CC-Log: Drastically Reducing Storage Requirements for Robots Using Classification and Compression

Santiago Gonzalez, Vijay Chidambaram, Jivko Sinapov, and Peter Stone

University of Texas at Austin
The Problem

• Robots have a growing number of increasingly sophisticated sensors

• Roboticists want to leverage this data to gain insights into system behavior

• High sampling rates and limited storage

• Storing everything is infeasible

  • Have to let something go
Can we build a system to log only the data we need?
Can we build a system to log only the data we need?

+ figure out what data we need?
CC-Log
A modular, event-centric logging solution for ROS.

- Uses ML to decide whether saving data is required
- Greatly reduced logging storage requirements
- Lossless; fine grained sampling for logged events
- Fits into ROS’s modular architecture
Outline

• Background

• The CC-Log system

• Evaluation

• Systems challenges in robotics

• Concluding remarks
Outline

• Background
  • The CC-Log system
• Evaluation
• Systems challenges in robotics
• Concluding remarks
BWIBot

- Building-Wide Intelligence
- Autonomous, mobile robots
- Roam for hours on a single charge
- Controlled by a PC running ROS (Robot Operating System)
Robot Operating System (ROS)
Robot Operating System (ROS)
Robot Operating System (ROS)
Robot Operating System (ROS)
Nodes, topics, and messages?

* from simulation
What does data look like?

ROS /odom topic

header:
  seq: 5229
  stamp:
    secs: 57
    nsecs: 530000000
  frame_id: odom
child_frame_id: base_footprint
pose:
  pose:
    position:
      x: 14.9999999995
      y: 110.0
      z: 0.0
    orientation:
      x: -3.50379416134e-07
      y: -2.89561146542e-05
      z: 7.86406532897e-09
      w: 0.999999999581
  covariance: [1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001]
twist:
  twist:
    linear:
      x: -3.55271378053e-12
      y: -6.45947936005e-12
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 1.08357767203e-10
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
What does data look like?

ROS /odom topic

```json
{"twist": {"twist": {"linear": {"y": -5.167583477804464e-12, "x": -3.5527137587950676e-12, "z": 0.0}, "angular": {"y": 0.0, "x": 0.0, "z": 1.114260199260157e-10}}, "covariance": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "header": {"secs": 55, "nsecs": 84000000, "seq": 5007}, "pose": {"pose": {"position": {"y": 109.99999999973956, "x": 14.999999999504467, "z": 0.0}, "orientation": {"y": -2.8818053449111213e-05, "x": -3.4870814337234784e-07, "z": 7.729987484413655e-09, "w": 0.999999995846992}, "covariance": [1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "seq": 5007}}}
```
Position Over Time

- x
- y
- z

Pose Position

Time

t = 0s

t = 63s
Orientation Over Time

- t = 0s
- t = 63s
Linear Twist Over Time

![Linear Twist Over Time Graph](image)

- **t = 0s**
- **t = 63s**
Angular Twist Over Time

- $t = 0s$
- $t = 63s$
Outline

• Background

• The CC-Log system

• Evaluation

• Systems challenges in robotics

• Concluding remarks
CC-Log
Classification and Compression

• Use a machine learning classifier to determine whether the system is currently in an anomalous state

• Anomalies trigger logging of a window of data extending into the past and into the future

• Saved data is compressed to achieve further space savings
Window Sampling

- Log Window provides flexible set of samples to log
- Sliding Window provides fixed set of samples for analysis
Window Sampling

- Log Window can grow as samples are deemed anomalous using history in Sliding Window

- How do we know if a sample is anomalous?
Anomaly Classifier

• Want to determine if a datapoint is an outlier along a set of dimensions
  • 100s to 1,000s of dimensions

• Anomaly detection has been used to great effect in numerous areas (e.g., structural integrity monitoring)

• CC-Log uses a 1-class RBF-SVM
Support Vector Machine (SVM)

