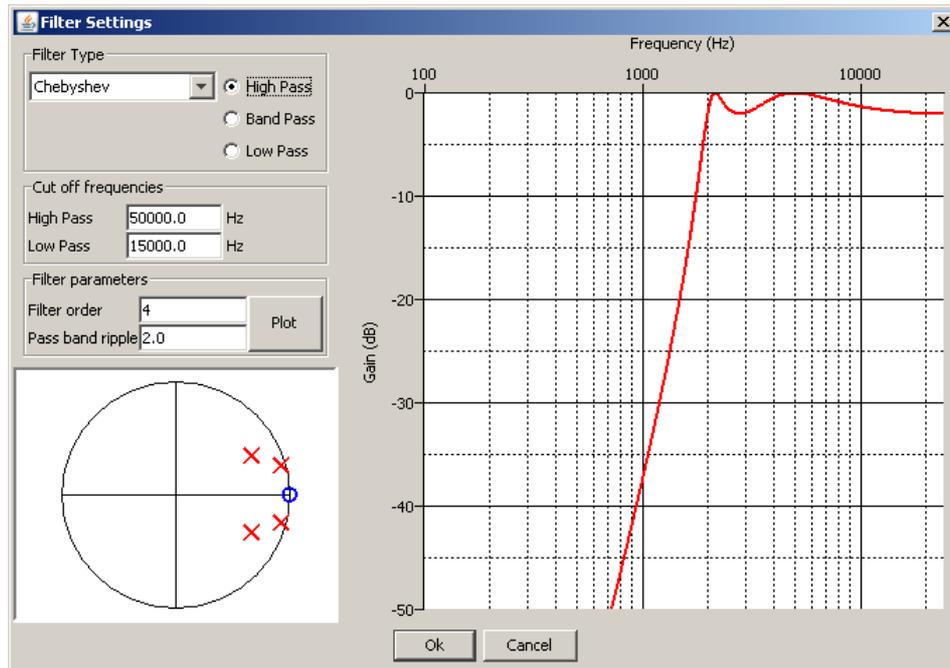


Figure 18 Filter configuration dialog

3.7. Sound Acquisition

3.7.1. Sound Play back

When re-analysing wav files, sound can be played back through the system sound card (Figure 19). Check boxes enable user to select which channels to listen to in multi channel recordings. When combined with a patch panel (see below) it is possible for the user to listen to multiple channels in each ear.

Future plans are to extend sound playback to work when analysing sound from sound cards and other devices in real time.



Figure 19 Sound play back control dialog.

3.7.2. Amplifier

A signal amplifier has been written (Figure 20). In addition to allowing different levels of gain to be applied to each channel it provides the ability to invert the signal on individual channels since we have in the past found that hydrophones are sometimes wired incorrectly so that the phase of signals is inverted on some channels, thus making it difficult to accurately measure time delays between signal arrivals on different channels using cross correlations.

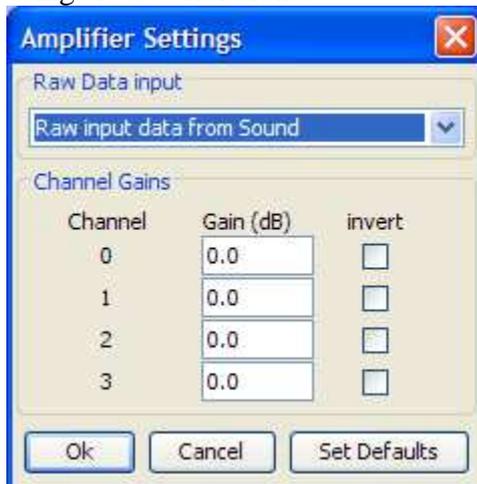


Figure 20 Amplifier control dialog.

3.7.3. Sound acquisition

ASIO

The ASIO protocol, specified by Steinberg, provides an interface with no specified limit on the number of channels. There are now many ASIO compliant devices available which offer more than 8 audio band, i.e. between 10Hz and 20KHz, input channels. The channels are commonly oversampled at rates up to 192kS/s.

Previously, PAMGUARD was unable to acquire acoustic signals on more than two channels, due to the restrictions inherent in standard Windows sound card devices and drivers. A new DLL (dynamic link library) has been created which now allows PAMGUARD to interface with sound acquisition devices which are supplied with ASIO (Audio Stream Input/Output) compliant drivers. Figure 21 shows the Audio Data Acquisition dialogue. ASIO sound acquisition has been tested using the Fireface and MOTU audio cards.

Other

A prototype interface to a high data rate National Instruments card has been developed. This card is capable of detecting high frequency vocalisations, e.g. from porpoises. To acquire at such high data rates currently requires a dedicated PC to interface with the card, with a UDP link to the PC running PAMGUARD. In the absence of a standard interface like ASIO, each individual high frequency card requires a specialised software interface.

While PAMGUARD potentially supports any ASIO card, the following table summarises the cards it has been used with.

| Card | Type of Interface | Data Rate | Status |
|--|-------------------|-----------|-----------|
| RME Fireface 800 | ASIO | 96KHz | Core |
| Motu 96KHz | ASIO | 96KHz | Core |
| National Instruments NI PCI-6123 Multi- function DAQ & NI PCI-6122 Multi- function DAQ | Bespoke UDP Link | 500KHz | Prototype |

Table of Validated Sound Cards

(HWU)

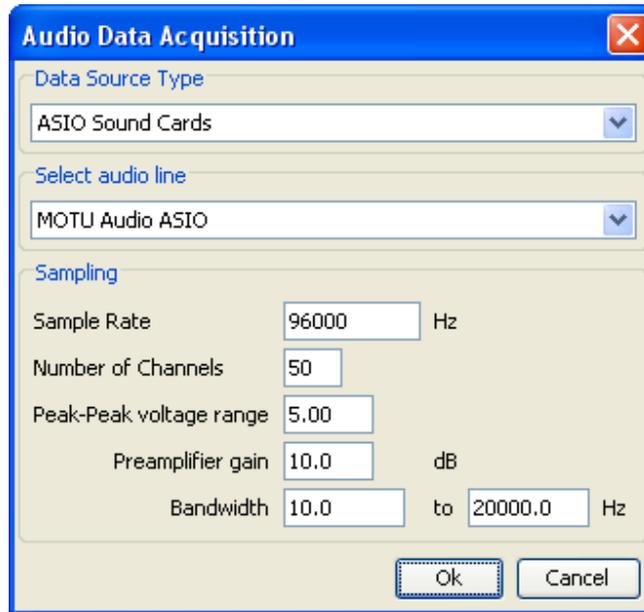


Figure 21 Selecting ASIO devices from the Audio Data Acquisition Dialogue

Sound can now be played back through the system sound card when re-analysing wav files. Check boxes enable to user to choose which channels to listen to in multi channel recordings. When combined with a patch panel (Figure 22) it is possible for the user to listen to multiple channels in each ear. The patch panel allows the user to mix different channels together.

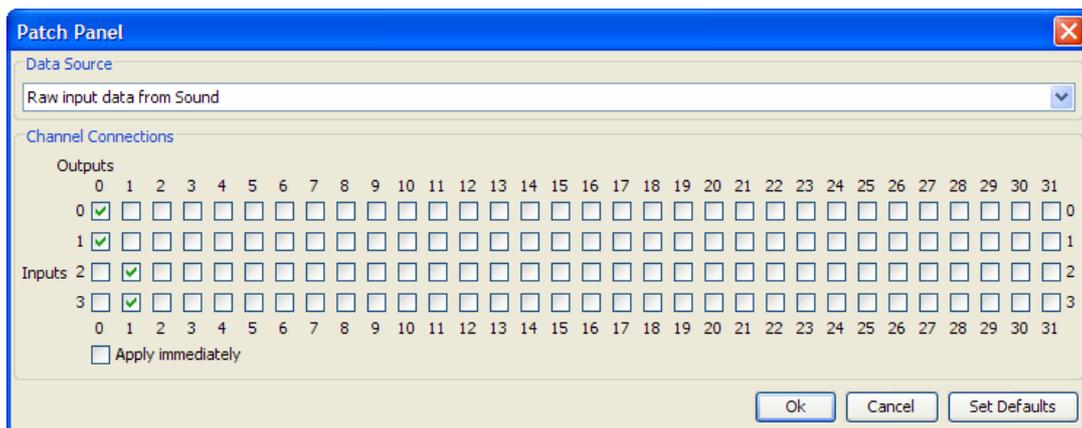


Figure 22 Patch Panel control dialog

Future plans are to extend sound playback to work when analysing sound from sound cards and other devices in real time.

File folder analysis functionality has been added for bulk processing of audio files. This module currently only understands file date formats used by PAMGUARD and IFAW Logger software. Further work is planned in collaboration with Dave Mellinger to improve this later in the year.

A simple signal amplifier has been written. One of its most useful functions is the ability to invert the signal on individual channels since we have in the past found that hydrophones are sometimes inverting the signal incorrectly on some channels, thus making it difficult to accurately measure time delays between signal arrivals on different channels.

Previously, when analysing data from files, the user could select either a single file, or an entire folder of files (including the contents of sub-folders). The user can now also select a selection of files for analysis from within one folder.

(DG)

3.8.ASIO sound card selection

PAMGUARD can now acquire multiple audio channels, up to a maximum of 32. We have implemented multiple channel selection for ASIO devices. Previously, when you tell the program to read a certain number of channels, it would only read channels 0, 1, 2 ... n-1. Now it can be sent a list of any channel numbers to read out. Figure 23 shows that 4 channels (0,1,2,4) have been selected. Figure 24 shows to create 4 panels for these selected 4 channels. Figure 25 shows the user display of these 4 channels. Figure 26 shows the result of 8 channels. The panels are used in the user display module, each panel shows the sound signals of each channel.

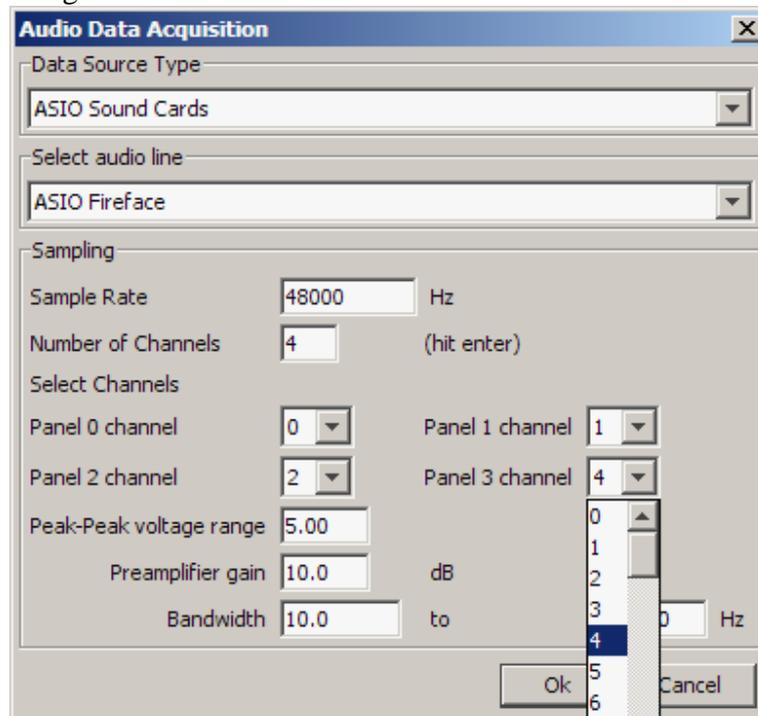


Figure 23 Channel Selection.

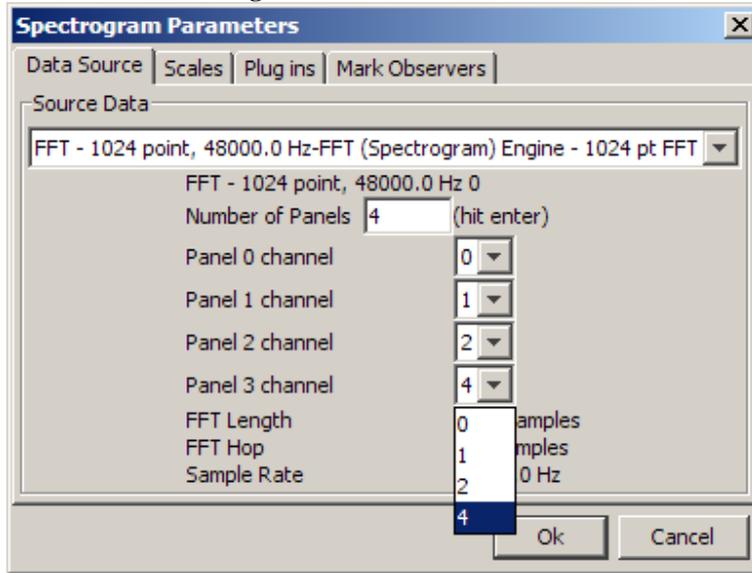


Figure 24 Panel Selection.

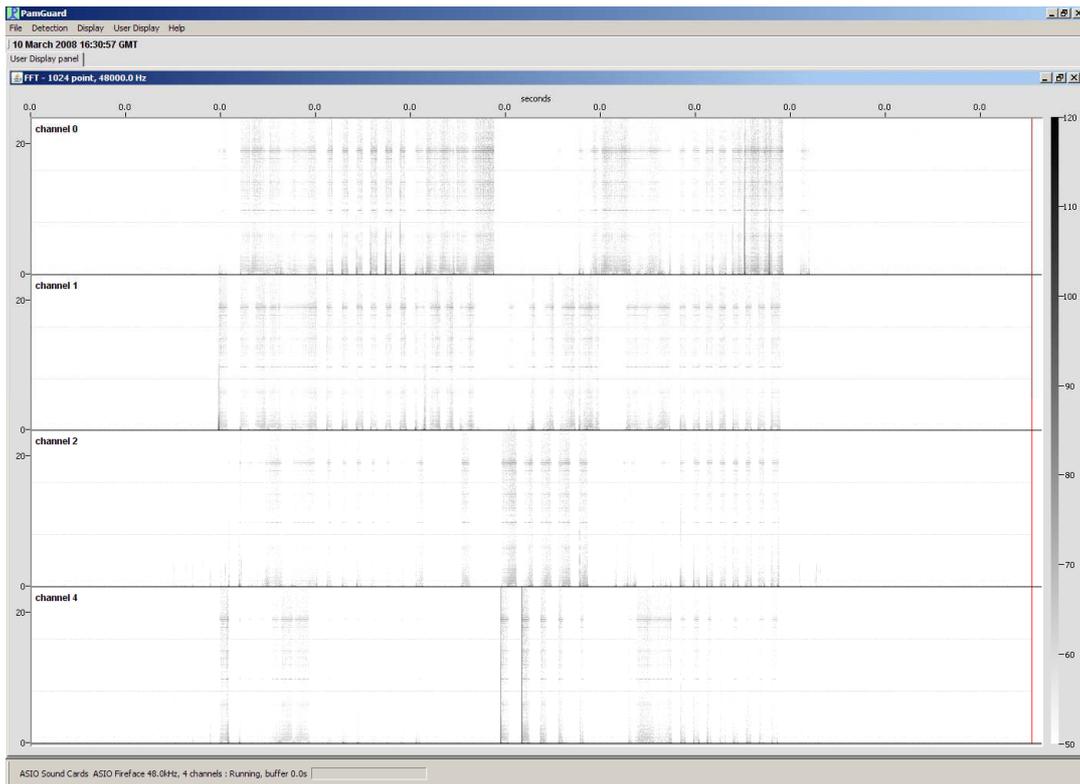


Figure 25 User Display of 4 Selected Channels

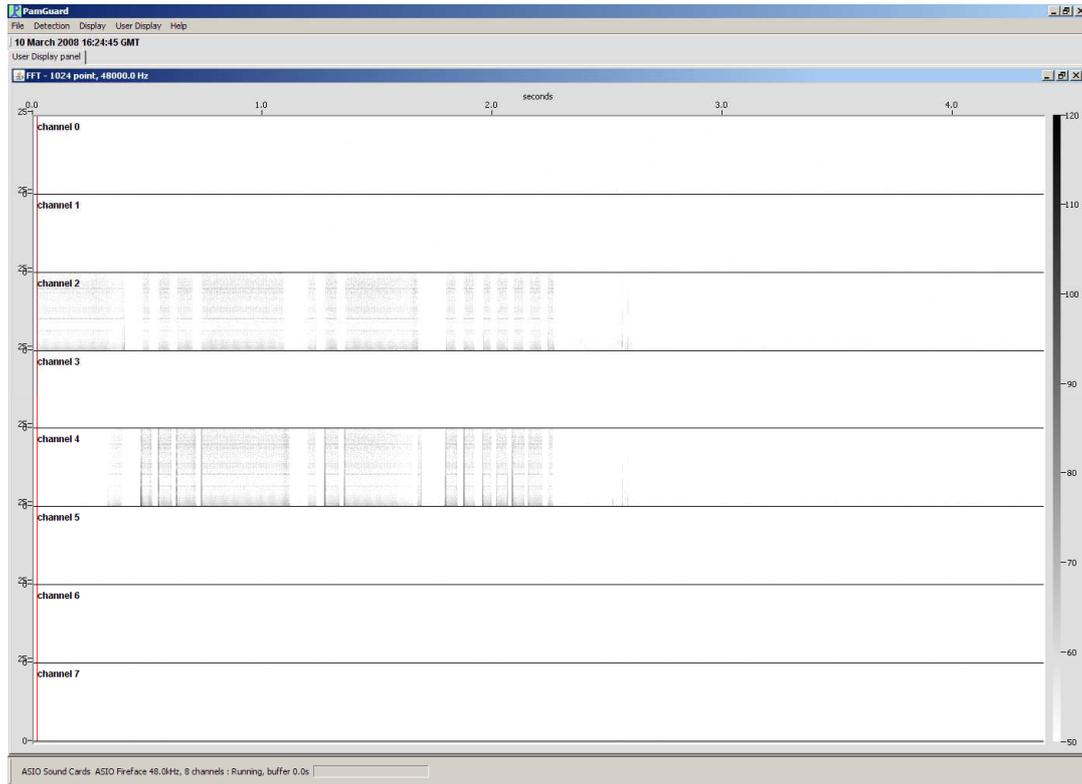


Figure 26 User Display of Selected 8 Channels

(HWU)

3.9.Simulator

PAMGUARD now features a simulator; whereby acoustic sources can be “placed” in the simulated environment. The sources produce periodic calls and their speed, depth and course can be altered by accessing the simulator side panel settings controls. The sources can also be dragged around the map with the mouse.

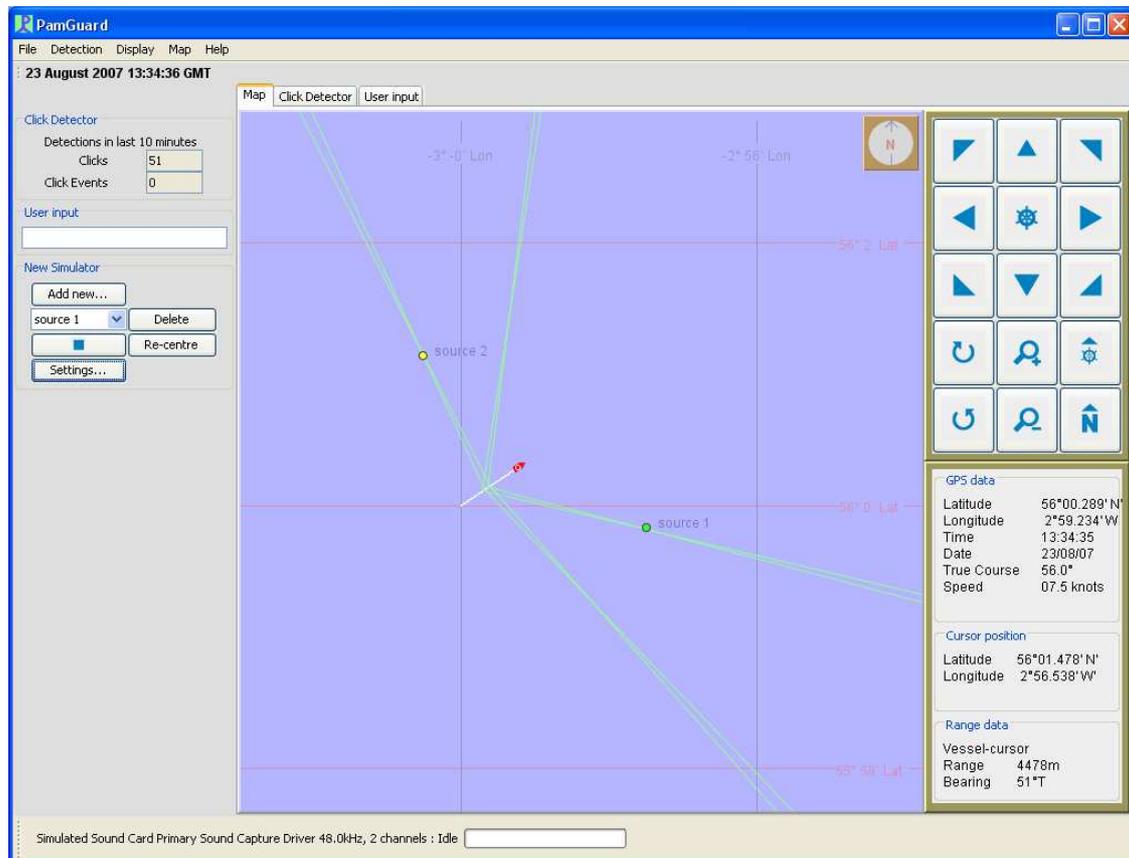


Figure 27 Simulator example

Figure 27 shows an example of the simulator in use, where tracked click bearing lines are shown on the map, corresponding with source positions.

The simulator must be used in conjunction with the “simulated sound card” sound source option, which utilises the system’s default sound card as a source for timing.

(HWU)

3.10. NMEA Data Acquisition

Previously, NMEA data acquisition was available to PAMGUARD indirectly via a UDP network connection. This was because there is no direct serial port support in Java. For situations whereby a NMEA network server was not present, users had to download the small C++ NMEAServer (IFAW) application from the PAMGUARD SourceForge download page. This provided users with a means to interface a serial port NMEA device with PAMGUARD. This was always intended as a work-around until another solution

was found which would allow PAMGUARD direct serial port access from a user's perspective.

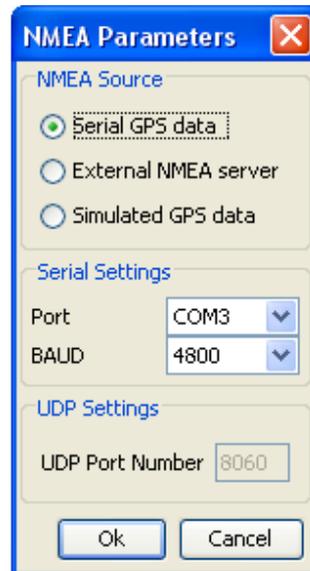


Figure 28 Serial GPS selection and settings

There is now a solution based on the RxTx package, an open source implementation of the Sun's Java CommAPI. This package has now been incorporated into PAMGUARD and users now have the choice of, for example, acquiring GPS data from the serial port or an external NMEA server, as well as simulated GPS data. Figure 28 shows the NMEA Parameters dialogue for the GPS, where the serial port can be selected and setup accordingly. (HWU)

3.11. 3D Tracker

Figure 29 shows a spectrogram and radar display of a simulated run of PAMGUARD, for a source 400 m deep and 400 meters horizontal range from the beam of a towing vessel. Channel 0 of the spectrogram displays the simulated data on the forward subarray, which was 200 m behind the towing vessel at 75 m depth. Channel 2 shows data on the rear subarray, 200 m behind the Channel 0 hydrophone, and also at 75 m depth. Each "doublet" visible in the plot indicates the arrival of a direct 'click' followed almost immediately by a surface-reflected multipath, which provides the animal's depth information.

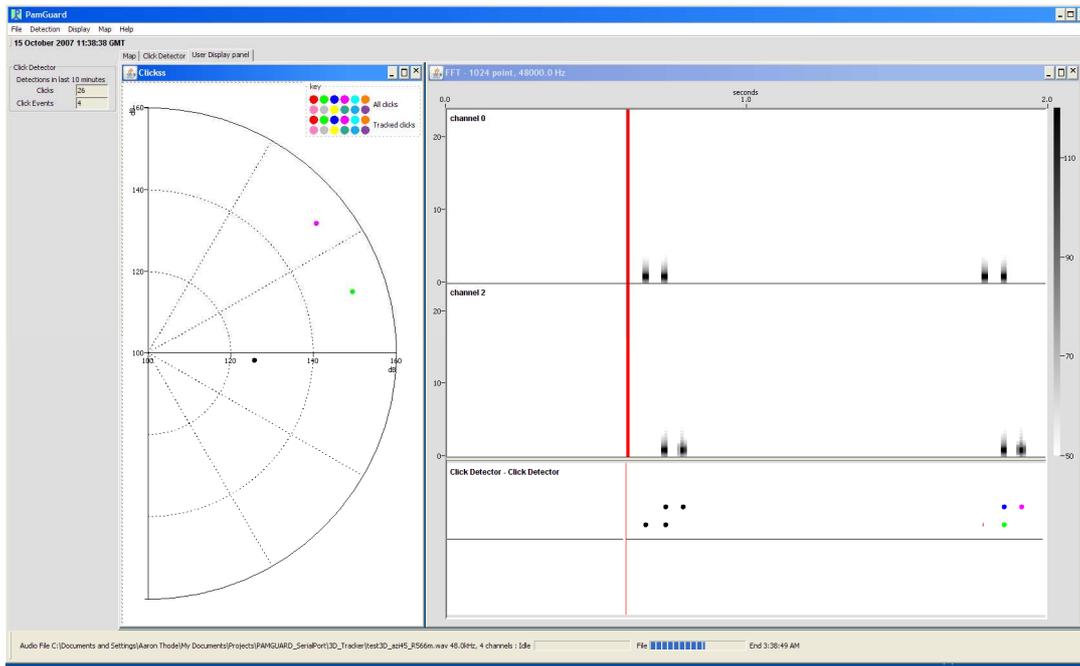


Figure 29 Spectrogram and radar plot of simulated 3-D source on PAMGUARD.

Figure 30 shows how a standard Bearing display in PAMGUARD plots the incoming data. Finally, Figure 31 shows how the 3D position is plotted in the Map view. The latitude and longitude of the animals' position is indicated by a red circle. The size of the circle is inversely proportional to the animal's depth, and the depth itself is printed next to the circle. Because of the bearing ambiguity two circles need to be plotted. Figure 32 shows an alternative simulation of a 200 m depth source at 60 degrees and 200 m horizontal range from the rear subarray. Note how the location circles are larger, since the animal's depth is shallower than was the case in Figure 31.

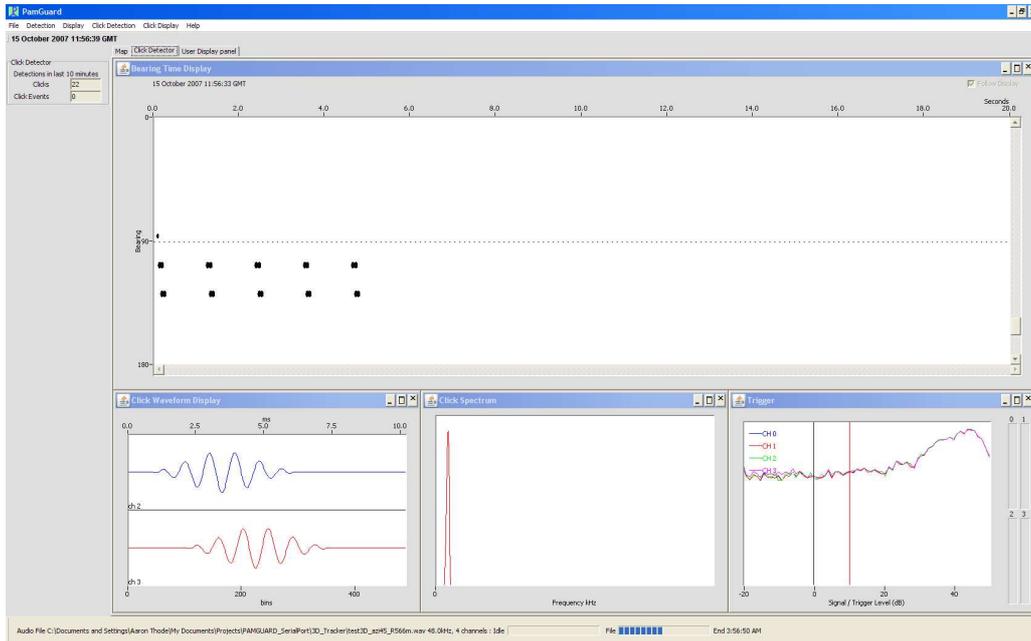


Figure 30 : Simulated 3D tracking data as it appears in Click Detector view.

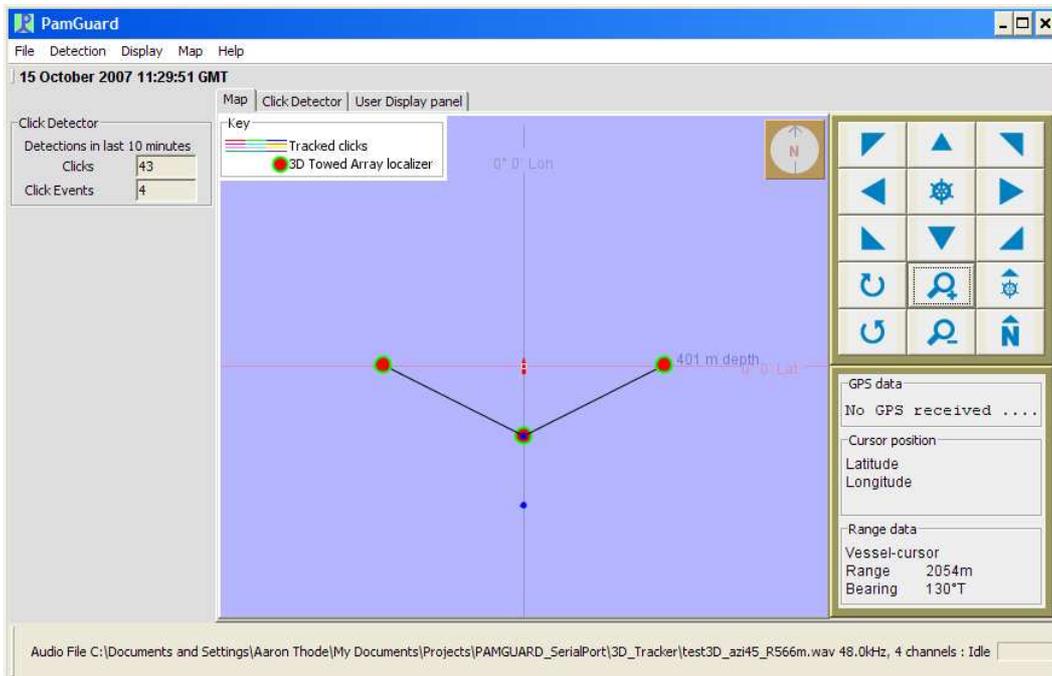


Figure 31 : Graphics output showing location of a 400 m deep source 400 m off the beam of the towing vessel, located using two sets of hydrophones towed 200 m and 400 m behind the vessel, respectively.

