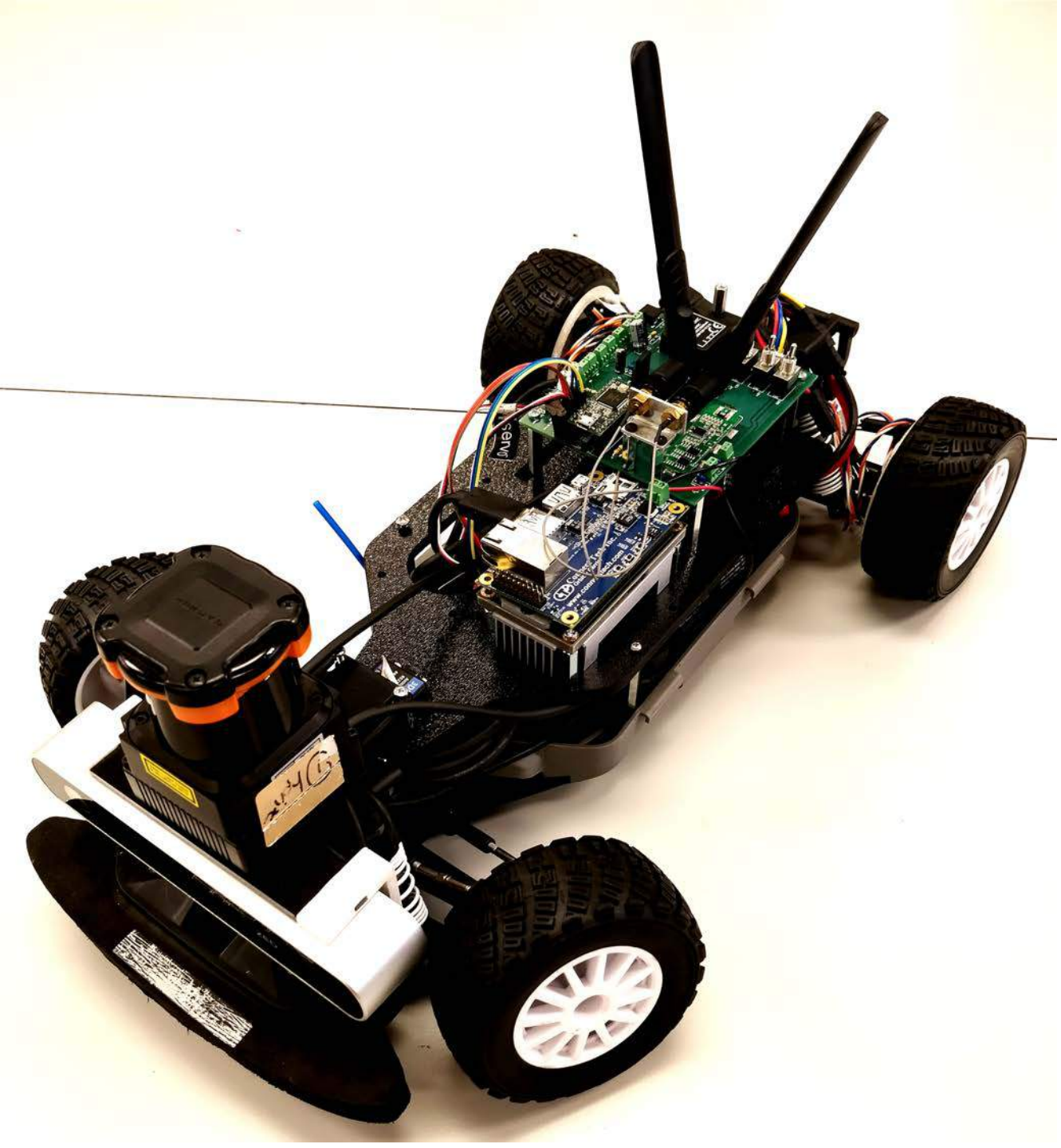


F1/10
Autonomous Racing

Perception
Localization

Madhur Behl

CS 4501
Spring 2020

Rice Hall 120

Localization and Mapping

Where am I ?

Scene Understanding

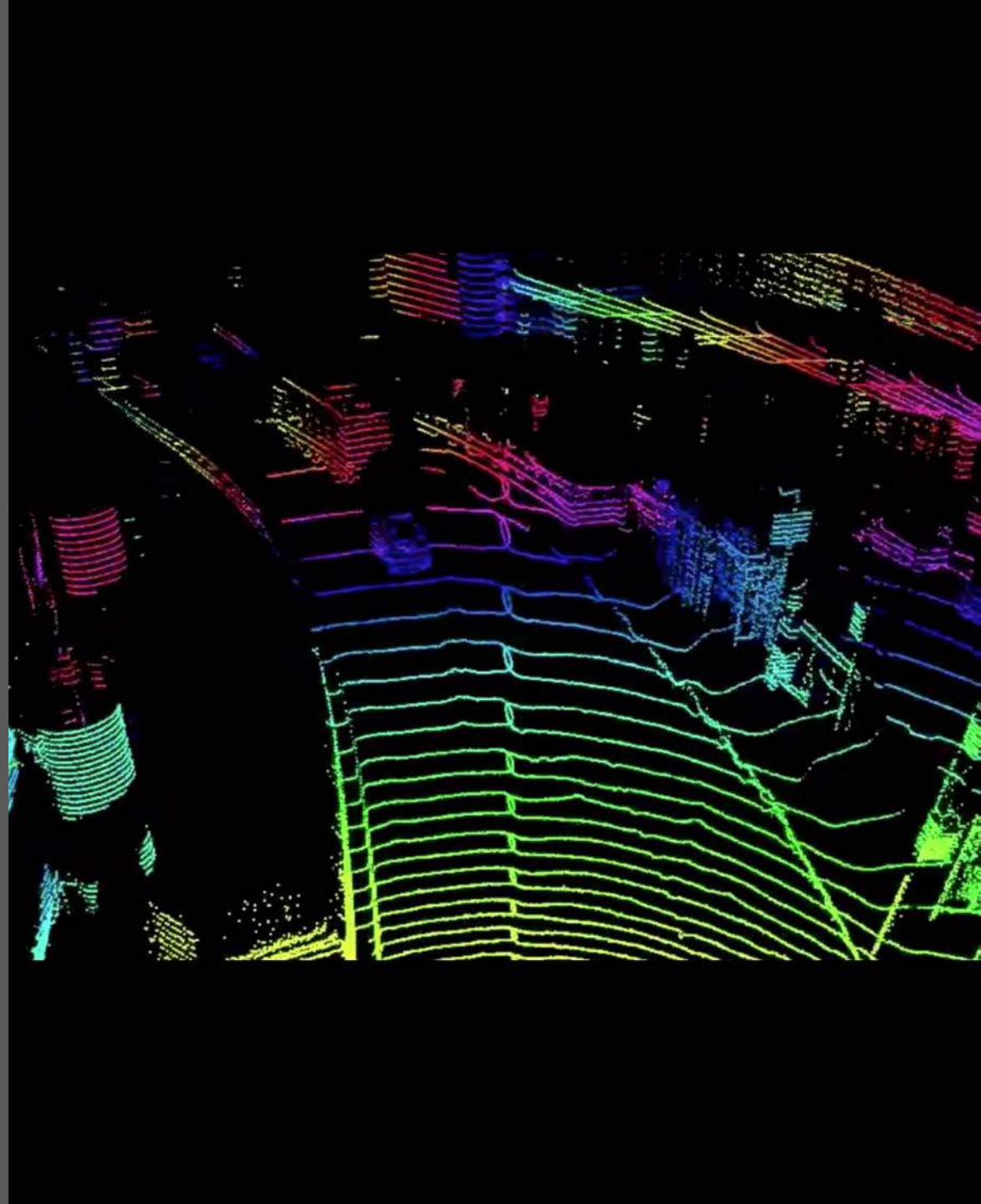
Where/who/what/why of
everyone/everything else ?

Trajectory Planning and Control

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

Human Interaction

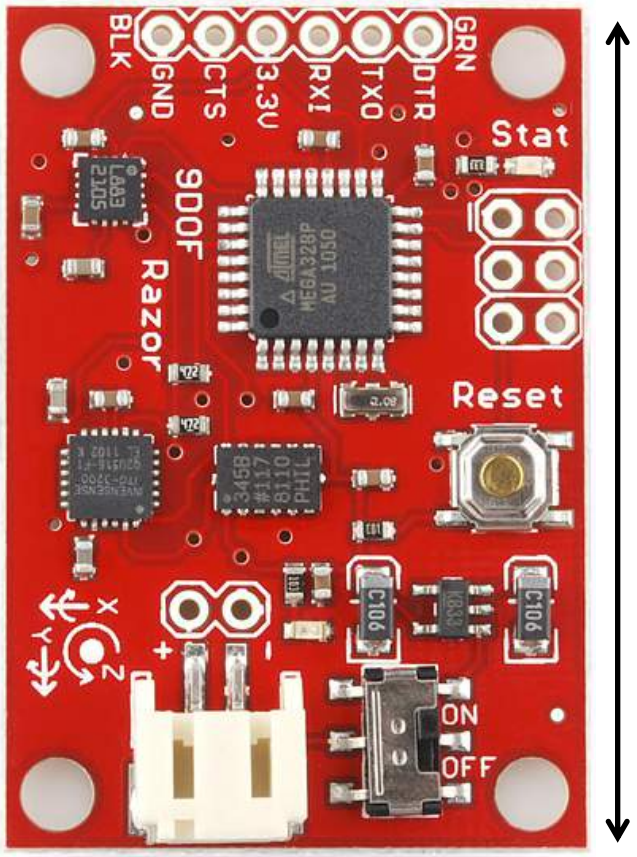
How do I convey my intent to the
passenger and everyone else ?



Perception

Sensors: specifications, placement and data

Inertial Measurement Unit



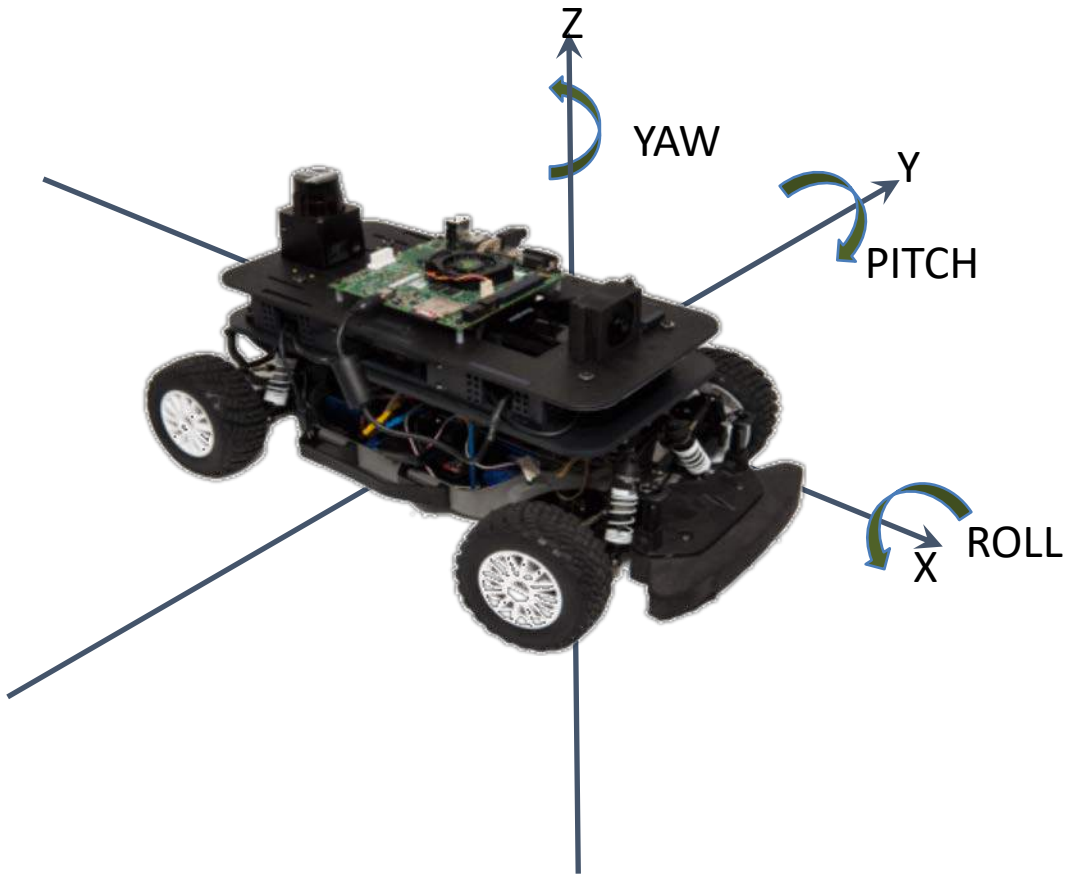
Razor IMU from Sparkfun

An **I**nertial **M**easurement **U**nit (IMU) provides acceleration, velocity and attitude measurements.

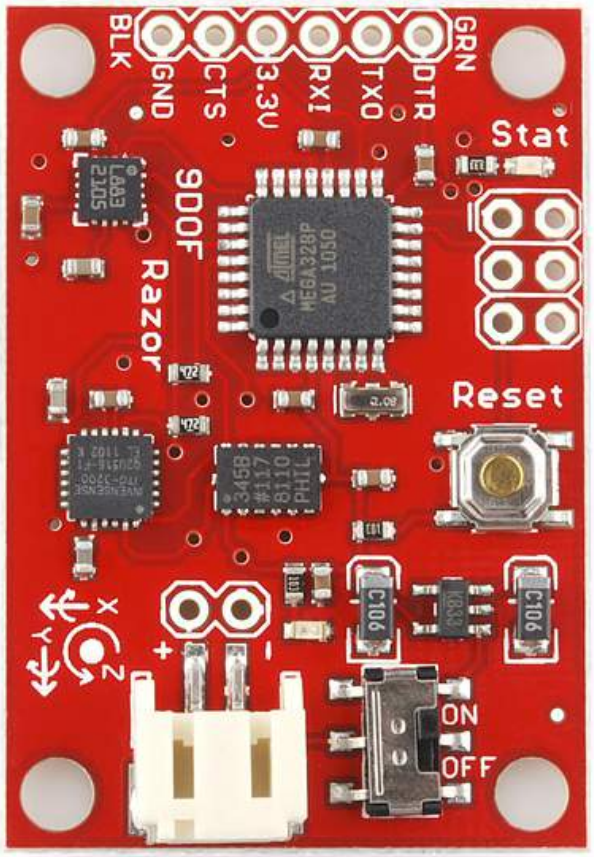
$$\dot{x} = v$$

$$\dot{v} = a$$

Inertial Measurement Unit:



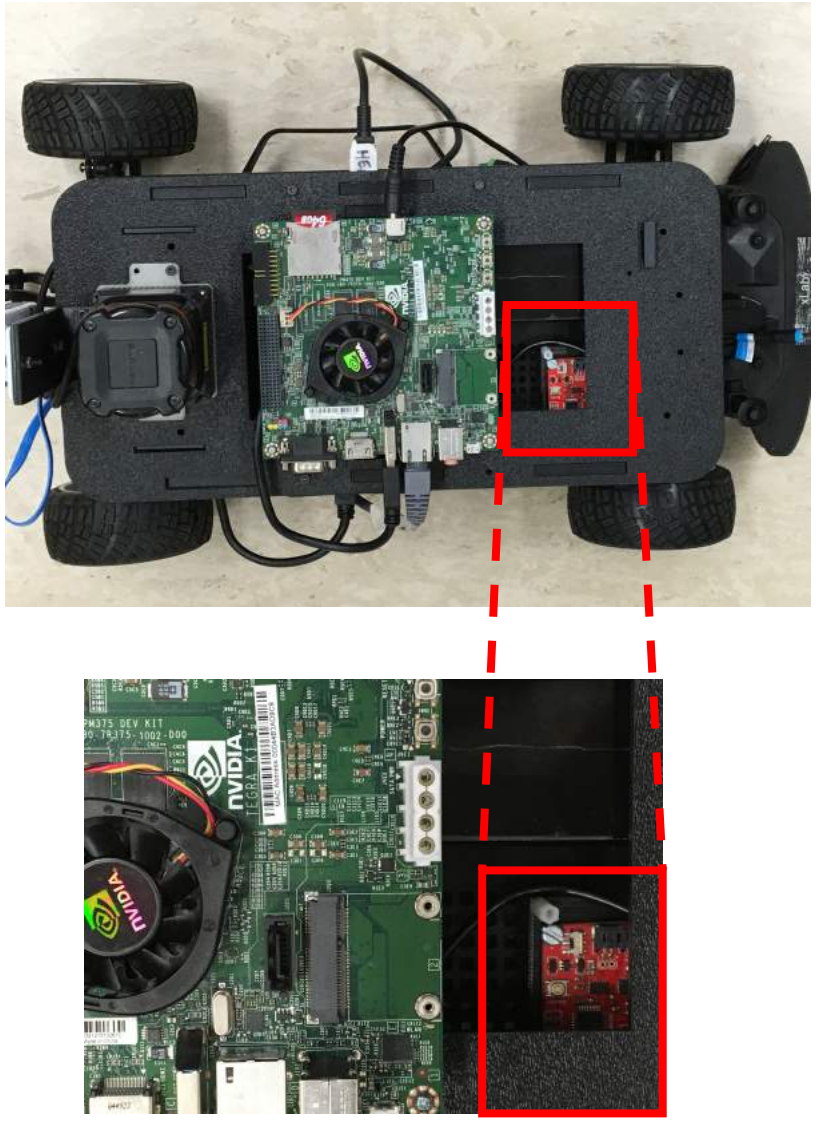
Inertial Measurement Unit



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

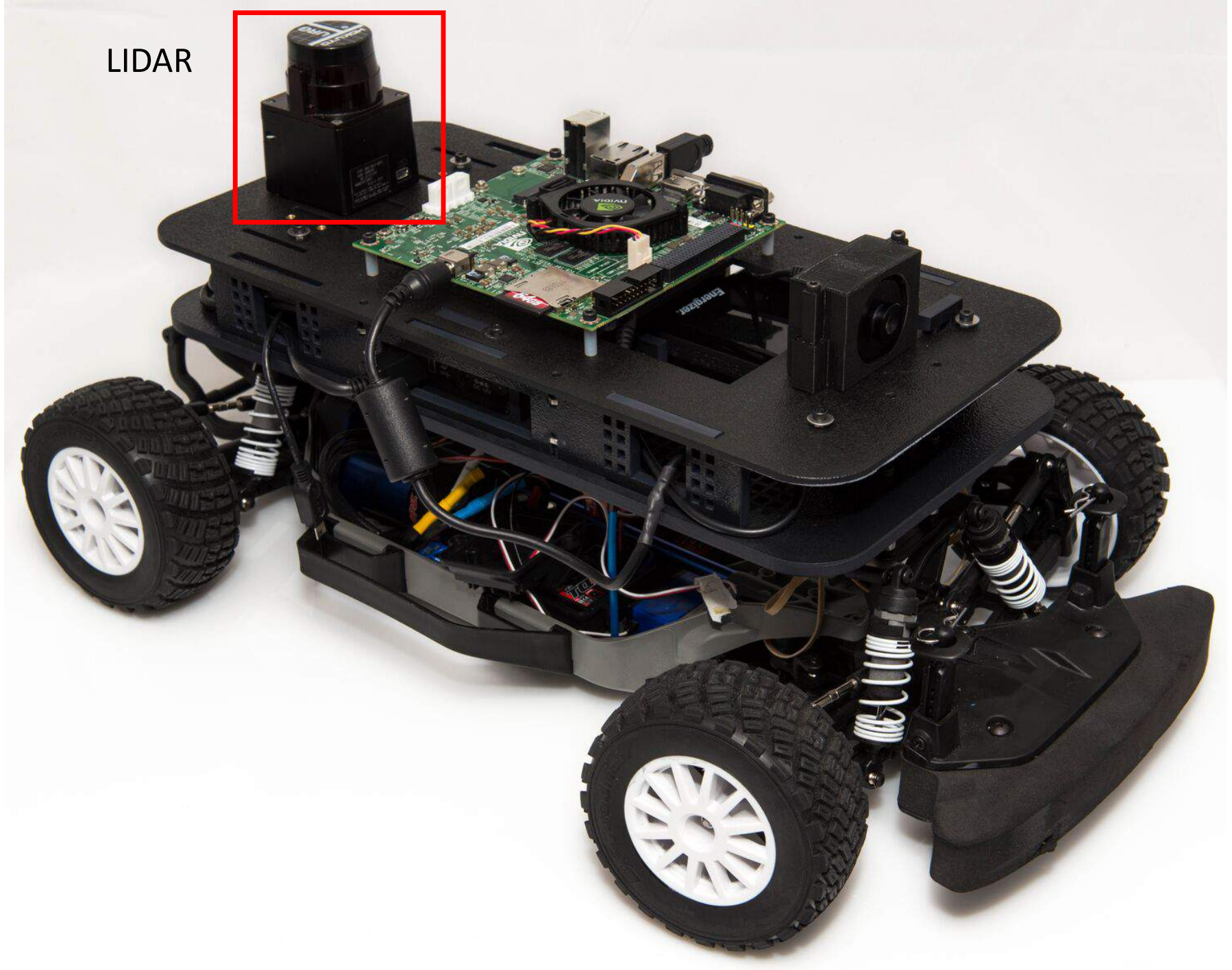
geometry_msgs/Quaternion orientation
float64[9] orientation_covariance # Row major about x, y, z axes

