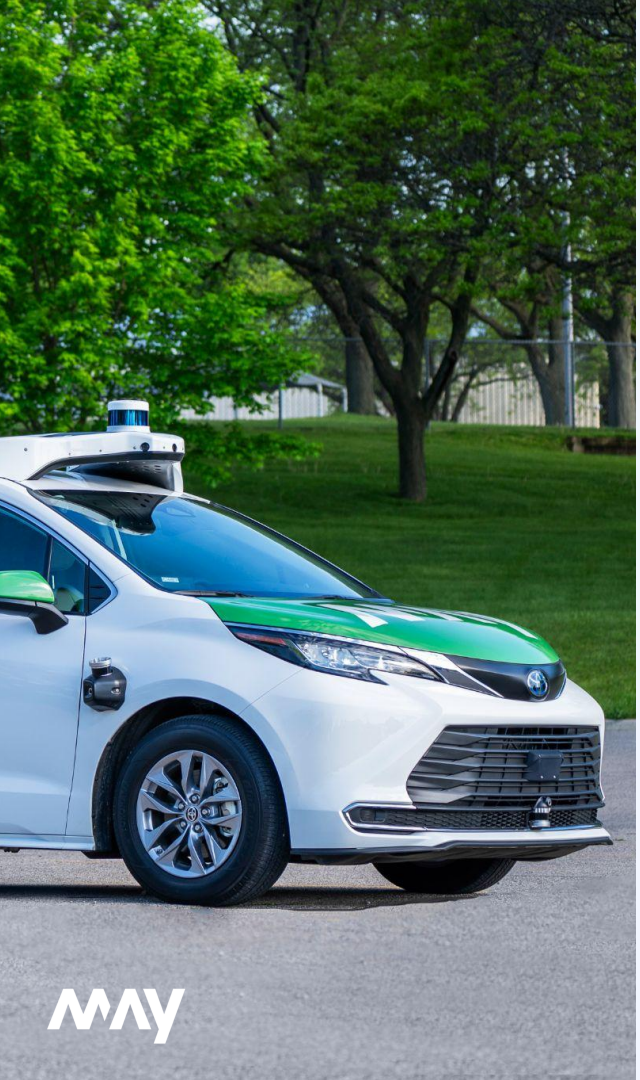
May mobility

Practical Data Science in Robotics using DataFusion

Timothy W. Saucer, Ph.D.

