

F1/10

Autonomous Racing

Perception
Localization

Madhur Behl

CS 4501/SYS 4582
Spring 2019
Rice Hall 120

Localization and Mapping

Where am I ?

Scene Understanding

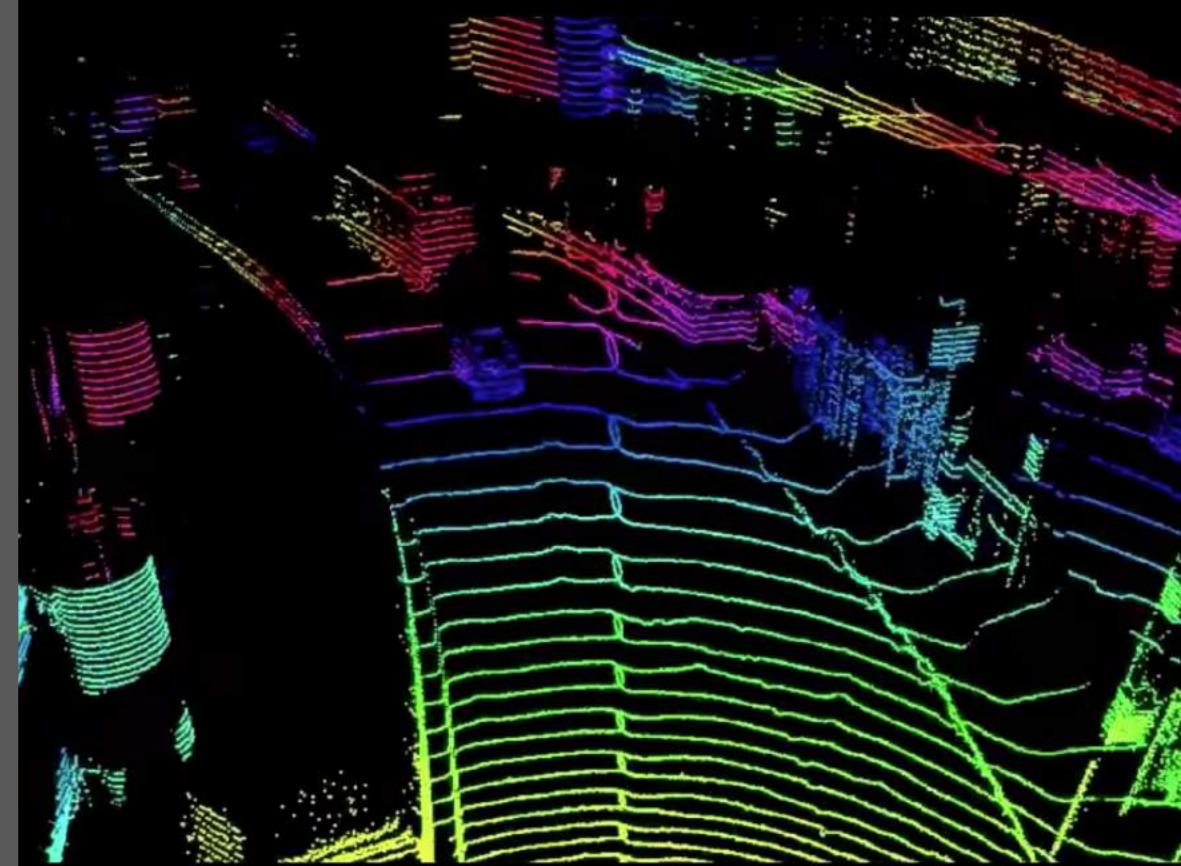
Where/who/what/why of everyone/everything else ?

Trajectory Planning and Control

Where should I go next ?
How do I steer and accelerate ?

Human Interaction

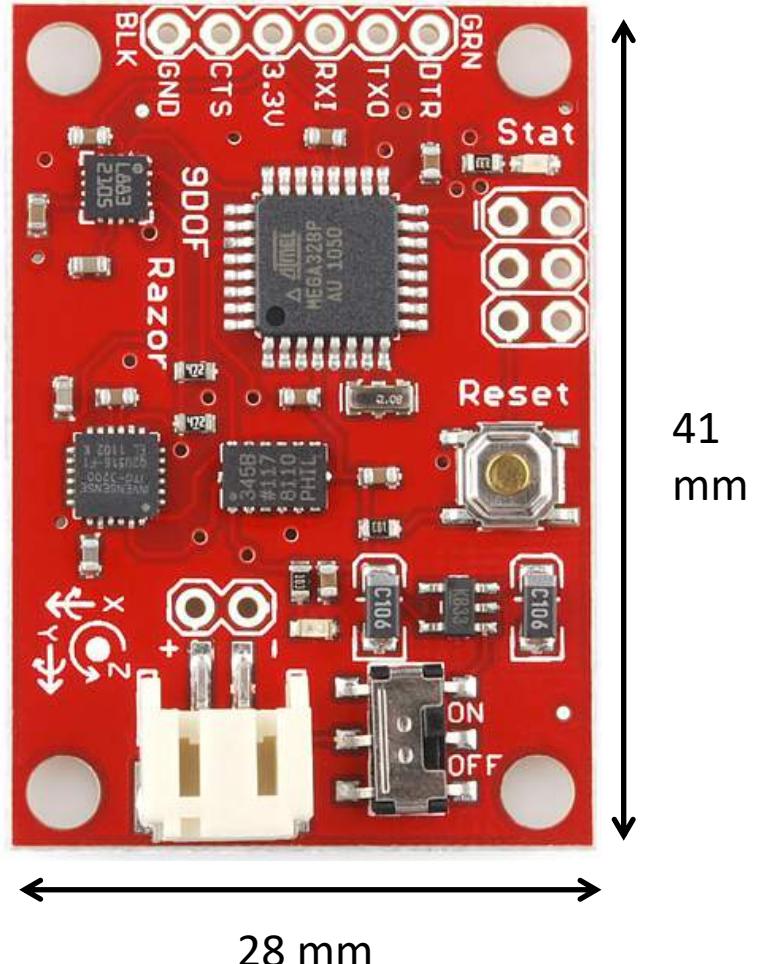
How do I convey my intent to the passenger and everyone else ?



Perception

Sensors: specifications, placement and data

Inertial Measurement Unit



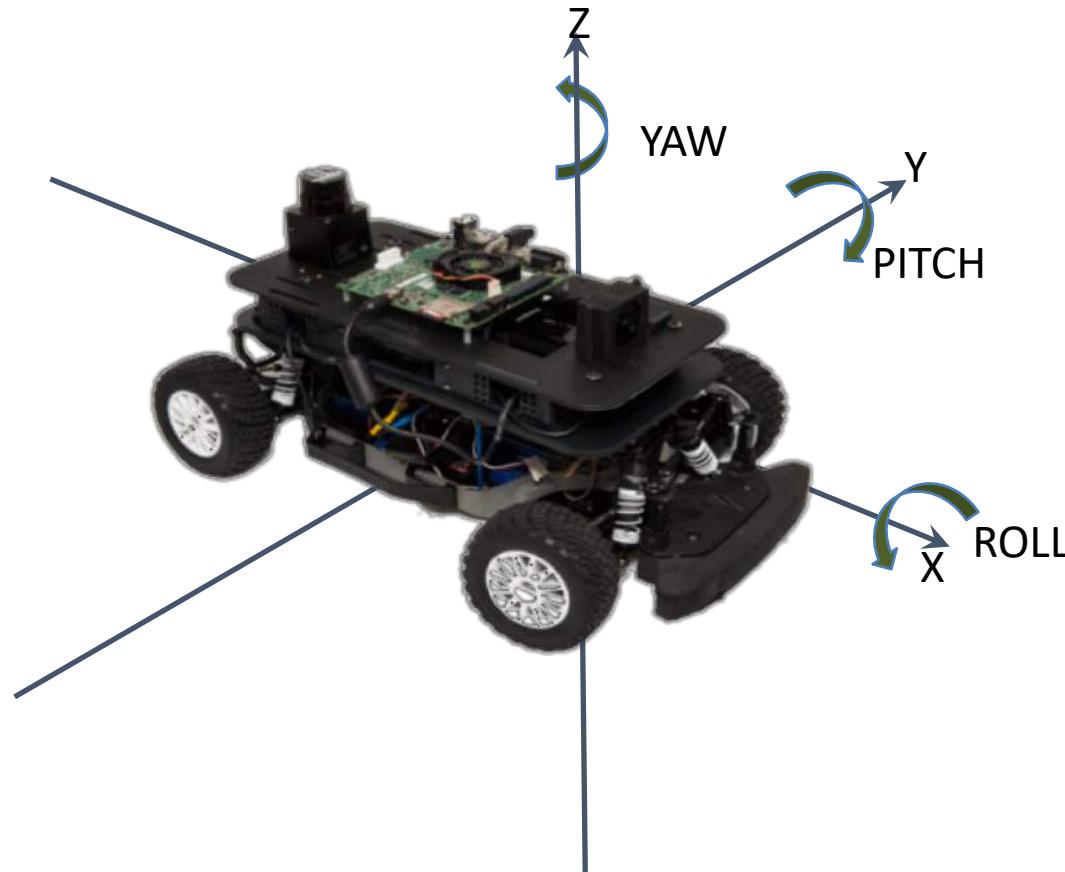
Razor IMU from Sparkfun

An Inertial Measurement Unit (IMU) provides acceleration, velocity and attitude measurements.

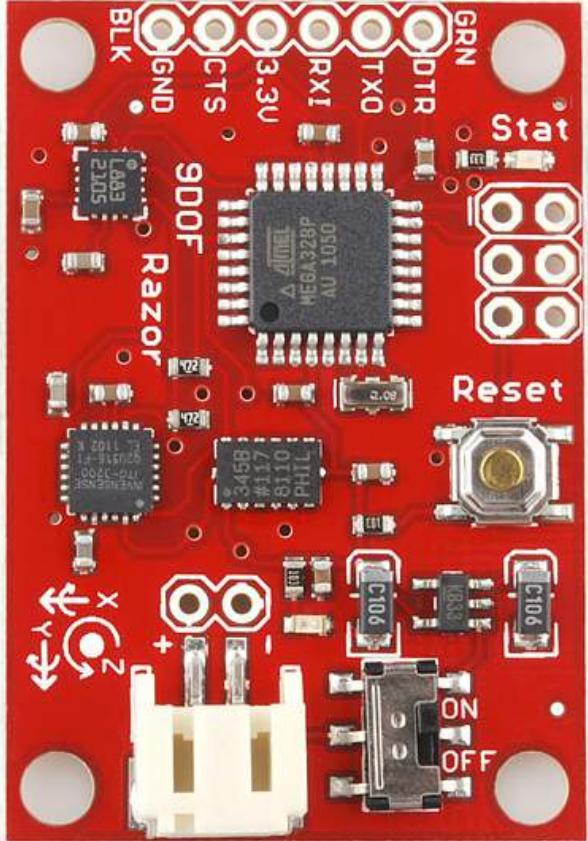
$$\dot{x} = v$$

$$\dot{v} = a$$

Inertial Measurement Unit: Attitude



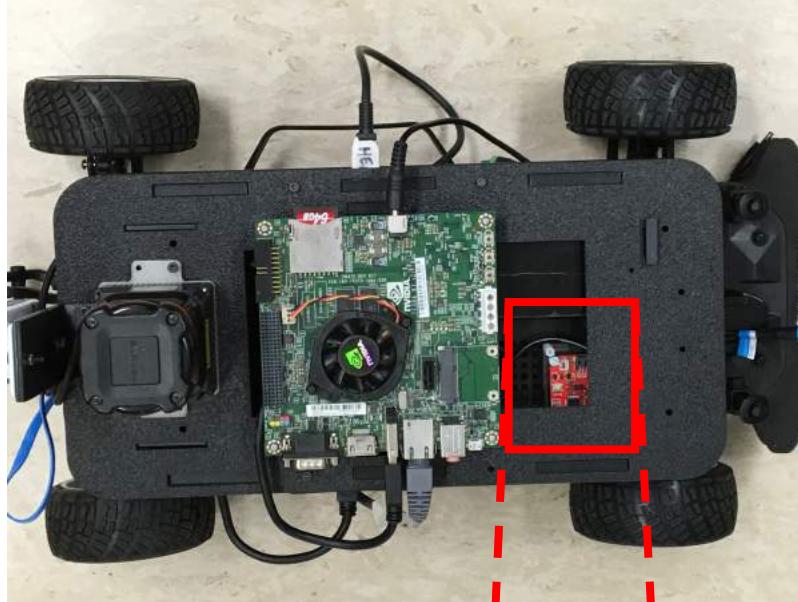
Inertial Measurement Unit



Razor IMU from Sparkfun

- Three gyroscopes measure the roll, pitch and yaw of the car.
- Three accelerometers measure the linear and rotational acceleration of the car.
- Three magnetometers

Inertial Measurement Unit: Placement



IMU: ROS message

```
# File: sensor_msgs/Imu.msg
```

```
# This is a message to hold data from an IMU (Inertial Measurement Unit)
# Accelerations should be in m/s^2 (not in g's), and rotational velocity should be in rad
/sec
```

```
Header header
```

```
geometry_msgs/Quaternion orientation
```

```
float64[9] orientation_covariance # Row major about x, y, z axes
```

```
geometry_msgs/Vector3 angular_velocity
```

```
float64[9] angular_velocity_covariance # Row major about x, y, z axes
```

```
geometry_msgs/Vector3 linear_acceleration
```

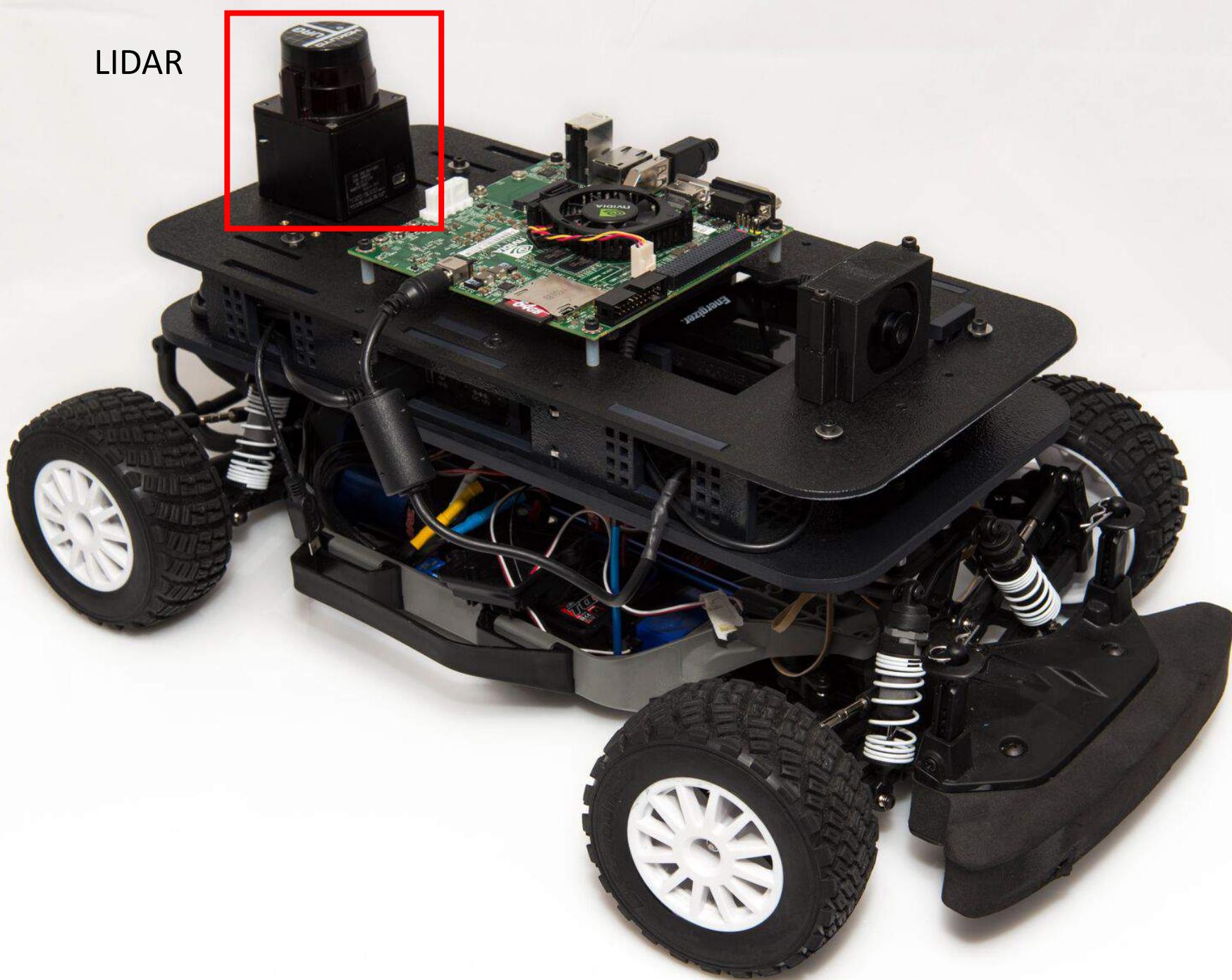
```
float64[9] linear_acceleration_covariance # Row major x, y z
```

LIDAR

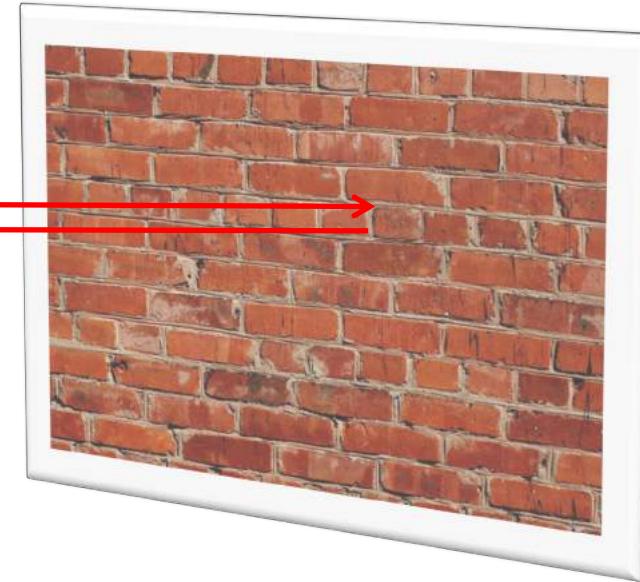
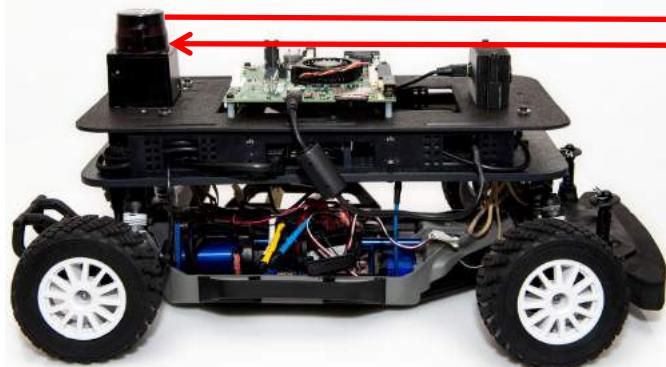


Depth information (how far are objects and obstacles around me)