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

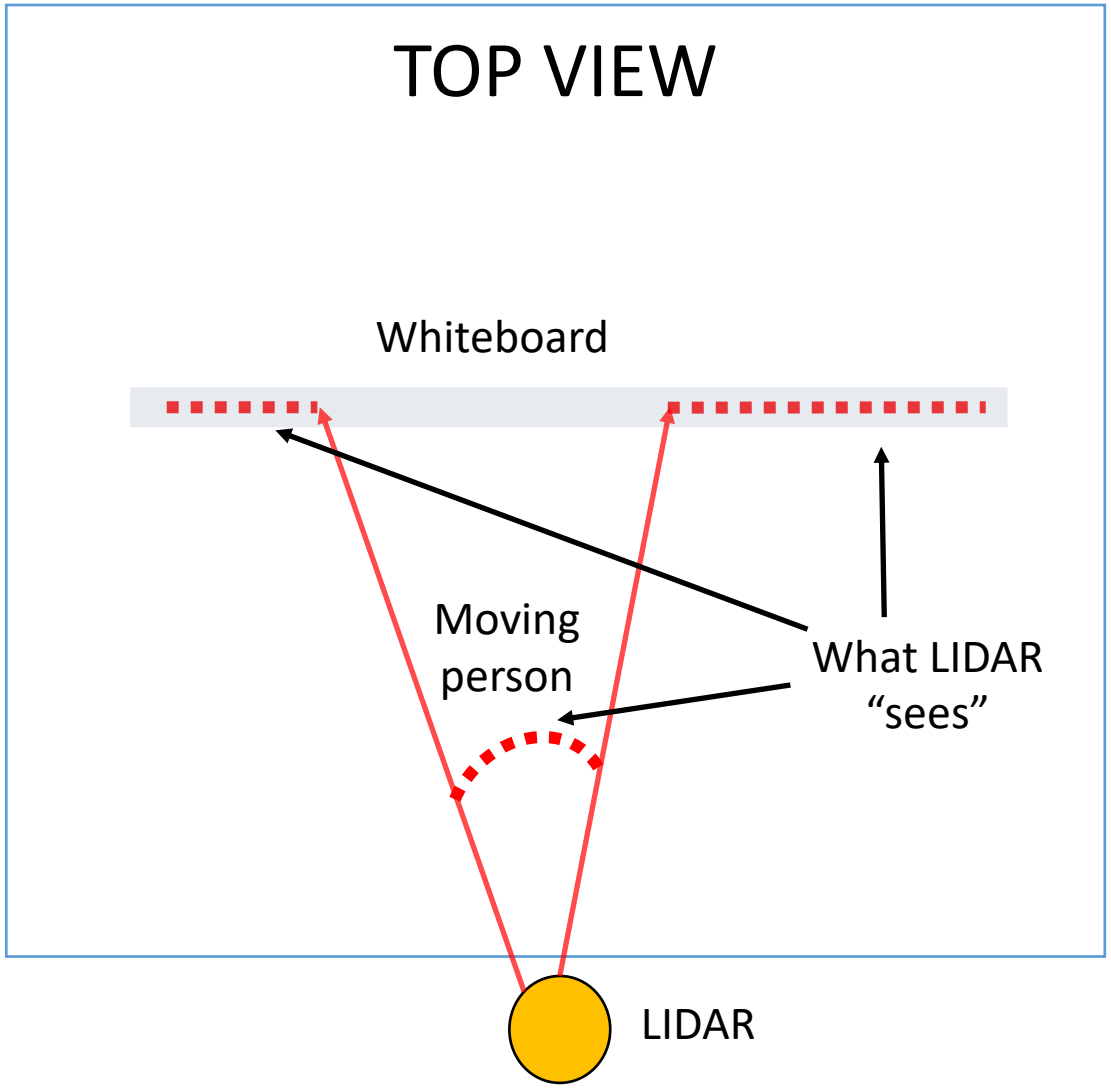
Depth information (how far are objects and obstacles around me)


