autoNomous, self-Learning, OPTImal and compLete Underwater Systems NOPTILUS

FP7-ICT-2009.6: Information and Communication Technologies

3rd Project Review

WP6 (Situation Understanding)

E. Orfanoudakis, N. Kofinas, and M. G. Lagoudakis
Telecommunication Systems Institute (TSI), Greece

November 27, 2014
Porto, Portugal

WP6: Situation Understanding

- **Event Recognition**
  - *events*: patterns in observation sequences
  - *observations*: raw sensor readings or estimated state
- **Models**
  - Probabilistic Context-Free Grammars (PCFGs)
- **Task 6.1: Event Recognition**
  - real-time, hierarchical parsing for on-line recognition
- **Task 6.2: Grammar Learning**
  - off-line learning of PCFGs from past AUV mission logs
- **Task 6.3: Integration**
  - on-line event recognition using learned PCFGs

WP6: Deliverables

- **Deliverable 6.1: Event Recognition**
  - hierarchical data stream parsing
  - on-line, PCFG-based event recognition
  - implementation within Dune

  3

- **Deliverable 6.2: Grammar Learning**
  - focus on normal vs. abnormal event recognition
  - off-line, Bayesian PCFG learning
  - exploitation of data from past AUV mission logs

  1

- **Deliverable 6.3: Integration**
  - standardized representation and configuration
  - documentation of software
  - experimentation

  2

An Example

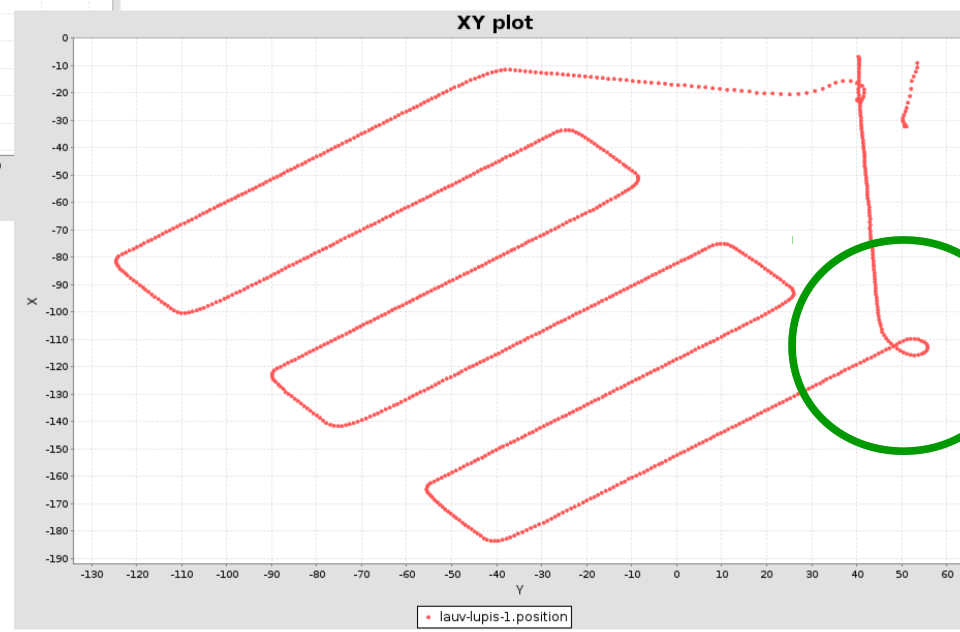
Learn to recognize abnormalities in AUV trajectories

AUV Normal and Abnormal Behavior

Normal
[Training]

May 2014

Abnormal
[Testing]



Learned Grammar on AUV Yaw

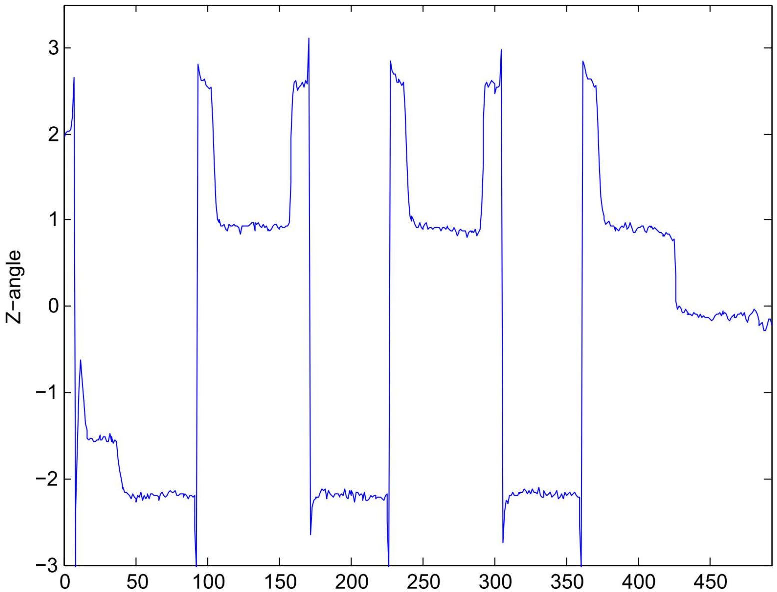
NOPTILUS

```
Start Rules:
N24
All Rules:
    N12 -> j (1)
    N13 -> k (1)
...
    N23 -> d (1)

    N24 -> N29 N29 (0.367347)
        -> N26 N26 N26 (0.27551)
        -> N27 N27 N27 (0.122449)
        -> N25 N25 (0.0612245)
...
    N25 -> N33 N14 (0.842105)
        -> N13 N19 N20 (0.157895)
    N26 -> N21 N21 (0.53795)
...
```