Dec 18, 2024

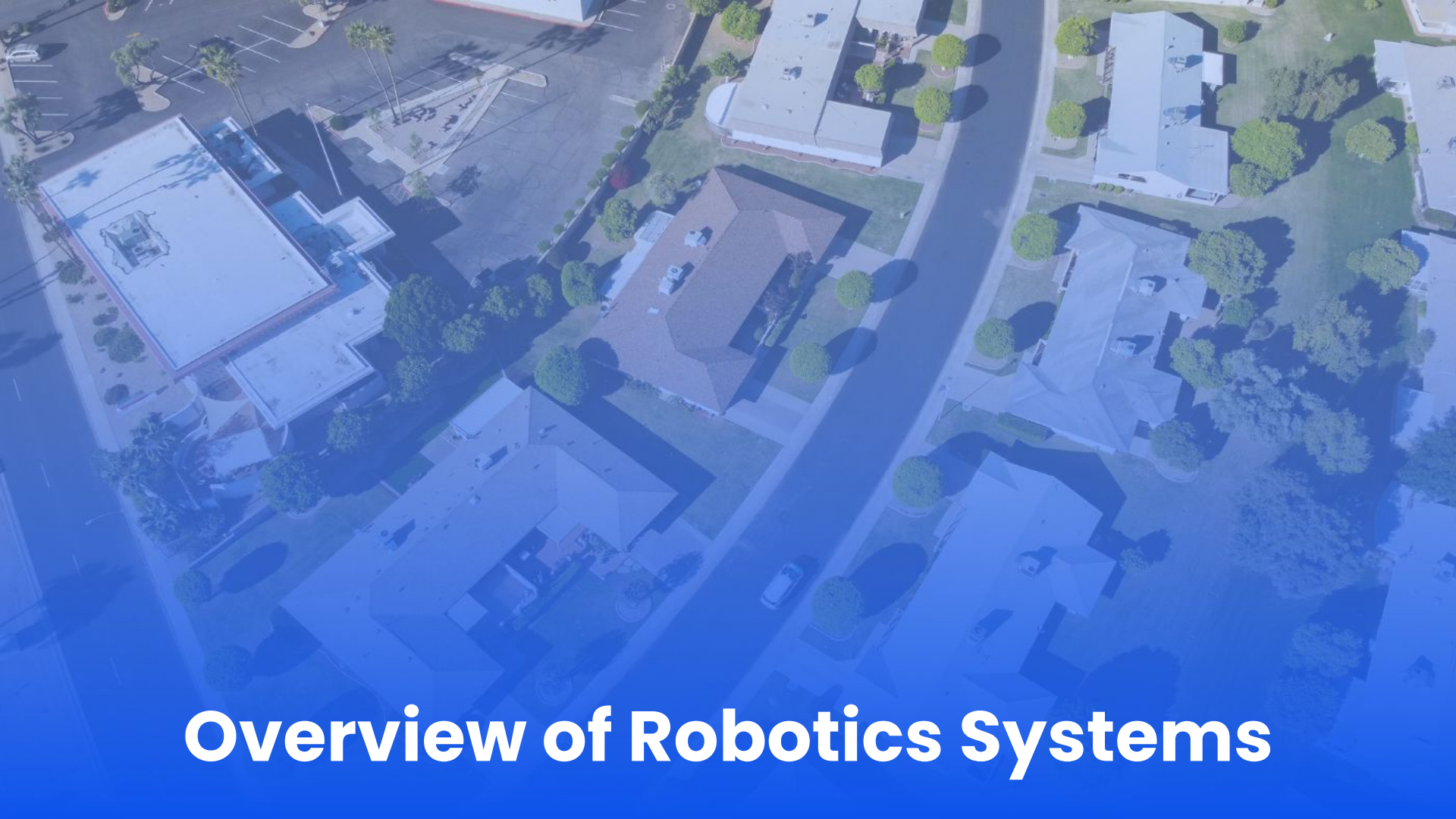


Agenda

1. Overview of robotics systems
2. Use Case Example
3. Common Problems and Approaches
4. Where DataFusion fits in

My background (briefly)

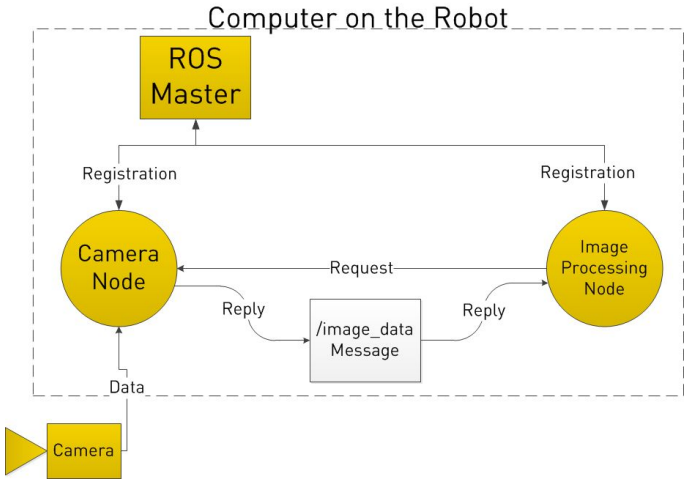
- May Mobility
  - Currently director for simulation and infrastructure
  - Heavily work in data engineering of features
  - Current main focus of using data to validate safety case
  - Prior: Autonomy engineer in decision making engine
- Soar Technology
  - Mostly worked on robotics projects for DARPA
  - Developed rust wrapper for cognitive architecture (soar)
- Dassault Systemes
  - Industrial robotics software (such as 6 DOF arms in an automotive plant)
- Ph.D. Physics - University of Michigan
  - Thesis used nature inspired search algorithms to optimize solutions in high dimensional parameter space
- US Navy - irrelevant, but some people say it's interesting