LIDAR

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

Hokuyo UST-10LX

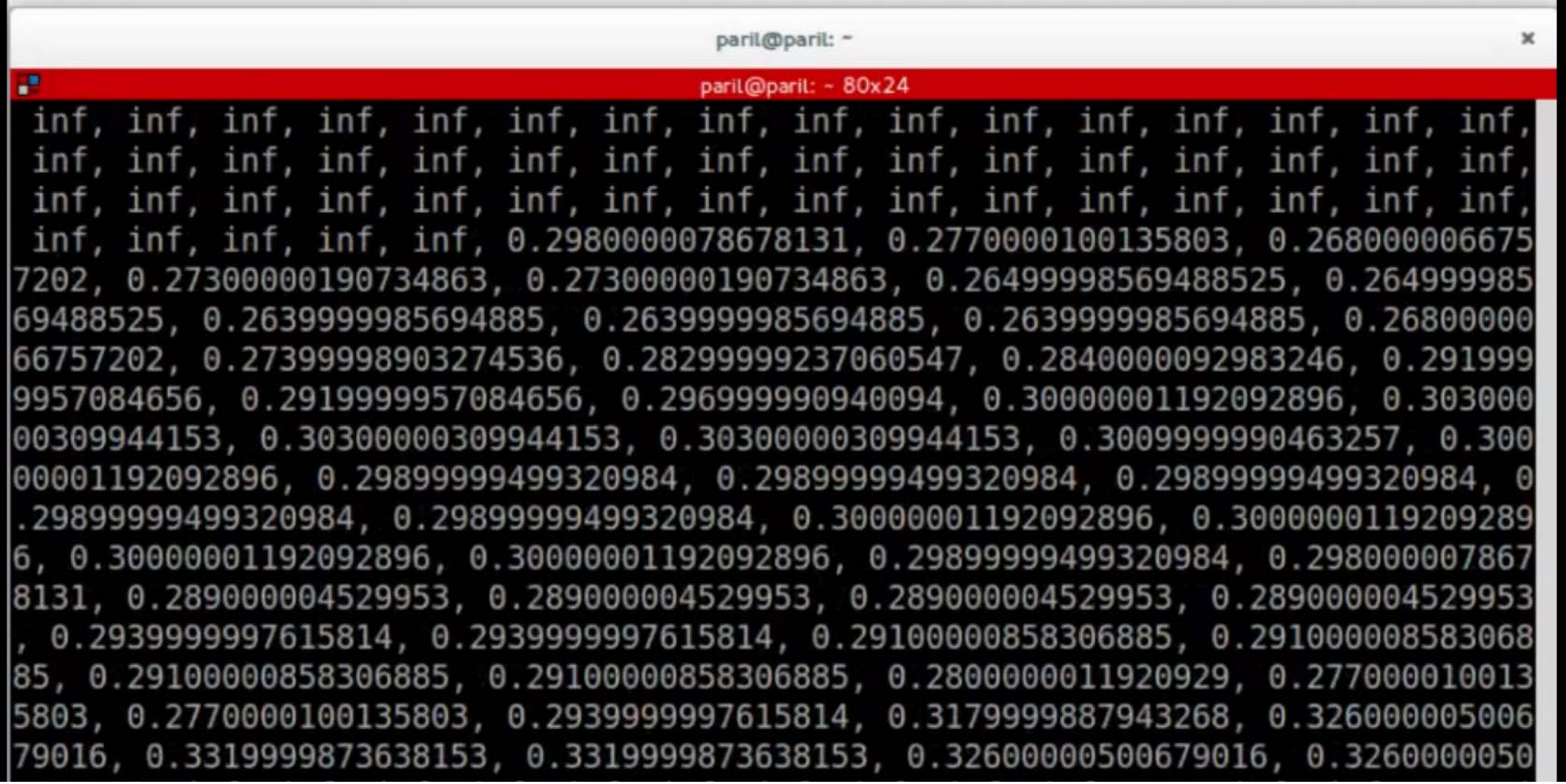
LIDAR: a simple scan sequence




Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
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].
```



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

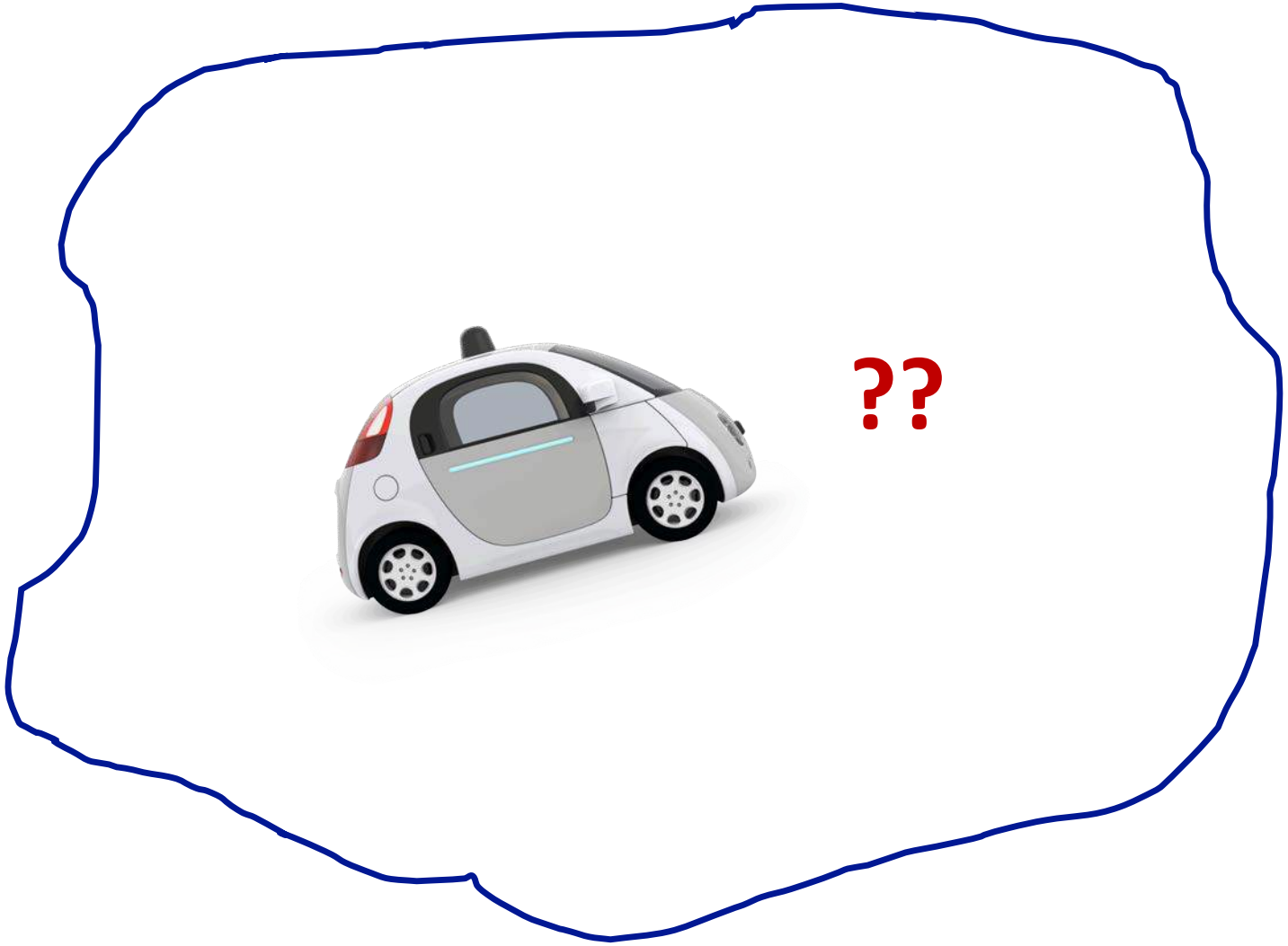
Measurements beyond the min and max range (like inf) *should be discarded.*

2D Nav Goal
Publish Point
Size (m) 0.01
Alpha 1
Decay Time 0
Position Transformer XYZ
Color Transformer Intensity
Queue Size 10
Channel Name intensity
Use rainbow
Invert Rainbow
Min Color 0; 0; 0
Max Color 255; 255; 255
Autocompute Intensit...
Size (m)
Point size in meters.
Add
Remove
Rename
Time
ROS Time: 1449818239.97
ROS Elapsed: 8.48
Wall Time: 1449818240.00
Wall Elapsed: 8.48
Experimental
Reset
30 fps

Localization

Know (where is) thyself:

Localization

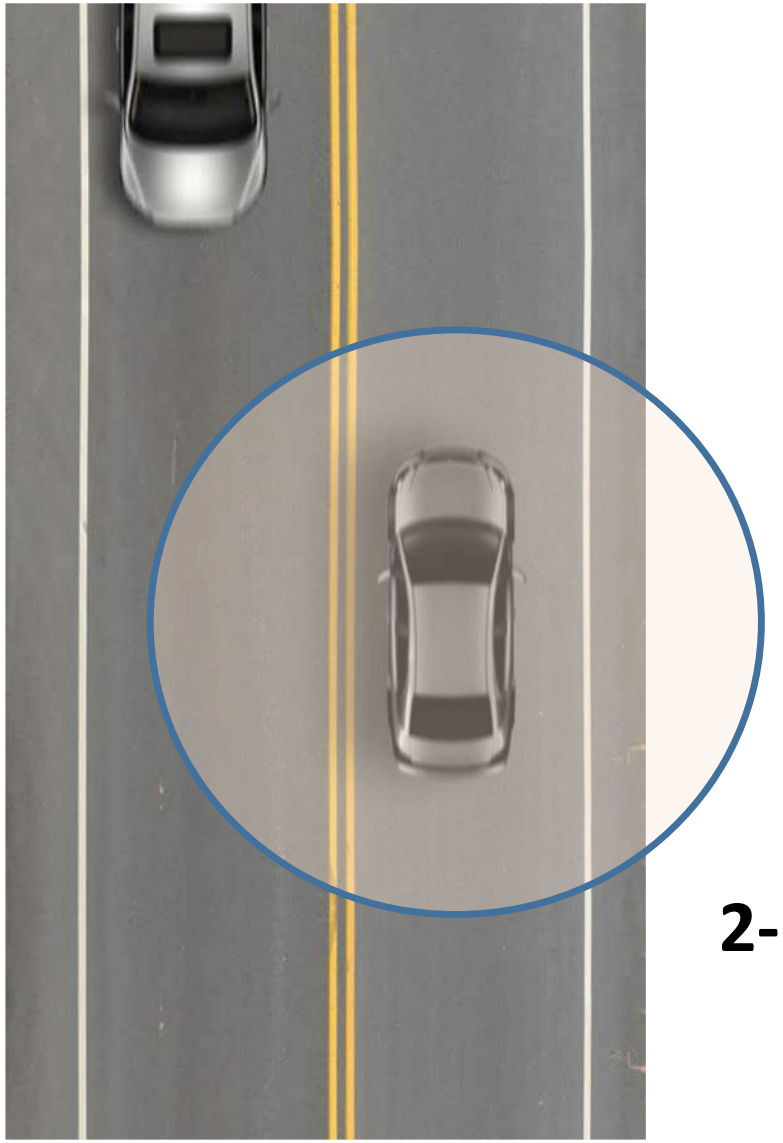


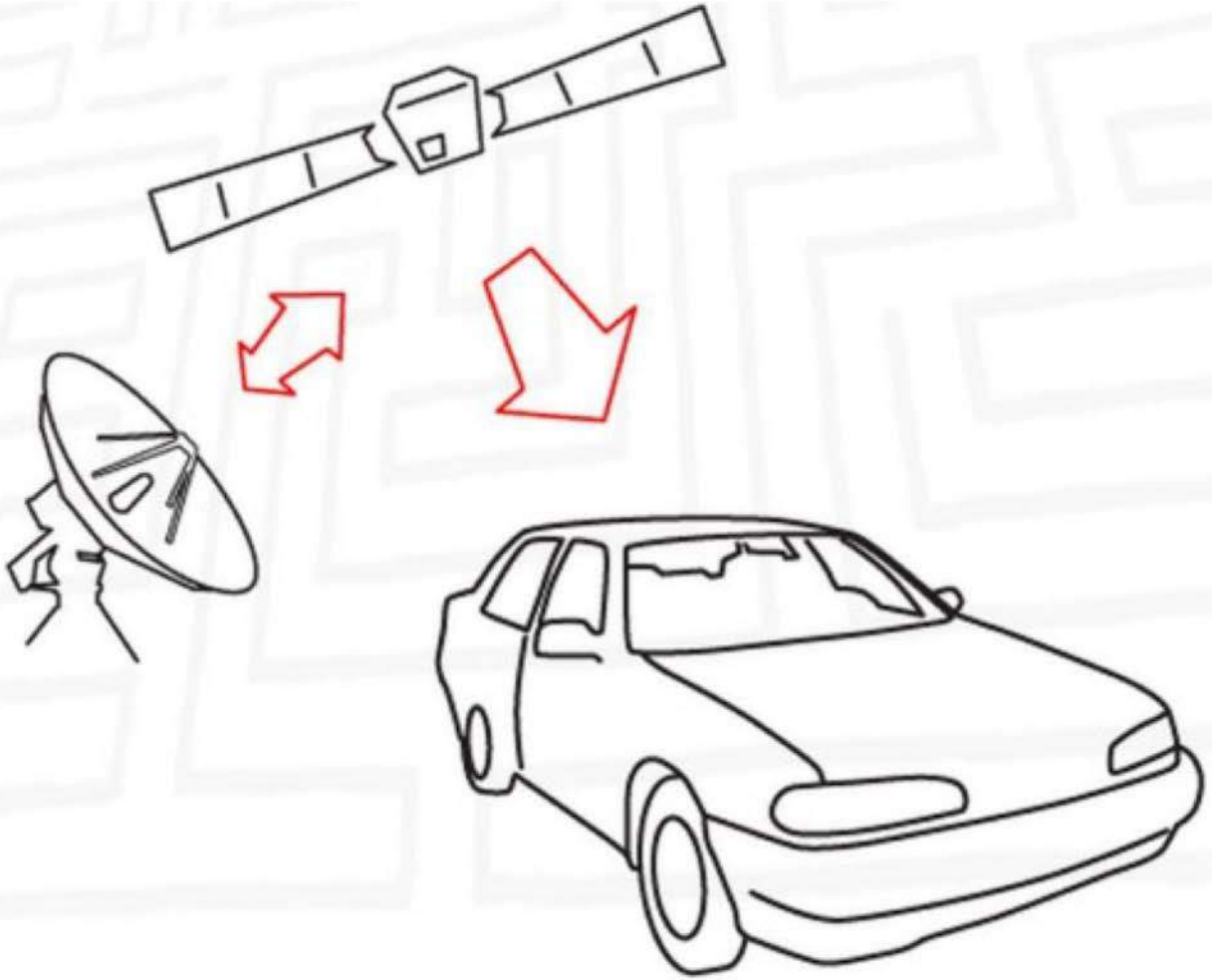
Localization

- Determine the state (position and orientation) of a robot with respect to the environment
- Localization can be within a global frame of a map, or relative to one's starting point
- Key component of all methods in the course moving forward!