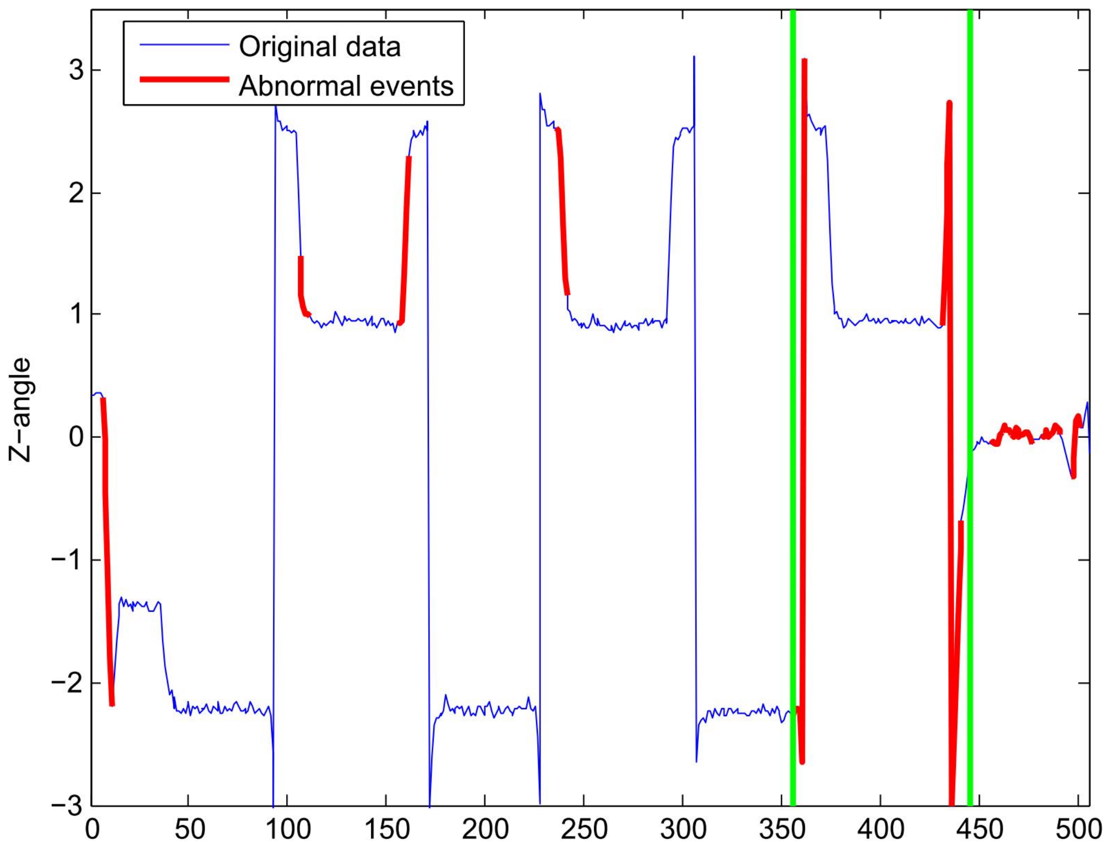
Recognition with Learned Grammar

- data stream: yaw (z-angle) of the AUV
- training corpus: 129 examples



Normal
[Training]

Abnormal [Testing]



PCFG Learning

Learning the complete structure of a grammar

Grammar Learning

- **Structured Prediction**
  - make a prediction about a structured object
- **Grammatical Inference**
  - infer a PCFG (symbols, rules, probabilities) from words
- **Training Data**
  - corpus of positive (normal) examples only
  - must generalize (but not too much) and must not overfit
- **Challenges**
  - the space of grammars cannot be generated systematically
  - the space of grammars is not a vector space
  - the neighborhood of a grammar is hard to define
  - cannot tell if the best possible grammar has been reached

Our Bayesian Approach

NOPTILUS

- **Learning objective**
  - given a corpus O, find G* that maximizes the posterior

$$G^* = \arg\max_G P(G|O)$$

- **Bayes Rule**

$$G^* = \arg\max_G \frac{P(G)P(O|G)}{P(O)} = \arg\max_G P(G)P(O|G)$$

- **Prior of G**

$$P(G) = \frac{1}{2^{|G|}}$$

- **Likelihood of O over G**

$$P(O|G) = \prod_{w \in O} P(w|G)$$

PCFG Initialization (Example)

NOPTILUS

$$O = \left\{ \begin{array}{cccccccc} a & b & c & c & c & c & & \\ a & a & a & b & c & & & \\ a & a & b & b & b & c & & \\ a & a & b & b & c & c & & \\ a & a & a & b & c & c & c & c \\ a & a & a & a & a & a & b & c \\ a & a & b & b & b & c & & \end{array} \right\}$$

$$G_{init} = (V, \Sigma, R, P, S)$$

$$RP = \left\{ \begin{array}{cclllllllll} S & \rightarrow & N_1 & N_2 & N_3 & N_3 & N_3 & N_3 & & & (1) \\ & | & N_1 & N_1 & N_1 & N_2 & N_3 & & & & (1) \\ & | & N_1 & N_1 & N_2 & N_2 & N_2 & N_3 & & & (2) \\ & | & N_1 & N_1 & N_2 & N_2 & N_3 & N_3 & & & (1) \\ & | & N_1 & N_1 & N_1 & N_2 & N_3 & N_3 & N_3 & N_3 & (1) \\ & | & N_1 & N_1 & N_1 & N_1 & N_1 & N_1 & N_2 & N_3 & (1) \\ N_1 & \rightarrow & a & & & & & & & & (19) \\ N_2 & \rightarrow & b & & & & & & & & (12) \\ N_3 & \rightarrow & c & & & & & & & & (14) \end{array} \right\}$$

