FFI RAPPORT

NavLab OneClick

SVARTVEIT Kristian

FFI/RAPPORT-2005/01361

NavLab OneClick

SVARTVEIT Kristian

FFI/RAPPORT-2005/01361

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
P O Box 25, NO-2027 Kjeller, Norway

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

P O BOX 25
N0-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2005/01361	UNCLASSIFIED	25
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFI	-	

4) TITLE
NavLab OneClick

5) NAMES OF AUTHOR(S) IN FULL (surname first)
SVARTVEIT Kristian

6) DISTRIBUTION STATEMENT
Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Navigation post processing	a) Etterprosessering av navigasjon
b) NavLab	b) NavLab
c) Smoothing	c) Glatting
d) Aided inertial navigation system	d) Integrert Treghetsnavigasjonssystem
e) Autonomous Underwater Vehicle (AUV)	e) Autonom undervannsfarkost

THESAURUS REFERENCE:

8) ABSTRACT

NavLab OneClick is a MatLab script for automatic NavLab pre-processing, estimation, and export of finished smoothed data. This means that the user will be relieved of data structure creation, wild point filtering, estimator initialisation, quality control, and export of the smoothed results. NavLab OneClick is a shell surrounding NavLab, performing many of the labouring tasks that otherwise requires user intervention.

This report holds the user guide for OneClick, including description of the necessary set up required before running NavLab OneClick the first time. All user defined parameters used by OneClick is thoroughly described. One chapter is devoted to a discussion on memory constraints, and some suggestions on how to improve the computer's performance. Finally the report gives a more detailed description of the code, what it does, and why it does it.

9) DATE	AUTHORIZED BY	POSITION
	This page only	
2005-04-29	Nils Størkersen	Director of Research

ISBN 82-464-0937-9

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

CONTENTS

1 EXCECUTIVE SUMMARY

NavLab OneClick is a MatLab script for automatic NavLab pre-processing, estimation, and export of finished smoothed data. This means that the user will be relieved of data structure creation, wild point filtering, estimator initialisation, quality control, and export of the smoothed results. NavLab OneClick is a shell surrounding NavLab, performing many of the labouring tasks that otherwise requires user intervention. See [1] and [2] for a thorough description of NavLab.

NavLab OneClick is a permanent part of the NavLab post-processing system. Thus, the name OneClick is only used for development purposes. Typical NavLab users occupied with NavLab post-processing of HUGIN navigation data, in the offshore survey industry or in the military, will view OneClick as a general upgrade to NavLab. An upgrade that improves their working conditions, assures quality, and reduces time and cost of post-processing substantially.

2 BACKGROUND

NavLab is developed as a powerful tool for navigation system research and development, navigation system accuracy analysis, and navigation data post-processing. This leads to a wide range of both users and usage. NavLab is versatile in its nature and makes all the different usages possible. However, the complexity of the system might make it elaborate for users with a narrow field of use. NavLab OneClick is intended for post-processing of HUGIN data using NavLab, and automates most user interactions with NavLab. This makes NavLab for post-processing faster and easier to use, and the risk of manual errors smaller.

2.1 Design philosophy

NavLab OneClick is implemented with emphasis on simplicity and fault tolerance. It shall be easy to use, and shall not require in-depth knowledge of inertial navigation and Kalman filtering. Nevertheless, the user will be presented with enough information to determine the quality of the navigation data. The end results shall be easy to navigate and locate for future use, and stored with enough information to reproduce the results.

2.2 Overview of this document

Chapter 3 is a user guide, including description of the necessary set up required before running NavLab OneClick the first time. Chapter 4 describes all user defined parameters used by OneClick. Chapter 5 discusses memory constraints and issues, and includes some suggestions on how to improve the computer's performance. Chapter 6 gives a description of the code, what it does, and why it does it.

3 RUNNING NAVLAB ONECLICK

This chapter gives a brief overview of the usage of NavLab OneClick. It states what input is needed and where to find the outputs. Some comments on the interpretation of the output are also given. For a thorough description of what NavLab OneClick does, and what algorithms are implemented, refer to Chapter 6.