Overview of Robotics Systems

Robotics Systems

- Talk is focused on modern robotic systems
  - Industrial robots (like 6 DOF manufacturing arms) don't usually operate like this
- Typically consist of 1+ CPUs, 1+ GPUs, and multiple sensors
- Frequently a mixture of ethernet enabled devices and some connected to a single CPU
- All processes operating asynchronously, some with additional threading
- Communication is typically done through message passing infrastructure, such as ROS defined by:
  - Communication Layer
  - Message Definitions
  - Serialization Protocols
- Some processes work on a pub/sub approach, others on a request/response, and some a mixture

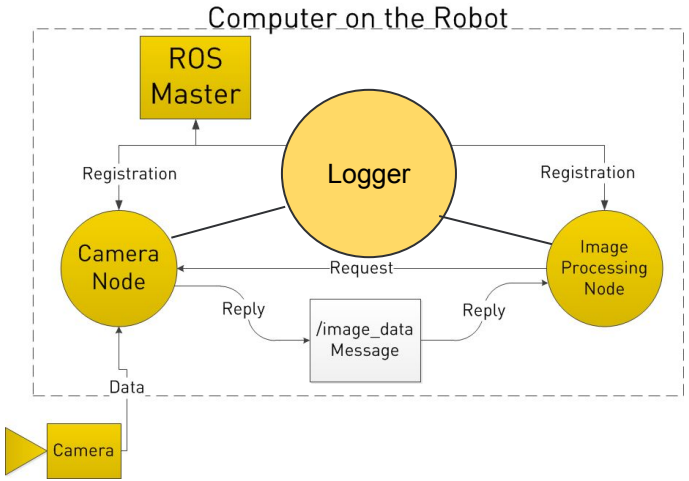


Typical node structure within a single CPU. Realistic systems often have multiple computers, each with tens to a hundred nodes.

Image courtesy Clearpath Robotics.

Data Offload

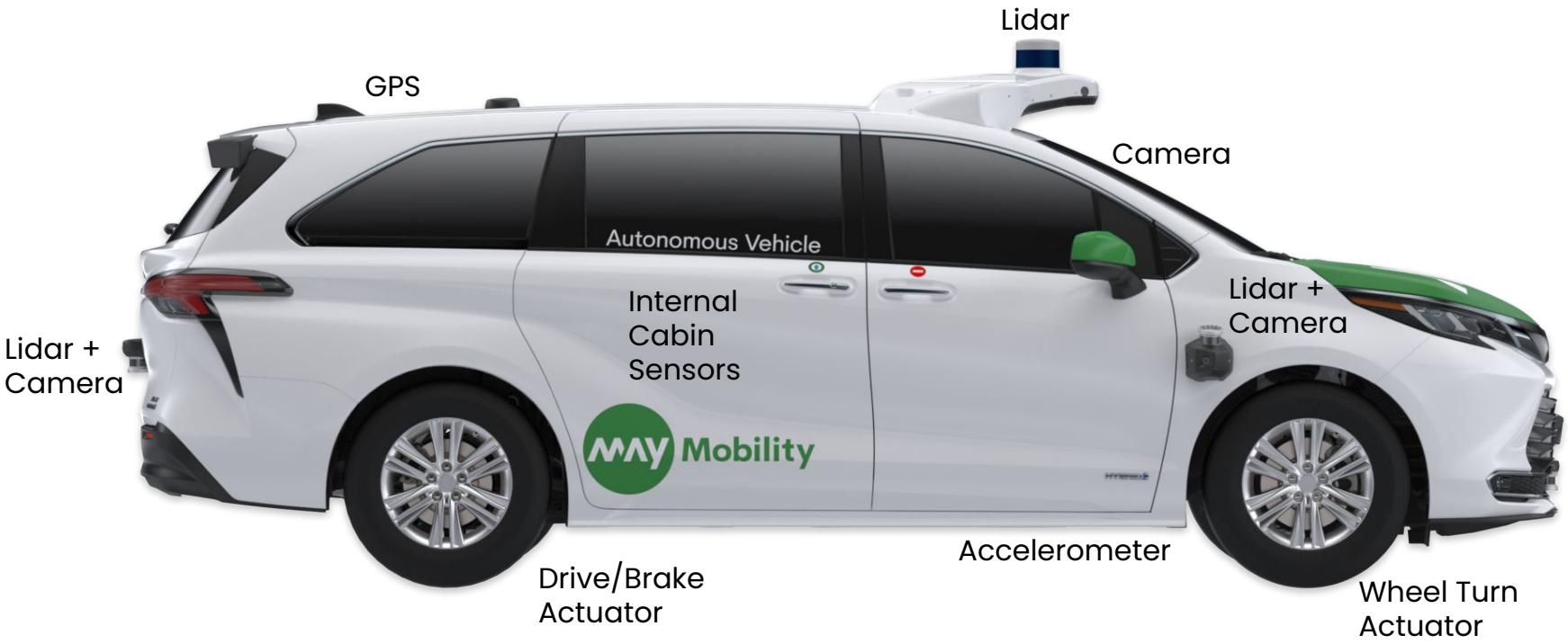
- Typically attach one or more nodes as loggers
  - This is a process that listens only and attempts to write the serialized data as fast as possible to a file(s)
- There *may* be no guarantees about the timing of messages
  - The current time for a process maybe:
    - A “trigger” based on a message received and it assumes that is the current time
    - An internal clock mechanism
    - A timing signal from an external source
- In the end you frequently have one or more log files that require some form of ingestion into your data analysis suite
  - Common log sizes are ~300-500 Gb/Hr of data



