Ain Shams University
Faculty of Engineering
Computer And Systems Engineering

Authors:
David Nader
George Mohsen
Kirollos Henry
Kirollos Wadie

Supervisor:
Dr. Hossam Hassan

Autonomous Robot for Mapping Unknown Environments (SLAM)

Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

Problem

- Build a robot that can autonomously explore an unknown environment and **build a 2D map** of it
- Uses include: exploring inaccessible places (e.g. caves), dangerous places (e.g. mine detection), mapping indoor locations, etc…
- Main problem: SLAM & Autonomous navigation

Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

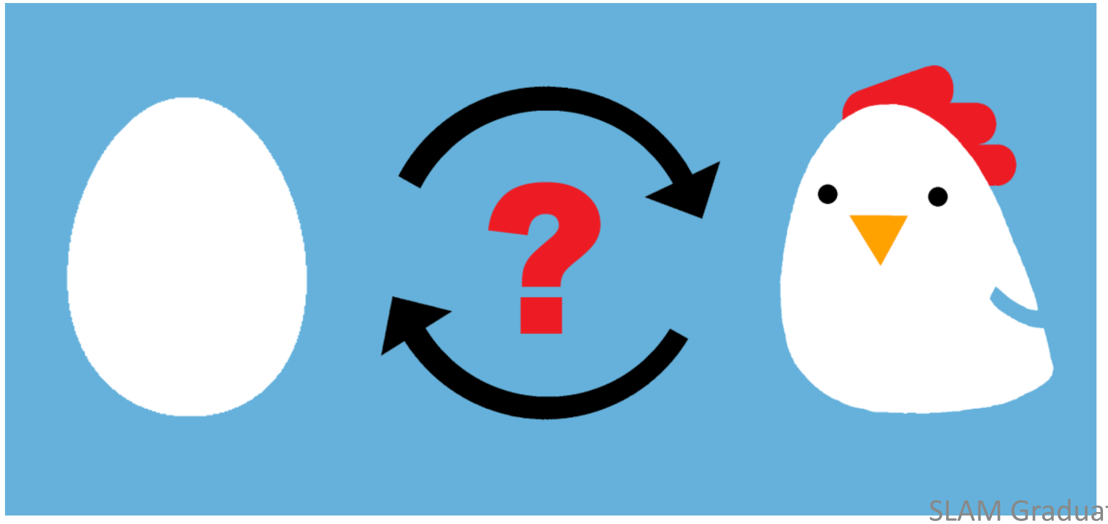
SLAM - Overview

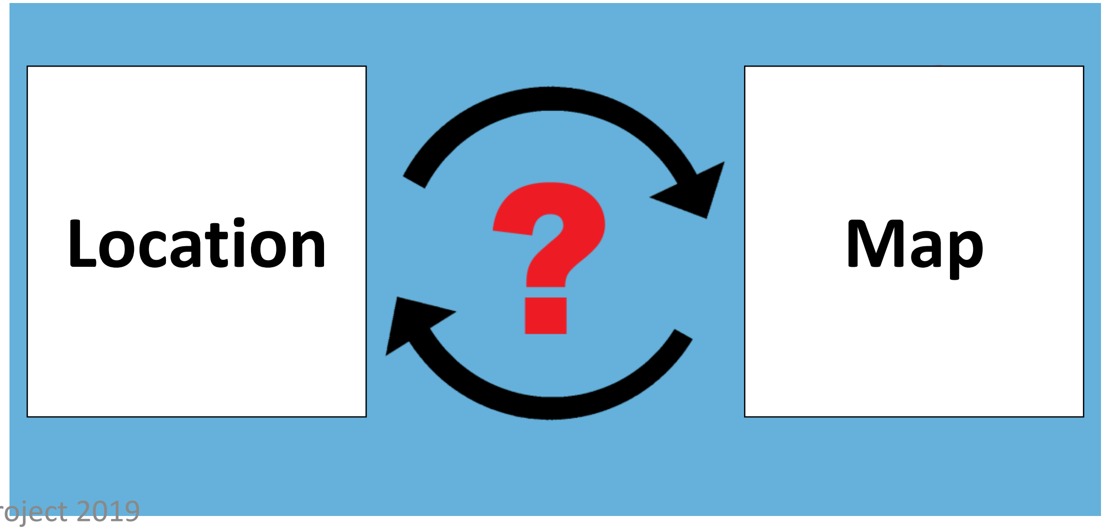
- Simultaneous Localization And Mapping
- Building a map of an unknown environment while keeping track of the position of the robot in it
- Well-known problem in autonomous robots
  - e.g. self-driving cars

SLAM - Overview

- Why “simultaneous”?
    - Need map to localize, and
    - Need location to map

- Called *chicken-or-egg* problem





SLAM - Overview

- Many methods exist to solve the SLAM problem
- Examples: Kalman filters, Particle filters, Graph SLAM, etc.
- We used *Hector SLAM* algorithm

SLAM - LIDAR

- Range-sensor used in many SLAM algorithms
- Key advantages:
  - Accuracy and resolution
    (e.g. <1° resolution)
  - High scan rate
    (e.g. 8000 samples/second)
  - Single sensor to measure in 360°

RPLIDAR A2

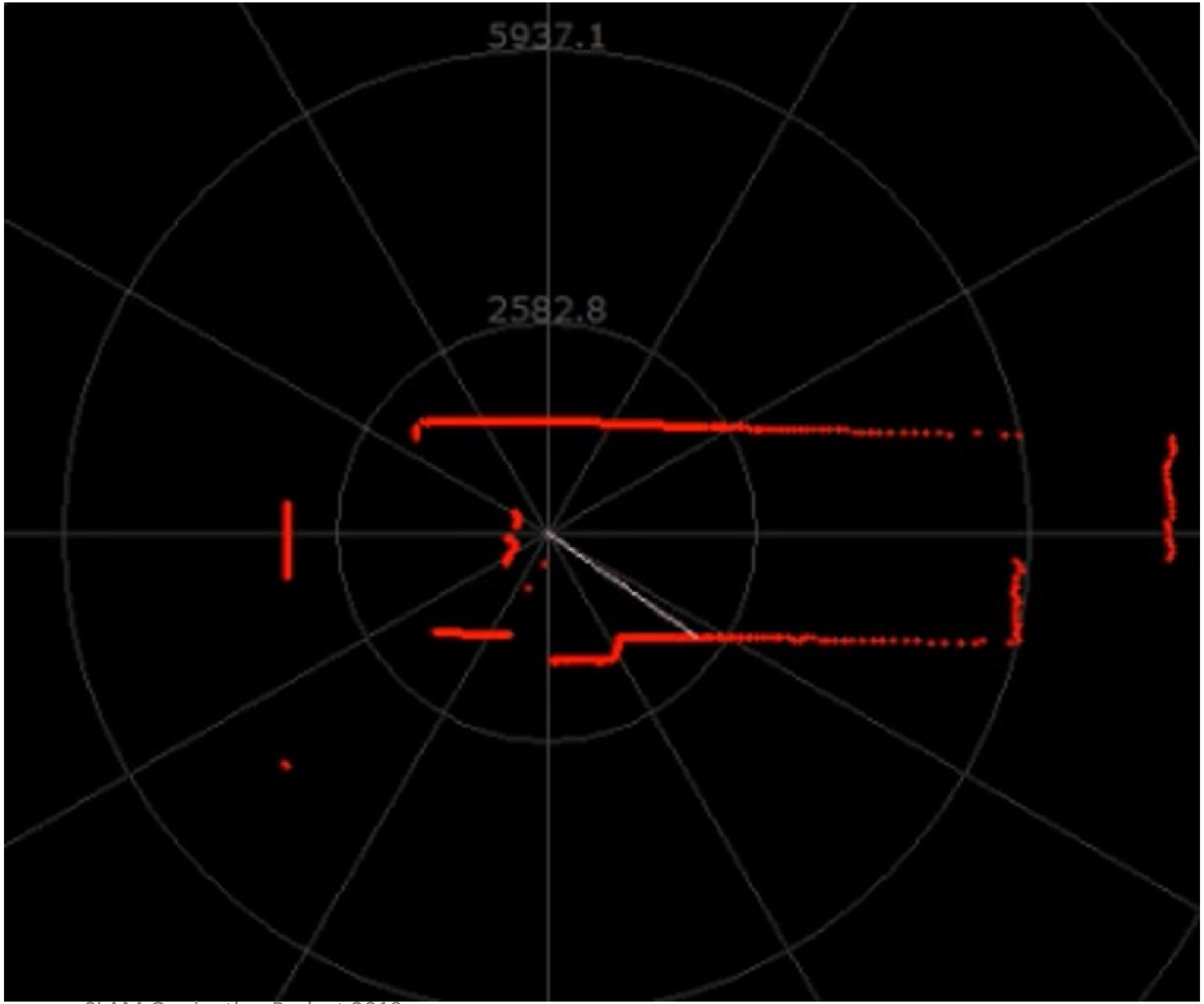
SLAM - LIDAR

- Compare to SONAR/Visual SLAM
  - Lower accuracy and resolution
  - Require multiple sensors for 360° ranging
  - Slower scan rate
  - Visual SLAM: no ranging directly
  - Visual SLAM: higher computation