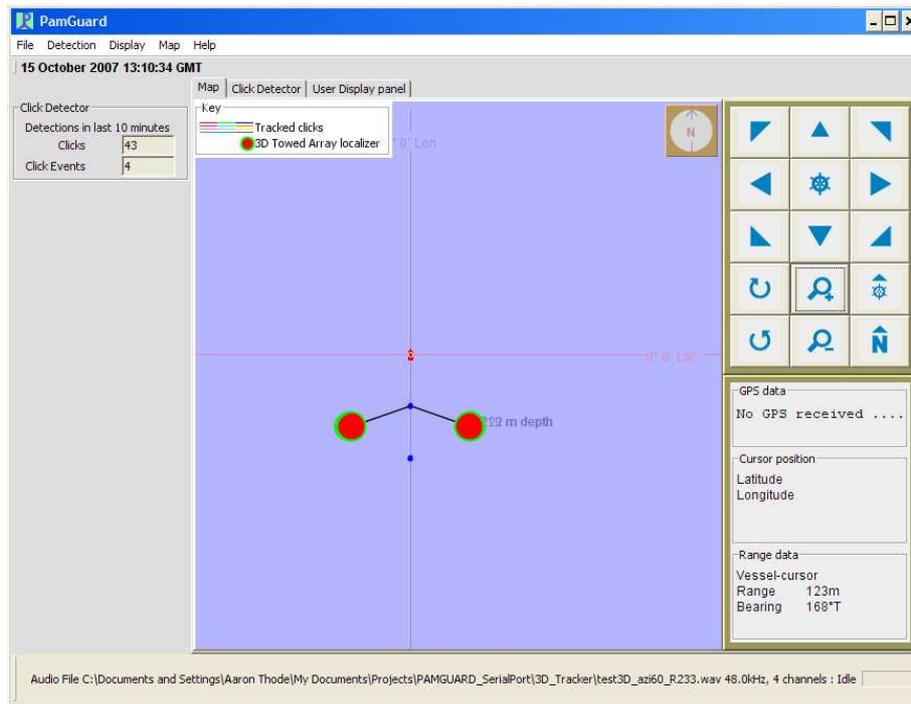


Figure 32 Graphics output of a different source at 200 m depth.

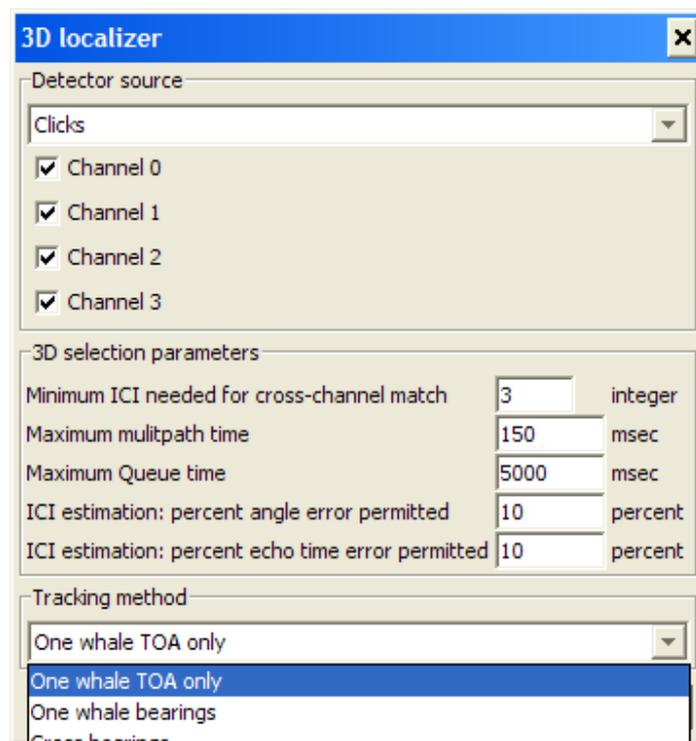


Figure 33 Parameter Dialog box for 3D tracker, showing pull-down menu for the three-different tracking algorithms.

The 3D tracking module permits three different tracking procedures, depending on the amount of multipath available on the array for tracking. The module also permits a simple 2D range/azimuth estimation by crossing bearings from the two sub-arrays. The particular tracking algorithms can be selected from the Parameter Dialog for the module, as seen in Figure 33.

(Scripps)

3.12. Ishmael detectors

PAMGUARD contains detection methods mirroring functionality present in Ishmael's detectors, including energy summation, spectrogram correlation (Figure 34a), and matched filtering, presenting the detection output in graphical form. During PAMGUARD 4, these detectors' classes were modified to stay up-to-date with changes to underlying classes that made the detectors fail. In addition, to implement logging of detections by PAMGUARD to a database, the classes implementing these algorithms now contain subclasses of PamDetectionLogging (as do the click and whistle detectors). See tutorial Exercise 8 for a more detailed description of how to use this functionality, and its capabilities.

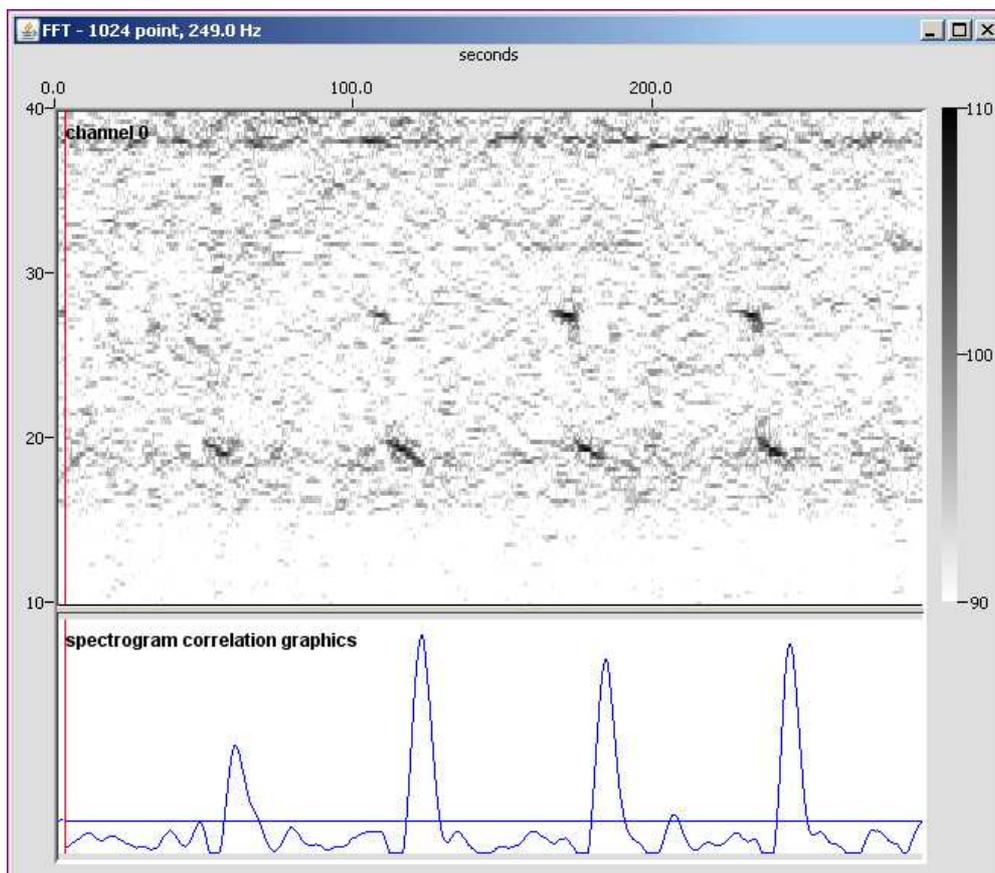


Figure 34a. Spectrogram correlation in PAMGUARD being used to detect calls of blue whales from Antarctica.

PAMGUARD also now incorporates an Ishmael “hyperbolic” localizer that locates vocalisations using multiple hydrophones. Figure 34b illustrates how to use it, and tutorial Exercise 9 gives a more detailed description of the localizer.

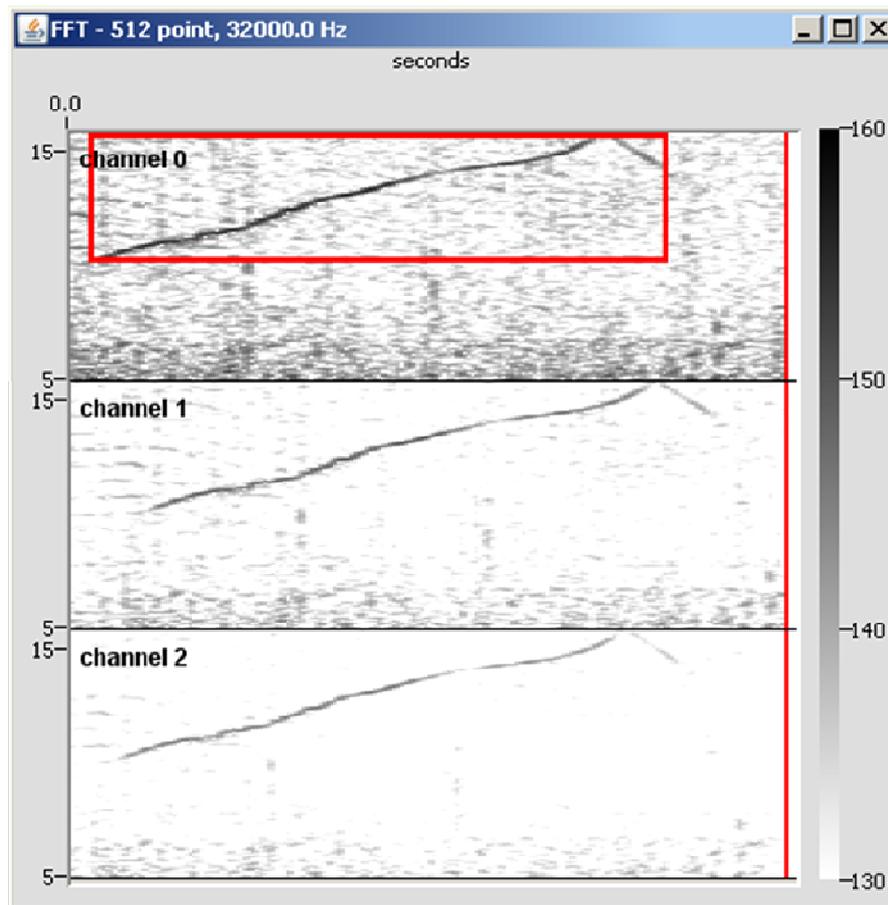


Figure 34b. Localizing a dolphin whistle; the red box indicates the selection to localize.

(OSU)

3.13. Other features

A channel spectra plugin display panel has been developed which displays FFT data for selected channels (Figure 35a).

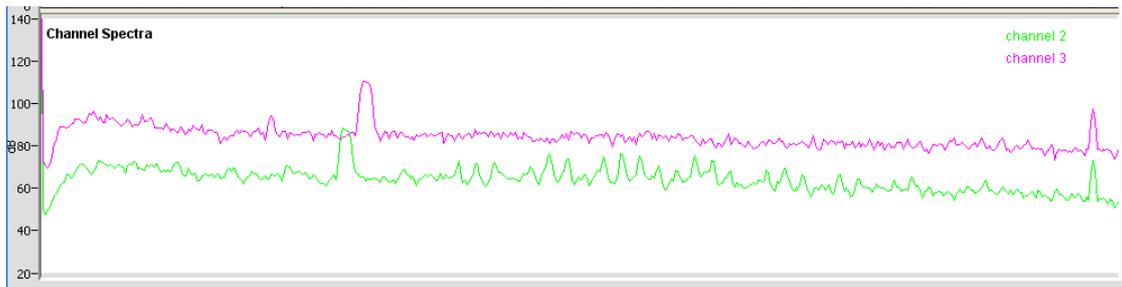


Figure 35a Example of channel spectra plugin display panel

There is a pop-up menu and dialogue which allow the user to set up the display. This allows users to select which channel(s) is(are) to be displayed. In addition, since the display of every individual FFT can often be difficult to interpret visually, a smoothing function has been added whereby users can select the number of individual FFTs used to provide an averaged spectra.

(HWU)

There is now better channel grouping in the Click detector for multi channel analysis and better displays for multi pair channel configurations. Figure 35b shows an example of multi-channel click detection using the channel grouping feature during the CODA/PAMGUARD trial in July. The first grouping (left) shows click bearing/time display for hydrophones 200m behind the monitoring vessel, and the second grouping (right) fir hydrophones 400m behind the vessel.

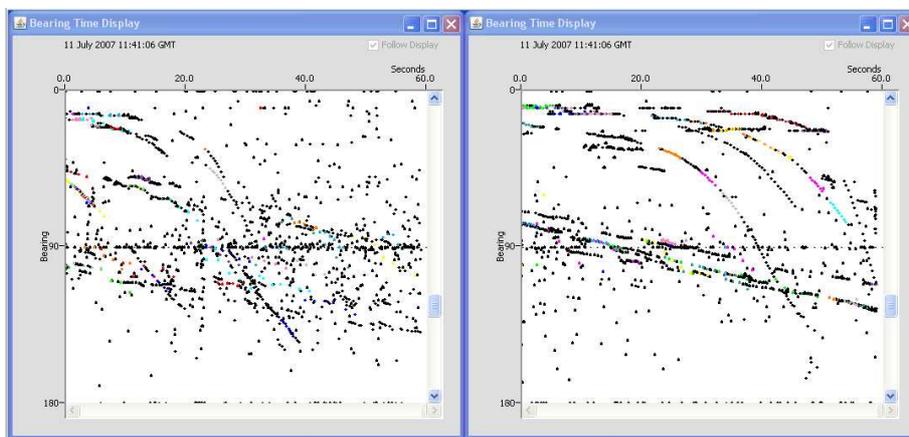


Figure 35b Four channel click detection using channel grouping feature

File writing has been speeded up and options to enable the user to switch off individual channels (e.g. if they are noisy) have been added.

(DG)

4. CODA/PAMGUARD Trial

4.1. Trial plan

A major field trial for PAMGUARD was carried out during the 2007 CODA trial in Block 3 (Northwest Spain). This trial was primarily organised by Doug Gillespie.

The original plan for this trial was for David McLaren to carry out leg one and Paul Redmond to carry out leg two. In light of Paul leaving the project at the end of June, it was thought best to employ the services of a professional PAM operator to carry out a user assessment, replacing Paul for leg two. Sebastian von Lüders, a member of “Marine Team” was to take up this role.

(DG)

The PAM system for PAMGUARD testing consisted of a 4 channel array with two separate groupings of two elements at 200, 203 meters and at 400,403 meters respectively. A RME FireFace 800 ASIO device was used for sound acquisition.

4.2. Leg 1

Leg one was on board B/A Investigadore and sailed from Vigo in North West Spain on the 1st of July and returned on the 15th of July. The main objective of this leg was to identify bugs in the code and, if possible, fix these during the cruise.

While much of each day was used for ruggedisation and debugging work, attempts were made to run the software for extended periods of time, especially when visual surveys were taking place. Attempts were made to track animals in real time using the graphic tools on the click detector.

The user input form was used to log detection/localisation information for subsequent comparison with visual data (enhancements were made to this module to make this easier – see section 4.2.5). Since PAMGUARD was being used intermittently to allow debugging, a master record of activities and audible detections was made using Logger.

This section describes which modules were tested and how and does not describe the bugs and deficiencies found in the software. The full bug record is in the full CODA report. In addition, a full daily diary for this leg is available on request.

On this leg, there were some problems with noise, predominantly on the 200m and 203m hydrophone channels. There were some spectral spikes which were reduced by re-arranging the earth configuration for the system. There were also transient noises which were investigated. The most likely sources were bubble plumes re-emerging from the

stern of the vessel around the first hydrophone pair (), and the second was found to relate to a loose boat rudder linkage being continuously agitated by the automatic pilot system.

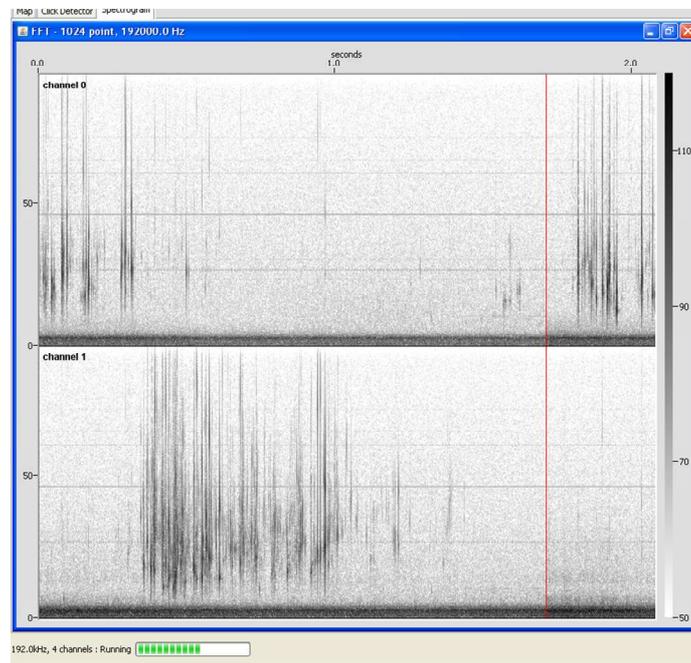


Figure 36 Transient noise on the 200m and 203m channels

The following sub-sections summarise which modules were tested and in which modes of operation.

4.2.1. Sound acquisition

The new ASIO sound card interface was tested extensively. A RME FireFace sound acquisition device was used and four input channels were acquired continuously. With de-bugging, 96kS/s operation was made reliable.

4.2.2. Database operation

During PAMGUARD sessions, either MySQL or MS Access databases were logging data continuously. Some bugs were identified; for example some modules were attempting to submit “illegal” data to the database so code was added to catch these entries and re-format as necessary. There were indications that the MySQL database introduced a performance reduction to the system when many detectors were running. This requires further investigation. Comments were continually made during the sessions and stored to the database. Logger was also used to store comments on PAMGUARD operation details, such as crashes and the “things heard” form in logger was also used as this feature is not present in PAMGUARD.

4.2.3. GPS acquisition

The GPS acquisition was found to be reliable. However, one of the GPS devices used was producing slightly erratic data. To reduce the impact of this on modules which required GPS data within PAMGUARD, a smoothing algorithm was added. This helped; however it is suggested that a more sophisticated algorithm is developed.

4.2.4. Map display

The Map display was tested extensively. For example, to ensure that the displayed information was geographically correct and that detector information was being displayed as it should be. A map file was loaded to ensure that this information was displayed. Figure 37 is an example of the map display showing the coastline, 100m and 1000m contours.

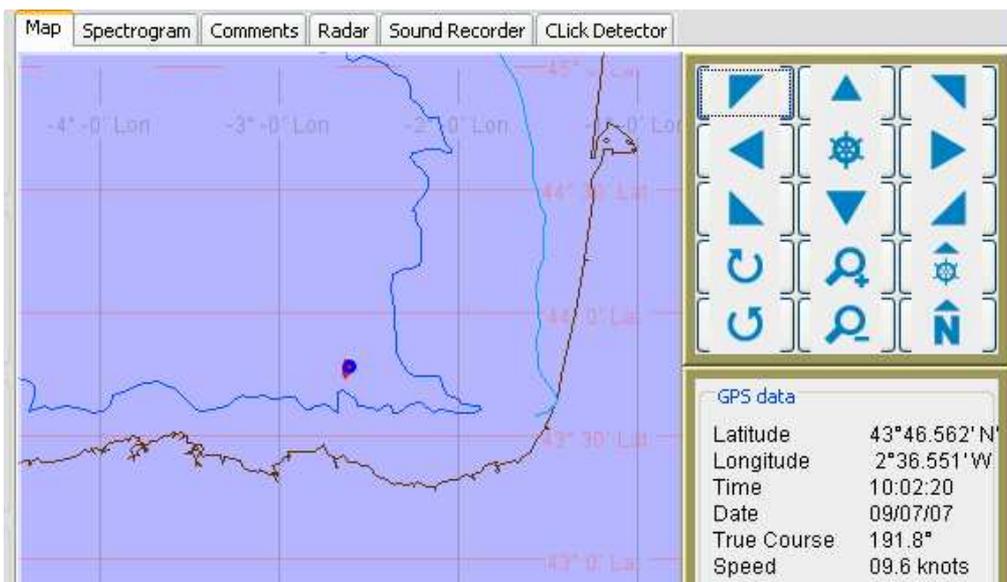


Figure 37 Map display example

4.2.5. User input module

The user input plugin was used to enter and log comments. Several new features were added to this plugin. For example, it became clear that changing to the user-input tab panel to enter a comment at busy times was inconvenient and took too long. To help enter information more rapidly, a side panel text entry field was created. In addition, to more quickly enter positional data, the latitude, longitude and bearing relative to vessel of the

last clicked mouse position on the map is copied to the system clipboard. These data can then be pasted rapidly into the user-input text entry field.

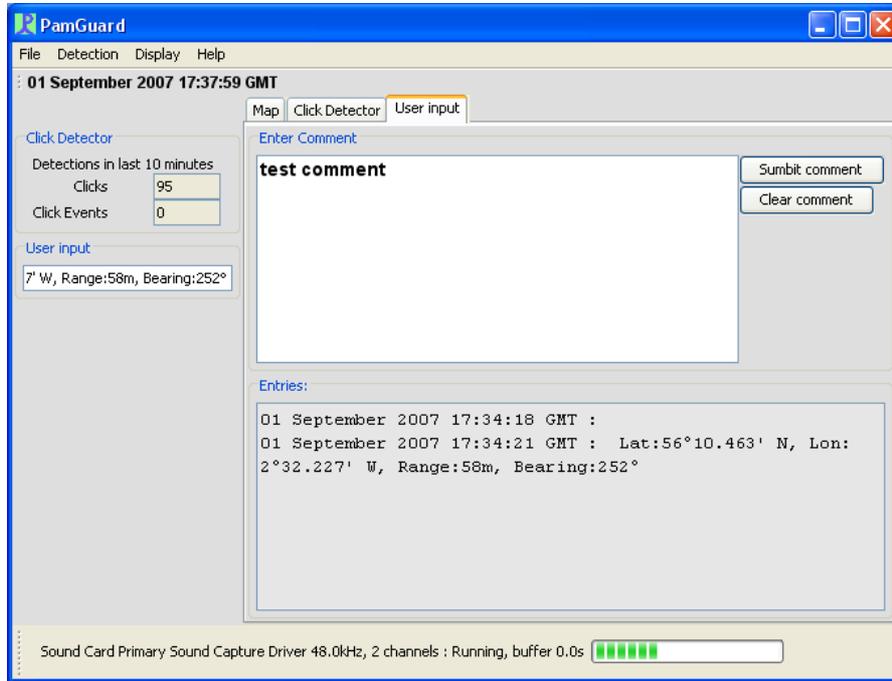


Figure 38 User input panel showing side-panel

4.2.6. User display panel

Multiple spectrogram and radar display panels were run simultaneously. For example, see Figure 39, where click and whistle detections are shown to be overlaid on both panels during an encounter with some dolphins.

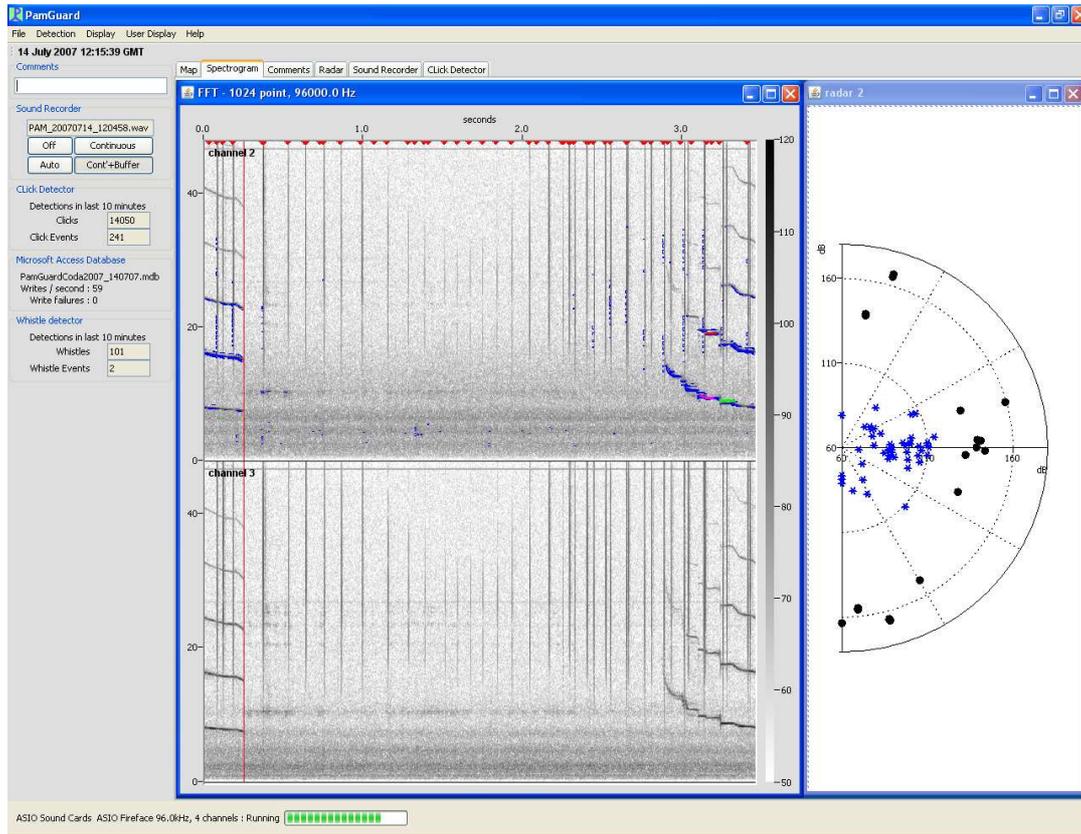


Figure 39 Spectrogram and radar displays

4.2.7. Detector modules

The click detector was predominantly run in 4 channel, 96kHz mode with hydrophones 1 and 2 (200m) in group 1, and hydrophones 3 and 4 in group 2.

Figure 40 shows manually tracked sperm whale click bearings plotted on the map. In this example the lines corresponding to each group are clear. Cross-bearing localisations were plotted rapidly by taking advantage of the physical separation between the hydrophone pairs associated with each group. While there has been no comparison between acoustic and visual data to date, this example shows promise.

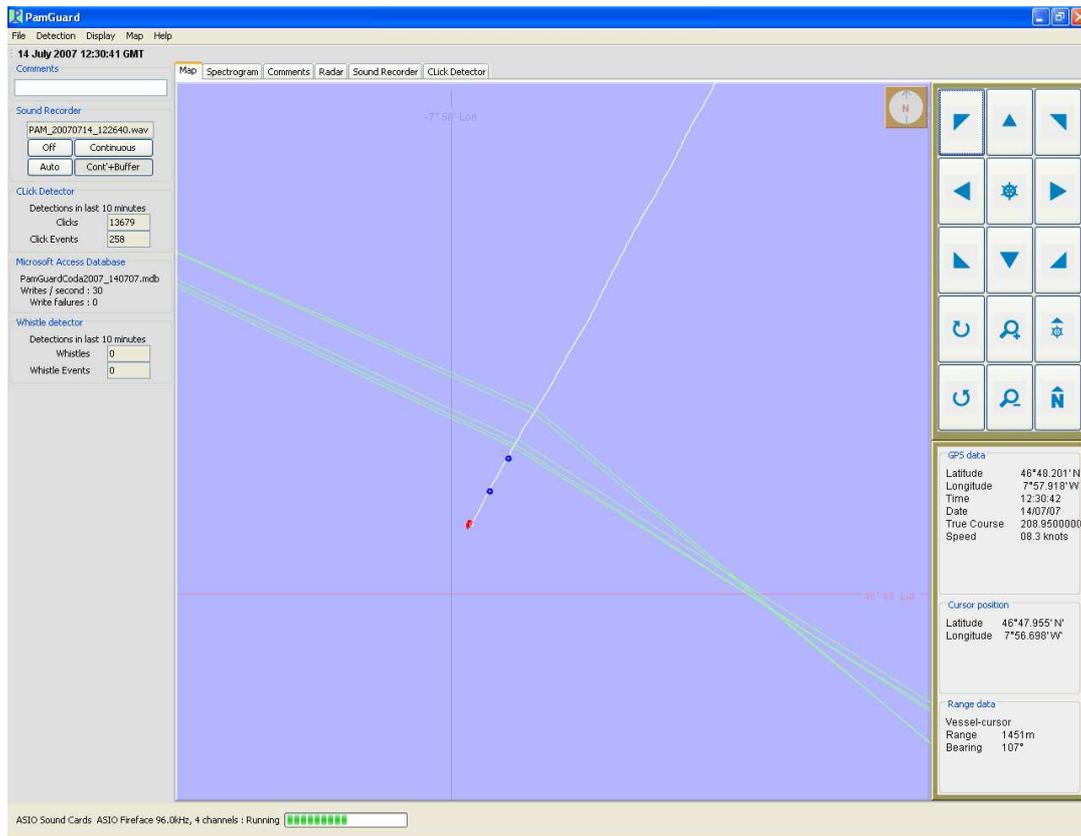


Figure 40 Manually tracked sperm whale bearings using two click detector channel groups

Since each hydrophone pair were situated in a substantially different acoustic environment (depth/noise-field, etc.) it was difficult to set up the pre- and trigger filters as there was and a compromise was required. It is therefore suggested that different filters can be applied to groups/individual hydrophones as the operational setup dictates.