3.1 Installation

You need an installation of NavLab on your computer, and the MatLab path should include all NavLab directories. At your preferred location a directory called `"OneClick"` should be placed, and added to the MatLab path. The most convenient way to manage this is through a `"startup.m"` file, or your pathdef. (See help in MatLab for more information).

The directories containing .ini files, the binary data converter, and the position outlier filter must be specified in `"OneClick.ini"`, if they are not subdirectories under `"OneClick"` named `"ini"`, `"BinaryDataToTxt"`, and `"NavLabPosFilter"` respectively. In addition, the output path and default run path can be specified.

OneClick will use the `"OneClick.ini"` found first in the path, this means you can have several copies and start OneClick from the appropriate location for different behaviour. Use the command `"which OneClick.ini"` to determine which `"OneClick.ini"` will be used from MatLab's current directory.

All parameters in `"OneClick.ini"` are described in Chapter 4.

3.2 Getting started

Typing `OneClick` in the MatLab command window starts the program. MatLab's current directory doesn't have to be the run directory, it can be any directory.

3.3 Selecting a run

The user is prompted for a run directory. If you find yourself always clicking down the same directory tree to locate your runs, a default starting directory can be specified in `"OneClick.ini"`. The chosen run directory must contain a subdirectory called `"navp\NavData"`. OneClick expects to find the sequential text files created from the HUGIN binary log-files here. If they are not, OneClick tries to create them from the log-files using `"BinaryDataToText.exe"`. If there is not at least one sequential text file for the following sensors: IMU, DVL, pressure and heading, OneClick is terminated with an error message.

OneClick creates a `"post"` directory adjacent to the `"navp"` directory in the run directory. If the post processed data is wanted elsewhere, an alternative location can be specified in

"OneClick.ini". If this directory doesn't exist, the user is prompted whether to stop or create it.

If there already exists a "NavLab" directory in the "post" directory, with results from a previous post-processing, the user is given two choices:

1. **Plot previous**: The previous post-processing is loaded from its .mat file, and 15 figures recommended for real data inspection are created.
2. **Process again**: The run is processed again.

If the run is processed again, the existing "NavLab" directory is moved to "NavLab_YYYYMMDD-HHMM", with the date and time when it was created.

3.4 Choosing position measurement aiding source

The user is prompted whether to use USBL, vehicle GPS, or both as position measurement aiding. If processing USBL, the "TopSideAUVPos.txt" file, or the older "pos.txt", must be available in the selected run directory.

3.5 Navigating the finished results

NavLab will produce a large amount of information displayed in the Matlab window. This is information regarding the status and progress of the various parts. All this information is saved in a log file called "OneClick_log.txt", and a separate log file for the preprocessing of each part called "Preproc_log.txt".

By default NavLab OneClick plots a latitude versus longitude overview with position measurements, real time Kalman filter positions, and smoothed positions. In addition, if specified in "OneClick.ini", all the figures recommended for inspection of the navigation results are also plotted. See [1] for guidelines regarding interpretation of these figures. The NavLab function plot_general will give access to the whole set of NavLab figures.

NavLab OneClick finishes with a quality check, which shows if the mean of any of the estimated biases are above 3σ. A short quality report gives the fraction of time a bias estimate was above 3σ, and the longest continuous interval the estimate was outside 3σ. If the estimated position accuracy is above the limit given in "OneClick.ini", a warning reports this.

The finished exported results are located in the "/NavLab" directory, as is the saved MatLab workspace as a .mat file. If the run was split, the .mat files are put in "/NavLab/part01", "/NavLab/part02", etc.

4 USER DEFINED PARAMETERS

In "OneClick.ini", a set of parameters specifying the behaviour of NavLab OneClick is defined.
Note: All string values must be without blanks (space, tabs, etc.). All blanks will be removed. This also means that the total path to a run directory must be without blanks.

