IOT AND THE AUTONOMOUS VEHICLE IN THE CLOUDS: SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) WITH KAFKA AND SPARK STREAMING

J. White Bear

IBM, Spark STC

About Me

Education

- University of Michigan- Computer Science
  - Databases, Machine Learning/Computational Biology, Cryptography
- University of California San Francisco, University of California Berkeley-
  - Multi-objective Optimization/Computational Biology/Bioinformatics
- McGill University
  - Machine Learning/ Multi-objective Optimization for Path Planning/ Cryptography

Industry

- IBM
- Amazon
- TeraGrid
- Pfizer
- Research at UC Berkeley, Purdue University, and every university I ever attended. ☺

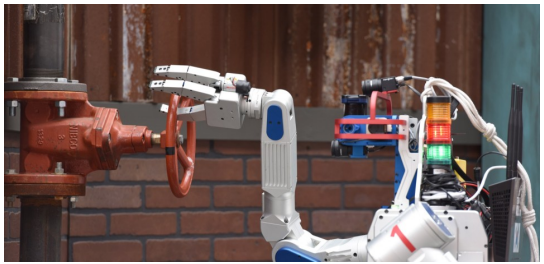
Fun Facts (?)

I love research for its own sake. I like robots, helping to cure diseases, advocating for social change and reform, and breaking encryptions. Also, all activities involving the Ocean and I usually hate taking pictures. ☺

Introduction: Robotics Today

FIRST Robotics World Championship
NASA Glenn Research Center in Cleveland sponsored Tri-C's team.

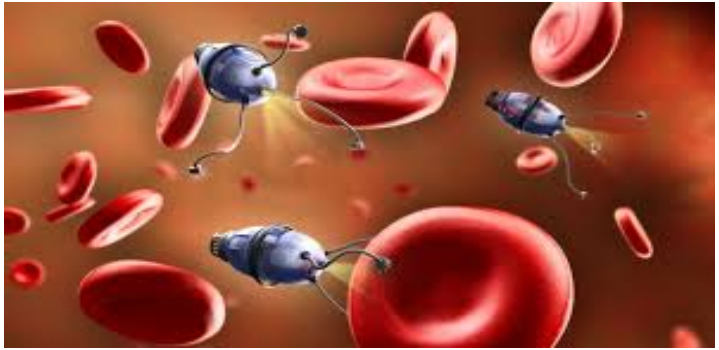
Amazon Drones



South Korean Team, KAIST wins the DARPA Robot Challenge

Tartan Racing's Boss, the robotic SUV that won the 2007 DARPA Urban Challenge,

Introduction: Robotics Tomorrow



Nanorobots wade through blood to deliver drugs

SLaM and ML on automated wheelchair

Navigate stores, museums and other indoor locations, with directions overlaid onto your surroundings. Google Tango

Space/underground/underwater rescue and exploration. Places humans can't go.

What is SLAM?

Simultaneous Localization and Mapping (SLAM)

- Formal Definition
  - Given a series of sensor observations over discrete time steps the SLAM problem is to compute an estimate of the agent's location and a map of the environment. All quantities are usually probabilistic, so the objective is to compute:

$$P(x_t|o_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|m_t, o_{1:t-1})/Z$$

  - Computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

  - SLAM algorithms use various implementations to attempt to find heuristics to make this problem tractable using machine learning and probabilistic models

  - GPS cannot account for unknown barriers, precision navigation, moving objects, or any areas with satellite interference including weather phenomena.

What is SLAM?

What are some of the key challenges in SLAM?

- **Computer vision** correctly and identifying images observed
- **Moving objects** Non-static environments, such as those containing other vehicles or pedestrians, continue to present research challenges. (collision detection)
- **Data Association**-refers to the problem of ascertaining which parts of one image correspond to which parts of another image, where differences are due to movement of the camera, the elapse of time, and/or movement of objects in the photos.
- **Loop closure** is the problem of recognizing a previously visited location and updating the states accordingly.

What is SLAM?

Why SLAM on IoT?

SLAM in IoT

- "[SLAM] is one of the fundamental challenges of robotics . . . [but it] seems that almost all the current approaches can not perform consistent maps for large areas, mainly due to **the increase of the computational cost and due to the uncertainties that become prohibitive when the scenario becomes larger."**[12] Generally, complete 3D SLAM solutions are highly computationally intensive as they use complex real-time particle filters, sub-mapping strategies or hierarchical combination of metric topological representations, etc. (Wiki)

- Computational costs become prohibitive on embedded systems, especially smaller robotic modules. The data becomes large and the calculations and corrections over time and space become much more important. **Specifically, SlaM increases exponentially with the number of landmarks found.**

- The state uncertainty increase with time and space, and must be bounded by some form of machine learning to predict and use accurate corrections in the algorithm

- Additional sensors, rapid movements, processing visual input adds additional computational burdens…

Why SLAM on IoT?

The Benefits

- Seamless integration and scaling allowing users to easily improve the heuristics of the algorithm without losing any of the performance expectations of an embedded system.
  - Including smart cities, lawn mowing, dog walking, kitchen appliances, or even communication inside the human body creating a truly unique interaction between humans and robotics
- Large scale evaluation of performance metrics for all IoT systems (Big Data)
- Monitoring and control of sensors based on stored data (eg reducing sensor usage to conserve power)

Why SLAM on IoT?

Current Approaches

- Robot Operating System (ROS) a collection of software frameworks for robot software development
  - Providing operating system-like functionality on a heterogeneous computer cluster.
  - Hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.

- No true real-time analytics! Despite the importance of reactivity and low latency in robot control, ROS is not a Realtime OS

- Difficult to scale in IoT! Adding a heterogenous swarm, or integrating interactions requires significant planning.

- There is a need! Are there any plans to build Kalman filtering and system identification into this framework? https://github.com/sryza/spark-timeseries/issues/19

- **We need a framework that can do this! Enter Apache Kafka and Spark Streaming!**

Why SLAM on IoT?