RPLIDAR A2

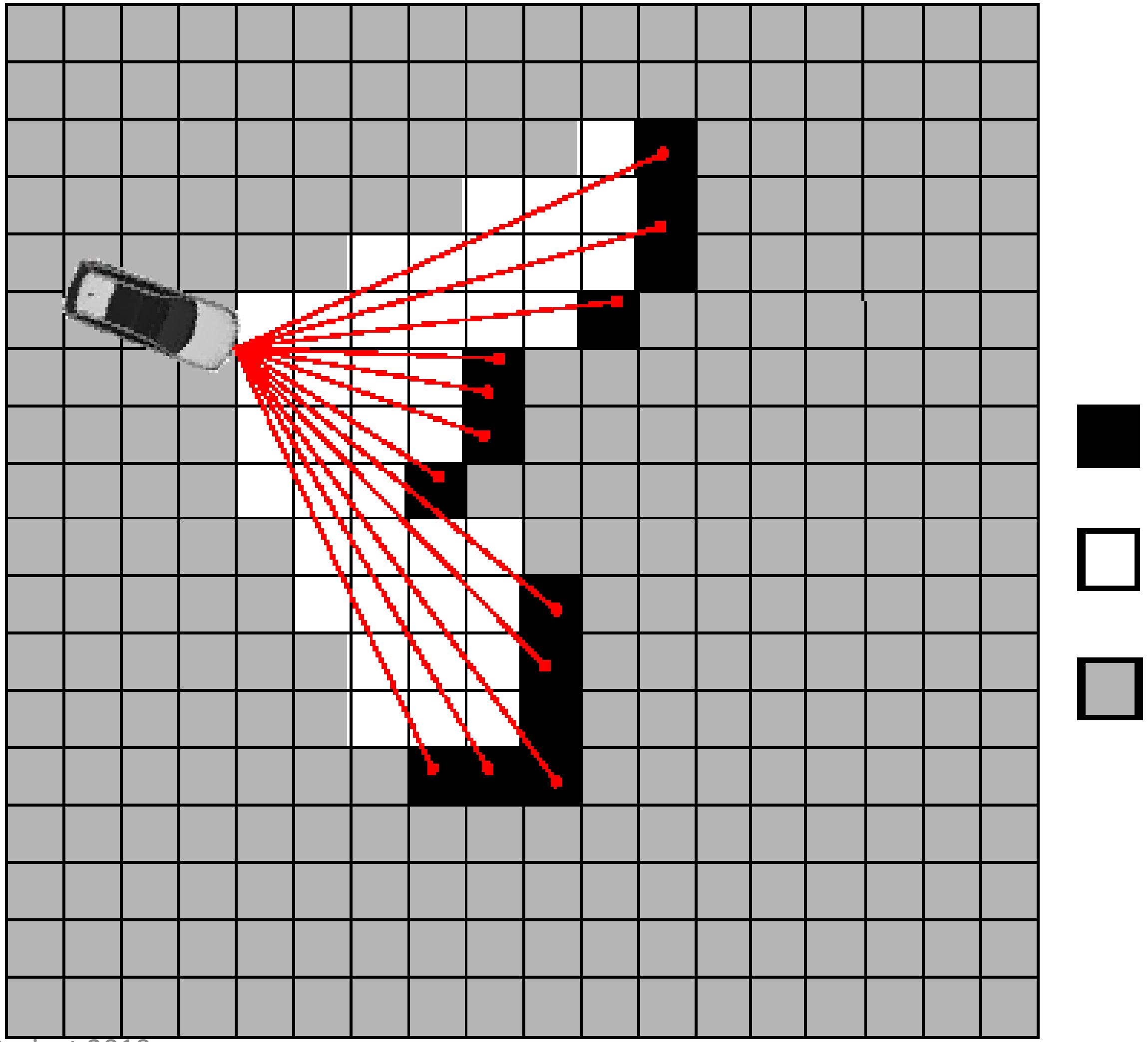
SLAM - LIDAR

- Sample output
- Can be visualized as polar coordinates (angle + magnitude)



SLAM - Occupancy Grid Map

- SLAM algorithms can produce different **types** of maps
- We're interested in Occupancy Grid maps
- Idea is: discretize the environment into grid cells (e.g. 10cm x 10cm)
- For each cell, we have one of 3 states (occupied, free, unknown)



Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

Hector SLAM - Overview

- Algorithm developed by *Team Hector* from “Technische Universität Darmstadt” (Technical University of Darmstadt) in Germany
- Presented in a published paper:

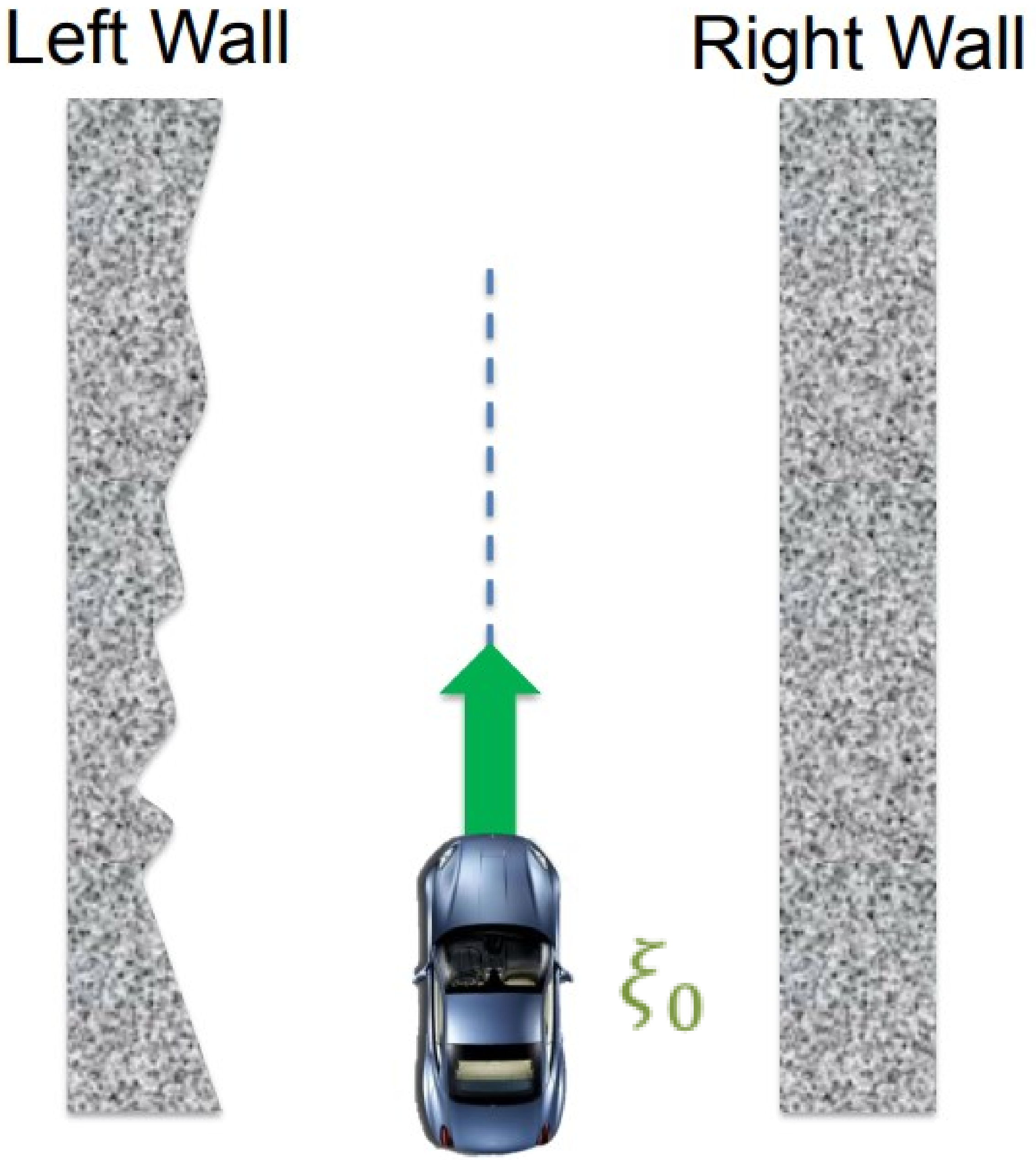
  Kohlbrecher, Stefan & Von Stryk, Oskar & Meyer, Johannes & Klingauf, Uwe. (2011). A flexible and scalable SLAM system with full 3D motion estimation. 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).

- Has an open-source implementation in ROS
- Does not require odometry like many other algorithms

Hector SLAM - Overview

- Initial position
- For motion in 2D plane, we have 3 DOF

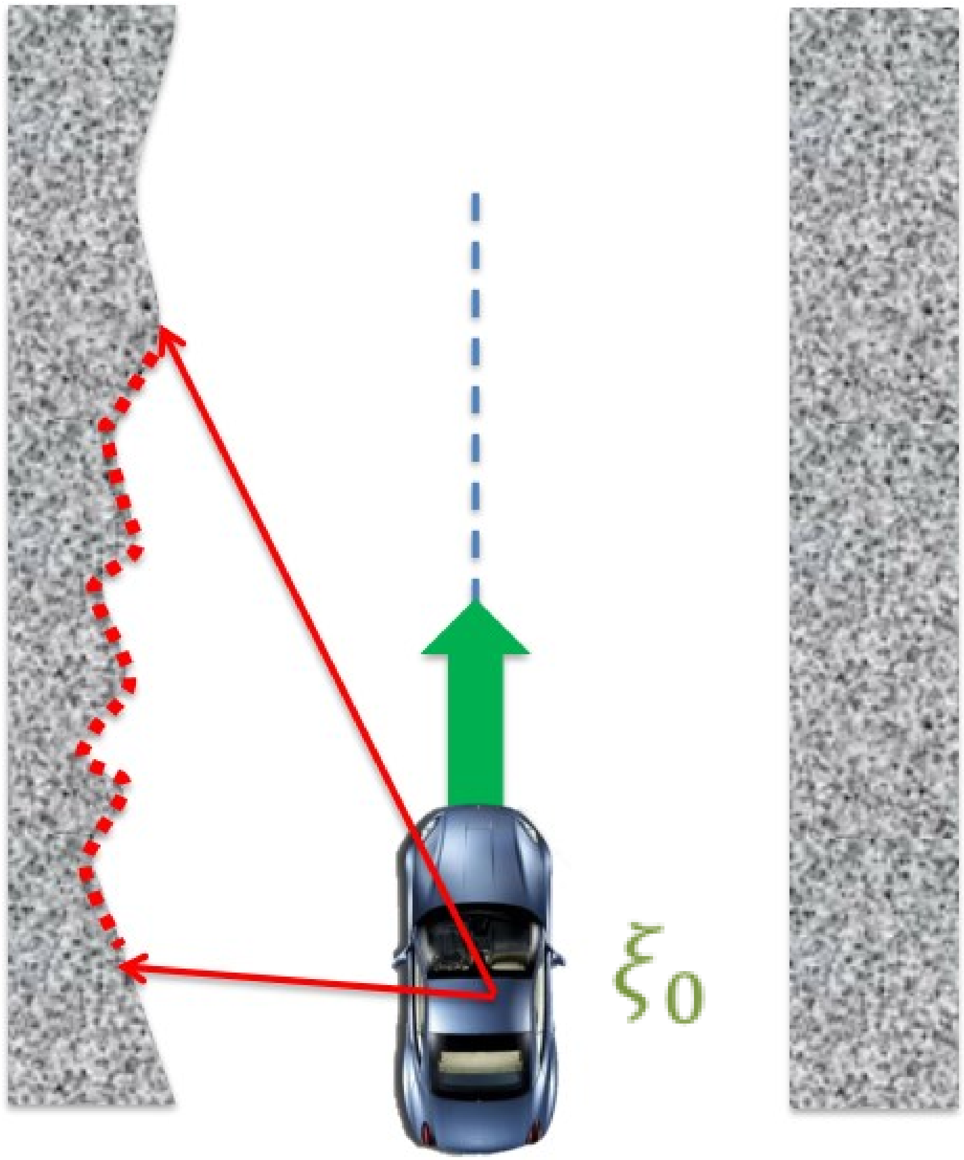
$$\boldsymbol{\xi} = (p_x, p_y, \psi)^{\mathrm{T}}$$