4.1 .ini-file location

OneClick copies "preproc.ini", "estimator.ini", and "cov_matrix.ini" from this location. The default location is a "ini" directory in your OneClick directory. Figure 4.1 shows the directory tree with default directories for .ini-files, BinaryDataToText, and NavLabPosFilter.

Parameter	Default value	Example values
ini_file_location	0	0 : Default
		C:\NavLab\OneClick\ini

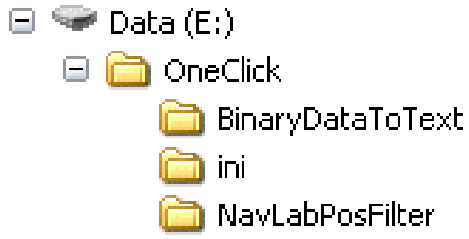

Figure 4.1 Default directories

4.2 BinaryDataToText location

If the .log files aren't extracted to sequential .txt files, OneClick will attempt to do this using BinaryDataToText.exe from this location. The default location is a "BinaryDataToText" directory in your OneClick directory.

Parameter	Default value	Example values
BinaryDataToText_location	0	0 : Default
		C:\NavLab\OneClick\BinaryDataToText

4.3 NavLabPosFilter location

The multiple Kalman filter system for automatic position wild point filtering is run from this location. The default location is a "NavLabPosFilter" directory in your OneClick directory.

Parameter	Default value	Example values
NavLabPosFilter_location	0	0 : Default
		C:\NavLab\OneClick\NavLabPosFilter

4.4 Default run path

When prompting the user for the run directory, OneClick will start in this directory. This is not a necessity, but convenient if all runs are located under the same tree structure.

Parameter	Default value	Example values
default_run_path	C:\	C:\mission
		/home/export/mission

4.5 Post path

The processed data from NavLab is by default placed in a "post" directory adjacent the "navp" directory in the chosen run directory. If another location is desired, it must be specified. In this directory, a new directory with the same name as the run directory will be created, with a "post" subdirectory.

Parameter	Default value	Example values
post_path	0	0 : Default
		C:\post

4.6 Plotting extra figures

OneClick will always plot an overview (latitude vs. longitude) and a summary of the estimation. In addition, OneClick can plot some 15 figures recommended for real data analysis.

Parameter	Default value	Example values
OneClick_figures	0	0 : No
		1 : Yes

4.7 Get mission waypoints from HUGIN OS

OneClick can plot the mission waypoints from HUGIN OS in the latitude vs. longitude overview plot. The HUGIN OS file "mission.mp" is needed in the run directory for this functionality to work.

Parameter	Default value	Example values
plot_mission_plan	1	0 : No
		1 : Yes

4.8 Position accuracy warning level

Estimated smooth position accuracy is displayed in NavLab figure number 25, if its largest value exceeds this limit, a warning is given in the summary.

Parameter	Default value	Example values
smooth_est_err_naveq_pos_limit	8	[meters]

4.9 Position aiding source

NavLab can use only one source of position measurement. This can be either vehicle GPS, DGPS-USBL, or a merge of the both created by OneClick. The user can be asked each time OneClick is run, or this parameter can control it.

Parameter	Default value	Example values
Position_aifing_source	0	0 : Ask the user
		1 : Vehicle GPS
		2 : DGPS-USBL
		3 : Both

4.10 Start stop time

Processing start and stop times can be based either on position or DVL measurements. If based on position, the time of the first position measurement is used as start time, and the time of the last IMU measurement is used as stop time. If based on DVL, the time of the first DVL measurement is start, and the last is stop (Limited by not being before or after the first and last position measurement). It is not possible to use parallel processing with DVL as start/stop base, because the splitting algorithm only assures the availability of position measurements at the start of each part, not DVL. The parallel processing will be disabled, and a warning given to the user, if DVL is start/stop base.