Automatic click train identification was used for the most part. Figure 41 shows an example of this feature in operation for a group of sperm whales.

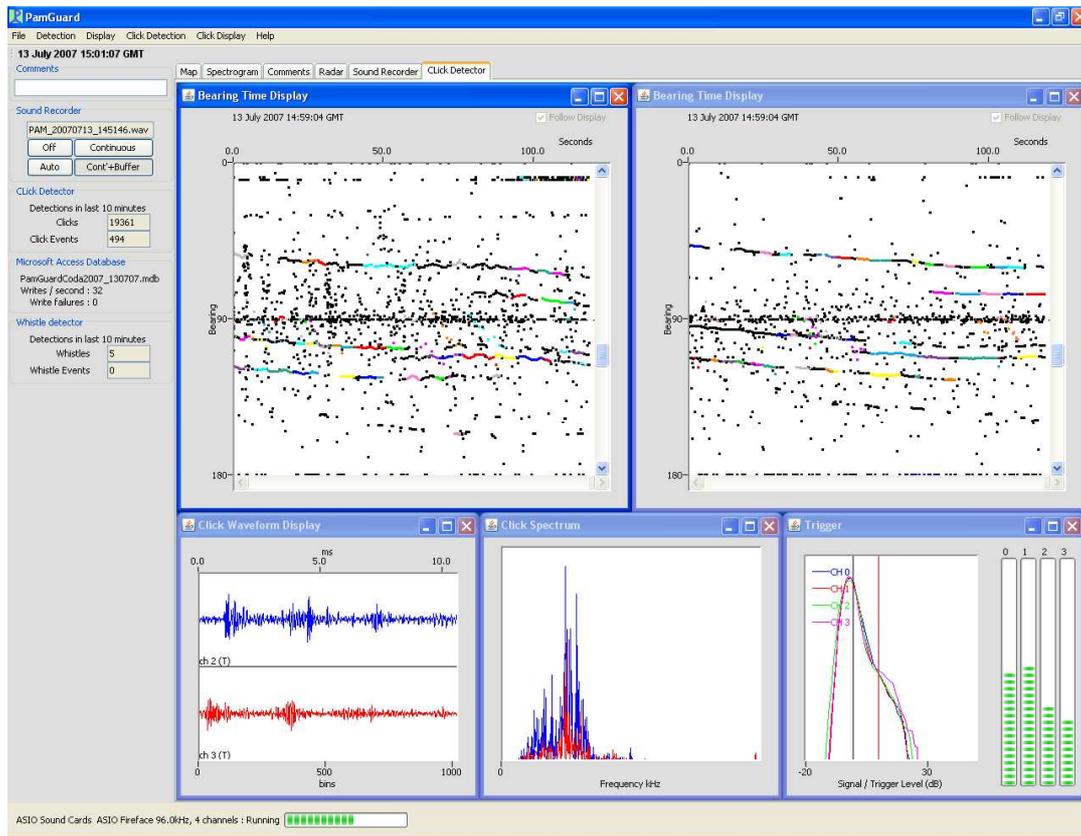


Figure 41 Automatic sperm whale click train identification example

4.2.8. Whistle Detector

The whistle detector was run using both pairs of hydrophones, although it was found to operate better using the rear pair, due to the reduced ambient and vessel noise. In general it was difficult to set up the whistle detector to reliably detect whistles for changing conditions and species. The detector was set up to run on channels 0/2, with bearings being calculated using channels 1/3.

Figure 42 shows whistle detection overlays on the spectrogram and radar panels.

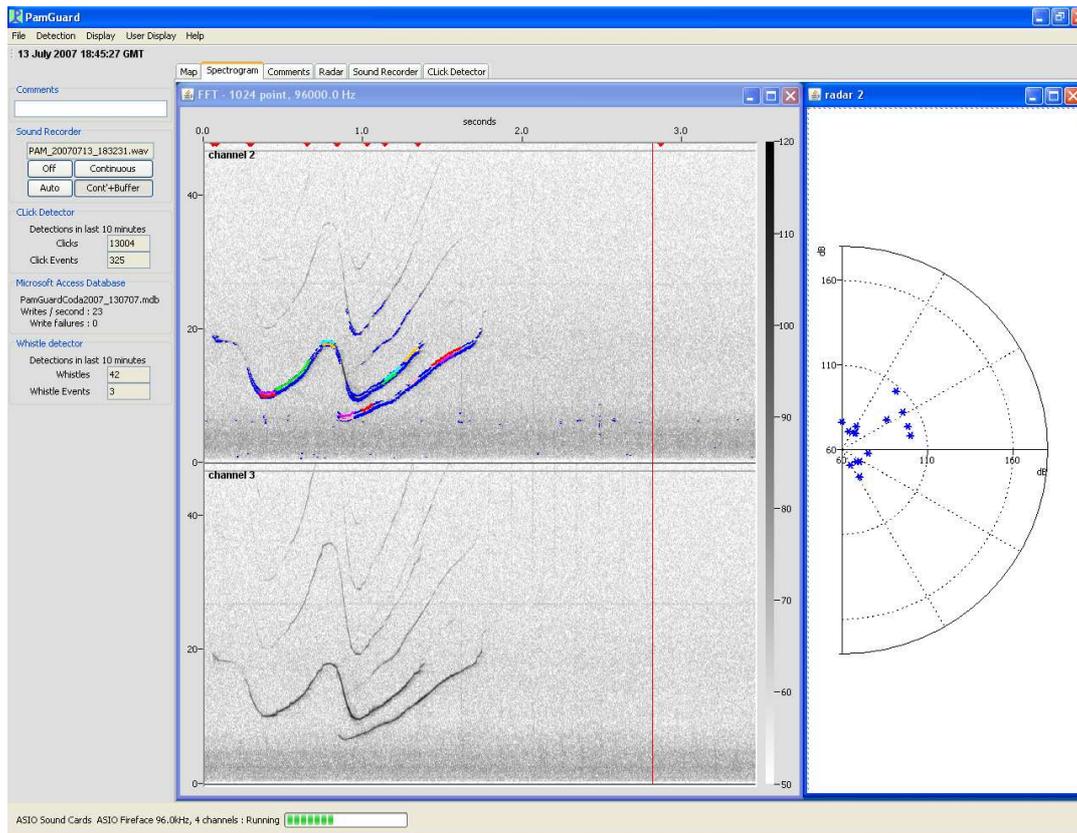


Figure 42 Example of whistle detection overlays

4.2.9. Sound Recorder

The sound recorder was used both in continuous and automatic trigger mode. Multiple channel recordings were made, up to four channels at a time. The system crashed several times when operating in automatic mode. However, many recordings were made successfully.

4.2.10. Aaron Thode's 3-D locator.

This locator was used in association with the click detector. While there were no bugs detected from its operation, it appeared that the program output was to the console and DJM was unsure of how to interpret the information.

4.2.11. Other modules tested

The following modules were also tested:

- Decimator
- IIRF Filters

Spectrogram smoothing
workshop detector

Each of these detectors performed well, although there were some problems which were added to the bugs list.

(HWU)

4.3.Leg 2

The second leg was on board the Spanish vessel Cornide De Saavedra and sailed between the 17th of July and the 1st of August.

For details of Sebastian's findings please see his report [ref. required]

5. Summary

The PAMGUARD project began investigating next generation Passive Acoustic Monitoring (PAM) software in 2004. By PAMGUARD 3 (2006 - 2007) functionality analogous to that of the “RainbowClick” software was available and ruggedised by sea-trials and functionality analogous to the “Ishmael” software was under development. 2D localisation was provided using stereo hydrophones.

This report documents PAMGUARD 4, the final phase of intensive PAMGUARD development. The code and user base have been supported as described in Section 2. The functionality of the core PAMGUARD architecture has increased substantially in this phase of the project as described in Section 3. The robustness of the software has been developed by a 2-week deployment on the CODA trial as described in Section 4. The PAMGUARD profile has been raised by ongoing web presence, software releases, conference presentations, and the PAMGUARD Workshop and training activities.

PAMGUARD phase 4 has met almost all the objectives and exceeded them in many parts. All of WP2 Maintenance has been delivered as detailed in section 2. The workshop, WP3, was delivered and is reported separately. Almost all of WP1 Core Architecture has been delivered.

Appendix A PAMGUARD User Tutorial (HWU/OSU)

PAMGUARD Overview

Many marine activities involve underwater sound emissions. These may be a by-product of the activity (e.g. piling or explosives), or a tool (e.g. air guns used for seismic surveys in oil and gas exploration, or military/commercial sonar). To mitigate against harm to marine mammals, observers are often employed to visually scan the sea surface for the presence of animals. In the event of a sighting, procedures such as suspension/delay of activities may be implemented to avoid harm.

Visual observations play a vital role, but marine mammals are difficult to spot on the sea surface, especially when weather and light conditions are poor. However, many marine mammals produce loud and distinctive vocalisations, which can often be detected more reliably than visual cues. For these species, passive acoustic monitoring (PAM) offers an effective means of detection. Furthermore, the creatures do not need to be on the surface to be detected.

A basic PAM system consists of a number of hydrophone elements (analogous to microphones in air, but used for underwater sound) arranged to form an array, hydrophone signal conditioning/amplification, a signal acquisition device (e.g. a sound card) and a computer running PAM software. Sounds in the water are converted by the hydrophones into electrical signals which are conditioned appropriately and converted into digital signals for processing by the PAM software.

The PAMGUARD project was set up to provide a standard software infrastructure for acoustic detection, localisation and classification for mitigation against harm to marine mammals, and for research into their abundance, distribution and behaviour.

PAMGUARD is open-source Passive Acoustic Modelling (PAM) software based on a platform-independent (e.g. Windows or Linux), flexible, modular architecture.

This architecture makes it relatively straightforward to incorporate new modules as they are developed to include additional detection, localisation, classification and sound visualisation functionalities. The software offers a versatile software/hardware interface to enable flexibility in the configuration of in-sea equipment (number of hydrophones, sensitivities, spacing and geometry).

PAMGUARD is a sophisticated software package that can be used by the expert user to set up industry/research PAM infrastructure. It can also be configured for operational use by MMO PAM operators. Central to the software design is a flexible core architecture which allows the integration of a range of additional plug-ins. For more information on PAM and PAMGUARD please visit pamguard.org.

This tutorial provides an introduction to PAMGUARD from a user's perspective. This involves exercises in setting up and running the software with a variety of configurations of plug-in modules.

The table below shows the core Built-in and a selection of the plug-in modules, some of which will be utilised in this tutorial. Core modules are always present in a version of the PAMGUARD application, whereas plug-In modules are optional, and can be added and removed as suits a given monitoring activity.

| Core Built-in Modules | Plug-in Modules |
|------------------------------------|---|
| Array configuration | MAP |
| Model Manager and Profiler | NMEA/GPS Acquisition |
| GUI Container for plugins | Acquisition devices |
| Help Manager | FFT (Spectrum) factories |
| Configuration and settings manager | Decimator |
| Standard graphics layout classes | Sound recording |
| | Spectrogram smoothing |
| | Spectrogram displays |
| | Radar displays |
| | Click Detector |
| | Whistle Detector |
| | Energy Sum Detector |
| | Spectrogram Correlation Detector |
| | SQL database interface (requires MySQL) |
| | Source simulation |

Tutorial Learning Outcomes

The purpose of this tutorial is to provide a basic overview of how to use PAMGUARD, including adding and configuring plug-in modules.

The main learning outcomes are:

- Loading and running PAMGUARD configurations
- Navigating the Graphical User Interface (GUI)
- Basic Map familiarisation
- Running and configuring a Click Detector module
- Adding and configuring User Display panels
- Overlaying detector graphics on display panels
- Adding and configuring FFT Engines
- Adding, configuring and running a Whistle Detector
- Using spectrogram and radar displays to view detection information
- Using “Ishmael” Plugins

These learning outcomes can be achieved through a series of exercises that now follow.

The exercises can be seen as a guide, so you’re encouraged to experiment with PAMGUARD as you work through each one.

Exercise 1. PAMGUARD familiarisation

1.1. Launching PAMGUARD

To launch PAMGUARD, locate the PAMGUARD.jar file on your computer and double-click it. The following dialogue will appear:

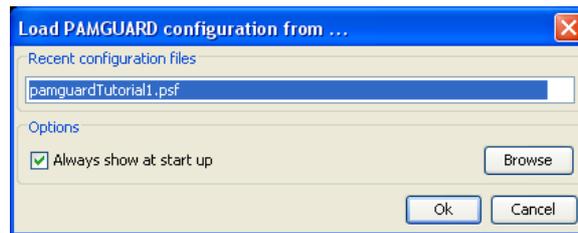


Figure 43. PAMGUARD configuration dialogue

This dialogue allows the user to choose from the selection of PAMGUARD configuration files available on your system. You may need to browse to locate the settings files.

Note: In operation, operators may have a suite of setup files available for particular types of monitoring tasks and/or customise these for particular tasks and situations.

Select the appropriate configuration file from the drop-down menu and click OK. In this case, this will be “pamguardTutorial1.psf”.

When *OK* is clicked, the dialogue will disappear and the PAMGUARD application will launch. During the launch, the following dialogue will appear.



Figure 44. Set PC Clock Dialogue

You should ignore this dialogue for this tutorial. After a few seconds it will disappear. In real operation this would synchronise your PC clock with the time provided by the GPS system.

1.2. PAMGUARD GUI overview

Once launched, the PAMGUARD Graphical User Interface (GUI) will look something like Figure 45.

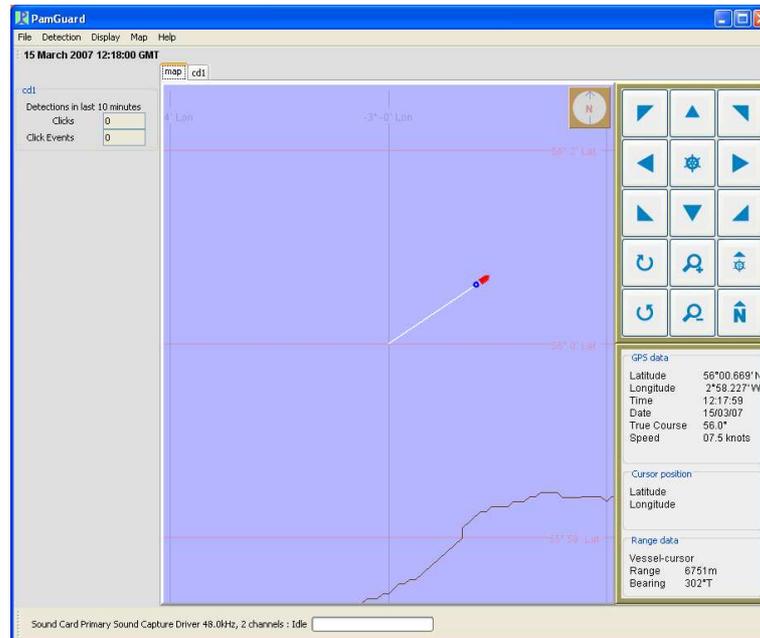


Figure 45. PAMGUARD Graphical User Interface

The GUI is subdivided as described in the following table.

| | |
|----------------------|--|
| <i>Menu bar</i> | This provides access to many PAMGUARD settings and controls, as well as access to the help system. PAMGUARD has a dynamic menu bar system, whereby the menu items change according to the current view and settings. |
| <i>Clock display</i> | Shows the current time. |
| <i>Side panel</i> | The side panel gives quick access to information and controls for currently active modules. |
| <i>Tab panel</i> | Tab panels provide access to the main visual interfaces of many of the PAMGUARD modules. In this example the map tab panel is shown, which consists of a map, compass and some GPS data etc. |
| <i>Status bar</i> | This bar provides information on the current status of PAMGUARD, e.g. whether it is running or idle |

Spend some time investigating the user interface.

1.3. Using the map

At this point, PAMGUARD should be running with the “exercise 1” configuration. Check that this is the case by selecting File ⇒ Load Configuration.

This will open the configuration dialogue. The current configuration will be selected in the drop down list. If this is not “exercise 1.psf”, locate and select it. Otherwise, click Cancel.

PamGuard can run in simulated or real GPS modes. In the tutorial configurations, the GPS is simulated and the “vessel” icon will appear on and move over the map display. If you cannot see this, make sure you have selected the map tab panel by clicking on its tab and click on the “centre map on ship” map control button (see table below).

Experiment with the map controls, which have functions as described below:



Centre map on vessel

This control pans the map such that the vessel which supplies PAMGUARD’s GPS feed is represented in the centre of the display by the  icon.



Zoom out/in

These controls allow the user to zoom in and out of the map view. These functions can also be achieved using the mouse wheel.



Pan

These controls allow the user to pan the map view in steps. “Free” panning can also be achieved by clicking and dragging the map with the mouse.



Ship/North orientation

These controls set the map’s orientation to ship’s heading and North, respectively.



Rotate map

These controls allow the user to rotate the map to a desired arbitrary orientation.

Note the information panel at the lower right hand corner of the map. This shows useful information such as the position of the vessel which is supplied by the GPS feed to PAMGUARD.

Move the mouse cursor around the map and note that the Latitude and Longitude of the cursor position is provided and in addition, the range from the mouse cursor to the host vessel. This is useful for range estimation.

1.4. Setting up a sound source

First of all you will need a source of sound. PAMGUARD modules can acquire sound data in a number of ways, e.g. sound can be acquired directly from the system sound card, from some other digital acquisition device or sound files can be loaded and “re-played”.

In this exercise, you will play a sound file in the computer’s media player (e.g. Windows Media Player) and PAMGUARD will acquire the data from the system’s sound card via the computer’s software mixer.

Start the computer’s media player and load the file named “GomSpermWhales.wav”. This is a recording of sperm whale vocalisations made from a research vessel in the Gulf of Mexico

Make sure that the media player is set to play the sound in a loop (set the player’s play settings to “Repeat” (for windows media player, press *Ctrl-T*).

Next, check that the computer’s software mixer’s recording settings are set to record the stereo mix. This may be called “Stereo Mix”, “Wave Out Mix” etc. For windows users, the mixer is accessed via the control panel and will look something like this:

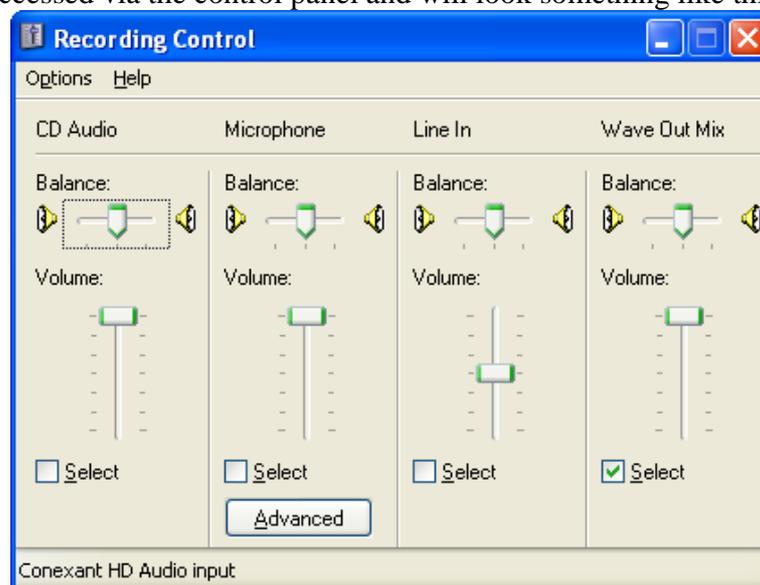


Figure 46. Setting the Windows mixer

If you have difficulty with these steps, ask a helper or consult the computer’s help system.

Select Play in the media player and, if you have headphones or speakers, you should be hearing the sound file. Once these steps are carried out, PAMGUARD will be ready to process the sound data.

Exercise 2. Running the Click Detector

The loaded configuration already has a Click Detector plug-in added. Clicking on the Click Detector module's tab will bring forward the main Click Detector interface panel (Figure 47).

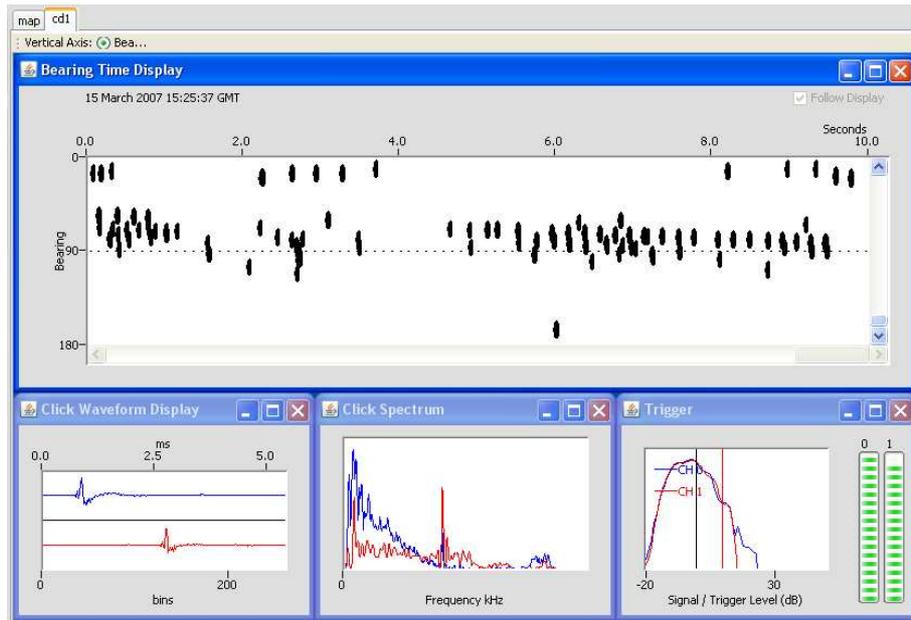


Figure 47. Click Detector Tab Panel

The panel has four main components:

| | |
|-------------------------------|---|
| <i>Bearing/Time display</i> | Each detected click is shown as a circle or ellipse on the scrolling display. The toolbar at the top of the display can be used to select the type of vertical axis - bearing, amplitude or inter click interval. |
| <i>Click waveform display</i> | The waveform of each detected click appears on the display as it is detected. |
| <i>Click spectrum</i> | The power spectrum of each click is displayed as it is detected. |
| <i>Trigger display</i> | The trigger window shows the amplitude of the signal on each channel as a decaying histogram. The vertical red line represents the trigger threshold set in the detection parameters dialog (cross reference). Level meters for each channel are also shown on the right hand side of the trigger window. |

Start detection by selecting **Detection** ⇒ **Start** from the main menu. You are now running PAMGUARD in active mode!

1.5. Tracking clicks manually

The bearing time display scrolls with time and detected clicks can be identified as ellipses. As clicks appear on the bearing time display, the Click Detector's side panel display (Figure 48) updates the number of clicks detected in the last 10 minutes.

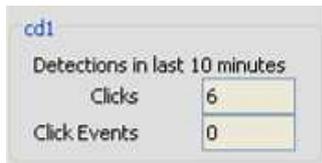


Figure 48. Click Detector side panel display

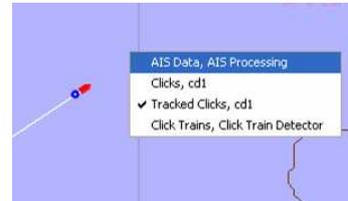


Figure 49. Map Panel drop-down menu

To track detected clicks manually, select a click in the Bearing Time display by left-clicking on it. This will draw a cross on that click **+** to confirm its status as being tracked. Next, click on the map panel tab to bring forward the map display. Using the right mouse button (or for Macs, hold the ctrl key and click) click on the map display to bring up the map detection drawing options. This provides a selectable list of items which are suitable for drawing on the map. An example is shown here:

Make sure that “Tracked Clicks” is selected (as indicated by a checkmark ✓) for the Click Detector which you are using – in this case “Click Detector 1”. This will draw the bearing lines for that click on the map panel, as indicated in Figure 50:

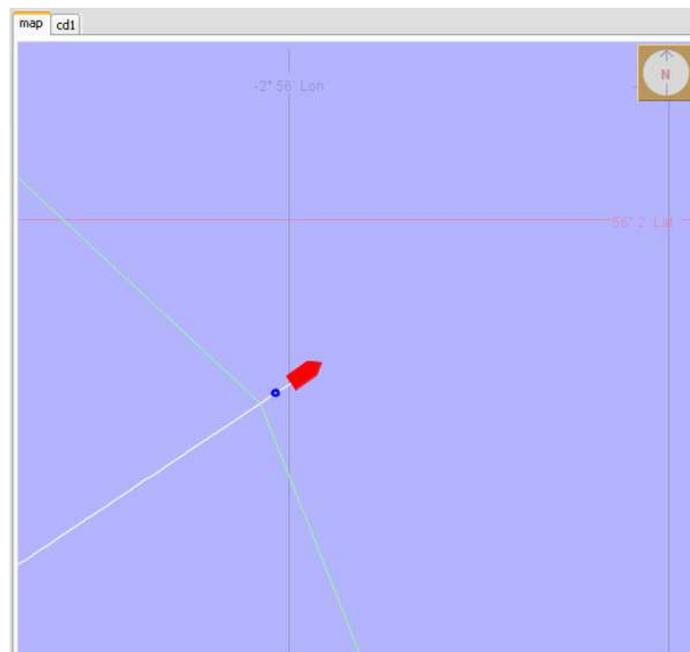


Figure 50. Tracked click bearing lines plotting on the map panel

1.6. Tracking clicks automatically

The Click Detector can automatically detect sequences of clicks on a consistent bearing with a consistent inter click interval. Clicks that satisfy these constraints are referred to as *Click Trains*.

To enable this feature, select the Click Detector tab panel. Then select *Click Detection* ⇒ *Click Train Identification...* from the menu bar. This will bring up the following dialogue:

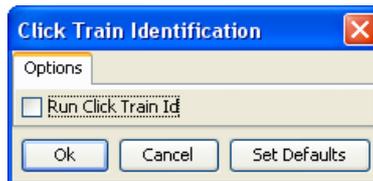


Figure 51. Click Train Identification Dialogue

Click on the *Run Click Train Id* check box and select *OK*.

Note that access to the same dialogue is also possible by selecting **Detection** ⇒ **"Click Detector Name"** ⇒ **Click Train Identification** from the main menu – where **"Click Detector Name"** is the name given to the particular Click Detector module running. For this exercise this will be "Click Detector 1".

Now look at the Bearing Time display. Identified click trains are indicated by coloured ellipses, as in the Figure 52.

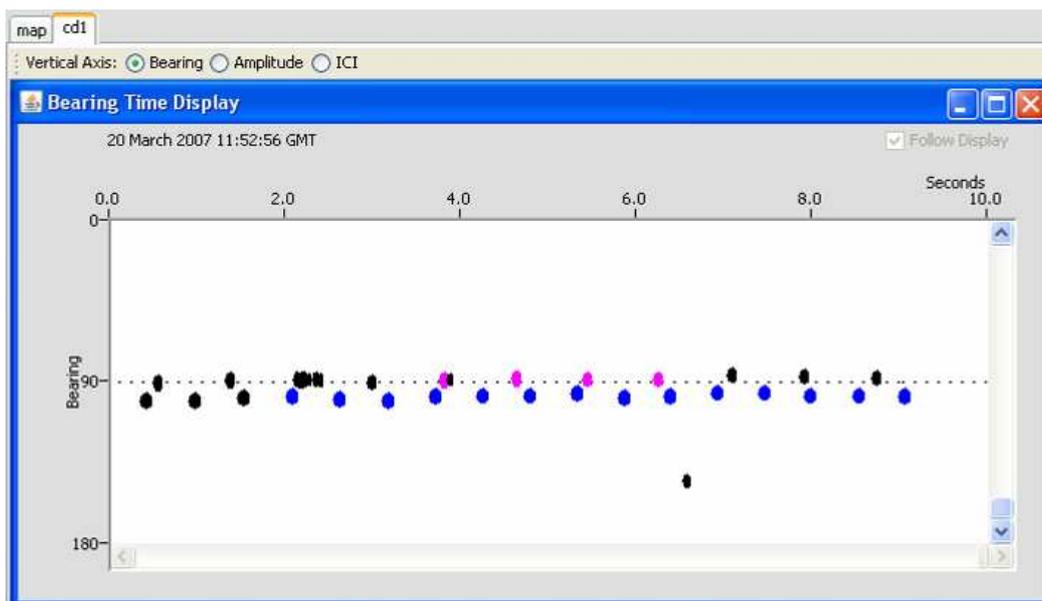


Figure 52. Click trains plotted on the bearing time display

Now switch to the map view.

Position the mouse somewhere on the blue map area and press the right mouse button to display the drop-down detector selection box and make sure that “Click Trains” is selected.

If a click train reaches sufficient length and the bearing change is adequate, target motion analysis automatically calculates a range and bearing to the sound source. The first and last bearing lines associated with this click train sequence are also automatically displayed on the map. An example is shown in Figure 53.