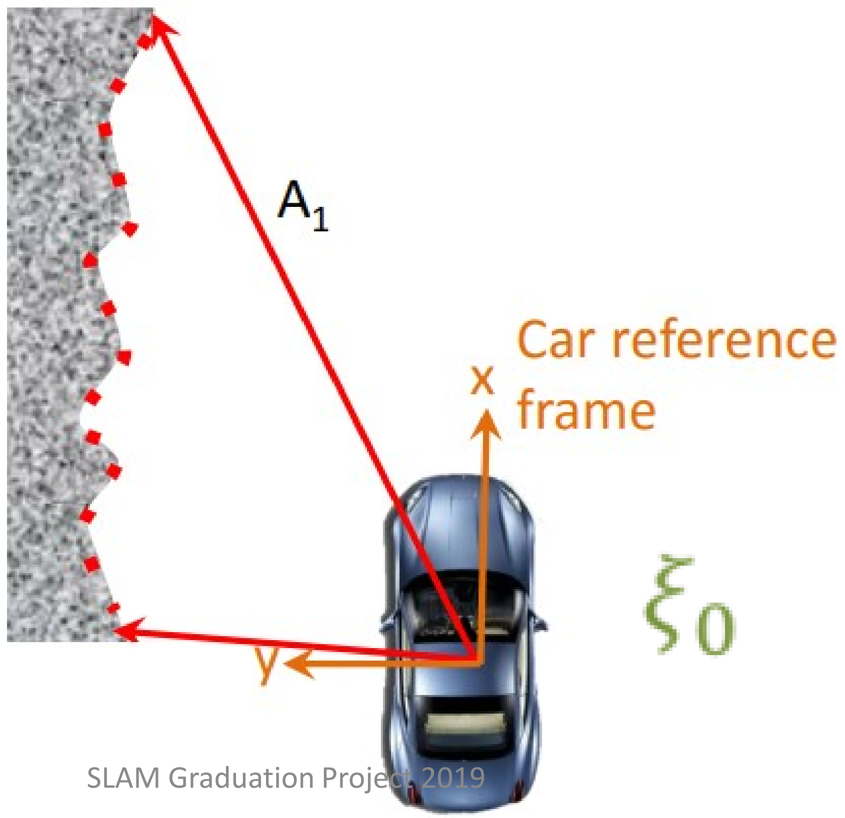
Slides adapted from slides of University of Pennsylvania (http://f1tenth.org)