GPS
Chester, Nuns Rd.
M1
New Kent Rd.
Newington Cswy.
1.9 km
11:37
02:46
178 km

Global positioning system

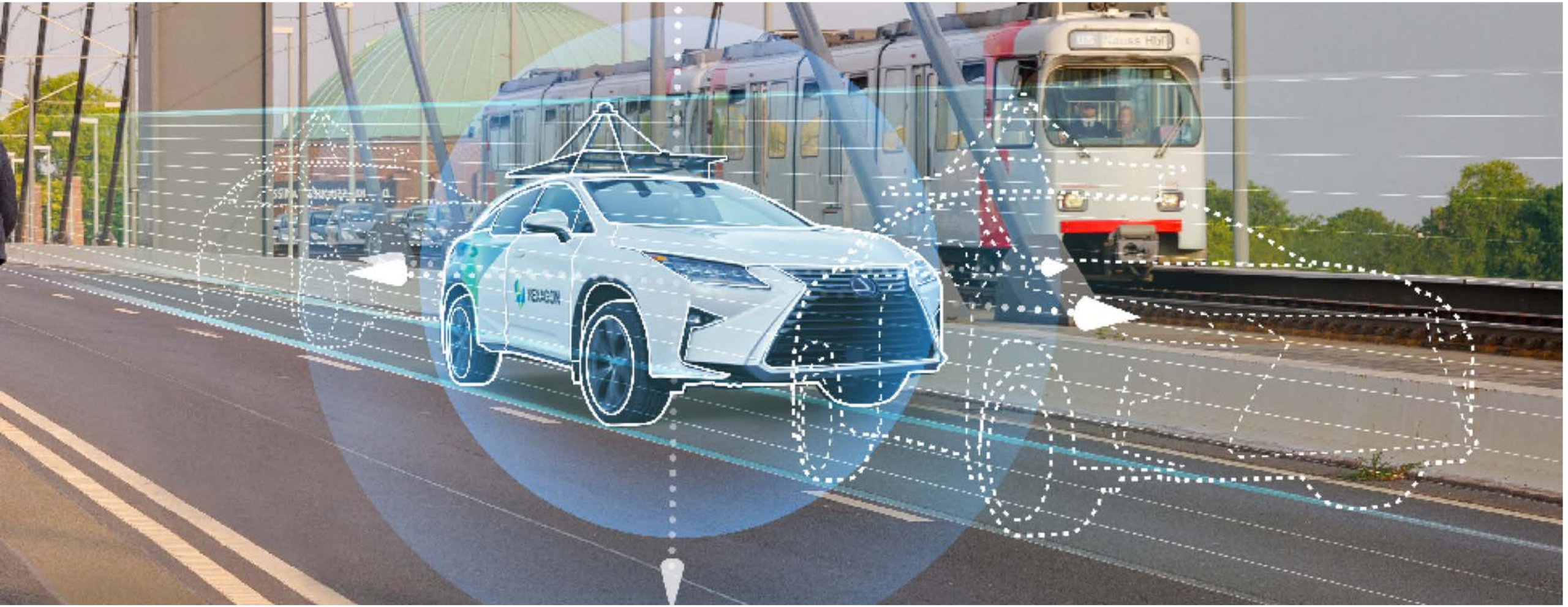




2-10m error!

Need 2-10cm error

Need decimeter level accuracy



Dead Reckoning

For Localization

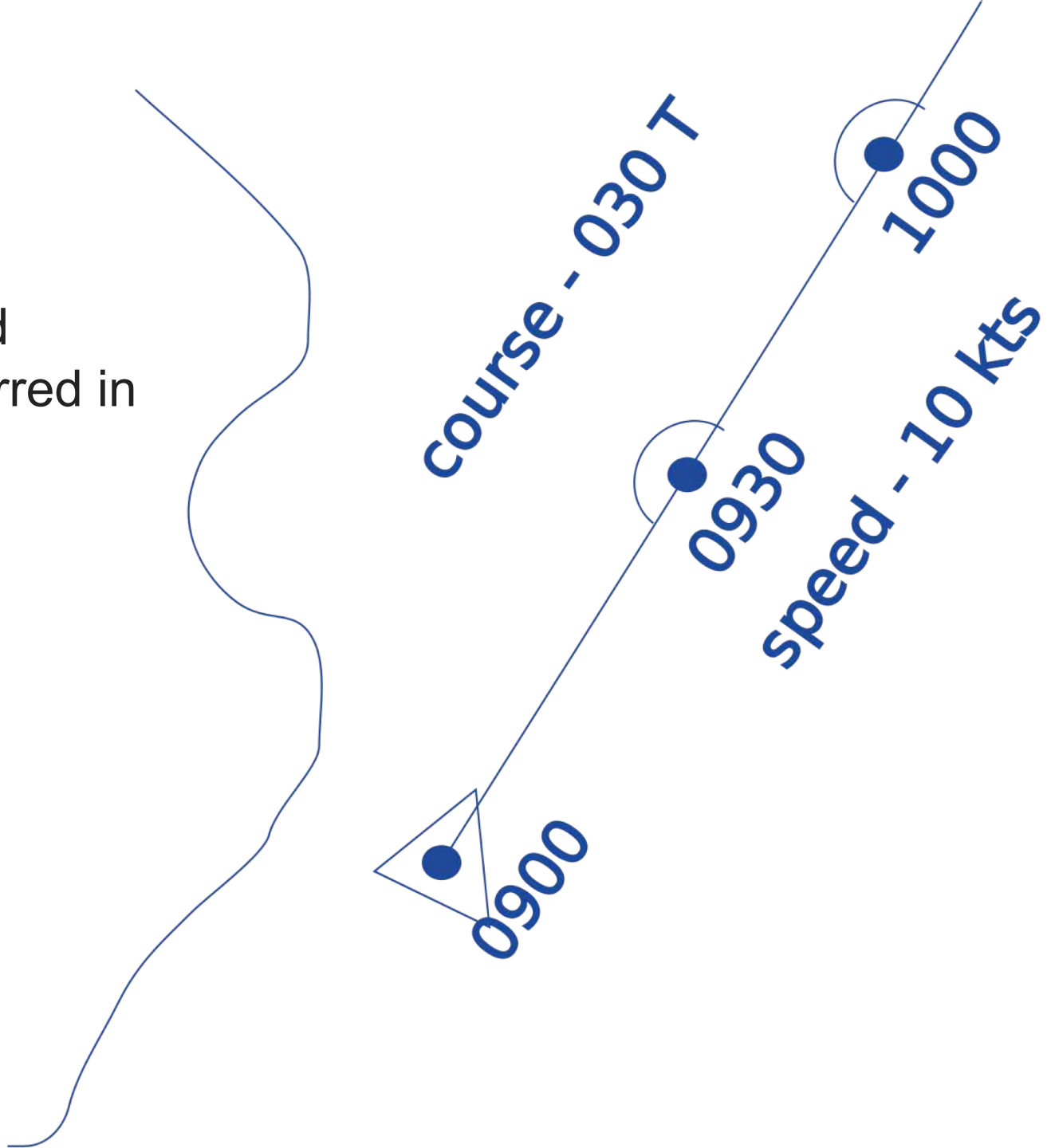
dead reckoning

estimating the value of any variable quantity by using an earlier value and adding whatever changes have occurred in the meantime

"deduced reckoning,"

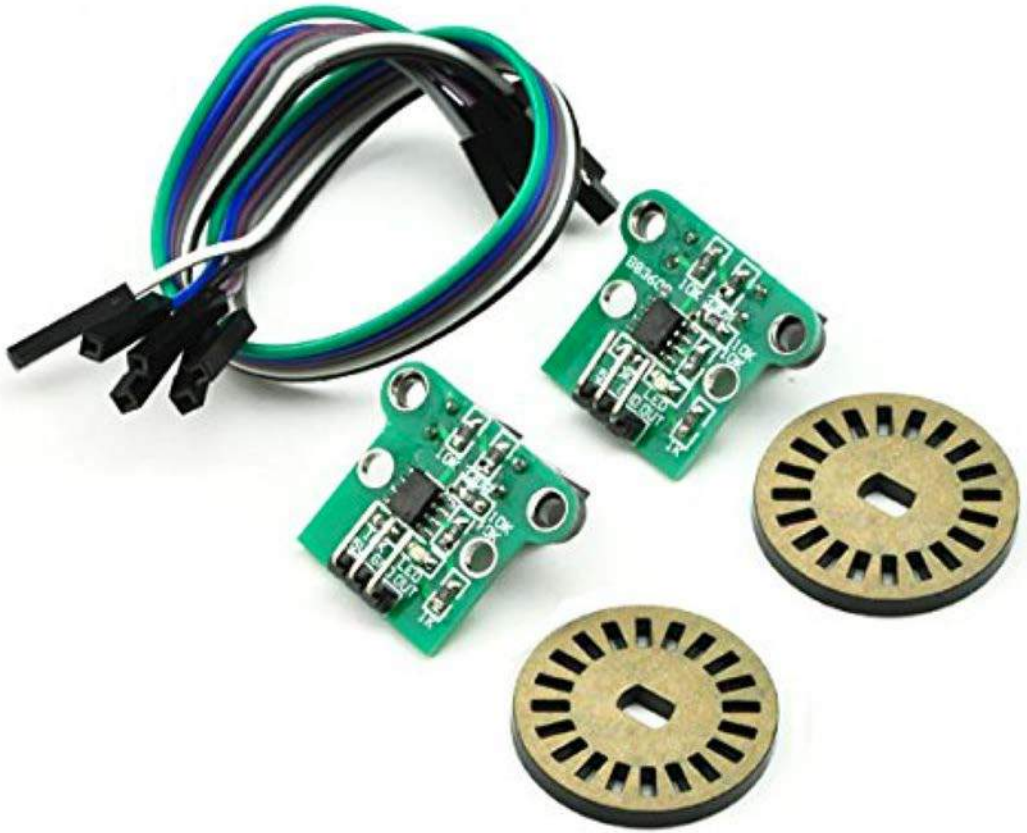
"dead ahead," reckoning

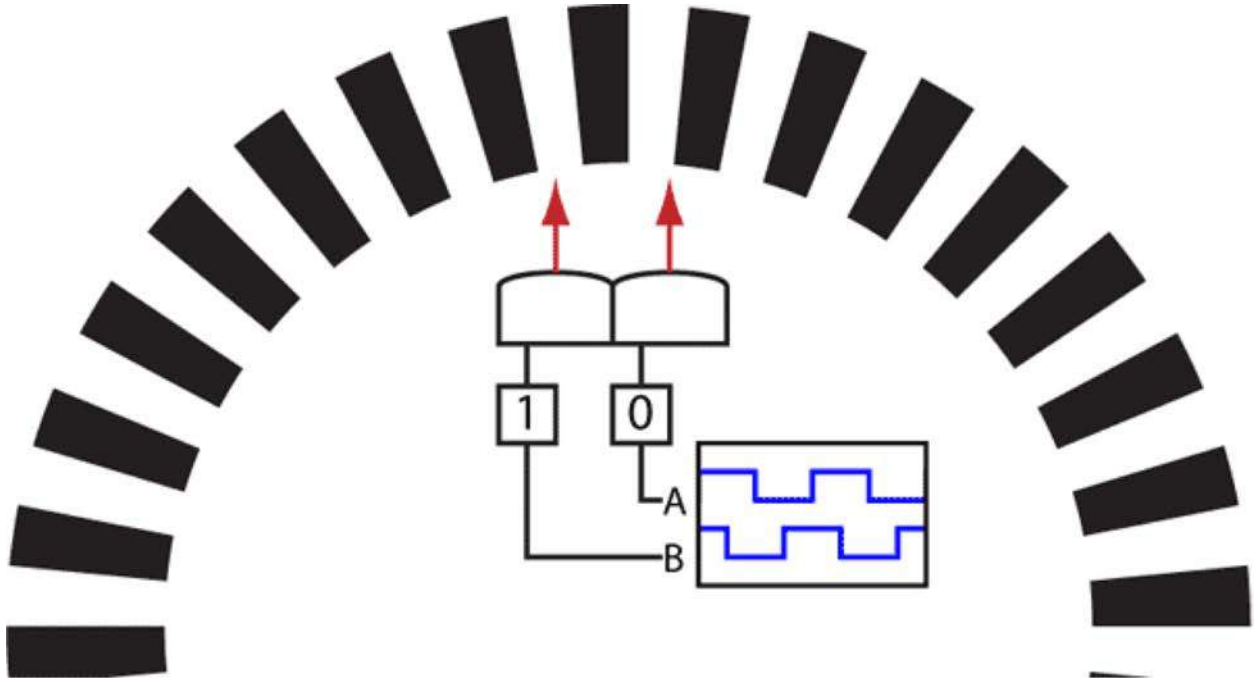
word “dead”: “of water, ‘still, standing,



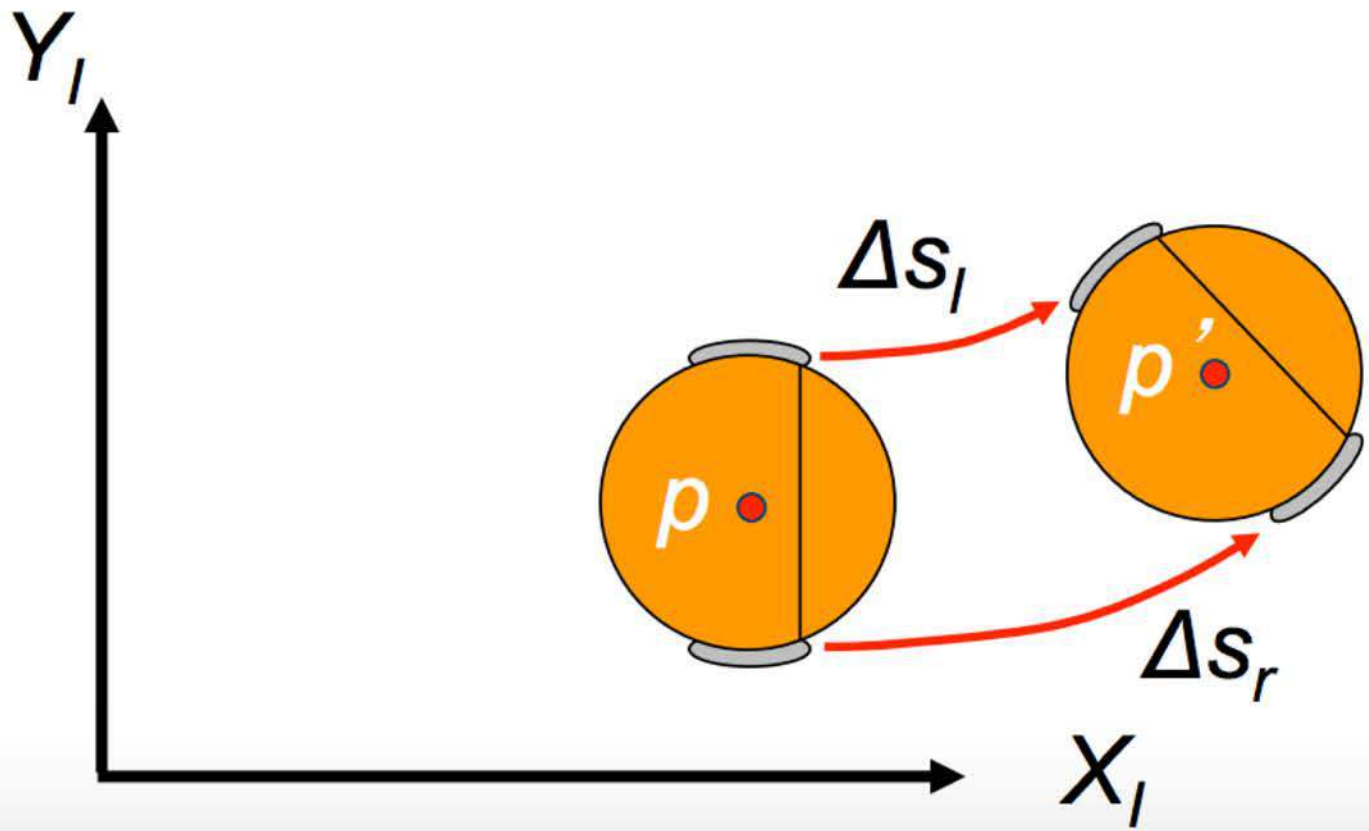
Odometry -Wheel Encoders

Odometry -Wheel Encoders

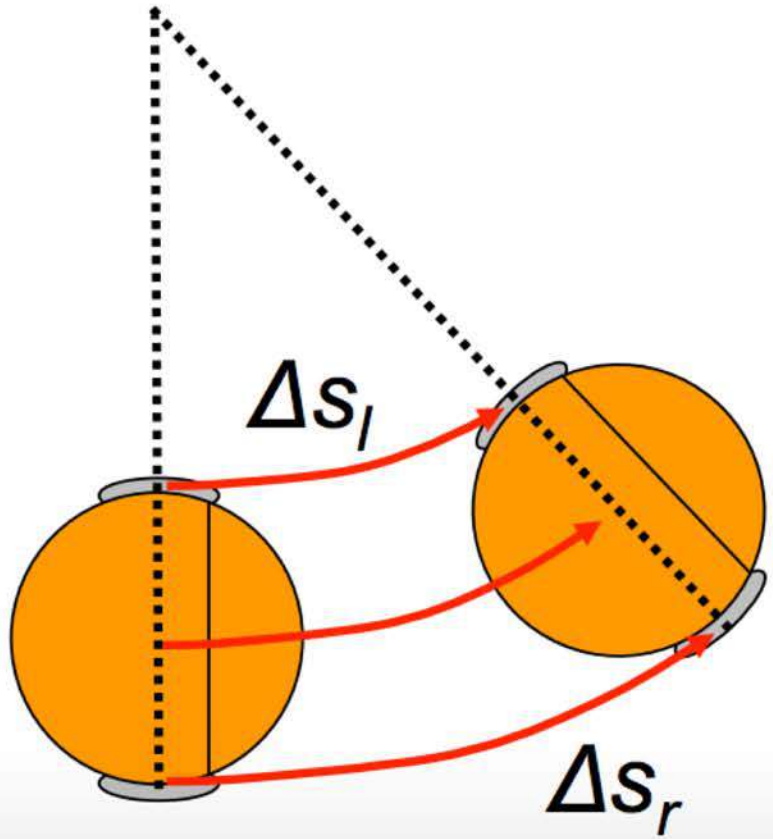




- If a robot starts from a position p, and the right and left wheels move respective distances Δs_r and Δs_l, what is the resulting new position p' ?

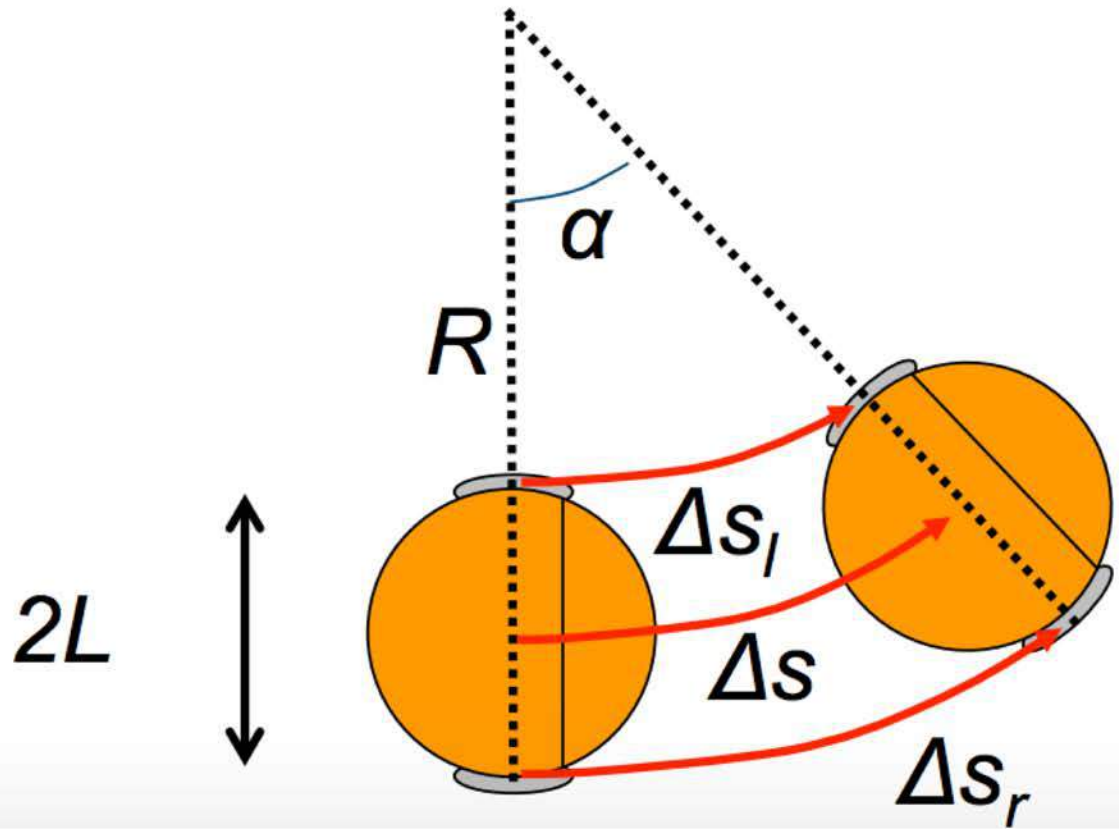


- To start, let's model the change in angle $\Delta\theta$ and distance travelled Δs by the robot.
  - Assume the robot is travelling on a circular arc of constant radius.



- Begin by noting the following holds for circular arcs:

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



