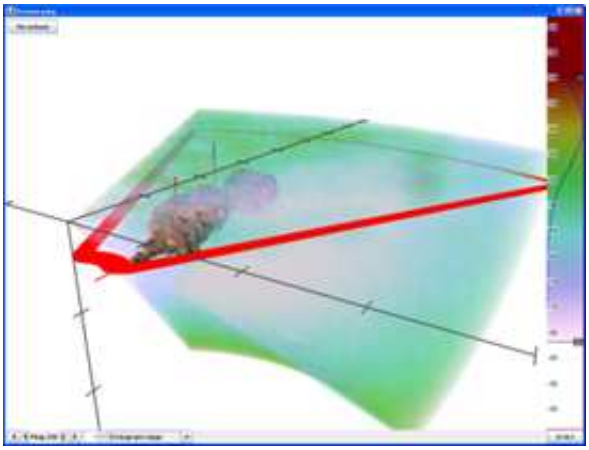


PROMUS-korona

User Guide

Version 1.11.0

June 30, 2016

Contents

- PROMUS preprocessor 1
  - Modules 2
    - DataReductionModule 3
    - DepthModulePromus 4
    - Display3DModule 5
    - EchoLineExtractor 6
    - Median spike filter module 8
    - Noise acceptance module 9
    - Noise median quantification module 10
    - Noise quantification file logger module 11
    - Phantom module 12
    - RegionModulePromus 14
    - Spatial spike filter 15
    - School cluster module 16
    - Temporal spike filter 17
    - ThresholdEchoLineExtractor 18

PROMUS

The PROMUS realtime processing of MS70 data is a plugin to the Korona realtime processing system.

Modules

DataReductionModule
DepthModulePromus
Display3DModule
EchoLineExtractor
NoiseAcceptanceModule
NoiseMedianQuantificationModule
NoiseQuantificationFileLoggerModule
RegionModulePromus
SchoolClusterModule
SpatialSpikeFilterModule
TemporalSpikeFilterModule

Echo line extractor

The data reduction module can be used to discard unwanted data. Data can be discarded based on range, depth and fan number.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Minimal range:`** Data with a range less than this value [m] will be discarded.
- **`Maximal range:`** Data with a range larger than this value [m] will be discarded.
- **`Minimal horizontal distance:`** Data with a horizontal distance from the transducer less than this values [m] will be discarded.
- **`Maximal horizontal distance:`** Data with a horizontal distance from the transducer larger than this value [m] will be discarded.
- **`Horizontal cut test position:`** Where in the sample to test for the horizontal cut. Allowed values:
  `Sample center`: Use the position of the sample center.
  `Sample extent`: Use the corners of the sample extent computed from the equivalent beam widths multiplied with **`Scaling of the equivalent beam width`**.
- **`Minimal depth:`** Data with depth less than this value [m] will be discarded.
- **`Maximal depth:`** Data with depth larger than this value [m] will be discarded.
- **`Depth cut test position:`** Where in the sample to test for the depth cut. Allowed values:
  `Sample center`: Use the position of the sample center.
  `Sample extent`: Use the corners of the sample extent computed from the 3dB opening angles (so not really the equivalent beam widths) multiplied with **`Scaling of the equivalent beam width`**.
- **`Scaling of the opening angles (and thereby also the equivalent beam width):`** Scaling of the opening angles (and thereby also equivalent beam widths) for use in the horizontal and depth cuts.
- **`Minimal fan number:`** Data in fans with fan number less than this value will be discarded. The deepest reaching fans have low numbers, and the fans closest to the surface have high numbers.
- **`Maximal fan number:`** Data in fans with fan number higher than this value will be discarded. The deepest reaching fans have low numbers, and the fans closest to the surface have high numbers.
- **`Minimal vertical fan number:`** Data in vertical fans with fan number less than this value will be discarded. The leftmost fans as seen from the transducer have the lowest fan numbers.
- **`Maximal vertical fan number:`** Data in vertical fans with fan number higher than this value will be discarded. The rightmost fans as seen from the transducer have the highest fan numbers.