Chunk and Merge Operations

➢ Chunk

- creates a new non-terminal to replace a sub-sequence

$$\begin{array}{lcllllll} N_1 & \rightarrow & a & a & a & b & c & & (15) \\ N_2 & \rightarrow & a & b & c & a & a & & (22) \\ N_3 & \rightarrow & d & d & a & b & c & a & (9) \end{array} \Longrightarrow \begin{array}{lcllllr} N_1 & \rightarrow & a & a & N_4 & & (15) \\ N_2 & \rightarrow & N_4 & a & a & & (22) \\ N_3 & \rightarrow & d & d & N_4 & a & (9) \\ N_4 & \rightarrow & a & b & c & & (46) \end{array}$$

➢ Merge

- combines two existing non-terminals into one

$$\begin{array}{lclllllr} N_1 & \rightarrow & a & N_2 & a & N_3 & c & (15) \\ N_2 & \rightarrow & a & N_1 & a & N_3 & d & (6) \\ N_3 & \rightarrow & d & N_2 & d & c & d & (32) \end{array} \Longrightarrow \begin{array}{lclllllr} N_1 & \rightarrow & a & N_2 & a & N_2 & c & (15) \\ N_2 & \rightarrow & a & N_1 & a & N_2 & d & (6) \\ & | & d & N_2 & d & c & d & (32) \end{array}$$

Grammar Search Strategy

Local Search

- Best-First Search strategy
- Beam Search strategy
  - parameters: beam depth and width
  - benefit: escape from local minima

Posterior Gain

- incremental computation over previous grammar
- likelihood gain and prior gain (may be negative!)

Recalculation of Probabilities

- merge operation invalidates the counts of the rules
- fix: Inside-Outside algorithm over the corpus
- pruning of highly unlike rules

Effectiveness and Efficiency

Efficiency

- set a maximum chunk size
- goal: limit the number of possible chunks

Numerical stability

- representation of log probability

$$G^* = \arg\max_G P(G|O) = \arg\max_G \left(\log P(G) + \log P(O|G) \right)$$

Regularization

- normalization of counts of initial grammar
- regularization factor λ on the grammar prior

$$G^* = \arg\max_G \left(\lambda \log P(G) + \log P(O|G) \right)$$

Incremental Learning/Search

1. Copy the initial corpus to the primary corpus
2. Sort words in primary corpus from most to less frequent
3. Form first training batch using the first k distinct words
4. Initialize PCFG using the first training batch
5. Execute the search procedure
6. Parse the entire primary corpus with the resulting PCFG
7. Transfer all parsed words from primary to the secondary corpus
8. Recalculate probabilities using the secondary corpus
9. If primary is empty, terminate, otherwise form a new batch
10. Append productions to S in order to parse all words in batch
11. Go to Step 5

Textbook Grammar Learning

Language	Corpus size: 10	100	1000	10000
$a^n b^n, n \geq 1$	✓	✓	✓	✓
$a^n c b^n, n \geq 1$	✓	✓	✓	✓
$(ab)^n (cd)^n, n \geq 1$	✓	✓	✓	✓
Parenthesis(()())()	✓	✓	✓	✓
$(a + (a + (a + a))$	✓	✓	✓	✓
$wcw^R, w \in \Sigma\{a, b\}$	OG	✓	✓	✓
$ww^R, w \in \Sigma\{a, b\}$	OF	✓*	✓*	✓*
$a^{2n} b^n, n \geq 1$	✓	✓	✓	✓
$c^n (a\|d)^n, n \geq 1$	✓	✓ $(\lambda = 0.4)$	✓ $(\lambda = 0.3)$	✓ $(\lambda = 0.4)$
$c^n (a\|d)^n a^{2m} b^m, n, m \geq 1$	OF	OG	OG	OG

Integration

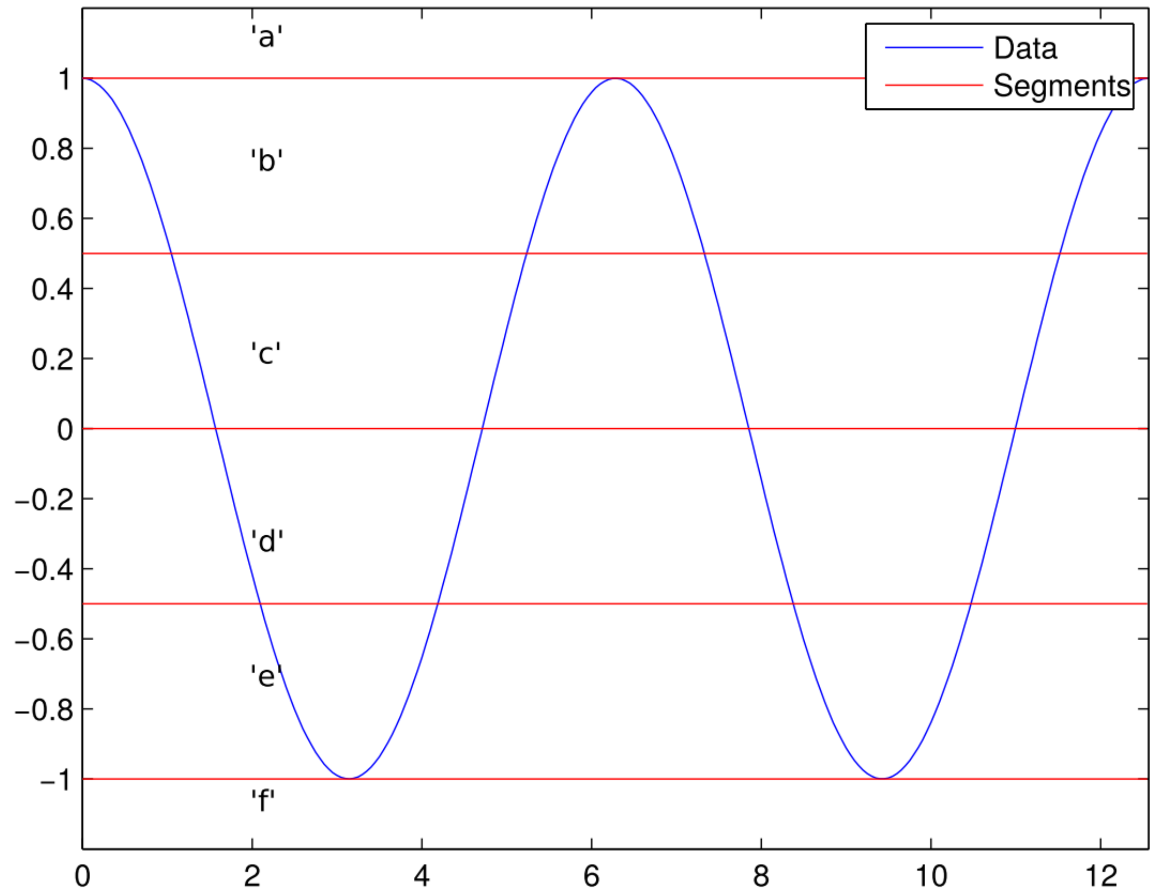
From learning a grammar to parsing in Dune