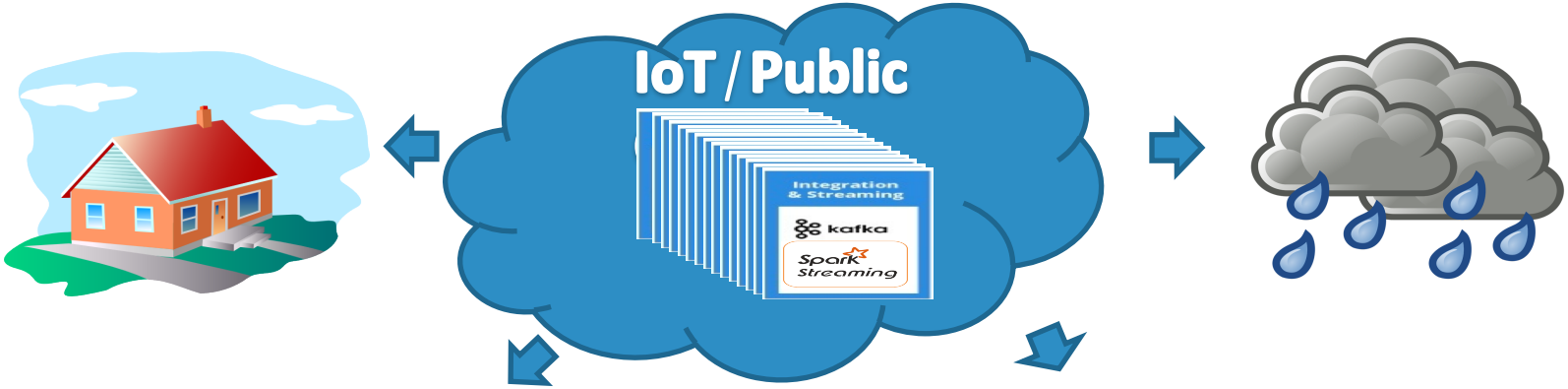
Why build an ATV in isolation?

"Google is trying to teach its cars to think more like humans. Google's explanation of the incident …."

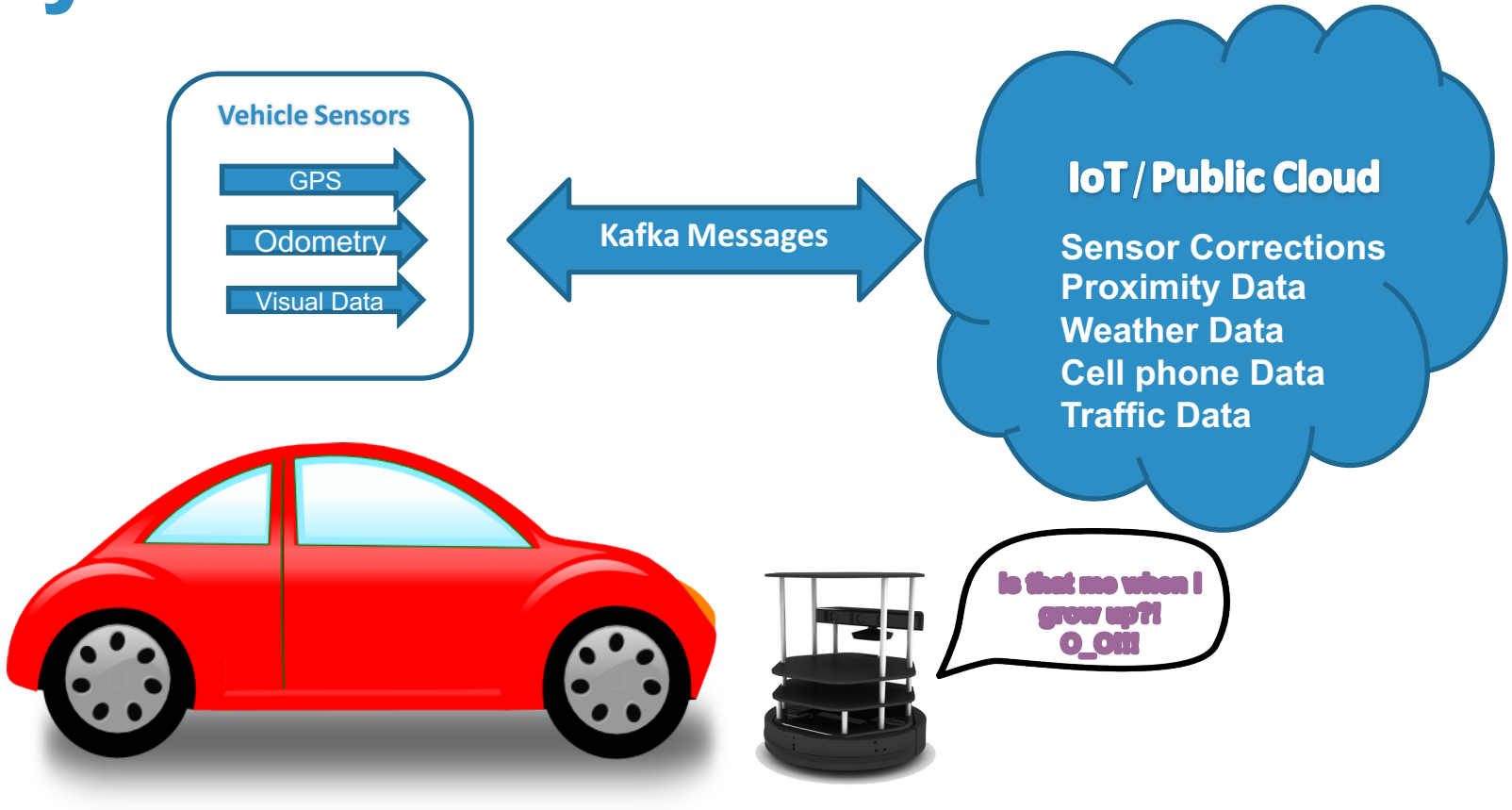
- **Google cars have been involved in multiple accidents. Like many drivers, Googles blames the other non ATV drivers on the roads, humans.**
  - **http://gizmodo.com/a-google-self-driving-car-got-into-a-crash-with-a-bus-1762007421**
  - **http://www.reuters.com/article/us-google-autos-accidents-idUSKBN0NX04I20150512**

- **"The autopilot sensors on the Model S failed to distinguish a white tractor-trailer crossing the highway against a bright sky."**
  - **Tesla cars, also, recently ran into a bus. Unfortunatetly resulting in a death.**
    - **https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk**