- Find a maximally separating hyperplane between two sets of linearly separable data
Support Vector Machine (SVM)

• Find a maximally separating hyperplane between two sets of linearly separable data
Radial Basis Function (RBF) SVM

- **The Kernel Trick:** Find a separating surface between two sets of data by embedding into a higher dimensional implicit feature space.
1-class RBF-SVM

Novelty Detection

- learned frontier
- training observations
- new regular observations
- new abnormal observations

error train: 19/200 ; errors novel regular: 3/40 ; errors novel abnormal: 0/40

graphic from Scikit-learn
CC-Log Operation

1. Full logging
2. Offline learning
3. Intelligent logging
CC-Log Architecture

Contained within ROS

Segbot System
- Sensors
- ROS Nodes
- Actuators

Record Node
- Sliding Window
  - Topic Callback
  - Build Feature Vec.
  - Anomaly Detector
  - Window Trigger
  - Validator
  - Data Formatter

Storage
- Training Data
- Testing Data
- Logged Windows
- Continuous Log

Data Formatter

Build Feature Vec.
CC-Log Architecture

Segbot System
- Sensors
- ROS Nodes
- Actuators

Record Node
- Sliding Window
- Build Feature Vec.
- Topic Callback
- Window Trigger
- Data Formatter

Anomaly Detector

Validator

Storage
- Training Data
- Testing Data
- Logged Windows
- Continuous Log

Note: Contained within ROS
CC-Log Architecture

Contained within ROS

Segbot System
- Sensors
- ROS Nodes
- Actuators

Record Node
- Sliding Window
  - Topic Callback
  - Build Feature Vec.
  - Window Trigger
- Build Feature Vec.
- Anomaly Detector
- Validator
- Data Formatter
- Data Formatter

Storage
- Training Data
- Testing Data
- Logged Windows
- Continuous Log
Full logging

Record Node

Sliding Window

Topic Callback

Build Feature Vec.

Build Feature Vec.

Window Trigger

Data Formatter

Anomaly Detector

Validator

Data Formatter

Training Data

Testing Data

Logged Windows

Continuous Log

Contained within ROS

Sensos

ROS Nodes

Actuators

Segbot System

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes

ROS Nodes
Offline learning

Record Node

Sliding Window

Build Feature Vec.

Anomaly Detector

Validator

Data Formatter

Data

Topic Callback

Build Feature Vec.

Window Trigger

Contain within ROS

Segbot System

Sensors

ROS Nodes

Actuators

Storage

Training Data

Testing Data

Logged Windows

Continuous Log
Intelligent logging

Contained within ROS

Segbot System

- Sensors
- ROS Nodes
- Actuators

Record Node

- Sliding Window
- Topic Callback
- Build Feature Vec.
- Window Trigger

- Build Feature Vec.
- Anomaly Detector
- Validator
- Data Formatter

Data Formatter

Storage

- Training Data
- Testing Data
- Logged Windows
- Continuous Log
Implementation

• Dependency and setup challenges

• VM used extensively

• Tricky to get system fully integrated into ROS

• Collecting data proved to be arduous
Outline

- Background
- The CC-Log system
- Evaluation
- Systems challenges in robotics
- Concluding remarks
Simulation

- Robot is shared resource, need lots of data
- Full featured simulation within ROS, based on Gazebo
- Different notions of nominal behavior, subset of reality
- Can’t simply train in simulation and test on physical robot
  - Domain adaptation outside of project scope
Simulation
In Silico Classifier Accuracy

Training: 983 nominal
Testing: 492 nominal, 20 anomalous
In Silico Classifier Accuracy