Hector SLAM - Overview

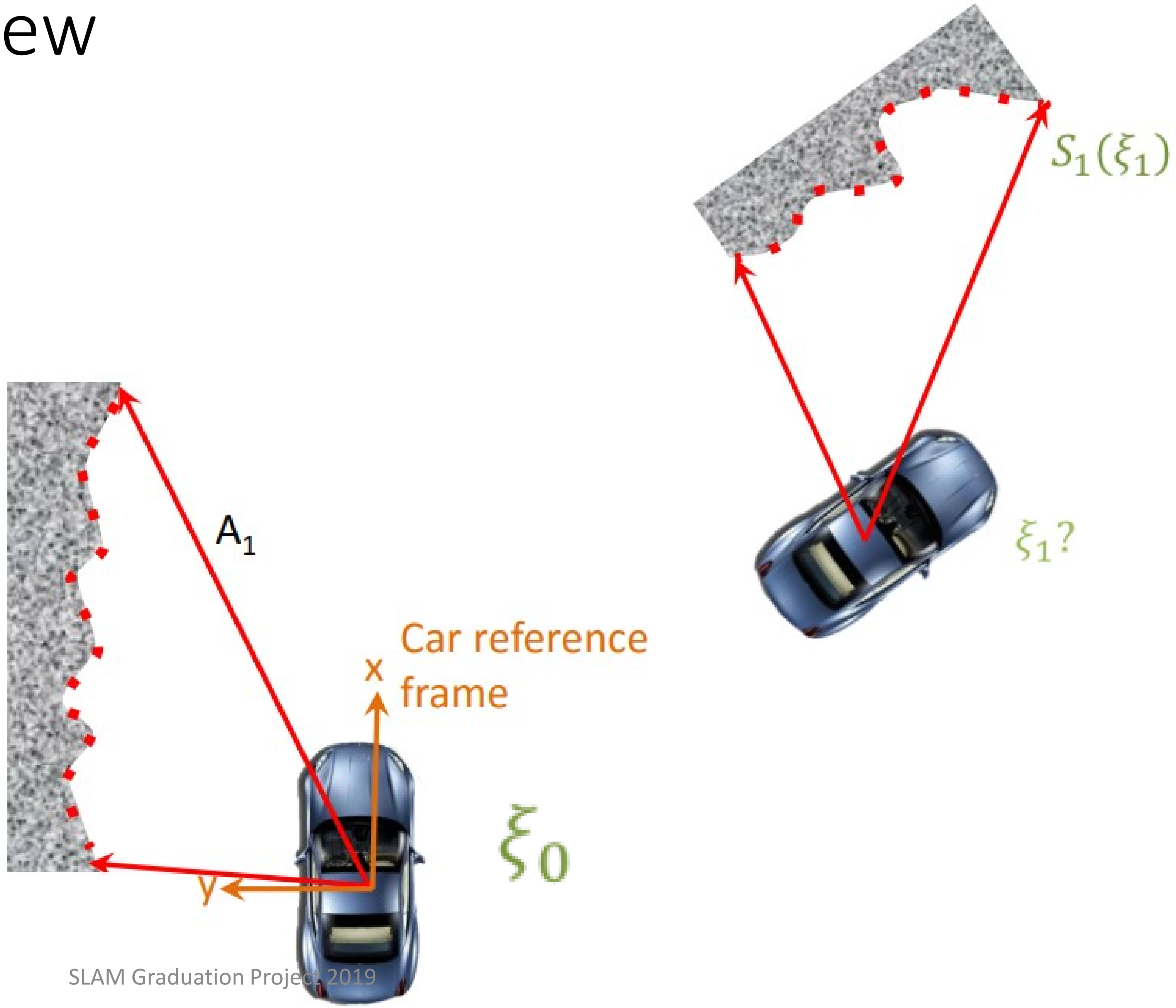
- First LIDAR scan



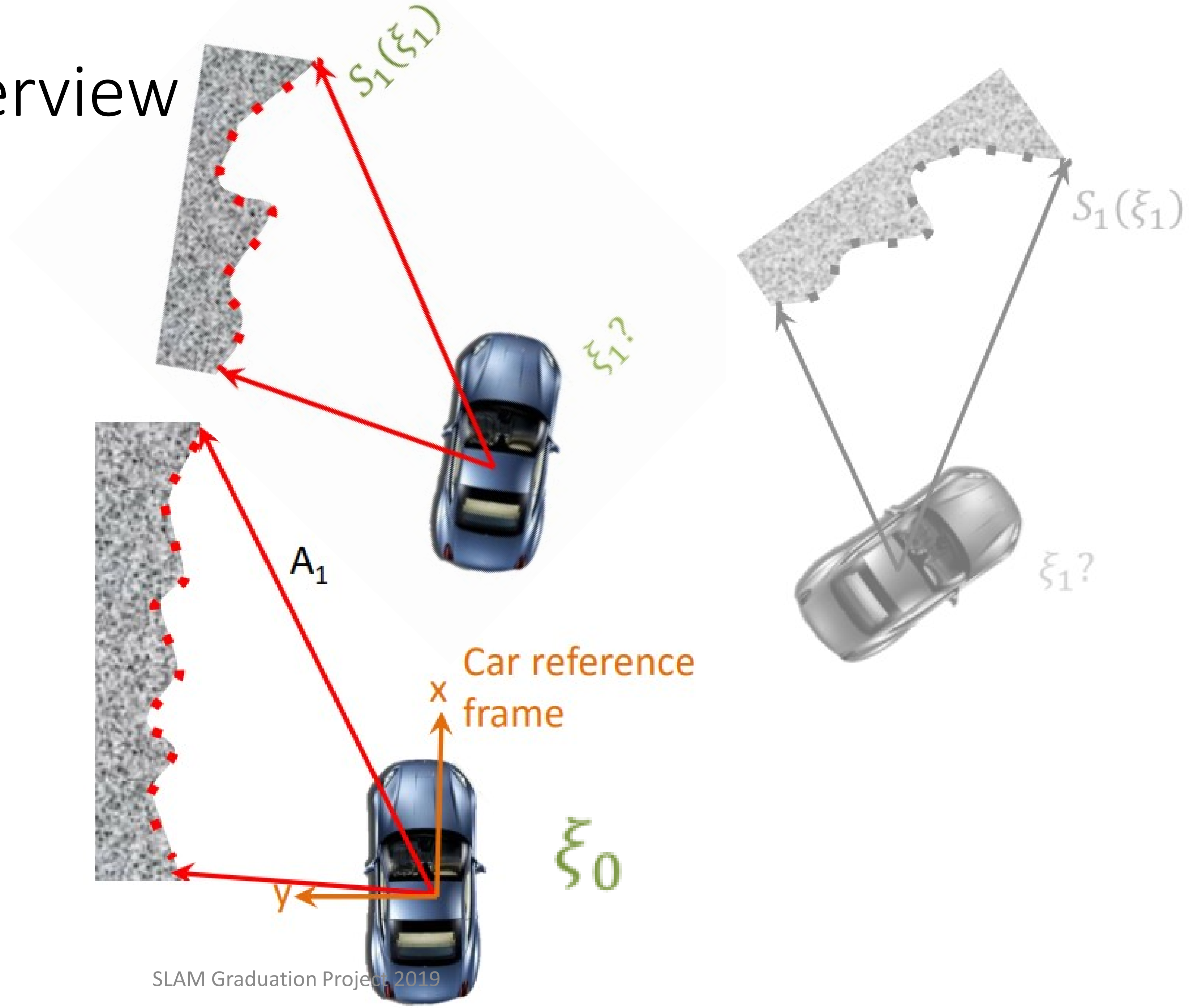
Hector SLAM - Overview



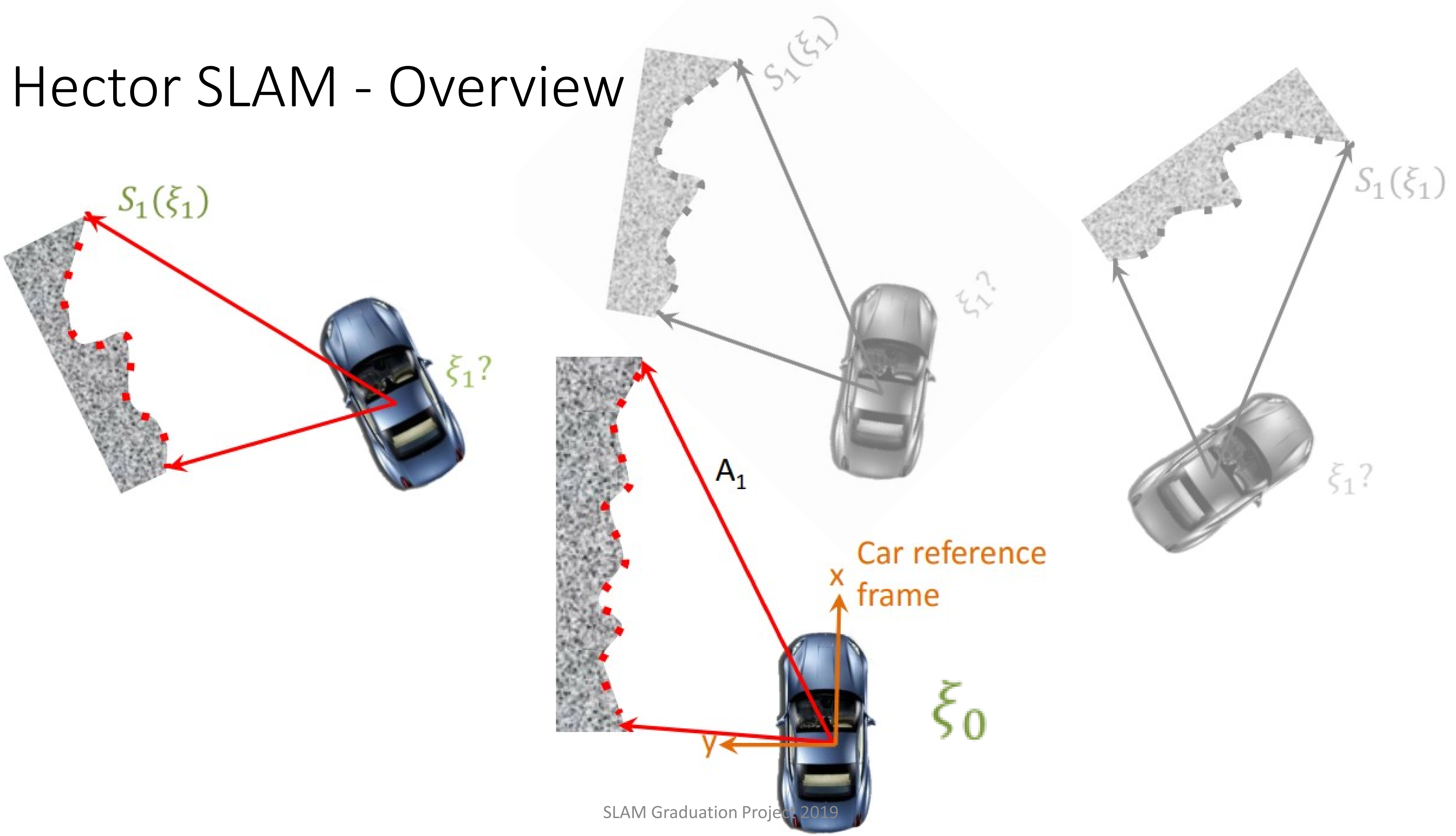
Hector SLAM - Overview



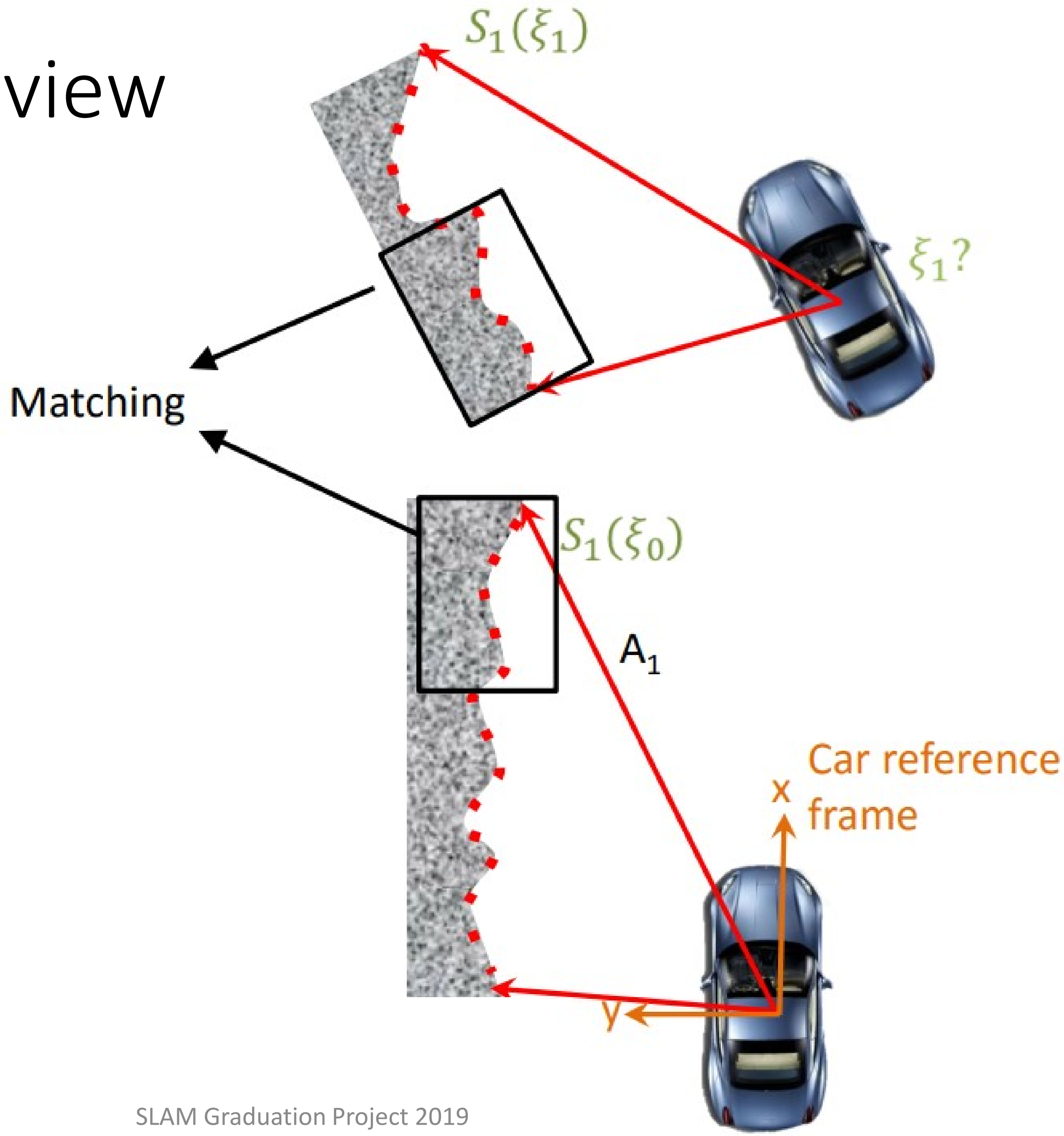
Hector SLAM - Overview



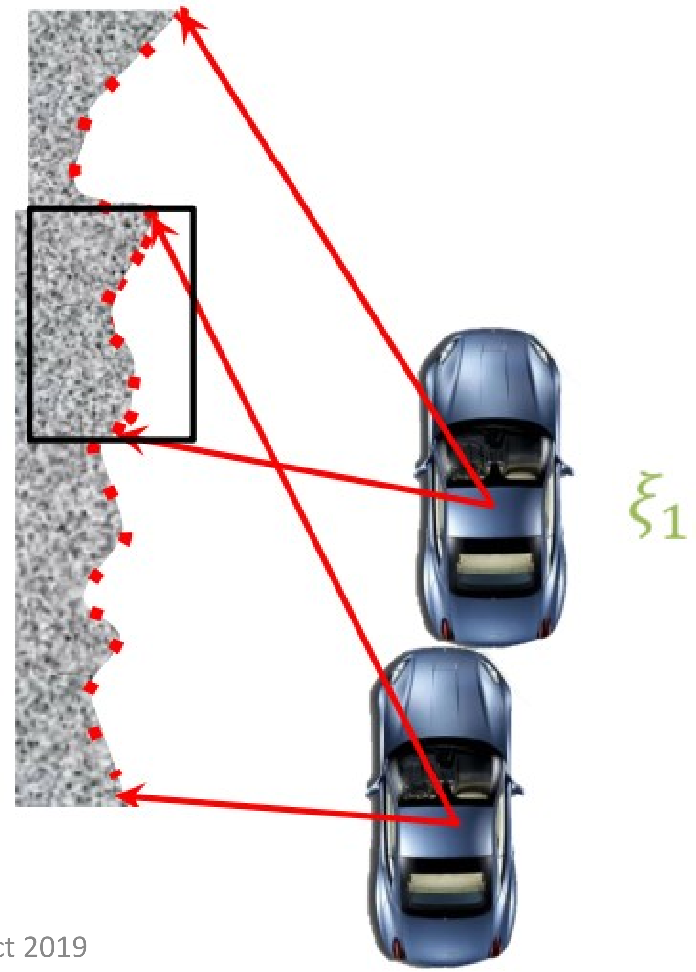
Hector SLAM - Overview



Hector SLAM - Overview



Hector SLAM - Overview



Hector SLAM - Overview

- This is an **optimization** problem

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

Hector SLAM - Overview

- This is an **optimization** problem
- Find ξ that minimizes the **square error** between the existing map and the new scan

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^{n} \boxed{\left[1 - M(\mathbf{S}_i(\xi))\right]^2}$$

Hector SLAM - Overview

- This is an **optimization** problem
- Find ξ that minimizes the **square error** between the existing map and the new scan
- Error is the **mismatch** between the existing map and the new scan

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - \underline{M(\mathbf{S}_i(\xi))}\right]^2$$

Hector SLAM - Overview

- After expansion and differentiation, we reach the **Gauss-Newton equation**, which is solved iteratively

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

$$\downarrow$$