LIDAR



Distance to obstacle = (speed of light * time traveled)/2





Hokuyo UST-10LX

270 degrees horizontal field
of view

Step 1079

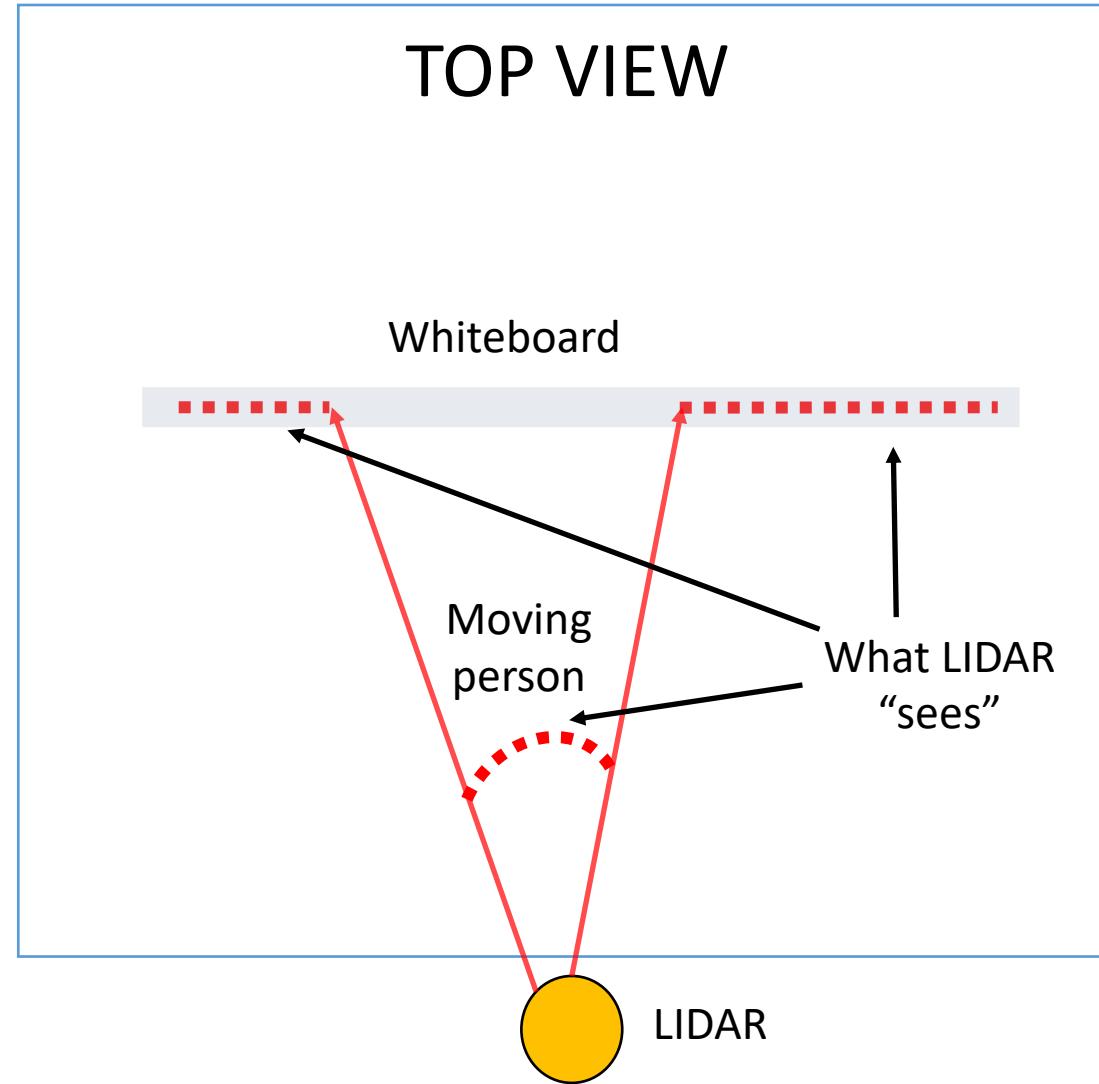
Range
10m

Step k+1
Step k

Angular resolution 0.25 degrees

Step 1
Step 0

LIDAR: a simple scan sequence





Time

ROS Time: 1456773883.70

ROS Elapsed: 362.13

Wall Time: 1456773883.74

Wall Elapsed: 362.13

Experimental

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

30 fp

LIDAR: ROS message structure

File: sensor_msgs/LaserScan.msg

```
Header header      # timestamp in the header is the acquisition time of  
                   # the first ray in the scan.  
  
float32 angle_min    # start angle of the scan [rad]  
float32 angle_max    # end angle of the scan [rad]  
float32 angle_increment # angular distance between measurements [rad]  
  
float32 time_increment # time between measurements [seconds]  
float32 scan_time      # time between scans [seconds]  
  
float32 range_min      # minimum range value [m]  
float32 range_max      # maximum range value [m]  
  
float32[] ranges      # range data [m] (Note: values < range_min or > range_max  
                      # should be discarded)  
float32[] intensities   # intensity data [device-specific units].
```

paril@paril: ~

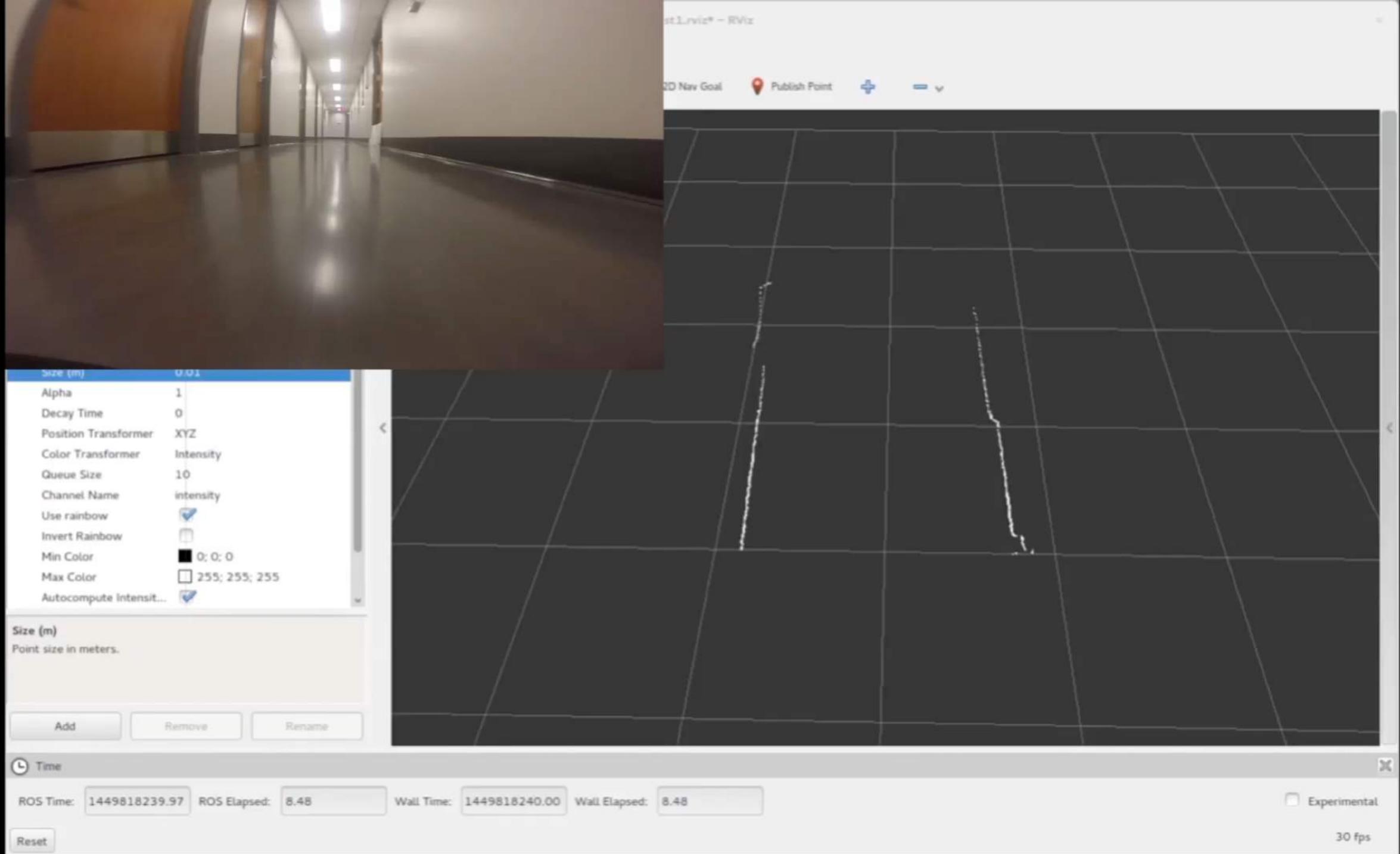
paril@paril: ~ 80x24

```
inf,  
inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf,  
inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf,  
inf, inf, inf, inf, inf, 0.2980000078678131, 0.2770000100135803, 0.268000006675  
7202, 0.27300000190734863, 0.27300000190734863, 0.26499998569488525, 0.264999985  
69488525, 0.2639999985694885, 0.2639999985694885, 0.2639999985694885, 0.26800000  
66757202, 0.27399998903274536, 0.28299999237060547, 0.2840000092983246, 0.291999  
9957084656, 0.2919999957084656, 0.296999990940094, 0.30000001192092896, 0.303000  
00309944153, 0.30300000309944153, 0.30300000309944153, 0.3009999990463257, 0.300  
00001192092896, 0.29899999499320984, 0.29899999499320984, 0.29899999499320984, 0  
.29899999499320984, 0.29899999499320984, 0.30000001192092896, 0.3000000119209289  
6, 0.30000001192092896, 0.30000001192092896, 0.29899999499320984, 0.298000007867  
8131, 0.289000004529953, 0.289000004529953, 0.289000004529953, 0.289000004529953  
, 0.293999997615814, 0.293999997615814, 0.29100000858306885, 0.291000008583068  
85, 0.29100000858306885, 0.29100000858306885, 0.2800000011920929, 0.277000010013  
5803, 0.2770000100135803, 0.293999997615814, 0.3179999887943268, 0.326000005006  
79016, 0.3319999873638153, 0.3319999873638153, 0.32600000500679016, 0.3260000050
```

Array ranges[1080]: ranges[i] is distance measurement of (i+1)st step.

Measurements beyond the min and max range (like inf) *should be discarded*.

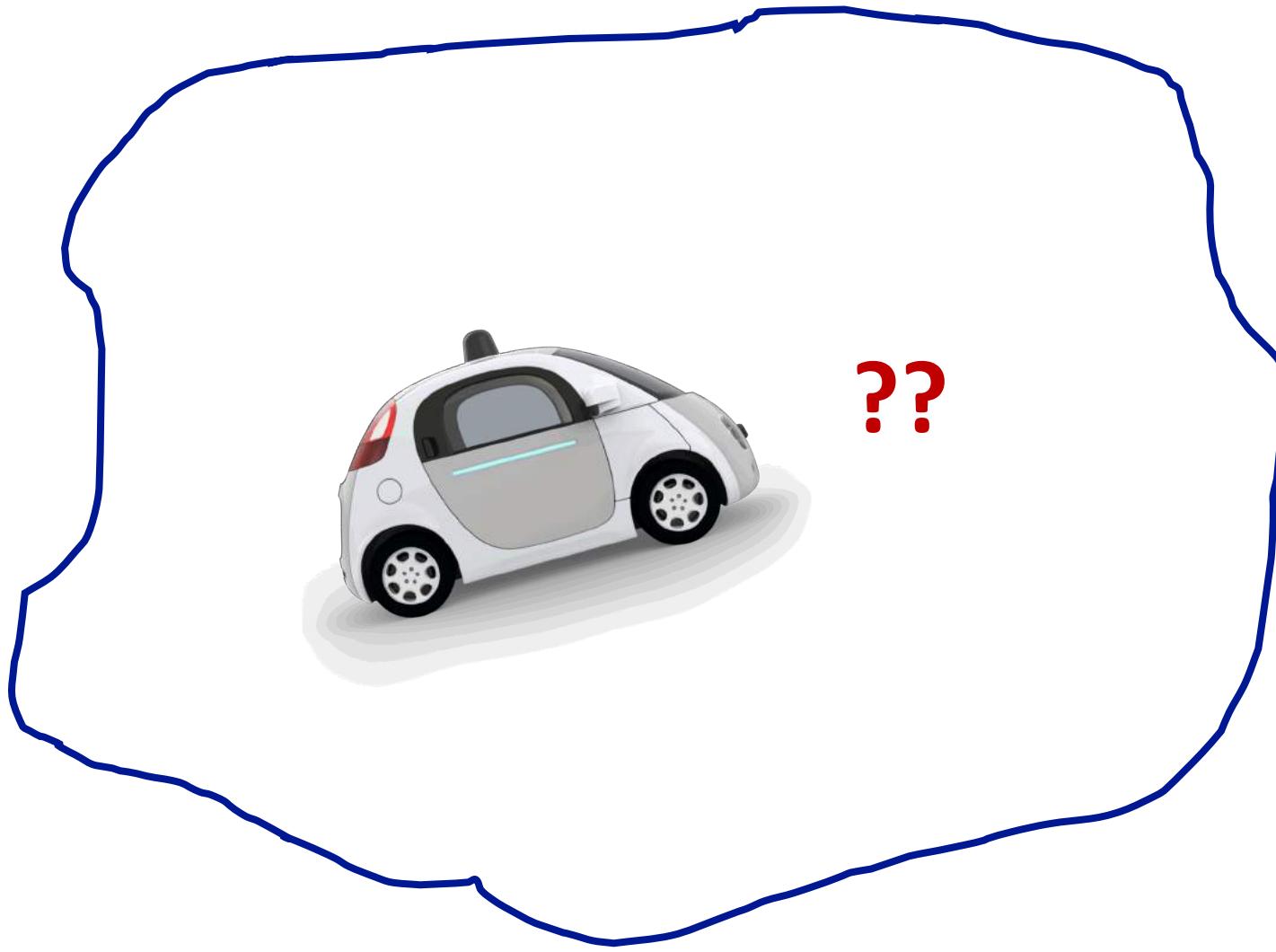
```
intensities: []  
---
```



Localization

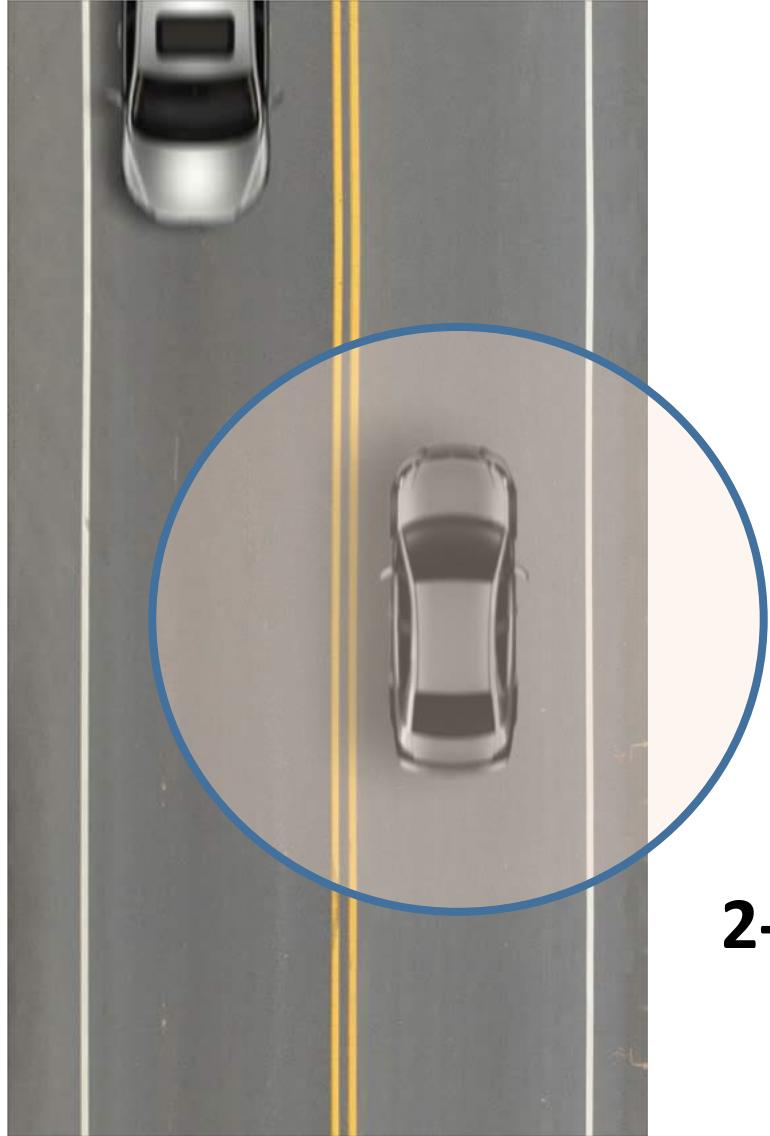
Know (where is) thyself:

Localization

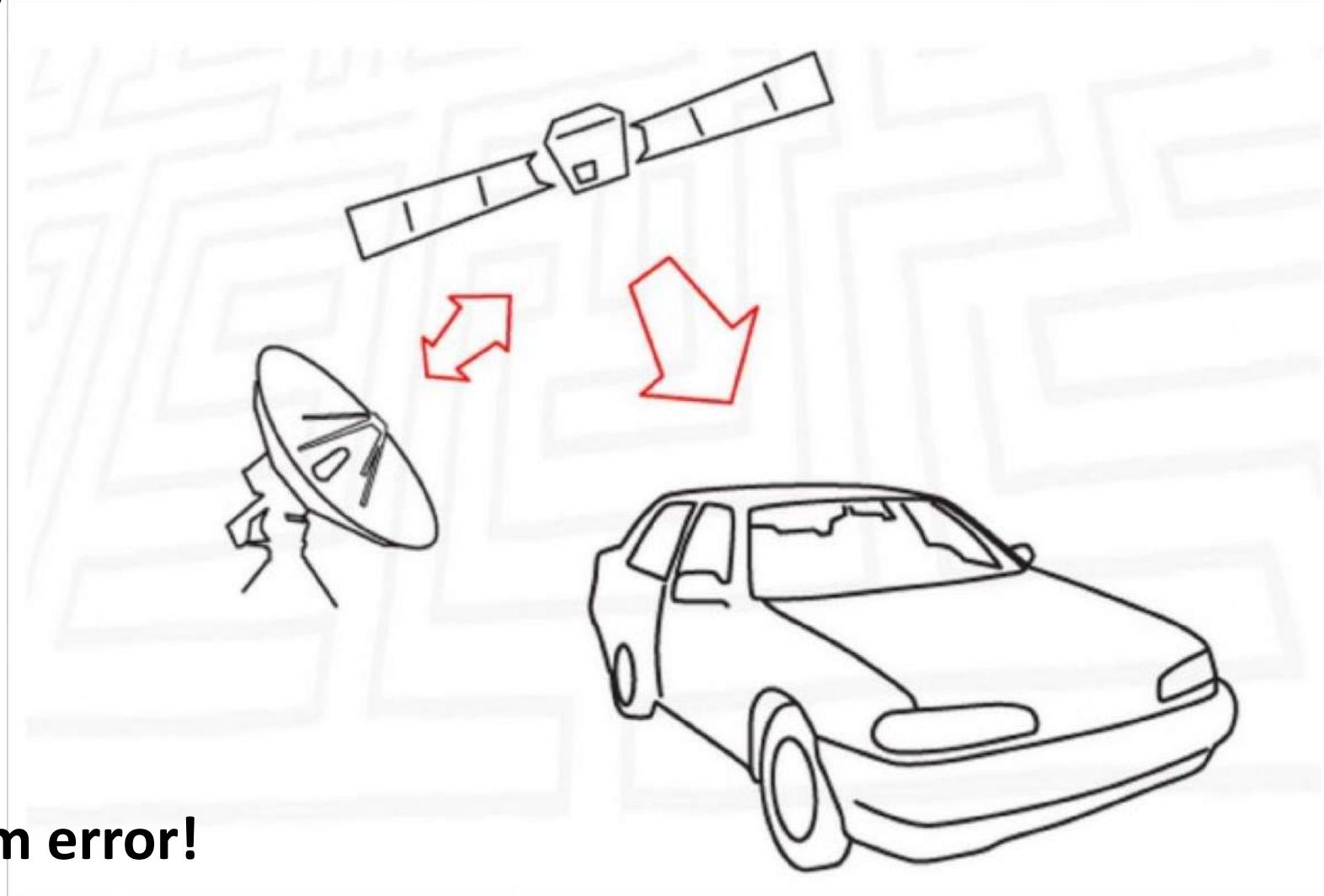




Global positioning system



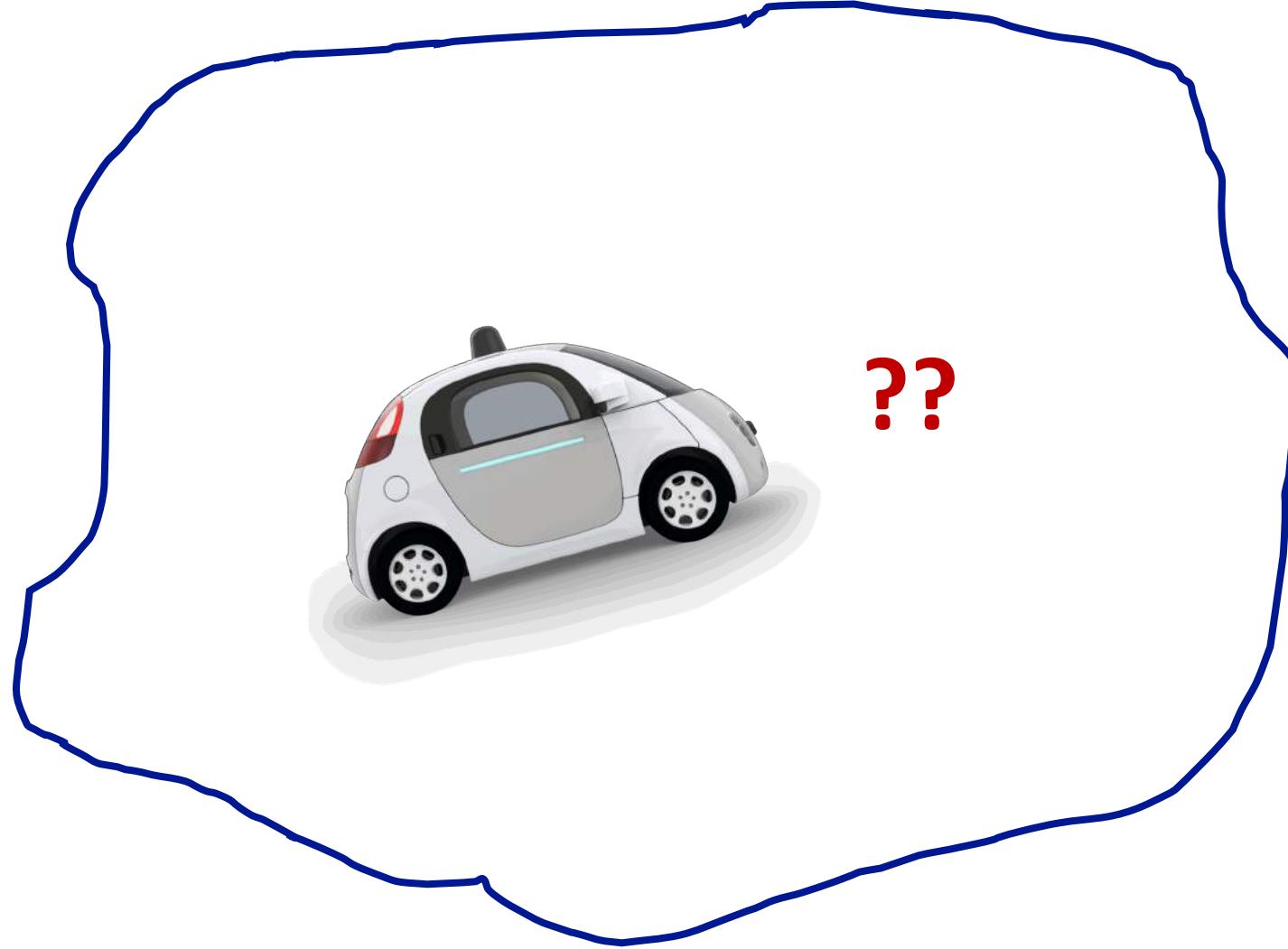
2-10m error!