The logging node typically sits on one or more CPUs within the robotics system.

Image courtesy Clearpath Robotics.

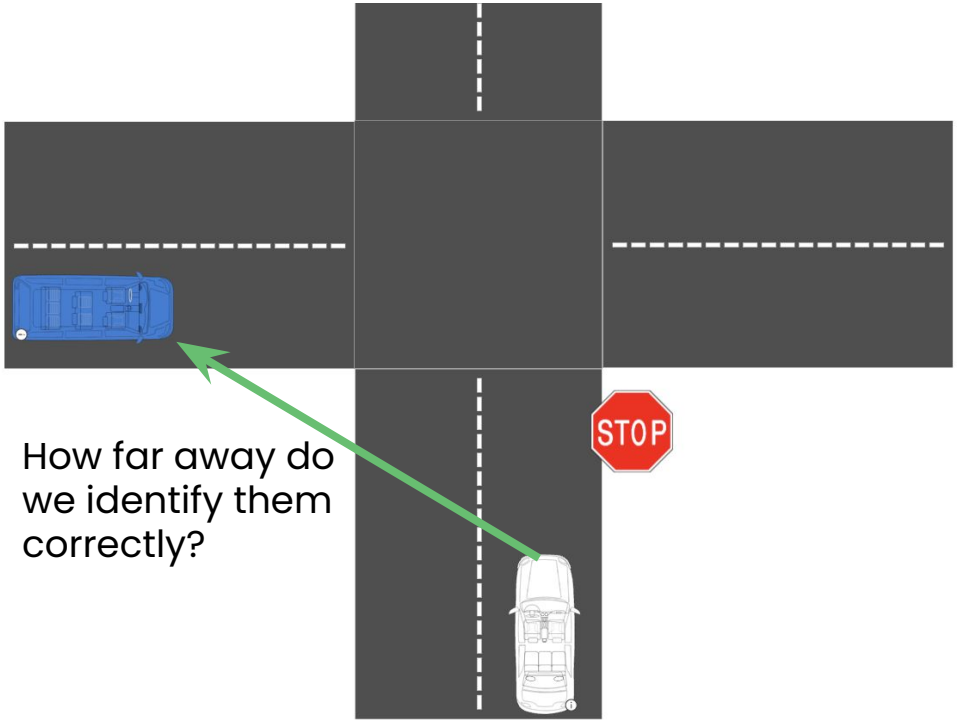
Example Robotic System



Use Case Example

Use Case: Cross Traffic Identification

- Define cross traffic identification as: For vehicles that are on the cross streets from ours, how far away they are from our vehicle when we get our first positive identification.
- Two useful, related metrics:
  - Distance from our vehicle to the entrance of the intersection when we get a positive ID X meters away.
  - When we are at the entrance to the intersection, the maximum distance away at which we get a positive ID.
- You may want to predicate these results on the type of vehicle (car, truck, bus, etc)
- Also useful is the distance of the first identification compared to the distance of the first positive ID
  - Far away tracks are hard to recognize



