Robotic Mapping & Localization

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 1

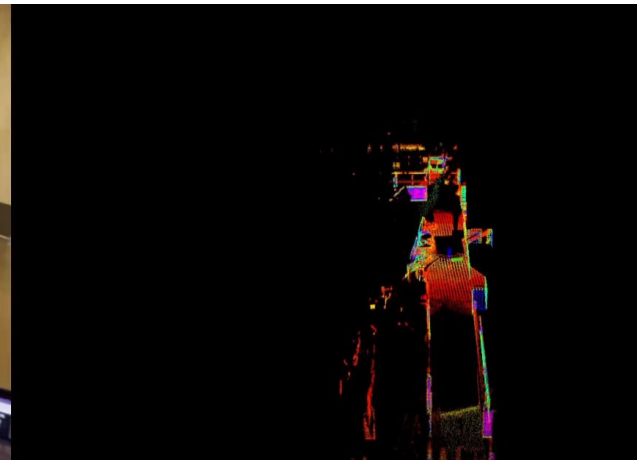
Instructor

Kaveh Fathian:

- Computer Science Assistant Professor
- Director of Autonomy, Robotics, & Intelligent Algorithms (ARIA) lab

ARIA Lab's Mission:

We pioneer algorithms for robotic perception & autonomy



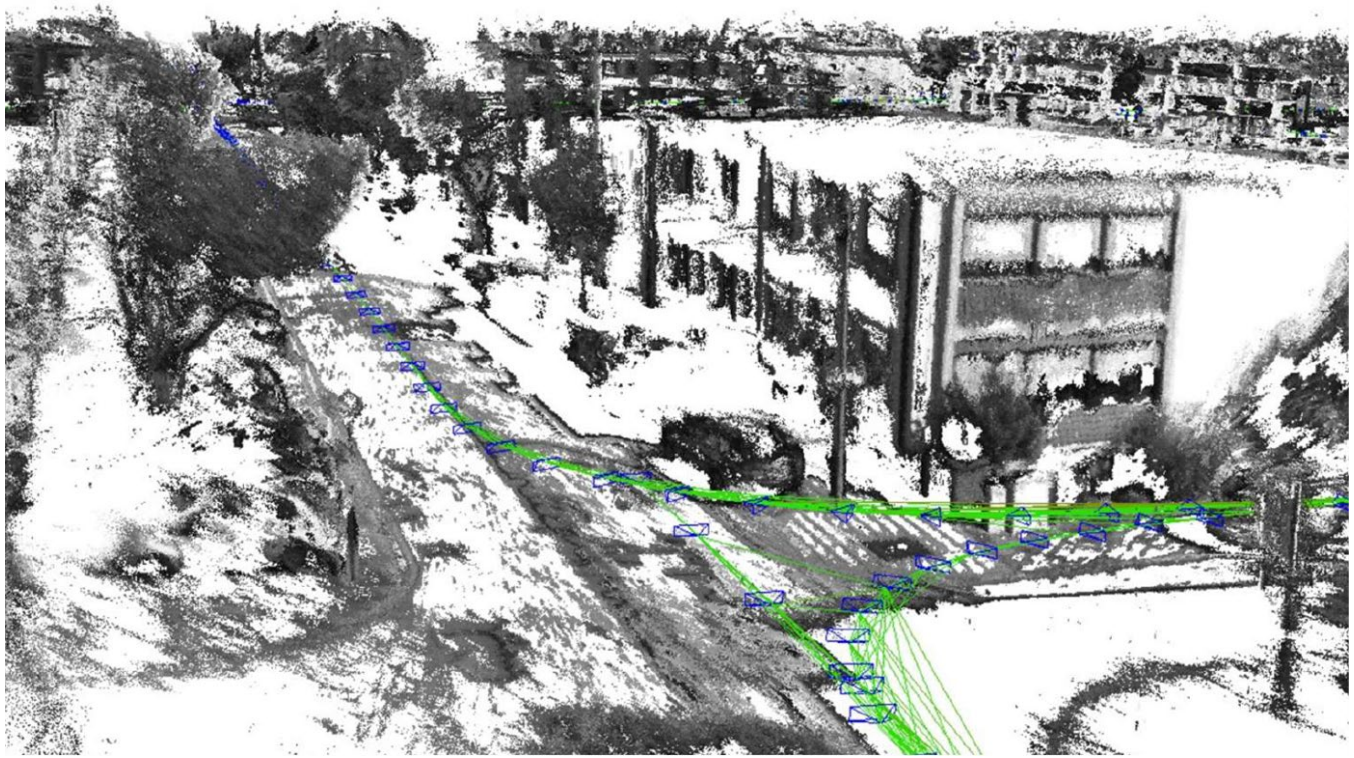
Want to join?

Visit https://www.ariarobotics.com/join-us

www.ariarobotics.com

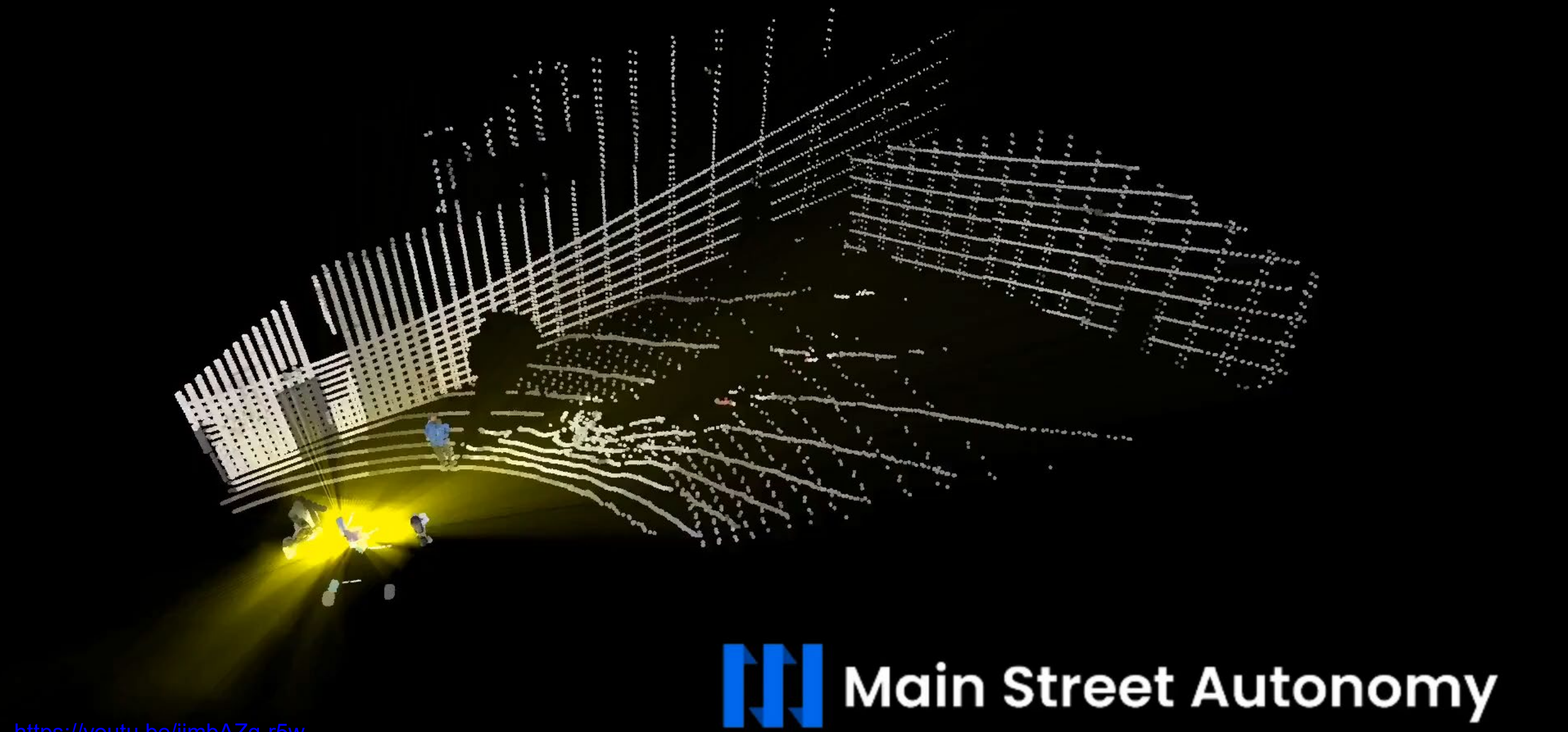
Lecture Outline

- **Robotic mapping & localization**
  - What is SLAM?
  - What is SLAM used for?
  - SLAM components
    - Sensors
    - Front-end
      - Odometry estimation
      - Loop closure
    - Back-end
      - Loop closure
      - Factor graph optimization



[Engel, IROS15]

Simultaneous Localization & Mapping



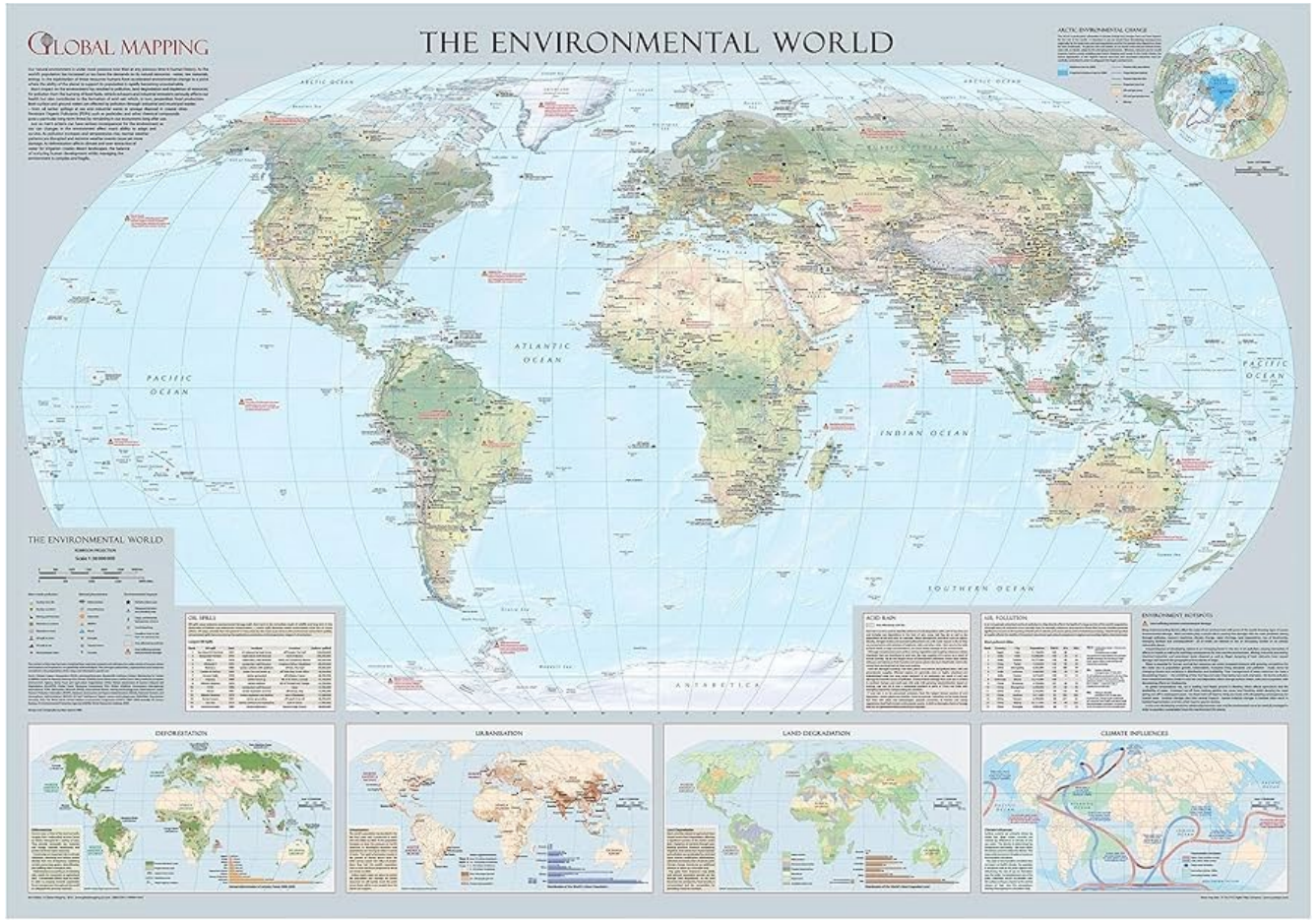
https://youtu.be/jjmbAZq-r5w

Simultaneous Localization & Mapping

https://youtu.be/8DISRmsO2YQ

Mapping

- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - Polygons/patches
  - Occupancy grid
  - Semantic/objects
  - Metric vs. topological