- Now manipulate first two equations:

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

To:

$$R\alpha = \Delta s_l$$

$$L\alpha = (\Delta s_r - R\alpha)/2$$

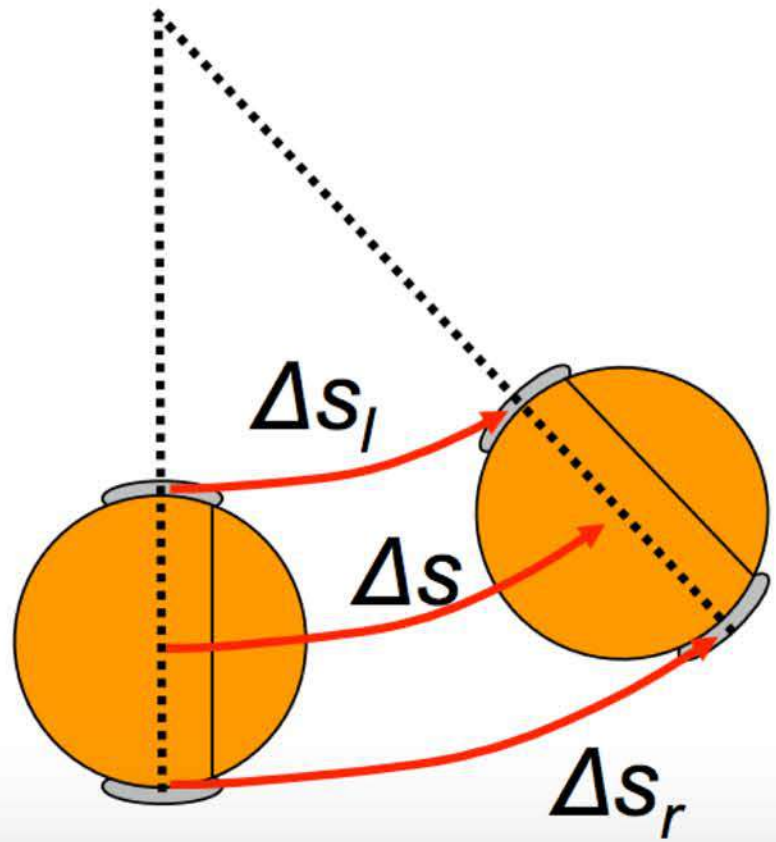
$$= \Delta s_r/2 - \Delta s_l/2$$

- Substitute this into last equation for Δs:

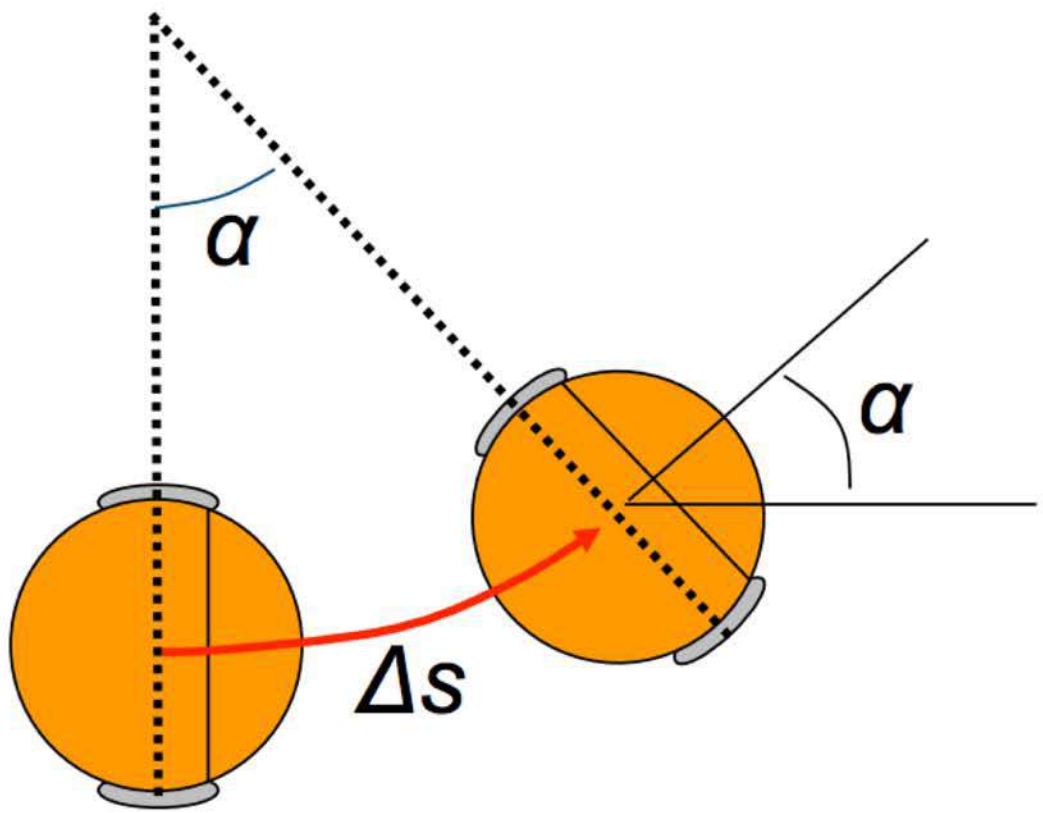
$$\begin{aligned}
\Delta s &= (R+L)\alpha \\
&= R\,\alpha + L\alpha \\
&= \Delta s_l + \Delta s_r/2 - \Delta s_l/2 \\
&= \Delta s_l/2 + \Delta s_r/2 \\
&= \frac{\Delta s_l + \Delta s_r}{2}
\end{aligned}$$

- 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



$$\Delta\theta = \alpha$$

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

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

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

$$\alpha = \Delta s_l / R$$

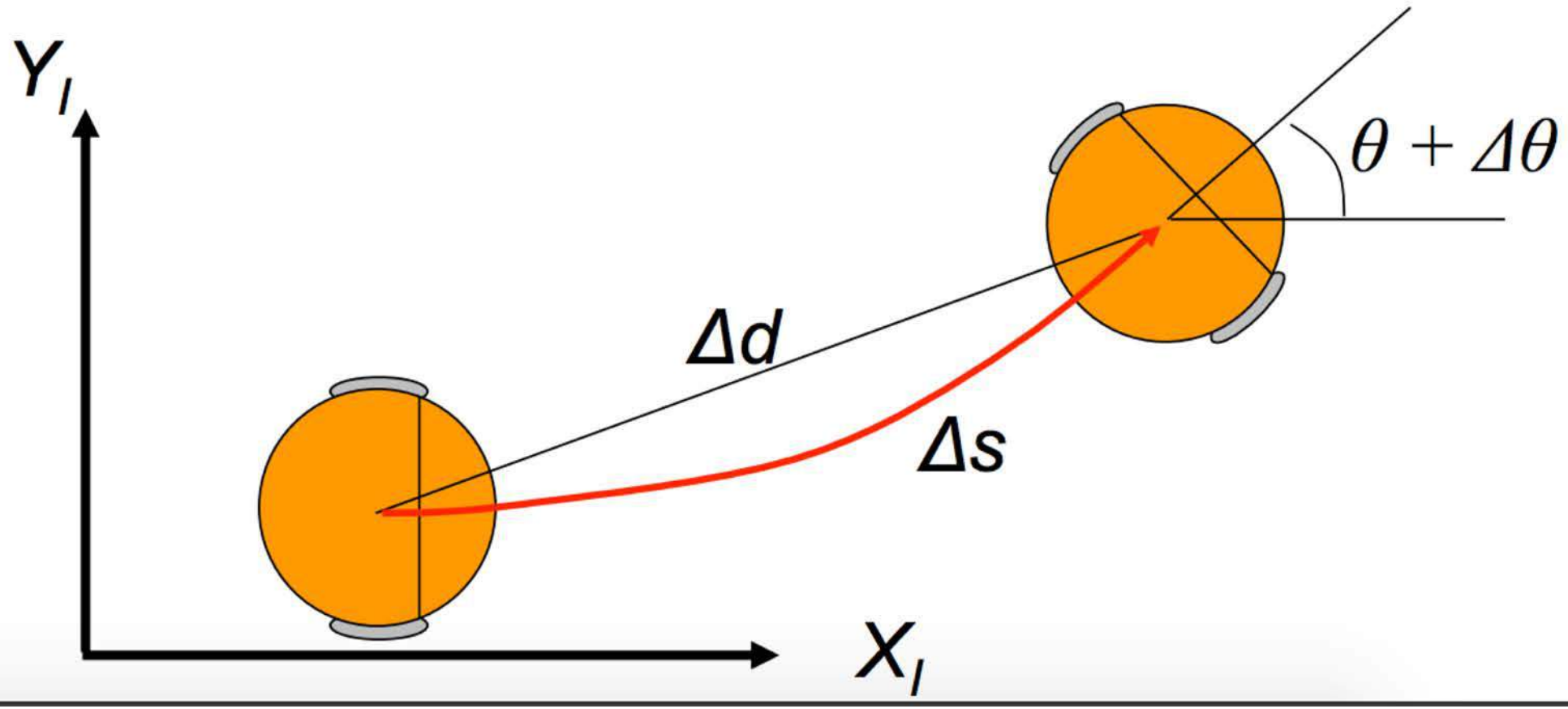
$$= \Delta s_l (\Delta s_r - \Delta s_l) / (2L \Delta s_l)$$

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

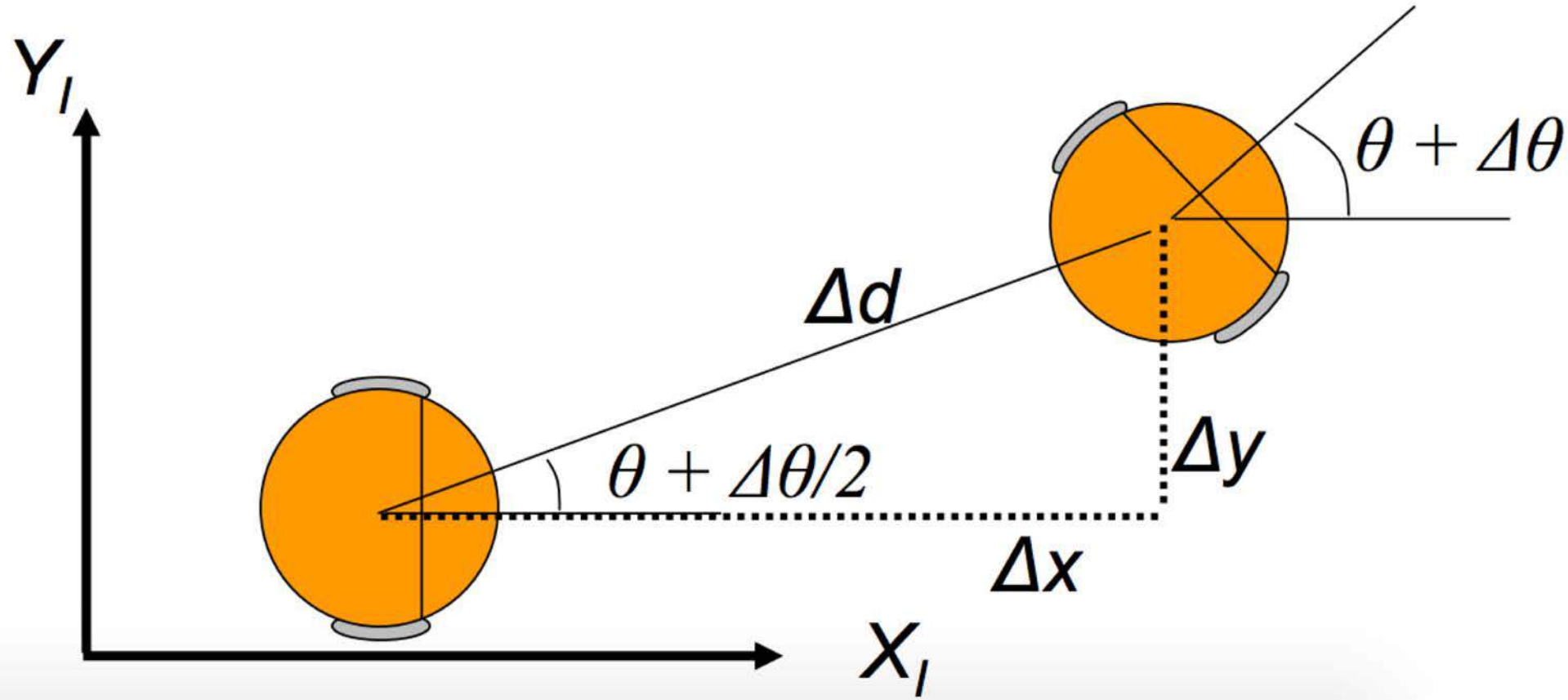
So…

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

- 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.



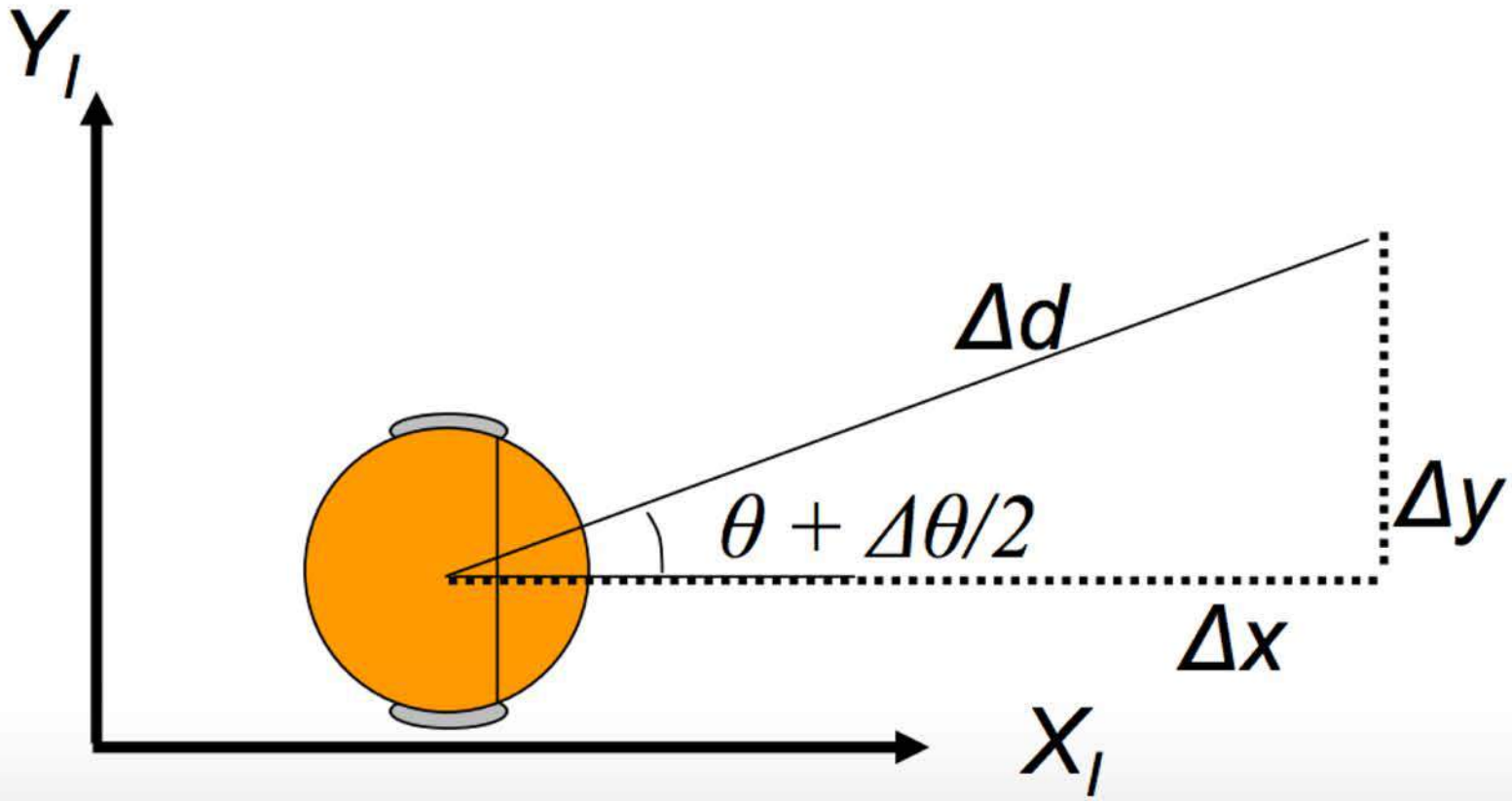
- Now calculate the change in position as a function of Δd.



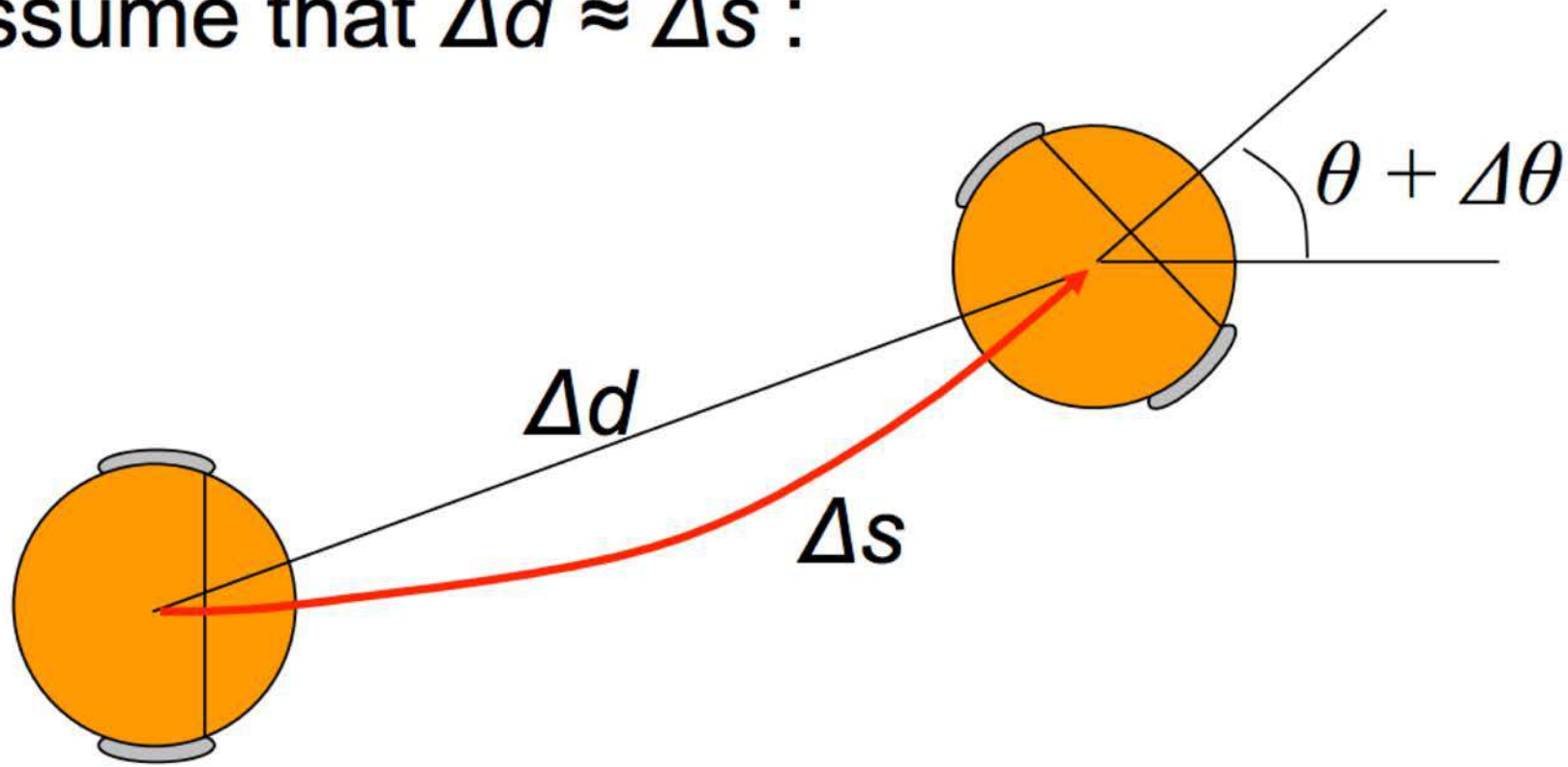