Cross Traffic - Technical Approach

- Identify distance from ego vehicle to the intersection
  - Combine GPS data with map data
    - Requires identification of what is defined as the “entry” of the intersection
- Identify the “real” classification of the other agent
  - Since you are not trying to process in real time, search for the agent’s maximum likelihood classification
    - **All robotics data are noisy**
- Identify the time range for which the classification is stable and the same as the maximum likelihood
- Compute summary statistics predicated on things like
  - Type of agent based on maximum likelihood
  - Differentiability (how certain can you be you got it right)
  - Particular intersection
    - Bonus points to identify intersections that are significantly different from the norm

when it comes to robotics data

Common Problems and Approaches

Common Robotics Challenges

- Noisy data - Loss of signal - process crash, buffer full, weak connection, etc
- Getting the data into a data store
  - This is basically “just” a traditional computer science problem of converting bits of type X into bits of type Y
  - Naive approaches are typically to hand encode these
  - In practice, often requires building generators to enable scalability
- Transforms!
  - Sensors are “dumb” and put out what they see
  - Define a transform to go from the reference point of the robot to the sensor
    - You may have multiple reference transforms between one and the other and these may change dynamically
      - Example: distance sensor on the end of a long arm with two joints along it
- Noisy data
- Spatial Analysis
  - This problem is more related to database approaches, not robotics in general
- Time Synchronization
  - Making sense of when the messages come in, and how to consistently apply timing information
- No, seriously, the data are noisy

Getting robotics data into a data store

- Robotics data are **highly** structured
- Two general approaches - sending messages via publish/subscribe or query/response
- Most difficult issue: How to convert the data into a usable format?
  - Should you try to ingest “raw”?
  - Can we automate the message translation process?
  - Where in your pipeline can/should you make the data more friendly to data analysis?
  - Trivial example: cartesian position (x, y, z) stored as a float[] in a message
    - We know *a priori* this is always a 3 element float, should we store this as a fixed length array element or should we store three columns pos_x, pos_y, pos_z?
- Real Time ingestion
  - Requires all messages to have an agreement about how timing information is stored (example: timestamp in std_msgs/Header)
- After the Fact ingestion
  - Usually you can use embeddings within the log file
  - Log file computer may be distant from the source, so you may need to account for this as well

sensor_msgs/Image

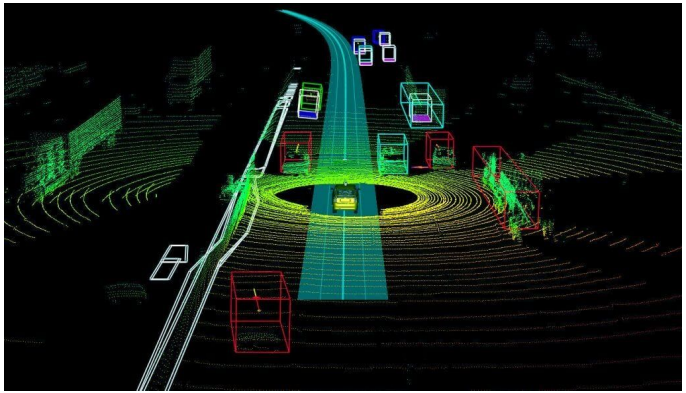
```
std_msgs/Header header
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

gps_common/GPSFix