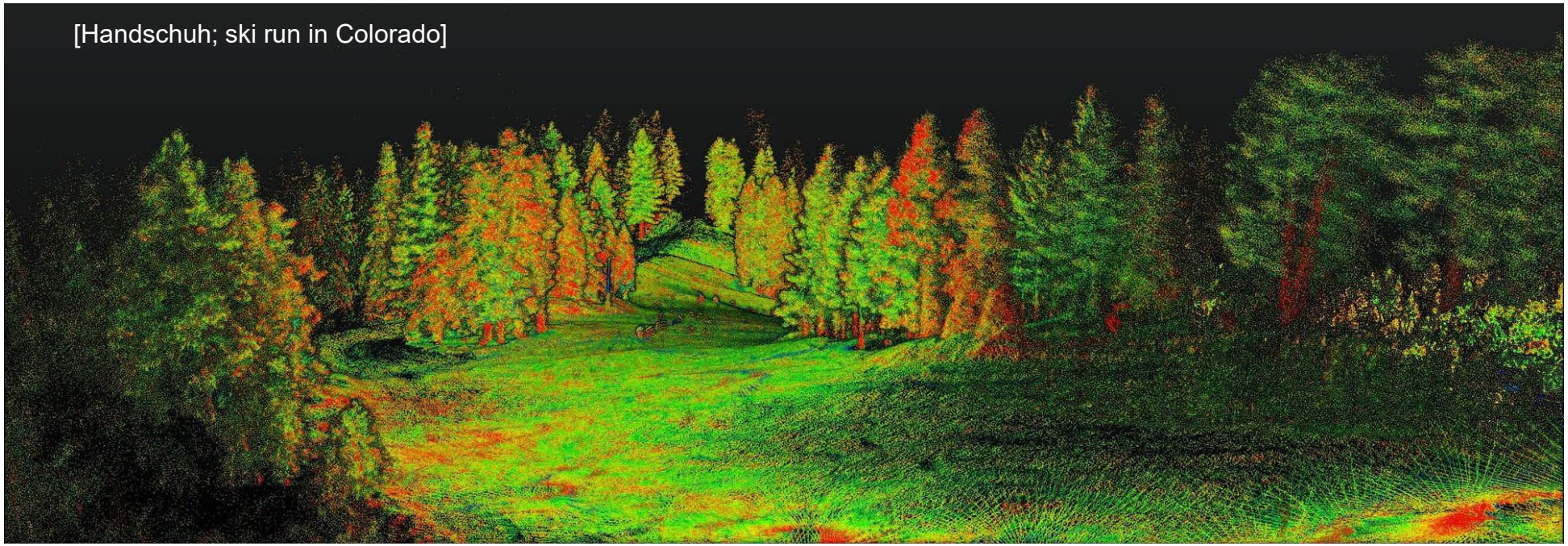
Mapping

- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - **Point clouds**
  - Polygons/patches
  - Occupancy grid
  - Semantic/objects
  - Metric vs. topological

[Lu, IEEE Access19]

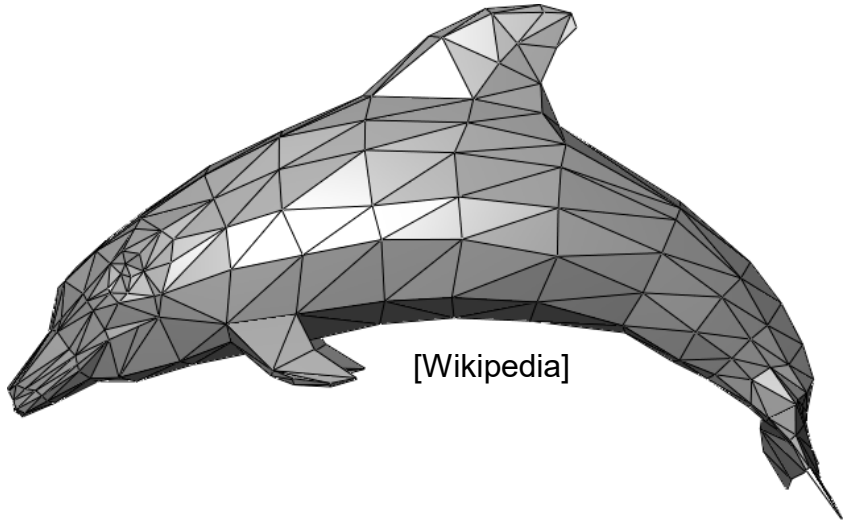
Sparse

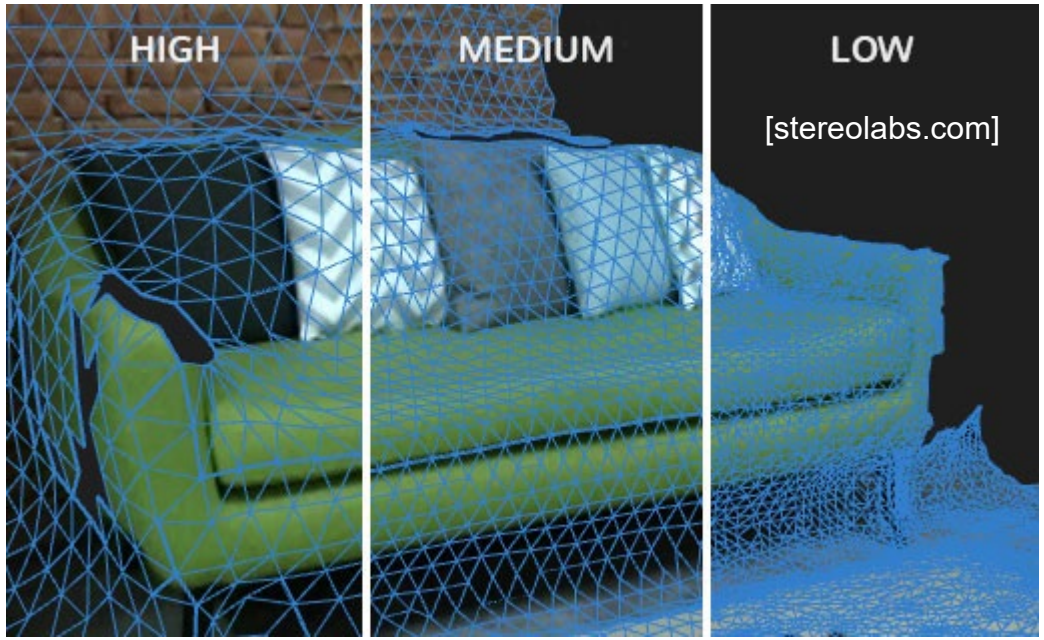
Dense



Mapping

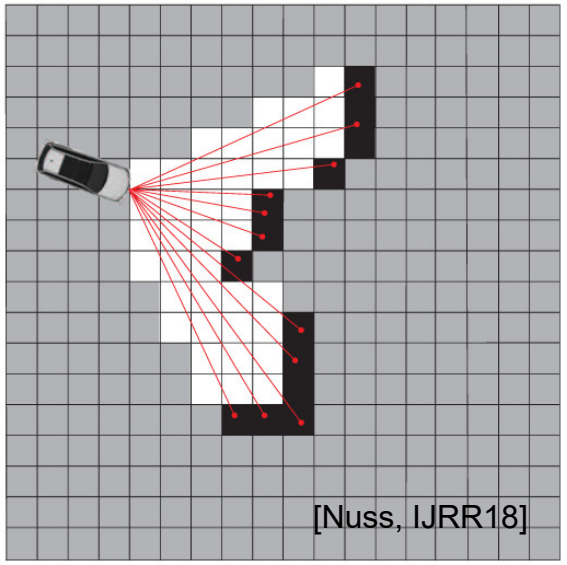
- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - **Polygons/patches**
  - Occupancy grid
  - Semantic/objects
  - Metric vs. topological



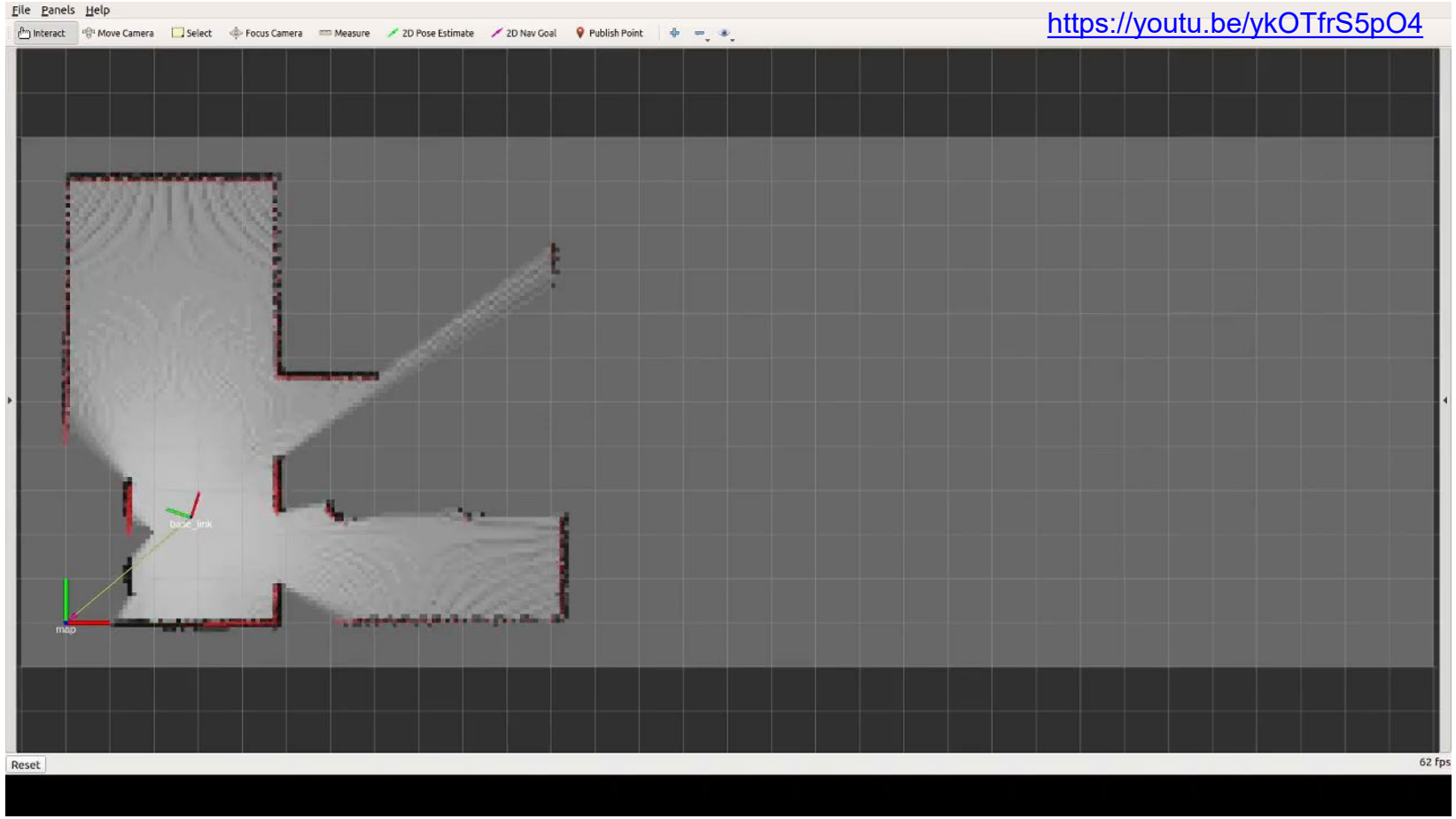


Mapping

- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - Polygons/patches
  - **Occupancy grid**
  - Semantic/objects
  - Metric vs. topological



https://youtu.be/ykOTfrS5pO4



Mapping

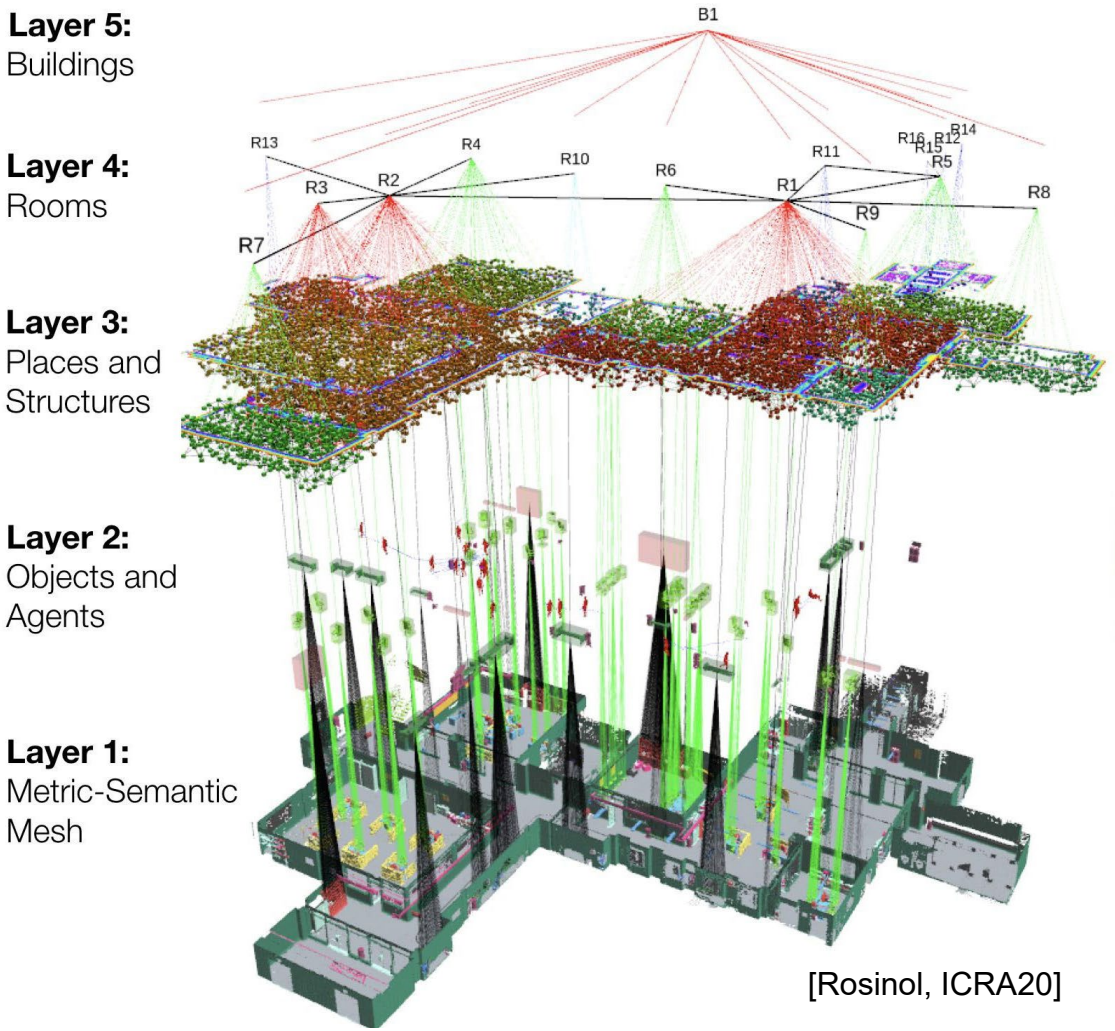
- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - Polygons/patches
  - Occupancy grid
  - **Semantic/objects**
  - Metric vs. topological