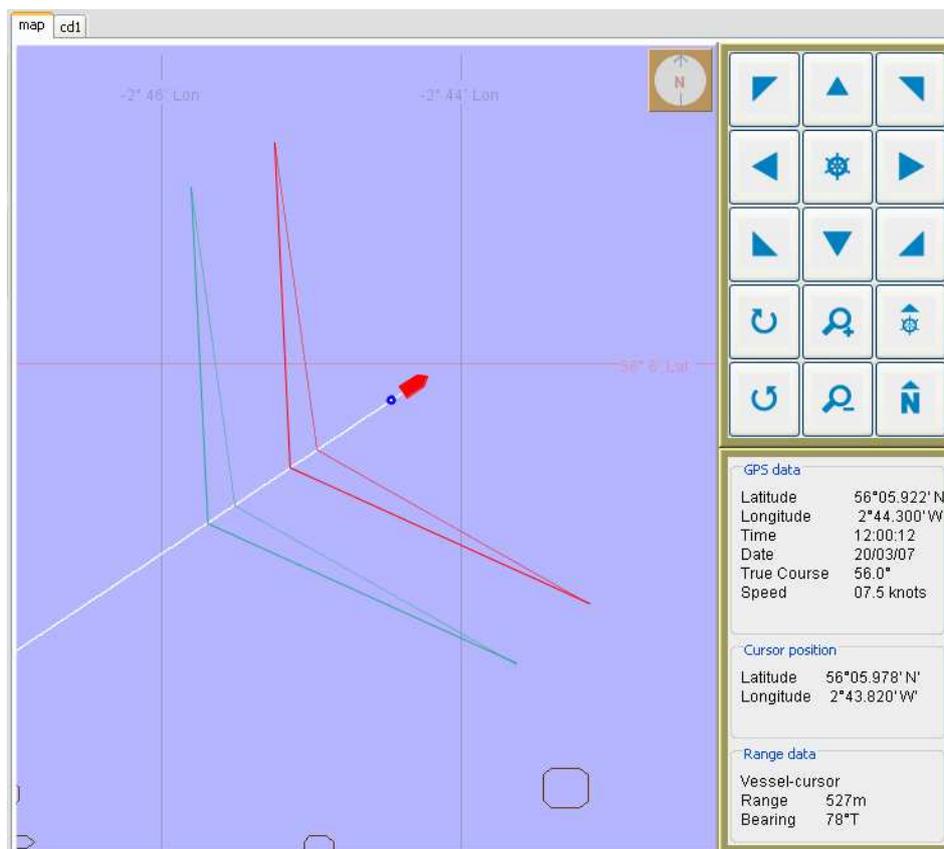


Figure 53. Click Train range estimates drawn on the map

Note that the lines terminate at the point of bisection, indicating the range estimate. Now hover the mouse cursor over this point and observe the Vessel-cursor range and bearing on the map information panel.

Take some time to experiment with the Click Detector and map views.

Try the program with the file “SpermSurvey.wav”, a recording made in the course of a real seismic survey from a survey vessel of Australia.

Select **Detection** ⇒ **Stop** from the main menu to stop the detector.

Exercise 3. Running the Whistle Detector

1.7. Load the sound file

To experiment with the whistle detection in PAMGUARD, load the sound file “whistles.wav” into your media player. Again, make sure your player is set to repeat playback (*Ctrl-T* in windows media player) and start the player.

1.8. Adding a Whistle Detector module

Next, we will add a Whistle Detector module to PAMGUARD. Ensure that Detection mode is not active (*Detection* \Rightarrow *Stop*). Select *File* \Rightarrow *Add Modules...* \Rightarrow *Whistle Detector* from the main menu. The following dialogue will appear:

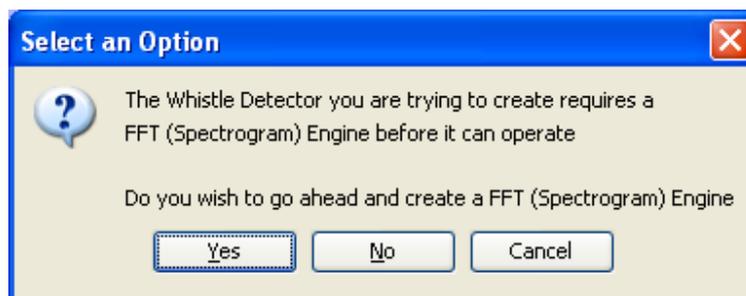


Figure 54. FFT Engine creation dialogue

Whistle detector modules require spectrogram data to work. FFT modules provide spectrogram data and PAMGUARD knows at this point that no FFT data producing modules are present. PAMGUARD therefore asks if you would like to make one. Click *Yes*. A new dialogue will appear asking you to name the FFT Engine module.



Figure 55. Naming the FFT Engine module

In this dialogue, enter *FFT1* as the name for the new FFT Engine and click OK. The next dialogue to appear is for the configuration of the FFT Engine.

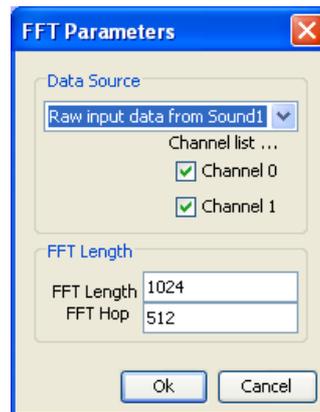


Figure 56. Setting the FFT data source

From the drop-down data source box, select “**Raw in put from Sound Acquisition**” and make sure that both channels are checked. Leave the rest of the parameters as default and click OK. Next, enter WD1 as the name for the Whistle Detector module.



Figure 57. Naming the Whistle Detector module

Click OK. Access to the Whistle Detector settings is from the main menu, **Detection ⇒ Whistle Settings...**

The Whistle Detector will also place an item on the side panel as shown in Figure 58.

Now that you have changed the PAMGUARD model configuration, click

File ⇒ Save configuration as... and save the configuration as “clickAndWhistleDetectors.psf”.



Figure 58. Whistle detector side panel display

Close down PAMGUARD (**File ⇒ Exit**) and re-launch the application. Choose “**clickAndWhistleDetectors.psf**” from the **Load PAMGUARD configuration from...** dialogue and click **OK**.

Note: This closing down operation is not necessary for PAMGUARD to operate with the new configuration – it’s just to give you practice!

1.9. Adding a Spectrogram Display

Whistle detections can be viewed in a number of PAMGUARD graphics modules. For this exercise you will firstly use a Spectrogram Display. Spectrogram Displays are placed on a generic User Display panel, so first of all you have to add a User Display panel to PAMGUARD. To add the new User Display panel, you will now use the Data Model display. Select **File** ⇒ **Show data model...** from the main menu. This will open up a new Data Model display window.

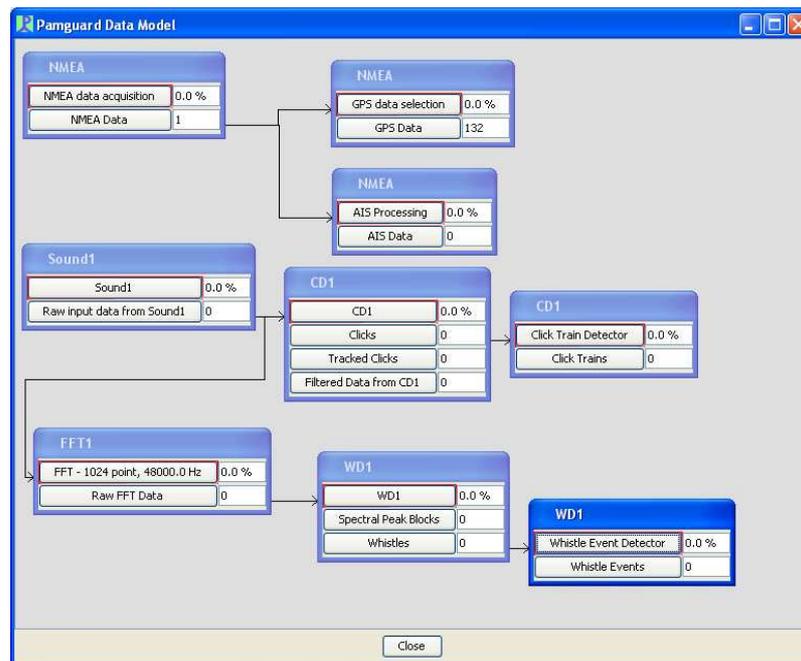


Figure 59. The PAMGUARD Data Model window

Place your cursor anywhere on an empty space on the data model display area and, clicking the right mouse button, select **Add Modules...** ⇒ **User Display panel** from the drop-down menus. Enter “User Display 1” as the name of your panel in the new module dialogue and click OK. Note that the User Display module does not appear as part of the data model. This is because the user display cannot receive or produce data, and acts merely as a canvas for other displays. Click *Close* at the bottom of the Data Model display window.

To add the Spectrogram Display to the new User Display panel, select the User Display tab on the tab panel selector. Notice the main menu items change as you select the various tabs. Select **User Display** ⇒ **New Spectrogram...** from the main menu. We are going to use 2 spectrogram panels in slightly different ways, so enter “2” in the number of panels box. Hit the *Enter* key on the keyboard to confirm the value and select channel 0 and 0 from the two channel selection drop-down lists. Click *OK* and the new Spectrogram Display will appear on the **User Display 1** panel.

1.10. Asio Sound Card Channel Selection

When an Asio card is plugged in, the PamGuard can detect it automatically. In the **Audio Data Acquisition** module, select ASIO Sound Cards, and select the type of your ASIO card, which is ASIO Fireface in Figure 18. In the PamGuard a certain number of channels can be read. **Figure 18** shows that 4 channels (0,1,2,4) have been selected. **Figure 19** shows to create 4 panels for these selected 4 channels. **Figure 20** shows the user display of these 4 channels. **Figure 21** shows the result of 8 channels.

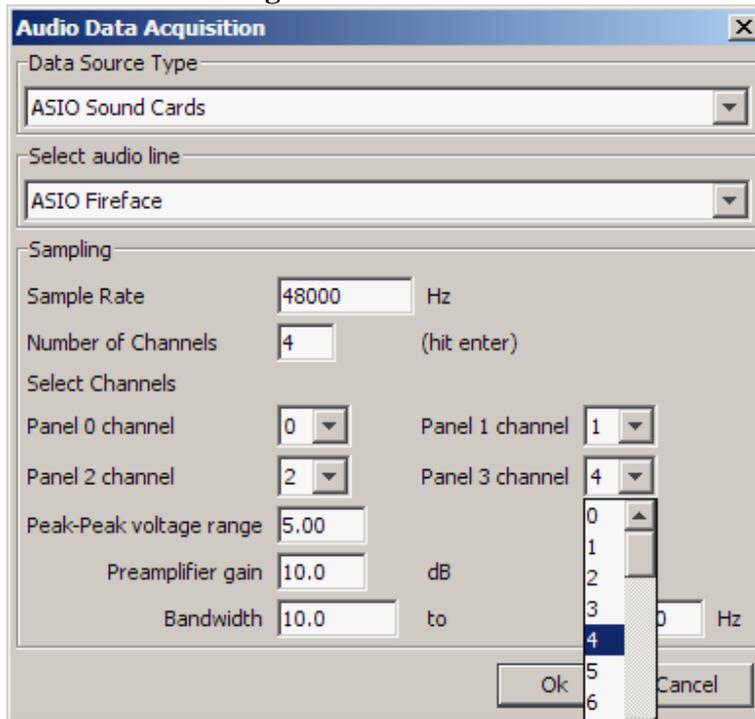


Figure 18: Channel Selection.

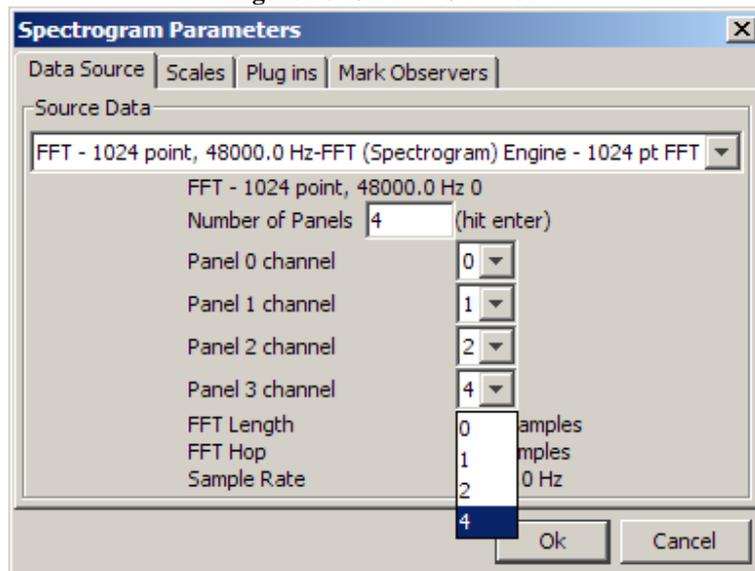


Figure 19. Panel Selection.

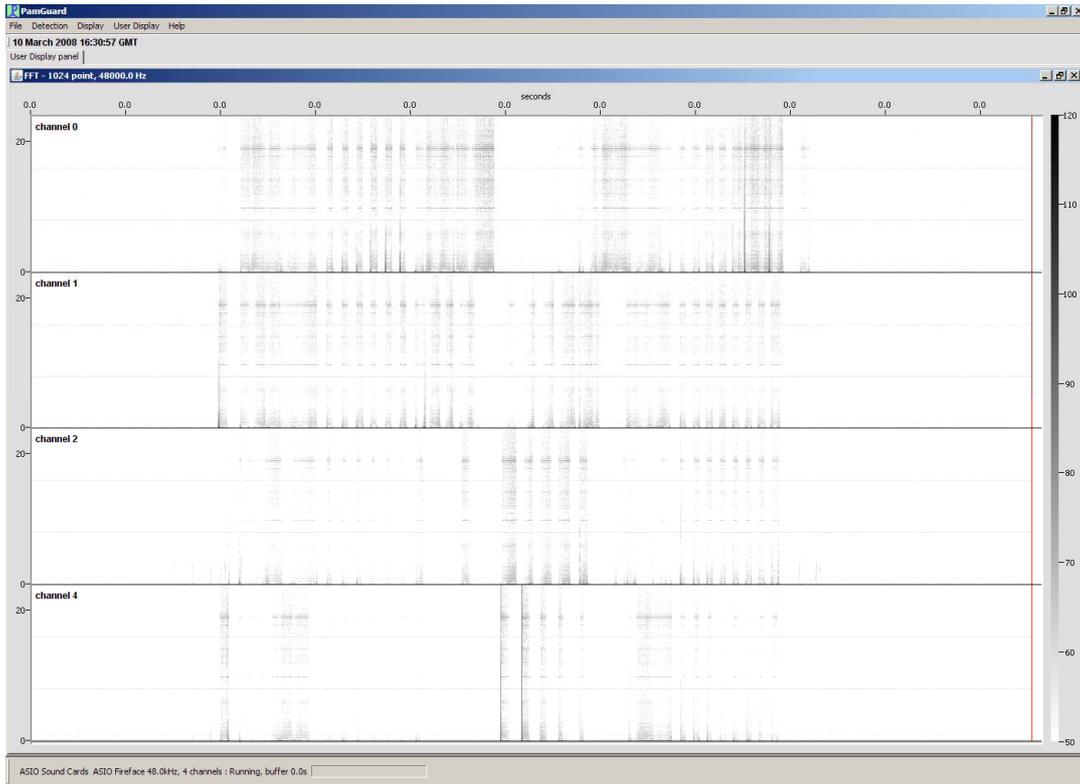


Figure 20: User Display of Selected 4 Channels.

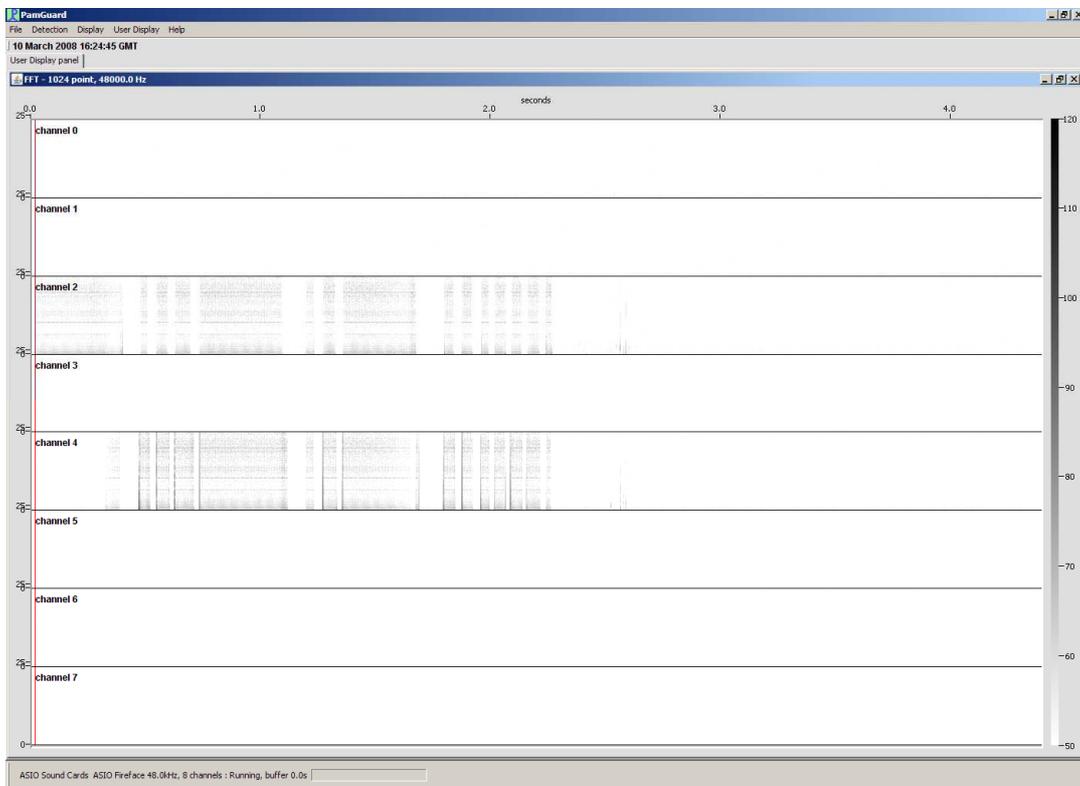


Figure 21: User Display of Selected 8 Channels

1.11. Viewing whistle detections

Start detection **Detection** \Rightarrow **Start** and the two spectrogram panels will show the spectrogram data in real time.

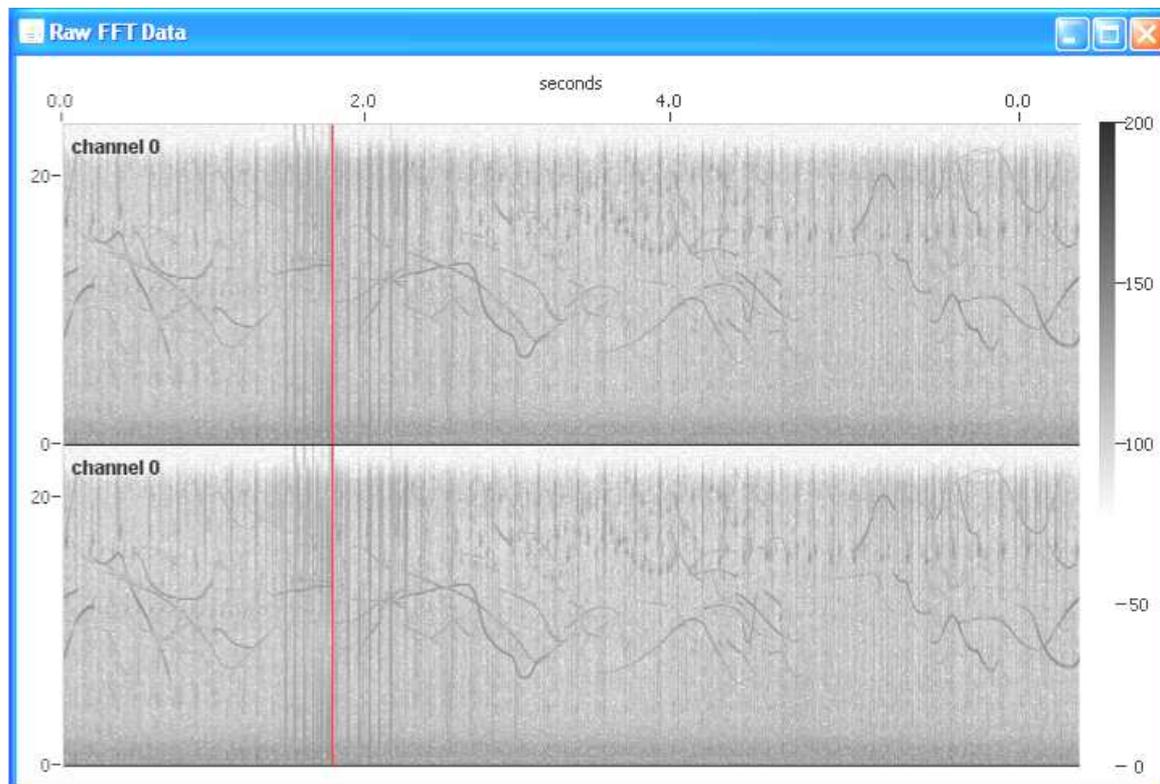


Figure 22. A Spectrogram panel display

Right-click over the spectrogram display and choose **Settings...** from the menu. Now click on the **Scales** tab. Experiment with the spectrogram display by adjusting the amplitude range settings and clicking OK. This changes the range over which the spectrogram displays spectral amplitude data. Repeat this until you are happy with the view. For the sound file you have running at the moment, you should see regular whistle contours as in the image above.

Now, right-click over the top panel on the spectrogram display and this time choose **Spectral Peak Blocks**. Small blue rectangles will now be overlaid on the channel 0 panel on top of peaks detected in the sound spectra. These peaks are candidate components for whistles.

Where consecutive peaks are connected in such a way as to satisfy the Whistle Detector criteria, a “whistle detection” is created. Now, right-click over the lower panel on the spectrogram display and this time choose **Whistles**.

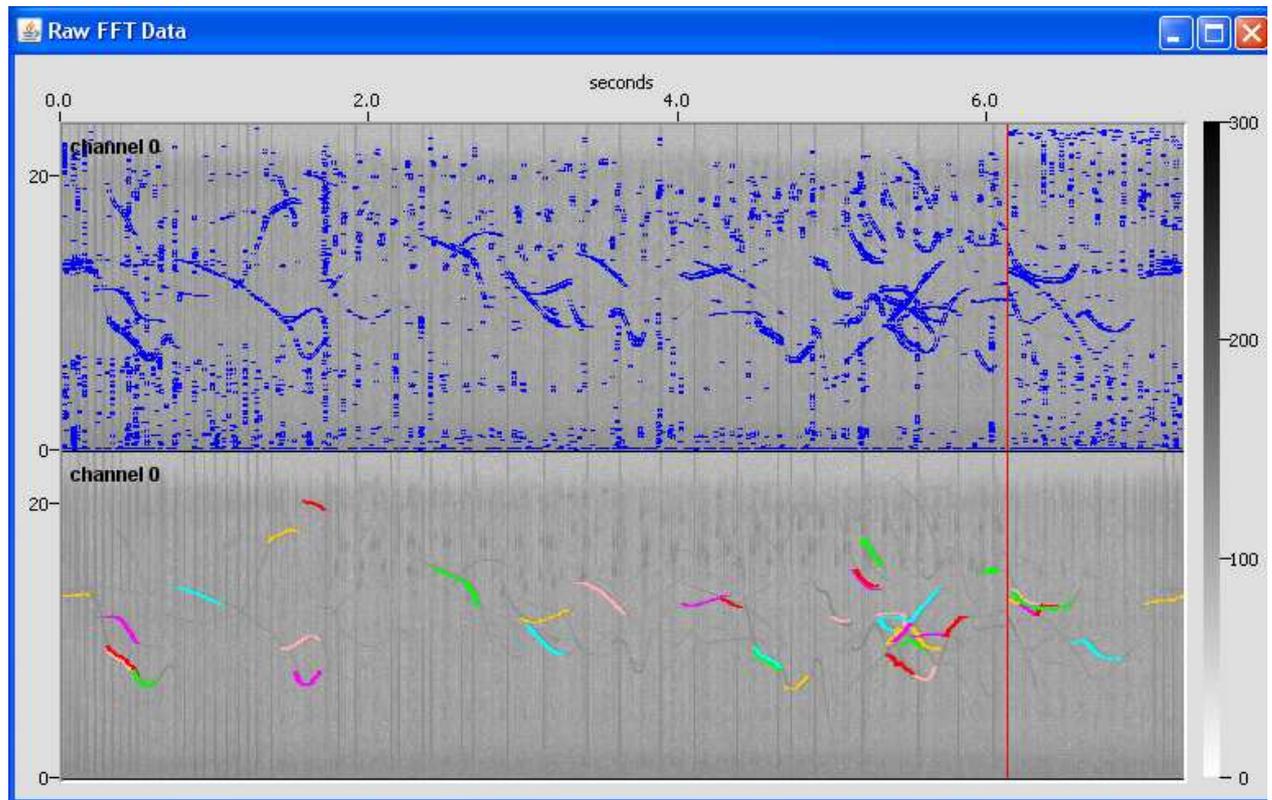


Figure 23. Spectral peaks and whistle detections on spectrograms

As is illustrated in the figure above, detected whistles will now be overlaid on the panel as coloured lines which track the detected whistles.

1.12. Data from Seismic Surveys

Try running the whistle detector with recording made during seismic mitigation exercises

File “whistlesurvey.wav” Was made from a seismic source vessel off Australia with electrical gating of the airgun pulse. The whistles come from a bottlenose dolphin.

File “whistlesguard.wav” was made from a guard vessel during a seismic survey on the Atlantic Frontier.

Exercise 4. Advanced display features

There are a more display types available in PAMGUARD which will be explored here.

1.13. Adding further displays to Spectrogram Panels

It is often useful to view multiple data and detection types simultaneously. For example, various detector modules active in PAMGUARD can draw detections on the Spectrogram panels. While detection is running, right-click over a spectrogram panel and choose “Clicks”. Detected clicks will now be plotted on the panel as red circles on the top of the plot.

In addition, plots can be added to the bottom of the Spectrogram panel. Right-click over the spectrogram display and choose *Settings...* from the menu. Now click on the **Plug-in** tab. Select additional plug-in display panels by clicking on the check boxes for **Raw input data from Sound1** and **Click Detector** ⇒ **Click Detector 1**. Click **OK**.

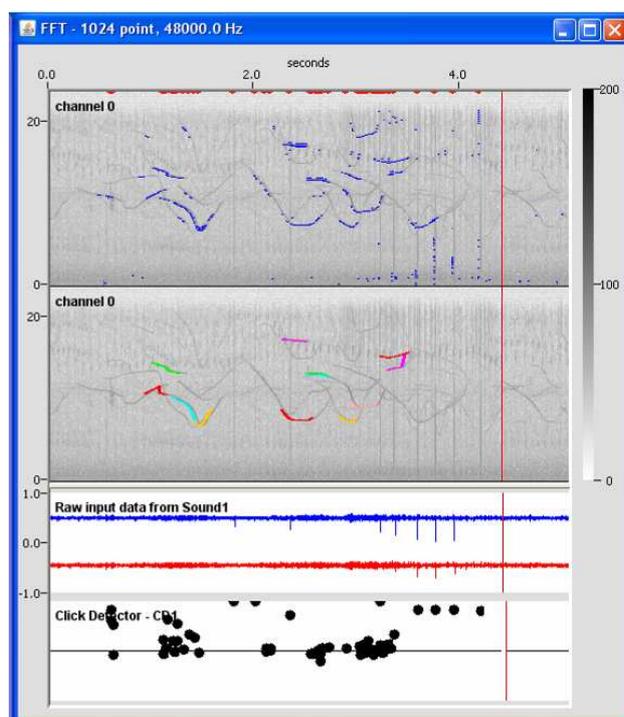


Figure 24. Raw data and Click Detector plugin panels added to a spectrogram display

As illustrated in the figure above, the raw sound waveforms are plotted and also a compact version of the Click Detector display is shown.

1.14. Adding a Radar Display to the User Display panel

We will now add a “Radar” type display to the User Display. Radar display windows can be used to display range, amplitude and bearing information to detected sounds. To add the Radar Display to the User Display panel *User Display 1*, select the *User Display 1* tab on the tab panel selector. Select *User Display* ⇒ *New Radar display...* from the main menu.

Since many detectors, particularly those using simple linear hydrophones, produce ambiguous bearing information, it is possible to display either the full display or only one half of the display. We will display a half display called *RD1*, so type in the name and select *Right half only* from the Style option drop-down list. Once the Radar Display is on User Display 1, select, *User Display* ⇒ *Arrange Windows...* ⇒ *Tile Vertical* from the main menu. The display should look something like Figure .

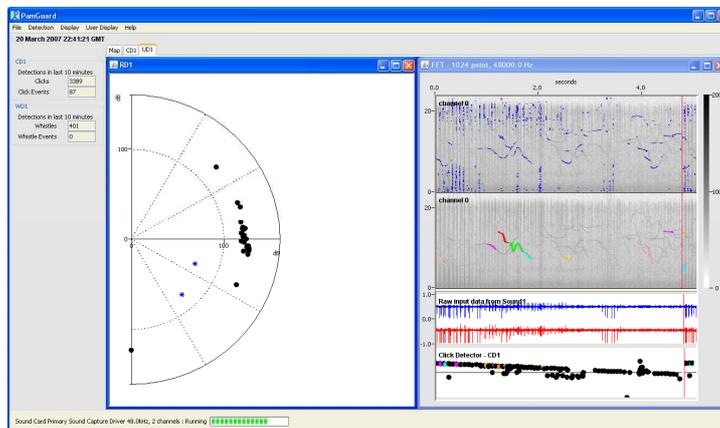


Figure 25. A Radar display added to the User Display Panel

Right-click over RD1 and choose *Settings...* ⇒ *Detectors*. From the Detectors tab, select the *Show Detector Data* settings as indicated in Figure .

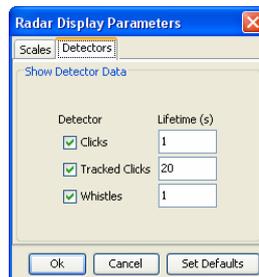


Figure 26. Detector selection and lifetime settings for Radar Display

Click *OK*. This will set the radar plot to display Clicks, Tracked Clicks and Whistle detections for 1, 20 and 1 seconds respectively. Experiment with the plot settings and, when you have finished, select *Detection* ⇒ *Stop* from the main menu in preparation for the next exercise.

Exercise 5. Detection during seismic source operation

1.15. Loading the sound file

First load the sound file “spermWhalesPlusSeismic.wav”.

This is a recording of sperm whale vocalisations taken during an active seismic survey. The presence of a significant sound source, such as firing air guns has implications for the operation of detection algorithms.

You will notice an artefact on the central axis (indicating no time delay between channels). This is noise picked up after the seismic pulse which has been gated out.