- Using Trig:

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

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



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

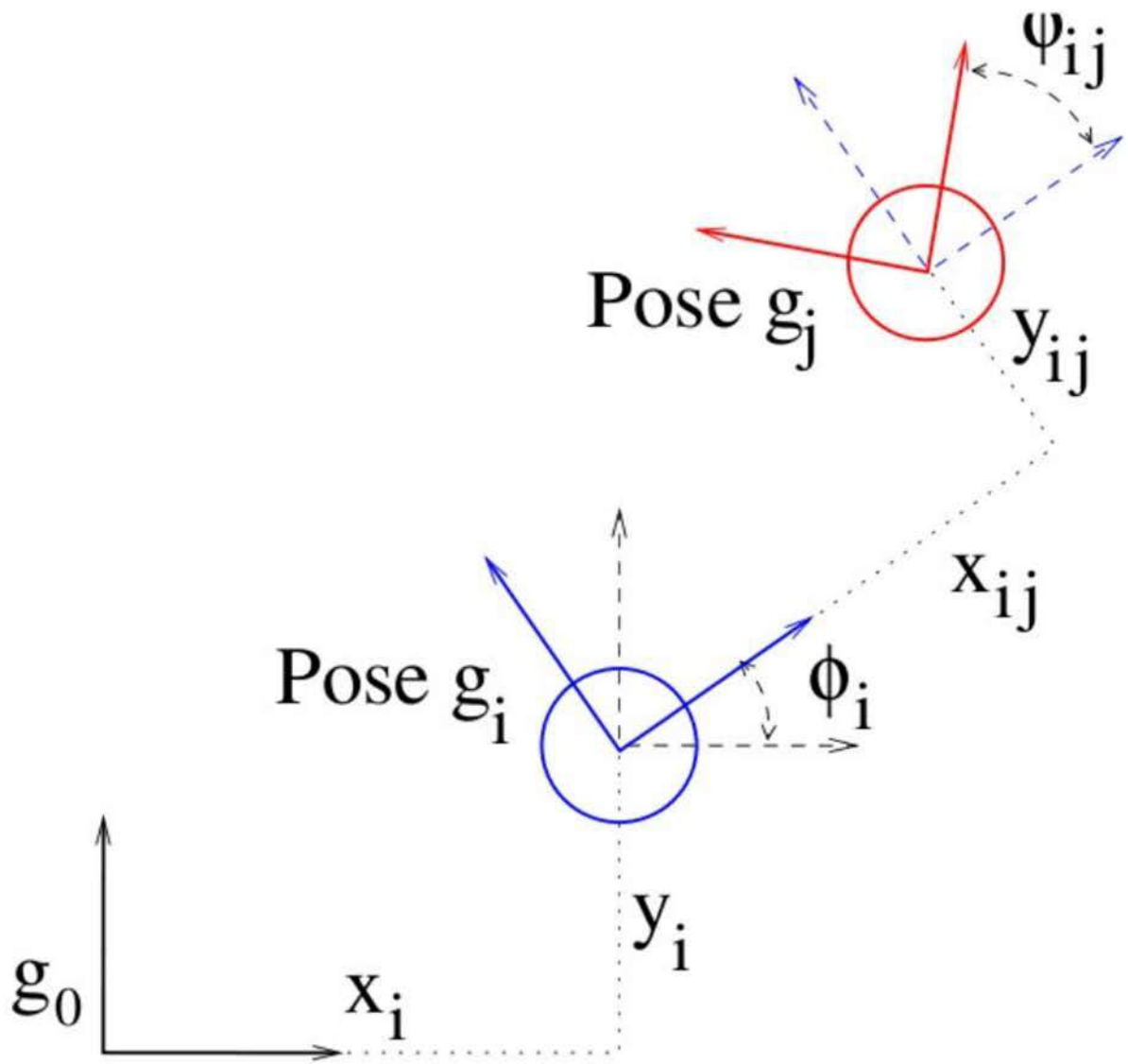


- So…

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

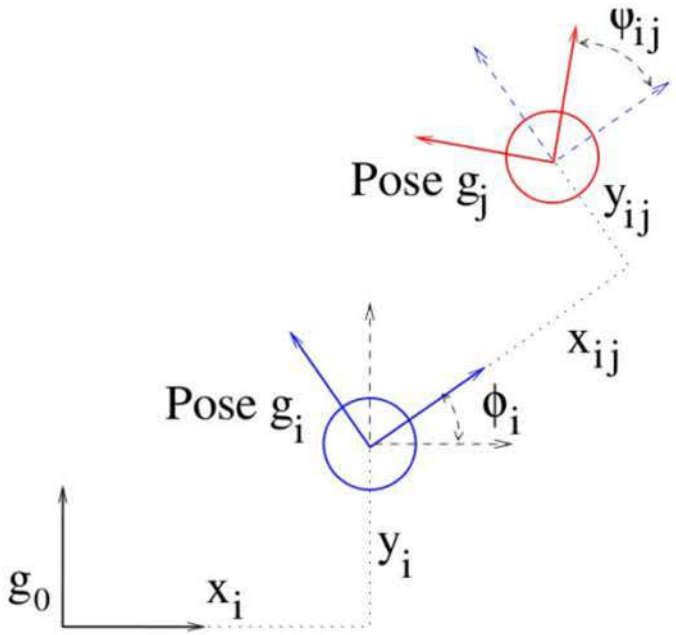
$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

Dead-reckoning



$$g_i = \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix}$$

In more general form

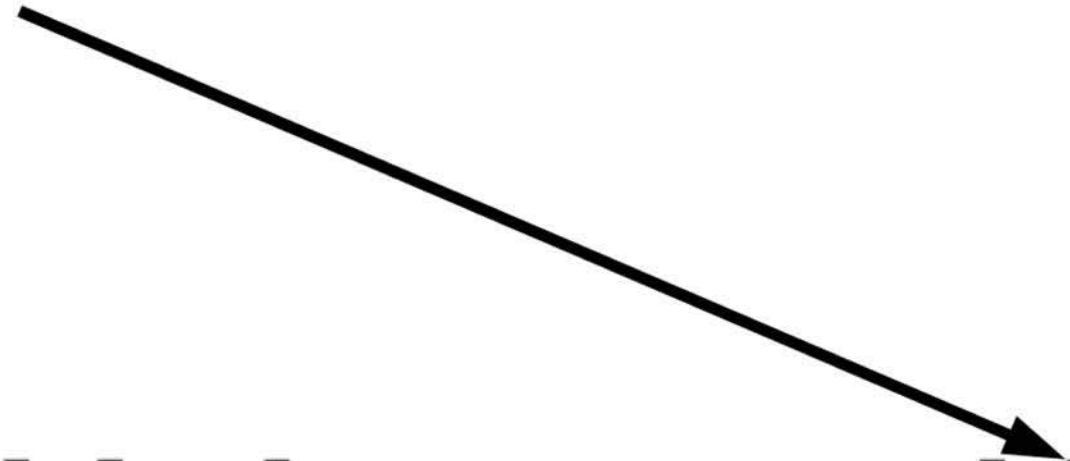


$$g_{ij} = g_i^{-1} g_j = \begin{bmatrix} x_{ij} \\ y_{ij} \\ \phi_{ij} \end{bmatrix}$$

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

Noise/Errors

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

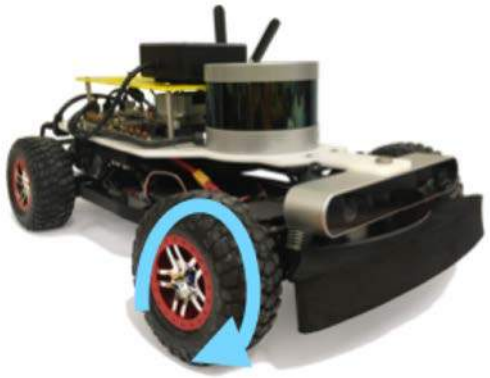


$$g_j = \begin{bmatrix} x_j \\ y_j \\ \phi_j \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix} + \begin{bmatrix} \cos(\phi_i) & -\sin(\phi_i) & 0 \\ \sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ij} \\ y_{ij} \\ \phi_{ij} \end{bmatrix}.$$

Motion Integration

Odometry: Start at known pose and integrate control and motion measurements to estimate the current pose

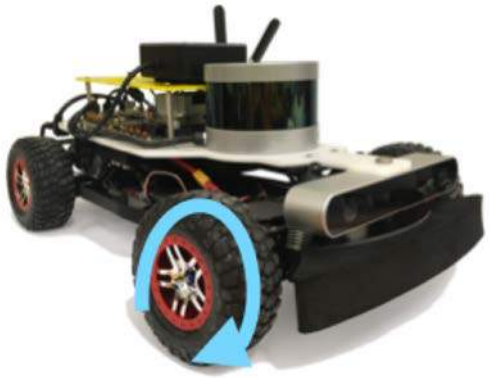