ScanNet scene0645_00 https://youtu.be/y2vzxm4SL2E

Speed x5

Mapping

- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - Polygons/patches
  - Occupancy grid
  - Semantic/objects
  - **Metric vs. topological**



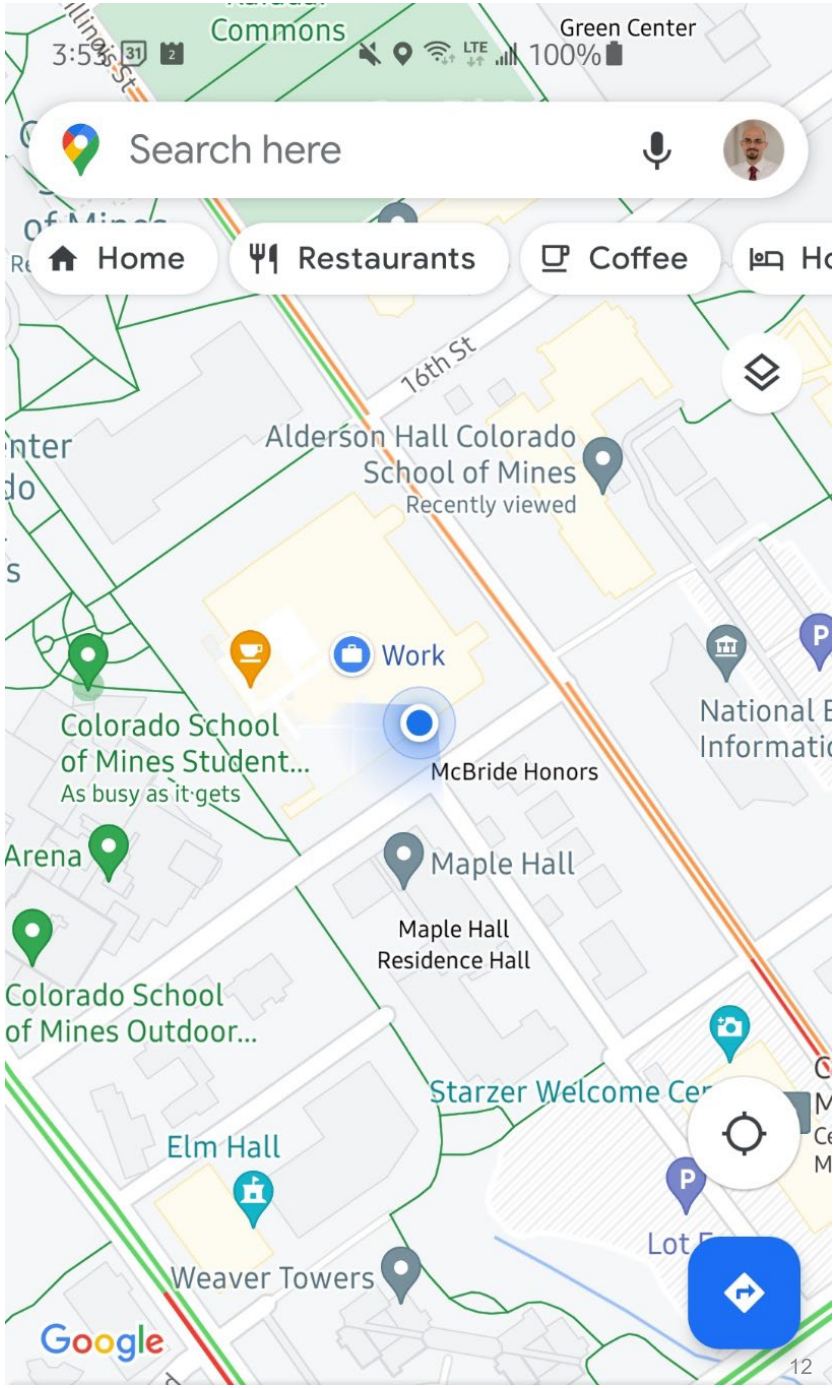
[Rosinol, ICRA20]

Localization

- **Localization**: is the process of determining precise **position** & **orientation** (aka **pose**) within environment

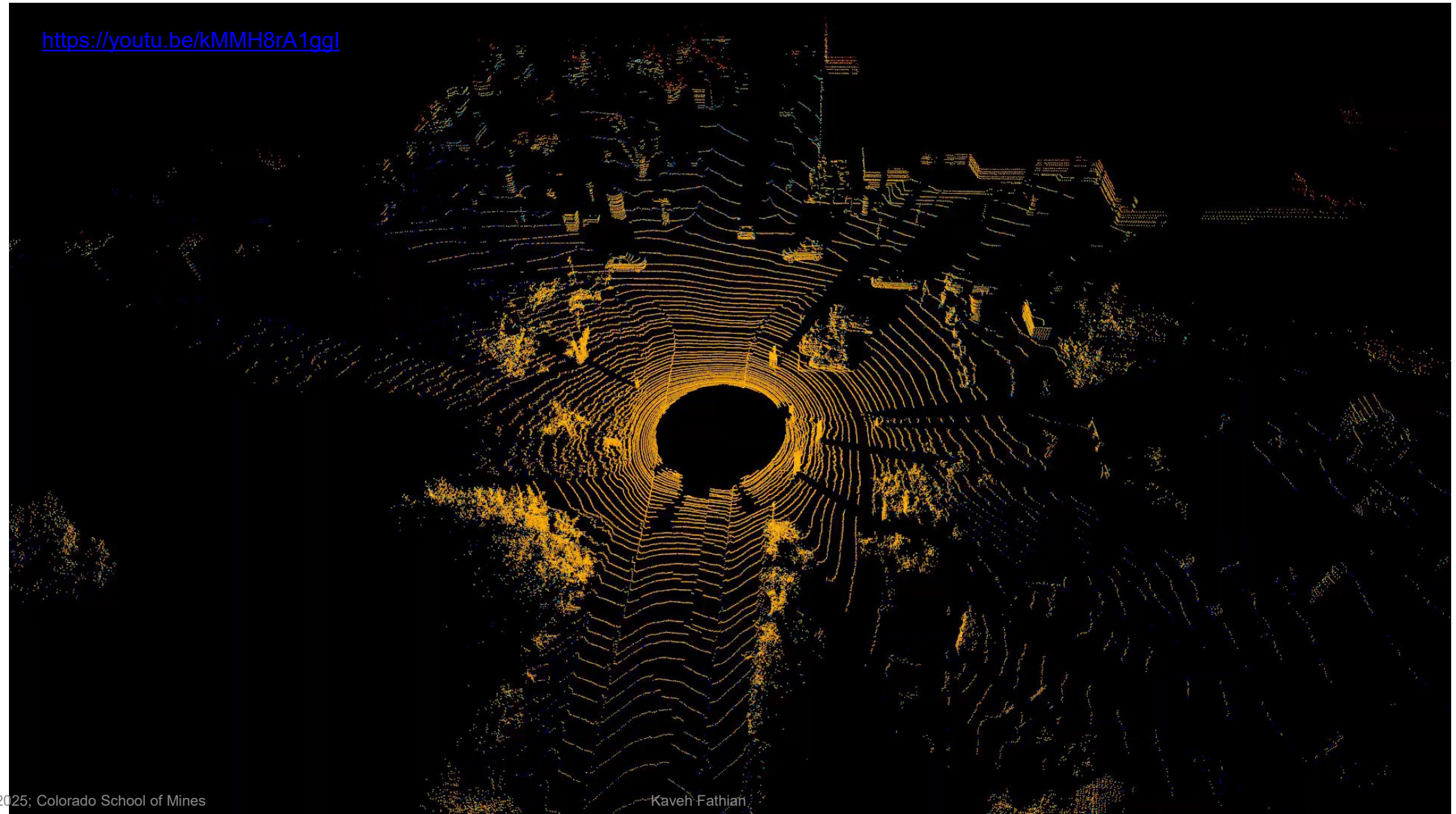
- Localization can be
  - In generated map
  - In given map—kidnapped robot problem!

- **Localization techniques**:
  - Particle Filters
  - Kalman Filters
  - Graph Optimization



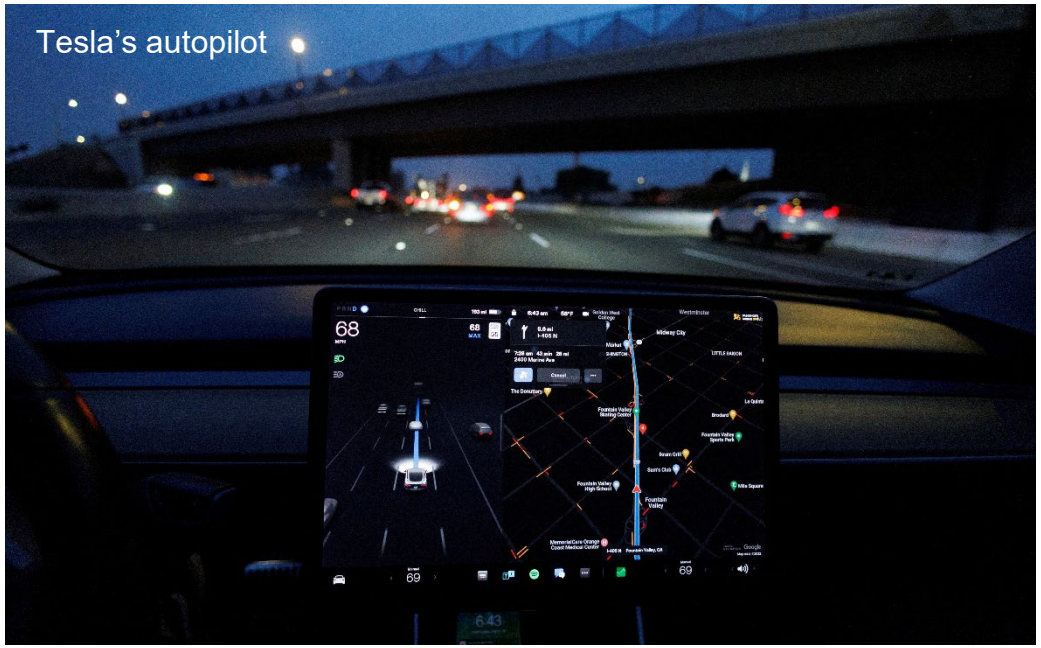
Simultaneous Localization & Mapping

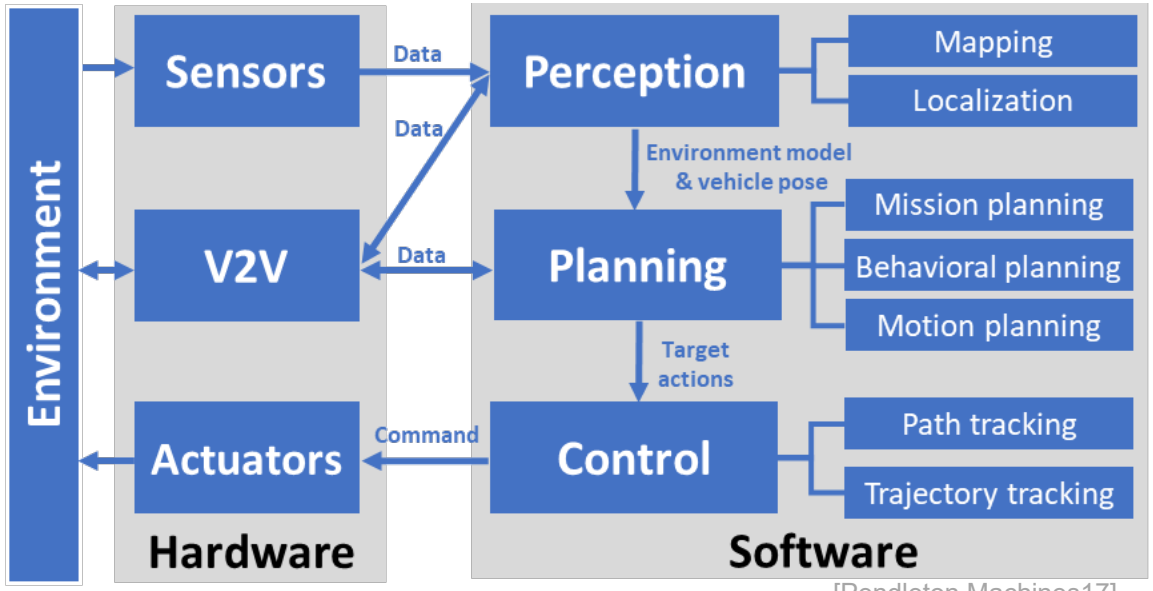
Mapping & localization happens **simultaneously** in SLAM



Why Do We Need SLAM?