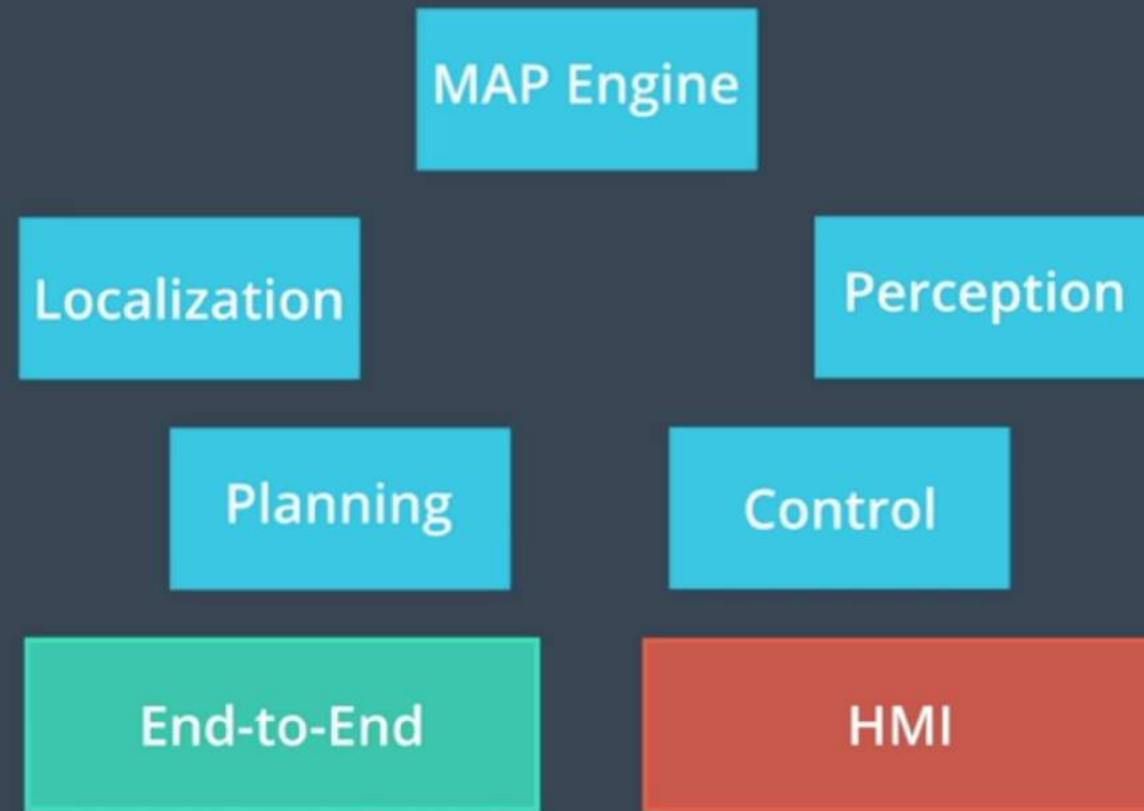


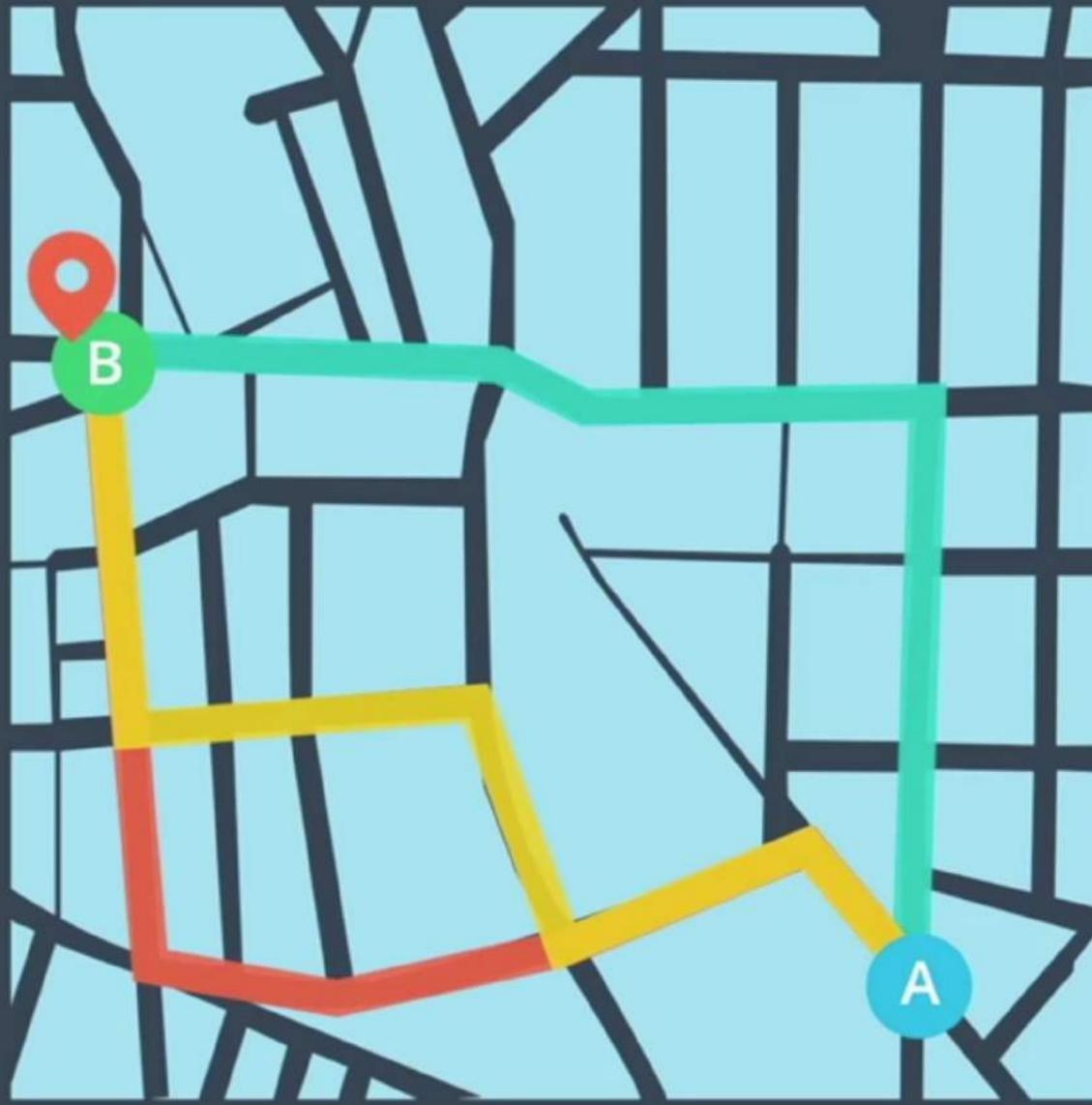
Need 2-10cm error

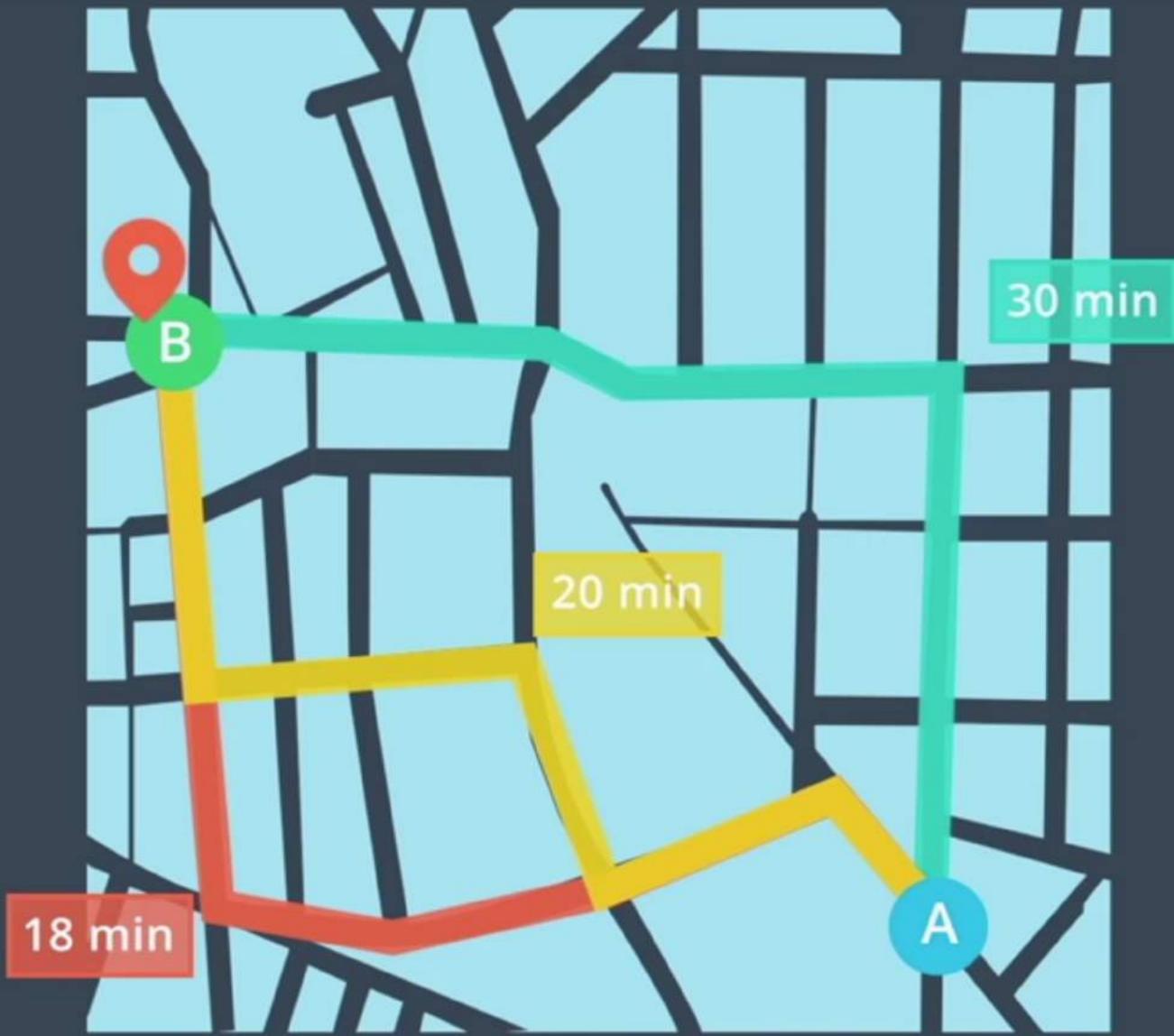


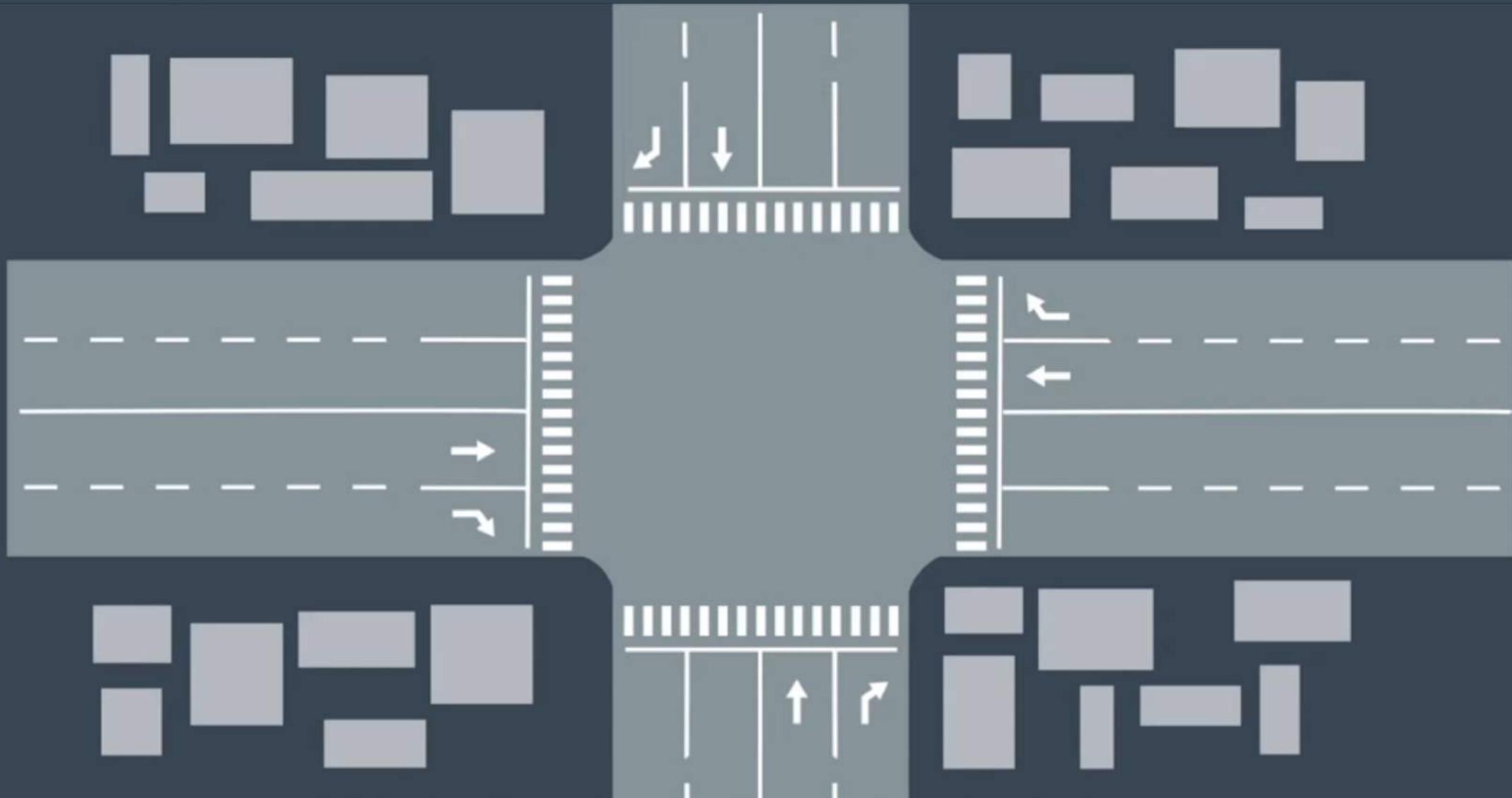
How can we know where we are with 10cm level accuracy ?