$$\Delta\boldsymbol{\xi} = \mathbf{H}^{-1} \sum_{i=1}^{n} \left[\nabla M(\mathbf{S}_i(\boldsymbol{\xi})) \frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right]^T \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]$$

$$\mathbf{H} = \left[\nabla M(\mathbf{S}_i(\boldsymbol{\xi})) \frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right]^T \left[\nabla M(\mathbf{S}_i(\boldsymbol{\xi})) \frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right]$$

Hector SLAM - Overview

- After each iteration, we get a new **$\Delta\xi$** towards the minimum error

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\xi))]^2$$

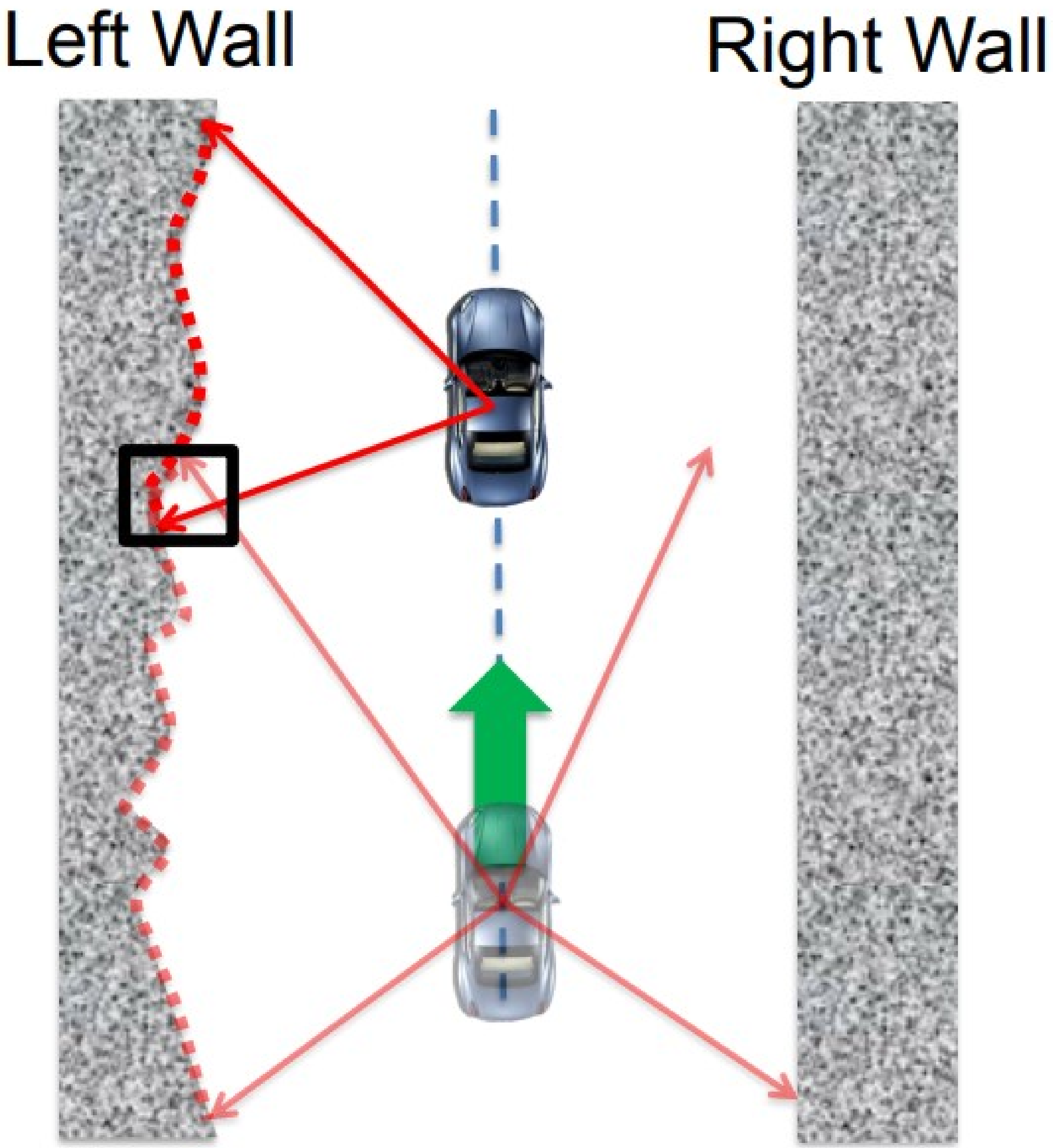
$$\downarrow$$

$$\Delta\xi = \mathbf{H}^{-1} \sum_{i=1}^{n} \left[\nabla M(\mathbf{S}_i(\xi)) \frac{\partial \mathbf{S}_i(\xi)}{\partial \xi} \right]^T [1 - M(\mathbf{S}_i(\xi))]$$

$$\mathbf{H} = \left[\nabla M(\mathbf{S}_i(\xi)) \frac{\partial \mathbf{S}_i(\xi)}{\partial \xi} \right]^T \left[\nabla M(\mathbf{S}_i(\xi)) \frac{\partial \mathbf{S}_i(\xi)}{\partial \xi} \right]$$

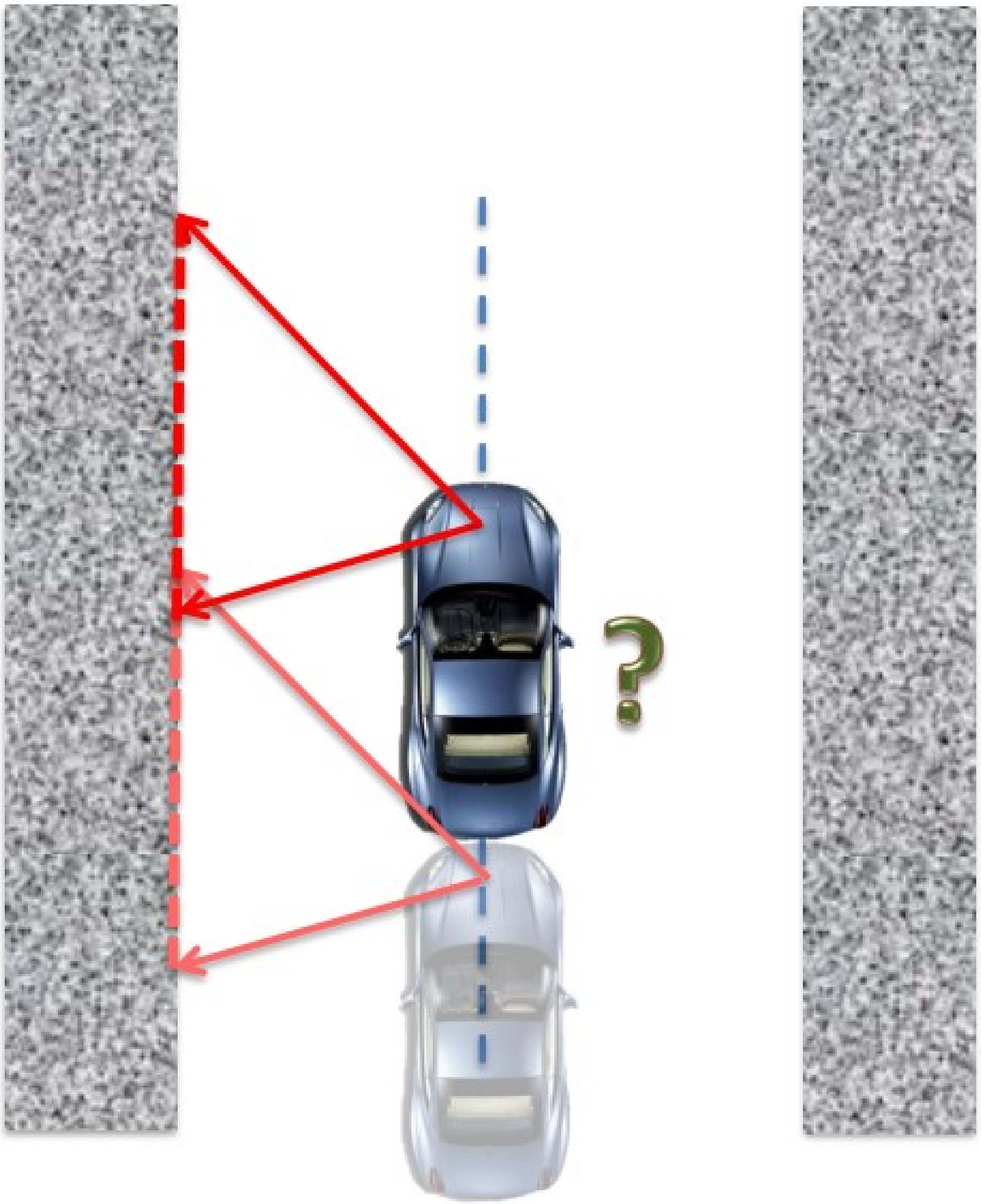
Hector SLAM - Requirements

- **Fast scan rate** with respect to the robot's speed (and computation)
- To find an overlap between the new scan and the existing map
- Otherwise, the algorithm may fail to localize the robot correctly
- This is an advantage of the high scan rate of LIDARs