- **Perception**/SLAM ➔ understanding of environment/surrounding
- **Autonomy** needs perception, planning, control





[Pendleton Machines17]

Why Do We Need SLAM?

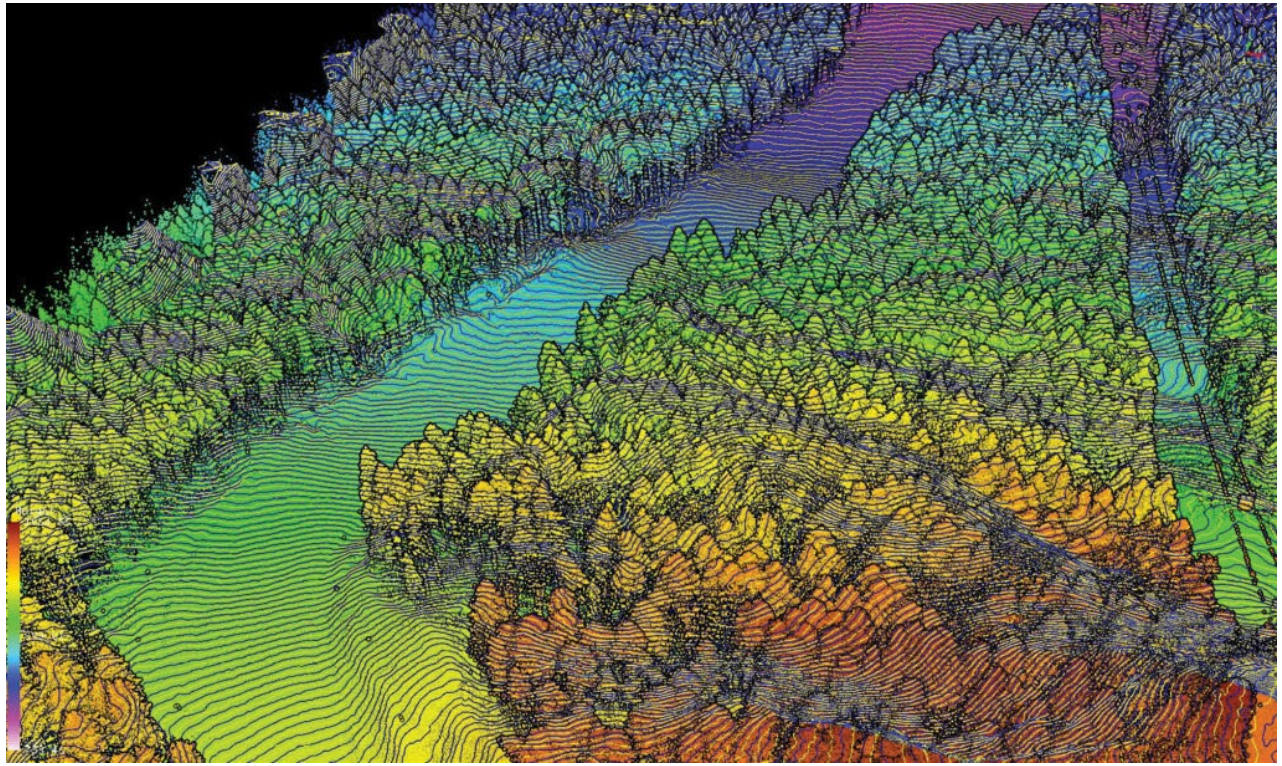
Perception is needed for

- **Robotics, autonomy**
  - **Self-driving cars; drones; autonomous robots**
- Surveying, mapping, exploration
  - Search & rescue missions
- Inspection, monitoring, surveillance
  - Construction, agriculture, military
- Augmented/virtual reality
- Healthcare

Why Do We Need SLAM?

Perception is needed for

- Robotics, autonomy
  - Self-driving cars; drones; autonomous robots
- **Surveying, mapping, exploration**
  - **Search & rescue missions**
- Inspection, monitoring, surveillance
  - Construction, agriculture, military
- Augmented/virtual reality
- Healthcare



https://lidarmag.com/

Why Do We Need SLAM?

Perception is needed for

- Robotics, autonomy
  - Self-driving cars; drones; autonomous robots
- Surveying, mapping, exploration
  - Search & rescue missions
- **Inspection, monitoring, surveillance**
  - **Construction, agriculture, military**
- Augmented/virtual reality
- Healthcare

Why Do We Need SLAM?

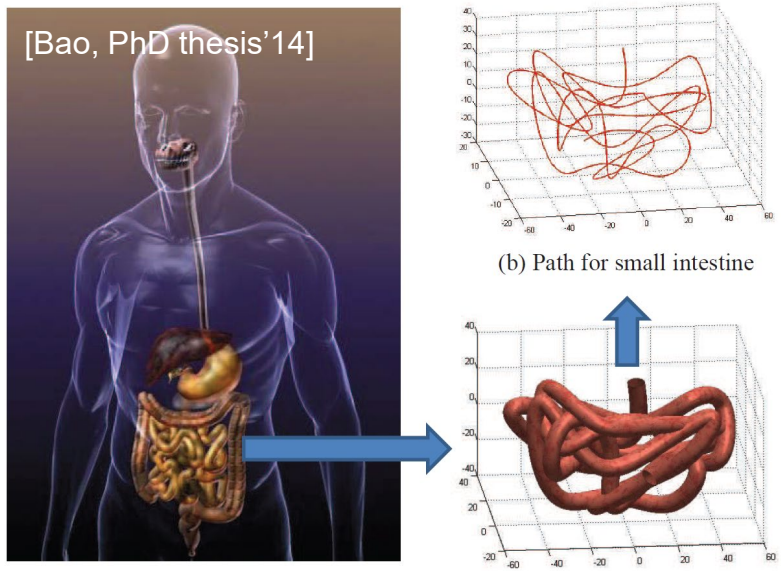
Perception is needed for

- Robotics, autonomy
  - Self-driving cars; drones; autonomous robots
- Surveying, mapping, exploration
  - Search & rescue missions
- Inspection, monitoring, surveillance
  - Construction, agriculture, military
- **Augmented/virtual reality**
- Healthcare

Why Do We Need SLAM?

Perception is required for

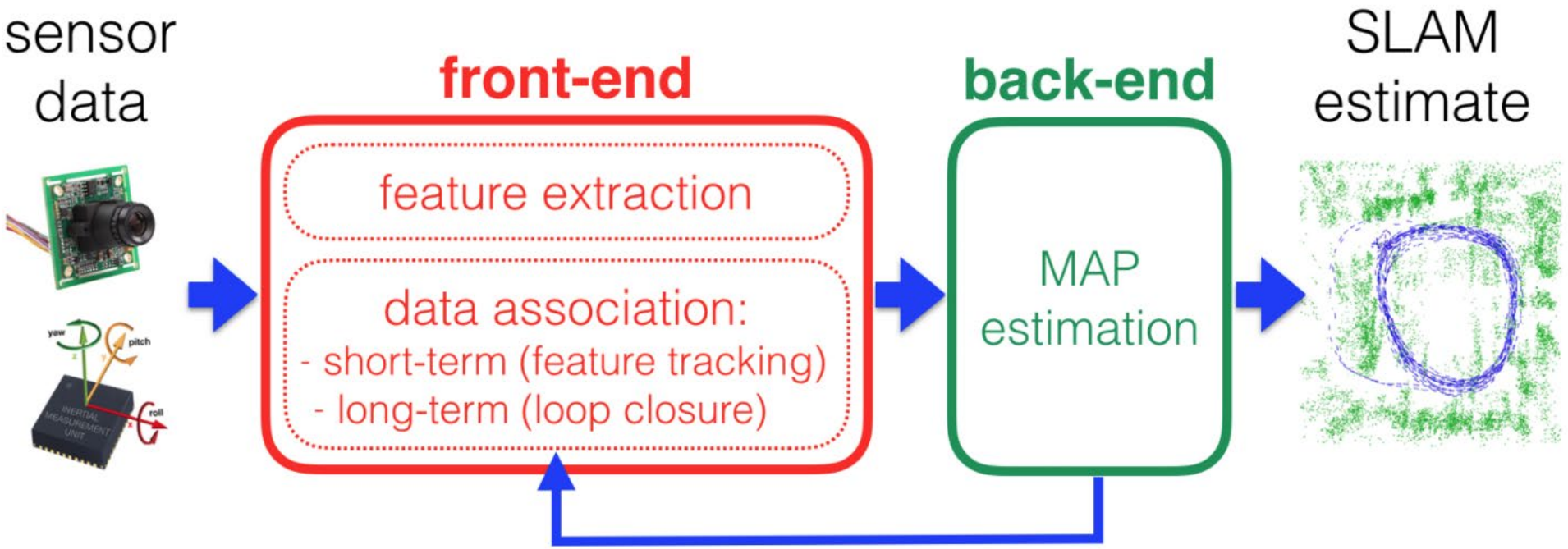
- Robotics, autonomy
  - Self-driving cars; drones; autonomous robots
- Surveying, mapping, exploration
  - Search & rescue missions
- Inspection, monitoring, surveillance
  - Construction, agriculture, military
- Augmented/virtual reality
- **Healthcare**



(b) Path for small intestine

Itero dental scanner: https://www.youtube.com/shorts/zYGeE6TM9Xk

SLAM Components



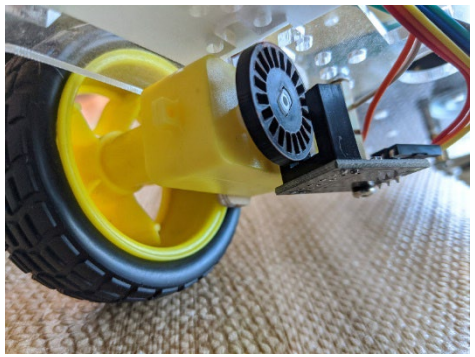
[Carlone, Visual Nav, MIT]

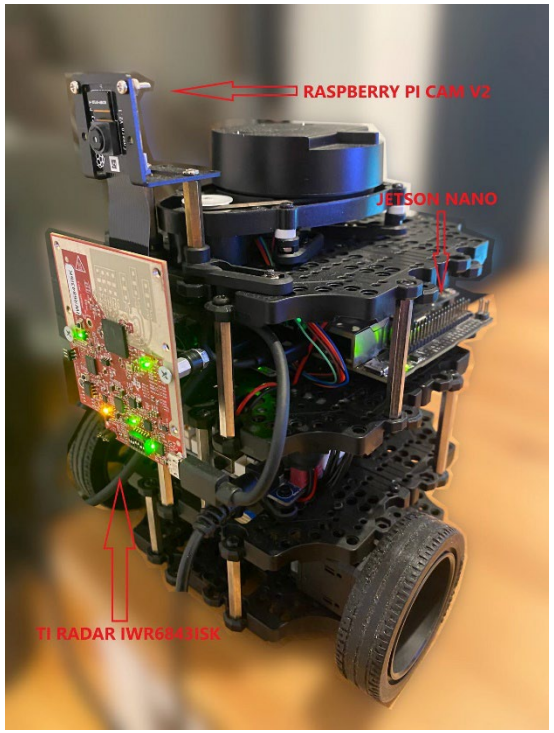
Standard SLAM pipeline consists of

- **Sensors**
- **Front-end** algorithms (real-time & online processing of sensor data, trajectory, map)
- **Back-end** algorithms (optimization & refinement of map, trajectory.)