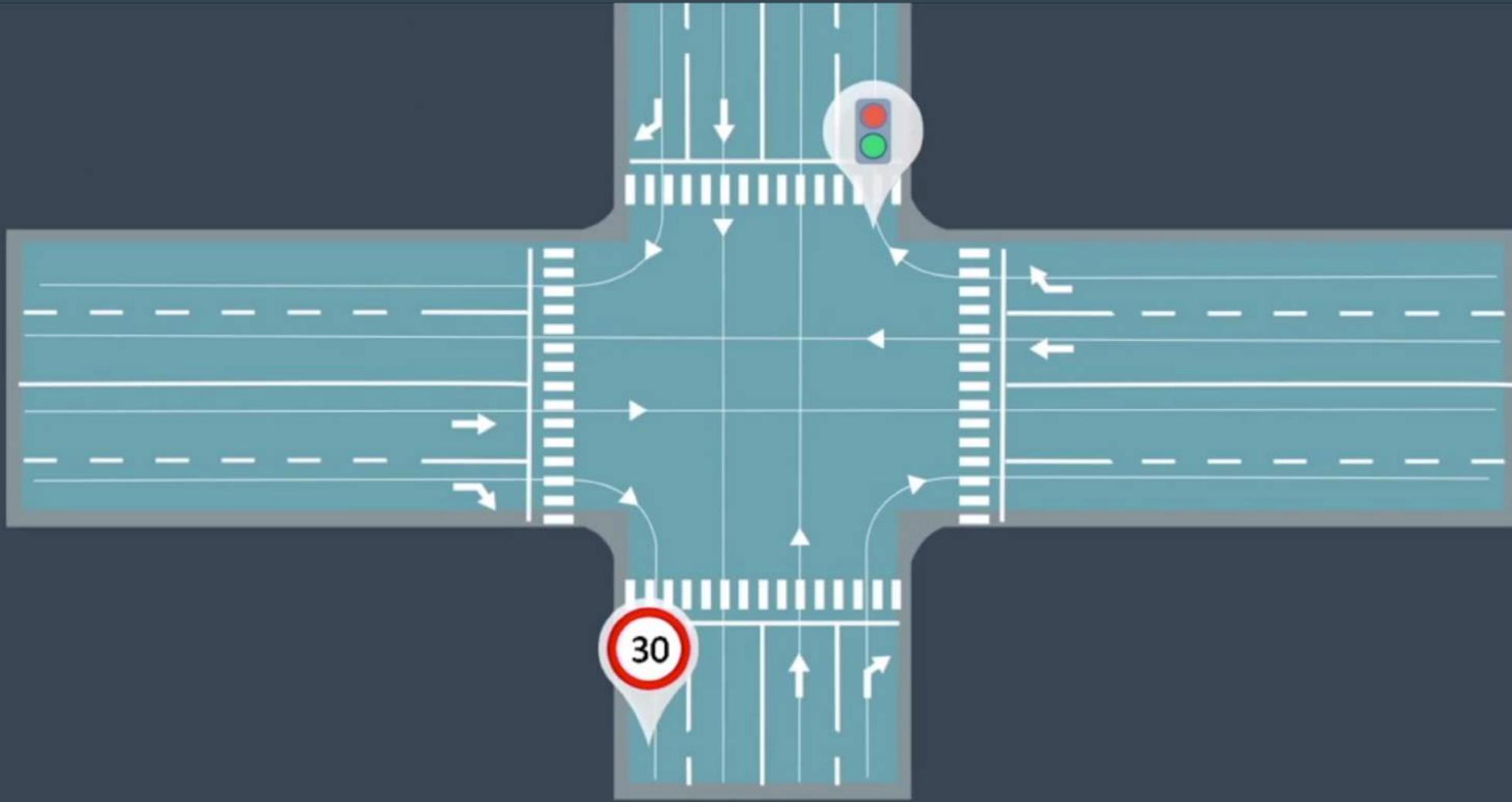




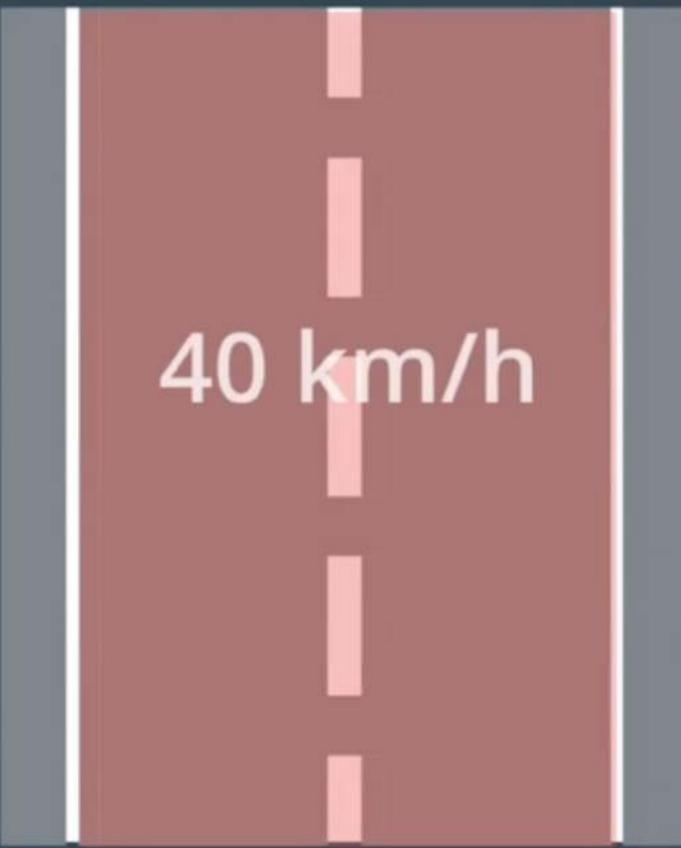


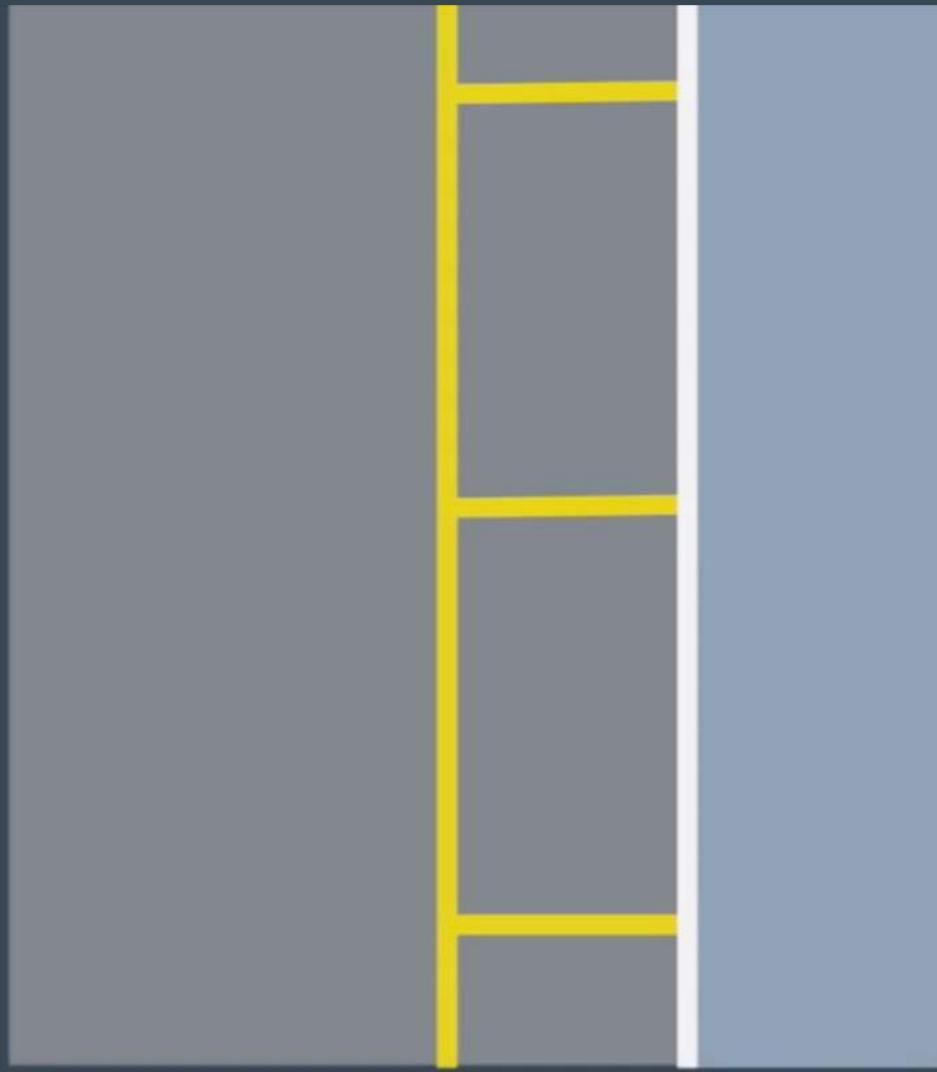


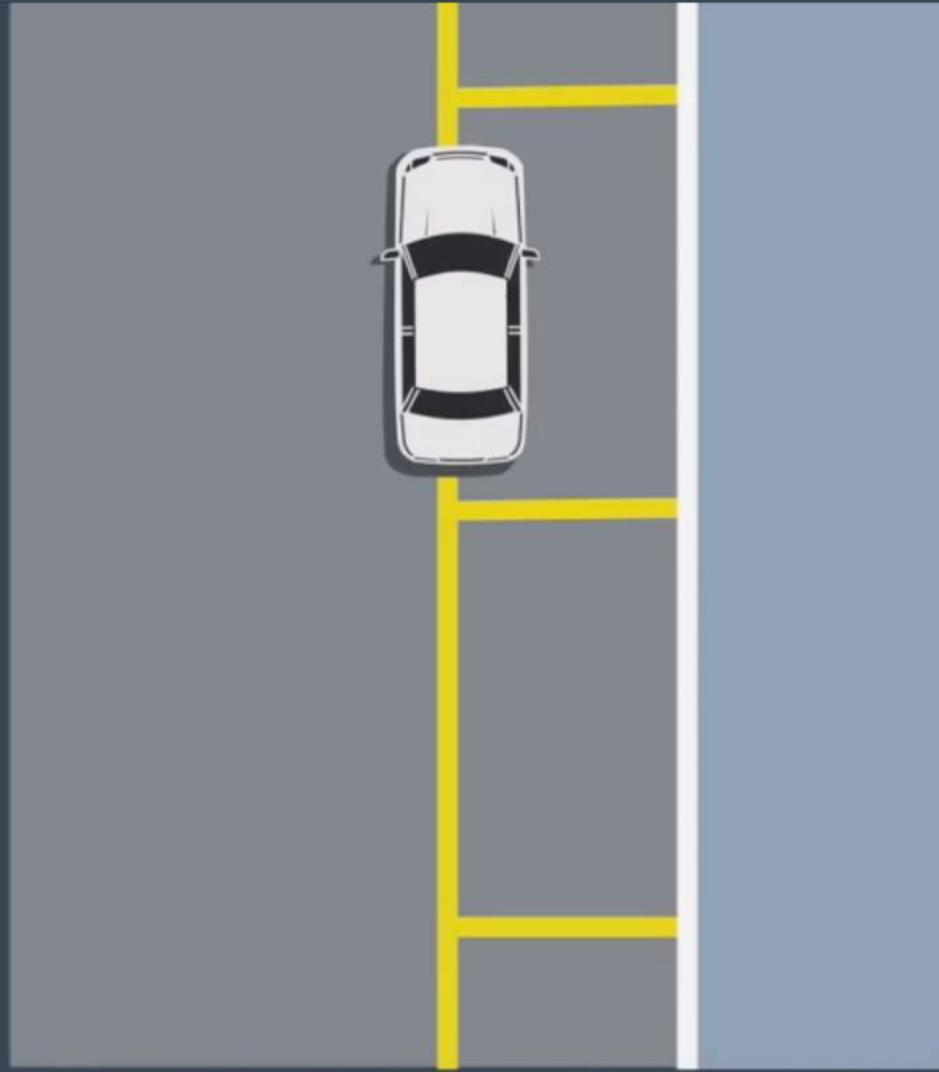
A high definition map contains a huge amount of driving assistance information.



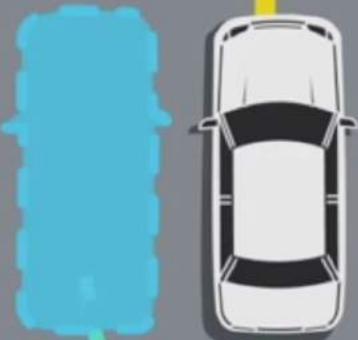
A high definition map also contains a lot of semantic information.

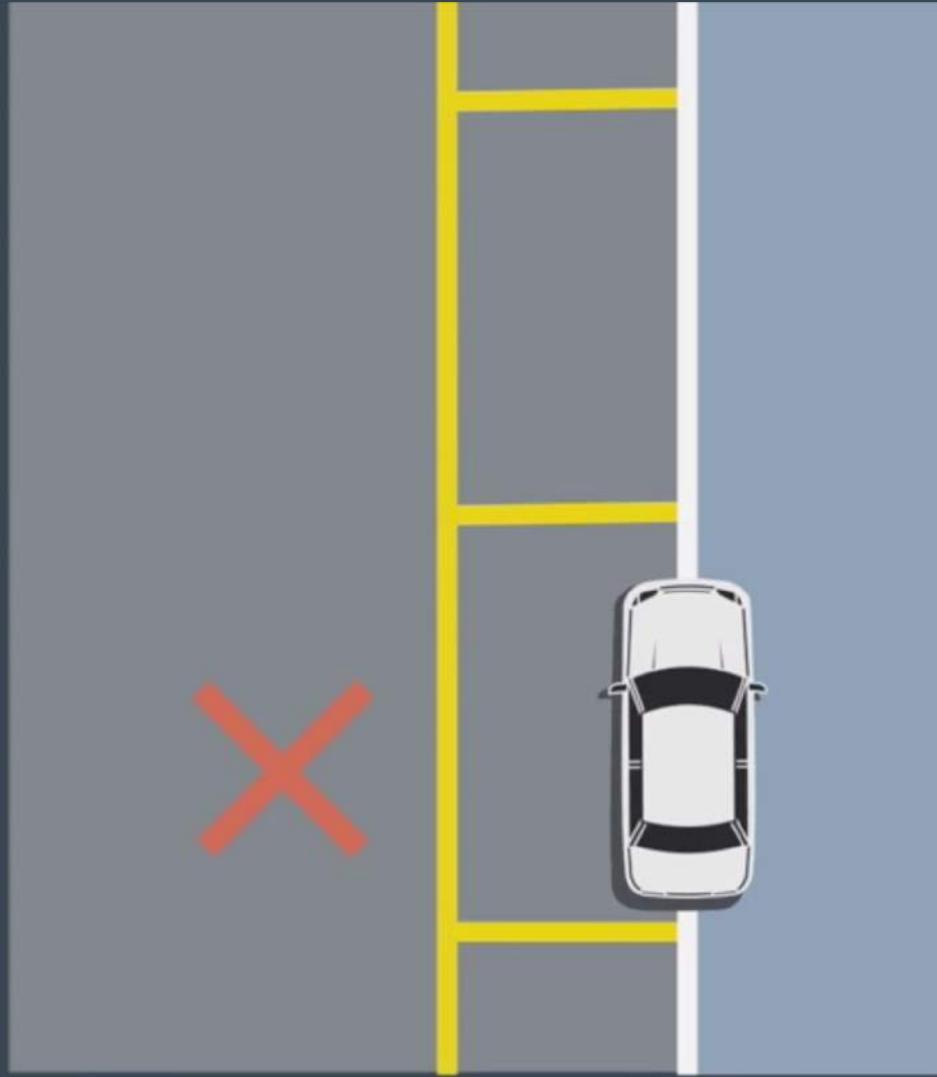






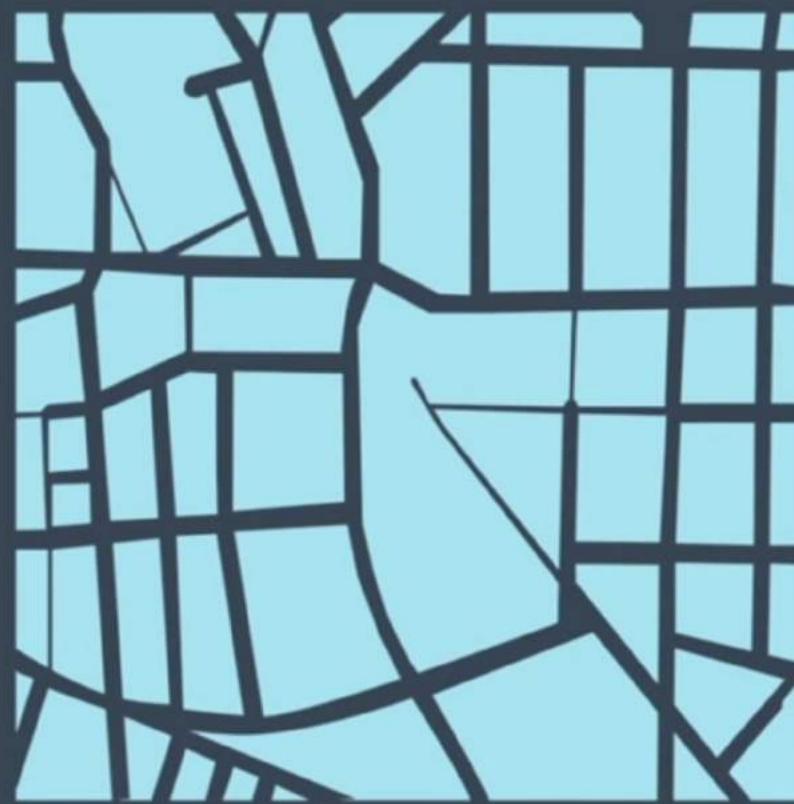
((GPS))







If I were to offer you both a map and a small piece of that same map,



could you find the exact position of the piece on the map?

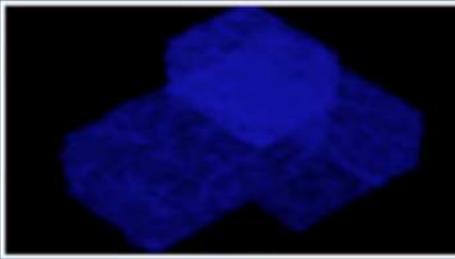
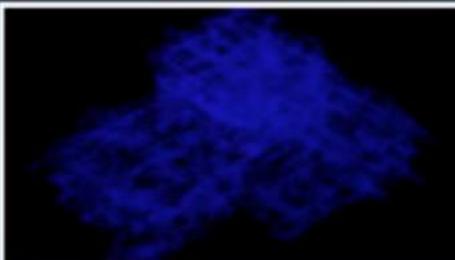
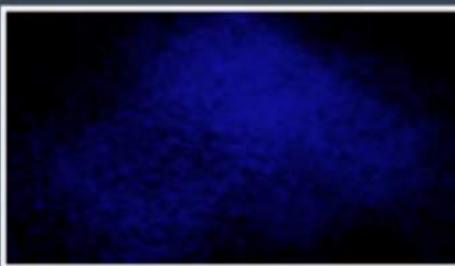
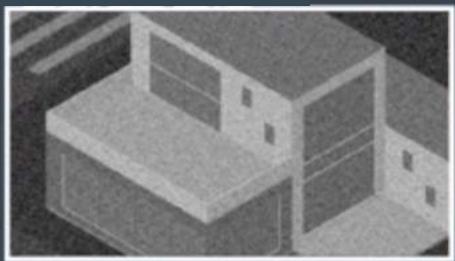


our autonomous vehicle needs to figure out exactly where it is on the map.

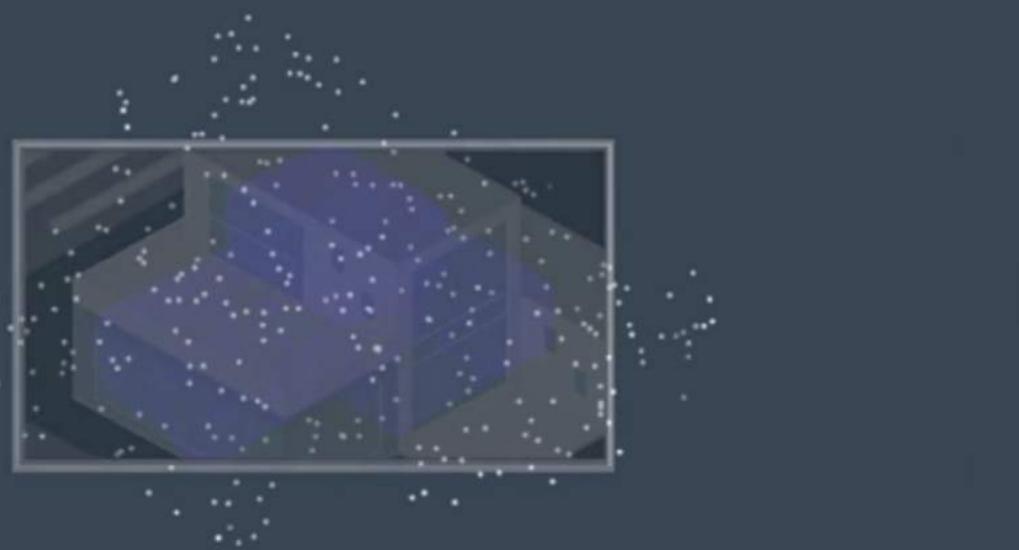


First, the vehicle might look for landmarks.

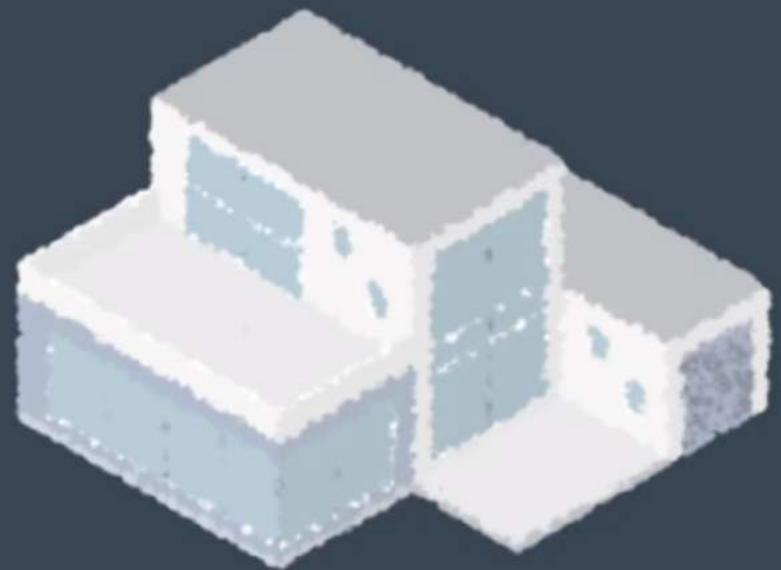


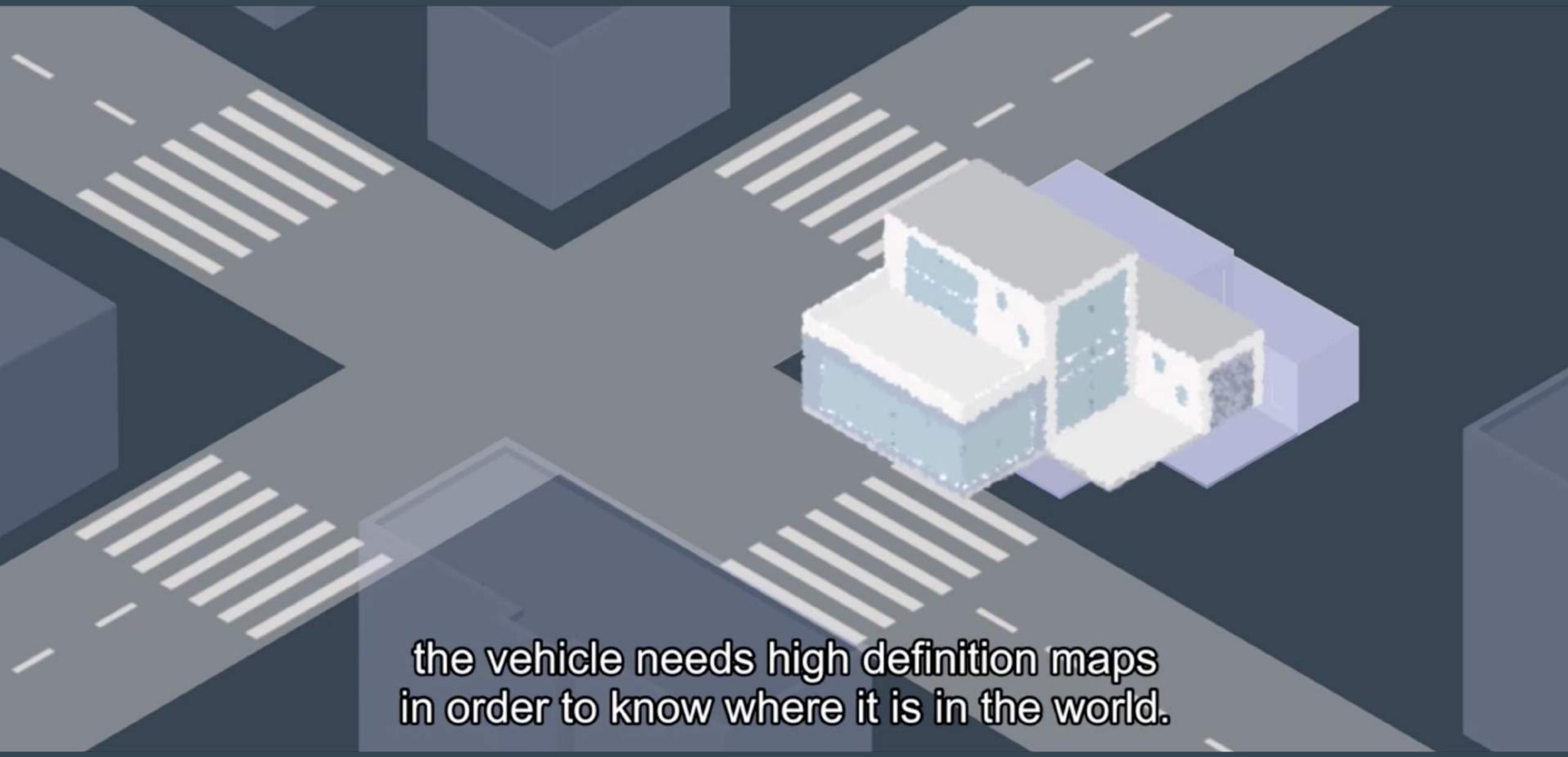


the known landmarks from its high definition map.