```
std_msgs/Header header
gps_common/GPSStatus status
float64 latitude
float64 longitude
float64 altitude
float64 track
...
```

Transforms

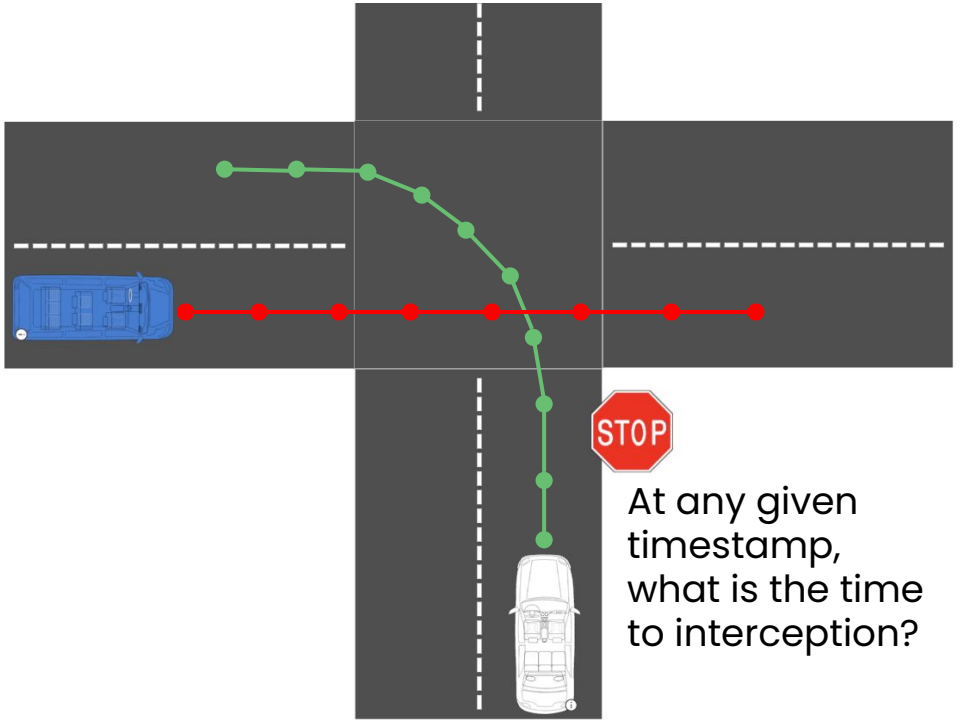
- Frequently your data comes “nearly” raw from the sensors
- Example: LiDAR
  - A spinning device that sends out laser beams and measures the time of flight for the returned signal.
  - This generates a “point cloud” of all of the returned signals
  - These are measured relative to where the LiDAR is mounted
- Transforms are *typically* defined as a rotation and translation to get from one coordinate frame to another
  - Do you translate first or rotate first? It matters.
  - Not everyone does it the same.
  - There are *conventions*, but not strictly followed
  - You must also track which frame is transformed from and to (some publish the inverse of what you need)
- Transform operations are not easily handled by things like DataFrame libraries or SQL
  - Minimum 7 entries to compute a transform: 3 for translation and 4 for rotation
  - The math is not trivial
  - Frequently you have multiple transforms to apply
    - Robotics tooling does this “built in”



"Your data cannot remain as it is. You must convert it—only then can you achieve greatness."

Spatial Analysis

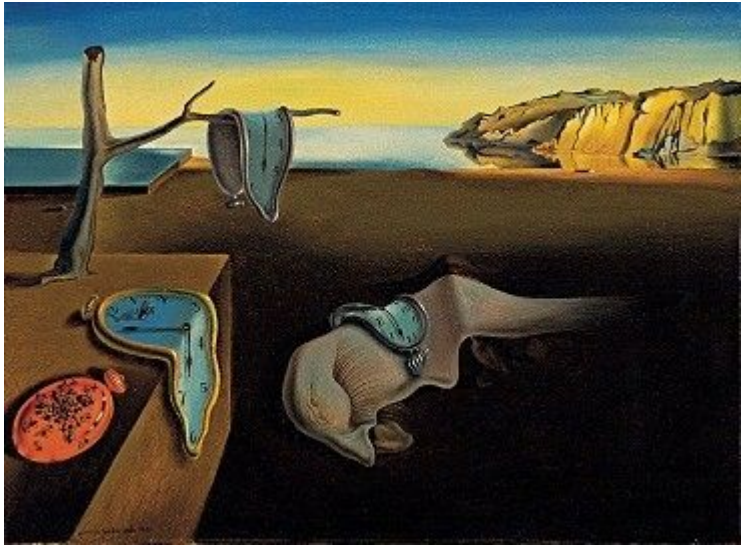
- Spatio-Temporal Analysis is one of the most common tools roboticists need to use
- Examples:
  - Path intersection (in picture)
  - Projection onto a curved route
  - Time to intersection
    - Made more fun by varying acceleration profiles
- Robotics engineers have *many* tools to handle these, but they don't always map well to database approaches
- Existing packages:
  - geopandas - by far the most used
  - Sedona - geo on PySpark, essentially
  - geoarrow-rs : newcomer and a member of the DataFusion community
- There is no native storage structure for geo data