Sensors

- **Proprioceptive** (perceive robot's internal state)
  - IMU (gyroscope, accelerometer, magnetometer)
  - Wheel encoder
- **Exteroceptive** (perceive external environment)
  - Camera (passive)
  - Lidar (active)
  - Radar (active)
  - Sonar (active)

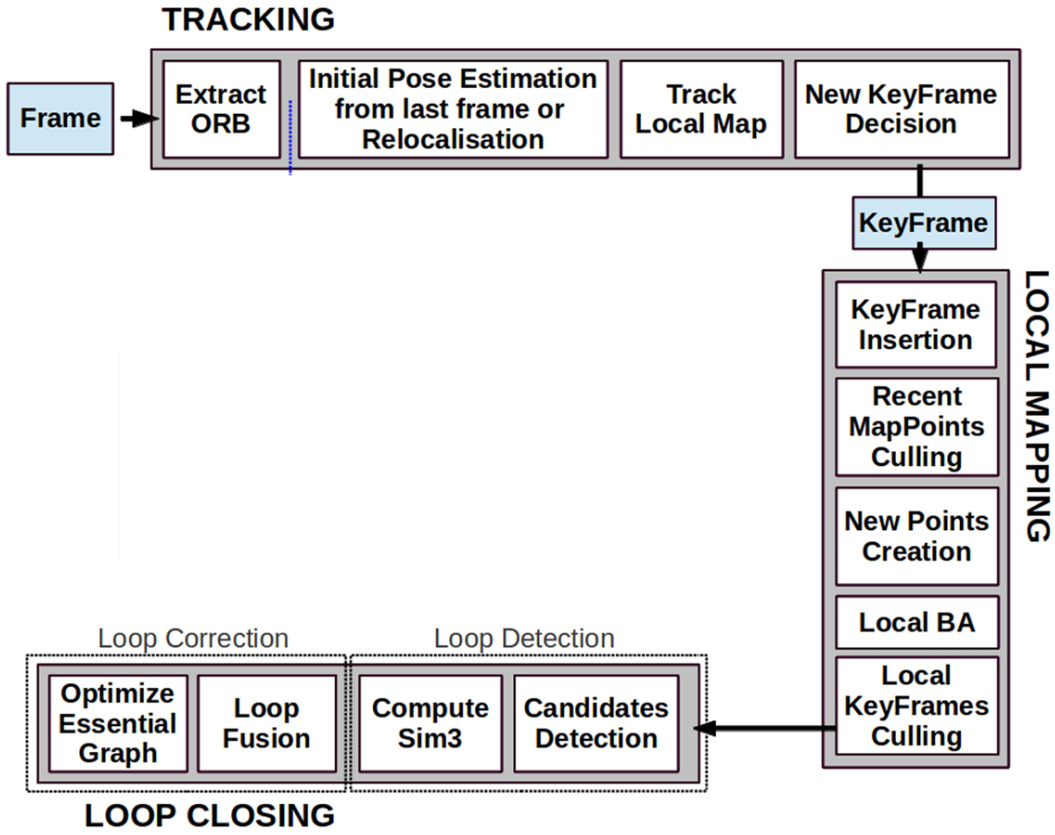




Front-End

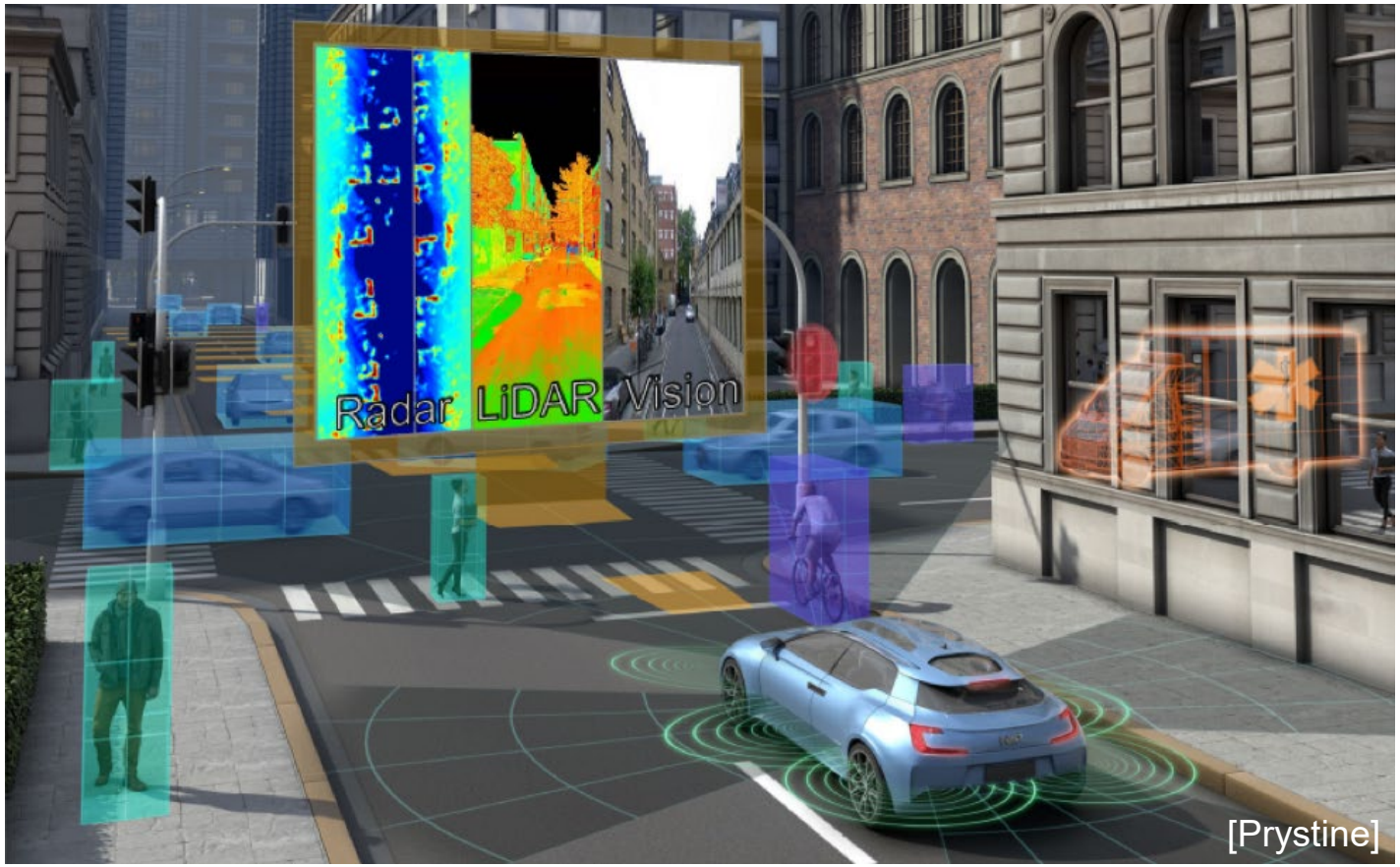
Front-end deals with real-time & online

- Processing of sensor data
- Feature extraction/tracking
- Data association
- Initial trajectory/pose estimate (odometry)
- Initial map estimate



Odometry Estimation

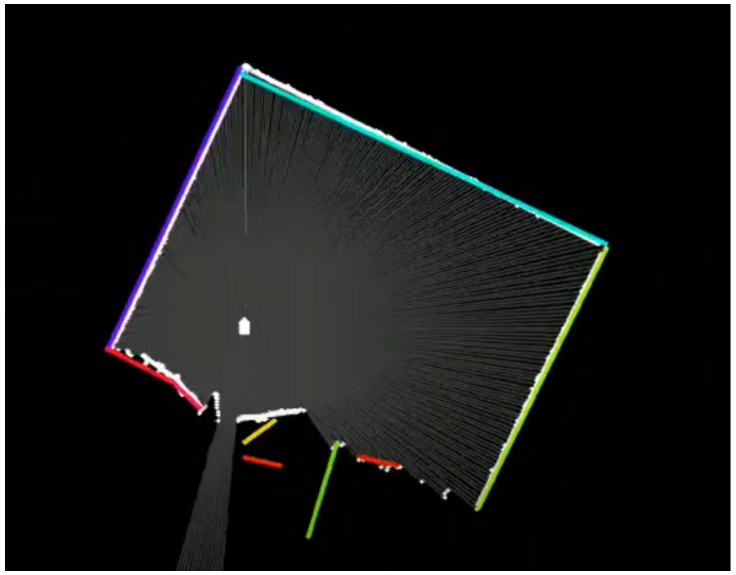
- Estimation techniques vary for each **sensing modality** (e.g., LiDAR, camera, radar)
- Even for a single modality, there are **many** estimation techniques:
  - Parametric vs. non-parametric
  - Model-base vs. learning-base

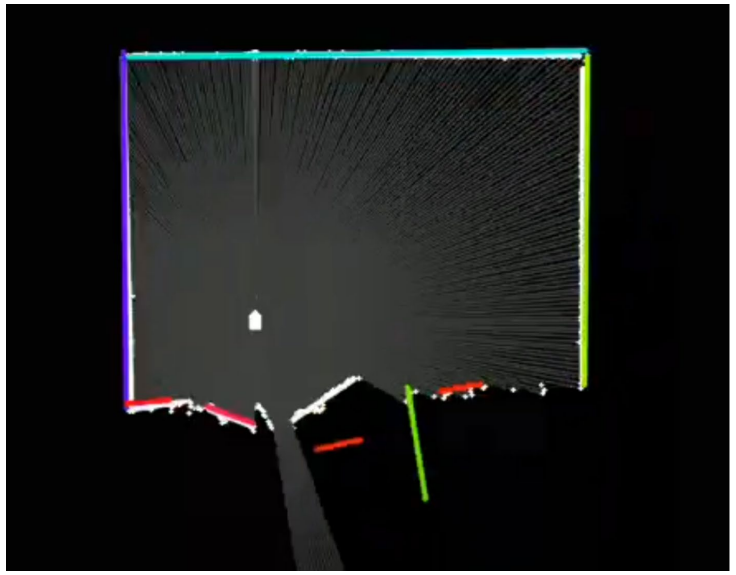


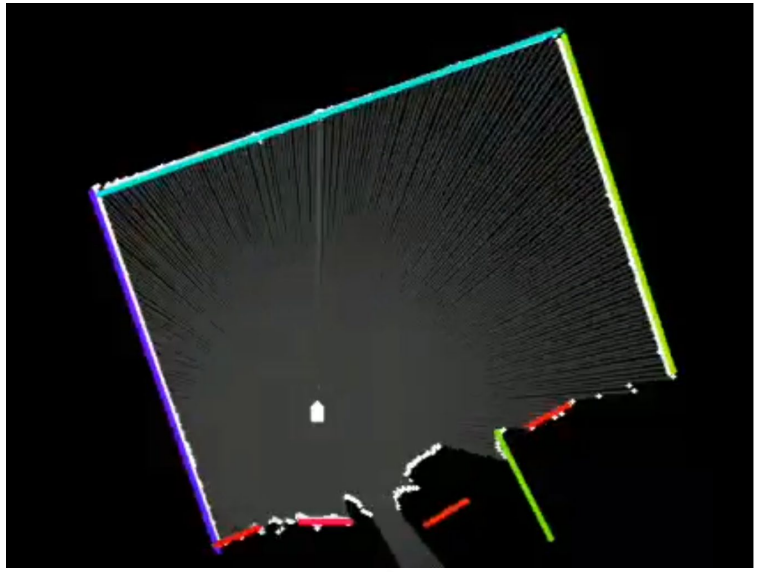
LiDAR Odometry

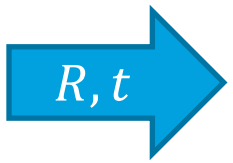
Objective: Want to estimate robot trajectory/pose from consecutive LiDAR scans & create a primitive 3D map

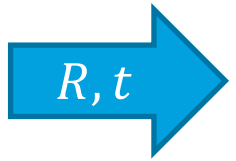
vigibot.com







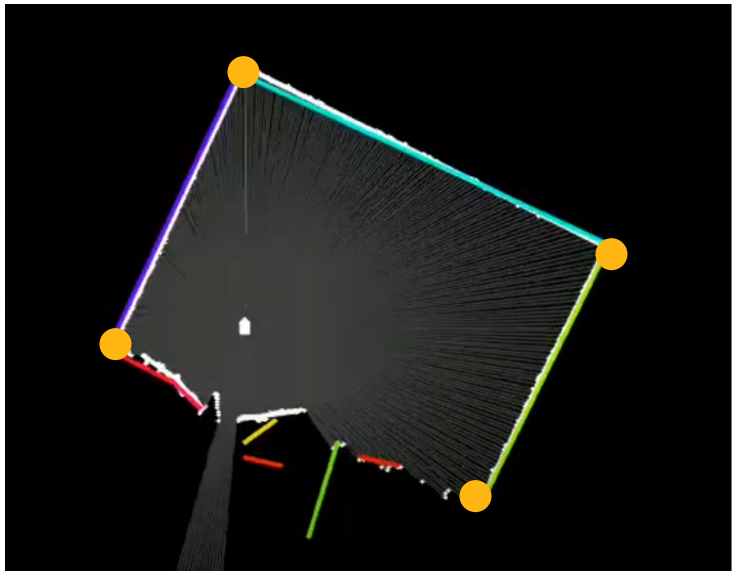


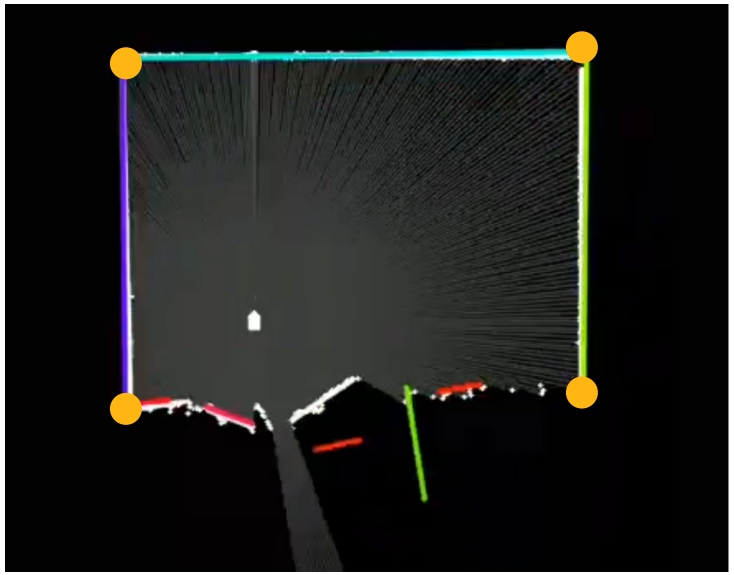


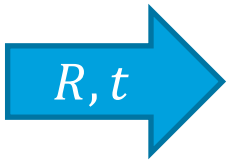
Lidar Odometry

Odometry estimation (parametric):