Grammar Configuration File

$$G = (V, \Sigma, R, S, P)$$
$$V = \{S, A, B\}$$
$$\Sigma = \{a, b\}$$
$$RP = \{S \rightarrow ASB\ (0.2),\ S \rightarrow AB\ (0.8),\ A \rightarrow a\ (1.0),\ B \rightarrow b\ (1.0)\}$$

```
4                  % The id of the start symbol S
2                  % Number of Terminal Symbols
0 2                % 'a' -> 0  A -> 2
1 3                % 'b' -> 1  B -> 3
2 1                % The representation of A -> 'a' [one production]
1 0 1.0 1          % "One symbol" "'a'" "probability" "Has terminal? 1"
3 1                % The representation of B -> 'a' [one production]
1 1 1.0 1          % "One symbol" "'a'" "probability" "Has terminal? 1"
4 2                % The representation of S -> ASB  AB [two productions]
3 2 4 3 0.2 0      % "Three symbols" "A" "S" "B" "probability" "Has terminal? 0"
2 2 3 0.8 0        % "Two symbols" "A" "B" "probability" "Has terminal? 0"
```

Quantization Configuration File




```
7
-inf
f
-1.0
e
-0.5
d
-0.0
c
0.5
b
1
a
inf
```

Dune Task Configuration File

```
[Monitors.GrammarParser/PCFG]
#Simulation, Always, Hardware
Enabled                 = Simulation
Entity Label            = PCFG
Execution Frequency = 5
#Which messages are used as inputs,
#must be in the correct order
Bind to =     ParserOutput.PCFG
#  Depth
#  EulerAngles.Pitch
Quantization = parser/SampleQuant.txt
Grammar = parser/SamplePCFG.txt
NormalAbnormal = true
Normal Threshold = 1e-5
Window Size = 8
Rolling Window = true
```