Here we will investigate changing the Click Detector settings in an effort to reduce the adverse effects of the unwanted noise.

Select **Detection** ⇒ **Click Detector 1** ⇒ **Detection Parameters...** from the main menu. Click on the Set defaults button to restore the detectors settings to their default values. Start Detection (**Detection** ⇒ **Start**) and observe the Click Detector tab panel displays. You should see something like this:

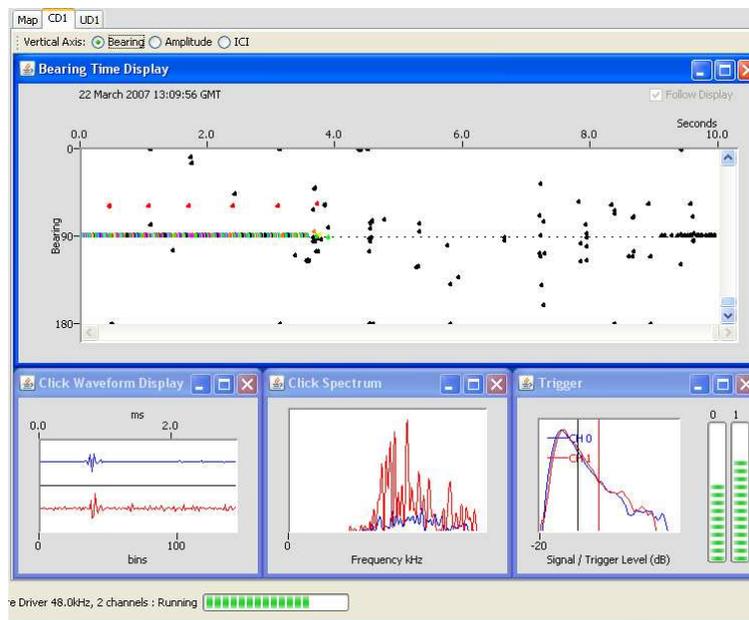


Figure 27. Click Detector display example for noisy seismic data

Notice the rapid click detections associated with the electrical noise on the 90° bearing.

1.16. Adjusting click detection parameters

Now select the *Amplitude* radio button on the *Click Detector Bearing Time Display*. This changes the Y-axis of the plot to represent the amplitude of detected clicks. Notice that the rapid clicks consistently appear at lower amplitude to the majority of the rest of the click detections (Figure). If all of the clicks appear at the top of the display, you may need to adjust your computer's mixer settings to reduce the amplitude of the captured sound. You can also adjust the amplitude scaling of the display (Figure 2960).

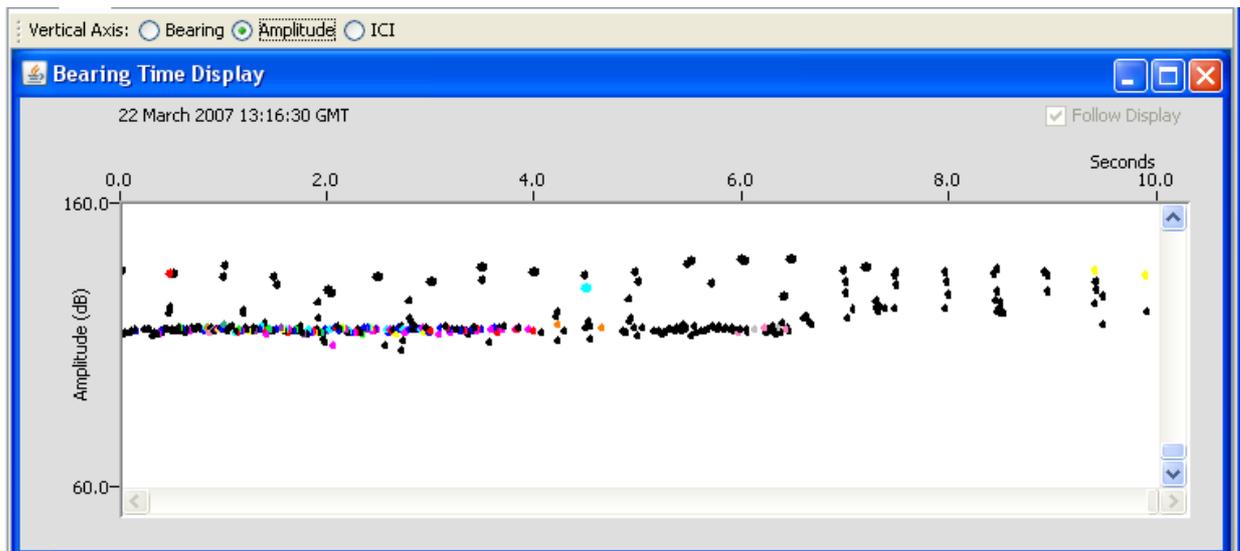


Figure 28. *Plotting click amplitude versus time*

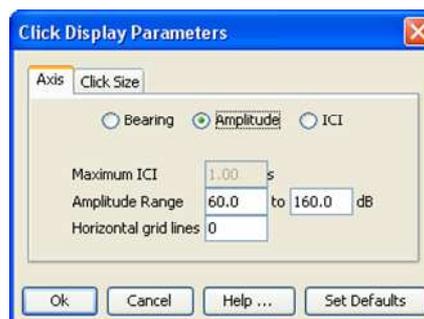


Figure 2960. *Adjusting the amplitude scaling for the Click Detector Amplitude display*

We will now adjust the click trigger threshold such that the Click Detector will not trigger for these lower amplitude clicks. Again select *Detection* ⇒ *Click Detector 1* ⇒ *Detection Parameters...* from the main menu. This time, increase the *Threshold* setting above the default setting of 10.0 dB. This will increase the level above the measured background which must be exceeded by clicks to become detection candidates.

Click **OK** and observe the Click Detector display. Wait for the algorithm to settle down with the new settings (up to 10 seconds).

If your setting is too high, then genuine clicks may not be exceeding the threshold and the click display will look rather sparse. If the setting is too low, you will still be seeing the interference clicks. Experiment with this setting until you are happy with the results.

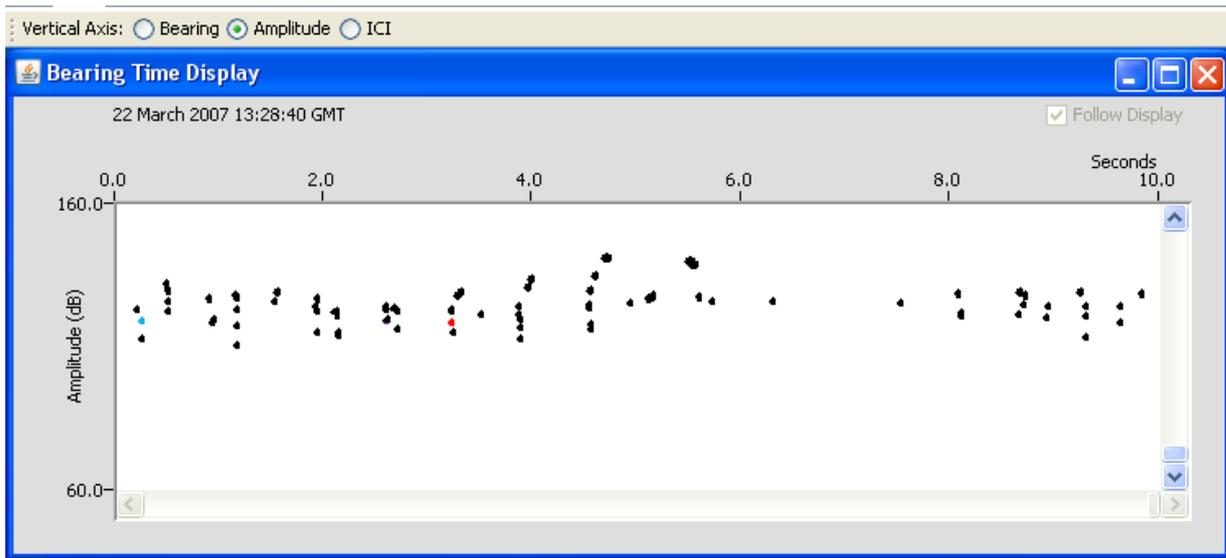


Figure 30. Results of adjusting Click Detector trigger threshold

Exercise 6. Energy sum detector

We are going to detect units in the humpback song by summing the “energy” in the 100-300 Hz frequency band. First load the sound file “humpback whale example.wav”, a recording of a singing humpback whale. On the menu, choose **Detection** ⇒ **FFT Parameters** (Fig. 31) and set the FFT length to 512 (samples) and the FFT Hop to 256 (samples). Close this dialog box.

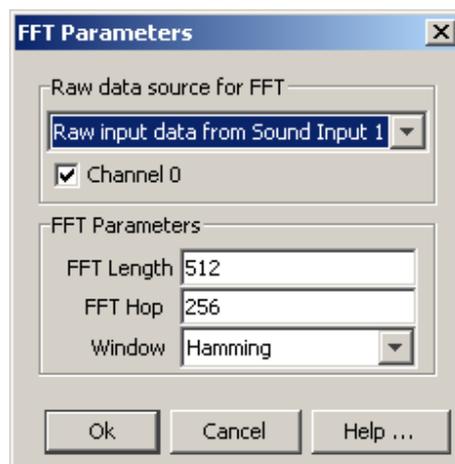


Figure 31. FFT parameter settings for the spectrogram correlation example.

Choose **File** ⇒ **Add Modules** ⇒ **Detectors** ⇒ **Ishmael Energy Sum** and give it the name **energy sum**. Next, choose **Detection** ⇒ **Energy Sum Settings** (Fig. 32). (Sometimes “Energy Sum Settings” doesn’t show up on the detection menu immediately. If this happens, you can get it either by quitting PAMGUARD and restarting it, or by opening **File** ⇒ **Show data model** and right-clicking on the red bar in the “**energy sum**” box.) Set the minimum frequency to 100 Hz, the maximum to 300 Hz, and uncheck the “Use log-scaled spectrogram” box if it’s checked. Also, set the vertical scale factor to 0.2 and the threshold to 1.5. The data source at the top should remain **Basic FFT**. Close this dialog box.

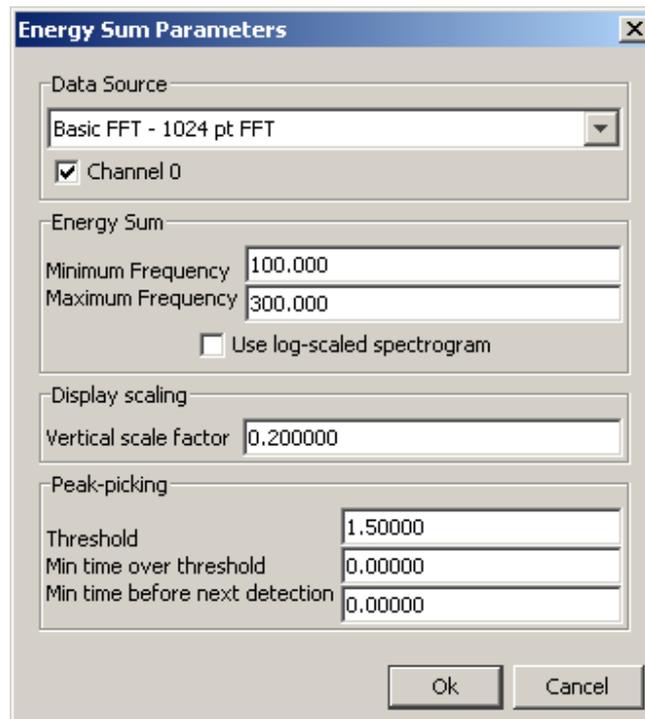


Figure 32. Parameter settings for the energy sum detector example.

Right-click on the spectrogram and choose "Settings". Select 1 panel and hit enter, then go over to the "Scales" tab (Fig. 32). There, choose a frequency range of 100-300 Hz by setting the min and max frequency, and set the amplitude range to 150-170 dB. The number of pixels per FFT will depend on the size of your display; try 3 to start.

Next, on the **Plugins** tab, make sure the **energy sum graphics** checkbox is checked. Click **Ok**.

Now you're ready for **Detection** ⇒ **Start** on the menu.

You should see humpback song units appear in the spectrogram (Fig. 33), and a detection function appear below the spectrogram. The detection function specifies, over time, the amount of energy in the frequency range you specified (100-300 Hz). The threshold you set will cause PAMGUARD to trigger a detection every time the detection function goes over this threshold.

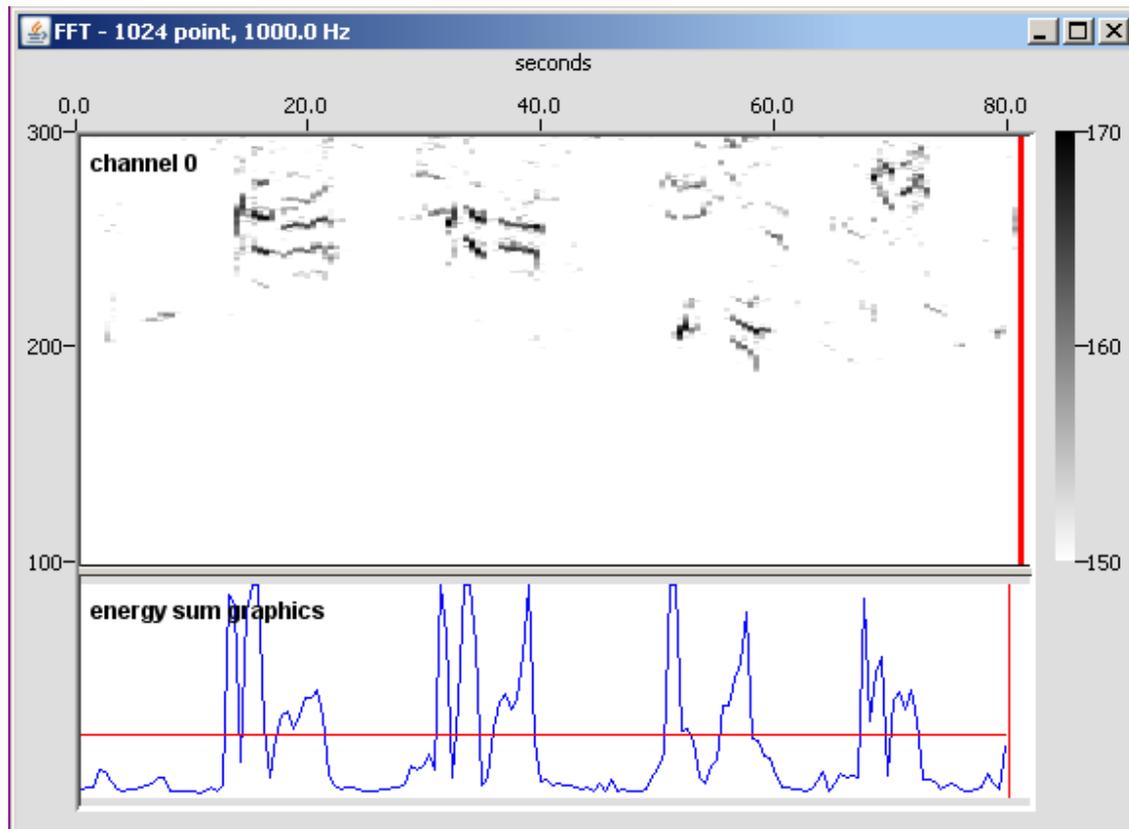


Figure 33. Example of the energy sum detector in operation.

Exercise 7. Spectrogram correlation detector

We are going to detect 'A' calls of Atlantic blue whales by using a spectrogram correlation detector. First load the sound file "blue whale-Atlantic.wav". On the menu, choose **Detection** \Rightarrow **FFT Parameters** (Fig. 34) and set the FFT length to 512 (samples), the FFT Hop to 256 (samples), and the window type to Hamming. Close this dialog box.

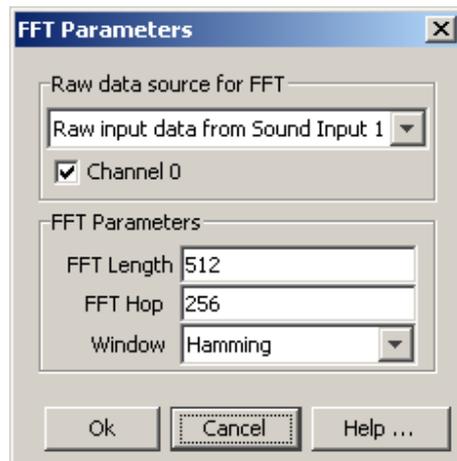


Figure 34. FFT settings for spectrogram correlation example.

Next, choose **File** ⇒ **Add Modules** ⇒ **Detectors** ⇒ **Ishmael spectrogram correlation** and give it the name **spectrogram correlation**. Then choose **Detection** ⇒ **Spectrogram Correlation Settings** (Fig. 35). (Sometimes "Spectrogram correlation Settings" doesn't show up on the detection menu immediately. If this happens, you can get it either by quitting PAMGUARD and restarting it, or by opening **File** ⇒ **Show data model** and right-clicking on the red bar in the **spectrogram correlation** box.) For segment 1, enter 0 for t0, 18 for f0, 10 for t1, and 17 for f1. For segment 2, enter 10 for t0, 17 for f0, 15 for t1, and 15 for f1. Leave the other segments blank, and use 1 Hz for the kernel width. Make the vertical scale factor 0.2. Close this dialog box.

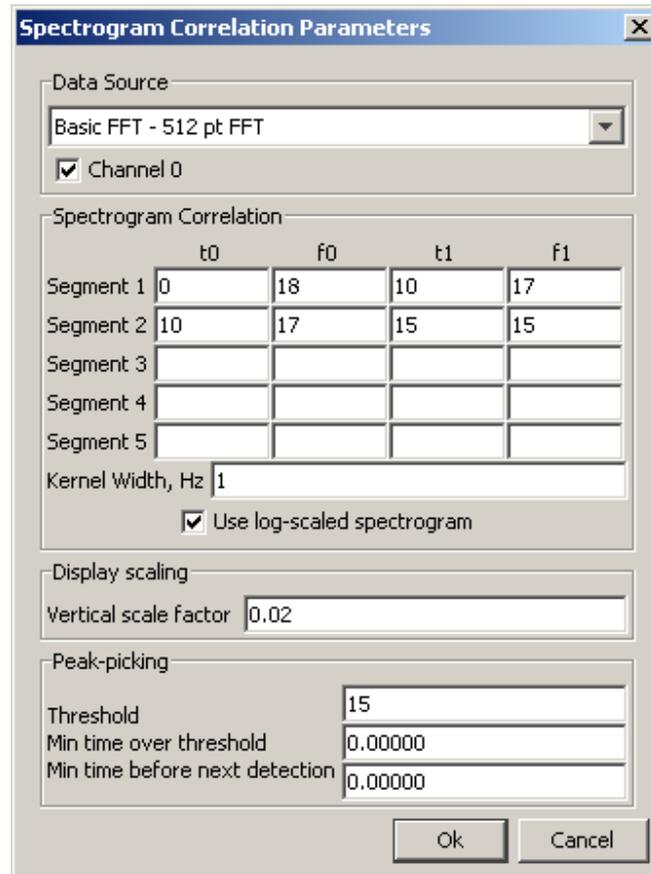


Figure 35. Parameter settings for the spectrogram correlation example.

Right-click on the spectrogram and choose **Settings**. Select 1 panel and hit enter, then go over to the **Scales** tab. There, choose a frequency range of 10-50 Hz by setting the min and max frequency, and set the amplitude range to 160-190 dB. The number of pixels per FFT will depend on the size of your display; try 3 to start. Next, on the **Plugins** tab, make sure the "spectrogram correlation graphics" checkbox is checked. Click **Ok**.

Now you're ready for **Detection->Start** on the menu. You should see blue whale calls appear in the spectrogram between about 15 and 20 Hz, and a detection function appear below the spectrogram. The detection function specifies, over time, the degree of match between the spectrogram and the frequency contour you've defined. By setting a threshold in this function, you cause PAMGUARD to trigger a detection every time the detection function goes over this threshold.

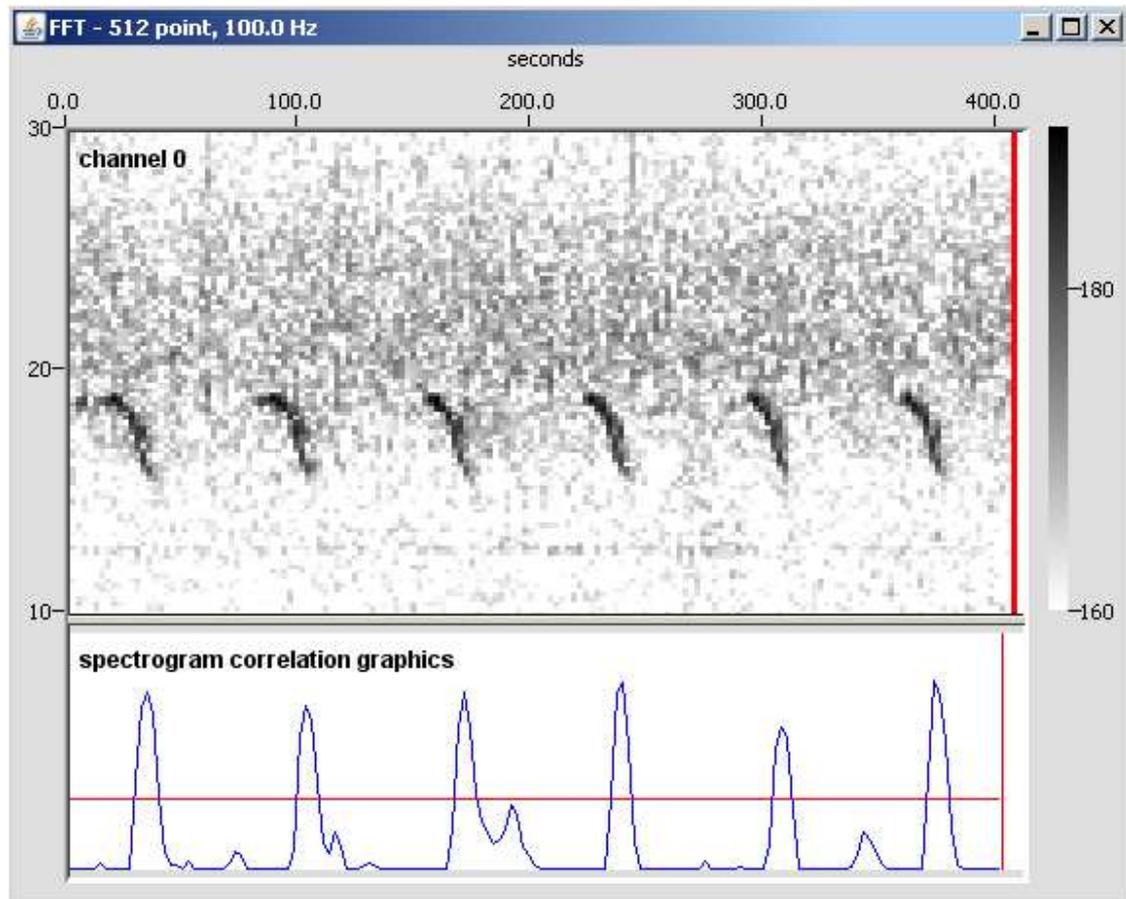


Figure 36. Example of blue whale call detection using spectrogram correlation.

Exercise 8. Matched filter detector

We are going to detect calls of northeast Pacific blue whales by using a matched filter detector. First load the sound file "blue whale-NE Pacific.wav". On the menu, choose **Detection** \Rightarrow **FFT Parameters** (Fig. 37) and set the FFT length to 1024 (samples) and the FFT Hop to 512 (samples), and the window type to Hamming. These parameters don't affect the matched filter, which operates on the time-domain sound signal, but they do make the spectrogram show the blue whale calls well. Click Ok to close the dialog box.

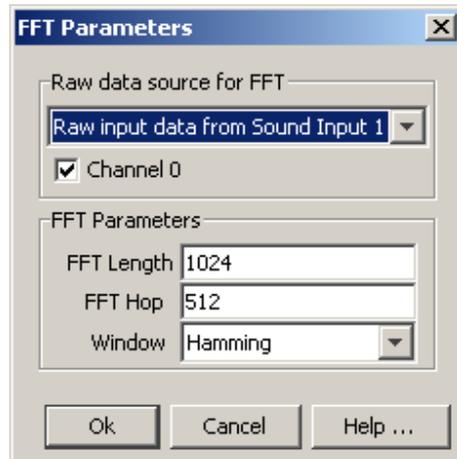


Figure 37. FFT parameters for the matched filter example.

Next, choose **File** ⇒ **Add Modules** ⇒ **Detectors** ⇒ **Ishmael matched filtering** and give it the name **matched filter**. Then choose **Detection** ⇒ **Matched Filter Settings** (Fig. 38). (Sometimes “Matched Filter Settings” doesn’t show up on the detection menu immediately. If this happens, you can get it either by quitting PAMGUARD and restarting it, or by opening **File** ⇒ **Show data model** and right-clicking on the red bar in the **matched filter** box.) In the “kernel sound file” area, click “Select another file” and choose the file “blueDemo-NEPac-kernel.wav”. Set the vertical scale factor (for the display) to 500 and the threshold to 0.0004. Click Ok to close the dialog box.

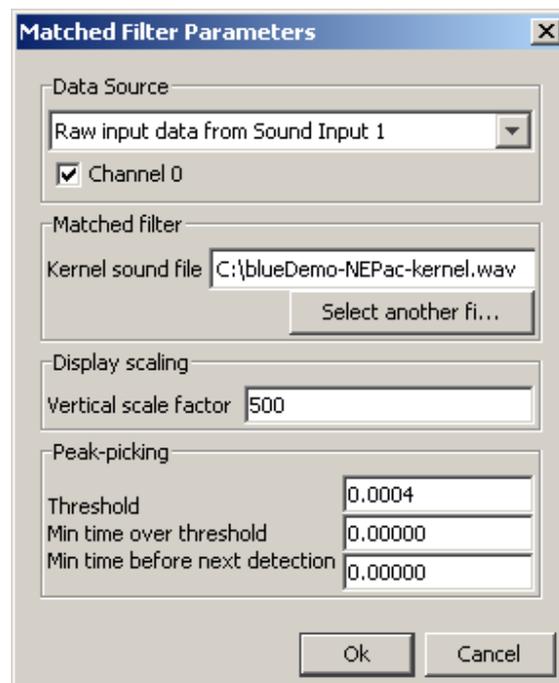


Figure 38. Parameter settings for the matched filter example.

Right-click on the spectrogram and choose **Settings**. Select 1 panel and hit enter, then go over to the **Scales** tab. There, choose a frequency range of 10-60 Hz by setting the min and max frequency, and set the amplitude range to 80-110 dB. The number of pixels per FFT will depend on the size of your display; try 1 to start. Next, on the **Plugins** tab, make sure the "matched filter graphics" checkbox is checked. Click Ok to close the dialog box.

Now you're ready for **Detection->Start** on the menu. You should see blue whale calls appear in the spectrogram, and a detection function appear below the spectrogram. The detection function specifies, over time, the degree of match between the spectrogram and the kernel sound you've chosen. You can set a threshold in this function and have PAMGUARD trigger a detection every time the detection function goes over this threshold.

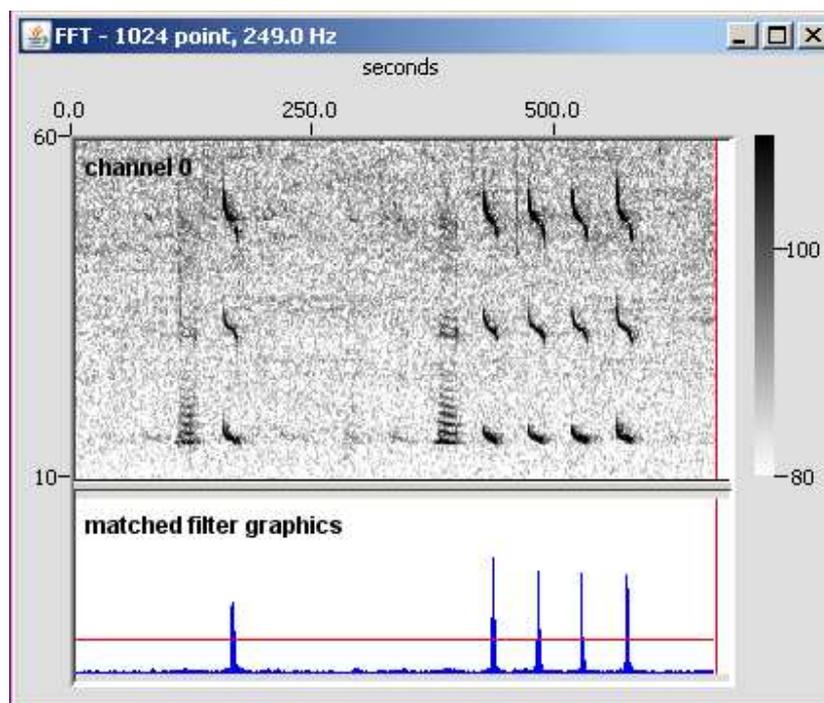


Figure 39. Example of a matched filter detecting calls of northeast Pacific blue whales.

Exercise 9. Ishmael "hyperbolic" localizer

You are going to locate a dolphin whistle using multiple hydrophones. First tell PAMGUARD the positions of your hydrophones via **File** ⇒ **Hydrophone array** (Fig. 40). Under "Array configuration" at the top, choose "Basic Linear Array",

and just below that choose "Static array". Next, using the "Add" and "Edit" buttons a bit farther down, set up 3 phones (numbered 0, 1, and 2) at these positions:

| <u>Phone Id</u> | <u>X</u> | <u>Y</u> | <u>depth</u> |
|-----------------|----------|----------|--------------|
| 0 | -100 | 0 | 5 |
| 1 | -20 | 200 | 5 |
| 2 | 150 | -50 | 5 |

The "Type" and "Bandwidth" fields don't matter here.

Under "channel configuration", make sure data channel 0 is connected to hydrophone 0, data channel 1 is connected to hydrophone 1, and data channel 2 is connected to hydrophone 2. (This is the default, so you probably won't have to do anything here.) Also, near the bottom of the dialog box, set the speed of sound to 1500 m/s. Click Ok.