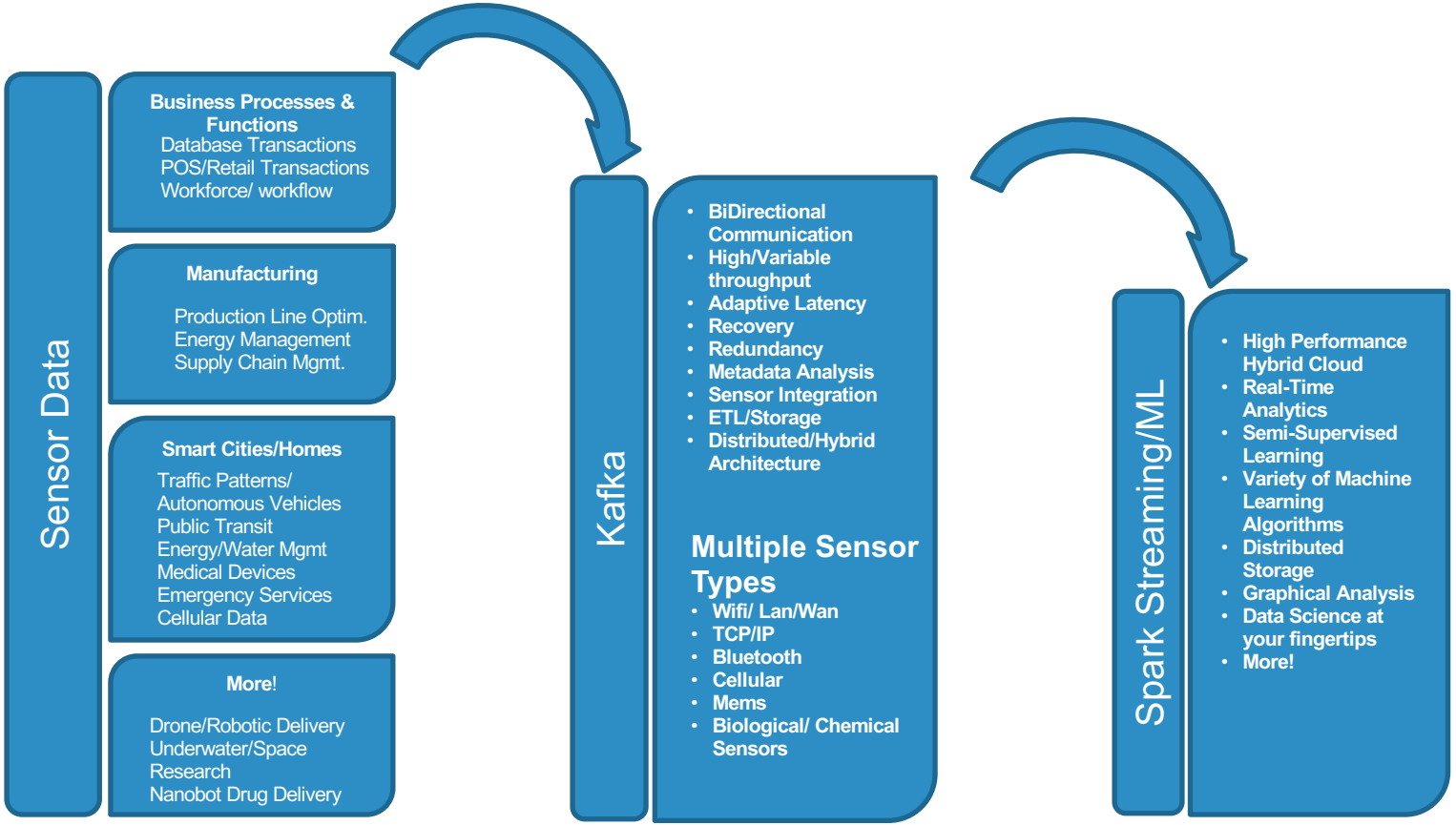
Why SLAM on IoT?



Why SLAM on IoT?

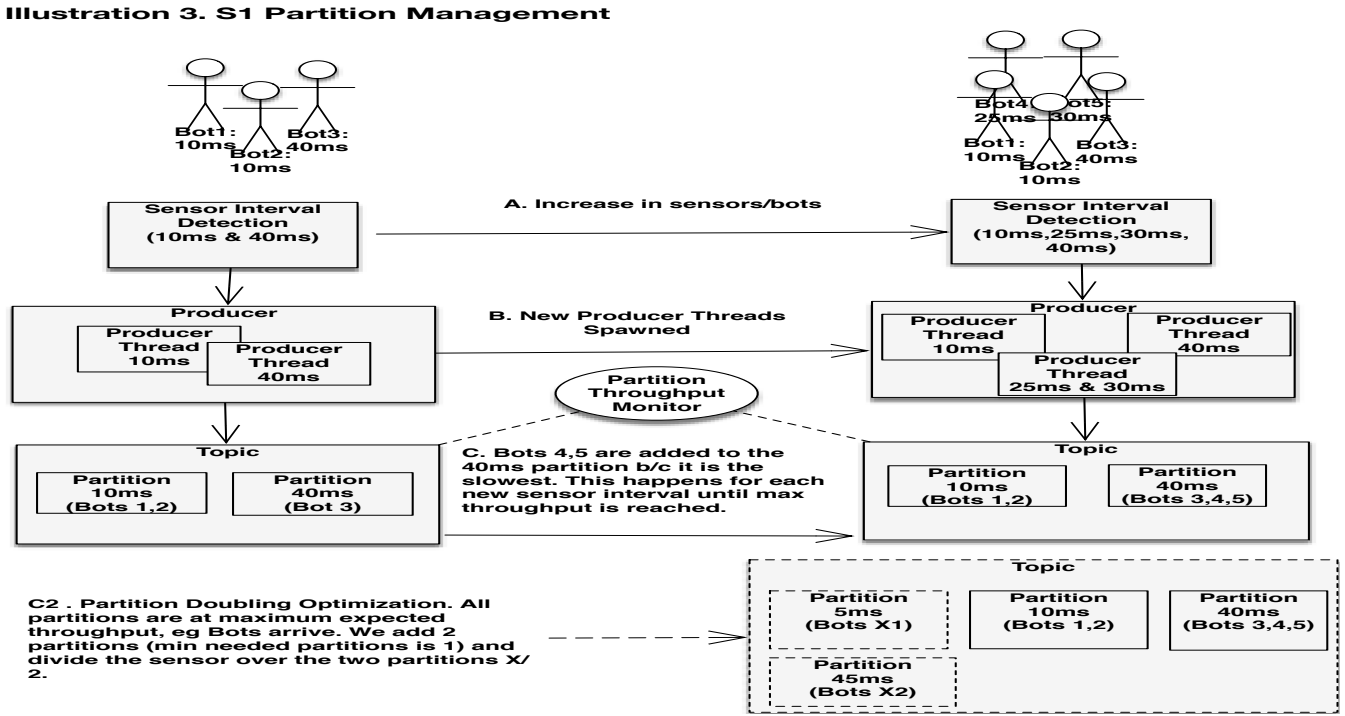


The Framework.



More sensors, more problems.

The Framework: The Kafka's in the Details.



The Framework: The Kafka's in the Details.