WP6 Code Repositories

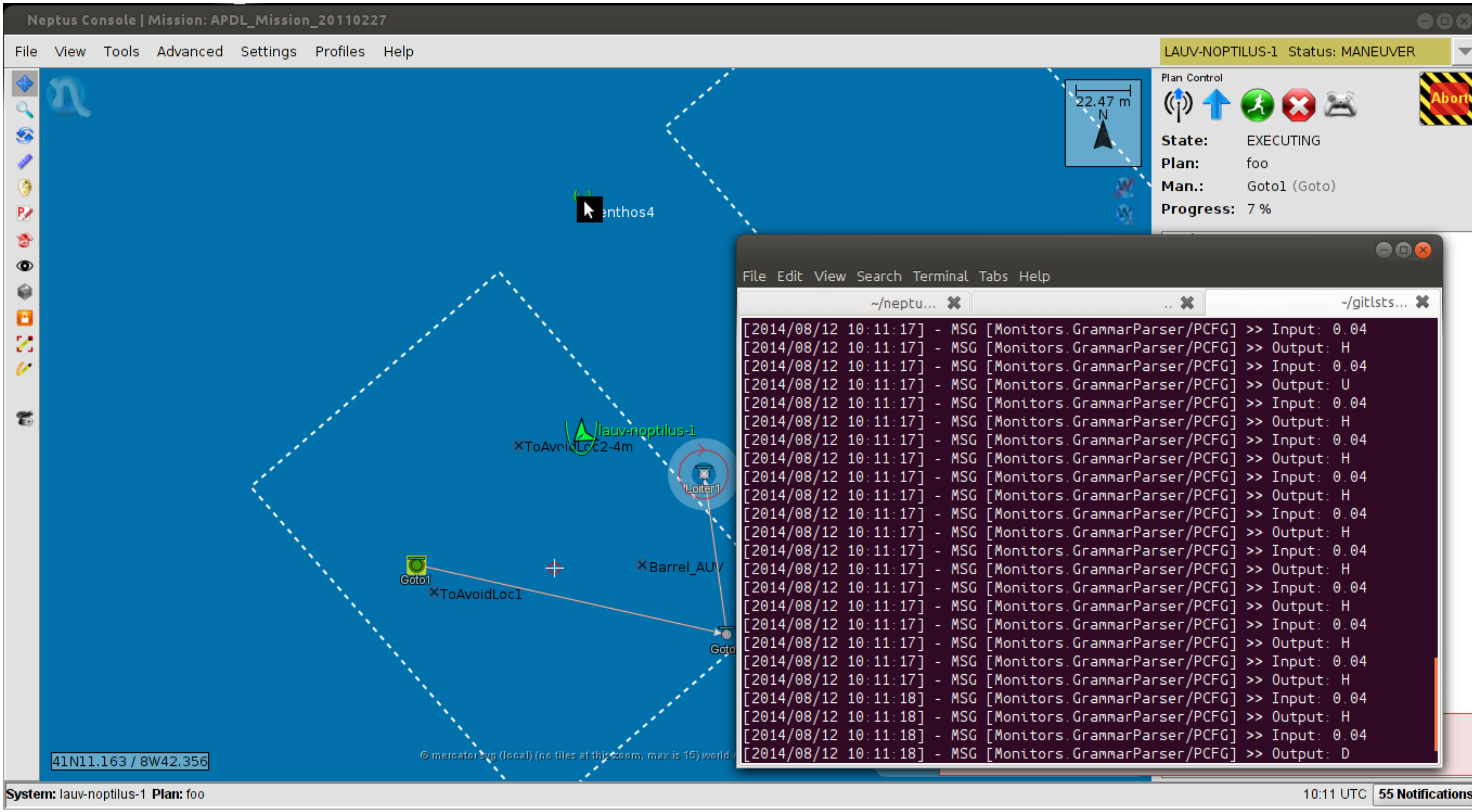
NOPTILUS

- **Off-line learning**
  - **`https://github.com/eldr4d/CFG-learner`**
  - stand-alone software for grammar learning
  - C++ code for data quantization and PCFG learning
  - two braches: **`master`** and **`NOPTILUS`**
  - includes the inside-outside algorithm implementation
- **On-line parsing**
  - **`https://github.com/vosk/dune`**
  - fork of the main LSTS-Dune repository
  - C++ Dune activity for on-line, hierarchical parsing
  - IMC input/output messages
  - customization via configuration files

