Robotics Observability

Scaling ROS from Prototype to Production

Adrian Macneil
Co-founder & CEO

foxglove
Robots have graduated from the lab
Perceptive Automata shuts down after funding dries up

Ford, VW-backed Argo AI is shutting down

Amazon Abandons Home Delivery Robot Tests in Latest Cost Cuts
- Slow-moving cooler-sized Scout had been in testing since
- Machine axed as Amazon adjusts to slower sales growth

Vertical farming robotics startup Fifth Season shuts down

Alphabet closes Everyday Robots among layoffs

What went wrong with Shopify’s quest to build a logistics business

Nuro plans for more layoffs as the AV sector’s economic woes deepen / The robot delivery company said it would pause production on its third-generation vehicle and scale back its operations.
The hardest part about robotics? Making it work

Does it operate effectively without intervention?

Can it deliver repeatable success?

Can it operate reliably at scale?
Success in production requires a different approach.

**Prototype**
- Get it working once
- SSH debugging
- Robots as pets

**Production**
- Reliability at scale
- Centralized logging
- Robots as cattle
What is Observability?

Your ability to determine the internal state of a system based on its observable outputs
Observability is understanding how your robots sense, think, and act.
Observing robots at scale is complex.

- Multimodal sensor data
- Semantic state
- Many devices
- Distributed facilities
- Limited bandwidth
The existing tools weren’t built for this

**Traditional robotics dev tools**
- RViz, Rqt, etc
- Support prototype development
- No cloud-based workflows
- Limited cross-platform support

**Server observability tools**
- Grafana, Datadog, etc
- Support logs/metrics/traces
- No multimodal data
- No support for edge recording

The existing tools weren’t built for this.
The four pillars of a robotics observability stack

Record | Upload | Process | Analyze
Categories of robotics data

Lightweight telemetry
- Pose, GPS, joint states, system state, etc
- Typically uploaded in near-real time
- Often useful for business insights / analytics

Downsampled sensor data
- Lower resolution or reduced framerate (e.g. 1hz front camera)
- Helpful for incident triage

Full resolution sensor data
- Raw camera images, lidar frames, etc
- Often necessary for debugging
- Can generate upwards of 1TB per hour!
Standardize log pipeline - avoid the "junk drawer" approach

- Don't record launch parameters to a separate .yaml file
- Don't stream logs to a .txt file - use /rosout and save in your bag
- Don't log video to a separate .mp4 file

ROS 1 -> .bag
ROS 2 -> .mcap

Recording files should be self-contained

- Everything necessary to reproduce internal state of the robot
- Simplifies parallel post-processing
- Guaranteed future accessibility
- Record server communication
- Don't forget latched topics!
File splitting
  ● Split based on time or file size (1 - 5 mins)
  ● Consider separate files for groups of topics
  ● Allows delete or upload discrete files - avoid further processing on-device

Rolling record
  ● ROS node or cron job
  ● Delete old files based on available disk space

Compression
  ● Saves disk space, but uses additional cpu/memory
  ● Chunk compression (lossless)
  ● Avoid whole-file compression
Bandwidth is always a problem

- Warehouse robots often limited to 10 - 100 Mbps per site
- Agricultural robots are often lucky to have internet at all
- But some robots can record over 1 Gbps
Post-process data in the cloud
- Transform sensor data from a proprietary message for visualization
- Regenerate deterministic data to save bandwidth
- ETL into other systems (data warehouse, time-series DB)

Key considerations
- Keep source data separate from processed data
- Parallel processing on self-contained files

Schema evolution
- Version robot code + post-processing code together
- Version robot code + post-processing code separately
Multimodal replay is table stakes
- 3d scene, URDF model, images, maps, plots, logs, etc
- High-level overview down to frame-by-frame debugging
- Web-based access saves hours of time
- Help people help themselves

Discovery
- Annotate events & incidents
- Review & triage workflow
- Categorize root causes or escalate to engineering
Business insights
- Task completion rate or success rate
- Identify misbehaving robots
- SQL Data Warehouse (e.g. Snowflake, BigQuery)

Time-Series Aggregations
- Calculate mean/median/percentile metrics
- Find and visualize outliers
- Time series database (e.g. Prometheus, InfluxDB)

Full text search
- Find instances of certain errors or exceptions
- Quick way to log unstructured data
- Full text database (e.g. Elasticsearch)
Additional considerations

Simulations
- Observability into simulations is equally important!
- Sims can be recorded/uploaded/analyzed the same as production data

AI/ML training
- Training is not part of observability, but often relies on the same data
- Incident triage can feed training dataset
- Version training datasets separately

Retention
- Often cheap to retain lightweight telemetry indefinitely
- Consider access patterns & storage budget
- Legal concerns
Thank you!

Adrian Macneil
Co-founder & CEO

foxglove