- **Associate** corresponding points across scans—**how**?
- Estimate the rigid transformation (R,t) from correspondences
  - Arun's method (aka Kabsch algorithm, orthogonal Procrustes problem) provides least-square solution [Arun, PAMI87]







Least-square formulation:

$$\min_{R,t} \sum_{i=1}^{N} \|p_i' - (R\, p_i + t)\|^2$$

Monocular Visual Odometry

Objective: Want to estimate robot trajectory/pose from consecutive images & create a primitive 3D map (similar to Lidar odometry)

Process:

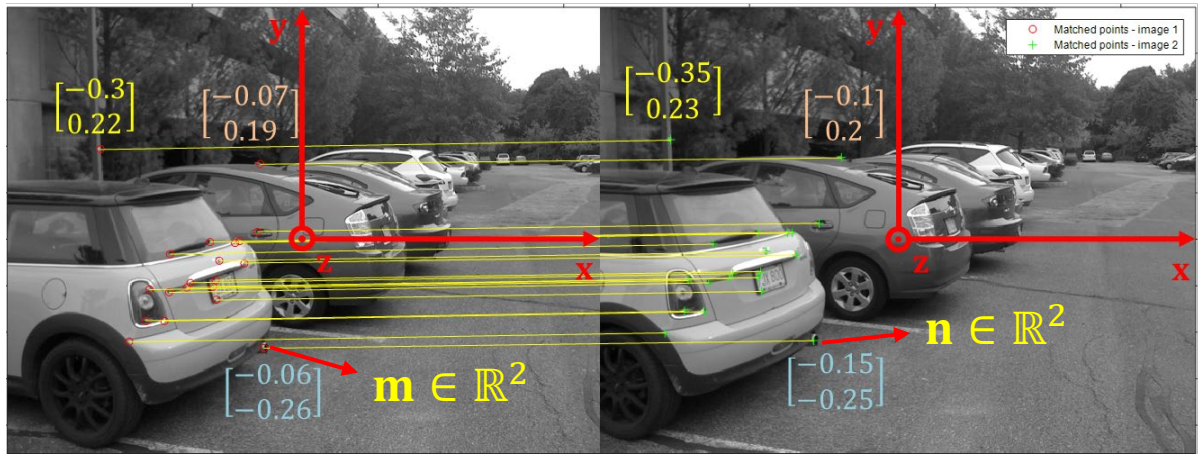
- Extract image features
- Match features across images (data association)
- Estimate relative pose—how?
- Reconstruct/triangulate features in 3D (landmarks)

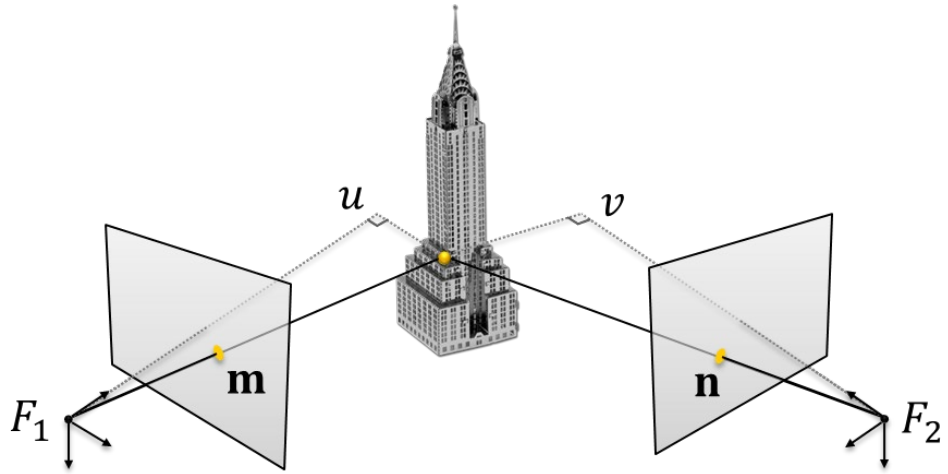
Monocular Visual Odometry

Relative pose estimation: Based on "rigid motion constraint" for each pair of *matched feature points:*

$$u\,\mathbf{R}\,\mathbf{m} + \mathbf{t} = v\,\mathbf{n}$$

Unknown variables



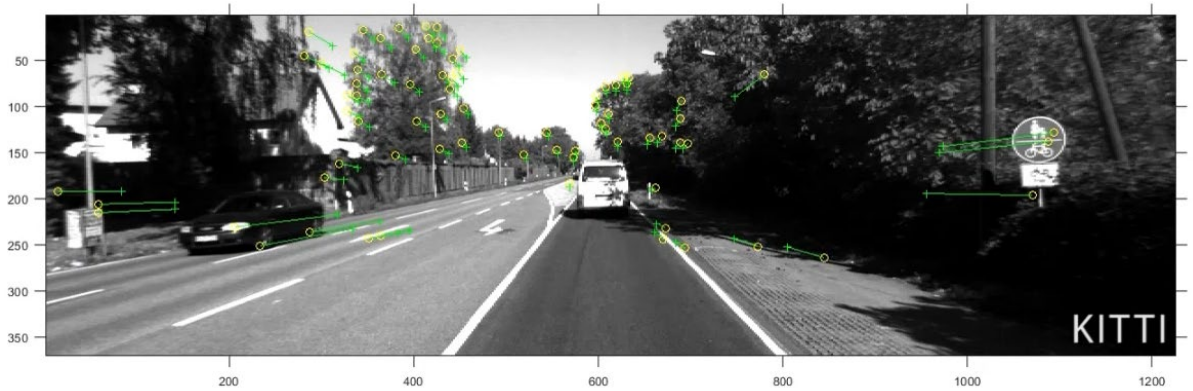


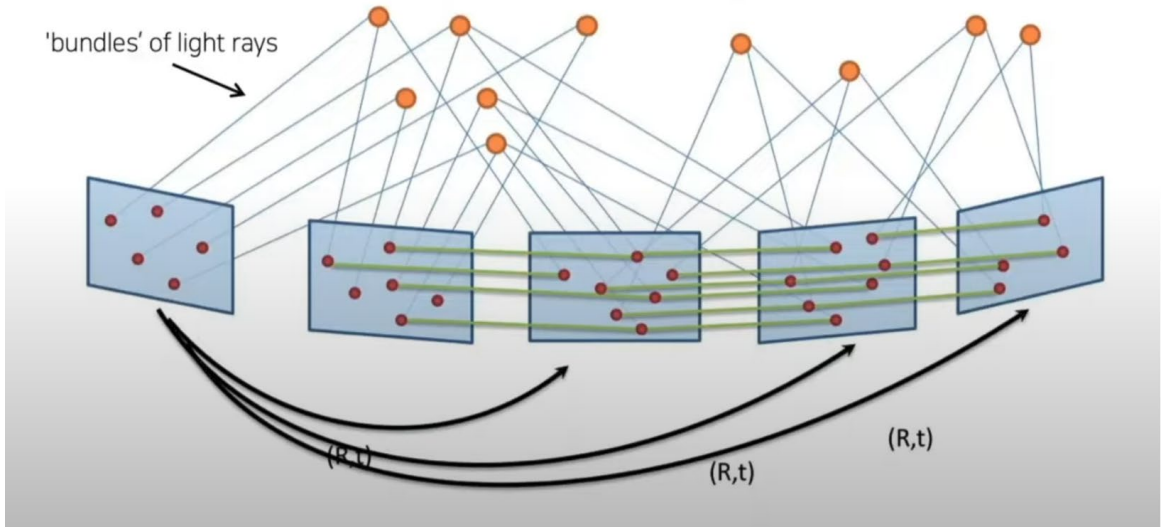
Algorithms:

- 8pt algorithm [Longuet-Higgins Nature81]
- 4pt algorithm (homography) [Hartley ECCV94]
- 5pt algorithms [Nister CVPR03; Stewenius JPRS06; Li ICPR06; Kukelova BMVC08]
- Direct estimation [Kneip Springer12]
- QuEst [Fathian RAL18]

Loop Closure

Front-end or short-term (feature tracking)



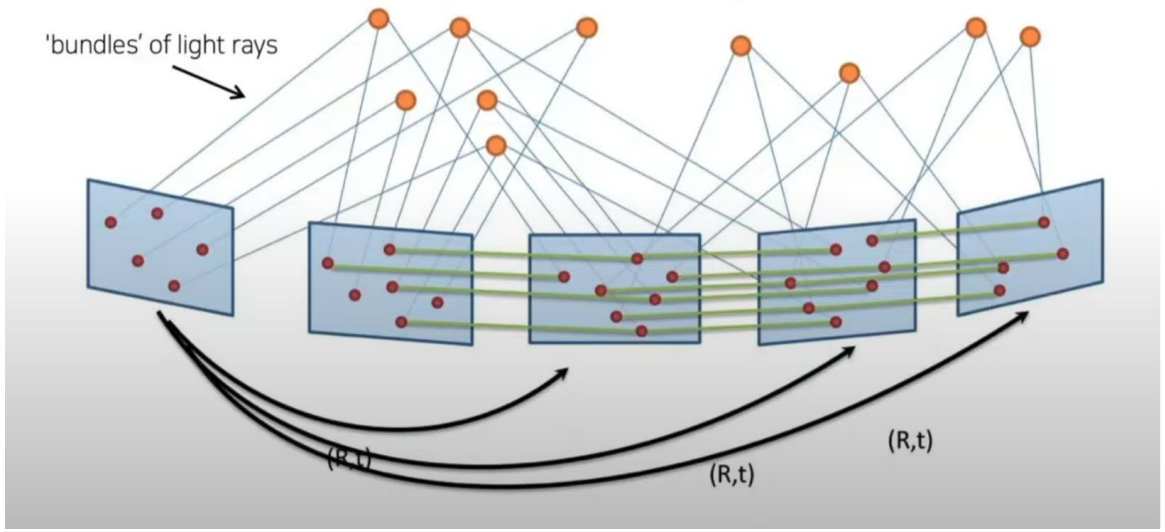


https://youtu.be/z4ldKGh12ok

https://youtu.be/OV6wNr62nqQ

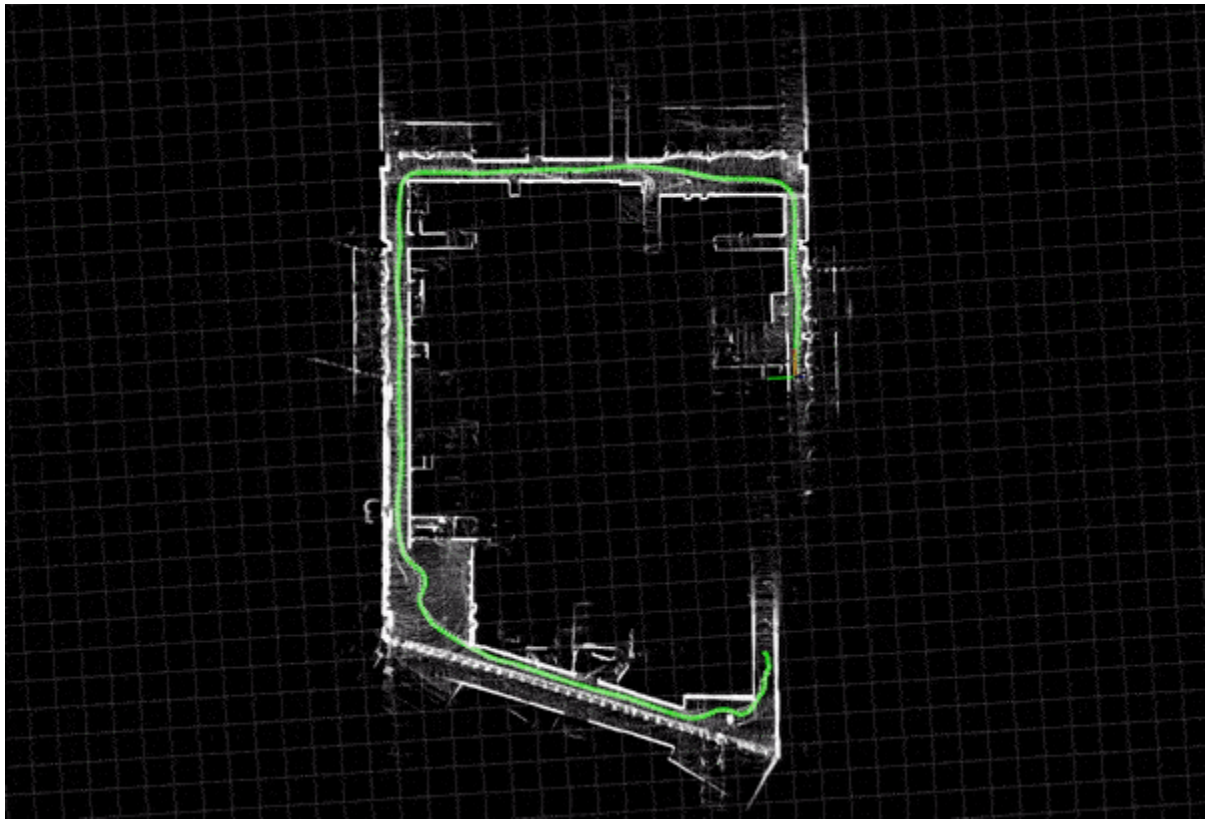
Loop Closure

Front-end or short-term (feature tracking)