Parameter	Default value	Example values
start_stop_base	0	0 : Position
		1 : DVL

4.11 Dividing long runs

If the run is too long for the computer's memory, OneClick will divide it into pieces and run them separately. The division is made such that each piece has position measurements at both the beginning and the end. Each piece ends at the last position measurement before `max_estimator_length` seconds after it starts plus overlap, as seen in Figure 4.2

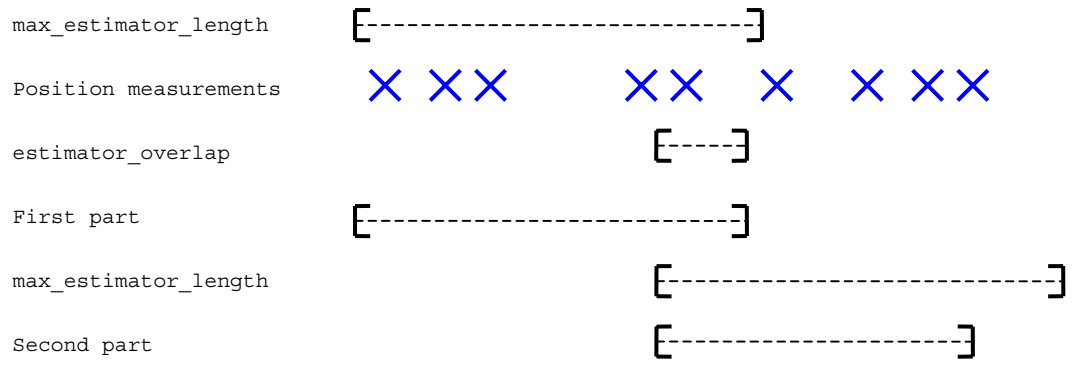

Figure 4.2 Dividing runs

To have good accuracy, an overlap between runs is needed. Two third of the defined overlap is before the cut, and one third is after. Increased overlap decreases differences in the intersection between two pieces, but will take longer time to process. The overlap should not be less than 6 minutes (360 seconds).

Parameter	**Default value**	**Example values**
max_estimator_length	8*3600	[seconds]
estimator_overlap	6*60	[seconds]

4.12 Parallel processing

If multiple processors are available, split runs can be processed in parallel. Each part of a split run will then be initialized based on available measurements, not the results from the previous estimation. Thus, longer overlap is recommended. After pre-processing a part, you may start another MatLab process, and run `"OneClick_parallel.m"`, which prompts you for a part to estimate. OneClick will skip all parts where OneClick_parallel has started. Note that if parts are not processed in parallel, the state and covariance are used for initialization as usual.

Parameter	**Default value**	**Example values**
enable_parallell_processing	0	0 : No
		1 : Yes

4.13 Manual wild point editing

Automatic DGPS-USBL wild point editing is done prior to preproc, and automatic DVL wild point editing is done afterwards. Manual wild-point editing of all sensor measurements can be done during preproc.

Parameter	Default value	Example values
manual_wildpoint_editing	0	0 : No
		1 : Yes

4.14 Automatic DGPS-USBL wild point filter

OneClick can run automatic wild point identification and removal on the DGPS-USBL measurements. Vehicle GPS is not checked for wild points, but measurements are removed based on reported accuracy from the GPS receiver. See [3] for further information on the USBL wild point filter.

Parameter	Default value	Example values
automatic_USBL_wildpoint_filter	1	0 : No
		1 : Yes
USBL_filter_max_speed	3	[m/s]
USBL_filter_vehicle_stability	2	1 : unstable
		2 : stable
		3 : very stable

4.15 DGPS-USBL measurements acceptance area

The parameters `USBL_min_accept_depth`, `USBL_max_accept_angle`, `USBL_max_accept_horizontal_distance`, and `USBL_max_accept_range` define the cut cone shown in Figure 4.3. When the AUV is outside this area, all USBL measurements are discarded.