This is the first of many machine learning entry points in our cloud. Let's assume a sensor is detected…..

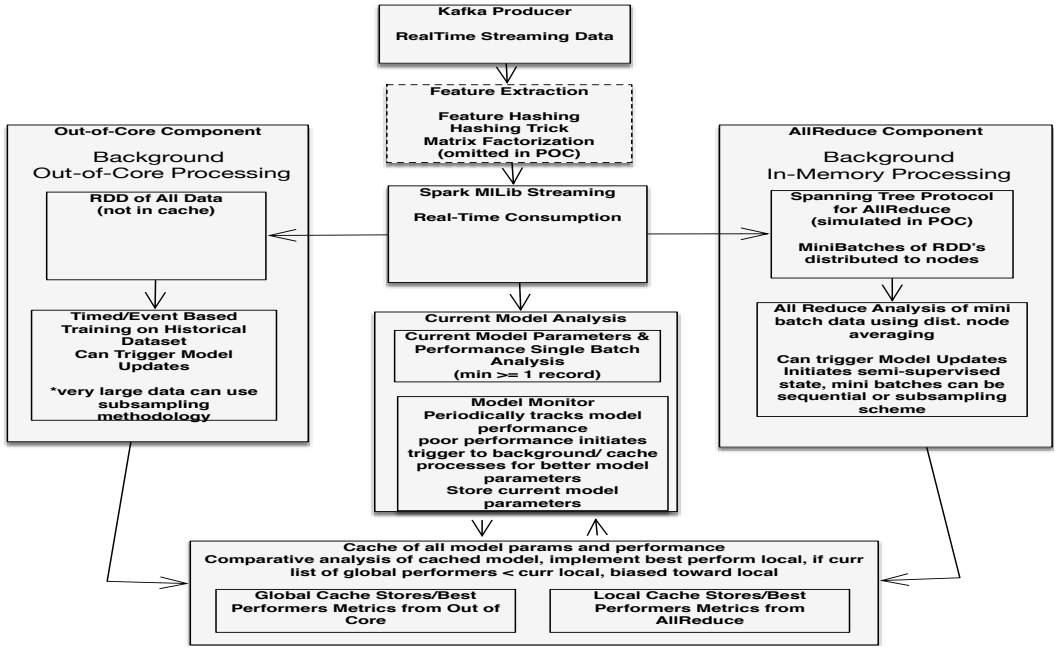
- **How do we determine what kind of sensor is it?**
- **What is the expected interval of communication of this sensor?**
- **What is an interval that indicates failure?**
- **Which vehicle is it connected to?**
- **What other sensors should we be expecting to integrate with it?**
- **What are the key intersections in this data, with our public data?**

The Framework: The Kafka's in the Details.

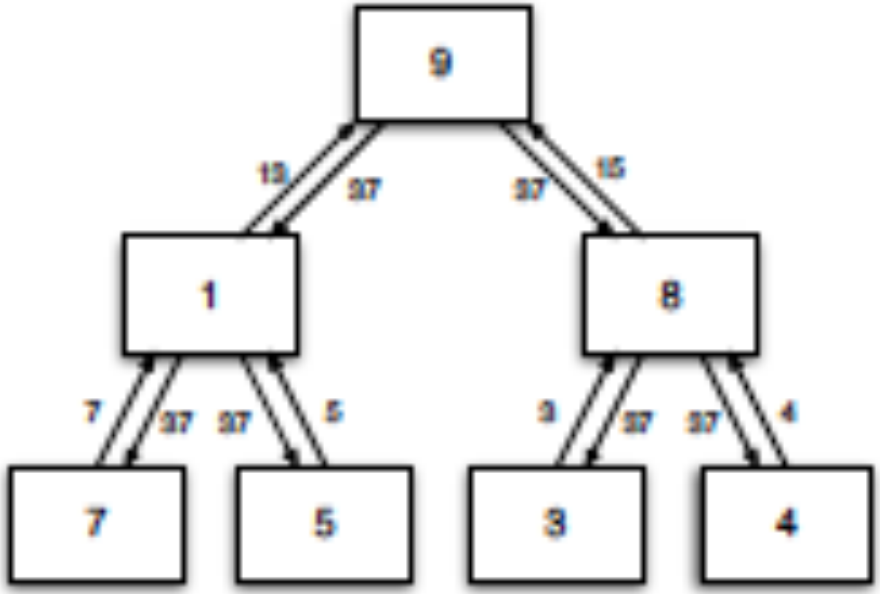
We need to model this as a machine learning problem and, in order to be efficient we need this to happen in real-time or near real-time.

- **Every sensor needs to be detected and distributed correctly in our messaging architecture.**
- **The attributes we looked at early can be modeled as features and hashed accordingly.**
- **The distribution needs to be correct and responsive.**
- **We need automated retraining on these algorithms to ensure our models don't diverge and remain within a certain boundary.**

The Framework: Enter Real-Time Learning…



The Framework: Enter Real-Time Learning…



A Reliable Effective Terascale Linear Learning System, Agarwal et al.