Pamguard hydrophone array

Array Configuration

Basic Linear Array

Static array Towed array Locator: Straight / rigid Hydrophone

Hydrophone reference point

change

change

| Id | x | y | depth | Type | Bandwidth |
|----|--------|-------|-------|---------|--------------|
| 0 | -100.0 | 0.0 | 5.0 | Unknown | 0.2-20.0 kHz |
| 1 | -20.0 | 200.0 | 5.0 | Unknown | 0.2-20.0 kHz |
| 2 | 150.0 | -50.0 | 5.0 | Unknown | 0.2-20.0 kHz |

Add... Edit... Delete

Channel Configuration

Audio File C:\Dave\EclipseWorkspace\PamguardExamples-Dave\dolphin whistle @ 20,20.wav 32.0kHz, 3 channels

| Data Channel | Hydrophone | Gain | Range | Bandwidth |
|--------------|------------|---------|----------|--------------|
| 0 | 0 | 10.0 dB | 5.0 Vp-p | 0.0-20.0 kHz |
| 1 | 1 | 10.0 dB | 5.0 Vp-p | 0.0-20.0 kHz |
| 2 | 2 | 10.0 dB | 5.0 Vp-p | 0.0-20.0 kHz |

Change hydrophone ...

Environment

Speed of sound 1500.0 m/s

Figure 40. Key parts of the hydrophone array configuration for the localization example.

Next load the sound file "dolphin whistle @ 20,20.wav". On the menu, choose **Detection** ⇒ **FFT Parameters** (Fig. 41) and set the FFT length to 512 (samples) and the FFT Hop to 100 (samples), and the window type to Hamming. Click Ok to close the dialog box.

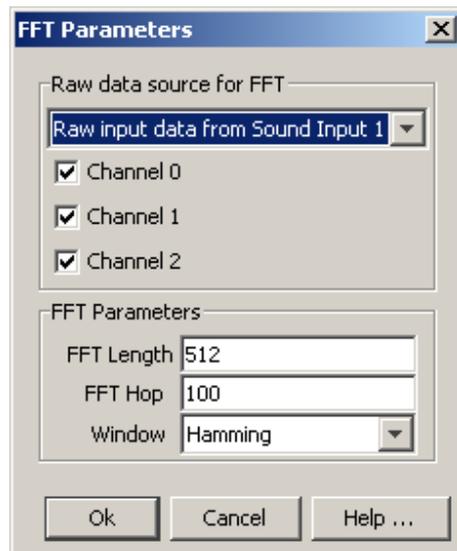


Figure 41. FFT parameters for the localization example.

Next choose **File** ⇒ **Add Modules** ⇒ **Detectors** ⇒ **Ishmael Locator Results** and give it the name "Ishmael locator". Click Ok. Then open the **Detection** ⇒ **Ishmael Location Settings** dialog box and make sure the number of dimensions is 2. Click Ok.

Next, right-click on the spectrogram and choose **Settings**. Under "Number of Panels", type 3 and press the Enter key on your keyboard. On the **Scales** tab (Fig. 42, left), set the frequency range to 5000-16000 Hz with the min and max frequency and the amplitude range to 130-160 dB. The number of pixels per FFT will depend on the size of your display; try 2 to start. Next click the **Mark Observers** tab (Fig. 42, right) and make sure the "Ishmael Locator" box is checked. Click Ok to close the dialog box.

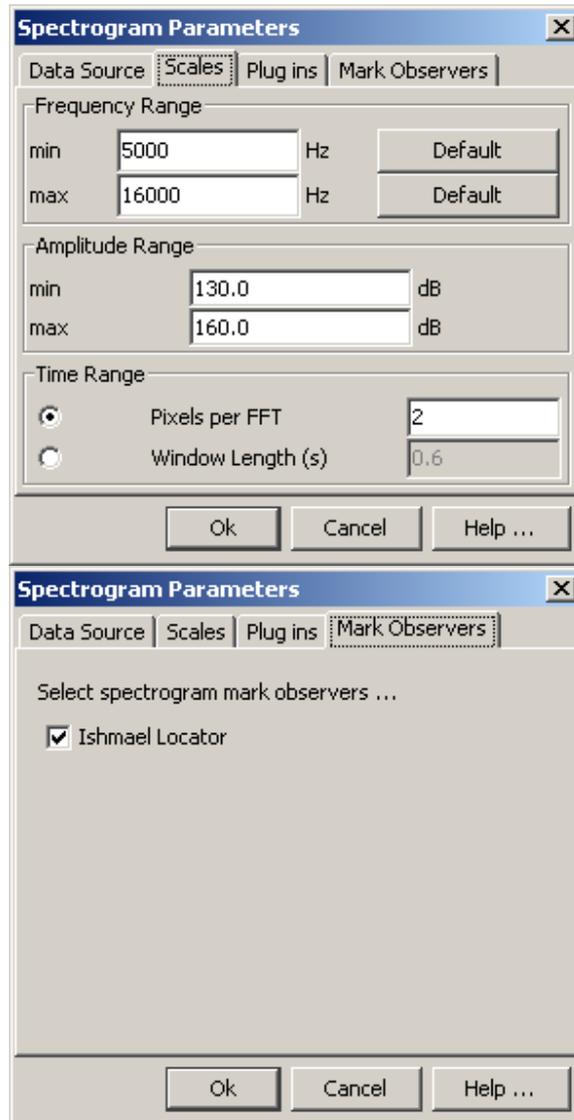


Figure 42. Spectrogram parameter settings for the Ishmael localization example: (left) **Scales** tab, (right) **Mark Observers** tab.

Now you can do **Detection** ⇒ **Start**. You should see a dolphin whistle displayed in 3 spectrogram panels. To locate a call (the whistle in this case), simply use the mouse to draw a box around the call or a part of the call (Fig. 43). The location will be computed and installed in PAMGUARD's set of located calls.

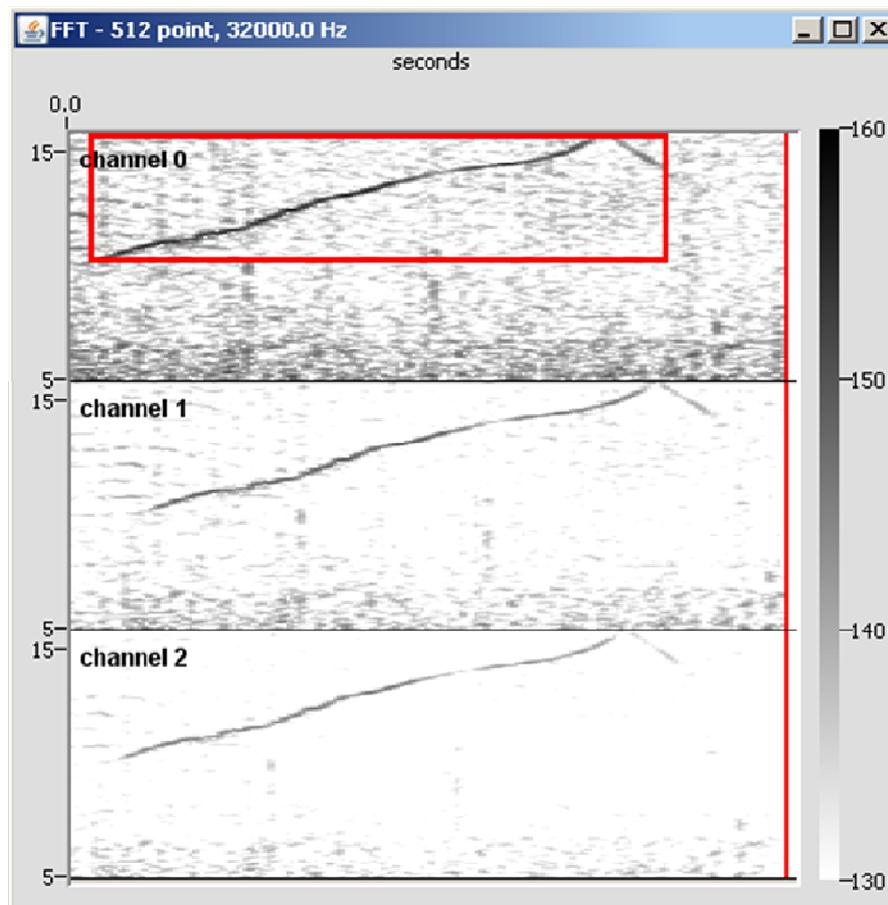


Figure 43. Localizing a dolphin whistle; the red box indicates the selection to localize.

Exercise 10. Running the PAMGUARD 3-D module

These instructions guide a user on how to configure the "3-D" module in PAMGUARD for estimating range and depth of a source using a long-aperture towed array. At the time of this writing the module has only been tested with synthetic WAV files and not real-time input, and only with a Static and Threading Hydrophone Locator.

The instructions begin with an overview of the data model arrangement for the model, the preliminary configuration for all upstream modules, and a description of the options for the 3-D tracking module itself. Two configuration files are available to demonstrate and confirm basic operability:

- (1) Sim3D_wavinput_noGPS_ICItest_Feb2008.psf, and
- (2) Sim3D_wavinput_noGPS_ICItest_twoanimals_Feb2008.psf.

The first configuration demonstrates the use of simulated data from a single source, while the latter loads simulated data for two sources, and which

requires an analysis of the inter-click interval (ICI) to correctly assign each pulsive detection across the array.

Preliminaries

If either configuration file is loaded, and Data Model view selected, one will see a basic setup for the 3-D tracking:

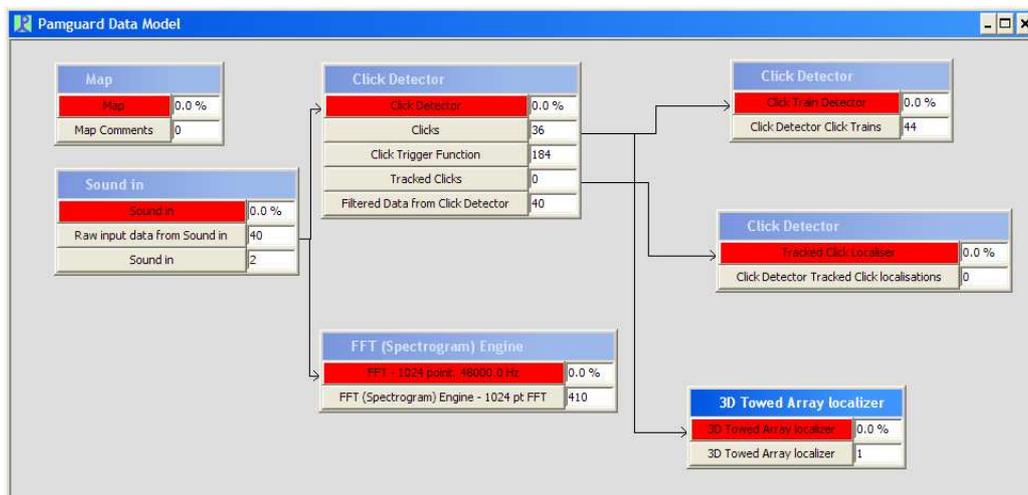


Figure 44: Data Model view of demo 3D tracking configuration file *Sim3D_wavinput_noGPS_ICItest_Feb2008.psf*

At a minimum the 3D Towed Array module requires an upstream Data acquisition module and a Click module (labeled Sound in and Click Detector in Figure 44). At present the 3D module is configured to only work with the Click Detector, and not any Ishmael detectors. A typical array configuration is illustrated in Figure 45, which appears under the Hydrophone display. At present only two pairs of hydrophones (4 phones total) are used in the module.

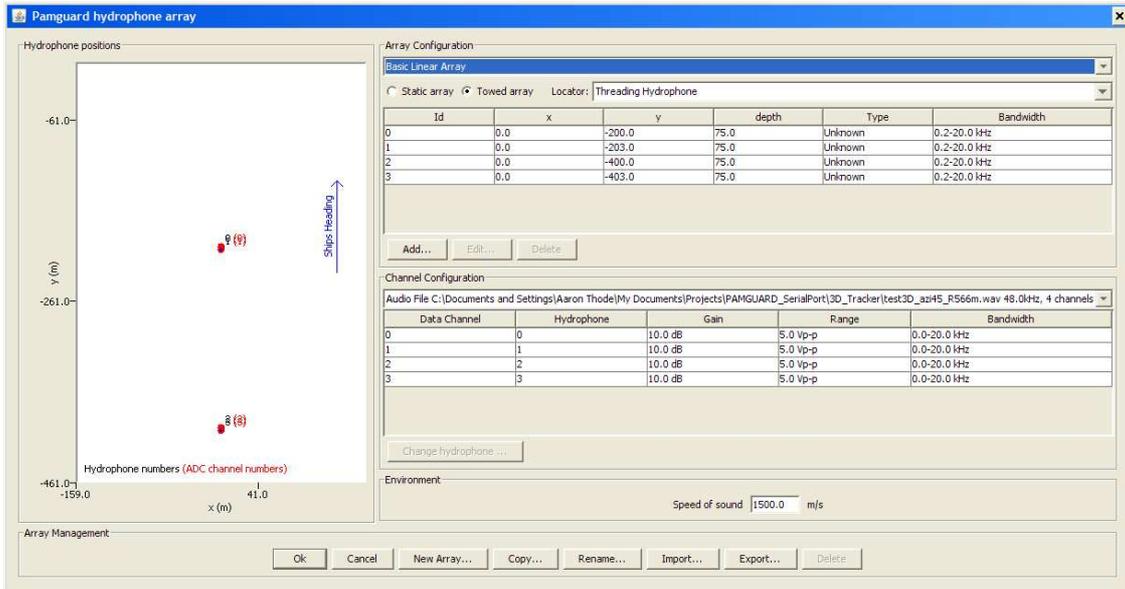


Figure 45: Hydrophone array configuration

Figure 46 shows the configuration menu for the Audio Data Acquisition module (SoundIn in Figure 44) should be configured to select a synthetic wav file, either test3D_azi45_R566m.wav for a single simulated source, or test3D_azi45_R566m_multanimals.wav for two sources. The wav file has 4 channels, with the hydrophone closest to the tow vessel being assigned the lowest channel.

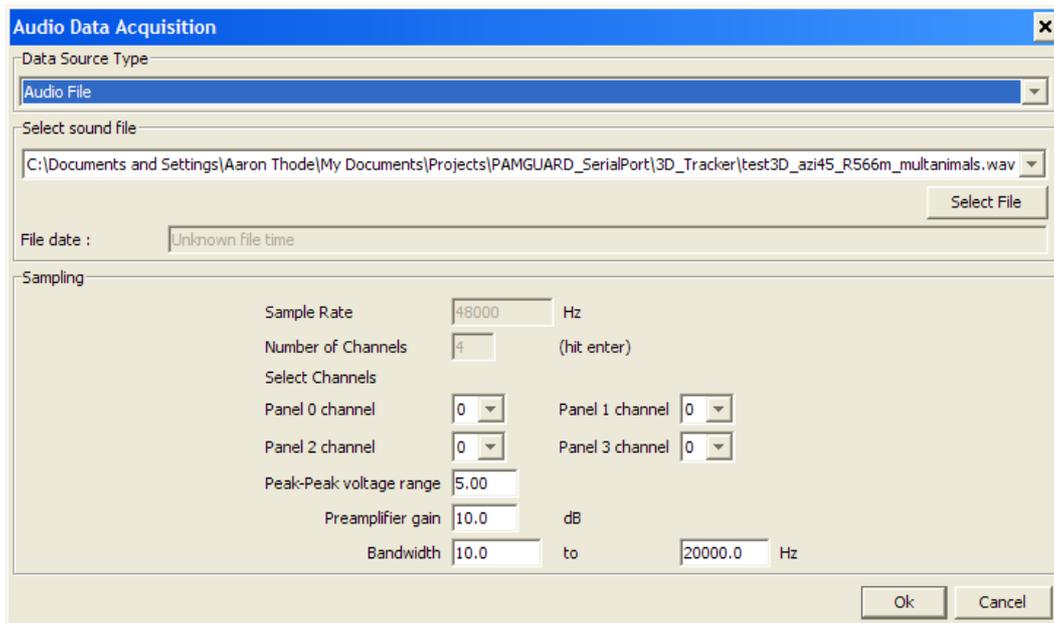


Figure 46: Settings window for importing simulated WAV data. The exact file path would need to be adjusted before starting the program.

The Click Detection module has many options, but only the following menu needs to be changed from the default (Figure 47):

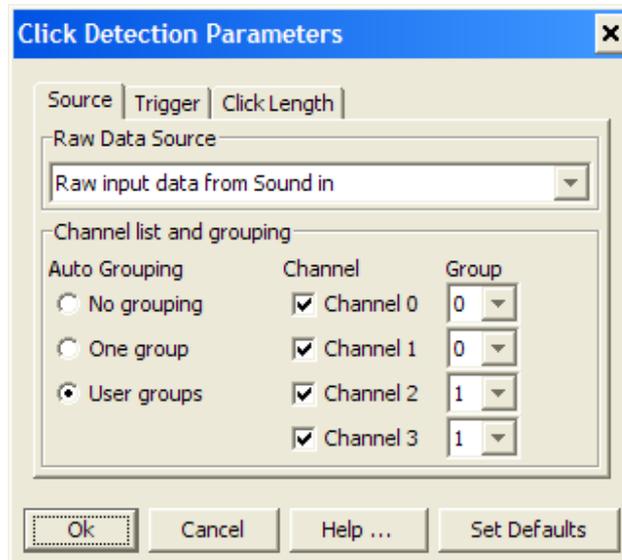


Figure 47: Detection Parameters Settings Dialog for Click Detector module, arranged for two two-element subarrays. All other parameters are default.

The "Source" tab arrangement shown here assigns the first two channels to Group 0 (thus generating a forward subarray bearing) and the last two channels to Group 1, (generating the rear subarray bearing).

3-D Towed Array Settings

The module assumes that the separation between the two pairs of hydrophones in the array is great enough that an ambiguity may exist as to how to match a particular detection on the rear subarray with candidate detections on the forward subarray. This ambiguity is more likely when multiple acoustically active animals are present. In general, the module also assumes that the array elements are deep enough that surface-reflected paths from pulsive sounds can be cleanly separated in time from the direct paths. The time delay between a direct and surface-reflected path will be defined as an "echo time" for the rest of this discussion. The echo time of a detection, along with the acoustic bearing of the detection, provide a set of features that permit identification of the signal between subarrays.

To match the appropriate signals the program builds a queue of detections for each subarray, and assigns an inter-click interval (ICI) to each detected click. The ICI of a new detection is computed by searching the queue for a detection that shares a similar bearing and "echo time" to the present detection. Once the ICI is assigned, the program can build a "linked ICI" list that provides the ICI of the N previous detections that share similar features to the current detection.

This ICI list can then be compared to lists on the forward array to locate the corresponding detection on the forward array.

Figure 48 shows the Settings menu for the 3-D Towed Array module. The Detector source should be set to 'Clicks', and all four channels selected. The selection parameters are as follows:

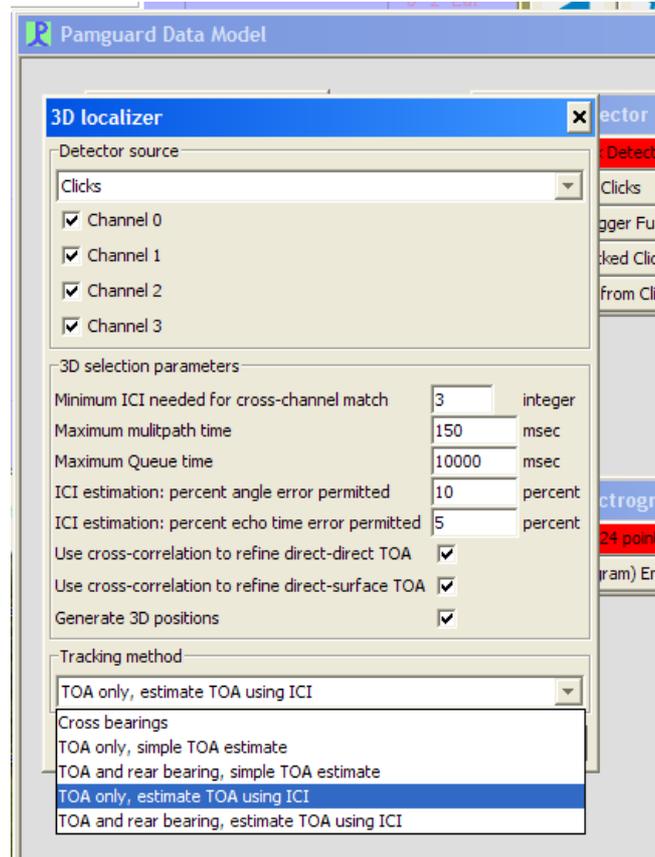


Figure 48: Settings window for 3D localizer, with "Tracking method" pull-down menu extended.

Minimum ICI needed: This parameter N defines the number of ICI to incorporate into a list for cross-array comparison. In general the larger the number the less likely a mistake will be made in assigning detections between forward and rear subarrays.

Maximum multipath time: Surface-reflected multipath is flagged by the fact it must arrive only a short time after a direct path. This number sets the maximum "echo time" that will be assigned by the system. Generally, this should be set to twice the hydrophone depth divided by the sound speed.

Maximum Queue Time: Sets the amount of time a particular detection will be retained by the subarray queue, to be available for ICI matching.

ICI estimation: percent angle error: The percent difference between the bearing of a new detection and a candidate detection must be less than this number, in order to use the candidate detection to assign an ICI to the new detection.

ICI estimation: echo time error: Same as above, except the error percentage is computed for the echo time. If an echo time is present, then the bearing and echo time are weighted equally in the decision to link two detections together. If no echo time is present, then only the bearing error is used.

Use cross-correction to refine TOA: The echo times and relative arrival times between subarrays are computed from the detection times provided by a Detection Data Unit. To achieve more precision the raw data between detections can be cross-correlated to increase the precision of the estimate of the relative arrival time. At present only the direct-surface TOA (echo time) feature is enabled.

Generate 3D positions: If not checked, no 2-D or 3-D positions are computed. This is a debug feature to permit the user to set up all other modules before attempting to localize.

Tracking methods

A pull-down menu permits the user to select five different tracking combinations.

Cross bearings: The simplest tracking algorithm simply uses the ICI analysis to generate bearings from the forward and rear subarray, and thus estimate slant range. This slant range is plotted as a horizontal range on the map.

TOA only, simple TOA estimate: If echo times are available on both subarrays, then a track can be computed without having to rely on acoustic bearings. This "TOA only" method is advantageous for situations where the sources are directly ahead or behind the tow vessel, and where the array cable inclination is not measured and cannot be estimated accurately. The "simple TOA estimate" simply means that a detection on the rear subarray is matched to a detection on the forward subarray by simply selecting the detections that have the smallest relative arrival times with respect to each other. For a single animal the "simple TOA estimate" should work more quickly than a full ICI analysis.

TOA and rear bearing, simple TOA estimate: Same as previous, except that the rear subarray bearing, instead of the forward subarray echo time, is used to compute position. Best used for sources off the beam of the vessel and situations where the echo time is not available on the forward subarray. Typical situations where this happens include noise masking from noisy vessels, and shallow hydrophones on the forward subarray.

TOA only, estimate TOA using ICI Same as TOA only option previously discussed, except that ICI patterns are used to match detections on the subarrays. Should be used when multiple animals are present.

TOA and rear bearing, estimate TOA using ICI: Same as TOA and rear bearing option discussed above, except that that ICI patterns are used to match detections on the subarrays. Should be used when multiple animals are present.

Any 3-D tracking method (TOA option in the Tracking method pull-down menu) will generate a Map image similar to that shown in Figure 49. The depth of the source is indicated by the size of the circle and by text printed next to the circle. The "Cross bearing" output is demonstrated in Figure 50.

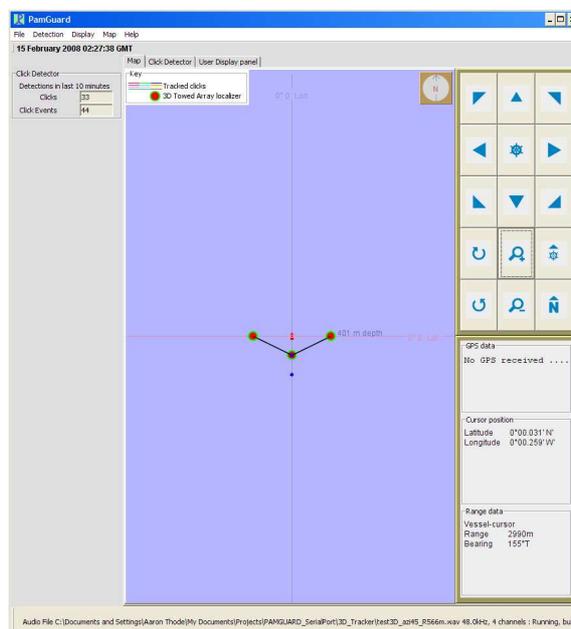


Figure 49: Graphics output showing location of a 400 m deep source 400 m off the beam of the towing vessel, located using two sets of hydrophones towed 200 m and 400 m behind the vessel, respectively. Any of the TOA options in the Tracking method pull down menu will generate this result.

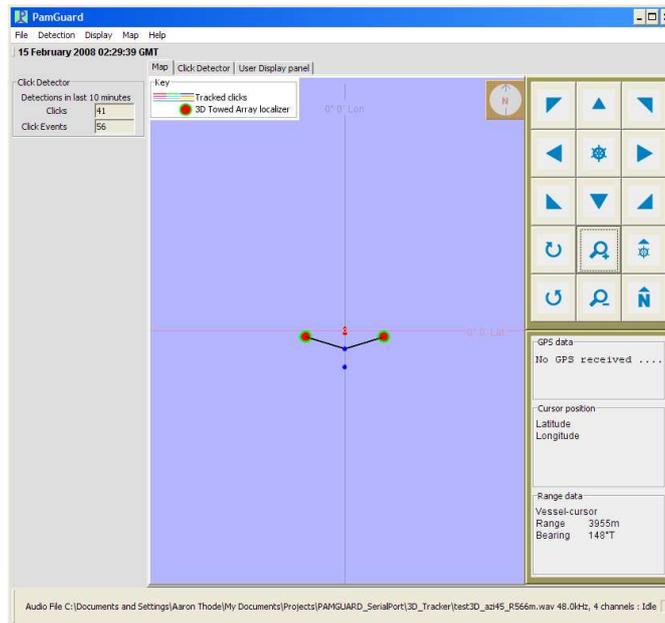


Figure 50: Output using the 'Cross bearings' feature, which ignores multipath information and simply tries to cross bearings.

Examples from the second configuration file,
 Sim3D_wavinput_noGPS_ICIttest_twoanimals_Feb2008.psf:

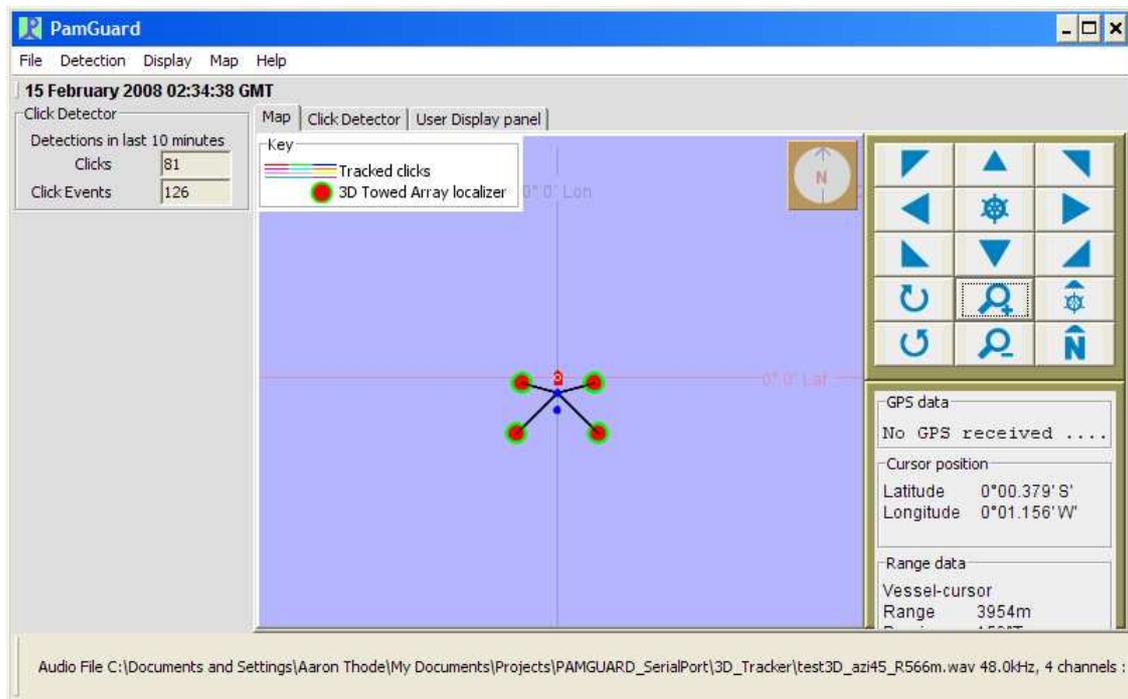


Figure 51: Output of CrossBearings tracker using test3D_azi45_R566m_multanimals.wav

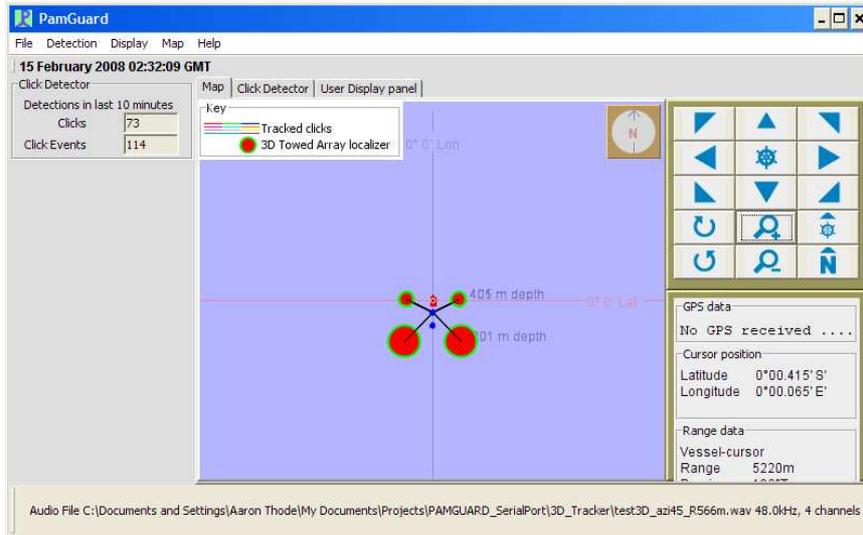


Figure 52: Output of a 3D tracker for two animals using test3D_azi45_R566m_multanimals.wav.