Training: 983 nominal
Testing: 492 nominal, 20 anomalous


**In Silico Classifier Accuracy**

Training: **983 nominal**  
Testing: **492 nominal, 20 anomalous**

<table>
<thead>
<tr>
<th>Total Events</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positives</td>
<td>20</td>
</tr>
<tr>
<td>False Positives</td>
<td>183</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0</td>
</tr>
<tr>
<td>True Negatives</td>
<td>309</td>
</tr>
</tbody>
</table>
In Silico Classifier Accuracy

Training: **983** nominal
Testing: **492** nominal, **20** anomalous

<table>
<thead>
<tr>
<th>Total Events</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positives</td>
<td>20</td>
</tr>
<tr>
<td>False Positives</td>
<td>183</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0</td>
</tr>
<tr>
<td>True Negatives</td>
<td>309</td>
</tr>
</tbody>
</table>
## Compression Schemes

### Compression Effectiveness

<table>
<thead>
<tr>
<th>Compression</th>
<th>File Size</th>
<th>Compression Ratio</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>4.6 MB</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>LZ4</td>
<td>596 KB</td>
<td>13.0%</td>
<td><code>lz4c -9</code></td>
</tr>
<tr>
<td>LZFSE</td>
<td>479 KB</td>
<td>10.4%</td>
<td>Using open-sourced implementation</td>
</tr>
<tr>
<td>ZIP</td>
<td>463 KB</td>
<td>10.1%</td>
<td>Under macOS</td>
</tr>
<tr>
<td>TAR gzip</td>
<td>463 KB</td>
<td>10.1%</td>
<td><code>tar -cvzf</code></td>
</tr>
<tr>
<td>LZMA</td>
<td>329 KB</td>
<td>7.2%</td>
<td>Level 6 LZMA</td>
</tr>
</tbody>
</table>
# Compression Schemes

## Compression Effectiveness

<table>
<thead>
<tr>
<th>Compression</th>
<th>File Size</th>
<th>Compression Ratio</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>4.6 MB</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>LZ4</td>
<td>596 KB</td>
<td>13.0%</td>
<td>lz4c -9</td>
</tr>
<tr>
<td>LZFSE</td>
<td>479 KB</td>
<td>10.4%</td>
<td>Using open-sourced implementation</td>
</tr>
<tr>
<td>ZIP</td>
<td>463 KB</td>
<td>10.1%</td>
<td>Under macOS</td>
</tr>
<tr>
<td>TAR_gzip</td>
<td>463 KB</td>
<td>10.1%</td>
<td>tar -cvzf</td>
</tr>
<tr>
<td>LZMA</td>
<td>329 KB</td>
<td>7.2%</td>
<td>Level 6 LZMA</td>
</tr>
</tbody>
</table>
Limitations

- Currently tailored for odometry data
- Adapting to real robot requires lots of clean running data
- Cannot capture aggregate data
- Simple classifier cannot fully capture certain intricacies
  - Need more data
  - Could be better served by HMM or LSTM based model
Future Work

• Collect more data and fine tune the classifier
• Incorporate more types of data into the system
• Course-grained continuous logging
• Integrate compressive sampling, such as RTV
Outline

• Background

• The CC-Log system

• Evaluation

• Systems challenges in robotics

• Concluding remarks
Scheduling

- Robots require more nuanced scheduling
- Data generated at different speeds
- Different nodes need to process data at different rates
- ROS has very primitive scheduling
Lightweight Processes

- BWIBot has sluggish performance after some time
  - Many concurrent ROS nodes
- Each ROS node is a process
- ROS nodes are too heavy for long-running processes
Storage

• CC-Log solves one facet of the storage problem

• Other use cases may require stratified sampling to get aggregate statistics

• Security and privacy
Continuous Learning

• Want robots to be able to train models “on-the-go”

• Continuous learning poses unique challenges
  • Data requirements change over time
  • How much data is enough data?
Retrospective

- Tackled a problem in robotics from a systems perspective
- Simple techniques can be very powerful
- Robotics / systems collaborations are great
- Building a working system end-to-end in ROS is somewhat difficult, collaboration should ameliorate this