- Integrate dynamics using information from VESC, wheel encoders, IMU, etc.

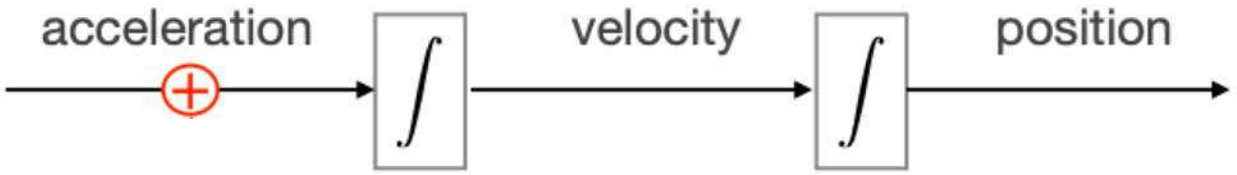


Motion Integration

Odometry: Start at known pose and integrate control and motion measurements to estimate the current pose

- Integrate dynamics using information from VESC, wheel encoders, IMU, etc.



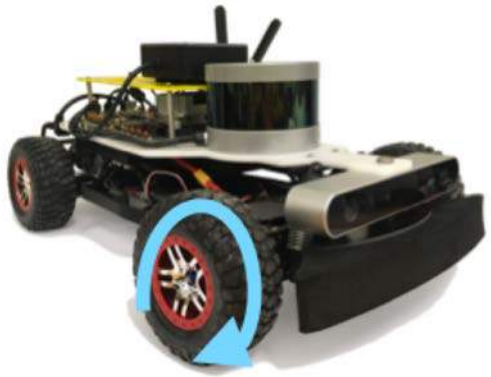


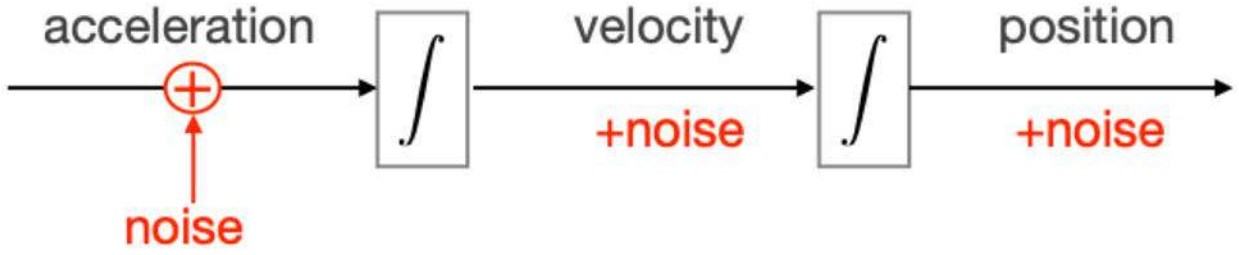
(credit: Luca Carlone, MIT 6.141/16.405)

Motion Integration

Odometry: Start at known pose and integrate control and motion measurements to estimate the current pose

- Integrate dynamics using information from VESC, wheel encoders, IMU, etc.





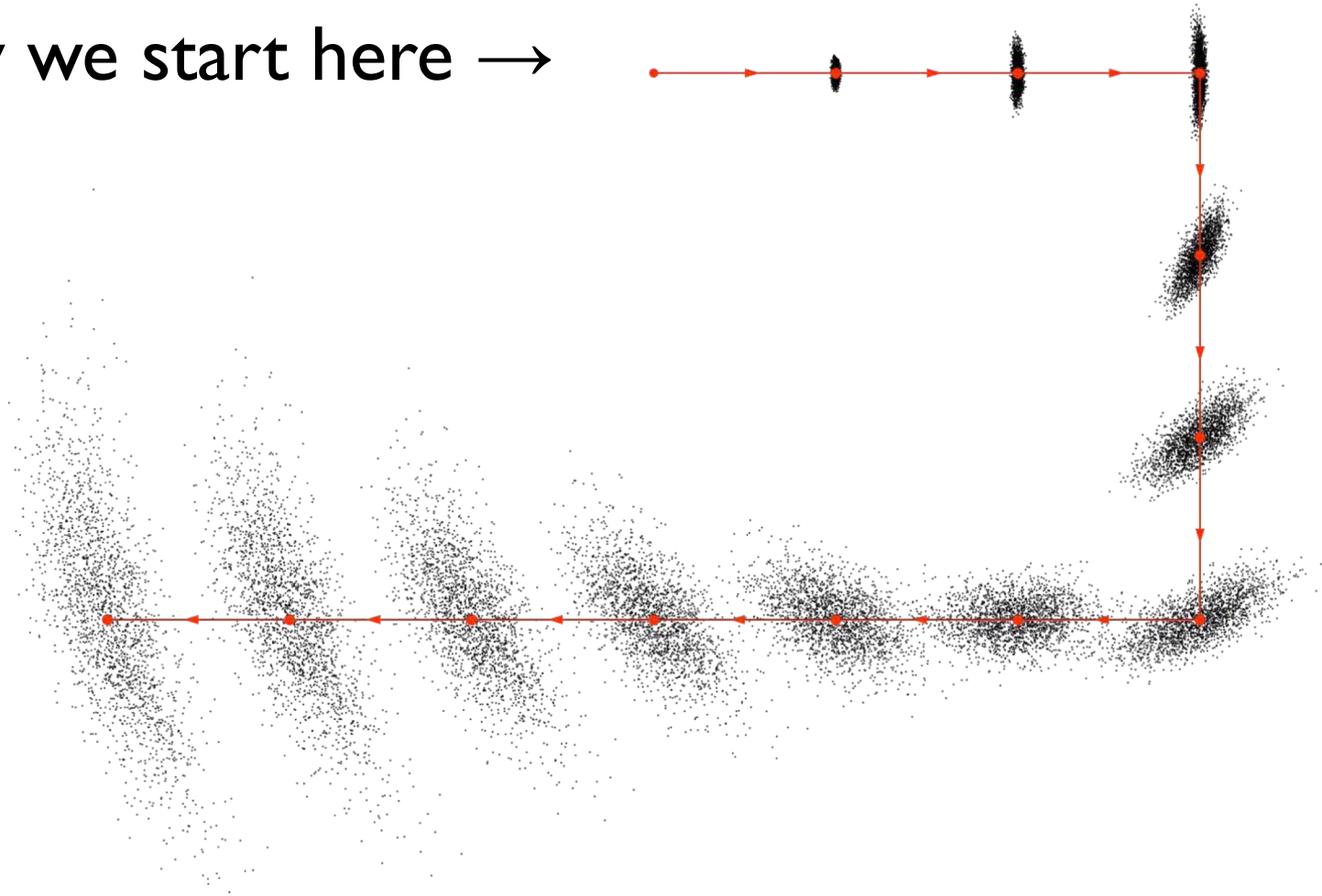
- Odometry = open loop estimation = error increases over time

(credit: Luca Carlone, MIT 6.141/16.405)

Motion Integration

Odometry = open loop estimation = error increases over time

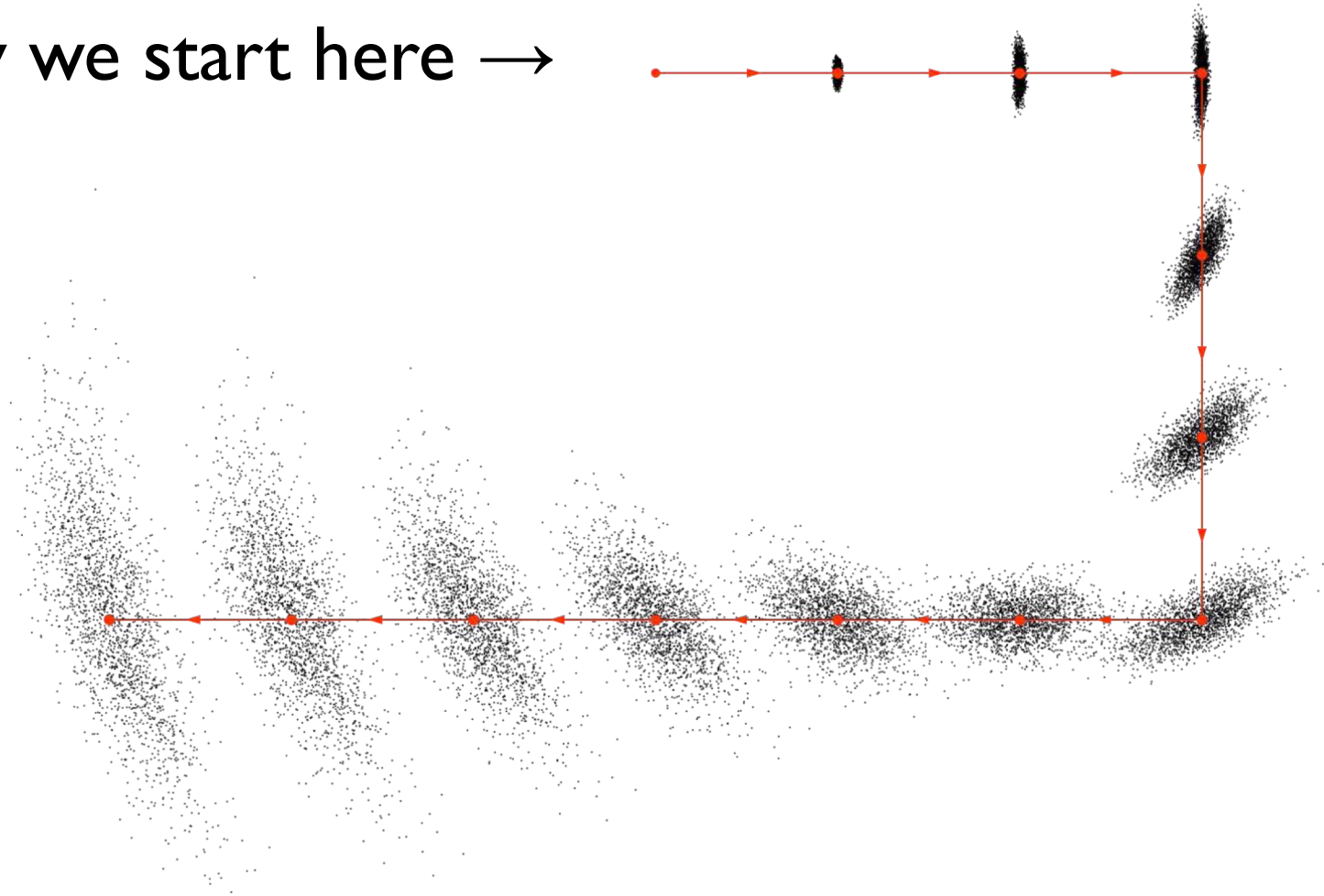
Let's say we start here →



Motion Integration

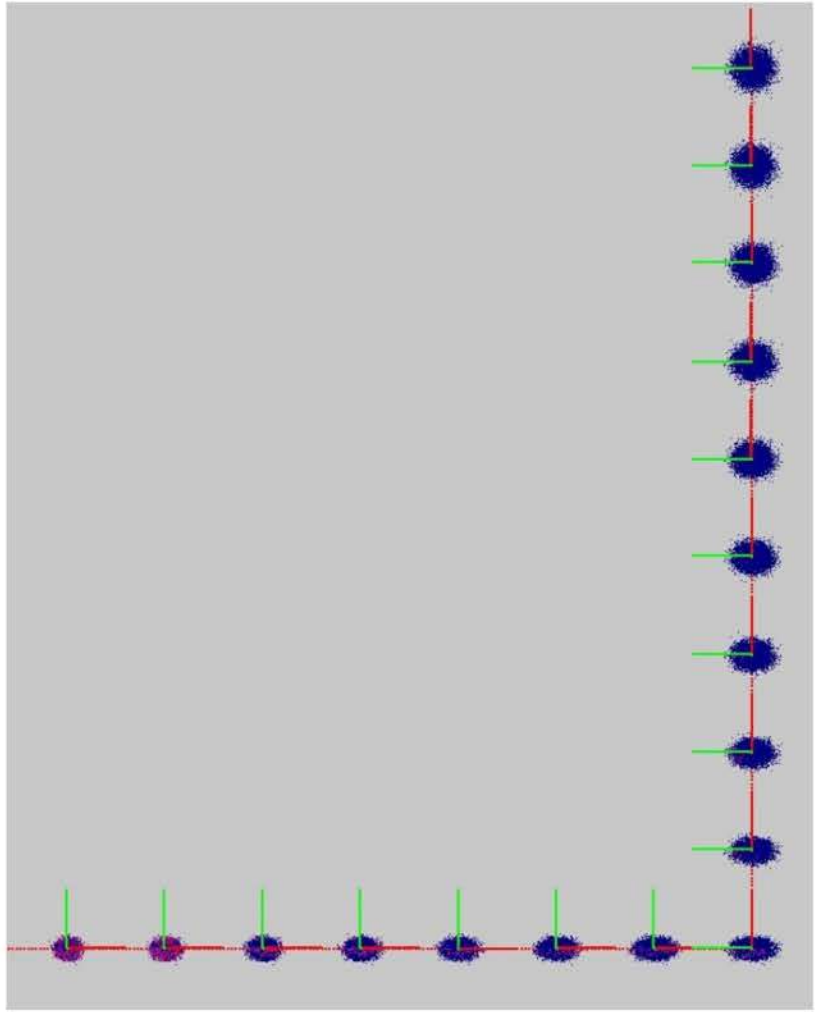
Odometry = open loop estimation = error increases over time

Let's say we start here →

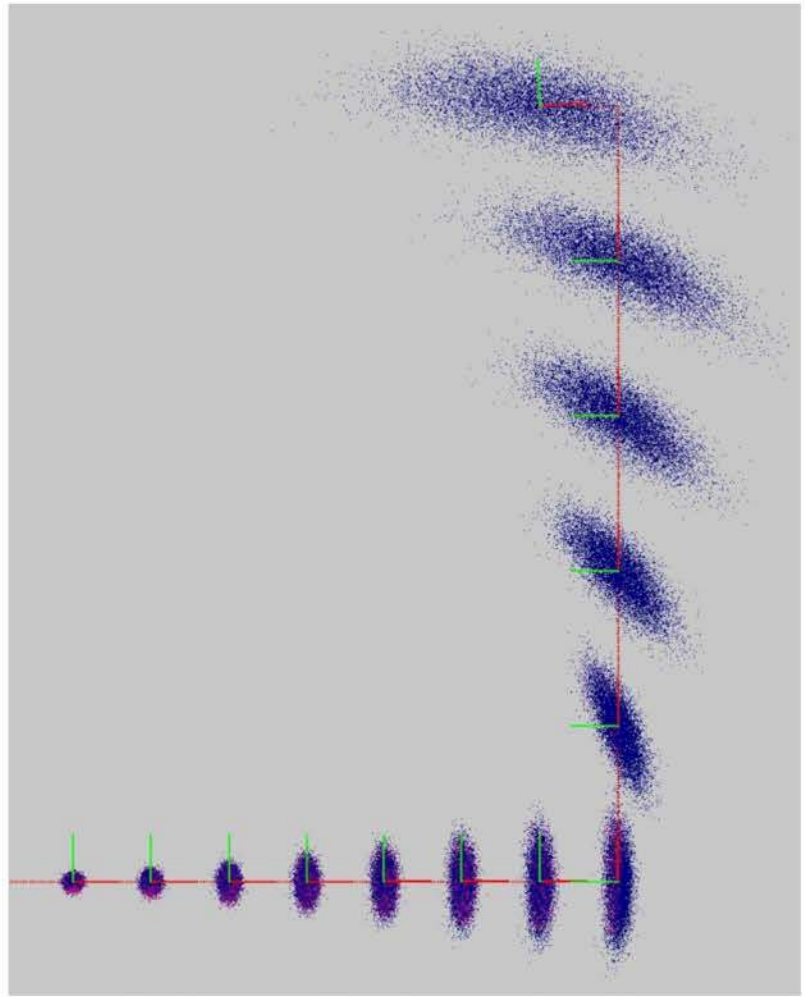


Notice how uncertainty increases →

Evolution of noise



No heading error



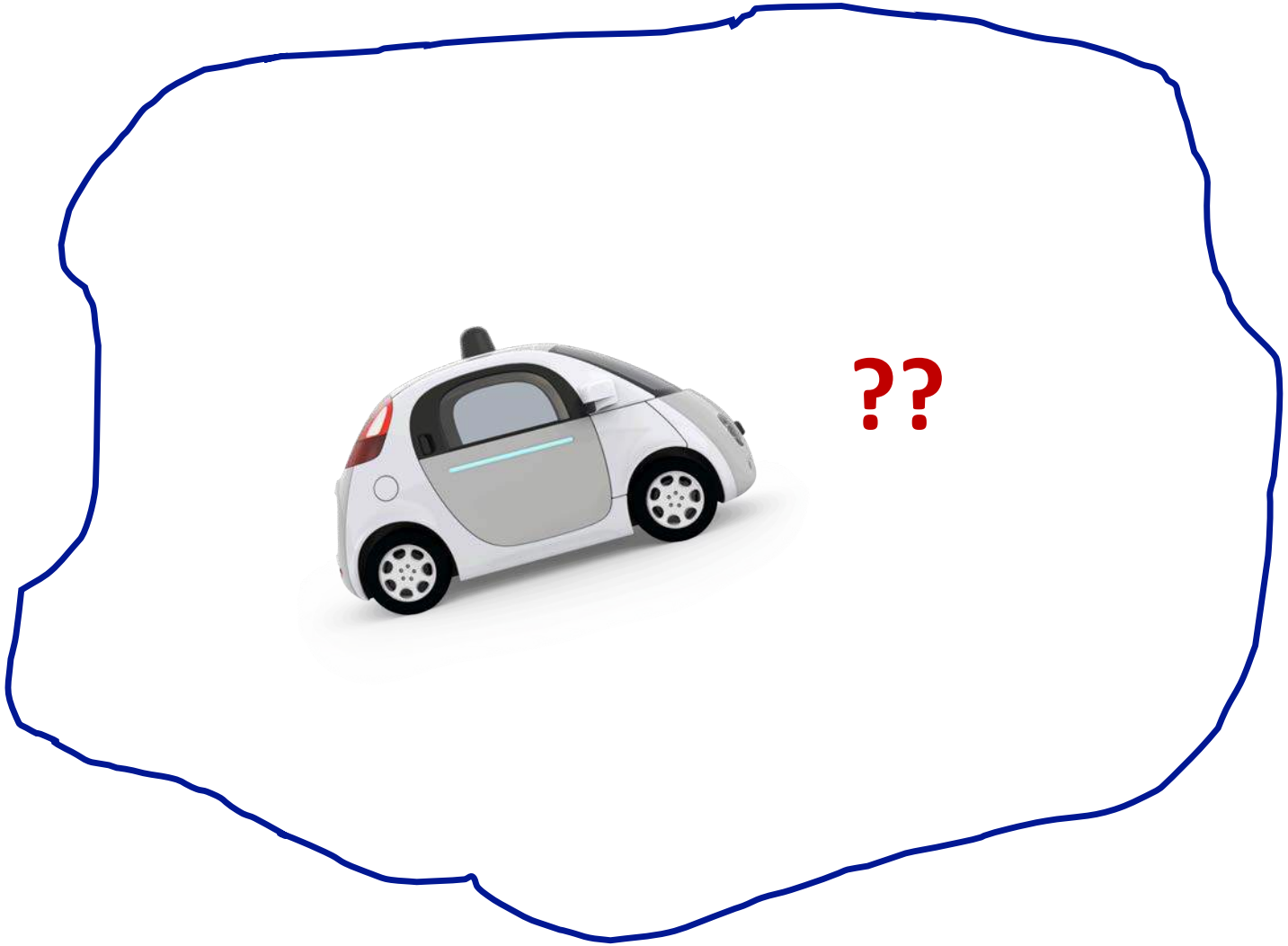
With heading error