DepthModulePromus

The DepthModulePromus has two ways of finding a bottom:

- Find a good bottom candidate for each beam individually.
- Fitting a plane to all bottom candidates and use the beam-plane intersection to find the depth for each individual beam

Algorithm

Find a good bottom candidate for each beam individually

Each beam is first traversed in search of local peaks. These peaks are used as bottom candidates. The width of the candidates are checked against an expected width, given a depth and incident angle, assuming a flat bottom. The candidate that best matches the expected width is selected as the bottom, provided the candidate passes cuts, see below. Pulse width is calculated as the standard deviation of a pulse.

Use fitted plane as bottom

Since it is not guaranteed that a bottom is detected for each beam (if the cuts are not passed), the bottom will in most cases have some "holes". A plane can be fitted to the detected points in order to get a bottom for each beam. The plane is fitted iteratively, for each fit outliers in terms of residuals are removed. This is done until there are no more outliers or a certain number of iterations has been done.

Settings

Below are the DepthModulePromus settings explained.

- **InitialDepthGuess** - Used for adjusting the search window for peaks (bottom candidates).
- **RemoveDepthOutliers** - Outliers in terms of depth are removed. An outlier is defined as < Q1 - 1.5(Q3 - Q1) or > Q3 + 1.5(Q3 - Q1) where Q1 and Q3 are the first and third quartile of the depth distribution respectively.
- **IterativeFit** - Fits a plane iteratively to the detected bottom points.
- **DoInterpolation** - When fitting is not used, this interpolates bottoms for beams that have no detected bottom. At least one neighbour must have bottom for this to be done. Interpolated bottom points are not used in further interpolations.
- **LogSvThreshold** - Min logSv value of a peak for it to be used as bottom candidate.
- **StabilityMeasure** - Criteria for deciding when to stop integrating the pulse. When the integral increases with less than this fraction over five samples, the pulse is taken to be stable.
- **PulseWidthDeviation** - The allowed fractional deviation of the measured pulse width wrt the expected pulse width.
- **UpperFan** - Fan cutoff. Only attempt to detect bottom below this fan.

Display3DModule

The Display3DModule provides a 3D view of a ping. The user can rotate, zoom and translate the ping. It is also possible to change the threshold, using the left bar threshold scale.

Controls

These are the controls used for exploring the ping:

- **Drag mouse (left mouse button)** - Rotate
- **Drag mouse (right mouse button)** - Translate
- **Scroll wheel** - Zoom
- **Alt + drag mouse** - Rotate in screen plane
- **Home** - Reset view to default view

Settings

Settings in the module configuration.

- **ShowBottomAsSurface** - Displays the detected bottom as a surface.

Echo line extractor

Module for scanning beams and extracting the parts which are a specified level above noise level. The extracted "echo lines" are stored in compact format in special raw datagrams, and the rest of the data is discarded. This module can thus be used to reduce the amount of data stored to disk.

The noise level can either be read from disk, or be taken from noise datagrams generated by earlier modules in the processing setup.

Warning: The noise level is range dependent, so the use of this module after noise removal will remove too much data at long ranges. Th use of many many processing modules after this module will uncompress data.

The user decides which noise file to compare against. Four choices are available:

- Check against file noise file: Checks against the median recorded noise for the file closest in time before the time of the currently processed file.
- Check against day noise file: Checks against the median recorded noise for the files with date equal to the date of the currently processed file.
- Check against survey noise file: Checks against the median recorded noise for all the files in the survey.
- Check against survey low-noise file: Checks against the weakest recorded noise for all beams extracted from all the files during the survey.

It is possible to check multiple of these choices, but the module will check against the first available file in the sequence file, day, survey. If for example all the choices are checked, the module will check against file noise if it is available. If file noise is not available, the module will check against day noise, and so on.