Data fusion merges data from different vehicles and different types of sensors.



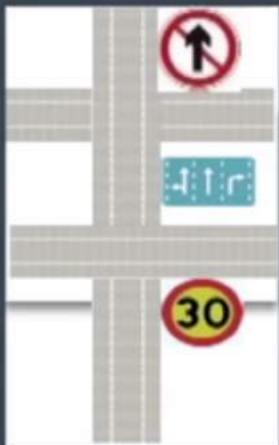


the vehicle needs high definition maps
in order to know where it is in the world.

Map Production

Data Sourcing

Road



The construction of high definition maps is composed of five processes: data sourcing,

Map Production

Data Sourcing

Road

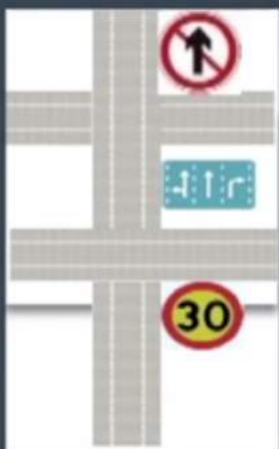


Data Processing

Point Cloud Registration



Localization

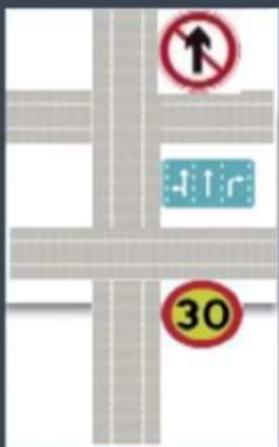


data processing, object detection,

Map Production

Data Sourcing

Road



Data Processing

Point Cloud Registration



Localization

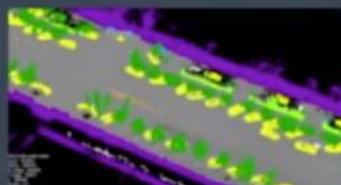


Object Detection

Object Detection



Point Cloud Classification



data processing, object detection,

Map Production

Data Sourcing

Road



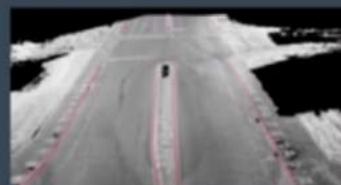
Data Processing

Point Cloud Registration



Object Detection

Object Detection



Manual Verification

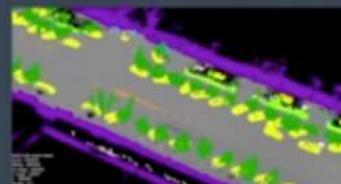
Lane , Curb



Localization



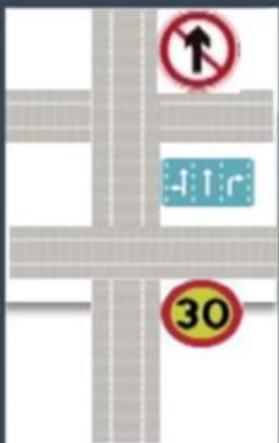
Point Cloud Classification



Traffic Light/Sign



Virtual Lane



Map Production

Data Sourcing

Road



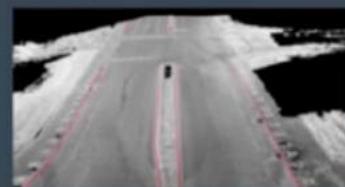
Data Processing

Point Cloud Registration



Object Detection

Object Detection



Manual Verification

Lane , Curb



Map Products

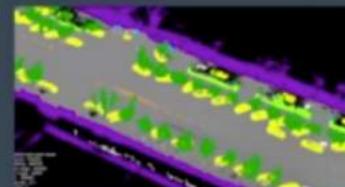
HD Map



Localization



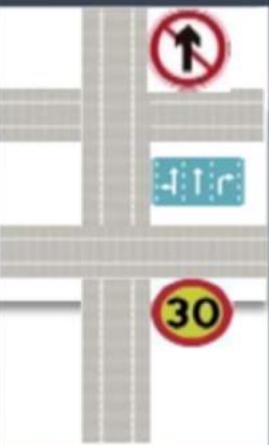
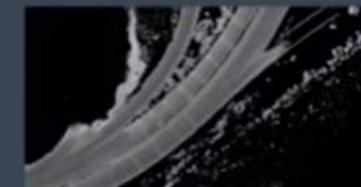
Point Cloud Classification



Traffic Light/Sign



Localization Map



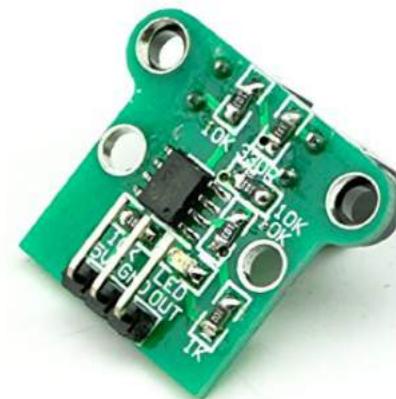
Virtual Lane



Point Cloud Map



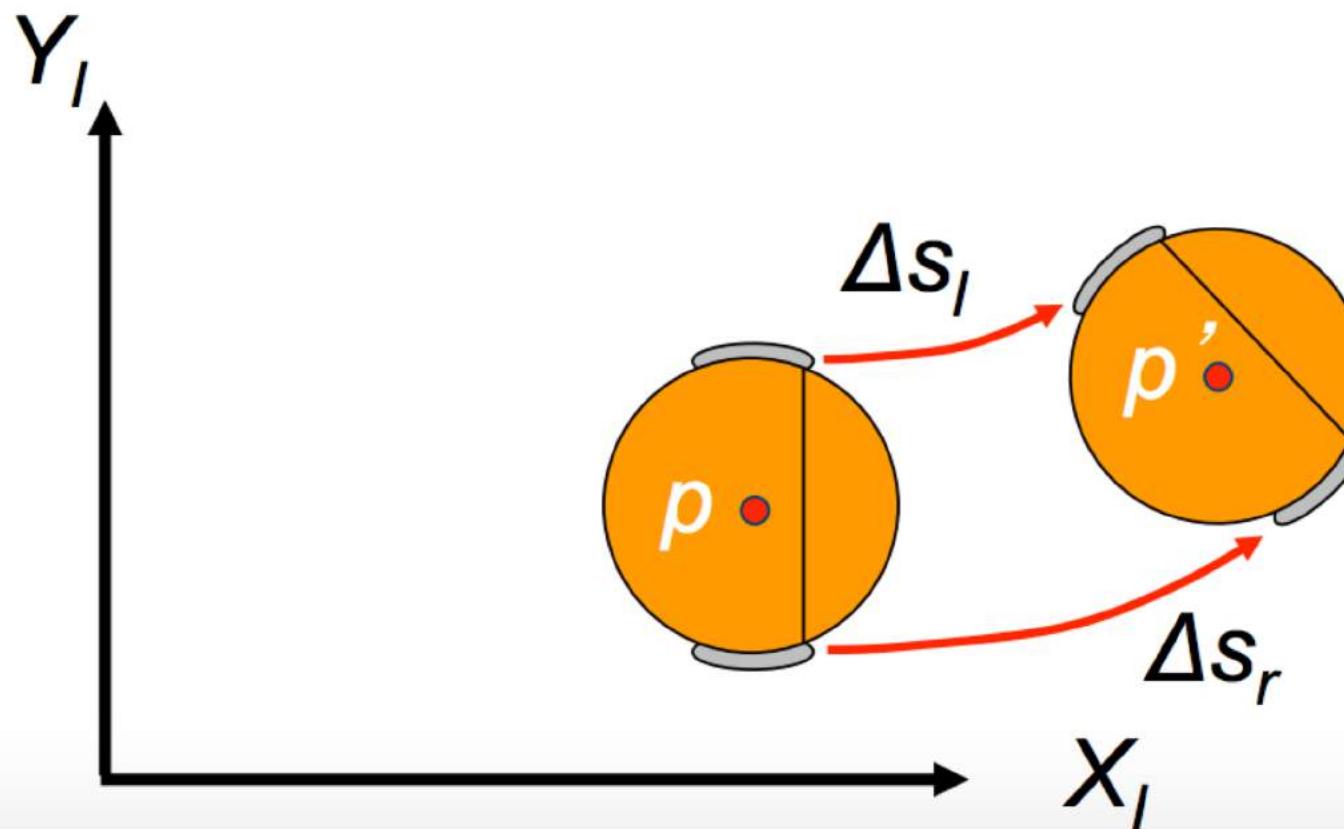
Odometry -Wheel Encoders



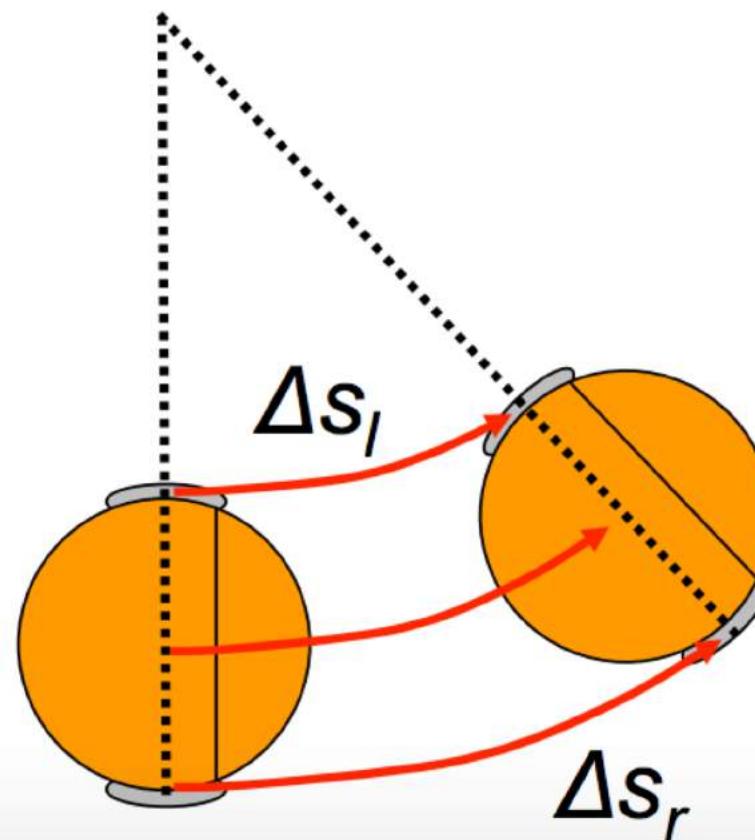
Dead-Reckoning/Odometry

- Estimation of position without reference
- The position is integrated using motion model or measurements
 - Encoders
 - Gyros
 - IMU
 - ...

- If a robot starts from a position p , and the right and left wheels move respective distances Δs_r and Δs_l , what is the resulting new position p' ?



- To start, let's model the change in angle $\Delta\theta$ and distance travelled Δs by the robot.
 - Assume the robot is travelling on a circular arc of constant radius.

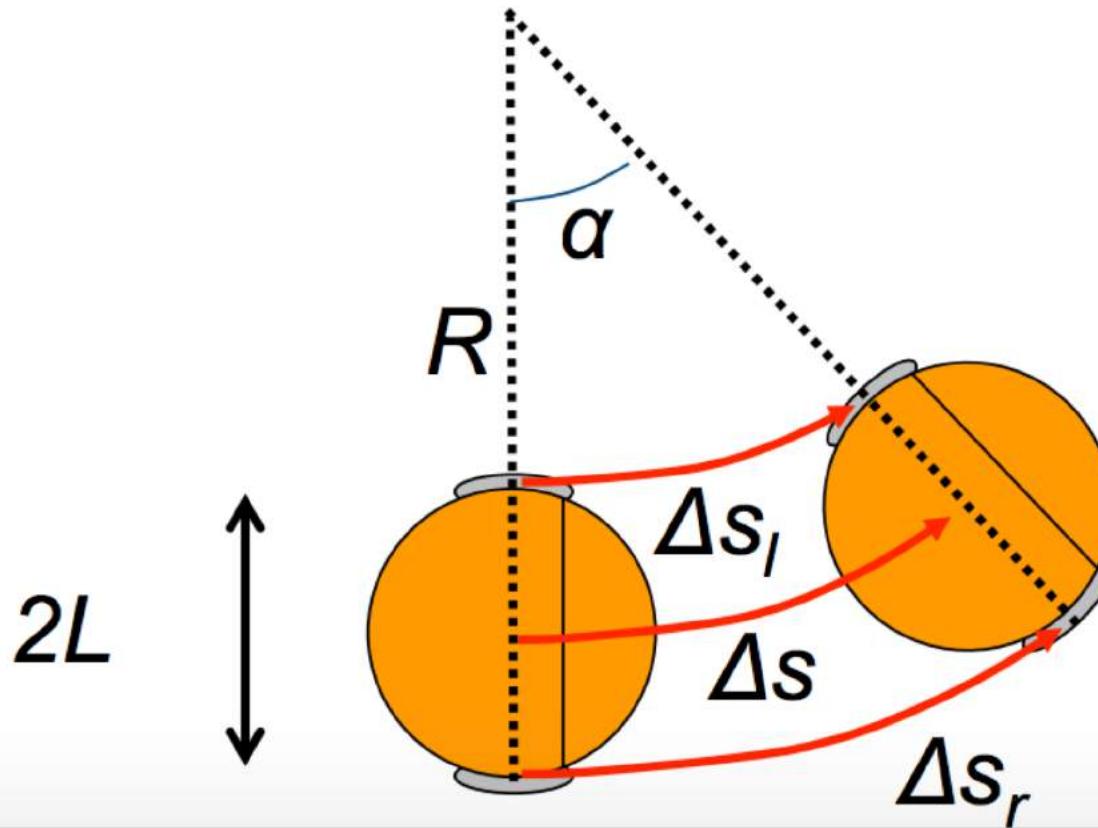


- Begin by noting the following holds for circular arcs:

$$\Delta s_i = R\alpha$$

$$\Delta s_r = (R+2L)\alpha$$

$$\Delta s = (R+L)\alpha$$



- Now manipulate first two equations:

$$\Delta s_l = Ra \quad \Delta s_r = (R+2L)a$$

To:

$$Ra = \Delta s_l$$

$$La = (\Delta s_r - Ra)/2$$

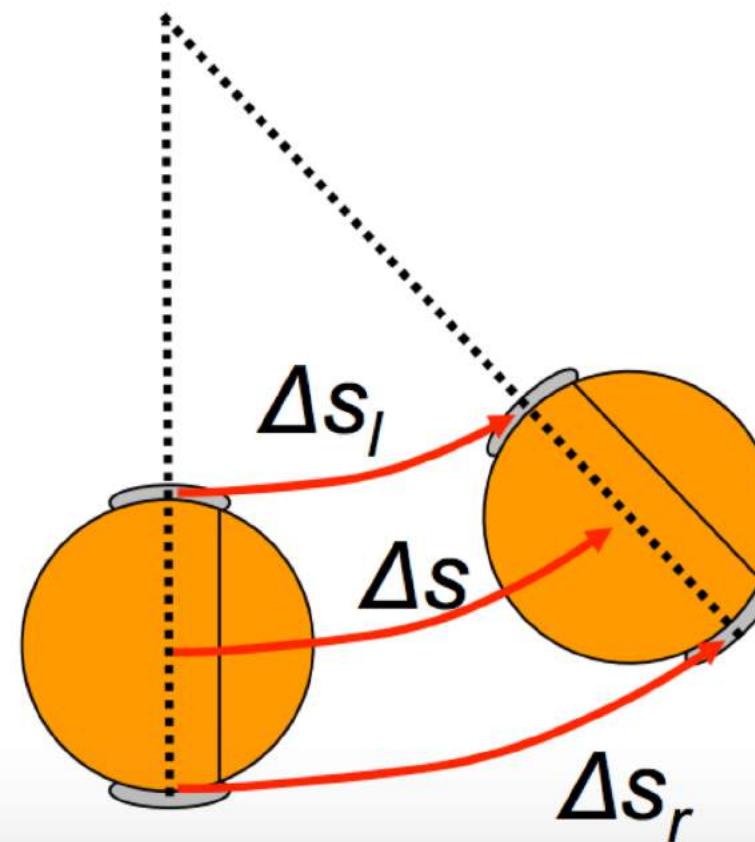
$$= \Delta s_r/2 - \Delta s_l/2$$

- Substitute this into last equation for Δs :

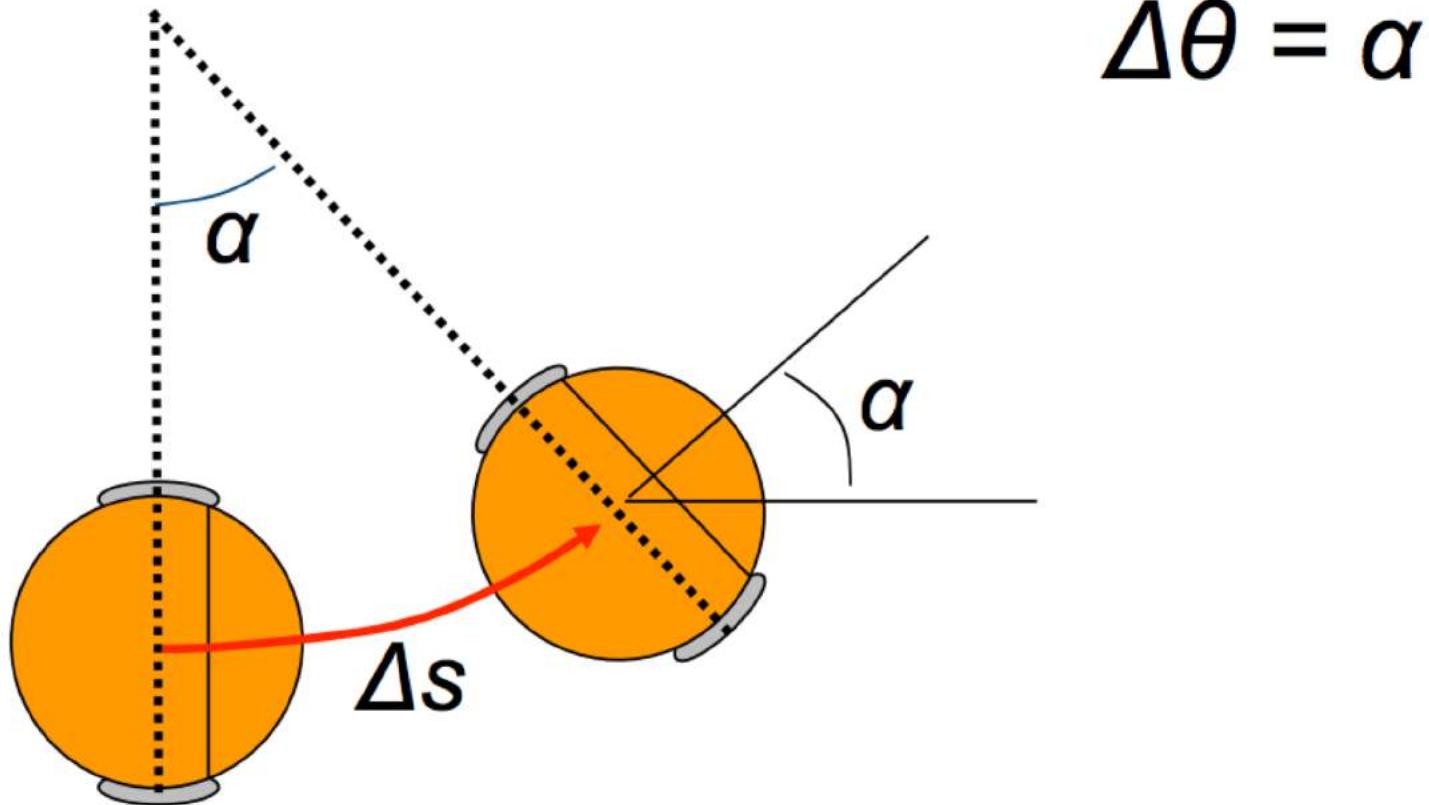
$$\begin{aligned}\Delta s &= (R+L)\alpha \\&= R\alpha + L\alpha \\&= \Delta s_l + \Delta s_r/2 - \Delta s_l/2 \\&= \Delta s_l/2 + \Delta s_r/2 \\&= \frac{\Delta s_l + \Delta s_r}{2}\end{aligned}$$