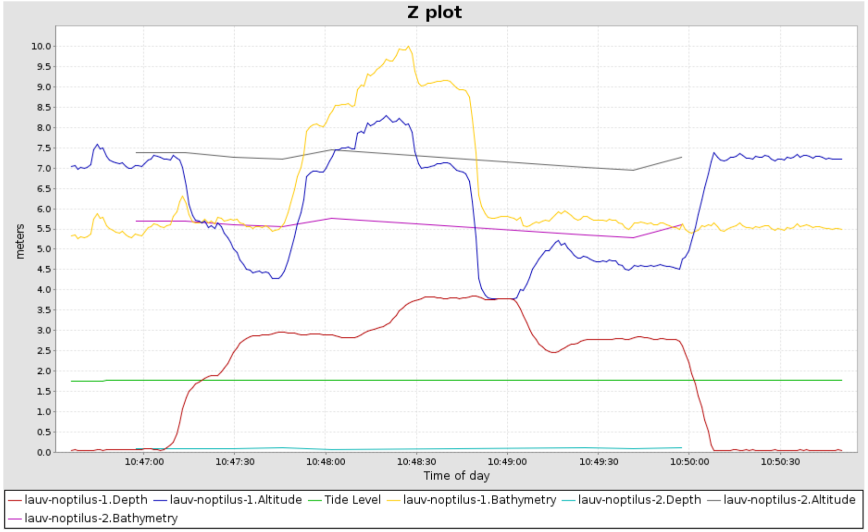
Event Recognition

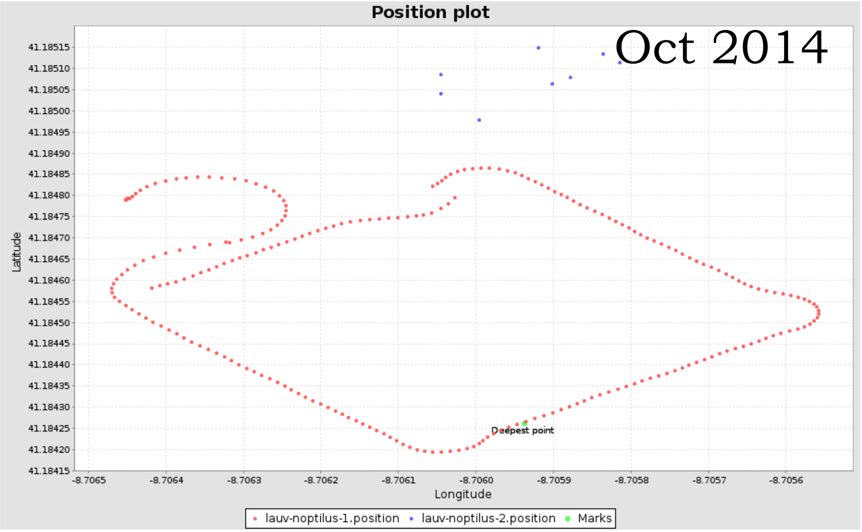
Putting everything together …

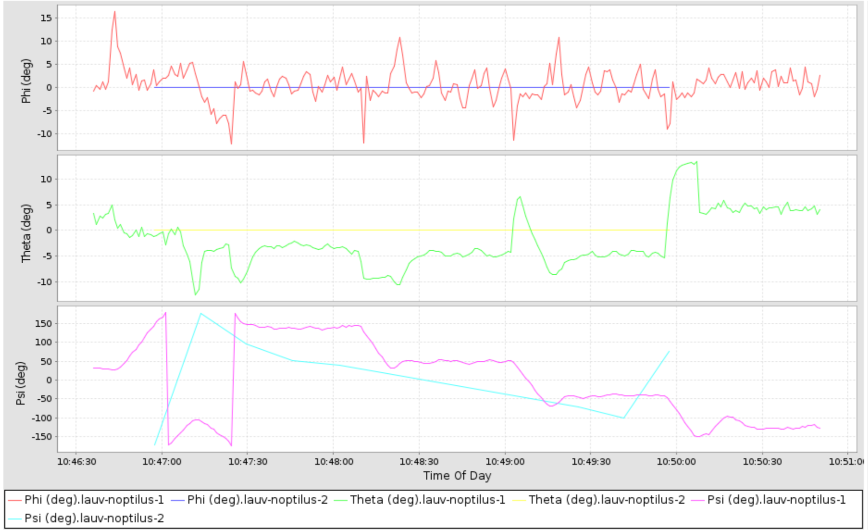
Dune+Neptus+Parser

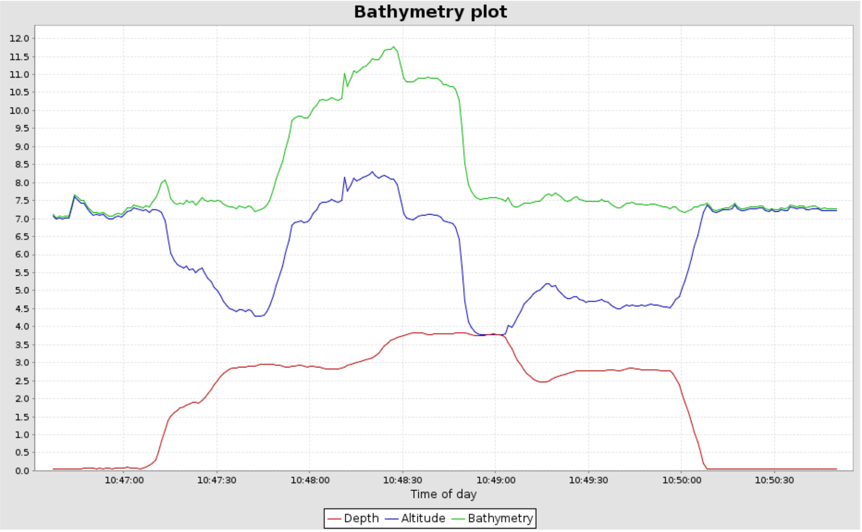


Mission with Normal Events

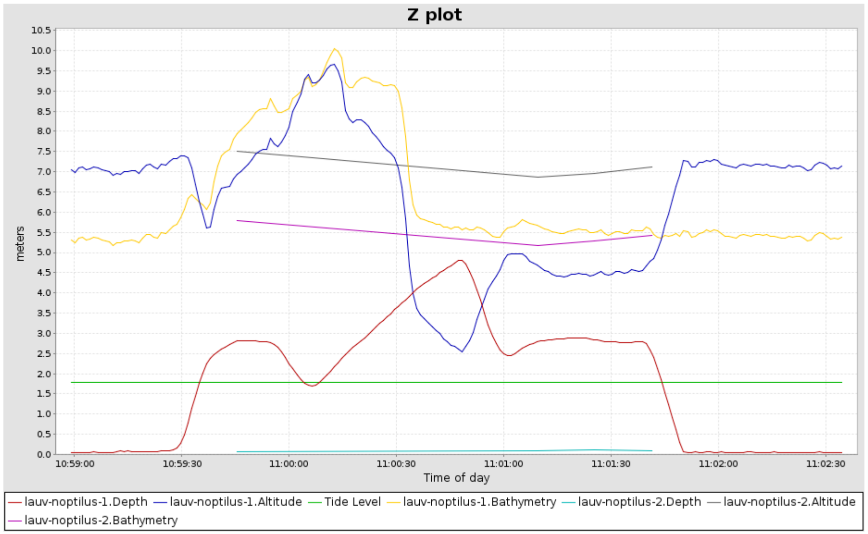


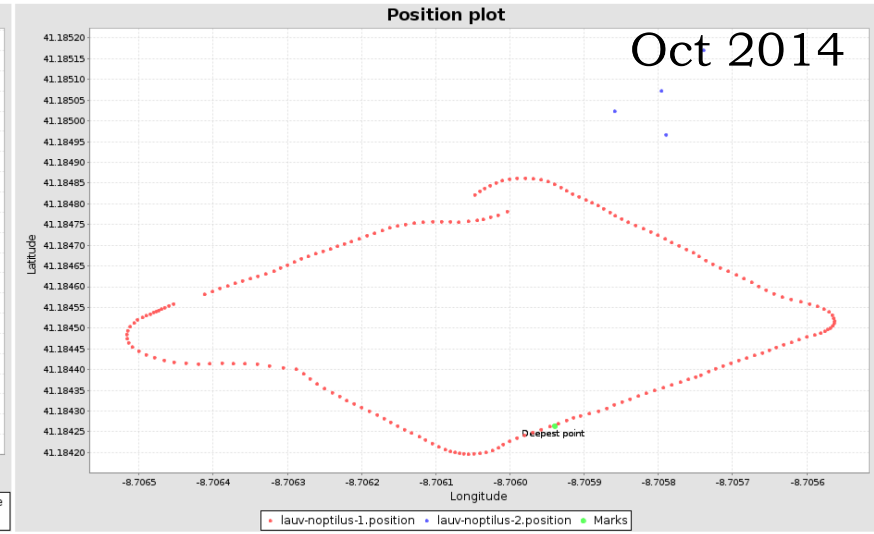


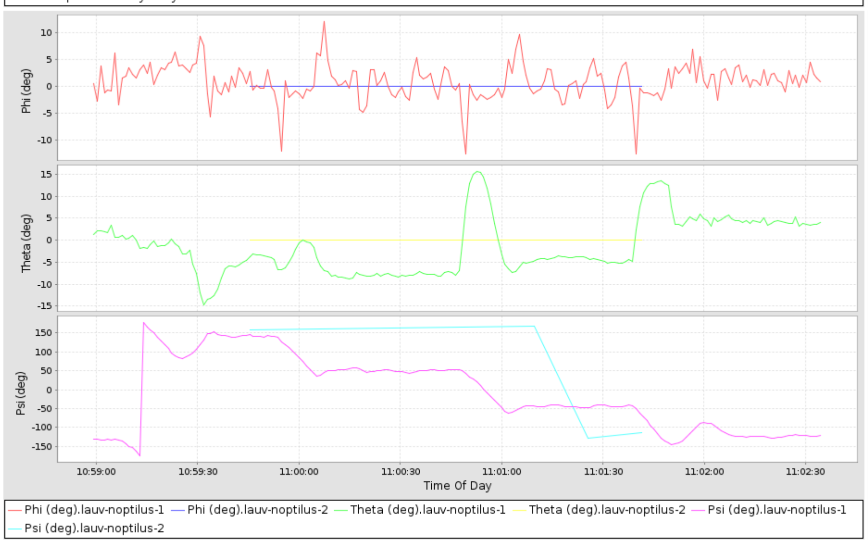


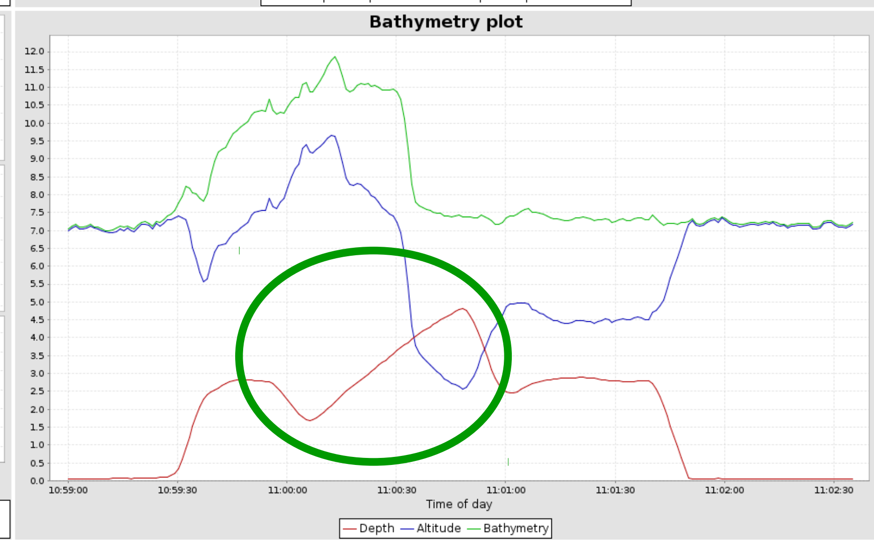