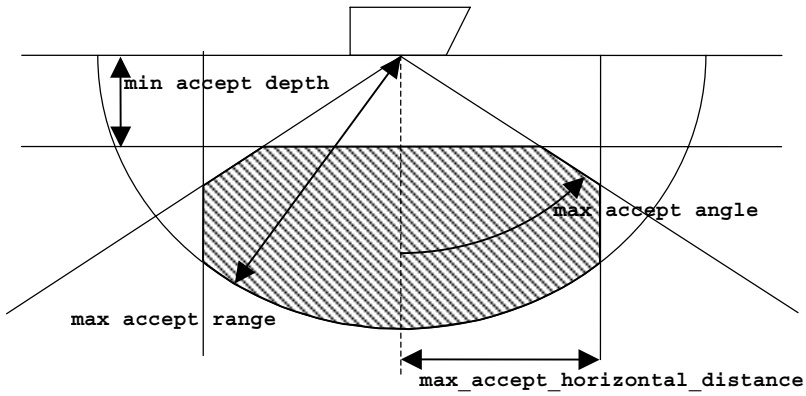

Figure 4.3 USBL acceptance area

Parameter	Default value	Example values
USBL_max_accept_angle_deg	60	[degrees]
USBL_min_accept_depth	10	[meters]
USBL_max_accept_horizontal_distance	300	[meters]
USBL_max_accept_range	3000	[meters]

4.16 Position measurement quality limit

If GPS, dGPS, or USBL position measurements have reported high uncertainty, they tend to be more inaccurate than reported. Thus, measurements with reported quality above the specified values are normally cut away, even though they might pass the wild-point filter. Note that the USBL quality is depth dependent, hence a larger value may be necessary if operating at large depths.

Parameter	Default value	Example values
USBL_quality_limit	14	[meters]
dGPS_quality_limit	10	[meters]
GPS_quality_limit	6	[meters]

4.17 Automatic DVL wild point filter

OneClick can run an automatic DVL wild point filter using a median based filter to detect outliers. See Chapter 6.7 for details.

Parameter	Default value	Example values
automatic_DVL_wildpoint_filter	1	0 : No
		1 : Yes

4.18 DVL start-up requirements

When the DVL first gets bottom track, it is advantageous to have a stable measurement rate before using it as an aiding sensor. All DVL measurements before a period of minimum `DVL_startup_interval_length` seconds with time between samples no larger than `DVL_startup_max_diff` will be discarded.

Parameter	Default value	Example values
DVL_startup_max_diff	4.5	[seconds]
DVL_startup_interval_length	60	[seconds]

4.19 Depth dependent DVL accuracy

Experience show that DVL performance tends to decrease with increased vehicle dynamics. Large vehicle dynamics is often present in the surface due to wave-induced motion. To counter this, OneClick can decrease the DVL accuracy when AUV depth is less than `DVL_depth_quality_limit`. In this case the low quality values in the table below are used.

Parameter	Default value	Example values
varying_DVL_est_quality	1	0 : No
		1 : Yes
DVL_depth_quality_limit	2.5	[meters]
low_quality_std_w_DVL_x	10e-3	[m/s]
low_quality_std_w_DVL_y	10e-3	[m/s]
low_quality_std_w_DVL_z	10e-3	[m/s]
low_quality_std_Dv_DVL_bias_x	10e-3	[m/s]
low_quality_std_Dv_DVL_bias_y	10e-3	[m/s]
low_quality_std_Dv_DVL_bias_z	10e-3	[m/s]
low_quality_T_Dv_DVL_bias_x	20	[seconds]
low_quality_T_Dv_DVL_bias_y	20	[seconds]
low_quality_T_Dv_DVL_bias_z	20	[seconds]

4.20 Depth dependent pressure sensor accuracy

A pressure sensor is intended for underwater use, and hence less accurate in air, and sometimes also during surfacing. Varying depth accuracy can compensate for this. The low quality values from the table below are used when AUV depth is above `depthm_depth_quality_limit`.

Parameter	Default value	Example values
varying_depthm_est_quality	0	0 : No
		1 : Yes
depthm_depth_quality_limit	2.5	[meters]
low_quality_std_w_depthm	0.02	[meters]
low_quality_std_Dz_depthm_bias	0.2	[meters]
low_quality_T_Dz_depthm_bias	200	[seconds]

5 MEMORY CONSIDERATIONS

Running NavLab can be very memory intensive, especially on long runs. And although you have lots of memory, MatLab requires large contiguous memory blocks to handle large data structures. If a run is longer than specified in `max_estimator_length` in `"OneClick.ini"`, the estimator is run several times to cover the whole length. The `max_estimator_length` parameter should be based on your hardware limitations.