- Or, note the distance the center travelled is simply the average distance of each wheel:

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$



- To calculate the change in angle $\Delta\theta$, observe that it equals the rotation about the circular arc's center point



- So we solve for α by equating α from the first two equations:

$$\Delta s_l = R\alpha \quad \Delta s_r = (R+2L)\alpha$$

This results in:

$$\Delta s_l / R = \Delta s_r / (R+2L)$$

$$(R+2L) \Delta s_l = R \Delta s_r$$

$$2L \Delta s_l = R (\Delta s_r - \Delta s_l)$$

$$\frac{2L \Delta s_l}{(\Delta s_r - \Delta s_l)} = R$$

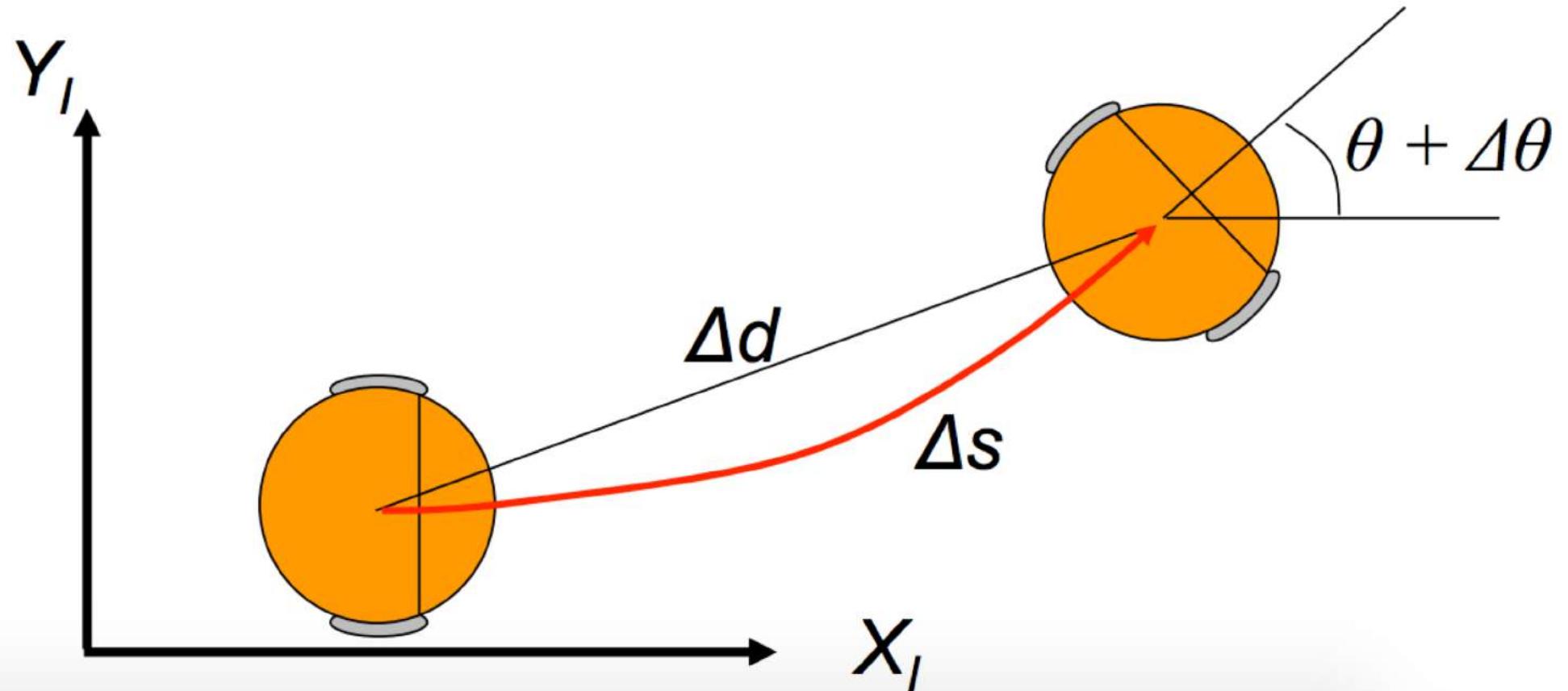
- Substitute R into

$$\begin{aligned}\alpha &= \Delta s_l / R \\ &= \Delta s_l (\Delta s_r - \Delta s_l) / (2L \Delta s_l) \\ &= \frac{(\Delta s_r - \Delta s_l)}{2L}\end{aligned}$$

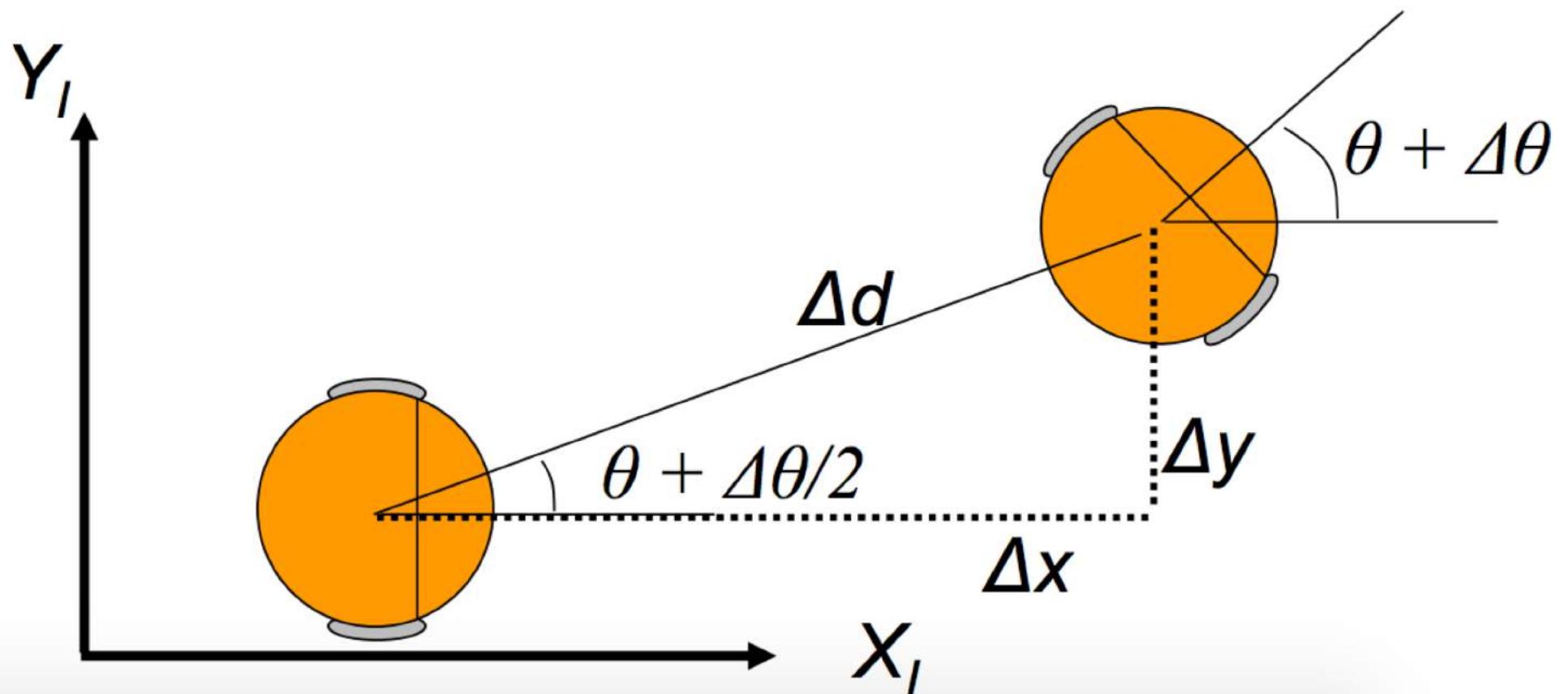
So...

$$\Delta\theta = \frac{(\Delta s_r - \Delta s_l)}{2L}$$

- Now that we have $\Delta\theta$ and Δs , we can calculate the position change in global coordinates.
 - We use a new segment of length Δd .



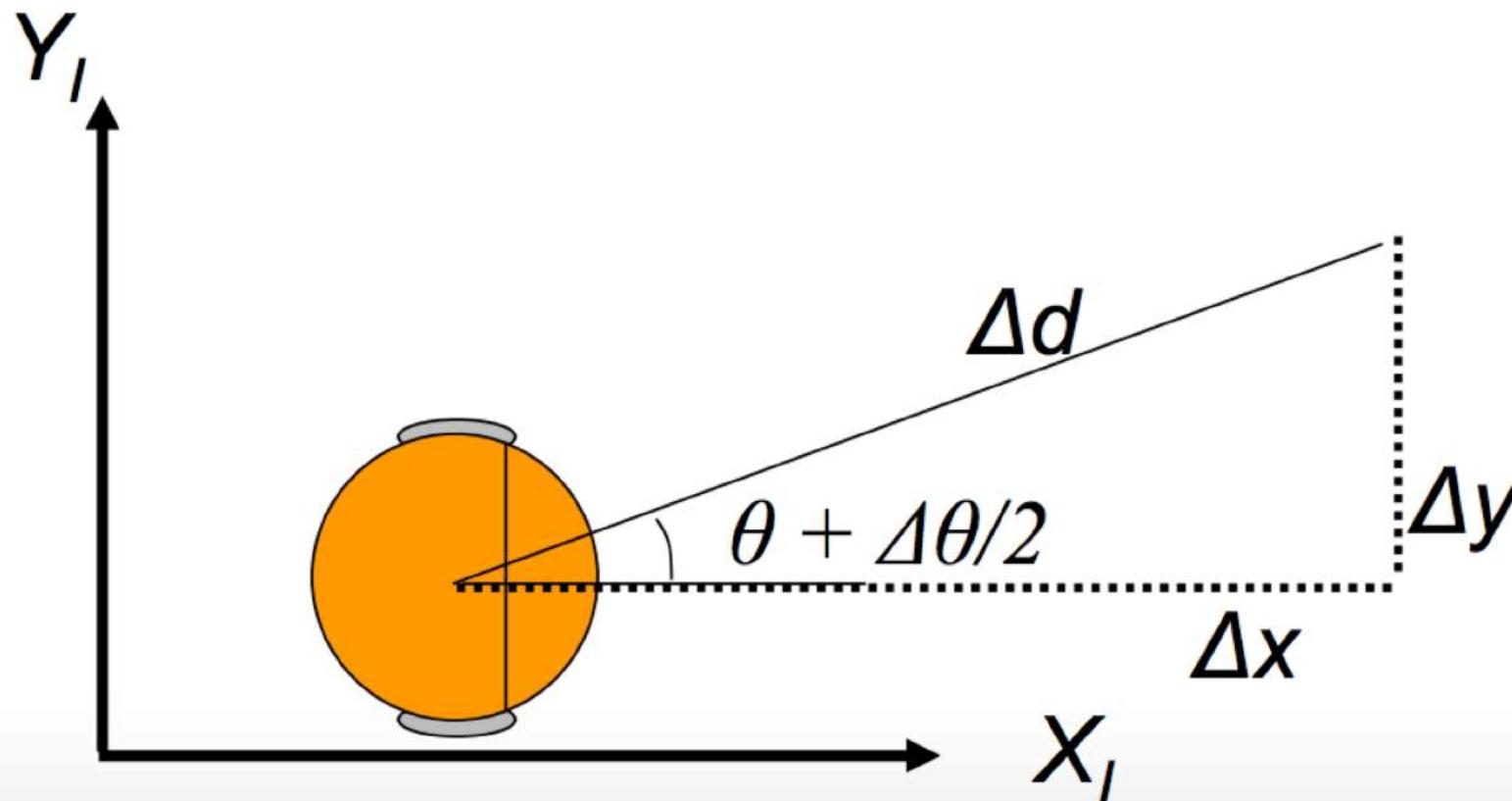
- Now calculate the change in position as a function of Δd .



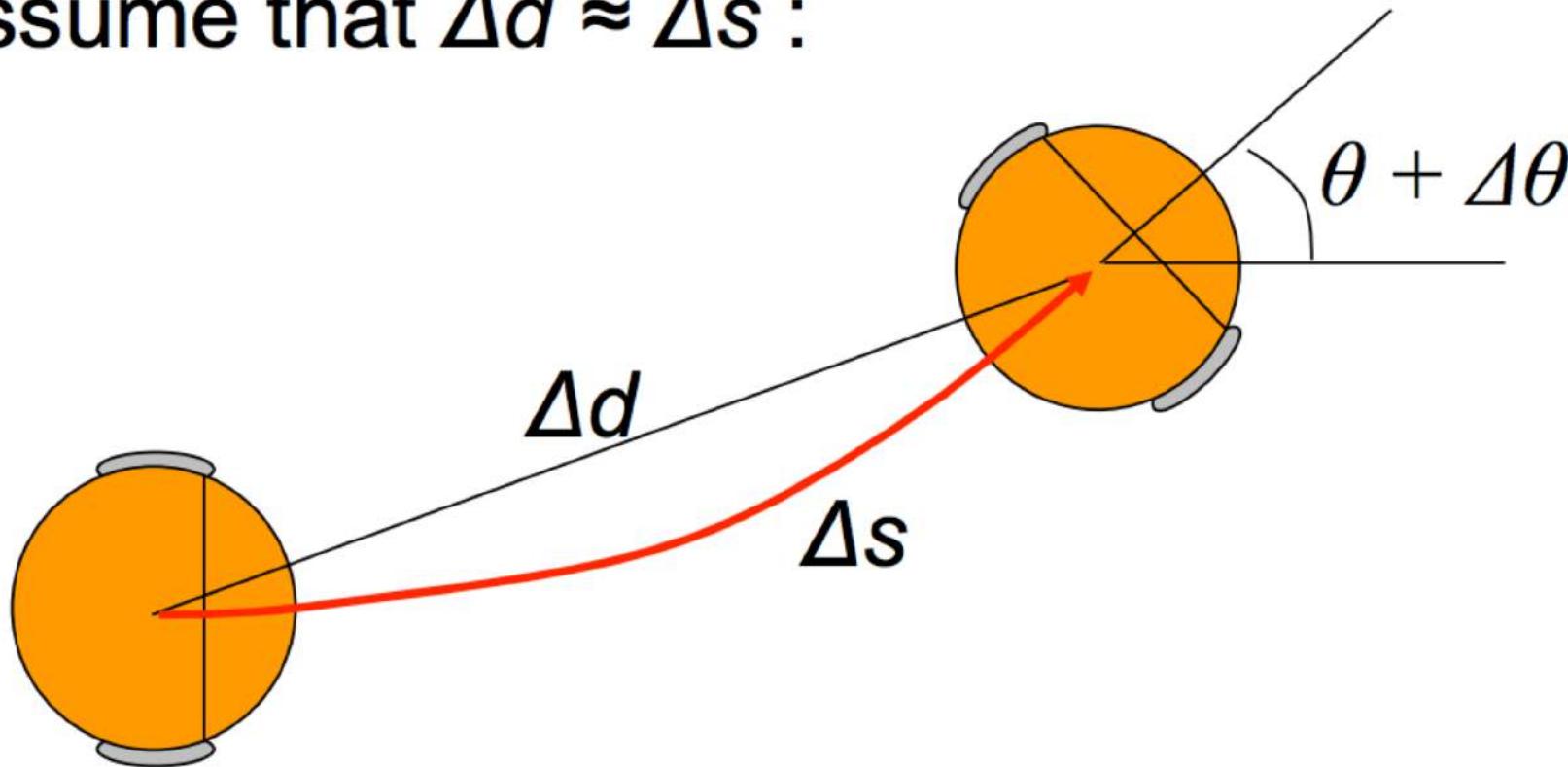
- Using Trig:

$$\Delta x = \Delta d \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta d \sin(\theta + \Delta\theta/2)$$



- Now if we assume that the motion is small, then we can assume that $\Delta d \approx \Delta s$:

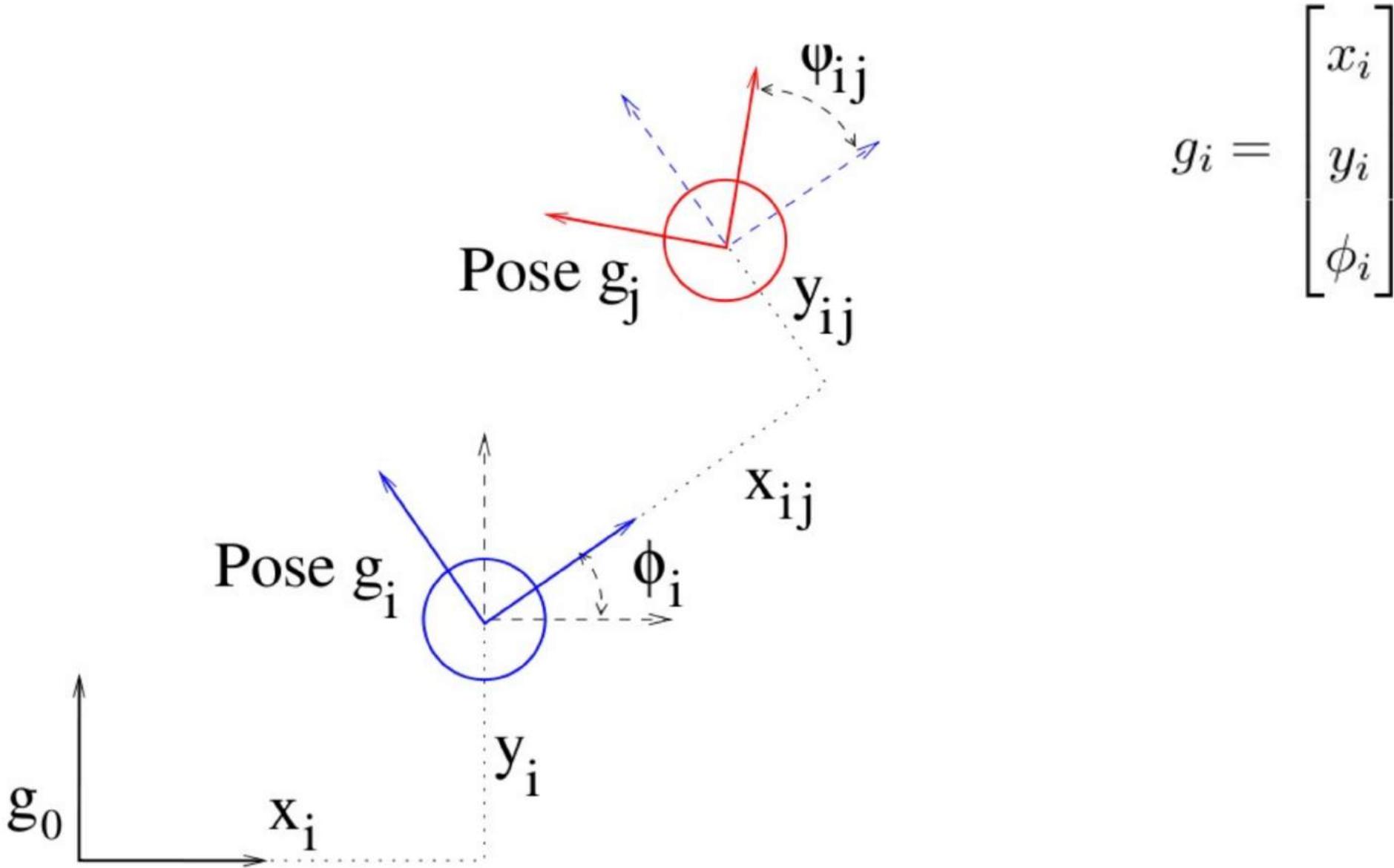


- So...