The Framework: Implementation

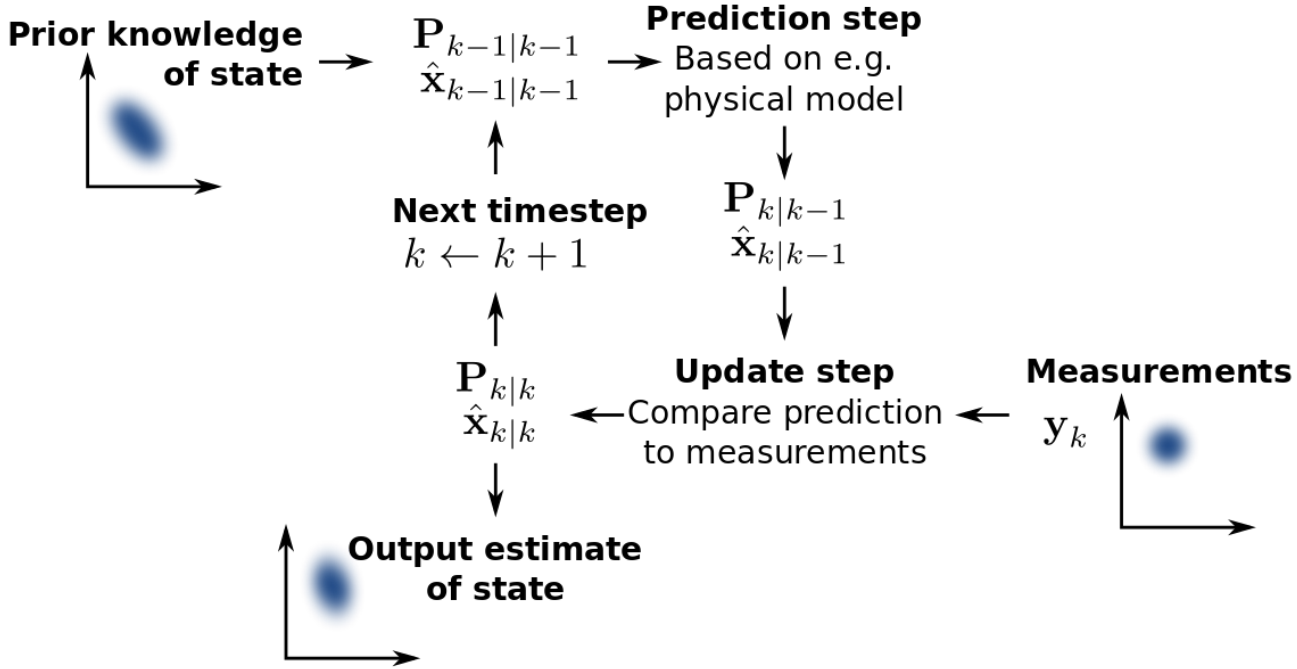
The Approach

- Extended Kalman Filter (matrix based update/estimation)
  - Nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. de facto standard in the theory of nonlinear state estimation eg navigation systems and GPS. (wiki)

- TurtleBot II (standard robotics research bot) (named PQ)

The Framework: Implementation

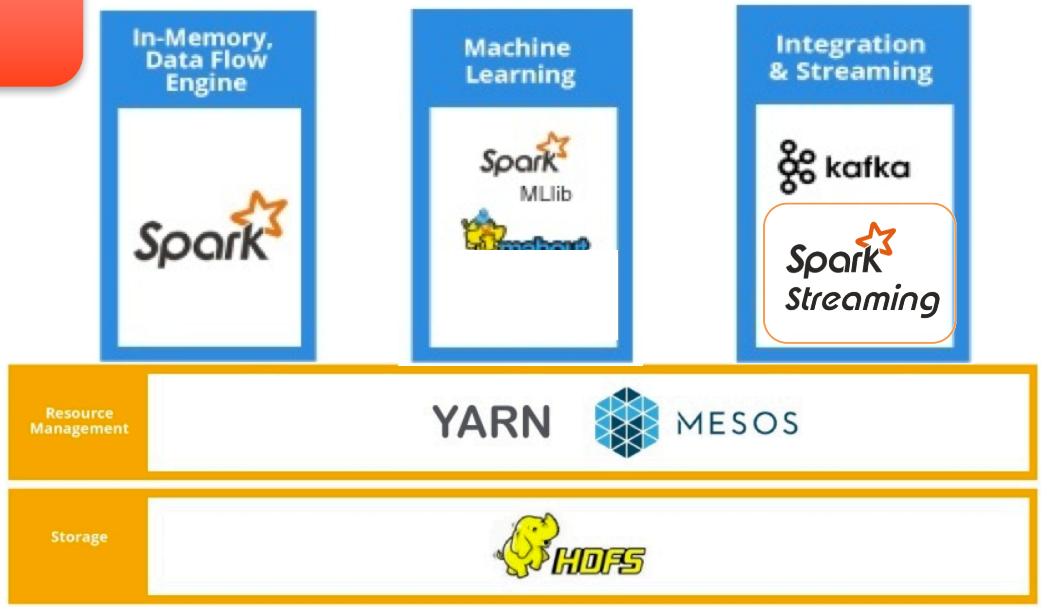
The Approach



The Framework

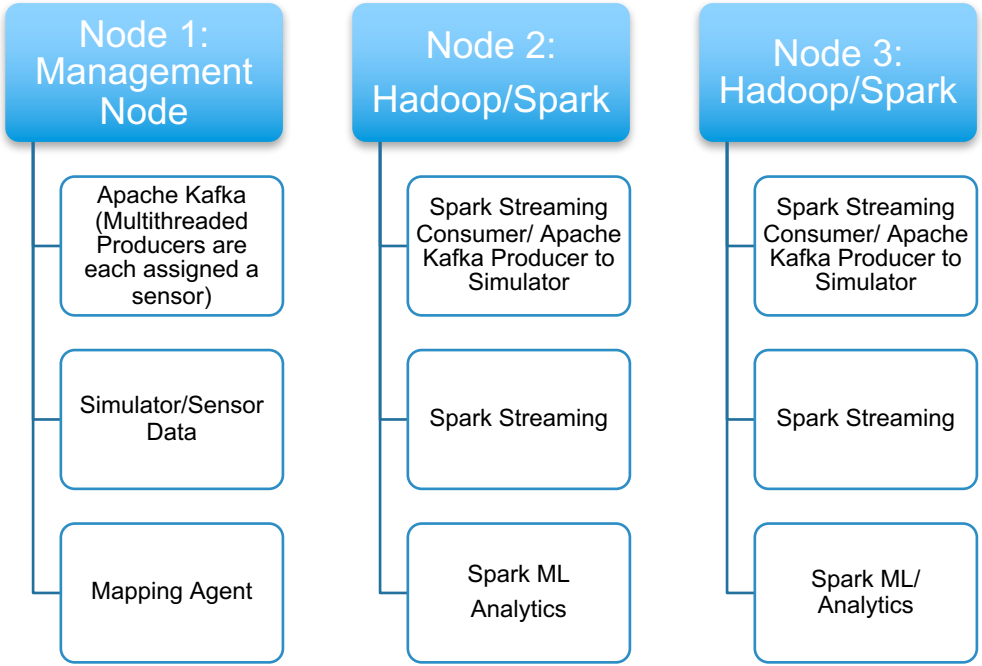
The Framework

Our cluster: IBM SoftLayer cluster with 3 Nodes.