Hector SLAM - Requirements

- Non-smooth, heterogeneous surfaces
- Otherwise, new scan looks exactly like existing map
- With no other source of odometry, algorithm thinks the robot didn't move
- Will be shown in the coming demo videos

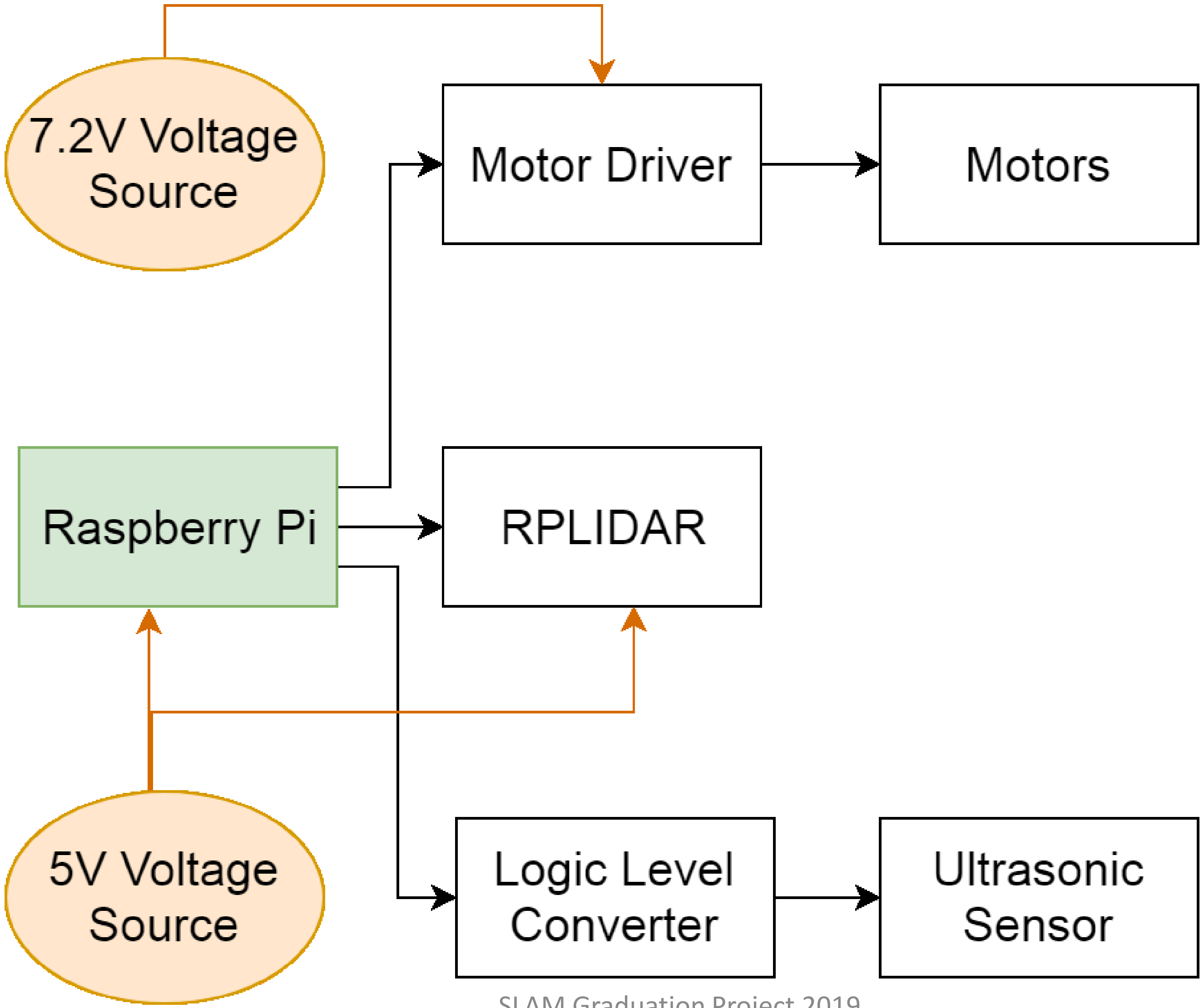


Outline

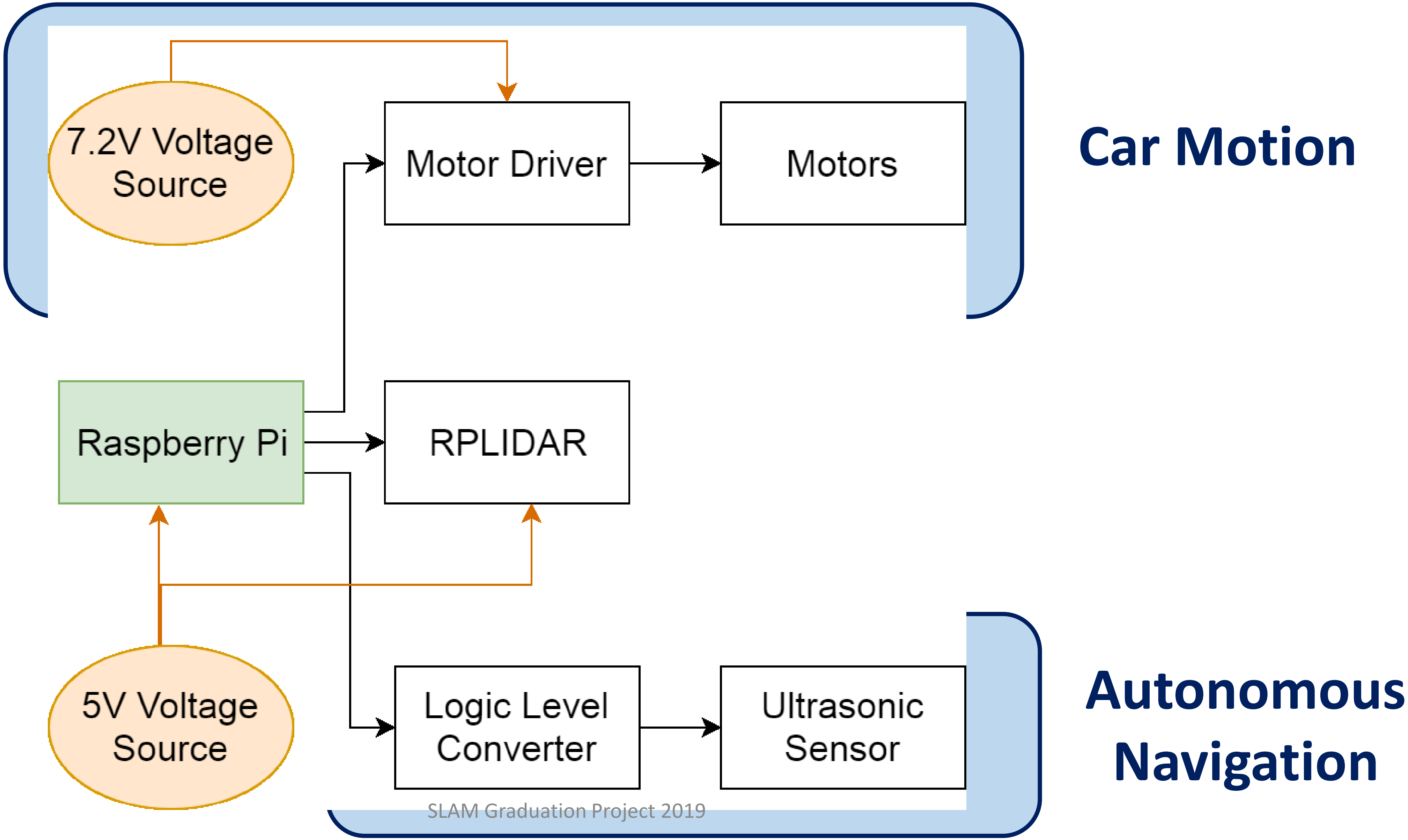
- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