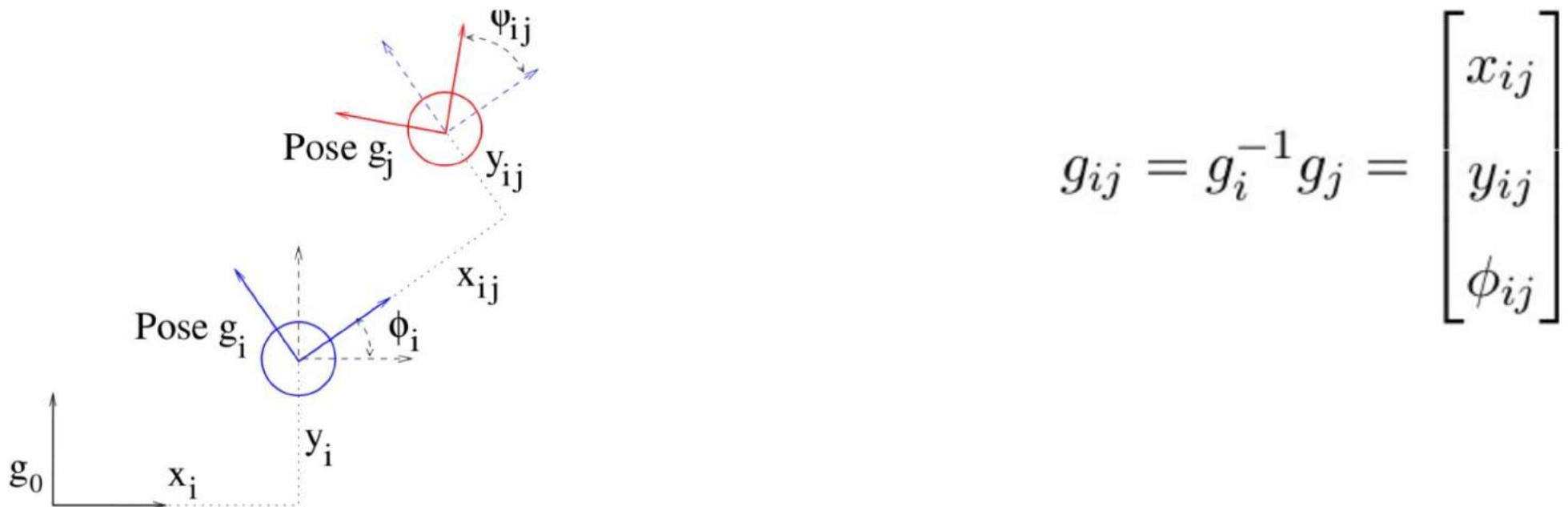
$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

Dead-reckoning



In more general form



$$g_{ij} = g_i^{-1} g_j = \begin{bmatrix} x_{ij} \\ y_{ij} \\ \phi_{ij} \end{bmatrix}$$

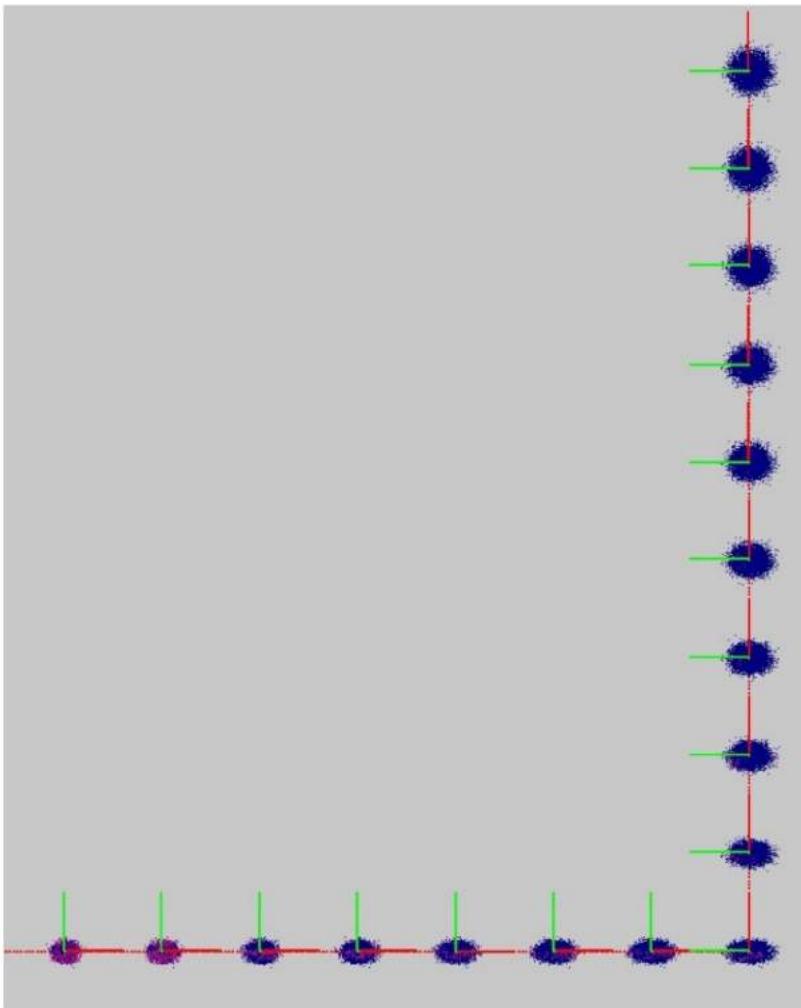
$$g_j = \begin{bmatrix} x_j \\ y_j \\ \phi_j \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix} + \begin{bmatrix} \cos(\phi_i) & -\sin(\phi_i) & 0 \\ \sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ij} \\ y_{ij} \\ \phi_{ij} \end{bmatrix}.$$

Noise/Errors

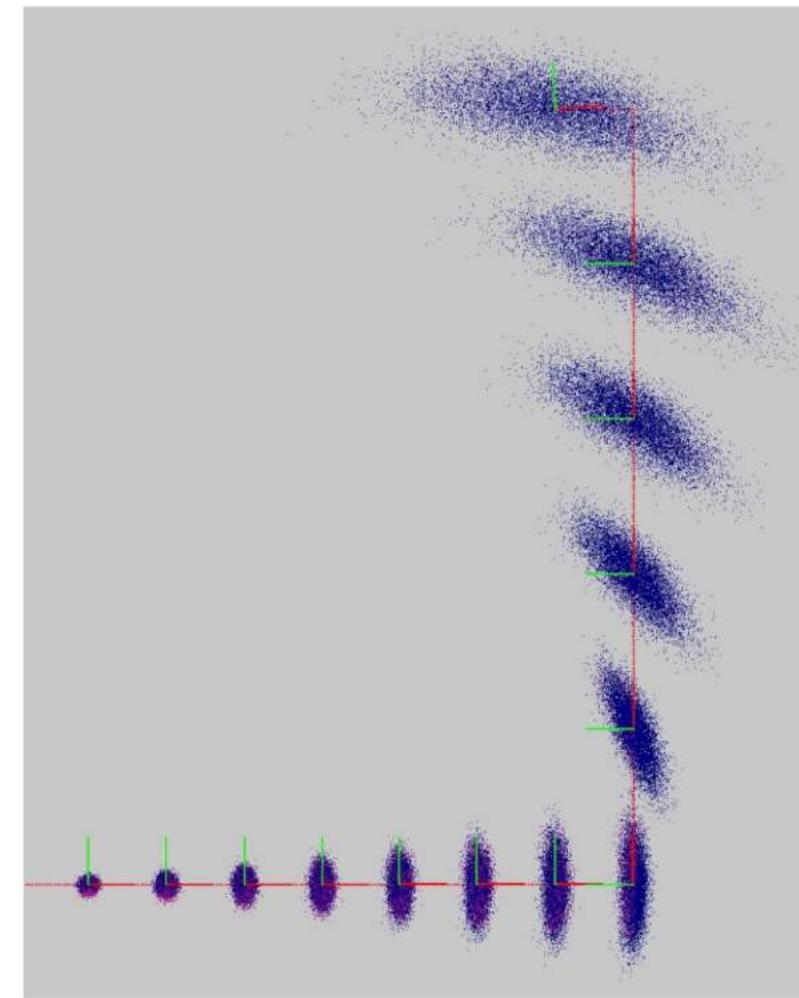
$$g_{ij} = \begin{bmatrix} x_{ij} + e_{x_{ij}} \\ y_{ij} + e_{y_{ij}} \\ \phi_{ij} + e_{\phi_{ij}} \end{bmatrix}$$

$$g_j = \begin{bmatrix} x_j \\ y_j \\ \phi_j \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix} + \begin{bmatrix} \cos(\phi_i) & -\sin(\phi_i) & 0 \\ \sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ij} \\ y_{ij} \\ \phi_{ij} \end{bmatrix}.$$

Evolution of noise



No heading error



With heading error

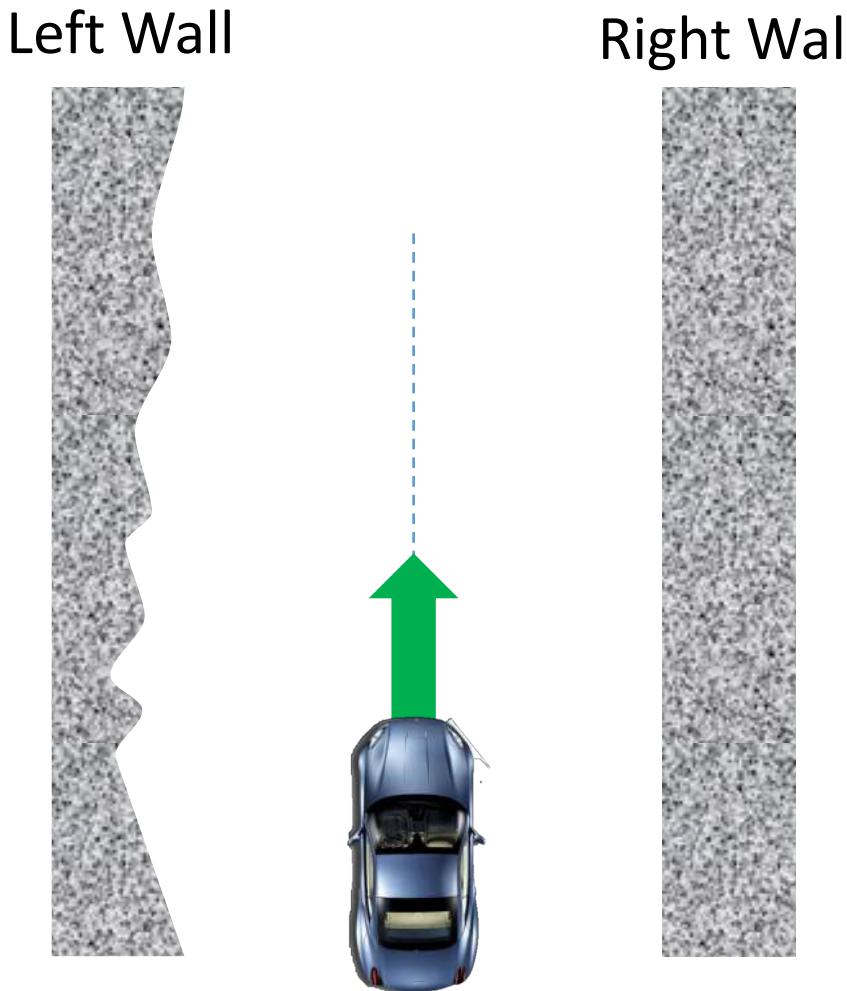
Scan Matching



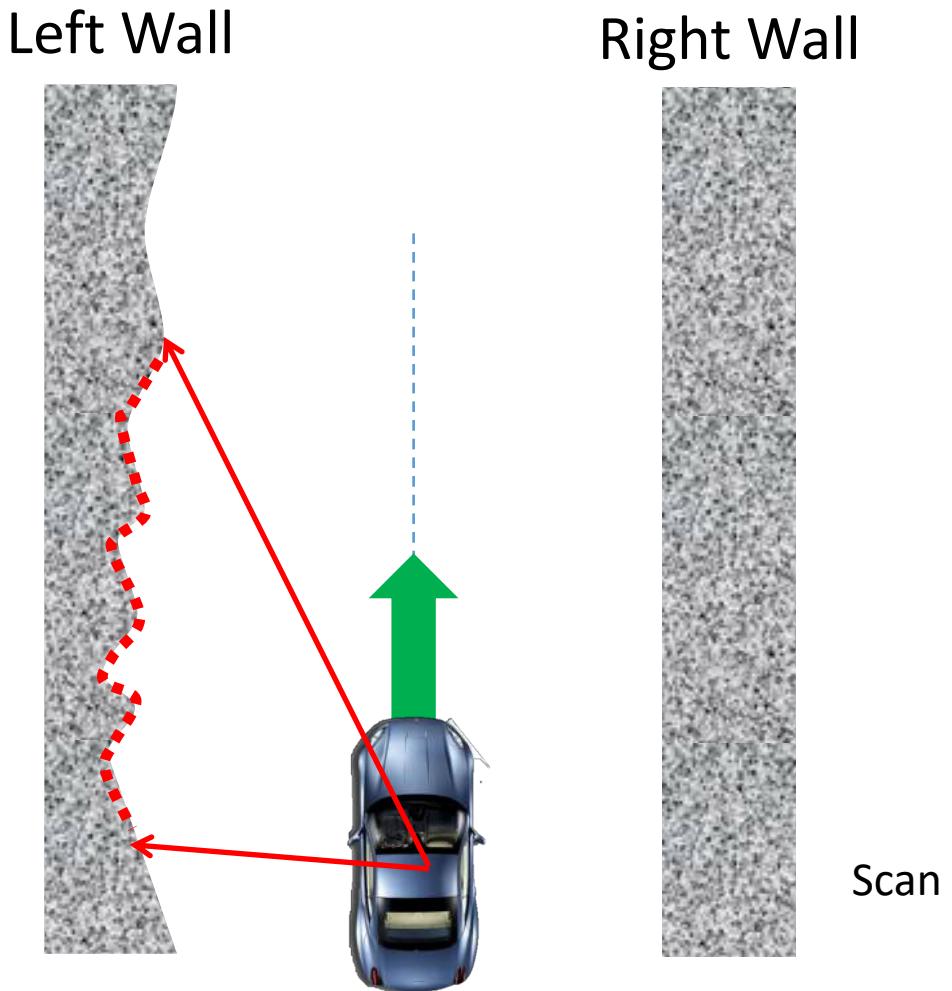
Odometry with LIDAR

- Odometry = Estimation of position (x, y, z) and attitude (θ, φ, ψ)
- In what follows, will use “position” as short for “position and attitude”