The Framework

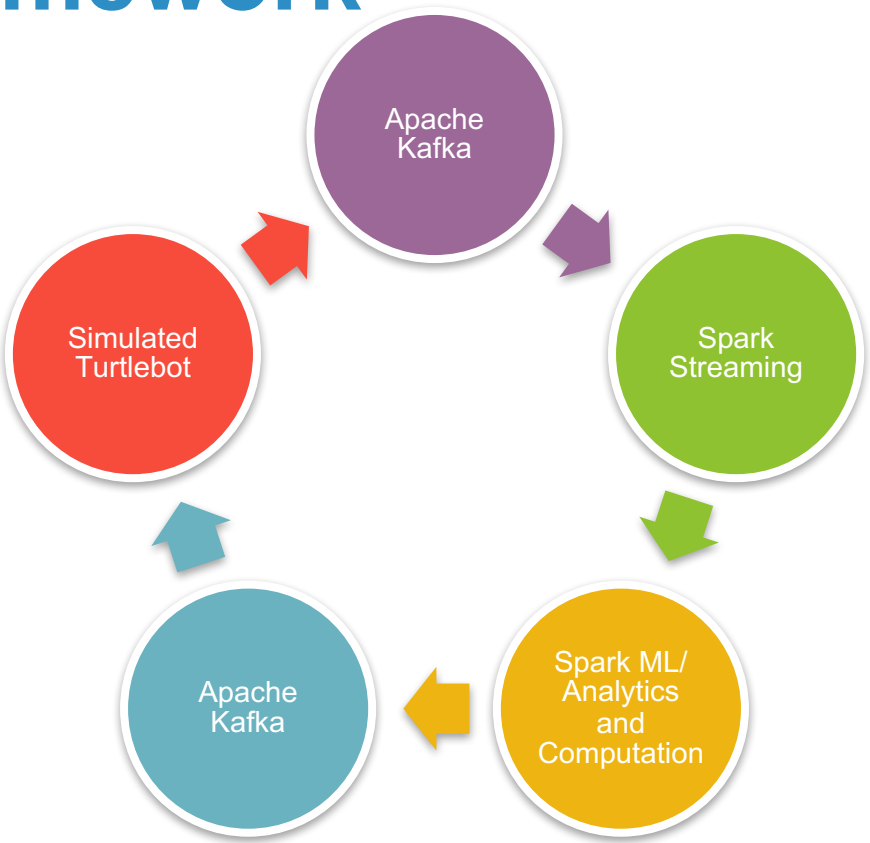
IBM SoftLayer cluster with 3 Nodes.



The Framework

A high performing plug n play cloud for smart robotics, drones and intelligent systems that allows easily tuneable interactions for scientists and industry in any environment!

- Odometry, pose and orientation data for every movement.
- Laser scan data every 30ms with over 1200 data points per read!
- One robot and not even all the sensors!



The Framework

A high performing plug n play cloud for smart robotics, drones and intelligent systems that allows easily tuneable interactions for scientists and industry in any environment!

- ••EKF is calculated primarily using matrix operations!
- ••Distributed raw sensor data using Apache Kafka. Number of sensors limited only by Kafka cluster!
- ••Improved performance using RDDs and Spark ML for computational intensive tasks!
- ••Fast/optimized learning and analytics!
- ••Real-time sensor messaging!
- ••Easy sensor integration and scaling!
- ••Retention of data over time for improved optimizations and accuracy!

The Framework: Apache Kafka


```
1  package com.kafka.producer;
2
3  import org.apache.kafka.clients.producer.Producer;
4  import org.apache.kafka.clients.producer.ProducerRecord;
5
6  public class ProducerSim {
7
8      public ProducerSim(ProducerSimConnect sim){
9          try {
10
11                  //publish odometry data
12                  Thread threadOdom = new Thread("threadOdomSim") {
13
14                        public void run(){
15                            //instance must be created inside run class
16                            ProducerClass prod = new ProducerClass();
17                            Producer<String,String> prodr = prod.getProducer()
18
19                           //get odom file
20                            try {
21                              while(true){
22                                   double[] odom = sim.getOdom();
23                                   String odomLine =
24                                           odom[1] +","+ odom[2] + ","+ odom[3]
25                                   prodr.send(new ProducerRecord<String, String
26                                       ("odom", String.valueOf(odom[0]),odomLin
27                                    prodr.flush();
28                                    System.out.println("odom " + odomLine);
29                                    //laser scanner intervals
30                                   Thread.sleep(1000);
31                              }
32
33                          } catch (Exception e) {
34                              // TODO Auto-generated catch block
35                              e.printStackTrace();
36                          }
37
38                      }
39                  };
40
41                  //publish odometry data
42                  Thread threadLaser = new Thread("threadLaserSim") {
43                        public void run(){
44                            //instance must be new in each thread to maintain
45                            ProducerClass prod = new ProducerClass();
46                            Producer<String,String> prodr = prod.getProducer()
```

The Framework: Spark Streaming

Spark Streaming Integration	
Apache Spark Streaming	**Apache Kafka Consumer**
Replaces Kafka Consumer	Producer feeds directly to Spark Streaming
Adheres to fault tolerance policies incl. WAL (write ahead logs to HDFS)	Not necessarily thread safe (Java Api)
KafkaUtils.createDirectStreamDirect w/o Receivers in new version, better access to low level Kafka metadata	Auto-commit feature, partition replication, integration with Zookeeper. Finely tuned metadata access and storage by topic and partition
Microbatch processing and better integration into Spark incl online learning	Buffered batches, developing streaming analytics capabilities

The Framework: Spark Streaming


```
1  public class Stream {
2      private Set<String> topicSet;
3      private Map<String, String> kafkaParams;
4      private ArrayList<double[]> odometry;
5      private ArrayList<ArrayList<double[]>> laserData;
6      private Data d;
7
8      public Stream(JavaSparkContext sc){
9          //storage
10         odometry = new ArrayList<double[]>();
11         laserData = new ArrayList<ArrayList<double[]>>();
12         d = new Data();
13
14         //set batch size
15         //streaming context
16         JavaStreamingContext jssc =
17                 new JavaStreamingContext(sc, Durations.seconds(1));
18
19         //topic list
20         topicSet = new HashSet<String>(Arrays.asList("odom", "laser"));
21         //parameter list
22         kafkaParams = new HashMap<String, String>();
23         kafkaParams.put("bootstrap.servers", "localhost:9092");
24         kafkaParams.put("group.id", "ekf");
25         //set at movement of robot
26         kafkaParams.put("auto.commit.interval.ms", "1");
27         kafkaParams.put("consumer.timeout.ms", "10");
28
29
30         //connect to tcp
31         /*JavaReceiverInputDStream<String> lines =
32                 jssc.socketTextStream("localhost", 10555);
33
34         lines.print();*/
35
36         JavaPairInputDStream<String, String> msg =
37                 KafkaUtils.createDirectStream(
38                             jssc,
39                             String.class,
40                             String.class,
41                             StringDecoder.class,
42                             StringDecoder.class,
43                             kafkaParams,
44                             topicSet
45                         );
46
```

The Framework: Spark ML, RANSAC

Spark ML with RANSAC

••RANSAC

•• One of many iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers.