5.1 Precautions

Before processing long runs, some steps can be taken to lower the probability of an 'Out of Memory' message from MatLab. More details can be found in MathWorks' "memory management guide" [4].

- Under Windows 32bit architecture, reserve 2GB of virtual memory through the control panel.
- If running MatLab R14 on Windows XP, use the /3G switch to allow applications to use 3 of the 4 GB memory address space. This setting is applied in the Windows file "boot.ini" located in the root directory of your Windows drive.
- Don't run other memory demanding applications simultaneously with NavLab, and if you have run many other applications lately, restart your computer, or use a memory defragmentation tool.
- Restart MatLab before each run.
- Start MatLab with the –nojvm option. This starts MatLab without java virtual machine, and frees approximately 250MB extra free memory. (R14)
- If you have the option, choose a computer equipped with a CPU (processor) with much cache, or better memory management. NavLab has been shown to perform very well on:
    - Pentium M **dothan** laptop processors with 2MB cache.
    - AMD Opteron processors

NavLab runs under 64 bit MatLab for Linux, where memory constraints are not a problem as the amount of memory address space is squared.

6 DESCRIPTION OF NAVLAB ONECLICK

This chapter gives a description of what the script "OneClick.m" does, and some background as to why it does what it does. A log file called OneClick_log.txt containing all echoed information is created, and moved to the NavLab directory when it has been created.

6.1 Reading and writing .ini files

OneClick will use the "OneClick.ini" found first in the path, this enables the use of several copies, and starting OneClick from the appropriate location for different behaviour.
All parameters from "OneClick.ini" are read and stored in a struct. As both "preproc.m" and "estimator.m" clears the workspace, this struct is saved in a .mat file before running either of the two, and loaded back afterwards.

In both "preproc.ini" and "estimator.ini" there are many parameters specifying the level of automation, and general behaviour. These are written in each copy, hence any of these .ini files with updated settings can be put in OneClick's ini directory.

6.2 Creating directories and copying files

OneClick prompts for a run directory, in which it assumes a "NavP\NavData" directory exists with the navigation data. When using the default output directory, a "post" directory is created at the same level as "NavP" with subdirectory "NavLab", as shown in Figure 6.1. The navigation results will be stored in the "NavLab" directory and its "data" subdirectory. If there already exists a "NavLab" directory it is moved to a directory called "NavLab_YYYYMMDD-HHMM". (Where YYYYMMDD-HHMM is the date and time of creation.)

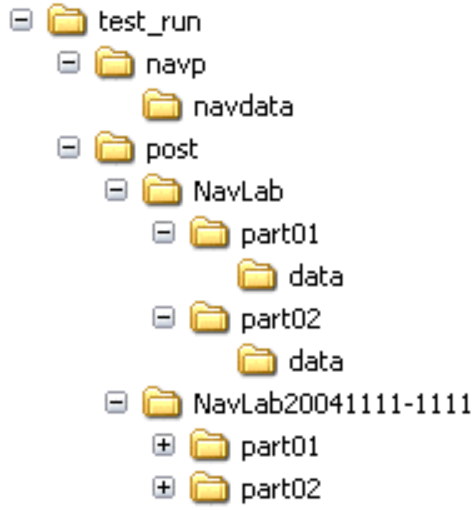

Figure 6.1 Directory tree

If NavData with sequential text files are missing, or OneClick cannot create them from the .log files, NavLab OneClick aborts with a short explanation. The three .ini-files, "preproc.ini",

"estimator.ini", and "cov_matrix.ini", are copied from the "ini" directory, and placed in the "NavLab" directory. In a split run, separate copies are made for each part and placed in the "Part01", "Part02"... directories.

6.3 Position wild point filter

Before sending the USBL measurements to the wild point filter, measurements not fulfilling the requirements in quality and relative AUV position described by parameters in "OneClick.ini", are removed.