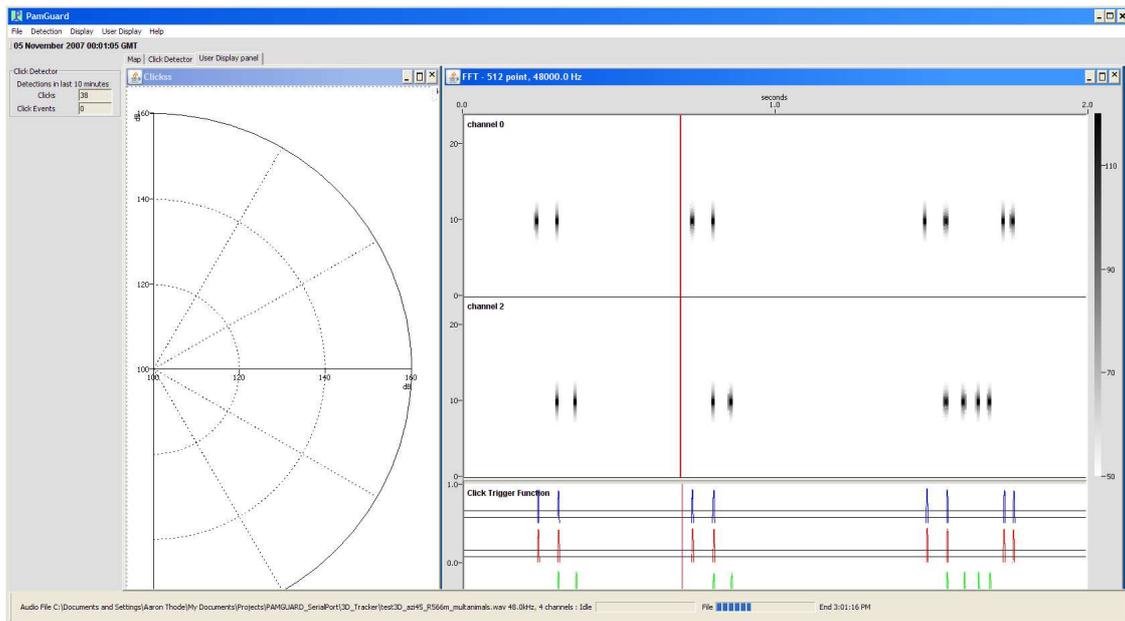


Figure 53: Spectrogram and radar plot of simulated 3-D sources on PAMGUARD.

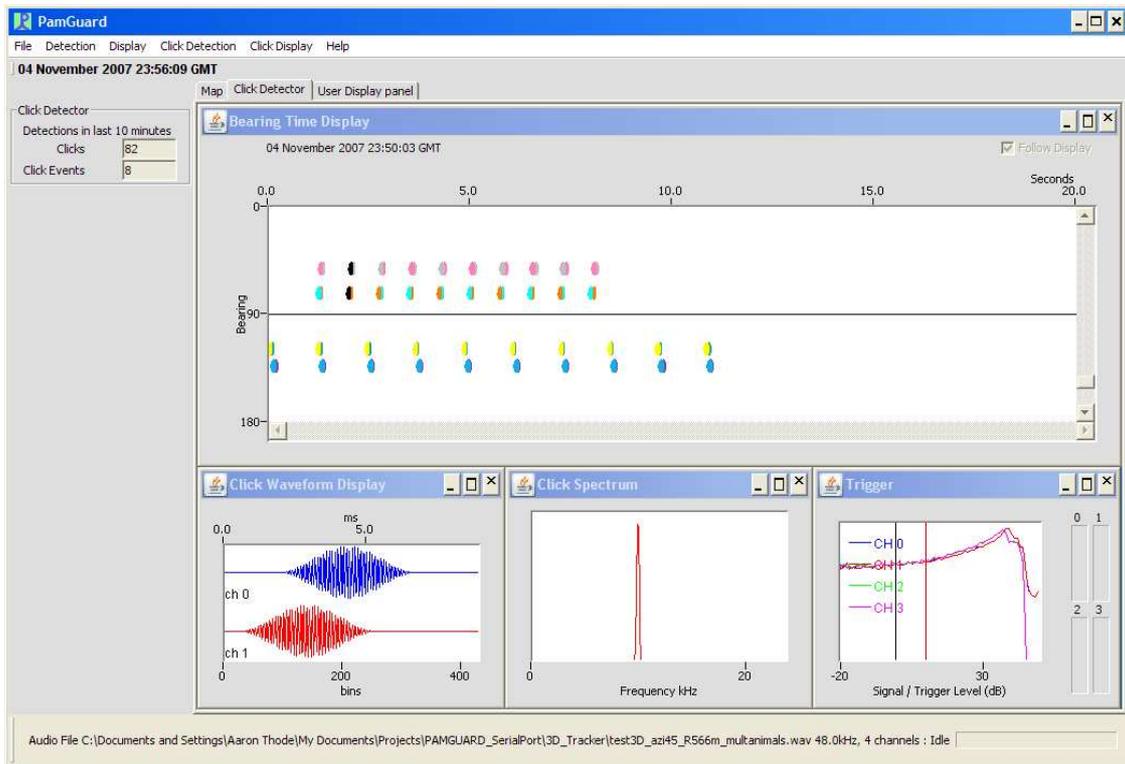


Figure 54: Simulated 3D tracking data for multiple animals as it appears in Click Detector view.

Don't stop now!

This tutorial has given you a brief introduction to setting up and using PAMGUARD. There are more PAMGUARD plug-ins and features for you to try out.

Here's some ideas to get you going:

- Set up a Sound Recorder module and trigger recordings from click trains.
- Use a decimator to provide lower sample rate data to a second spectrogram.
- Display smoothed spectral data on a spectrogram.

Try these out with the other example sound data files.

The PAMGUARD developers are keen to know what you think of the software, so any comments (good or bad!) are very welcome. Also, if you think this tutorial can be improved in any way, please let us know at feedback@pamguard.org.

Thanks for taking the time to do this tutorial. Hopefully you've enjoyed this and learned some more about PAMGUARD. You can contact us and keep up with PAMGUARD news through visiting www.pamguard.org.

Appendix B. Developer Tutorial (DG)

1. Introduction

Pamguard is an open source project, and as such, developers are welcome to look at any part of the code, comment on it, modify it, and hopefully improve it !

The purpose of this session is to concentrate on the development of new detector modules using the existing Pamguard infrastructure. We assume that participants in this session have some programming experience. If you're already a JAVA expert, you'll find it easy. If you've experience of C++, then JAVA will look pretty familiar, and you'll probably be pleased at how easy it is in comparison. If you're new to object orientated programming, then hang on in there, we hope you'll still get something out of this session !

This tutorial is laid out in two principal parts

1. An overview of the Pamguard program structure and how the different modules relate to one another and pass data between them.
2. A walk through example of a really simple detector, which has been written primarily to demonstrate and show off the Pamguard core structure and components rather than to be the ultimate in signal detection.

1.1. Documentation

The Pamguard source code is extensively documented using the Javadoc tool from Sun Microsystems. This documentation may be viewed as comments within the source code, may appear as pop-up help from within some development environments (such as Eclipse, which you will be using during the tutorial) and is also available on the web at <http://www.pamguard.org/devdocs/index.html>.

1.2. Eclipse

The Eclipse integrated development environment (IDE) (www.eclipse.org) and the Java software development kit (Sun's JDK 1.5) are already installed on the machines you'll be using. When you launch Eclipse, it will automatically take you to the Pamguard project. From the Window menu, select 'Open Perspective' and the 'Java Browsing' and your display should look something like this:

Working along the top of the top of the window, you'll see that the Pamssoft2006 'Project' is divided into a number of 'Packages', each of which contains a number of 'Types' – Java classes or interfaces, each of which contains 'Members' – functions of variables. Take some time to browse around, opening and closing the various 'Types' in the main window.

To run Pamguard from within the Eclipse IDE, press the run button (solid green triangle) in the toolbar. To debug Pamguard, press the debug button (icon with a picture of a little bug) just to the left of the green run button.

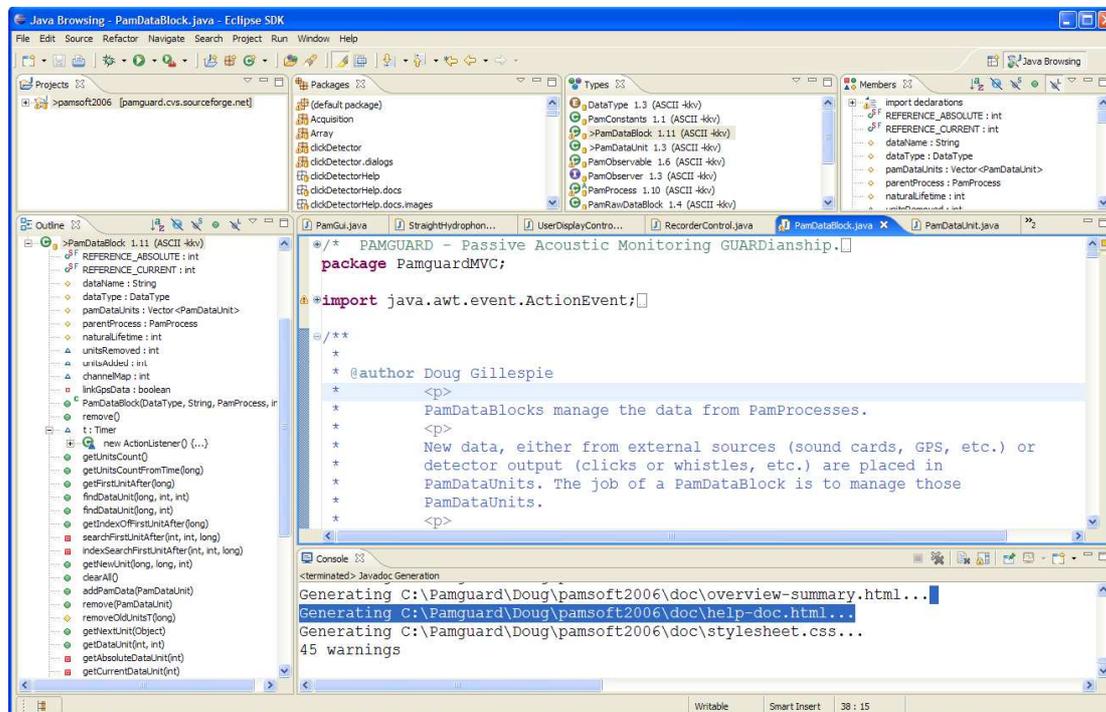


Figure 1. The Eclipse IDE.

When writing code within Eclipse, there is an excellent ‘auto complete’ function. Start typing a command and press ‘Ctrl’ and the space bar together and a list of possible commands / classes should appear. To find out what’s available in the class you’re working on, try typing the word this. (to refer to the current object) and holding pressing ‘Ctrl-space’ and see what’s there ! Using auto complete will not only tell you what commands are available, but will spell them correctly for you (Java is a case sensitive language) and it will also insert any ‘import’s’ you need in order to access classes from other packages (Java imports are similar to C header files).

Hover the mouse over almost anything and you’ll probably get a pop-up telling you all about it.

Eclipse is very good at telling you if something is wrong. Any code that will not run will be underlined and the type / package / project will be highlighted with a little cross and a little red marker near the scroll bar so you can sort out errors in your code before you even try to compile and run it.

A function you may find useful as you try to understand the code is `System.out.println(any object)` which will print the output (or more specifically, the `objects toString()` member function) to the Eclipse console.

To add breakpoints for the debugger, right click in the margin to the left of the line you wish to pause at.

If you implement a Java interface, don't type the names of all the functions yourself ! Just right click on the java file, select 'Source, then 'override/implement methods...' and you'll get a pop up box of possible functions with the ones you **must** implement as part of the interface already selected.

That's all you need to know about Eclipse for the time being.

2. Pamguard Structure (a brief Introduction)

2.1.Pamguard Data flow.

Most data within Pamguard are handled with `PamDataBlocks` and `PamDataUnits`. Check out the Javadoc help for these classes, starting at <http://www.pamguard.org/devdocs/PamguardMVC/PamDataBlock.html> or by looking at the source code, which you'll find in the `PamguardMVC` package.

2.2.Pamguard plug-in packages

Pamguard consists of a number of plug in packages each of which may or may not process data, may or may not have a display and may or may not have some user controls or configurable options. Every plug in inherits it's basic behaviour from a class `PamControlledUnit`. Check out the java doc for this class at <http://www.pamguard.org/devdocs/PamController/PamControlledUnit.html> and follow the link the 'how to make Pamguard plug-ins' overview.

A Pamguard plug in should be able to do absolutely anything you want, including drawing on the map, writing to the database, drawing over spectrogram displays, etc. without having to modify the map package, without having to modify the database package, without having to modify the spectrogram, etc. All the programmer has to do is to make a class based on `PamControlledUnit` and add one or two lines to the `PamModel` class to let Pamguard know that's it's available and everything else will be automatic.

3. The Example Package

An example plug in package 'WorkshopDemo' has been written specifically to demonstrate how to use `PamDataBlocks`, `PamDataUnits` and main Pamguard display

features. It consists of a relatively simple detector that runs and ‘in band energy detector’ on spectrogram data. A measure of mean background noise in a given energy band is made as new data arrive. The instantaneous energy in that band is compared to the background and a detection is made whenever the energy / background ratio goes over threshold and then falls back down again.

The signal to noise ratio of the detector may be displayed as a plug in window on the bottom of spectrogram displays. If a detection is made, it may be displayed as a coloured rectangle overlaid on top of the spectrogram displays and as a symbol on the map display.

3.1. Running the detector

You should already be familiar with how to use Pamguard. Create an instance of the ‘Workshop Demo’ detector from the File / Add modules ...’ menu. If necessary, you will be prompted to create an FFT data block to convert raw audio data into a spectrogram. You will also need at least one ‘User Display Panel’ on which you should create a spectrogram display.

To view the signal to noise ratio display, right click on the spectrogram and select ‘Settings...’ go to the ‘Plug ins’ tab and check that ‘Workshop demo detector’ is selected. The panel should then be visible at the bottom of the spectrogram display.

To view detections as overlays on the spectrogram, again right click on the spectrogram panel and select your detector from the pop-up menu.

Play a sound file back through the PC sound system, as per the user demonstration, and start Pamguard detection from the ‘Detection / Start’ menu.

Detection parameters may be adjusted from the ‘Detection / Workshop demo detector parameters’ menu.

3.2.A Walk through the code.

The example code consists of seven main Java classes and two extra lines in the PamModel class. These are listed in the order:

| | |
|-----------------------------|--|
| WorkshopController | is the main plug in class controlling the detector |
| WorkshopProcess | is the main detection process (does the work) |
| WorkshopProcessParameters | contains parameters controlling the detection process |
| WorkshopParametersDialog | is a dialog box used to set detection parameters |
| WorkshopOverlayGraphics | provides graphics functions for displaying detections on top of spectrogram displays |
| WorkshopPlugInPanelProvider | provides plug in panels for the bottom of spectrogram displays. |
| WorkshopSQLLogging | provides functionality for logging detections to the |

| | |
|--|--------------------|
| | Pamguard database. |
|--|--------------------|

The two lines added to PamModel are

```
mi = PamModuleInfo.registerControlledUnit("WorkshopDemo.WorkshopController",
    "Workshop Demo Detector");
and
mi.addDependency(new PamDependency(DataType.FFT, "fftManager.PamFFTControl"));
```

The first of which tells the Pamguard Model that this detector is available. The second line tells Pamguard that this detector requires FFT data and that the preferred source of FFT data would be an instance of `fftManager.PamFFTControl`.

3.2.1. WorkshopController

This is a subclass of `PamControlledUnit` (see Javadoc)

It creates instances of the `WorkshopDetectionParameters`, the `WorkshopProcess` and the `WorkshopPluginPanelProvider`.

It overrides the `PamControlledUnit` function `NotifyModelChanged` to make the detector aware when other modules are added or removed from Pamguard.

It overrides the `PamControlledUnit` functions `createDisplayMenu()` and `createDetectionMenu()` in order to set up menu commands for adjusting detection and display parameters. These menu items will automatically be incorporated into Pamguards main Detection and Display menus.

It implements the interface `PamSettings` and registers with the `PamSettingsManager` so that the `WorkshopDetectionParameters` are stored along with other Pamguard configuration settings when Pamguard is closed and restarted.

3.2.2. WorkshopProcess

`WorkshopProcess` does the actual work of making detections. Since the detector may be required to operate on several channels simultaneously, some of the functions are contained within an inner class `ChannelDetector`.

`WorkshopProcess` is a sub class of `PamProcess`. `PamProcess` already implements the `PamObserver` interface, enabling it to listen out for new `PamDataUnits`.

`WorkshopProcess` only has to override the `newData()` function to get these notifications from the `PamProcess` superclass. When new data arrive, the channel number of the `PamDataUnit` is examined, and the data passed on to the appropriate `ChannelDetector`.

`WorkshopProcess` uses three different `PamDataBlocks`, one source data block and two output data blocks.

The source dataBlock, fftDataSource, is the PamDataBlock the detector will subscribe to (become an observer of) in order to get notifications when new FFT data units are created.

Two output data blocks are created:

1. outputDataBlock will contain detections, (if and when they occur).
2. backgroundDataBlock will contain information required by the spectrogram plug in panels, if any are ever created.

Note that outputDataBlock is registered with the Pamguard system using the addOutputDataBlock() function. This will enable other Pamguard modules, such as the map and spectrogram display, to find the data block in the system and update their own options menus and dialogs accordingly. The backgroundDataBlock is only used internally within this module, so there is no need to tell the rest of Pamguard about it.

Two additional functionalities are added to outputDataBlock: setOverlayDraw uses the WorkshopOverlayGraphics class to provide drawing functionality for both the map and the spectrogram displays. setLogging uses the WorkshopSQLLogging class to provide functionality for logging data to the Pamguard database.

prepareProcess is called whenever the Pamguard model changes (i.e. modules are added or removed) and whenever detection parameters are adjusted in WorkshopController. It's here that the WorkshopProcess subscribes to the correct data source and sets up the individual channel detectors.

PamStart is called by the Pamguard system just before detection starts. It doesn't do much apart from reset some of the detection flags in each ChannelDetector. Real detection will start when the first FFT datablock arrives.

ChannelDetector inner class

This does the actual work of making detections on an individual channel. Once a detection is made, it uses the outputDataBlock to create new PamDataUnits and send them off to the Pamguard system. *Note that unless the database is active or one of the Pamguard displays is using this data in some way absolutely nothing will happen to this data and it will very likely get deleted about a second later !*

3.2.3. WorkshopProcessParameters

Is a simple set of parameters controlling detection and display for this detector. They are all put into one class like this so that the PamguardSettingsManager can store them easily and efficiently in a serialised binary file (the ones you select when you start up Pamguard). Any class that will be used by the settings manager must implement the Serializable interface. This one also implements Cloneable so that it's easy to copy.

3.2.4. WorkshopParametersDialog

Is a dialog for setting workshop parameters. It's called from the WorkshopController menu ActionListener. It uses the PamDialog class to implement some standard

functionality such as the SourcePanel class which provides a quick and easy way of generating a drop down list of available data sources and channel numbers. This is just an example of how dialogs are made – you can do it any other way you want depending how much you like the various Java LayoutManagers.

3.2.5. WorkshopOverlayGraphics

Implements PanelOverlayDraw and allows detections to be drawn on the map and on spectrogram displays. The Pamguard system can find all the different data blocks, by working through lists of ControlledUnits, PamProcesses and output data blocks. Our output data block has had an overlay graphics member set. Different displays within Pamguard will call WorkshopOverlayGraphics. CanDraw(GeneralProjector) to see if this particular implementation of PanelOverlayDraw knows how to draw on those particular displays. This depends on the axis types set in the GeneralProjector, which are Latitude and Longitude for the map and Time and Frequency for the spectrogram. The projector will turn Latitude and Longitude or Time and Frequency into screen coordinates without the WorkshopOverlayGraphics class having to know anything at all about spectrogram scales, map orientation or scales, etc.

If the map (or spectrogram) is set to draw Workshop Demos DataUnit's, then the map (or spectrogram) will subscribe to the data units. When new units arrive at the map (or spectrogram), or when the map (or spectrogram) redraws, it will call WorkshopOverlayDraw with it's projector as an argument. WorkshopOverlayGraphics can work out what type of projector it is (map or spectrogram) and call the appropriate drawing function.

3.2.6. WorkshopPluginPanelProvider

As the name suggests, this class provides plug in display panels. These can currently only be 'plugged' into the bottom of the spectrogram display, it is possible however that other displays will support the same plug ins in the future.

The WorkshopPluginPanelProvider contains an inner class WorkshopPluginPanel, which extends DisplayPanel and implements PamObserver.

The WorkshopPluginPanels each show a line graph of signal to noise ratio (SNR) for each channel, they also have fixed lines at $SNR = 0$ and at the value of the detection threshold.

The various Pamguard displays will make as many individual WorkshopPluginPanels as necessary and each will run independently (for instance, they may end up with different x and y scales depending on what they are plugged into)

WorkshopPluginPanel implements PamObserver and each instance subscribes to the backgroundDataBlock from workshopProcess in order to receive updates from the process as new SNR values are calculated for each channel.

Drawing on plug in panels takes place in two different places

1. Each time the update position on the spectrogram display changes, `containerNotification()` is called which informs the `PluginPanel` that scales or scale offsets have changed. In this example, all that happens is that the plot is cleared for a few pixels to the right of the current x coordinate on the spectrogram container. (Try commenting out these lines and you'll see that the line graphs are continually drawing over themselves).
2. Each time new data arrive from the `backgroundDataBlock`, those new data are drawn on the line graphs. Information required to calculate x coordinates from the time of the `backgroundDataUnit` and from the spectrograms current x coordinate and current time is used. Previous values are stored locally so that lines can be joined up.

3.2.7. WorkshopSQLLogging

This class is used to add functionality to the main detection output `DataBlock` which will allow data to be written to the Pamguard database. Pamguard currently only supports MySQL databases and MySQL is not installed on the teaching machines making it difficult to demonstrate this code.

The abstract `SQLLogging` class has been created to hide all the unpleasantness of writing database Structured Query Language (SQL) commands from the Pamguard user. `SQLLogging` will automatically create the appropriate database table, and populate that table with data in the format chosen by the developer. If additional table columns are added to contain additional information at a later date in the development process, those columns will get added to an existing table. As we implement support for other databases (such as MS Access), the developers code will add tables to those databases and populate them in the exact same way.

`SQLLogging` has two main functions, the first provides the table definition, the second (`setTableData(...)`) gets called back by the database manager when a new `PamDataUnit` is created and requires the developer to populate the database fields.

When defining a database table format, it is possible to set cross reference information to other database tables. In this example, we store a reference to the most recent entry into the `GPSData` table.

Each database column is defined with a specific format (e.g. `TIMESTAMP`, `FLOAT`, `LONG`, etc.). The types of the data set in `setTableData` must match the column types or the write operation may fail.

4. Suggested Exercises

4.1.Improve the detector:

1. Set a maximum length for detections

2. Set a minimum gap within detections, so that if the gap is small, detections merge into one.
3. Include any parameters controlling these functions to the dialog and the WorkshopProcessParameters.

4.2. Use more Pamguard functionality

1. Add a counting module that appears as a side panel giving the number of detections in the last 10 minutes (Hint: to do this, copy the ClickDetector.ClickSidePanel class and override the getSidePanel() function in WorkshopController).
2. Automatically trigger recordings whenever the detector makes a detection. For this to work, you will need to create at least one instance of the Pamguard Sound Recorder from the Pamguard 'File/Add modules...' menu, then add the following code:
 - a) Create an inner class in the WorkshopProcess which implements the RecorderTrigger interface.
 - b) Create an instance of this class and register it with the class RecorderControl using RecorderControl.registerRecorderTrigger (...)
 - c) Whenever you have an interesting event (i.e. at the time you create a new PamDataUnit) call RecorderControl.actionRecorderTrigger(...) which will tell all sound recorders in your system that you want to record.

What will happen ?

- a) When you register your recorder trigger, on each sound recorder panel, you should get to see a check box allowing you to enable or disable the trigger for that particular recorder.
- b) When you call actionRecorderTrigger, all sound recorders that have selected triggering from your detector will start a recording, taking historical data that's been stored in a buffer and adding it to the start of the recording and then continuing that recording for the amount of time defined in the member functions of your class that implemented RecorderTrigger.

See the Click Train Detector class ClickDetector.ClickTrainDetector for an example of this code.

Note that this is a rather naive example since this simple detector will probably trigger quite often, so you'll end up recording almost continually. Automatic recorder triggering is much better suited to events that are relatively rare, such as the detection of a sperm whale click train.

4.3. Add your own detector

We hope that Pamguard will be able to provide you with a framework within which you can develop your own detector modules and also hope that you will then share your detectors with the rest of the research community as we have shared ours.

The workshop demo detector took one of the Pamguard core development team a bit less than one day to write from scratch, so it's unlikely that you will have time to write your own detector in this training session. However, please take the opportunity to discuss what your detector does, what it needs as input and what it would provide as output with

one of the core team and we will be able to advise you on how to proceed. (For info, the workshop demo detector took one of the core team one day to write from scratch).

Appendix C Tables of Functionality

| Objective | Team | Status | Notes |
|---|----------------------------|---------------|---|
| Adaptations to support offline re-analysis | <i>OSU</i> <i>/SMRU</i> | Due June 2008 | |
| <i>Advanced configuration utilities (wizards, save as, standard set-ups, etc)</i> | <i>SMRU</i> | complete | PAMGUARD now supports settings save/as and export. |
| <i>Advanced displays 3D/airgun/vessel/ etc)</i> | <i>SMRU</i> | complete | |
| 3D localization | <i>Scripps</i> | complete | On Beta version |
| Sound Acquisition | <i>HWU</i> | complete | A new DLL (dynamic link library) has been created which now allows PAMGUARD to interface with multi-channel sound acquisition devices which are supplied with ASIO (Audio Stream Input/Output) compliant drivers. |
| Asio Sound Card Selection | <i>HWU</i> | complete | |
| NMEA Data Acquisition | <i>HWU</i> | complete | |
| FFT data for selected channels | <i>HWU</i> | complete | |
| Simulator | <i>HWU</i> | complete | PAMGUARD now features a simulator; whereby acoustic sources can be "placed" in the simulated environment. |
| New Database interface | <i>SMRU</i> | complete | |
| Filter module | <i>SMRU</i> | complete | |
| Sound playback | <i>SMRU</i> | complete | |
| Amplifier | <i>SMRU</i> | complete | |
| Patch panel | <i>SMRU</i> | complete | |
| Ishmael Detector/ Ishmael Locator | <i>OSU</i> | complete | |
| Bug-fixes from CODA trials | <i>SMRU</i> <i>HWU</i> | complete | |

Table C.1 PAMGUARD IV Delivered Functionality

| Module | Data Input | Output | Number | Function |
|--------------------------------|---|----------------------------------|--------|--|
| <u>Maps and Mapping</u> | | | | |
| NMEA data collection | Serial port or server | Strings of NMEA data | Any | Acquire NMEA data for use by other modules. |
| GPS Processing | NMEA data | GPS locations | 0-1 | Identified GPS data and unpacks NMEA strings |
| AIS Processing | NMEA data | Ship locations | 0-1 | Unpacks AIS data |
| Airgun display | GPS or AIS Data | Graphic overlay for map | Any | Displays the locations of airguns and mitigation zones referenced to either the PAMGUARD vessel or some other vessel broadcasting Its position over AIS. |
| Map | GPS and all detectors | Map graphics panel; map comments | Any | Displays vessel location and data from other modules. |
| <u>Utilities</u> | | | | |
| Simulator | None | Simulated audio data | 0-1 | Simulates clicks on all audio channels for a whale at a set location. |
| ODBC database | Multiple modules | Database | 0-1 | Stores data from multiple modules in a Microsoft Access or MySQL database |
| Depth Readout* | Analogue depth sensors | Depth data | 0-1 | Depth data are displayed on screen, used in real time to update hydrophone locations and sent to the database. |
| Terella Control* | Terella depth, heading and tilt sensors | Depth, heading and tilt data. | 0-1 | As depth data. |
| Seismic Veto* | FFT and raw data | FFT and raw data | Any | Detects very loud sounds using an in band energy detector and replaces these with randomly generated coloured noise based on recent measurements of the local noise field. |
| <u>Displays</u> | | | | |
| User Input | Typed comments | | 0-1 | Time stamps and stores any comment entered by the user |
| User Display | | Displays | 0-1 | Provides a container for spectrogram |

| | | | | |
|--------------------------------|---|---------------------------------|-----|--|
| Panel | | | | and radar like displays. Data from detectors can be overlaid on top of these displays. |
| | | | | |
| <u>Sound Processing</u> | | | | |
| Sound Acquisition | Sound cards, audio files and other acquisition devices. | Packed chunks of raw audio data | Any | Controls a data acquisition device and passes it's data on to other modules |
| Sound Output | Audio data | Output to the system sound card | Any | Outputs audio data through a sound card so it can be listened to on headphones |
| FFT (Spectrogram) Engine | Audio data | FFT Data | Any | Computes spectrograms (Fourier transforms of multiple overlapping chunks of audio data). |
| IIRF Filters | Audio data | Audio data | Any | Filters audio data using either Butterworth or Chebychev filter. |
| Decimator | Audio data | Audio data | Any | Filters and converts the sample data of audio data |
| Sound Recorder | Audio data | Wav or Aif files | Any | Writes audio data to file. Recording may be initiated by the operator or triggered by a detector. |
| Signal Amplifier | Audio data | Audio data | Any | Amplified audio data by some scaling factor (can invert data if required) |
| Patch Panel | Audio data | Audio data | Any | Mixes data between multiple channels, changes channel ordering, etc. |
| Spectrogram smoothing kernel | FFT Data | FFT Data | Any | Smooths successive frames in a spectrogram by convolving them with a Gaussian smoothing kernel. |
| | | | | |
| <u>Detectors</u> | | | | |
| Click Detector | Audio Data | Detected Clicks | Any | Searches for transient sounds, attempts to assign species, measure bearings to source, group into click trains, etc. |
| Whistle Detector | FFT Data | Detected Whistles | Any | Searches for tonal noises. Measures bearings and locations of source. |
| Whistle Classifier* | Detected Whistles | Species probabilities. | Any | Analyses multiple whistle contours and uses a discriminant analysis function to assign to species. |

| | | | | |
|---------------------------------|-----------------------|-----------------|-----|--|
| Ishmael Energy Sum | FFT Data | Detected sounds | Any | Detects sounds with energy in a specific frequency band |
| Ishmael Spectrogram Correlation | FFT Data | Detected sounds | Any | Detects sounds matching a user defined 'shape' on a spectrogram |
| Ishmael Matched Filtering | Audio Data | Detected Sounds | Any | Detects sounds using a user defined matched filter. |
| Ishmael Locator | FFT Data / User input | Located sounds | Any | Locates sounds extracted from areas marked out on a spectrogram display |
| 3D Locator | Detected Clicks | Located sounds | Any | Locates sounds detected by the click detector using surface echo's to obtain slant angles and generate a 3-D location. |

* Module under development

Table C.2 PAMGUARD modules and module dependencies.