https://youtu.be/z4ldKGh12ok

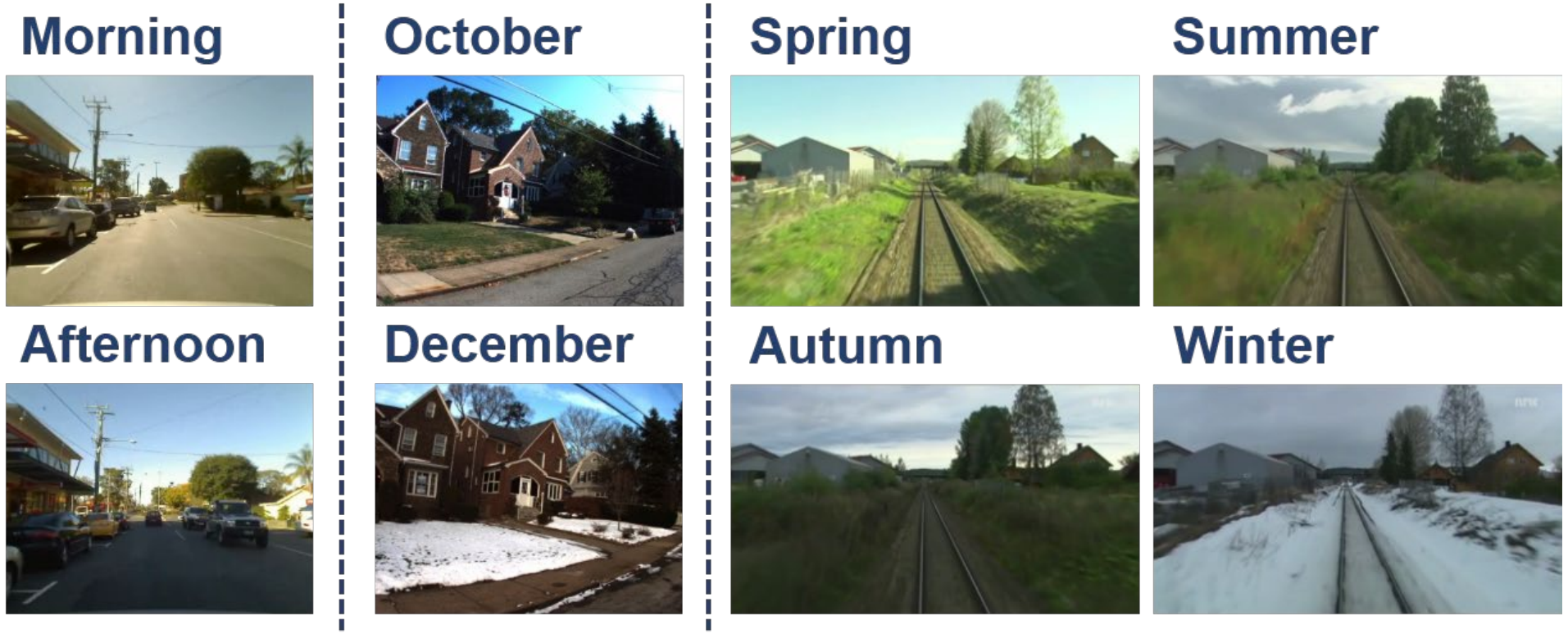
Back-end or long-term (loop closure detection)



https://youtu.be/OV6wNr62nqQ

Loop Closure—Challenges

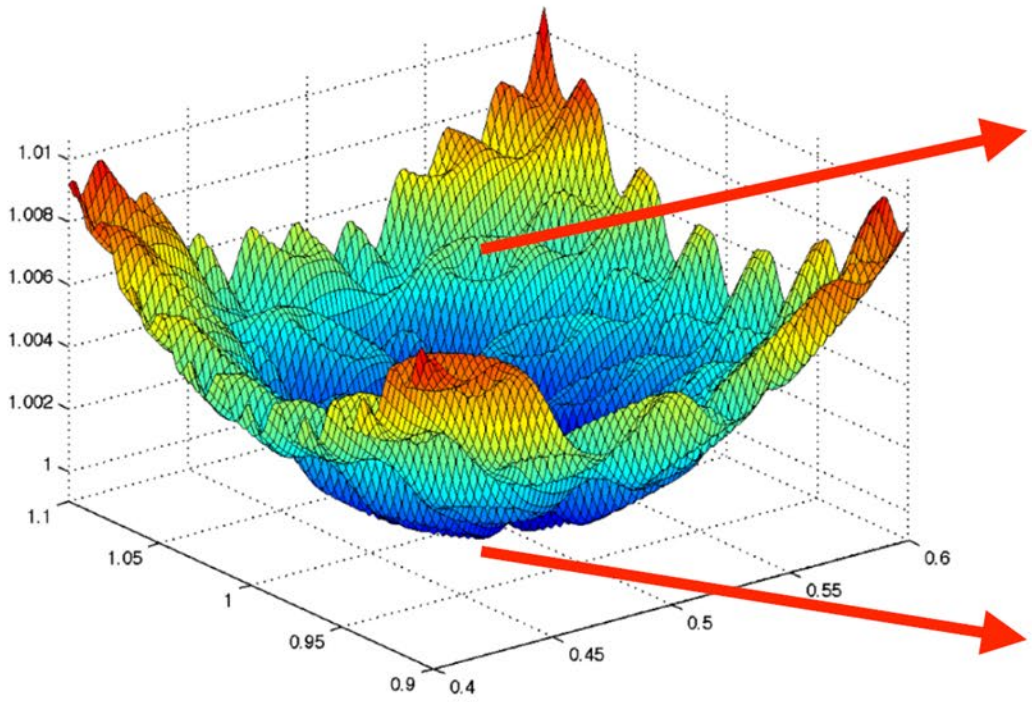
- Loop closure detection has many challenges:
  - Seasons; time of day; weather; viewpoint; dynamic environment, etc.
- Several methods: Bag of Words (BoW), deep-learning-based, geometric matching, etc.
- No perfect solution; all methods have pros & cons
- Wrong loop closure is catastrophic! ➔ need outlier removal

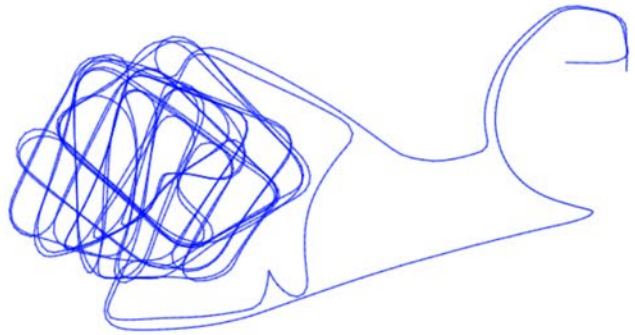


Back-End

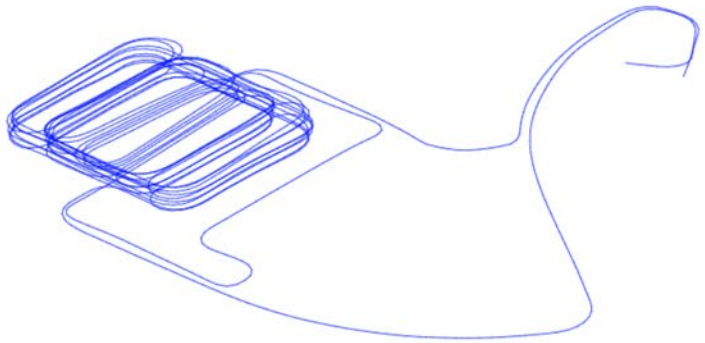
Back-end: Deals with

- Optimization/refinement of map & trajectory (for **consistency**)
- Finding loop closures correcting drift





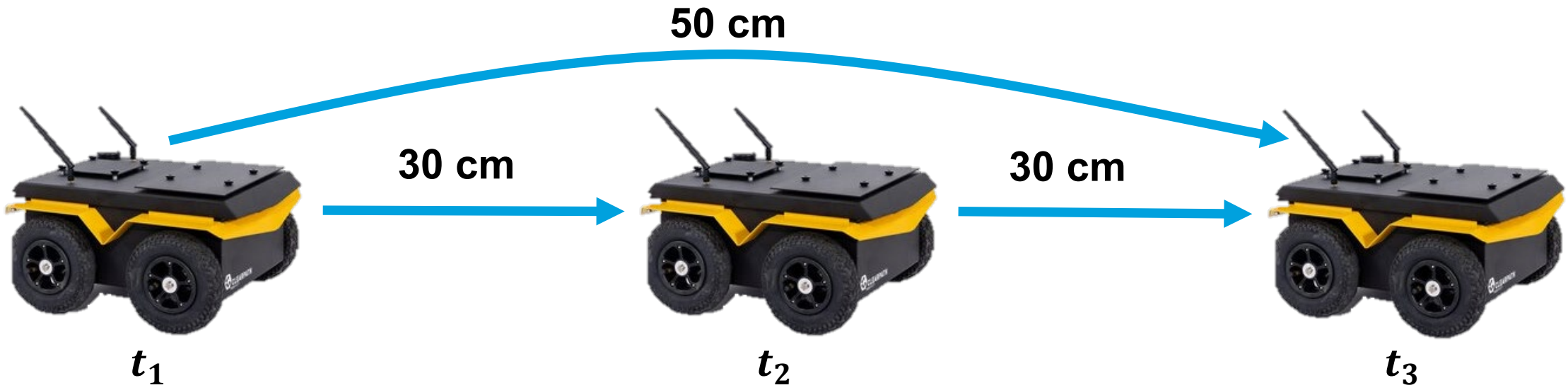
Suboptimal critical point



Optimal estimate

Geometric Consistency

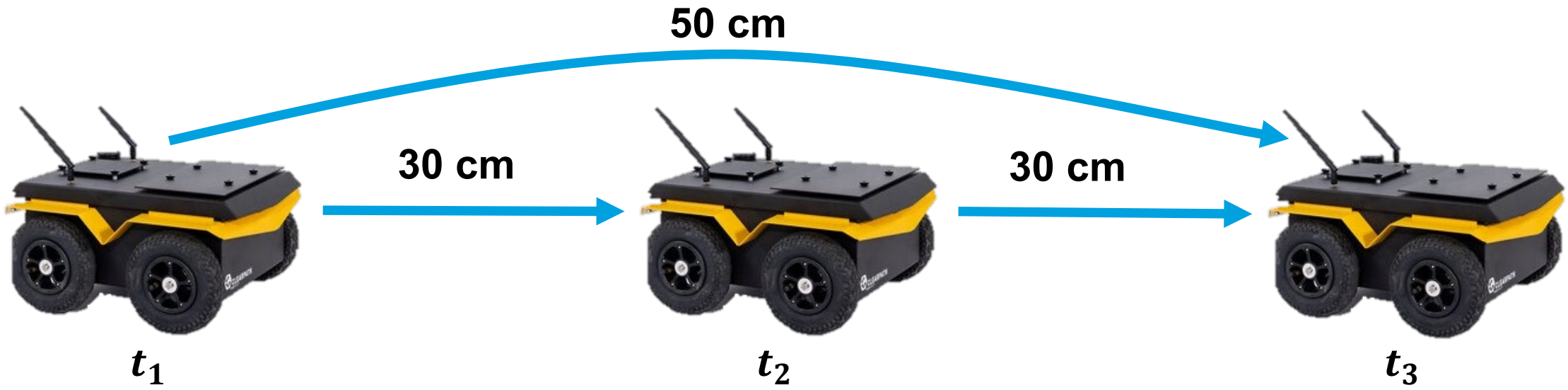
Example: Suppose we compute odometry estimates between $(t_1, t_2), (t_2, t_3), (t_1, t_3)$



Geometric Consistency

Example: Suppose we compute odometry estimates between $(t_1, t_2), (t_2, t_3), (t_1, t_3)$

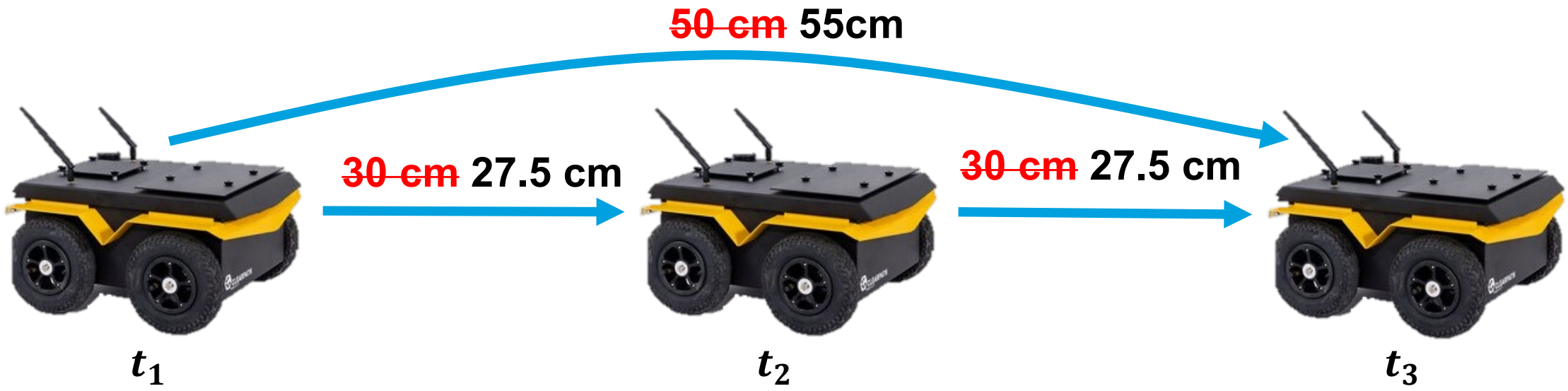
- Estimates can be geometrically inconsistent (always are in practice!)