•• Default methodology for determining whether a series of landmark forms a wall or structure

••Ideal for consumption with high-throughput batches in Spark Streaming!

••Integrated as an online learning algorithm (This framework) as back-end iterative process in Spark Streaming/ Spark!

The Framework: RANSAC


```
1       /* Tuning Parameters
2        * N — Max number of times to attempt to find lines.
3            S — Number of samples to compute initial line.
4            D — Degrees from initial reading to sample from.
5            X — Max distance a reading may be from line to get associated to lin
6            C — Number of points that must lie on a line for it to be taken as a
7        */
8
9       //RANSAC Parameters
10      final static int MAX_ITERATIONS = 10;
11      final static int SAMPLES = 5;
12      final static int DEGREE_RANGE = 10;
13      final static int MAX_INLIER_DISTANCE = 3; //in cm/inches
14      final static int MIN_LINE_SIZE = 10; //in number of points
15
16      //ml vars
17      SQLContext sqlContext;
18      DataFrame training;
19
20      //SLAM vars
21      final static int LIFE = 40; //time to discard landmark ???
22      final static double MAXRANGE = 1;
23      final static double DEGREESPERSCAN = 0.5; //??
24
25      //landmark
26      ArrayList<Landmark2> landmarkDB; //track landmarks, change to RDD
27
28      public Landmark2(){
29          //landmarkDB = new ArrayList<Landmark2>();
30
31          sqlContext = new org.apache.spark.sql.SQLContext(sc);
32
33          // Load a text file and convert each line to a JavaBean.
34          JavaRDD<String> readings =
35                  sc.textFile("sample_laser_range_cartesion_data.txt");
36
37          // The schema is encoded in a string
38          String schemaString = "x y";
39
40          // Generate the schema based on the string of schema
41          List<StructField> fields = new ArrayList<StructField>();
42          for (String fieldName: schemaString.split(" ")) {
43            fields.add(DataTypes.createStructField(fieldName, DataTypes.String
44          }
45          fields.add(new StructField("features", new VectorUDT(), false, Metad
46          fields.add(new StructField("label", DataTypes.DoubleType, false, Met
```

The Results

Key Challenges

- Network Latency
  - Embedded vs Framework
- Matrix computations and updates to large matrices
  - Jacobian (derivatives), Inversion, Transpositon, Multiplication, Addition/ Subtraction, Gaussian
- Covariance/Estimation computations
- Coordinating movement with computation
- Spark ML to correctly interpret visual landmark data, minimizing errors

The Results

Challenges

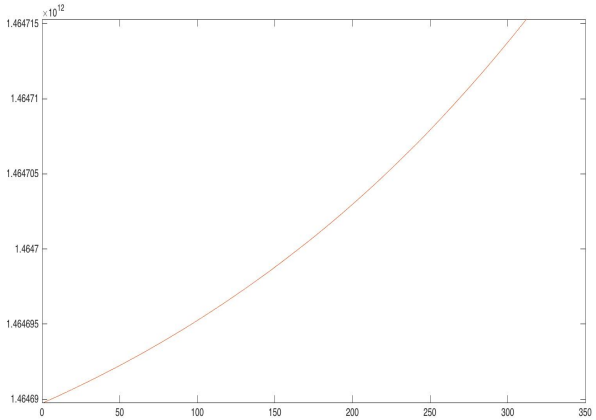
- ••~4KLOC (Java != verbose ☺)
- •• Java lambda documentation
- ••Kafka topics from Spark Streaming consumer
- ••Real-life latency depends on the type of connection and creates additional noise
  - ••Matrix computation
- ••Defining heuristics

The Results

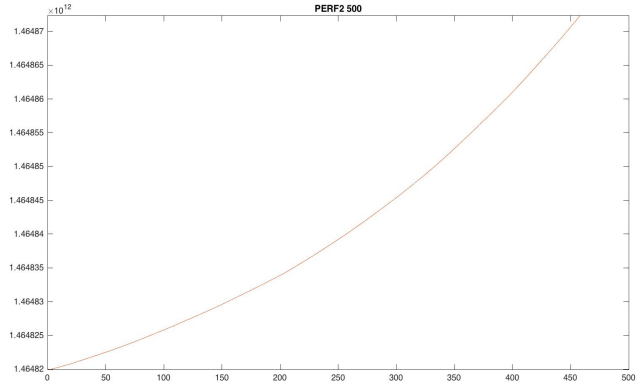
Measuring landmark acquisition and cpu time Embedded vs Framework at 500 iterations.

The Results

Measuring landmark acquisition and cpu time Embedded vs Framework at 500 iterations.



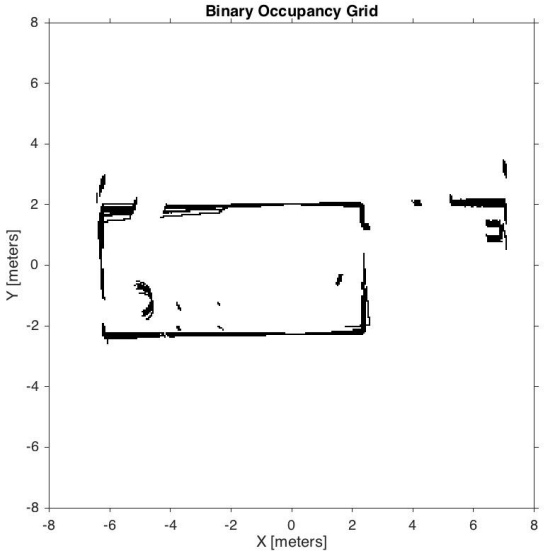
Embedded failed to complete at 500 iterations (up to ~300)



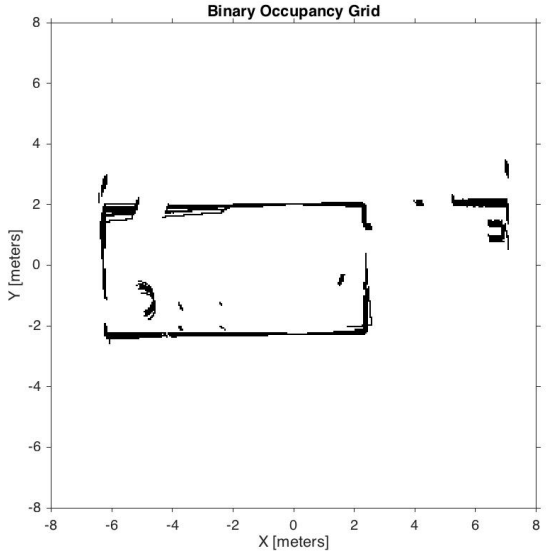
Framework completed 500 iterations with expected exponential growth

The Results

Measuring landmark acquisition and cpu time Embedded vs Framework for complete map. Both installations were run until the number of landmarks/maps were roughly equivalent and iterations marked.