A multi hypothesis filter [3] using up to five Kalman filters is used to identify and remove position wild points from the USBL measurements.

6.4 Pre-processing

The pre-processing is then done, with USBL position measurement if they exist, or without any position measurement if they don't. The pre-processing will plot several figures used for manual wild point editing and raw data inspection. If manual wild point editing is disabled in "OneClick.ini", these figures will be closed immediately after the pre-processed measurements have been saved to files. If you wish to inspect them, enable manual wild point editing.

6.5 Pre-processing vehicle GPS

If vehicle GPS measurements are available, and chosen to be used as aiding measurements, they are processed after the other sensors, as the NavLab pre-processing currently not supports more than one position measurement source.

Vehicle GPS position measurements are not wild point filtered, but measurements with too low reported accuracy are removed. The lever arms in "preproc.ini" are switched to vehicle GPS, and lever arm compensation is performed.

6.6 Merging position measurements from two sources

If both DGPS-USBL and vehicle GPS position measurements are available and chosen, they must be merged to a single measurement text file, as the estimator currently does not support more than one position measurement source. The two sensors will not produce good quality measurements simultaneously, thus they are well suited to be combined as a single measurement.

The vehicle GPS is considered the most reliable and accurate sensor of the two, and makes the basis for the new joint measurement data. All USBL measurements less than one minute away from any vehicle GPS measurement are discarded. The two are thereafter joined in one single text file.

As the two sensors have their own bias, and only one bias can be modelled in the estimator, a decorrelation in time is done at the last samples before switching between sensors.

6.7 DVL wild point filter

The measurement series are divided into overlapping intervals. An accept criterion is calculated based on a constant, the dynamics of the interval, and each individual elements distance in time from the median. If a measurements distance from the median is above this accept criterion in each interval it is a member of, the measurement is removed as a wild point. In Figure 6.2 measurements from a nine samples long interval are shown. The median is marked by a red circle, and the accept criterion drawn as black lines. The seventh measurement is seen to be outside the acceptance region.

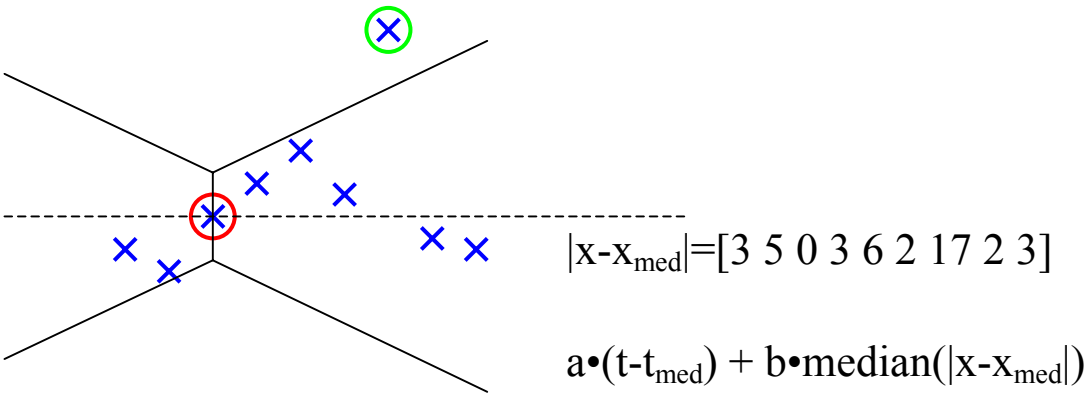

Figure 6.2 Wild point filter

The parameters used (a and b in the figure equations) are based on automatic tuning effort on some twenty real runs, where the number of wild points was predetermined by manual inspection. The values were chosen conservatively; to make sure no good values were cut, and accept a few possible wild points and doubt cases. It is measurements that have an error clearly not normally distributed,that is most critical to remove before running the Kalman filter, as they violate the assumptions guaranteeing optimality.