Geometric Consistency

Example: Suppose we compute odometry estimates between $(t_1, t_2), (t_2, t_3), (t_1, t_3)$

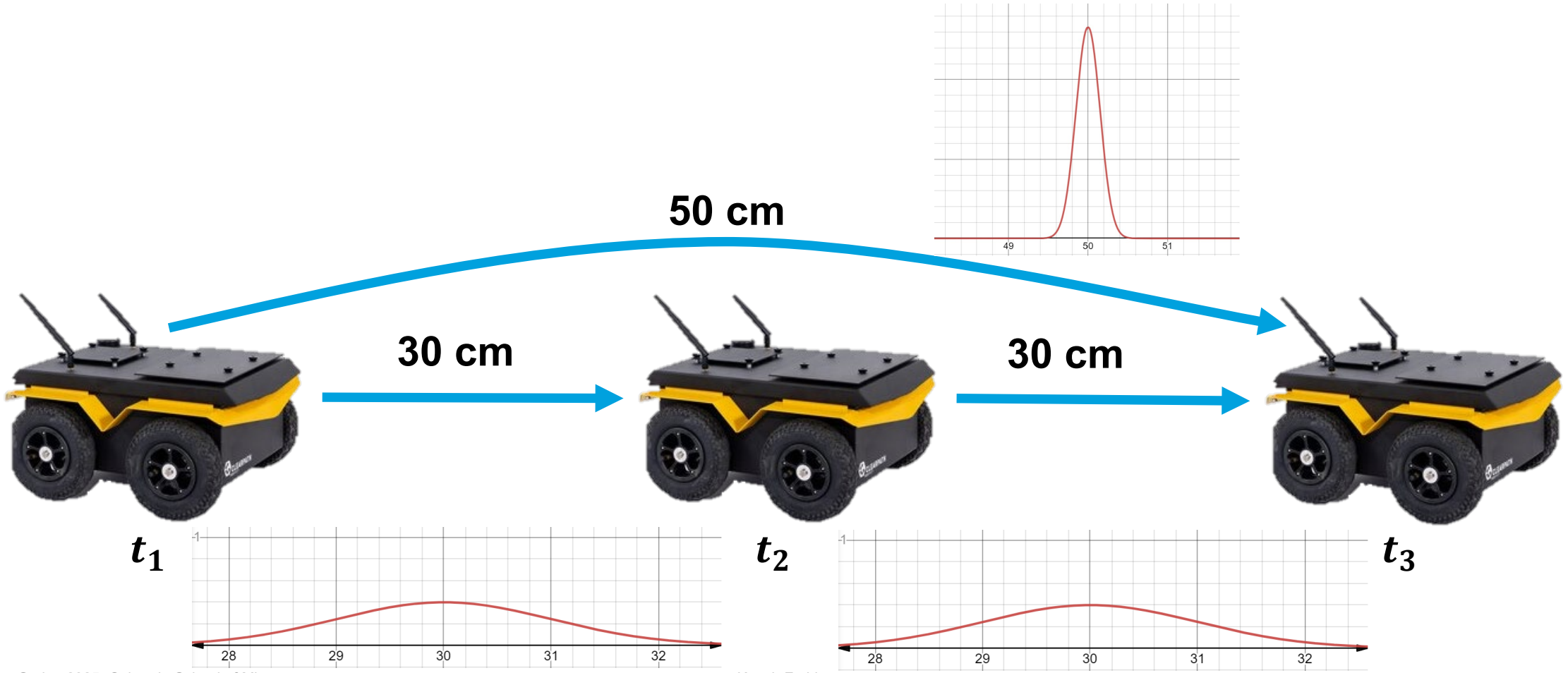
- Estimates can be geometrically inconsistent (always are in practice!)
- We can take "**average**" of the estimates to obtain consistent results



Geometric Consistency

Example: Suppose we compute odometry estimates between $(t_1, t_2), (t_2, t_3), (t_1, t_3)$

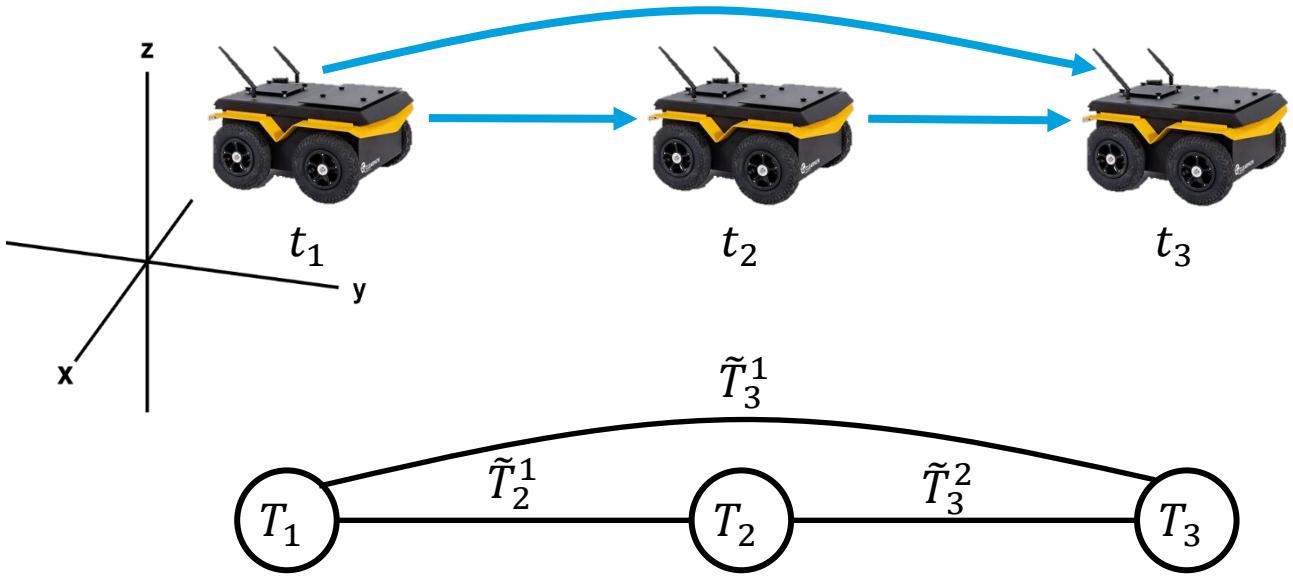
- Estimates can be geometrically inconsistent (always are in practice!)
- We can take "**average**" of the estimates to obtain consistent results
- What if we have a **probability** for correctness of estimates?



Factor Graph Optimization

Factor graph: Mathematical framework for solving estimation/inference problems & obtain consistency

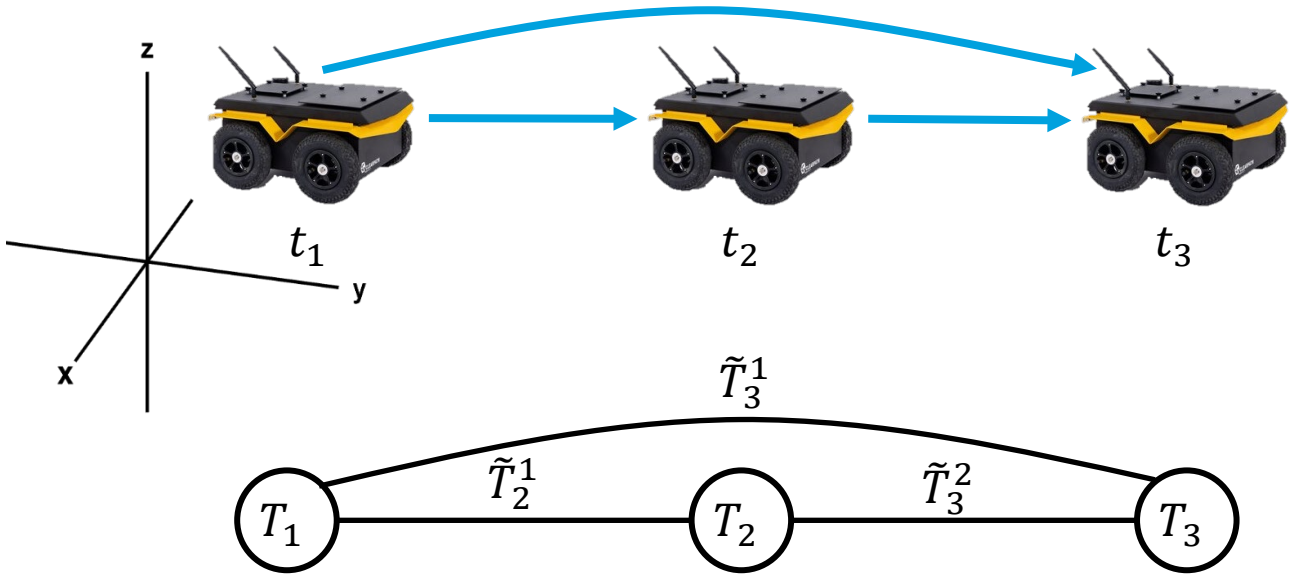
- Problem representation as a graph
- **Nodes/Vertices**: are optimization variables. Can be of different types (continuous or discrete, poses or landmarks, etc.)
- **Factors/Edges**: represent relationship/constraints between variables (e.g. relative pose estimates, sensor measurements)



Factor Graph Optimization

Factor graph: Mathematical framework for solving estimation/inference problems & obtain consistency

- Problem representation as a graph
- **Nodes/Vertices**: are optimization variables. Can be of different types (continuous or discrete, poses or landmarks, etc.)
- **Factors/Edges**: represent relationship/constraints between variables (e.g. relative pose estimates, sensor measurements)



Pose/factor graph optimization:

$$\min_{T_t \in \mathrm{SE}(3)} \sum_{(i,j)\in\varepsilon} \left\| T_i^{-1} T_j - \tilde{T}_{ij} \right\|_{\Sigma_{ij}}^2$$

Solvers:

- GTSAM (https://gtsam.org/)
- G2o (https://openslam-org.github.io/g2o.html)
- Ceres (http://ceres-solver.org/)

Course Objectives

- Learn fundamental techniques in 3D mapping & localization (SLAM, structure from motion, outlier rejection, robust estimation, semantic understanding,…)
- Learn/develop/exercise theoretical tools necessary for robotics research (optimization, linear algebra, graph theory, …)
- Learn/practice C++, ROS (Robot Operating System)
- Implement SOTA localization & mapping algorithms
- Get an overview of open problems in the field
- Possibility to use physical robots for final projects (Clearpath Jackal/Husky)


[iRobot]


[Amazon]


[Forbes]

Self-driving cars, drones, domestic and service robots, virtual and augmented reality, etc.

Warning!

1) You must have **math background**: calculus, linear algebra, probability, and statistics

2) You must have **coding** experience: C++, Python, algorithms

3) If you do not like coding (in C++/Python), you will not like this class!

4) This is not an ML course (but we will use ML in some lectures)

Resources

- No required textbook; references will be provided each lecture

- Lecture slides, notes, code, will be available at:
  https://www.ariarobotics.com/courses

https://www.ariarobotics.com/courses

Summary

In “**Robotic Mapping & Localization**”:

- You need knowledge of linear algebra, calculus, probability, statistics
- You need knowledge of algorithms, Linux, C++, and Python
- Will briefly review optimization, graph theory, C++
- Will introduce the Robot Operating System (ROS)
- Will cover front-end & back-end SLAM techniques in depth
- You will implement a SOTA SLAM pipeline & improve it (as final project)
- Opportunity to work with robots & publish academic papers

www.ariarobotics.com