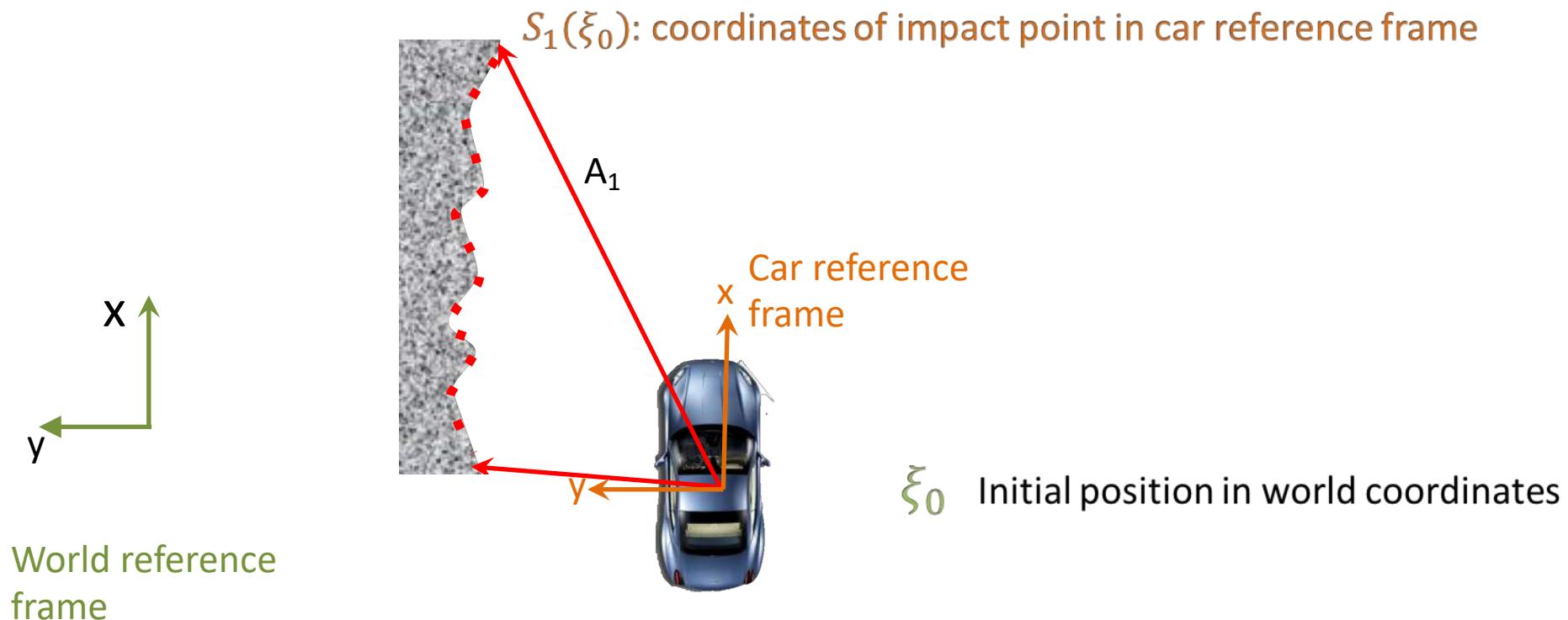
LIDAR for odometry: Scan matching



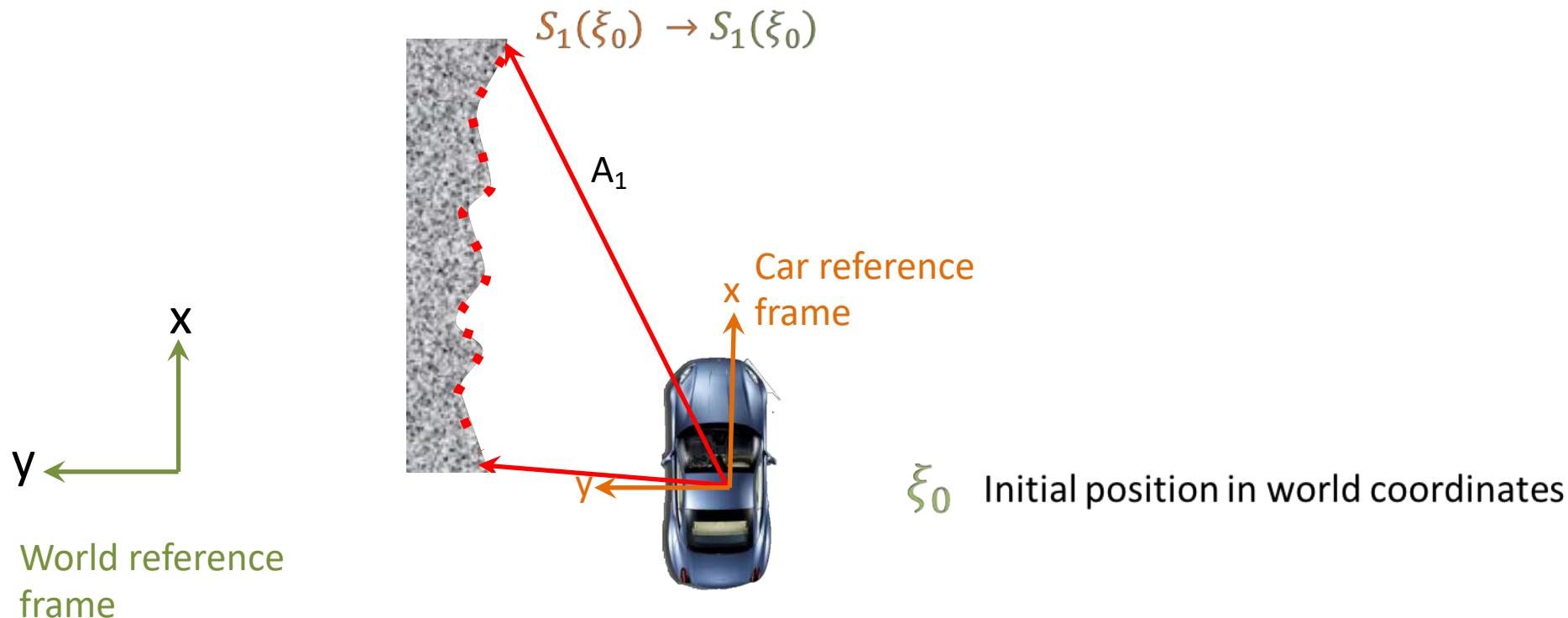
LIDAR for odometry: Scan matching

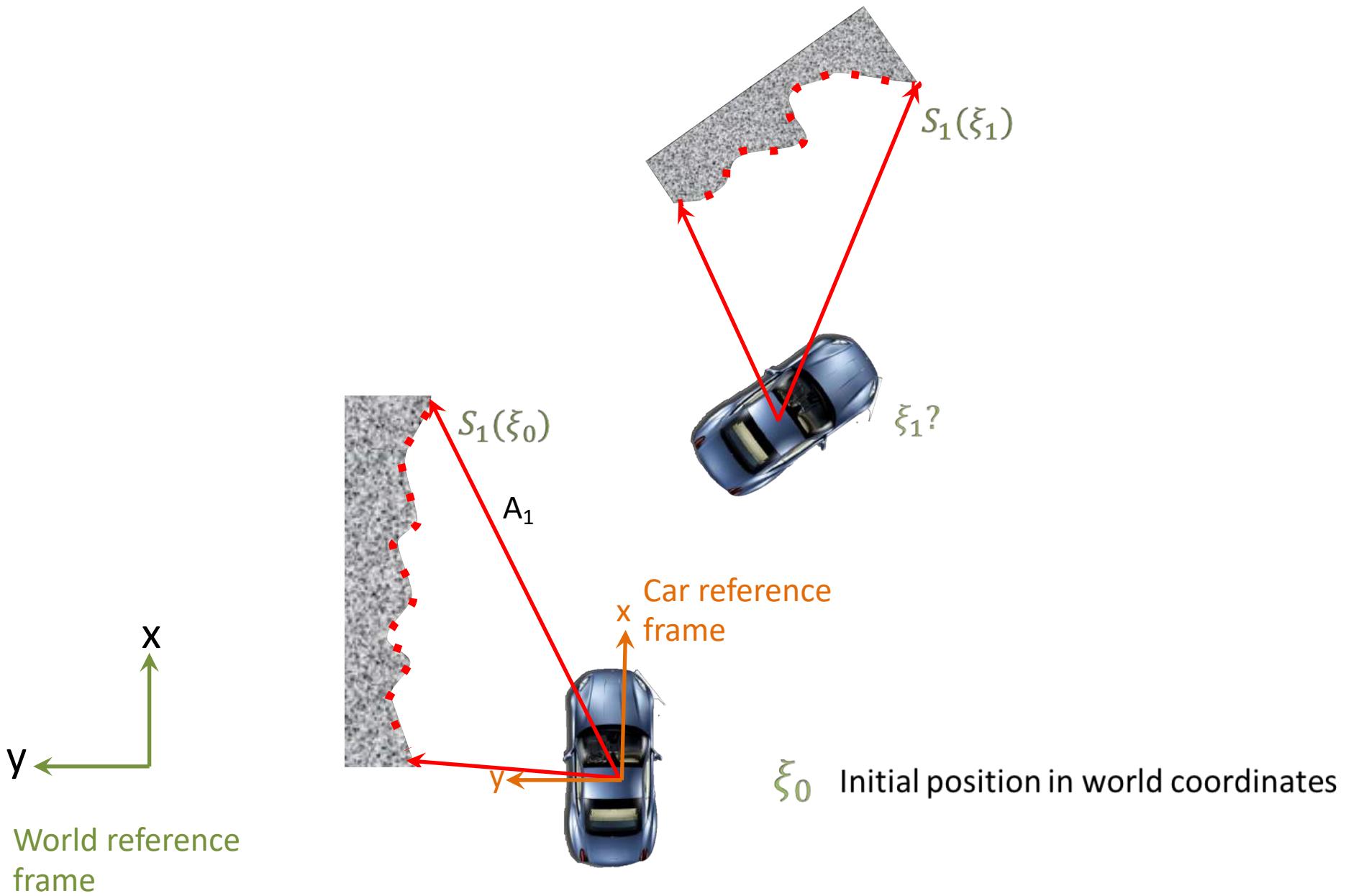


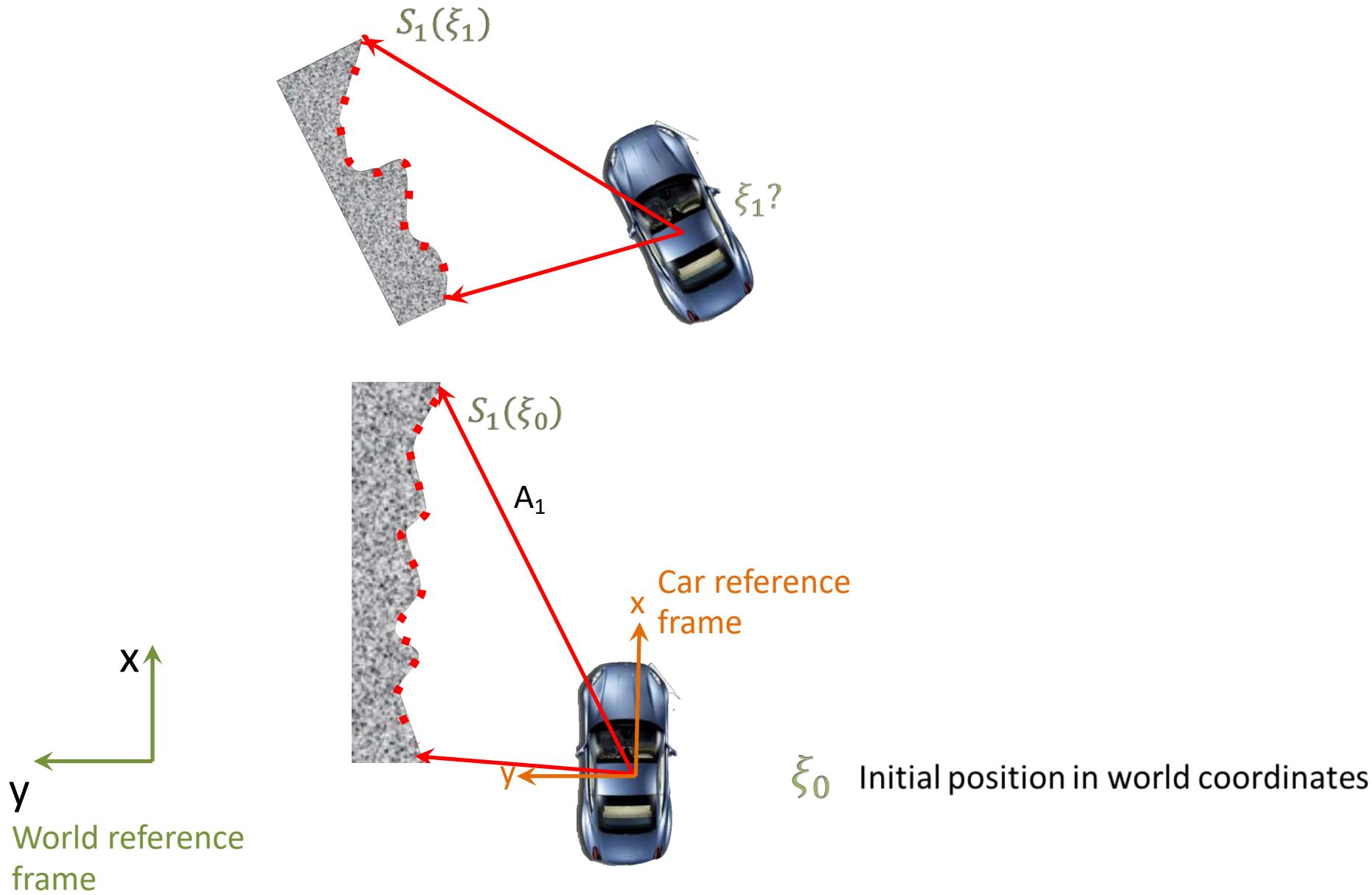
LIDAR for odometry: Scan 1

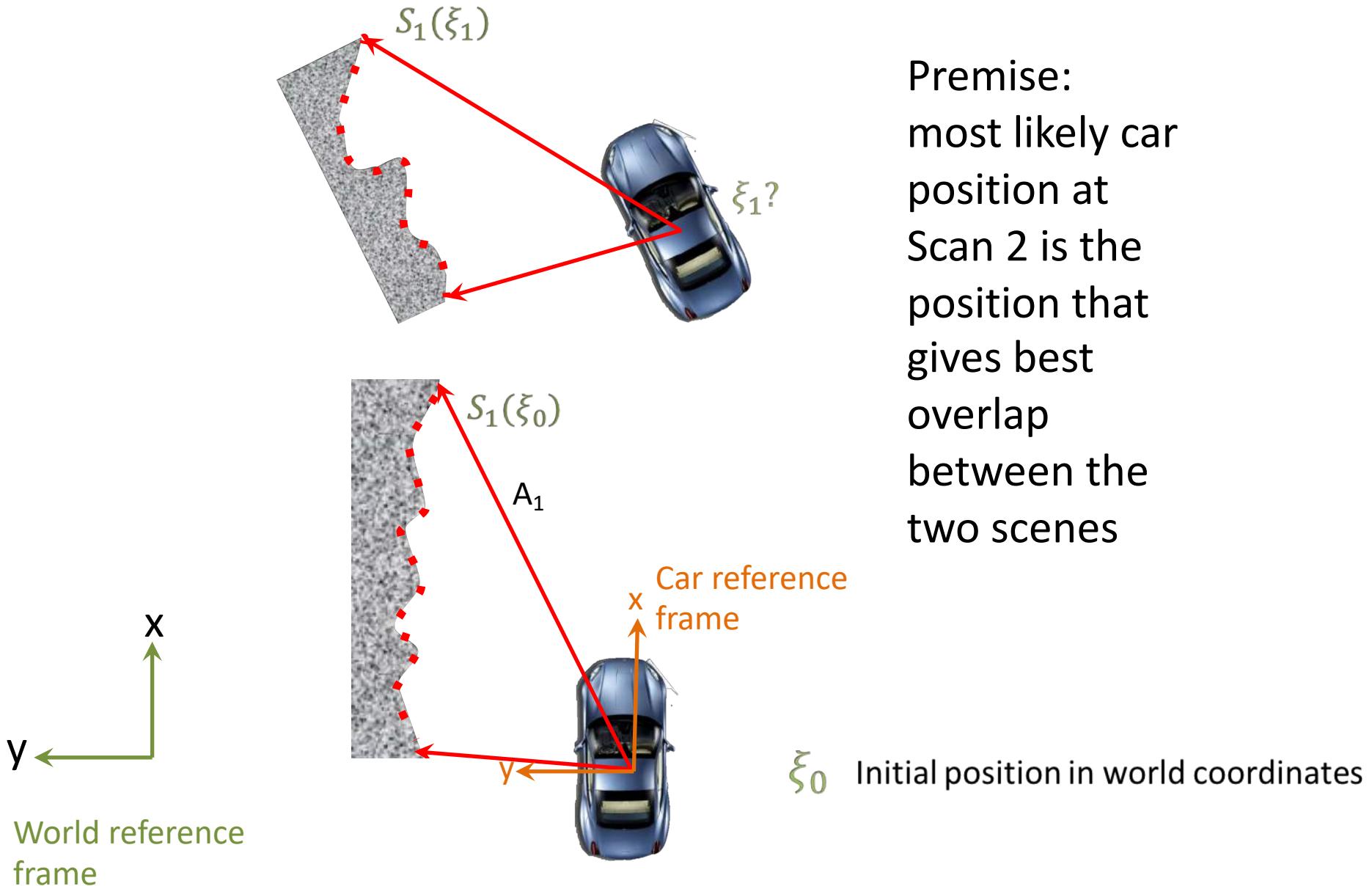


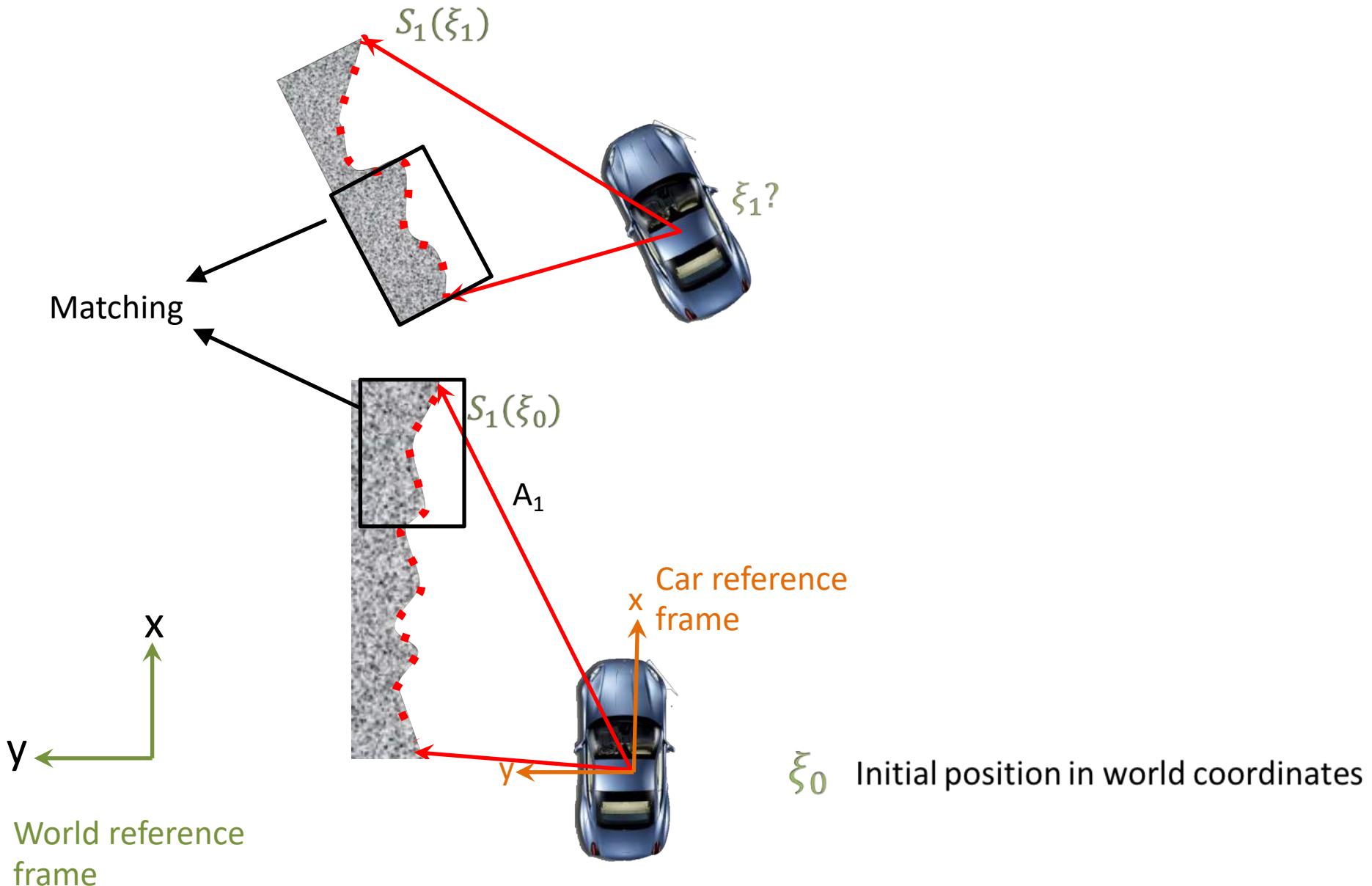
LIDAR for odometry: Scan 1

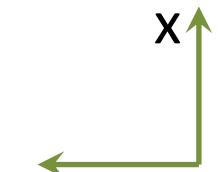


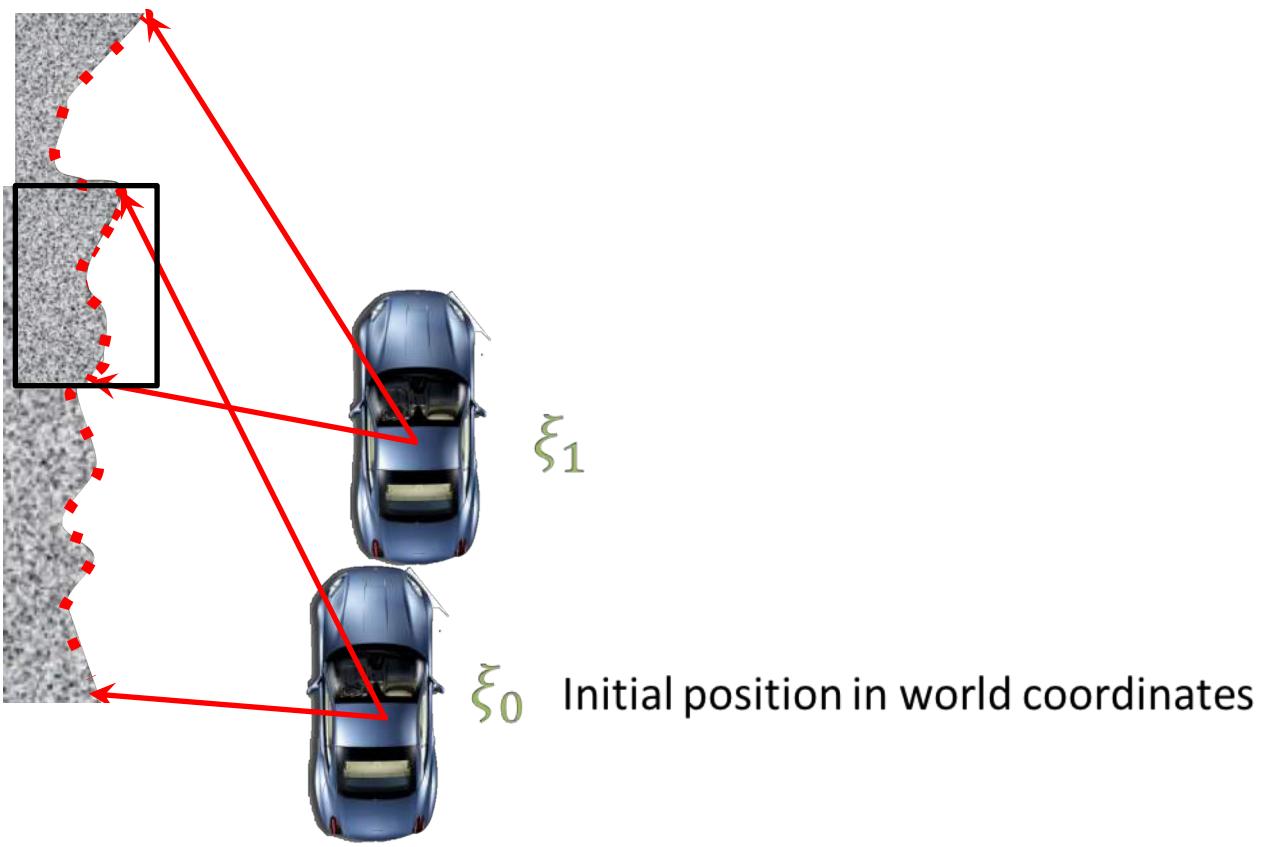




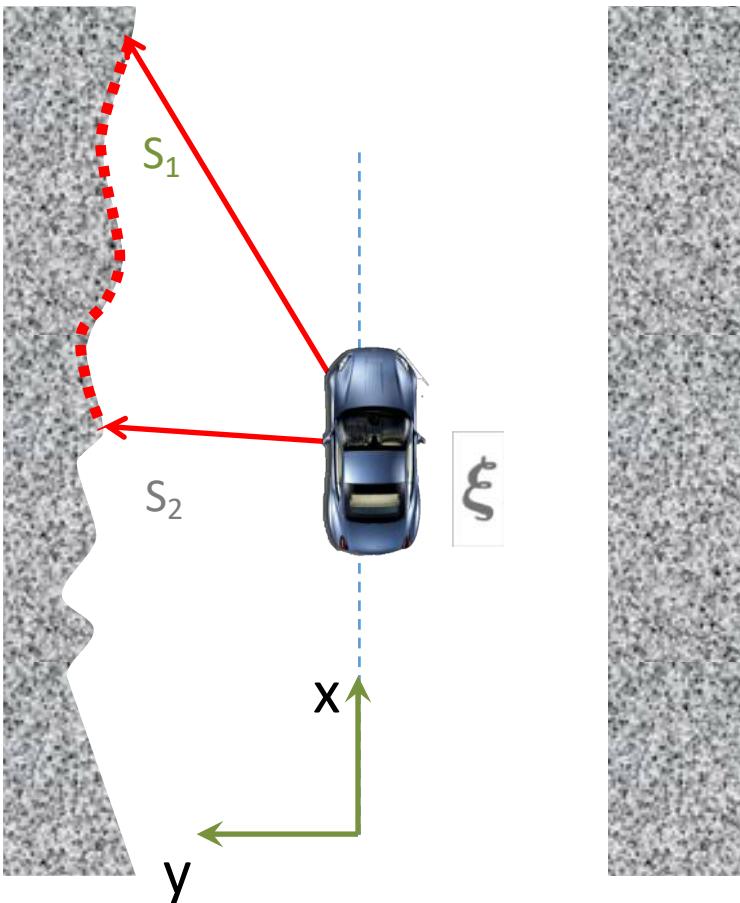




 World reference frame



Scan matching: optimization



Impact coordinates of i^{th} step in world frame

Total of n steps ($n = 1084$)

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

Measure of match