6.8 DVL start up requirements

When the DVL first gets bottom track, experience have shown it to be advantageous to have a stable measurement rate before using it as an aiding sensor. All DVL measurements before a period of minimum `DVL_interval_length` seconds with time between samples no longer than `DVL_max_diff` will be discarded. Both the interval length and the maximum difference are specified in `"OneClick.ini"`.

6.9 Choosing start and stop time

The start and stop times can be based on either position measurements or DVL measurements as chosen in `"OneClick.ini"`. If based on position measurements, the time of the first position measurement is chosen as start point, and the stop time is the time of the last recorded

IMU sample. When based on DVL measurements, the first and last DVL measurement times are start and stop time. If there are no position measurements before the first DVL measurement, the time of the first position measurement is used as start time, and if the there are DVL measurements after the last position measurement the time of the last position measurement is used as stop time.

If the run, due to memory considerations, is too long for a single estimator processing, the run is divided in several pieces. The split is made such that all parts have positions measurements both at the beginning and end. See chapter 0 for description of the parameters in `"OneClick.ini"` defining the division scheme. The start and stop times are stored, and `"estimator.ini"` is changed before each consecutive estimator run.

6.10 Varying sensor quality

Position measurements have time varying quality based on reported quality from the sensor itself and/or depth. This is implemented in preproc, and is also available for other sensors.

6.10.1 Depth measurement

The bias and white noise model of the depth measurement used in the Kalman filter is valid under water. When starting the navigation system on the deck of a ship, there is unmodelled behaviour in the pressure sensor. Specifying larger uncertainties above a certain depth level compensates for this.

6.10.2 DVL measurements

Experience show lowered DVL measurement quality when the AUV is in the surface zone. OneClick can switch between high and low sensor accuracy when AUV depth crosses the limit defined in `"OneClick.ini"`. Decorrelation in time when switching from large to low bias model is implemented.

6.11 Initial values

The script `get_sensor_values_for_init.m` is used to find initial values for the Kalman filter. This script interpolates between measured values if necessary. If there are no DVL measurements available at start time, zero is used, and the uncertainty described in `"cov_matrix.ini"` is raised to 2 m/s (1σ).

If the run has been divided, all but the first initial values are instead obtained from the real time data produced in the previous piece. A debug script is inserted to run in the estimator, picking out the state and covariance matrix at the desired start time of the next piece. The obtained initial values are written to `"estimator.ini"`.

6.12 Export

The workspace is saved in a .mat file named after the run directory. In split runs this is done after each estimator run, placing them in directories named `"part01"`, `"part02"`, etc. The smooth position, velocity, and attitude is exported and placed in the run directory in .txt files. If more than one estimator run was done, the data is merged to form a single long .txt file.

6.13 Quality check

The script `quality_check.m` performs a test on all estimated biases. If any of the real time or smoothed biases has a mean above three times the standard deviation used in the Kalman filter model, a warning is plotted. When a sensor has time varying bias specification, the statistical distance is calculated by dividing the estimated bias by the modelled bias. If the mean statistical distance is above three, the warning signals this. A warning will also be generated if the estimated position error has been above acceptable position accuracy at any time.

A short error report is also generated. This report tells the fraction of time the bias estimates was above three sigma, and gives the longest continuous interval with bias estimates above three sigma.

7 REFERENCES

[1] Gade, K (2003): NavLab - Overview and User Guide November 2003. FFI/RAPPORT-2003/02128

[2] Gade, K (2004): NavLab, a Generic Simulation and Post-processing Tool for Navigation. European Journal of Navigation, Volume 2, Number 4, November 2004, pp. 51-59.

[3] Jacobsen, Hans Petter (2004): NavLabPosFilter - Windows exe program to remove outliers.

[4] MathWorks, (2005): Memory Management Guide, http://www.mathworks.com/support/tech-notes/1100/1106.html

APPENDIX

A ABBREVIATIONS

AUV	Autonomous Underwater Vehicle
DGPS	Differential Global Positioning System
DVL	Doppler Velocity Log
IMU	Inertial Measurement Unit
USBL	Ultra Short BaseLine system