  - Usually encoded as text (WKT) or binary (WKB)



At any given timestamp, what is the time to interception?

Time Synchronization

- On a distributed robotics system, there is a fundamental question of what time means
- Some common approaches:
  - The local time of the computer for which the data were generated
  - The time embedded within a message that triggered a process to run
  - The time of receipt of a message according to a specified computer (such as the logger)
  - The time specified by a time server at which a message was published
- This is one of the most important features with the data from a robotics system. How we organize data along temporal dimension will impact nearly all analysis.
  - **You must get this right.**



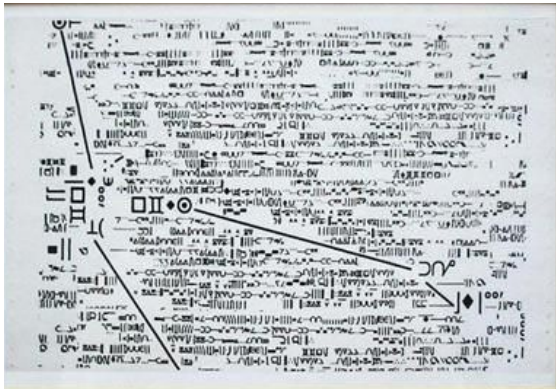
The Persistence of Memory by Salvador Dali

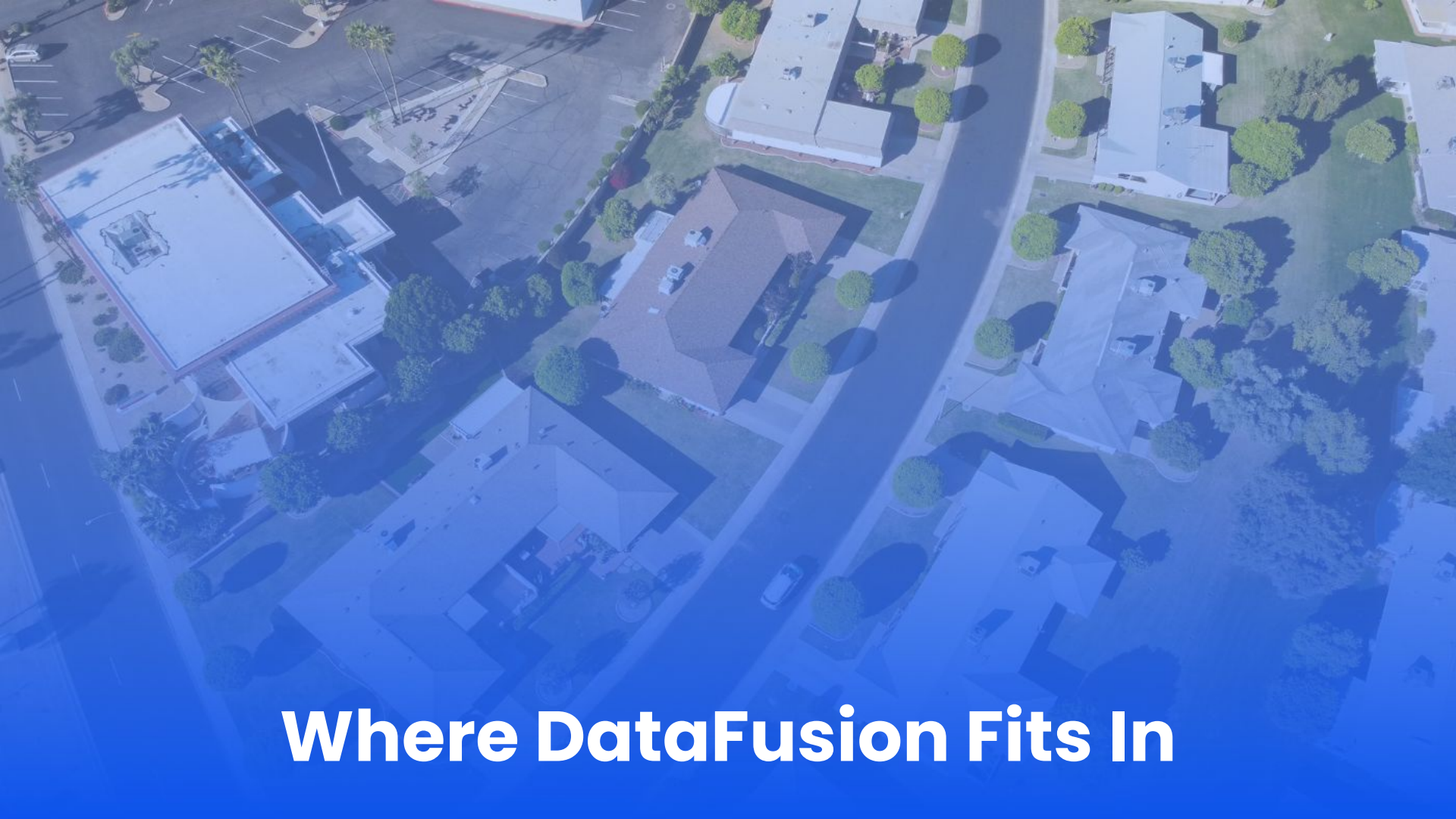
Noisy Data

Tim's opinionated view:

To be successful in robotics, you must cultivate an attitude of distrust of the data.

- Thresholding does not work due to boundary effects
  - Smoothing helps
  - Dynamic boundaries is a more robust solution
- Build in detections for known problems such as loss of data and duplicate data
  - Duplicate may not be exactly duplicate, so frequently need more nuance, such as some fields to ignore
- Build in outlier detections into all data analysis workflows
  - Don't tack it on at the end, make it part of your flow
  - Rust got a huge win from treating errors as part of the normal flow. Can we do the same in data?
- How to handle errors in the data is a top line problem, so make it front and center of your solution
  - When reviewing results, explain the reasoning for acceptance/rejection criteria





Where DataFusion Fits In

DataFusion for Robotics Data Science

- Python is the language for Data Science, almost exclusively with DataFrame APIs