Appendix D Developer Guide for Branched CVS Repository (HWU)

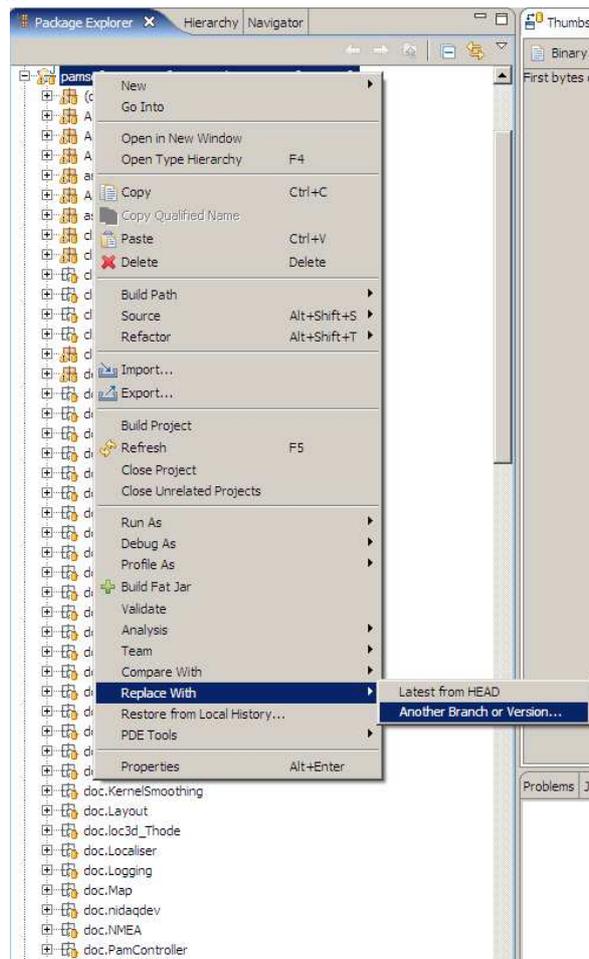
D.1. Introduction

On the 22nd January 2008 the PamGuard CVS repository was split into two branches. The HEAD (Core) and PamBranch (Beta) versions. The HEAD is the source code for the Core release, please do a full test before you check code in to HEAD. The intention is that development is checked into PamBranch before your full test.

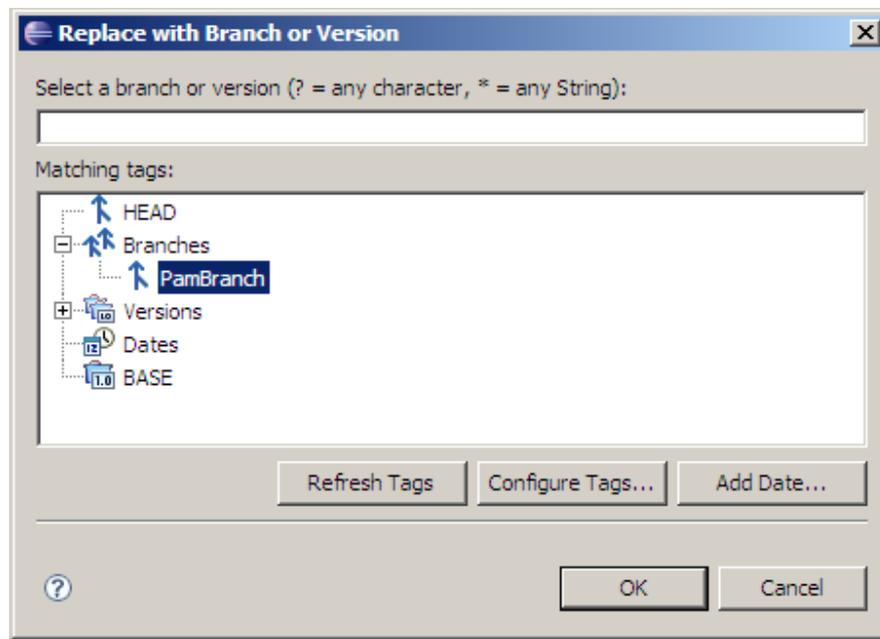
D.2. To Switch Between the Two Branches

D.2.1 From HEAD to PamBranch:

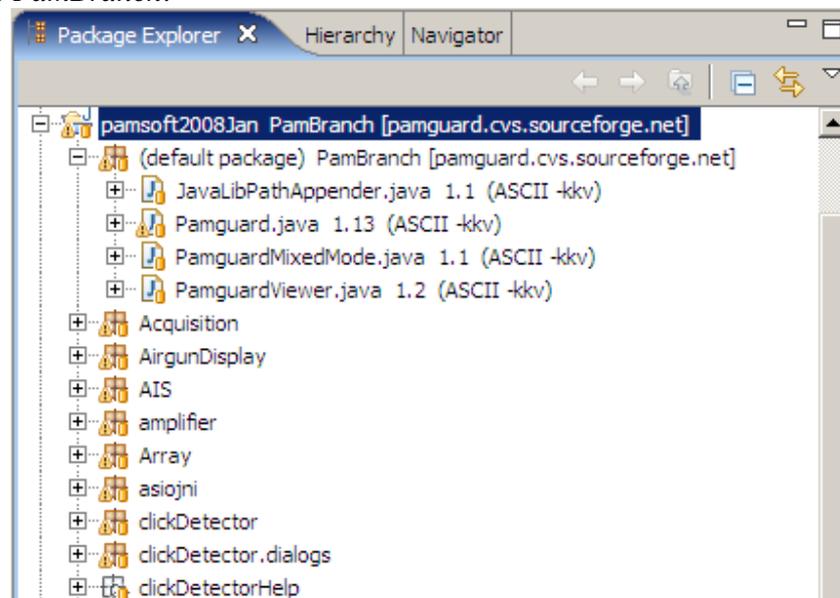
If this is the first time you are using the CVS after the CVS repository has been split, your current local version should be Head. To switch the project contents to that of the PamBranch, right-click on the project in the Navigator view and select **Replace with --> Branch or Version**



You should see a branch selection similar to the one below. (You may have to click **Refresh Tags** to see them all.) To switch to PamBranch, select the PamBranch under Branches. Then click **OK**.

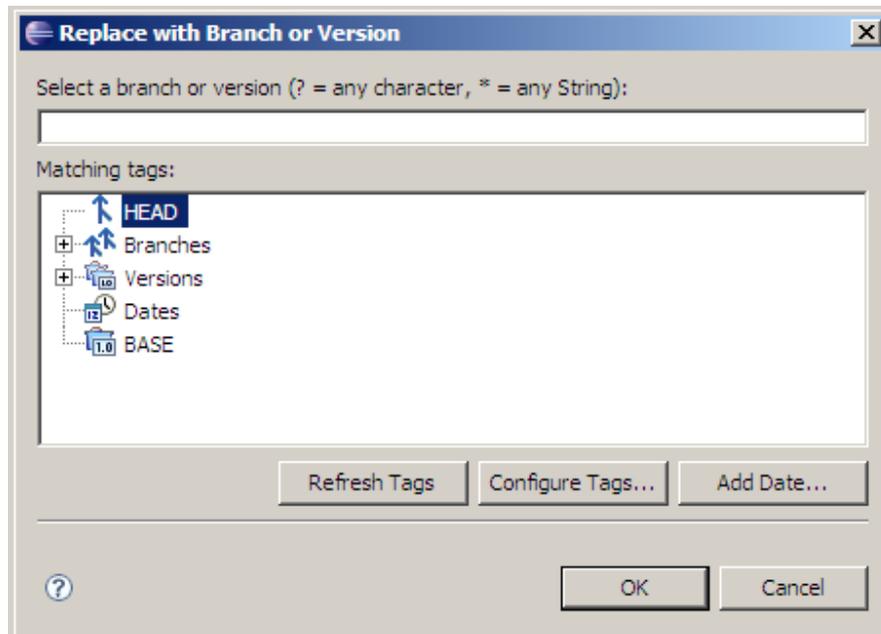


The resource view should look like the one below. The branch name is gone which implies the *PamBranch*.



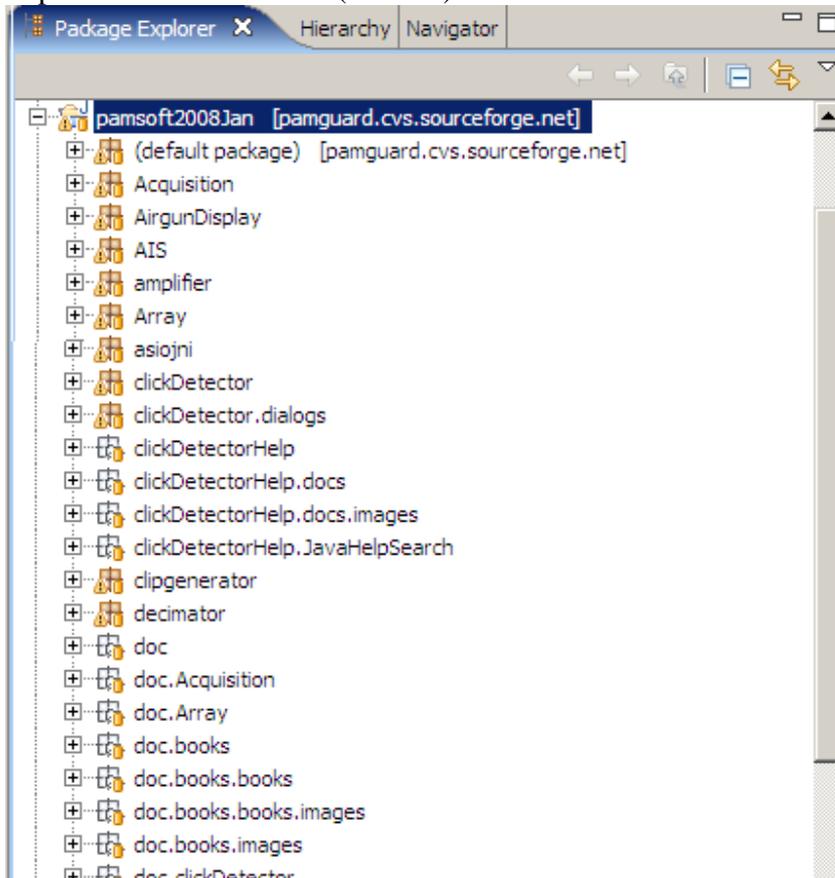
D.2.2 From PamBranch to HEAD:

To switch the project contents to that of the main branch, right-click on the **pamssoft2008Jan** project in the Navigator view and select **Replace with --> Branch or Version ...**. You should see a branch selection similar to the one below.



Select HEAD, then click **OK**.

The resource view should look like the one below. The branch name is gone which implies the *HEAD* branch (or main).

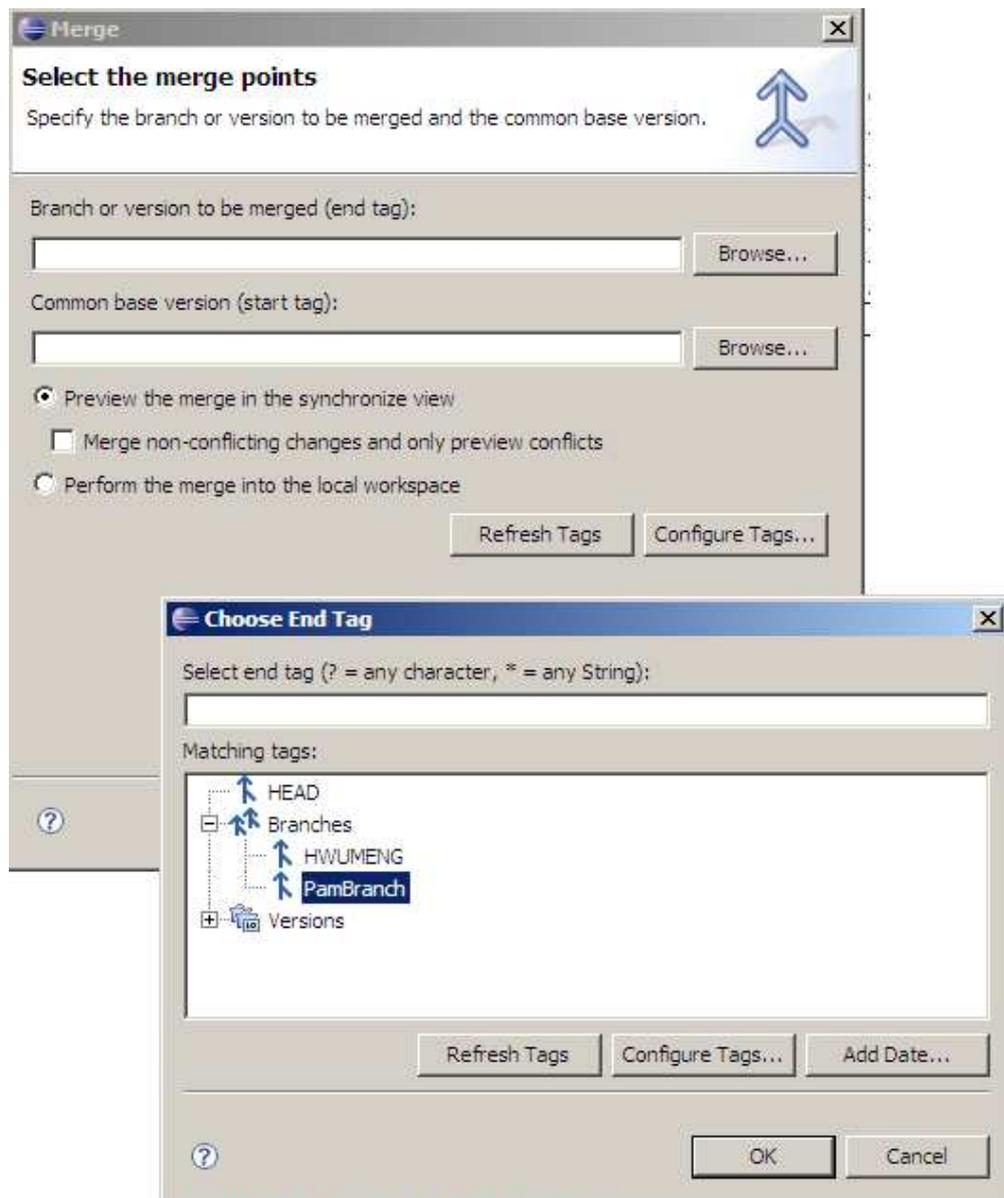


D.3. To Merge

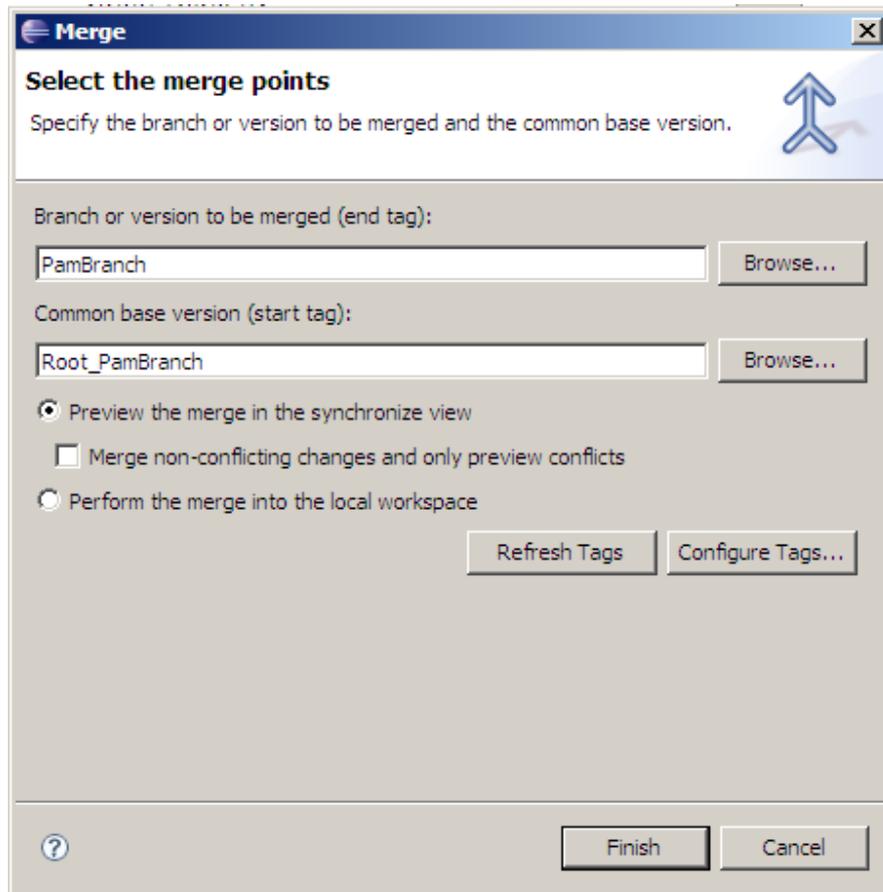
D.3.1 To Merge your Changes in PamBranch into HEAD:

The first step of the merge is to point the workspace to the target branch. In this case, the target of the merge is the main branch, *HEAD*. To switch the project contents to that of the *HEAD* (See section 2.2). All the version numbers are two digits. The branch name is gone which implies the *HEAD* branch (or main).

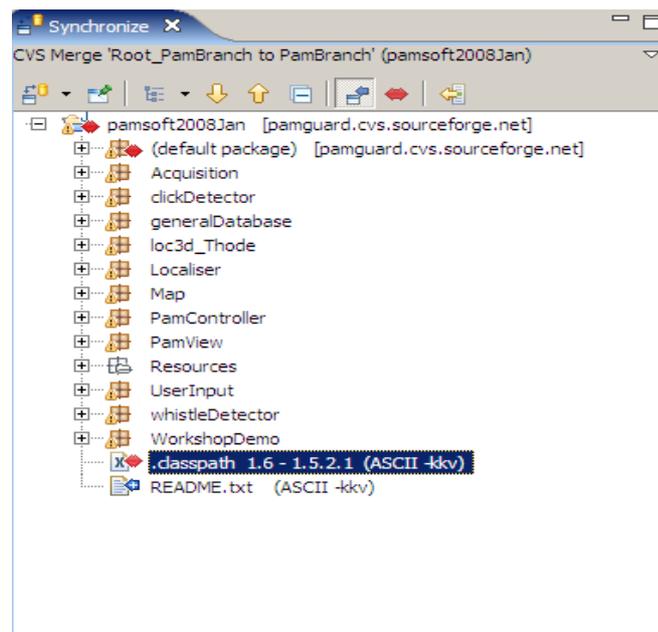
1. Right-click on the **pamsoft2008Jan** project and select **Team** → **Merge ...**. Click the **Browse** button next to the **Branch or version to be merged** field. Choose the **PamBranch** branch from the **Choose End Tag** dialog box. If you don't see the branch, you may have to click **Refresh tags**. Click **OK**.



2. The start tag should be filled automatically with **Root_PamBranch**. In this case, the dialog anticipated correctly. For this merge, we have elected **not** to check the **Merge non-conflicting changes** checkbox. In the merge-again scenario, we will check it and compare the difference. Click **Finish**.



3. This is where the fun begins. You will be presented with the *Synchronize* view similar to the following merge results:



Right-click on the **.classpath** and select **Merge**. CVS will attempt to work out if there are any changes that are in conflict between Local version and the repository.

In our example, it updates **.classpath** as it was able to resolve the conflicts there. However, **readme.txt** presents it with a challenge and Eclipse will alert you that it was not able to automatically merge the changes.



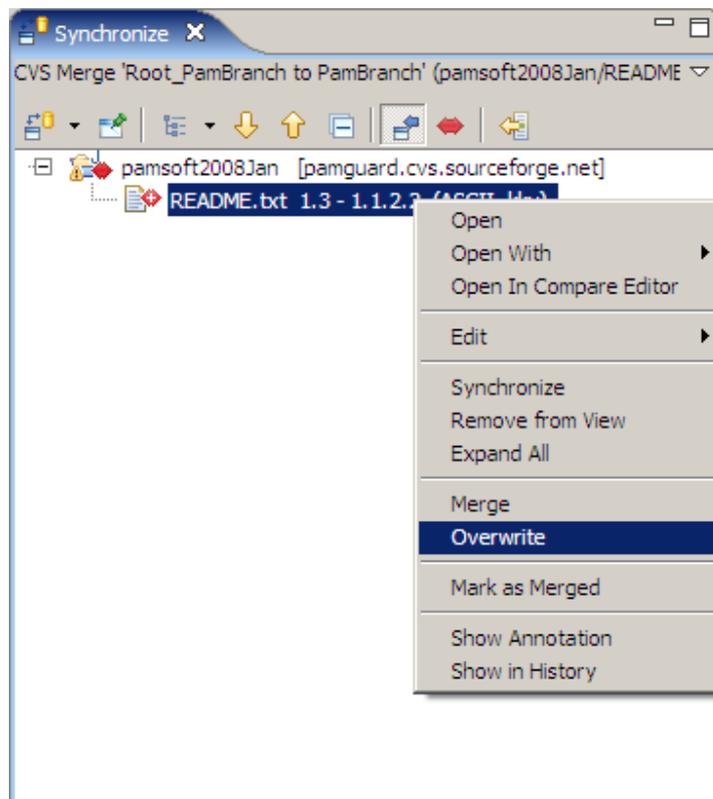
As ominous as this appears, it's merely a reminder that we have to resolve some merge conflicts because they were not auto-mergeable. We could have avoided the spectacle of this message if we had simply checked the **Merge non-conflicting changes** checkbox.

Now let's resolve the conflicts.

4. Double-click **readme.txt** in the *Synchronize* view to open the *Compare* editor of the merge tool. The left side of the *Text Compare* panel is the local copy of the main branch contents. The right side is the **PamBranch** branch contents.

We can't resolve this conflict by simply using one side or the other. For this case, we edit the text in the left side (local file) to say "This line changed by Paul and Wing in iter1." Right-click in this same text window and select **Save**.

Or we can right click **readme.txt** in the *Synchronize* view select **overwrite**. This will write the remote context to the local workspace.

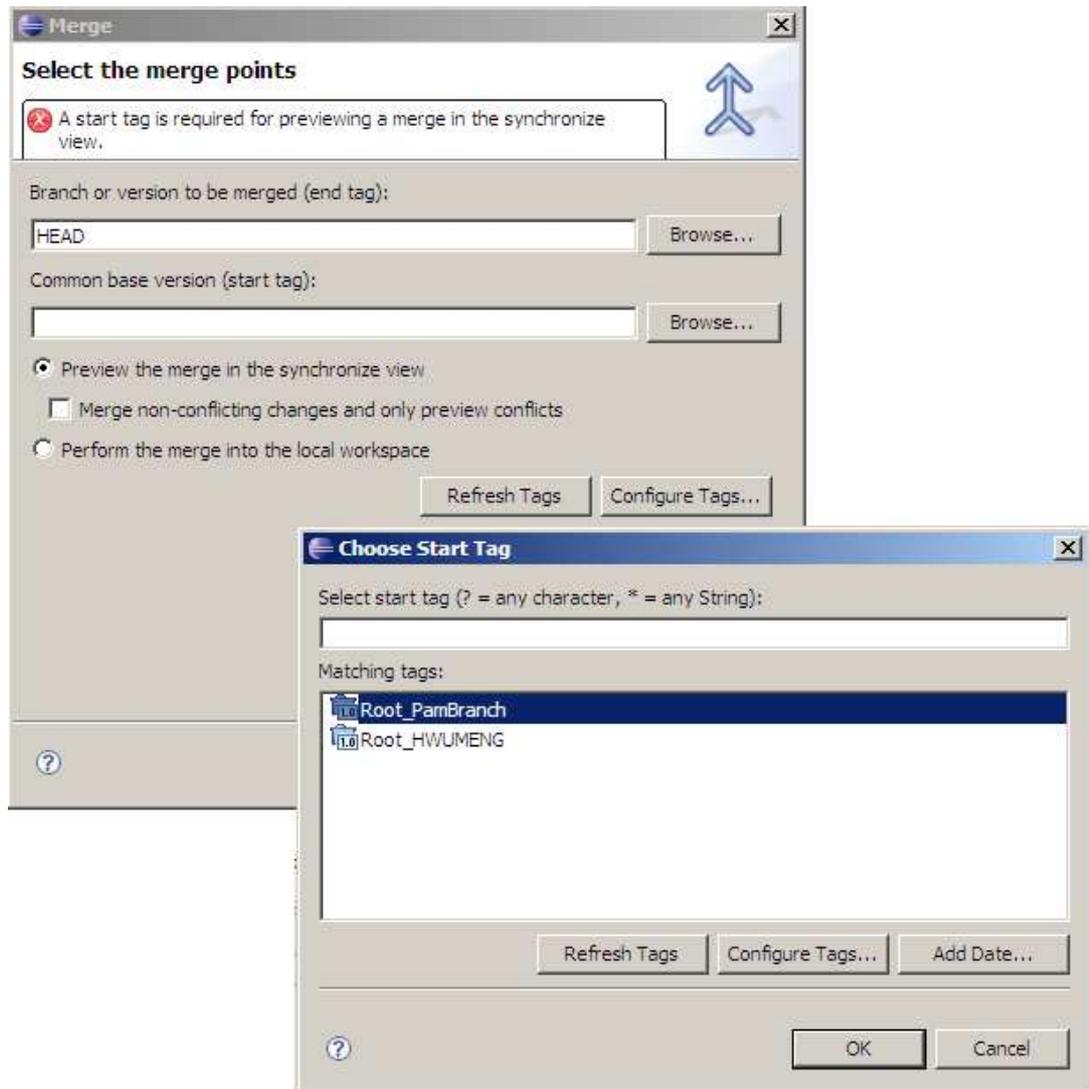


- At this point, the merged copy only exists in our workspace. We still need to commit it to the CVS repository. Select the project in the *Navigator* view and select **Team** → **Synchronize with Repository**. These changes should not present any conflicts and then commit it.

D.3.2 To Merge your Changes from HEAD into PamBranch:

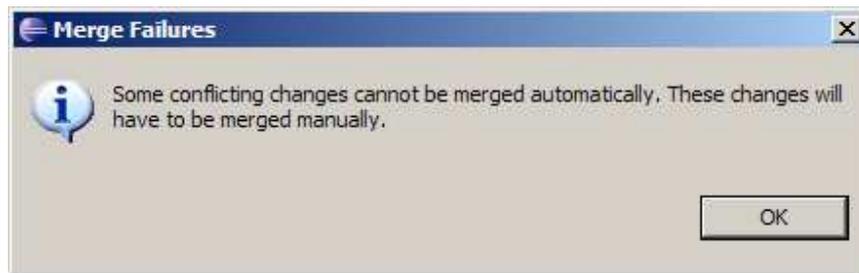
Now it is time to merge your changes from the main branch (HEAD) into the **PamBranch** branch. This merge is similar to merging your changes from **PamBranch** into HEAD. To do this, you need to switch the project contents to that of the PamBranch (See section 2.1).

Right-click on the **pamssoft2008Jan** project and select **Team** → **Merge ...**. Click the **Browse** button next to the **Branch or version to be merged** field. Choose the **HEAD** branch from the **Choose End Tag** dialog box. (If you don't see the branch, you may have to click **Refresh tags**. In the **start tag** field, click browse and select **Root_PamBranch** from the list of tags.



As for the rest of the panel, select **Preview the merge in the synchronize view** to take advantage of Eclipse CVS support for resolving merge conflicts. Also check **Merge non-conflicting changes and only preview conflicts**. Otherwise we will have to merge the non-conflicting changes manually. We will use these settings for all our merges in this article.

Click **Finish**. You should see the following message.



This is not as dire as it might seem. It just means there are conflicts we have to resolve. Click **OK** to continue.

The following steps are the same as that in section 3.1, which are merge from PamBranch to HEAD.

After solving all the conflicts, do not forget to commit you local space to CVS.

D.4. Others

Other activities such as synchronize, commit, update, overwrite, etc. are the same as that in non-branching CVS.

D.5 References

Some useful references, in addition to the Eclipse documentation are given below.

- *Version Management with CVS*, Per Cederqvist et al.
<http://ximbiot.com/cvs/manual>. This manual is the standard CVS reference.
- *CVS Best Practices*, Vivek Venugopalan 2002 html, pdf(75k)
- *Branching with Eclipse and CVS, Part 1: The Basics*
<http://www.eclipse.org/articles/article.php?file=Article-BranchingWithEclipseAndCVS/article1.html>
- *Branching with Eclipse and CVS, Part 2: Rebasing*
<http://www.eclipse.org/articles/article.php?file=Article-BranchingWithEclipseAndCVS/article2.html>