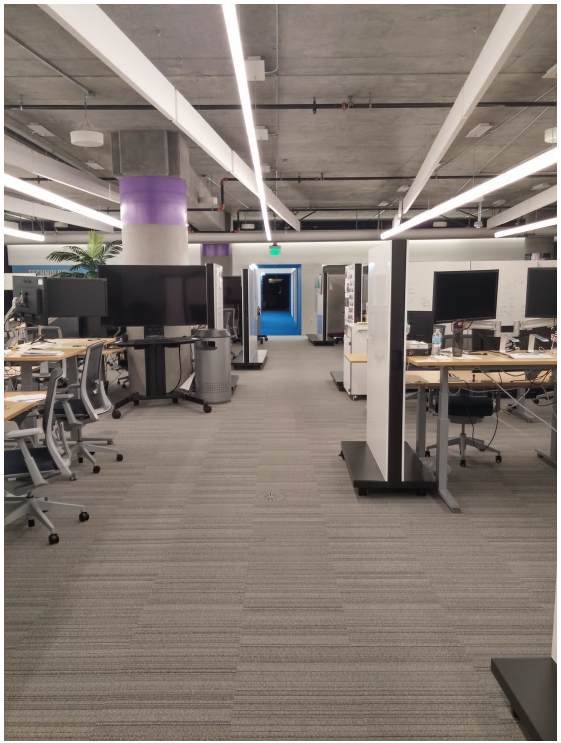


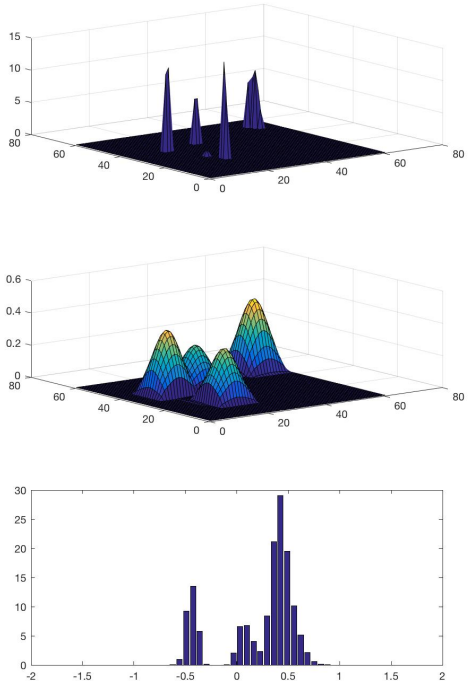
Iterations: ~100, Time ~2 min

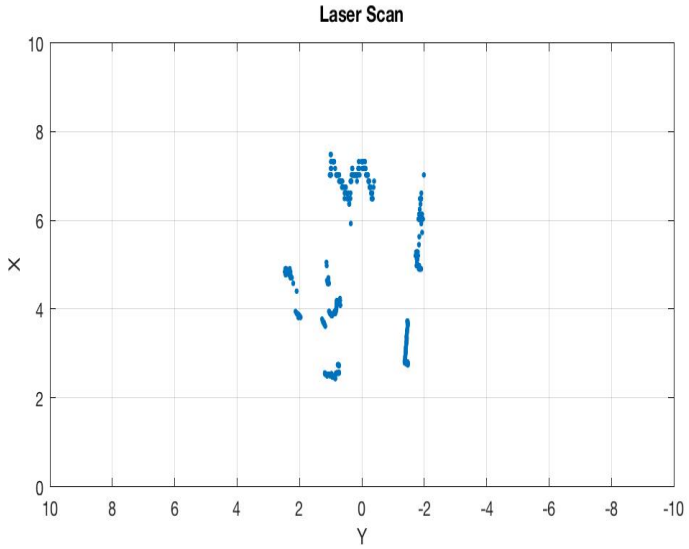


Iterations: ~100, Time ~30-40s

The Results



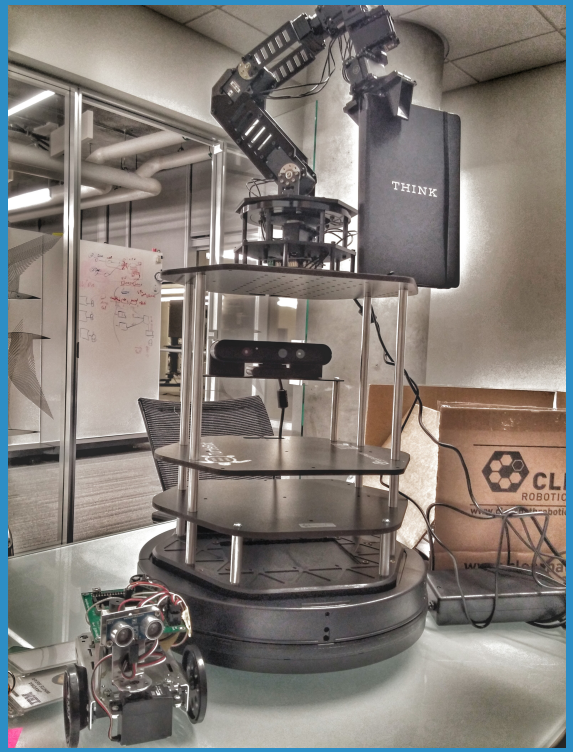

15
10
5
0
80
60
40
20
0
0
20
40
60
80
0.6
0.4
0.2
0
80
60
40
20
0
0
20
40
60
80
30
25
20
15
10
5
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Laser Scan
X
Y
10
8
6
4
2
0
10
8
6
4
2
0
-2
-4
-6
-8
-10

Next Steps

- **Automated Message Identification and Partition Management**
- **Expanded stochastic analysis beyond gradient descent**
  - Kalman Filter and Extended Kalman Filter
- **Improving accuracy and precision** with an end to end pipeline that allows customization/optimization
- **Path Planning** algorithms to improve search and search times
- **Incorporate swarms/particles**
- **A complete robotics library** or even extension to handle robotics, computer vision or any of the ai/machine learning problems specifics to robotics publishable and opens the door to a whole new group of scientists.
- **Further scaling and optimization** with robotic swarms and rapid/increased volume sensor data

Demo



Thank You,
Spark Summit!!

Contact Information:
J. White Bear (jwhiteb@us.ibm.com)
IBM Spark Technology Center
425 Market St San Francisco, CA

SPARK SUMMIT EAST 2017