  - Overwhelmingly: PySpark for scale, Pandas for single node
- For DataFusion to compete it must have comparable functionality and some clear advantage

Tim's Opinionated View:

Rust backed python UDFs is the killer feature for doing robotics data science in DataFusion.

Reasoning:

- Roboticists and Data Scientists have different skill sets
- Rust UDFs can be written in a way that Roboticists understand (ideally by Roboticists)
- Package up a set of reusable message oriented rust backed UDFs as an analytics toolkit
- Data Scientists can leverage this toolkit to get:
  - Highly performant functions that operate at native speed
  - Abstract away the portion of the problem that does not match their skill set

But to win, we need distribution. Under heavy development:

- ballista
- datafusion-ray

Abstraction will set you free.

How DataFusion might address...

- Getting the data into a data store
  - I don't think it impacts here, **but** my work on DataFusion led me to design an ingestion system that is saving so much money it pays my entire salary
- Transforms
  - Easy win here - These are trivially implemented in Rust
  - Good crates already exist - it's simply a matter of data agreements about struct/column layout
- Spatial Analysis
  - Geo-arrow is a top contender to win here
  - Ideally we can build in extension types, but this could break workflows
    - Suppose you want to read from a table which has these extension types but you aren't using those columns. Do you have to still load in the extensions to your workflow?
- Time Synchronization
  - Potentially a win by using a user defined async join?
  - We currently have a process to merge, sort on a time column, fill forward empty data
    - OOM problems on large data sets, so still needs work
  - Still needs more thought
- Noisy Data
  - User defined window functions!

Questions?