The beam scanning works as follows:

- The beam is scanned from the sample closest to the transducer to the sample furthest from the transducer.
- If a sample value is above the provided detection threshold factor times the noise level for the beam:
  - Backstep to the first value which is below the backstep level factor times the noise level for the beam. This is the start of the echoline candidate.
  - Step forward until the first value which is below the backstep level factor times the noise level for the beam. This is the end of the echoline candidate.
  - Test the number of samples in the echoline candidate against the minimum line length criterion. If the echoline candidate is longer than the criterion, the echoline is accepted.
- Continue until all samples in the beam is scanned. This is done for all beams in the ping.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`DetectionThresholdFactor:`** Detection threshold factor. The sample must be above this factor times the noise level to start extracting an echo line.
- **`BackstepLevelFactor:`** Backstep level factor.

- **`MinLineLength:`** The minimum line length criterion in number of samples.
- **`UseNoiseFile:`** If toggled, noise data is read from file. Otherwise the noise data is taken from datagrams generated by modules earlier in the processing setup.
- **`UseFileNoiseFile:`** Check the noise against file noise file, if available.
- **`UseDayNoiseFile:`** Check the noise against day noise file, if available.
- **`UseSurveyNoiseFile:`** Check the noise against survey noise file, if available.

Median spike filter module

A module for removing spikes in the 3D pings. The module works on a per sample basis, and compares each sample to surrounding samples in a user defined *search window*. The search window is defined by three parameters; the number of pings, X (X = 3, not user controlled), the search height, H, of 2 * **`mWindowMedianSearchHeight`** + 1, and the search width in horizontal direction, WX, of 2 * **`mWindowMedianHorizontalSearchWidth`** + 1, and the search width in the vertical direction, WY, of 2 * **`mWindowMedianVerticalSearchWidth`** + 1. The search height is in the beam direction, while the search widths considers neighboring beams.

To keep the complexity down, the search width is only used in the current (center) ping, so the number of samples used in the median calculation is (2 * WX + 1) * (2 * WY + 1) * (2 * H + 1) + X * (2 * H + 1).

If the center sample is greater than search window median + **`TotalDelta`**, the current sample is replaced by the search window median value.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`TotalDelta:`** Require center sample > (window median + TotalDelta) to consider sample part of a spike. Given in log(sv).
- **`WindowMedianSearchHeight`** The search window height in the beam direction.
- **`WindowMedianHorizontalSearchWidth`** The search window in horizontal direction, considering neighboring beams.
- **`WindowMedianVerticalSearchWidth`** The search window in vertical direction, considering neighboring beams.
- **`Debug:`** Replace spikes by 0 instead of median.

Noise acceptance module

This module compares noise detected by noise detection modules to values recorded on files. If the deviation between detected and recorded noise is large, the module replaces the detected noise value with recorded noise value. The idea behind this is to avoid spikes in the detected noise by comparing the values to values recorded over larger time intervals.

The user decides which noise file to compare against. Three choices are available:

- Check against file noise file: Checks against the median recorded noise for the file closest in time before the time of the currently processed file.
- Check against day noise file: Checks against the median recorded noise for the files with date equal to the date of the currently processed file.
- Check against survey noise file: Checks against the median recorded noise for all the files in the survey.

It is possible to check multiple of these choices, but the module will check against the first available file in the sequence file, day, survey. If for example all the choices are checked, the module will check against file noise if it is available. If file noise is not available, the module will check against day noise, and so on.

The comparison against noise on file is performed by computing the ratio detected noise to recorded noise. If this ratio is outside the specified noise fraction range, the detected noise is replaced with the recorded noise.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Noise fraction range:`** The fraction of detected noise to recorded noise should lie within this range.
- **`UseFileNoiseFile:`** Check the noise against file noise file, if available.
- **`UseDayNoiseFile:`** Check the noise against day noise file, if available.
- **`UseSurveyNoiseFile:`** Check the noise against survey noise file, if available.

Noise median quantification module

This is a quantification module that can be used to quantify noise while the sonar operates in active mode.

The noise is quantified by picking a user defined number of equidistant samples per beam over a specified number of pings and computing the median. Only the outer samples of the beam is used. If the maximum range of a beam is denoted MR, and the user specifies a detection distance of x meters, only the samples in the range [MR-x, MR] is used. The median is compared with the noise values in a specified noise file folder. The module automatically selects the noise file with time and date less than and closest in time to the file being processed. The computed noise estimates are used if they are within a specified fraction range of the noise values recorded to file. If they are outside the range for a given beam, the noise values recorded to file is used instead and the discrepancy is shown to the user in a list visualisation.