Mission with Abnormal Event

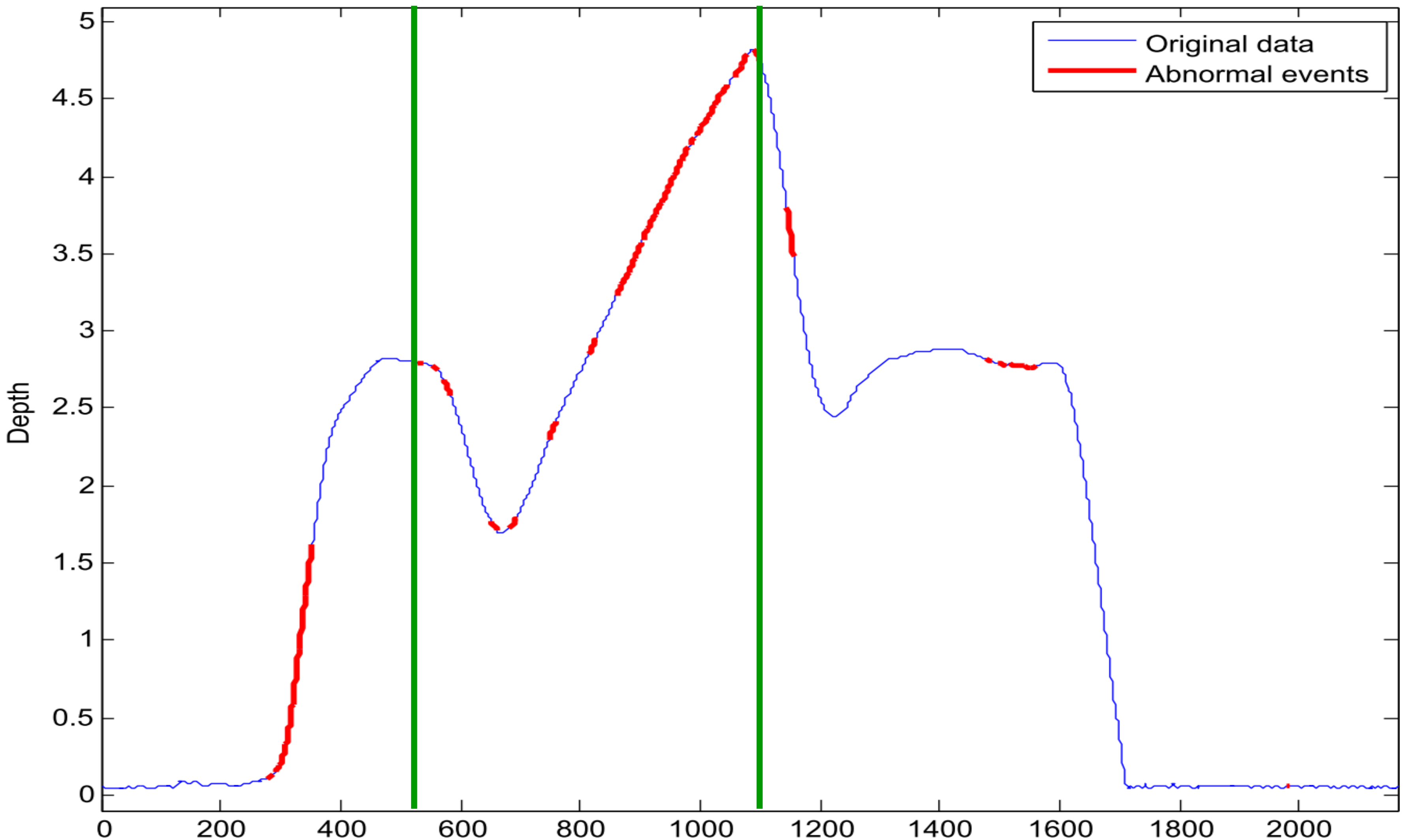




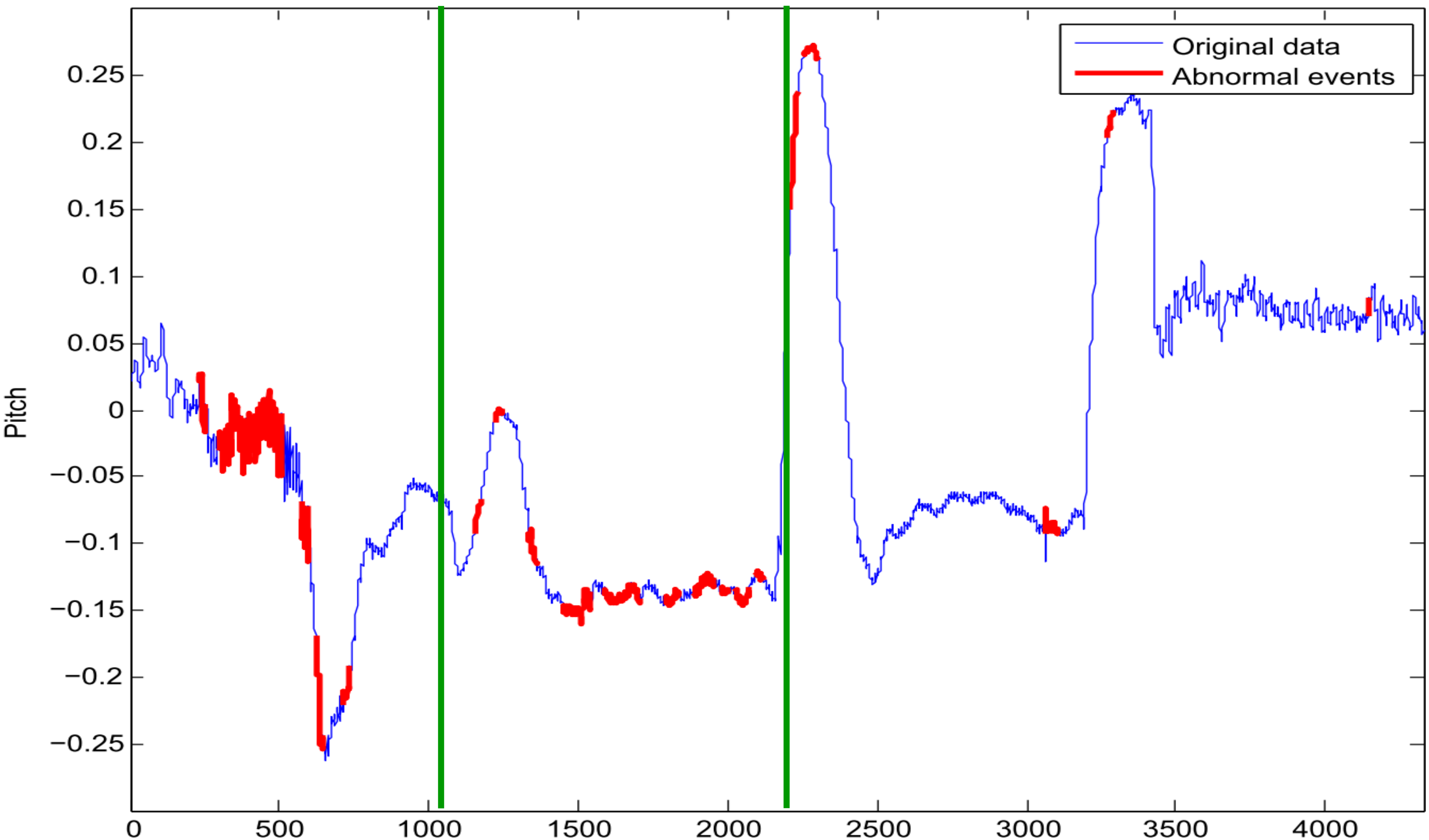




Recognition with Depth Data



Recognition with Pitch Data



WP6 Concluding Plans

What's left? What's next?

WP6 Next Steps

- **Software**
  - fix compatibility with current version of Dune
  - add a graphical user interface in Neptus
  - test performance of parser(s) onboard the AUVs
- **Experimentation**
  - run real AUV missions with event recognizers active
  - test learning and recognition on other data of interest
- **Outreach**
  - M.Sc. Thesis, *Event Recognition via Grammatical Parsing*, E. Orfanoudakis, ECE, Tech Univ of Crete, exp. Jan 2015.
  - M.Sc. Thesis, *Grammatical Inference for Event Recognition*, N. Kofinas, ECE, Tech Univ of Crete, July 2014.
  - publication plans: 2 conference, 1 journal papers

Thank you!

NOPTILUS