Robot

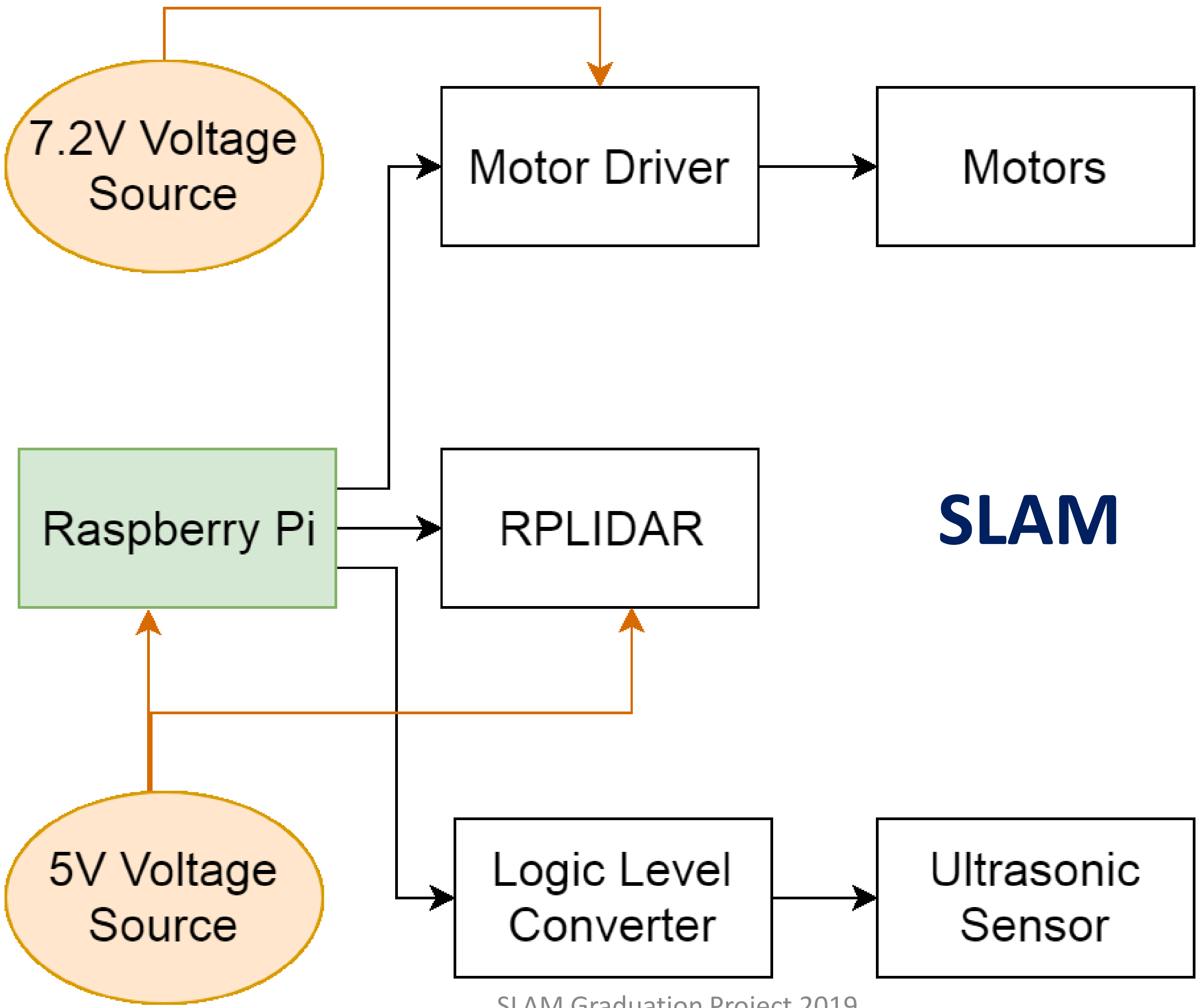
Robot - Hardware Block Diagram



Robot - Hardware Block Diagram

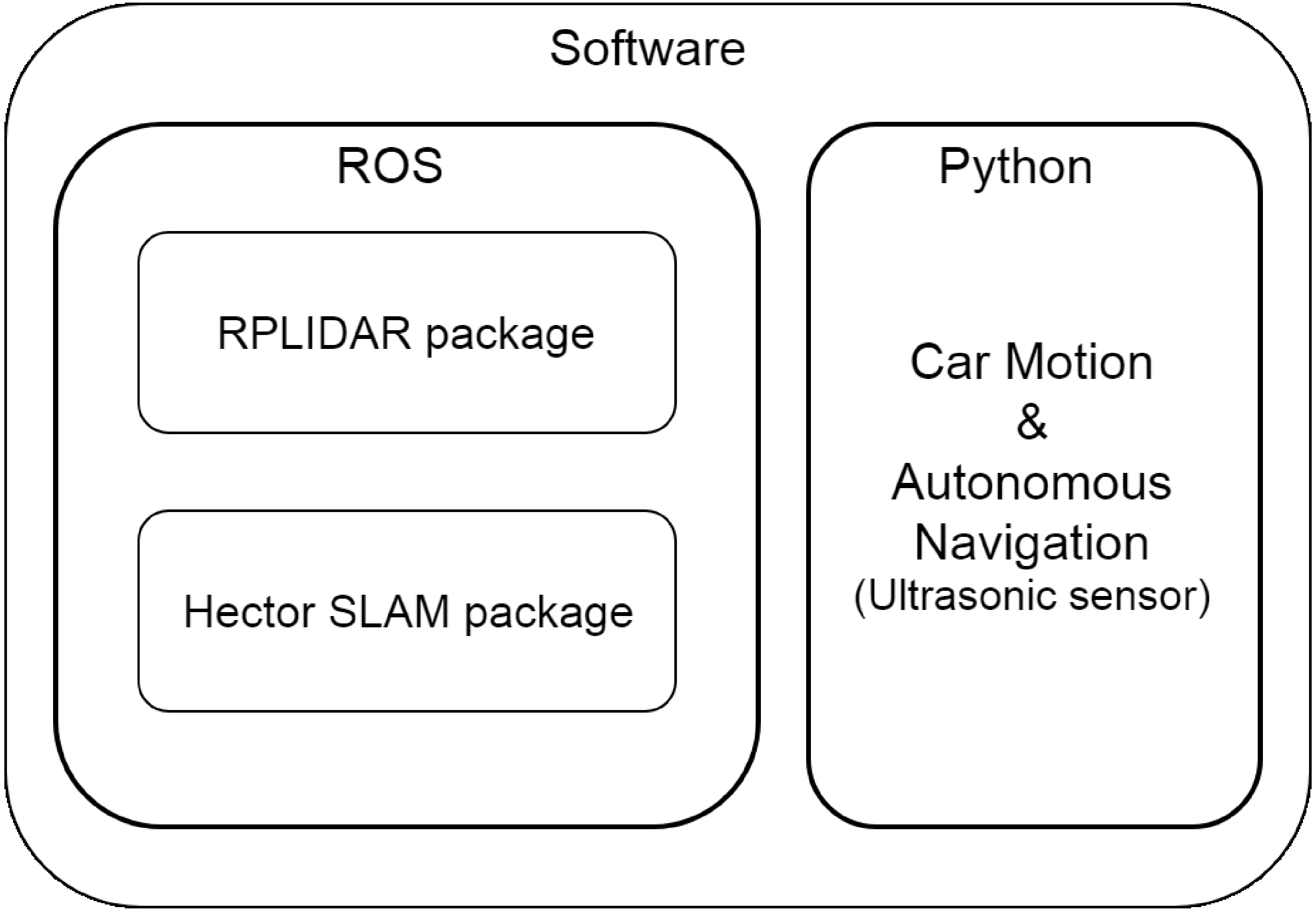


Robot - Hardware Block Diagram



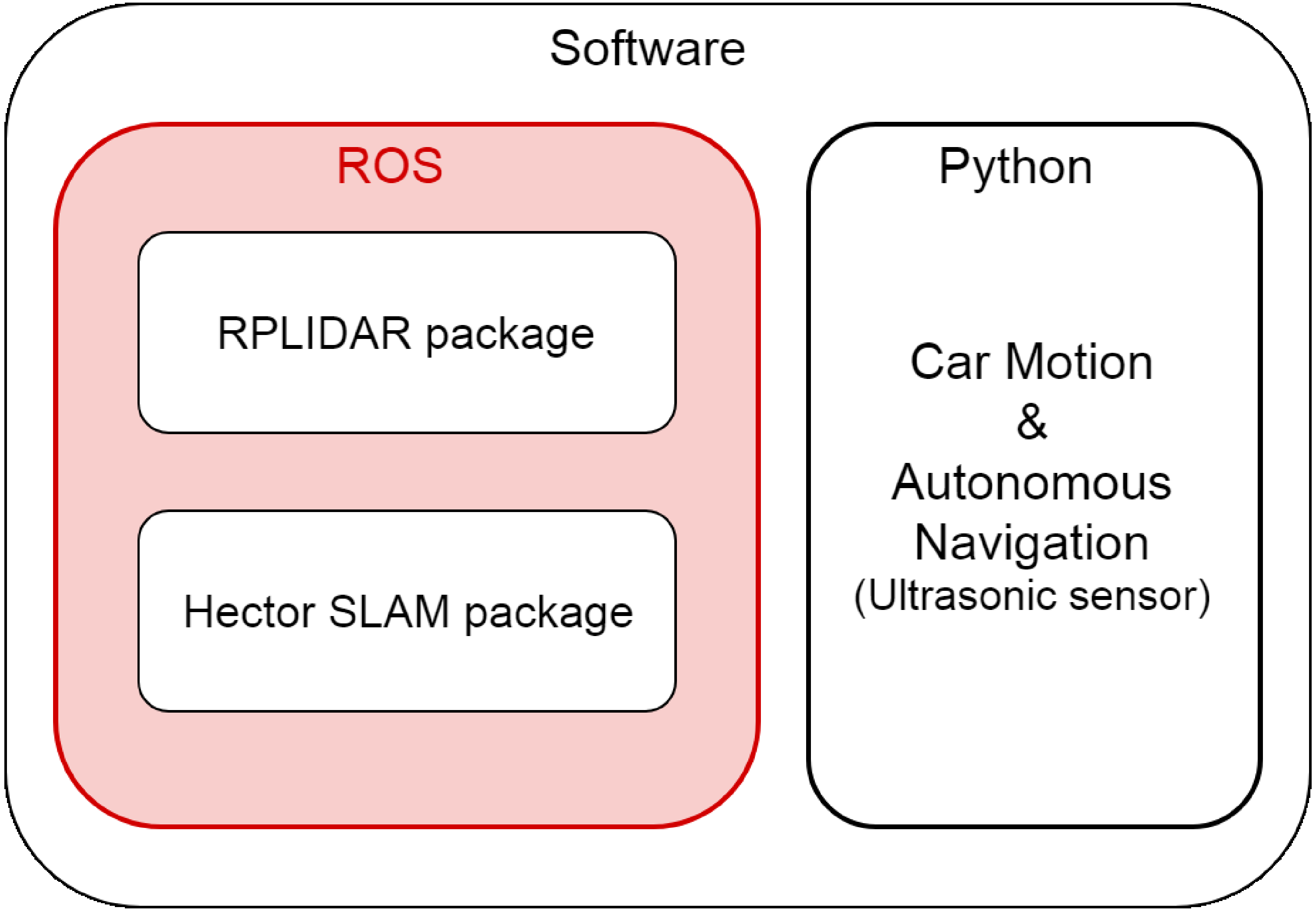
Robot - Software

- Software runs on Raspberry Pi 3
- Two main components
  - ROS
  - Python code



Robot - Software

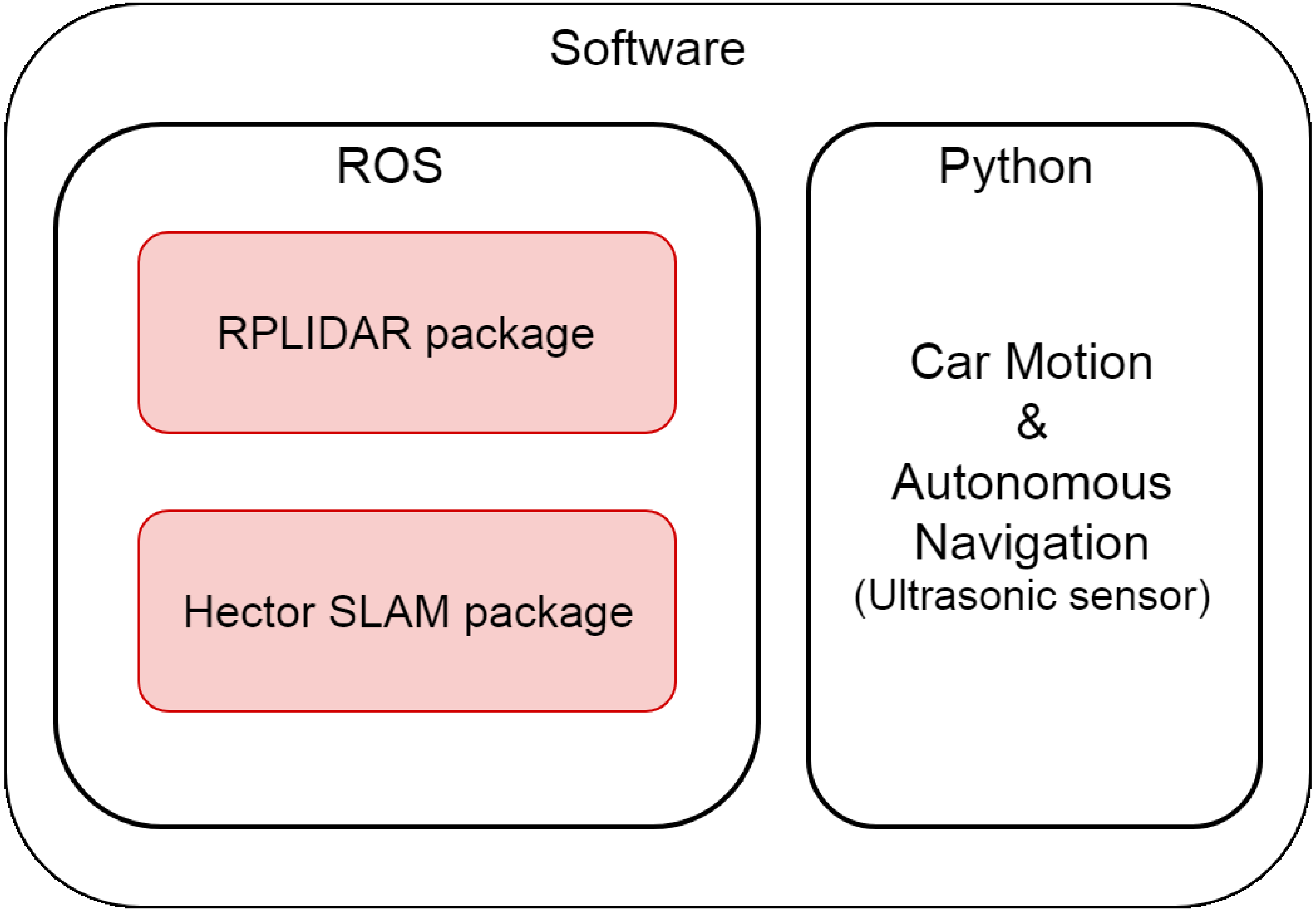
- Robot Operating System
- Open-source, runs on UNIX
- Includes
  - Implementation of commonly-used functionality
  - Message-passing between processes
  - Package management
  - Etc.



Robot - Software

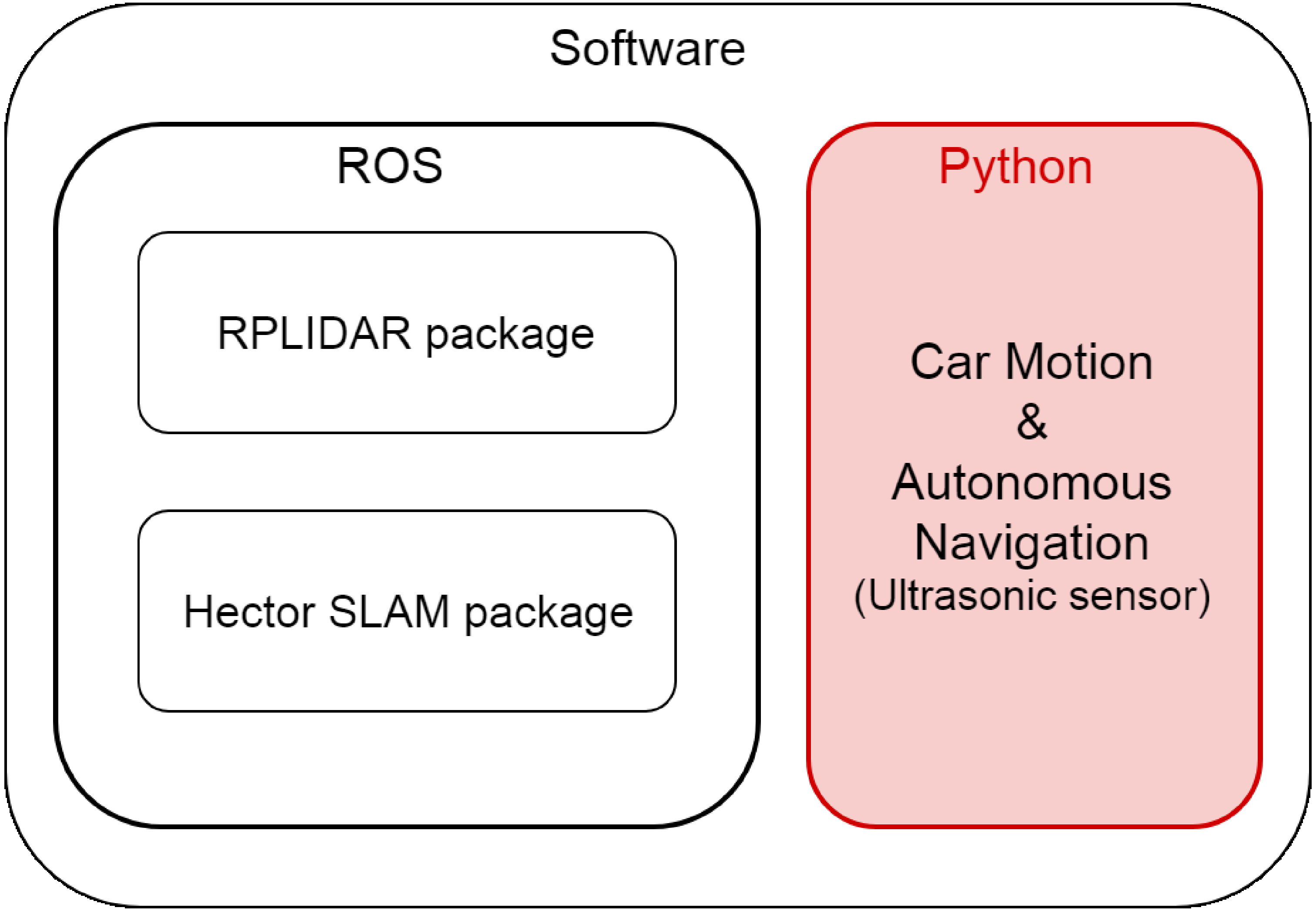
- We used 2 packages from ROS

1. RPLIDAR
   - Operates the LIDAR
   - Publishes LIDAR readings
2. Hector SLAM
   - Subscribes to LIDAR readings
   - Builds the map
   - Estimates the robot's pose and trajectory



Robot - Software

- We wrote separate python code to:
  1. Control the motor and move the car
  2. Drive the ultrasonic sensor for autonomous navigation



Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- **Autonomous Navigation**
- Results

Autonomous Navigation

- Used Ultrasonic sensor
- Simple method: turn on detecting an obstacle
- Limited access to LIDAR
- **Future work:** use LIDAR instead of ultrasonic for navigation
  - LIDAR is more accurate
  - 360° ranging
  - Can implement *Active SLAM*

    Active SLAM: deciding how should the robot moves in order to explore unvisited parts of the map
- **Future work:** improve mechanical build of the robot

Outline

- Problem
- SLAM
- Hector SLAM Algorithm
- Robot details
- Autonomous Navigation
- Results

Demo videos

Thank You

Contact us:

Kirollos Henry	+201224959553	kirollostadros@gmail.com
David Nader	+201226269655	david.nader.g@gmail.com