The module creates and updates noise files per input file, per day and per survey. The files are written to at the end of the processing of the current file, and are placed in a subdirectory to the processed files.

NB: This module creates noise datagrams, the noise is removed with the NoiseRemoverModule in the standard Korona package.

New datagrams

- Noise quantification datagrams, Nqp0Datagram.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Ping history (# pings):`** Number of historical pings to use when computing new noise values.
- **`Noise samples per beam:`** Number of samples to extract per beam for noise calculation (median).
- **`Detection distance:`** The outer distance for each beam [m] used for noise detection.

Noise quantification file-logger module

This module logs estimated noise values to file. The noise should preferably be logged while the sonar records data in passive mode. The noise level is estimated individually for each beam as the lowest noise level detected during the noise level recording.

File requirements

The **`Promus noise directory`** must be specified. This is the directory where the file with the detected noise levels will be placed.

Parameters

The parameters are the same as the Korona NoiseQuantificationModule, and are repeated here:

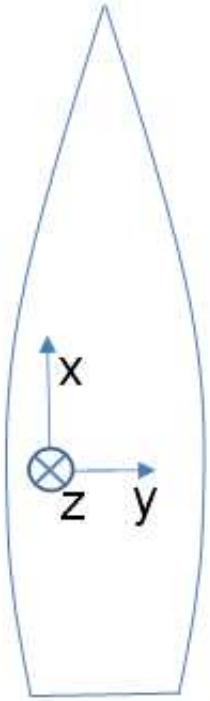
- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Smooth:`** Type of smoothing.
  Allowed values: `{ none, running averager, averager }`
- **`SmoothInterval:`** Number of samples to smooth vertically before making histogram.
  Allowed values: `SmoothInterval` >= 0
- **`Mask:`** Type of masking.
  Allowed values: `{ none, dynamic }`
- **`MinimumQuality:`** Minimum quality of histogram data.
  Allowed values: 0.0 <= `MinimumQuality` <= 100.0
- **`ThresholdMasking:`** Masks values above a threshold and reflections below bottom.
- **`Threshold:`** The threshold to use for threshold masking.
- **`Histogram:`** Type of histogram.
  Allowed values: `{ geometric, linear, constant }`
- **`UseTimeStepBuffer:`** Try to center noise parameters.
- **`TimeStepBufferMaxSize:`** Maximum number of time steps in buffer.
- **`HistogramInitializationCellCount:`** Number of cells in the histograms at initialization.
  Allowed values: `HistogramInitializationCellCount` >= 1
- **`HistogramMaximumCellCount:`** Maximum number of cells in the histograms.
  Allowed values: `HistogramMaximumCellCount` >= 1
- **`HistogramInitializationSampleCount:`** Minimum count of noise samples at initialization of the histograms.
  Allowed values: `HistogramInitializationSampleCount` >= 1
- **`HistogramMinimumSampleCount:`** Minimum number of noise samples in the histograms.
  Allowed values: `HistogramMinimumSampleCount` >= 1
- **`HistogramSmooth:`** Whether or not to smooth the probability distribution.
- **`HistogramSmoothFactor:`** Diffusion time step = HistogramSmoothFactor * dx^2.
  Allowed values: `HistogramSmoothFactor` > 0.0

P Phantom module

The phantom module creates synthetic "phantom" echosounder beams from the 3D sonar data. The name phantom comes from the view that the data can be interpreted as being collected from phantom vessels surveying parallel transects to the real vessel.

This module alters the data stream because it converts a 3D MS70 ping to one or more echosounder beams. This means that modules specific to MS70 sonar data cannot be placed after this module in a processing setup.

The coordinate system for specifying the positions and direction of the phantom beams is the ship system, and is included here for reference. The z-axis points downwards.



Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Beam direction type:`** Selection of how the beam direction is to be specified. By selecting `Vertical`, all phantom beams are vertical and placed equidistantly along the same angle in the horizontal plane. By selecting `Horizontal`, all beams are horizontal and placed equidistantly oriented along the same angle with the horizontal plane. Selecting `General` gives the option of specifying the placement and direction of the phantom beams in more detail.
  Allowed values: `{ Vertical, Horizontal, General }`

Common options for all beam direction types

- **`Number of phantom beams:`** The number of phantom beams that should be produced by this module. If this value is set larger than 1, the position of the last phantom beam should also be provided, and the intermediate beams will be equidistantly distributed between the first and last beam.
- **`Sample distance:`** The sample distance of the phantom beams.
- **`Fan aggregation type:`** Gives the option of aggregating along fans with the same angle with the vertical axis.
  Allowed values: `{ None, Average, Max }`
- **`Half width of fan aggregation window:`** The half width of the fan aggregation window. The algorithm will attempt to aggregate over this many samples to each side of the phantom sample, but will not extend past the min/max index in each fan. The maximal aggregation window will thus be 2 times the half width plus 1.

Common options for beam direction type `Vertical` and `Horizontal`

- **`Vertical plane angle:`** The angle in the horizontal plane of the vertical cut-plane where all the phantom beams will lie.

Special options for beam direction type `Vertical`

- **`Inner distance:`** The distance from the transducer of the innermost phantom beam.
- **`Outer distance:`** The distance from the transducer of the outermost phantom beam.

Special options for beam direction type `Horizontal`

- **`Upper depth:`** The depth of the uppermost phantom beam.
- **`Lower depth:`** The depth of the lowermost phantom beam.

Special options for beam direction type `General`

- **`X position:`** The x-position in the ship system of the first phantom beam.
- **`Y position:`** The y-position in the ship system of the first phantom beam.
- **`Z position:`** The z-position in the ship system of the first phantom beam.
- **`Last X position:`** The x-position in the ship system of the last phantom beam.
- **`Last Y position:`** The x-position in the ship system of the last phantom beam.
- **`Last Z position:`** The x-position in the ship system of the last phantom beam.
- **`X dir:`** The x-direction of the phantom beams given in the ship system.
- **`Y dir:`** The y-direction of the phantom beams given in the ship system.
- **`Z dir:`** The z-direction of the phantom beams given in the ship system.

RegionModulePromus

RegionModulePromus runs 2D school detection on some selected beams in order to give an indication of interesting data. It is a combination of Korona modules, set up in order to give a good school detection.

It necessary that a DepthModulePromus is run before RegionModulePromus, but this is left out of this module, so that it can be configured for optimal results.

Settings

The settings available in the Promus Module Editor

- **Channels** - Comma separated list of beams numbers to run school detection on.
- **Threshold** - The threshold defining the school.

In addition there are settings for limiting school parameters.

Modules

This presents a list of the modules RegionModulePromus is composed of.

- **ChannelCopyModule** - Creates copies of the channels to be traced. The filtering and tracing is run on the copies, so that original data is not altered.
- **RemoveBottomModule** - Removes data below bottom.
- **RemoveBottomModule** - Remove some data above bottom (100 samples).
- **3 x MedianModule + ErodeModule + DilateModule** - Filtering
- **TraceBorderModule** - Does the actual tracing of borders.
- **ChannelRemoveModule** - Removes the copied channels.

Spatial spike filter module

The purpose of the temporal spike filter module is to remove spikes which appear in a spatial sense, i.e. abnormally high backscatter value in one sample compared with the 4 surrounding samples in the same ping. Detected spikes are given the zero-value (the lowest possible s_v which can be represented in the data format).

The criteria for detecting a spike

- The sample in question must have a S_V value higher than the selected "spike threshold".
- The sample in question must have a S_V value which is "spike height" higher than the maximum S_V value in the neighbouring samples. The neighbouring samples are picked from the surrounding beams that are directly ’above’, ’below’, ’to the left’ and ’to the right’ of the beam in question. A neighbouring sample is a sample in a surrounding beam with the same sample index as the sample being tested. Some of the surrounding beams may be unavailable on the outer part of the beam array, and the maximum value to compare against is then taken among those that are available. If no corresponding samples are available, the test is skipped and the sample in question will not be detected as a spike.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`SpikeThreshold:`** The sample to be removed must have a value higher than this value [dB].
- **`SpikeHeight:`** The value of samples around the peak must be this much less than the peak value [dB].

School cluster module

This module runs a k-means clustering algorithm on the promus data to identify potential schools in each ping. Clusters are also matched to clusters in the previous ping to create school candidates extending over several pings. This is referred to as a cluster time series.

It has the following settings:

- **NumberOfClusters** - the number of clusters to use in the k-means.
- **MinSv** - minimum log sv to take into account for clustering. Samples below are ignored by the clustering.
- **Max depth** - samples below this are ignored.
- **Min depth** - samples above this are ignored.
- **Ignore fans** - the n uppermost fans are ignored.
- **Max sample** - samples with sample no. larger than this are ignored.
- **Min sample** - samples with sample no. smaller than this are ignored.
- **Color clusters** - Replaces clustered data with sv values (for debug only).
- **MinVolume** - min volume of cluster
- **ClusterSphereRatio** - avg spatial std dev of cluster is calculated, and the volume of a sphere with this ratio is calculated. Volume(cluster)/Volume(sphere) has to be larger than this number.

The algorithm initially creates a number of seed cluster points defined by **NumberOfClusters**. The k-means clustering is then run on these, with all samples above **MinSv** as input data. The metric used for the clustering is the Euclidean distance between cluster and sample. The algorithm is run iteratively until the cluster center do not change.

For each cluster, the total volume and the standard deviation for each spatial dimension is calculated. The volume and spatial extent can then be used to evaluate the school. The average standard deviation is calculated, and the cluster is checked against **MinVolume** and **ClusterSphereRatio**.

If the cluster passes these cuts, it is considered a school candidate, and is stored as a SchoolClusterDatagram.

If there were clusters in the previous ping, the current clusters is matched against these. A match is defined as follows: A vector defining the absolute difference in cluster center for the current cluster and a cluster in the previous ping is found. If the absolute values of the vector components is smaller than the sum of the standard deviation for the two clusters in the corresponding dimensions, the clusters are defined as matching. The matching clusters defines a cluster time series.

If the cluster does not match any previous clusters, a new time series of clusters are created with this cluster as the first cluster. At the end of the processing of a ping, if any cluster time series from the previous ping was not matched to any clusters in this ping, the series is terminated and written as a ClusterInfoDatagram.

References:
S. Lloyd, Last square quantization in PCM's. Bell Telephone Laboratories Paper (1957). S. P. Lloyd. Least squares quantization in PCM. Special issue on quantization, IEEE Trans. Inform. Theory, 28:129-137, 1982.
http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html

Temporal spike filter module

The purpose of the temporal spike filter module is to remove spikes which appear in a temporal sense, i.e. abnormally high backscatter value in one sample in one ping compared with the corresponding sample in pings immediately previous and after the ping. Detected spikes are given the zero-value (the lowest possible s_v which can be represented in the data format).

The criteria for detecting a spike

- The sample in question must have a S_V value higher than the selected "spike threshold".
- The sample in question must have a S_V value which is "spike height" higher than the maximum S_V value of the corresponding samples in the previous and the next ping. The corresponding sample in this context means the sample with the same beam number and same sample index as the sample being tested. If no corresponding samples are available, the test is skipped and the sample in question will not be detected as a spike.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`SpikeThreshold:`** The sample to be removed must have a value higher than this value [dB].
- **`SpikeHeight:`** The value before and after the current sample (in time) must be this much less than the peak value [dB].

P Threshold echo line extractor

A simpler version of the module Echo line extractor. Instead of the detected noise level, it uses a fixed threshold [dB] for extracting the echo lines. There is also no backstep level.

Parameters

- **`Active:`** If the module is not active then all datagrams pass unaltered through it.
- **`Threshold:`** The detection threshold in dB.
- **`MinLineLength:`** The minimum line length of the echo line in number of samples.