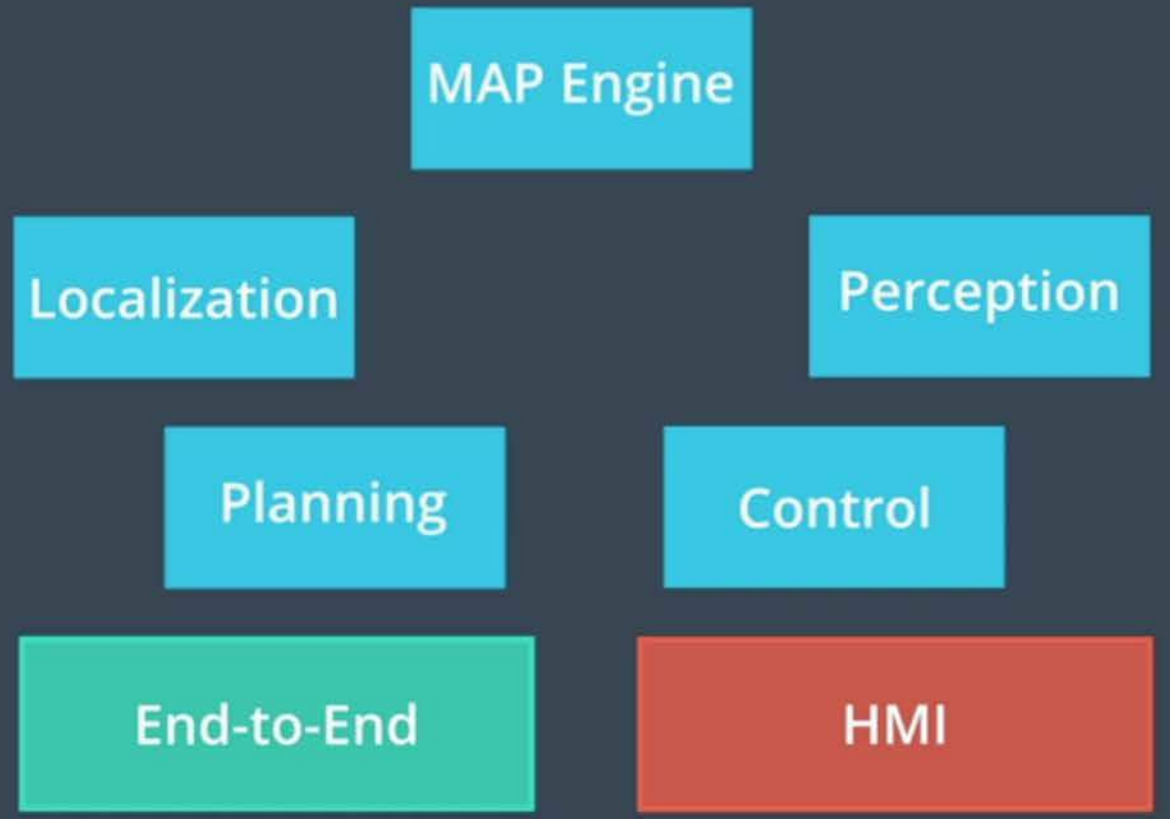
Beyond odometry

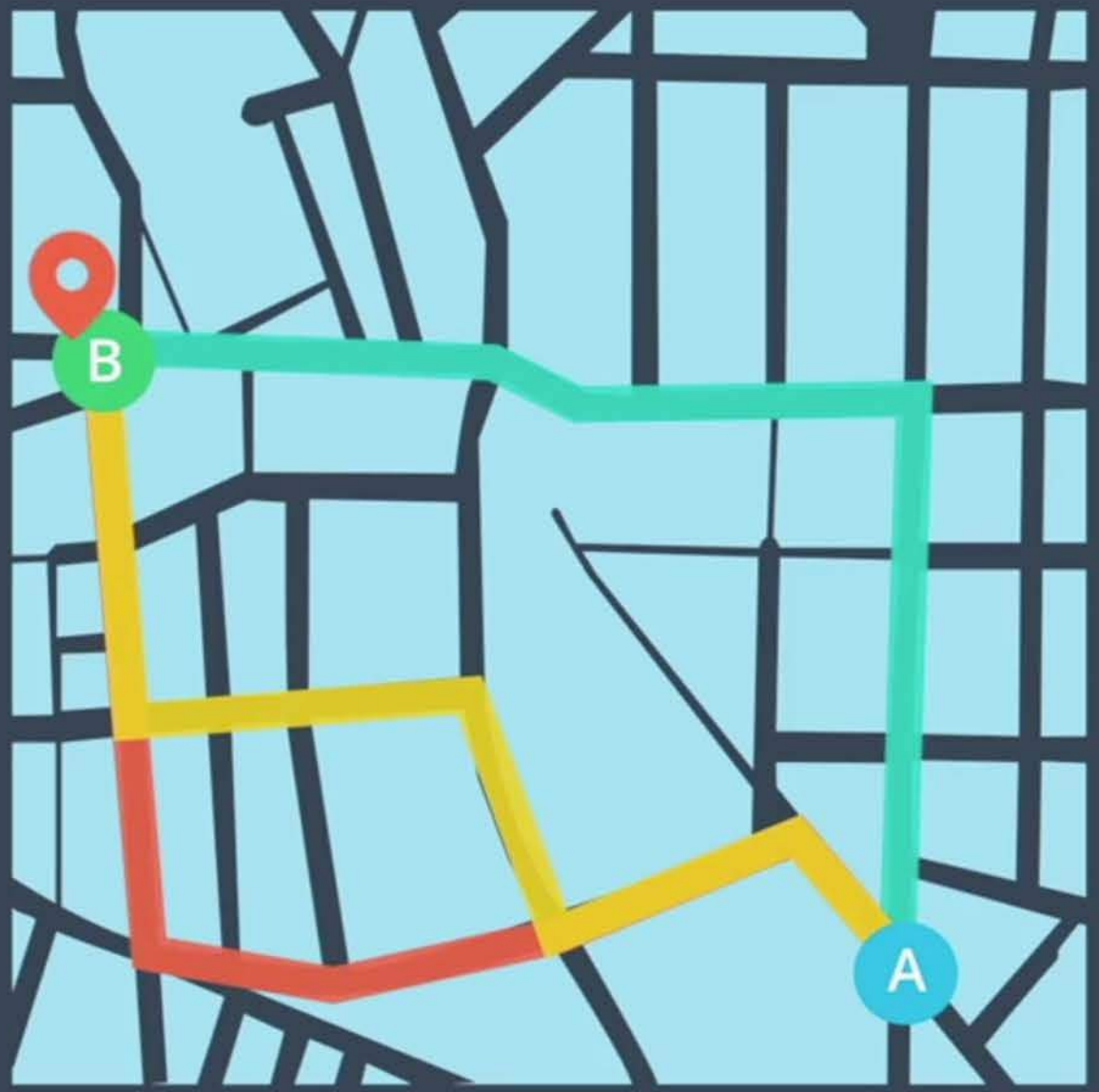
- Using odometry for localization is known as “dead reckoning” in aircraft and maritime navigation

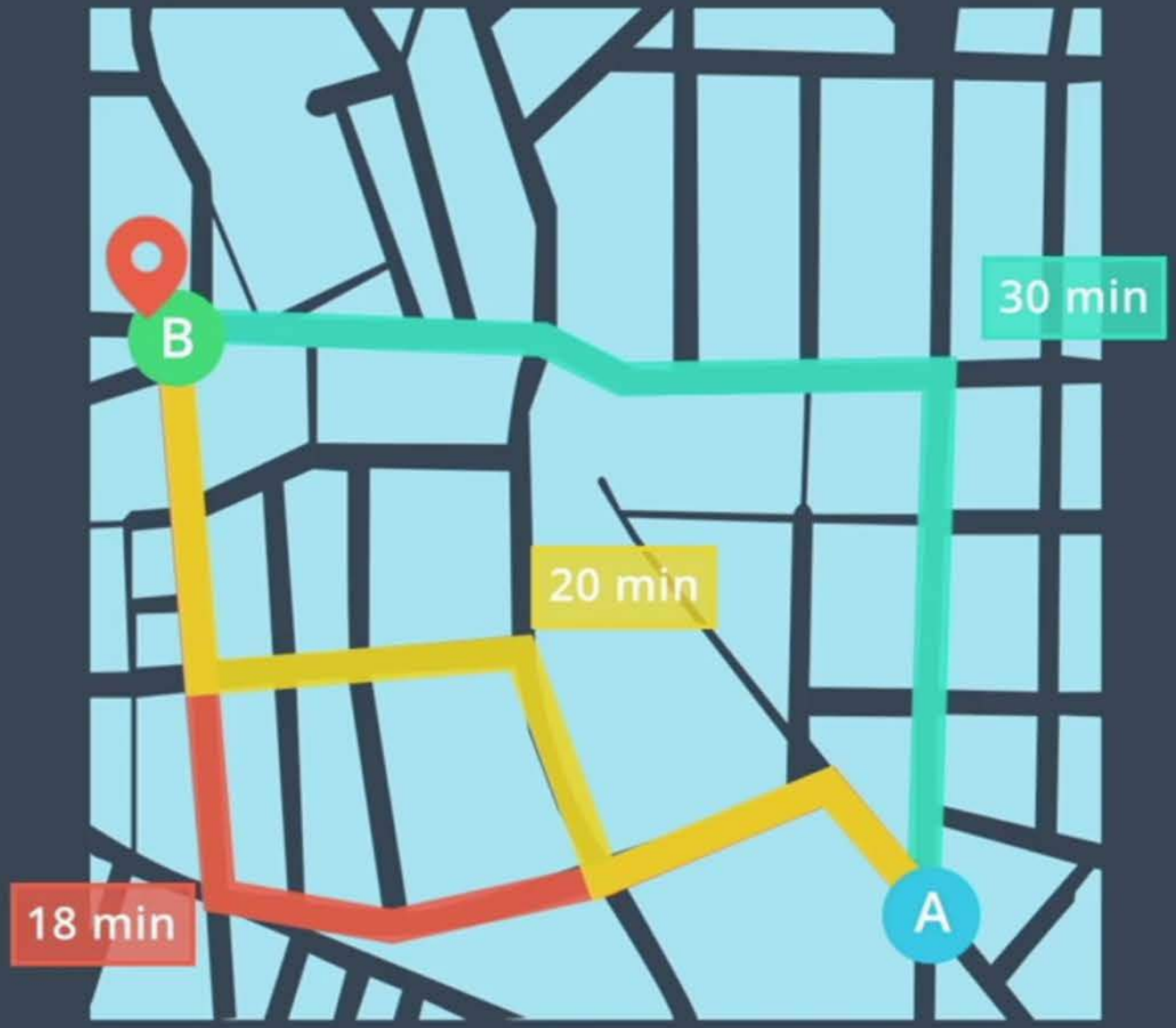
- How do we utilize that information, such as range sensor measurements to localize?

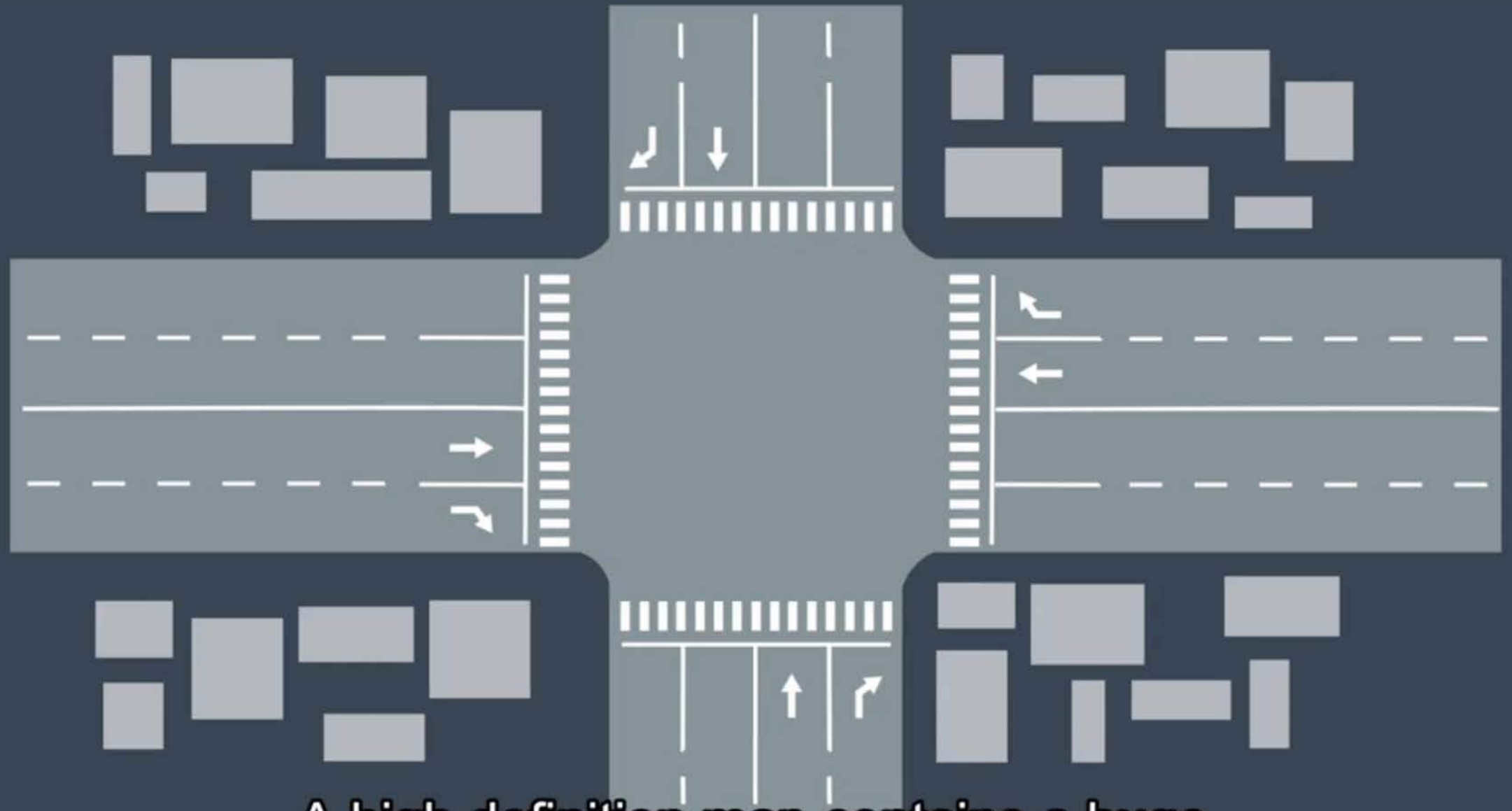
How can we know where we are with 10cm level accuracy ?




MAP Engine
Localization
Perception
Planning
Control
End-to-End
HMI


B
A


B
30 min
20 min
18 min
A

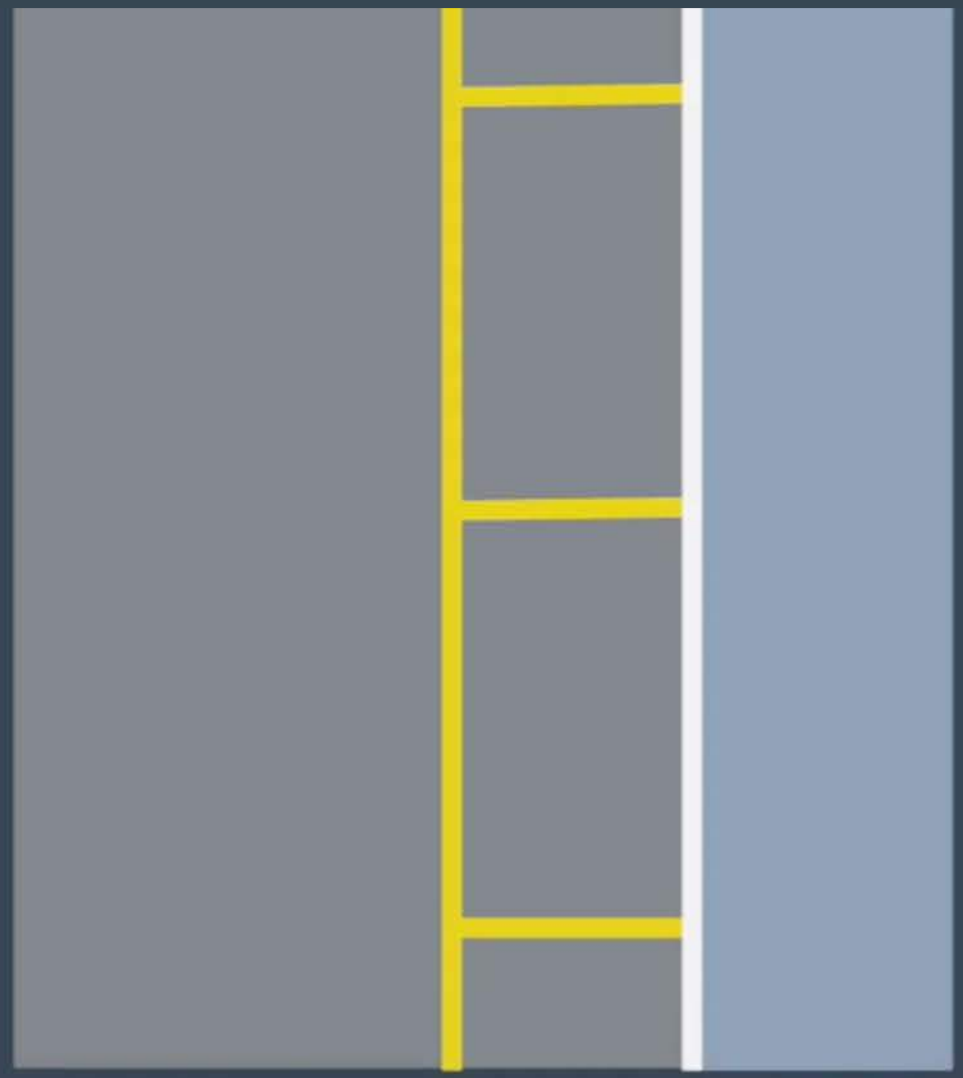


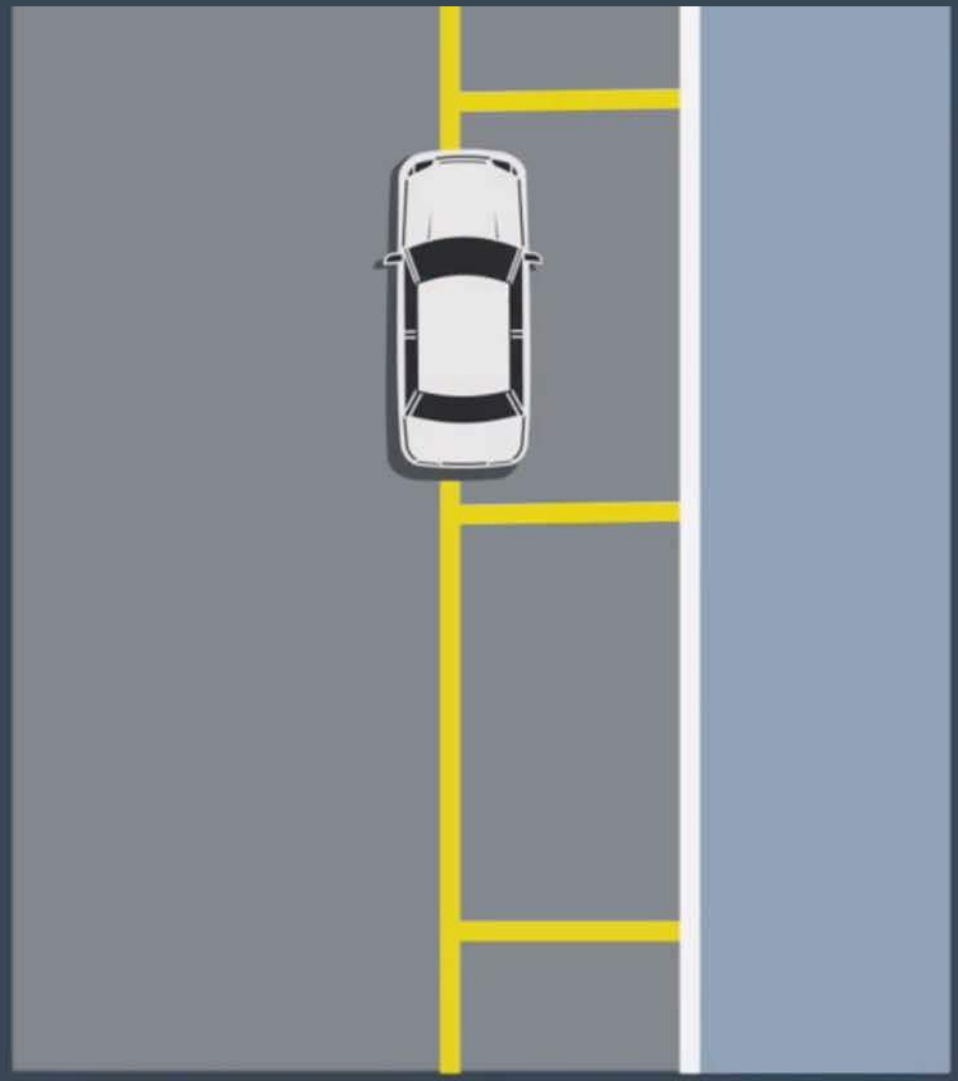
A high definition map contains a huge amount of driving assistance information.

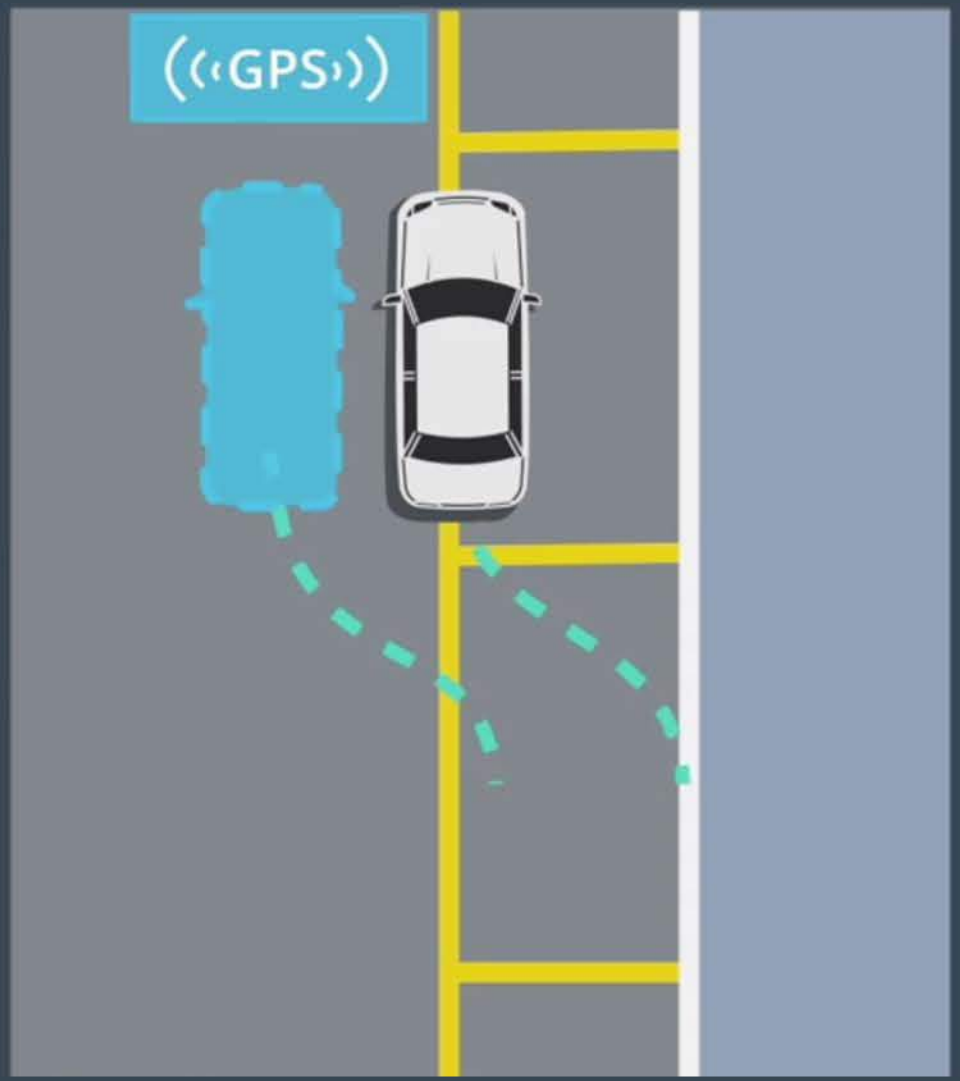
A high definition map also contains a lot of semantic information.

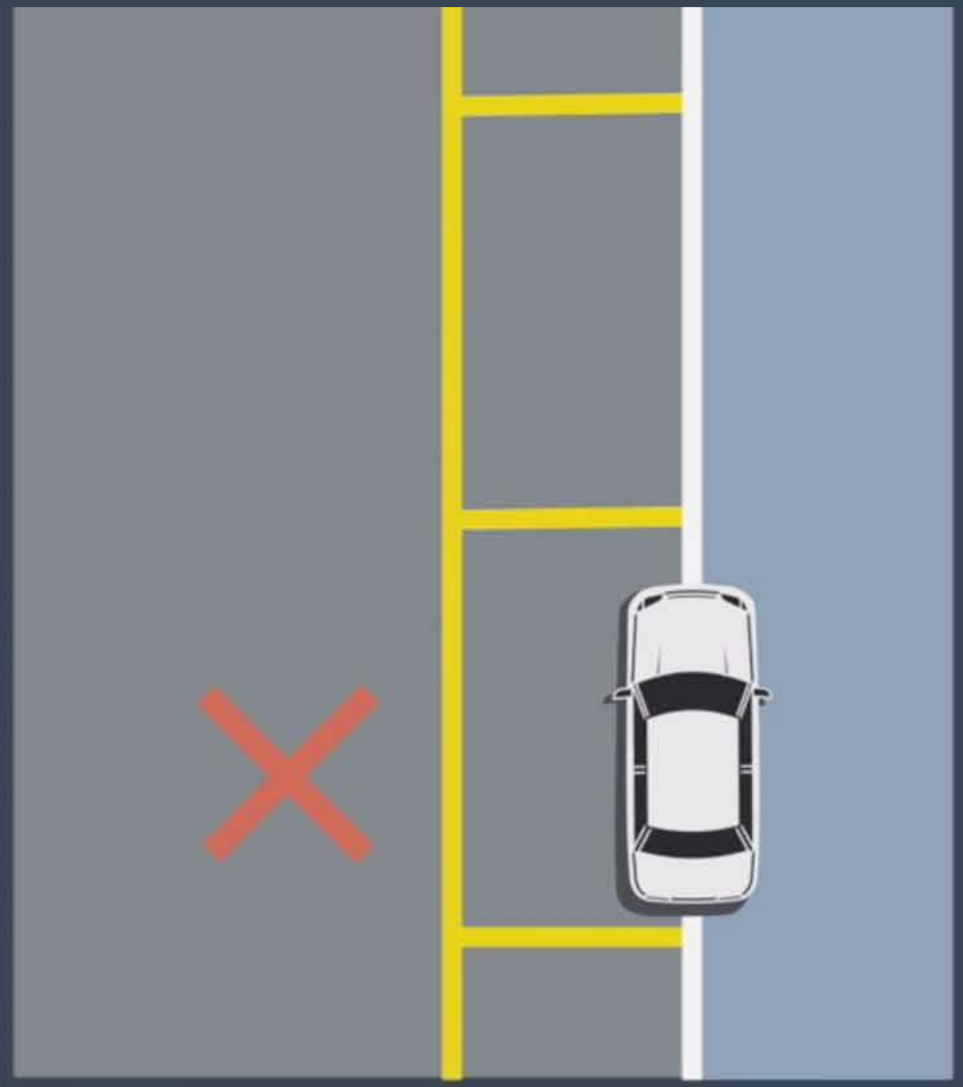

40 km/h


Left Turn

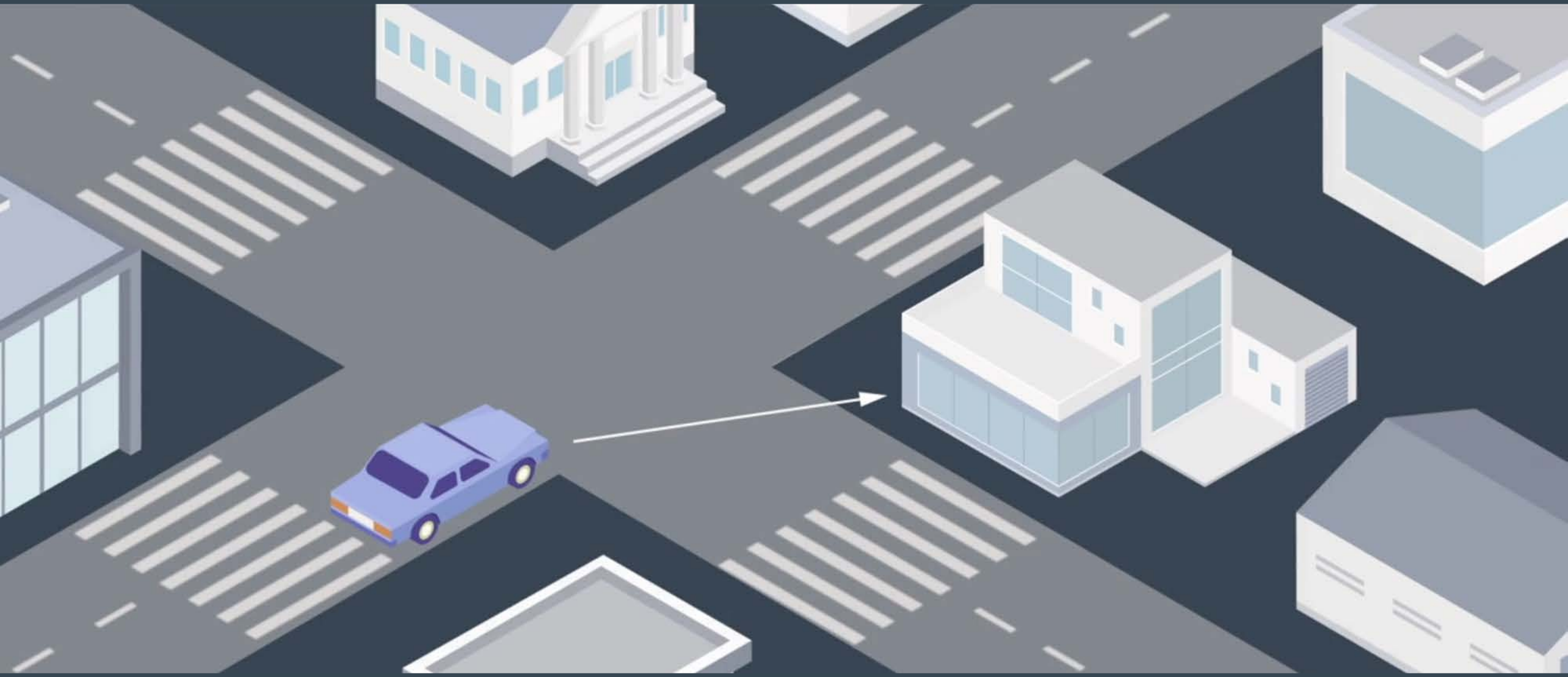


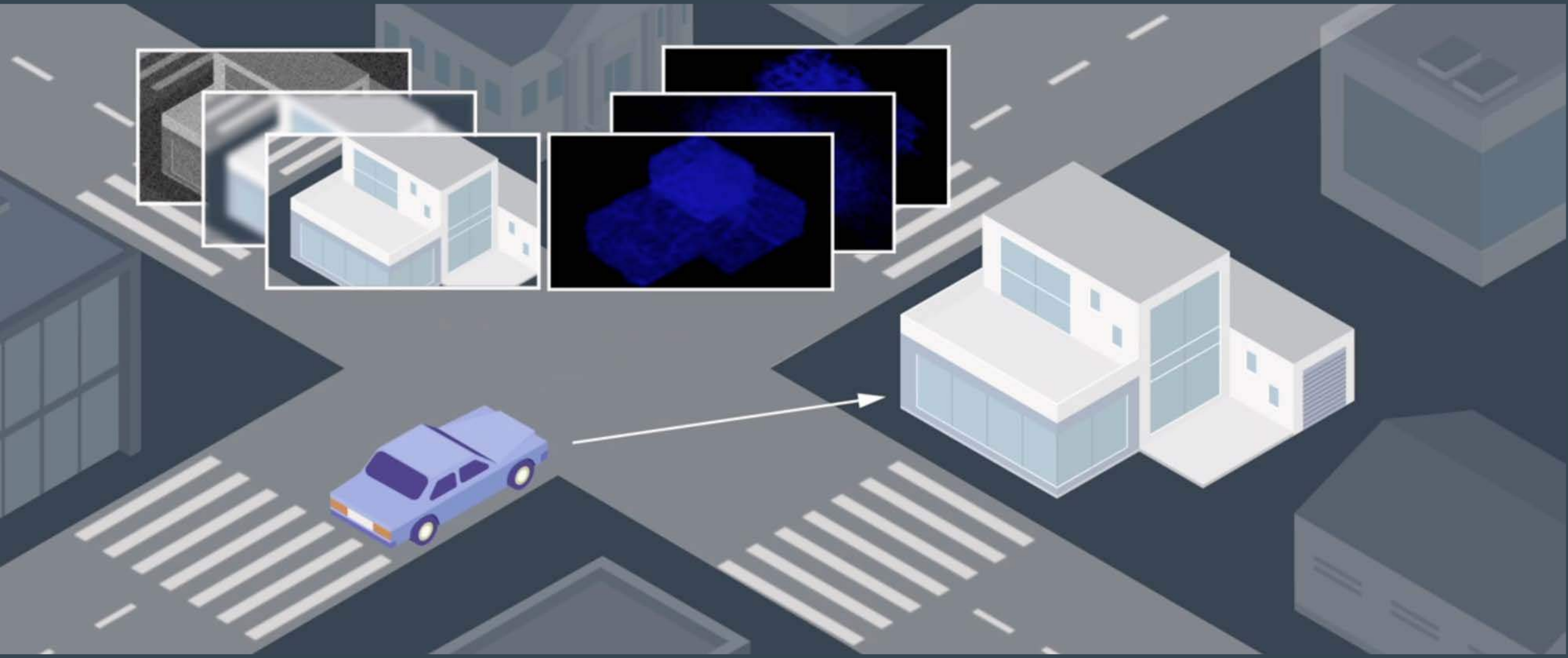


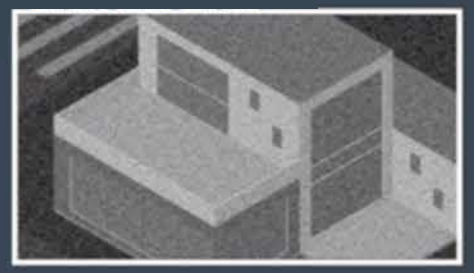

((GPS))

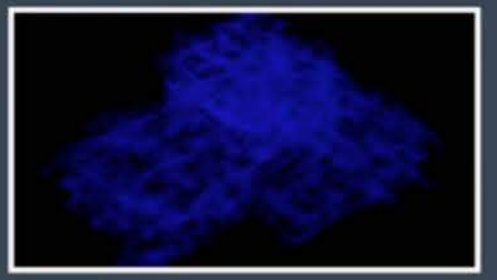


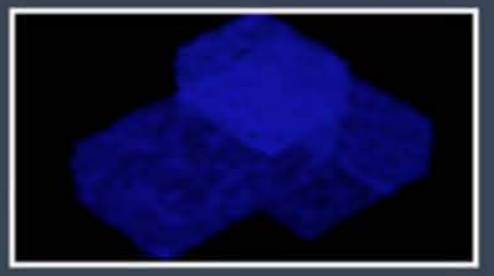
Localization And Scan Matching

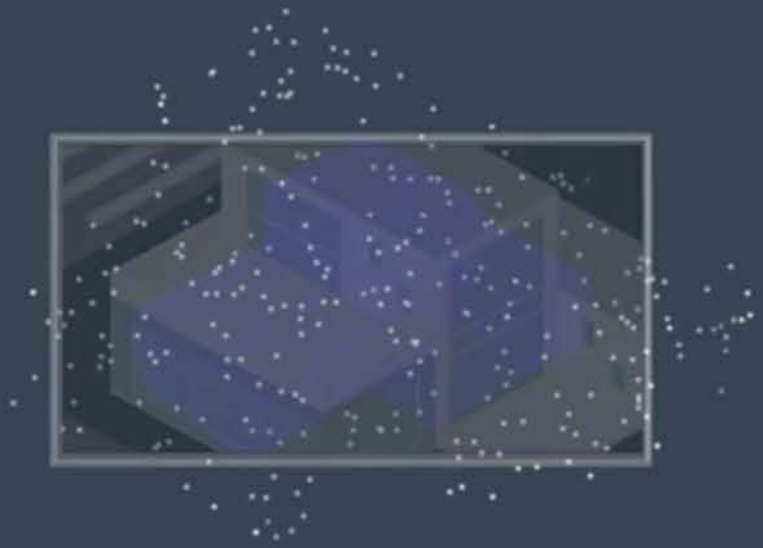




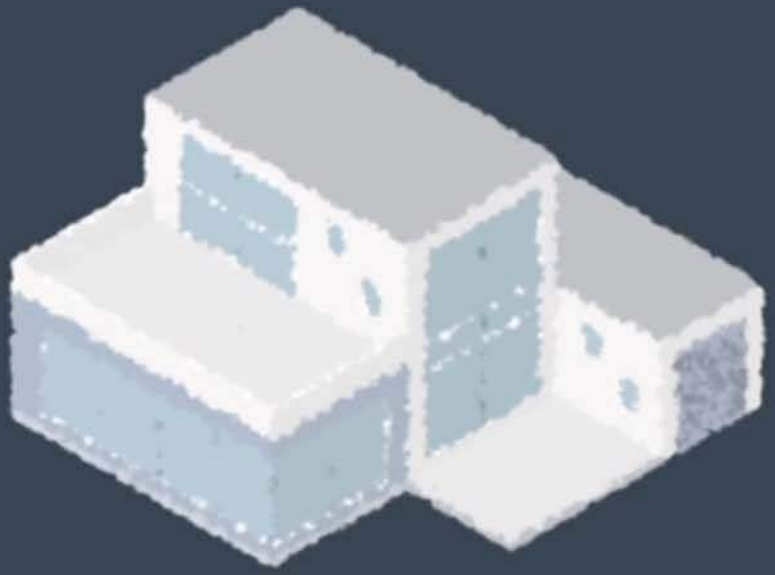


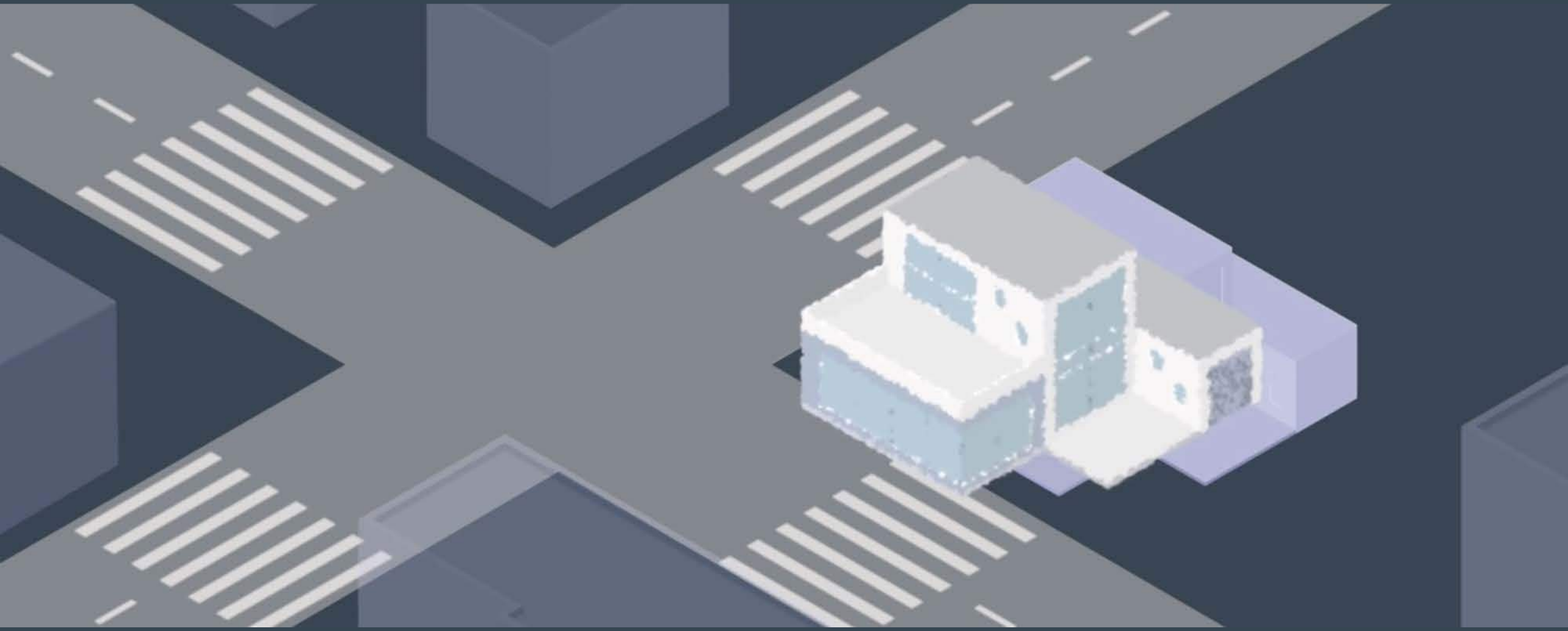


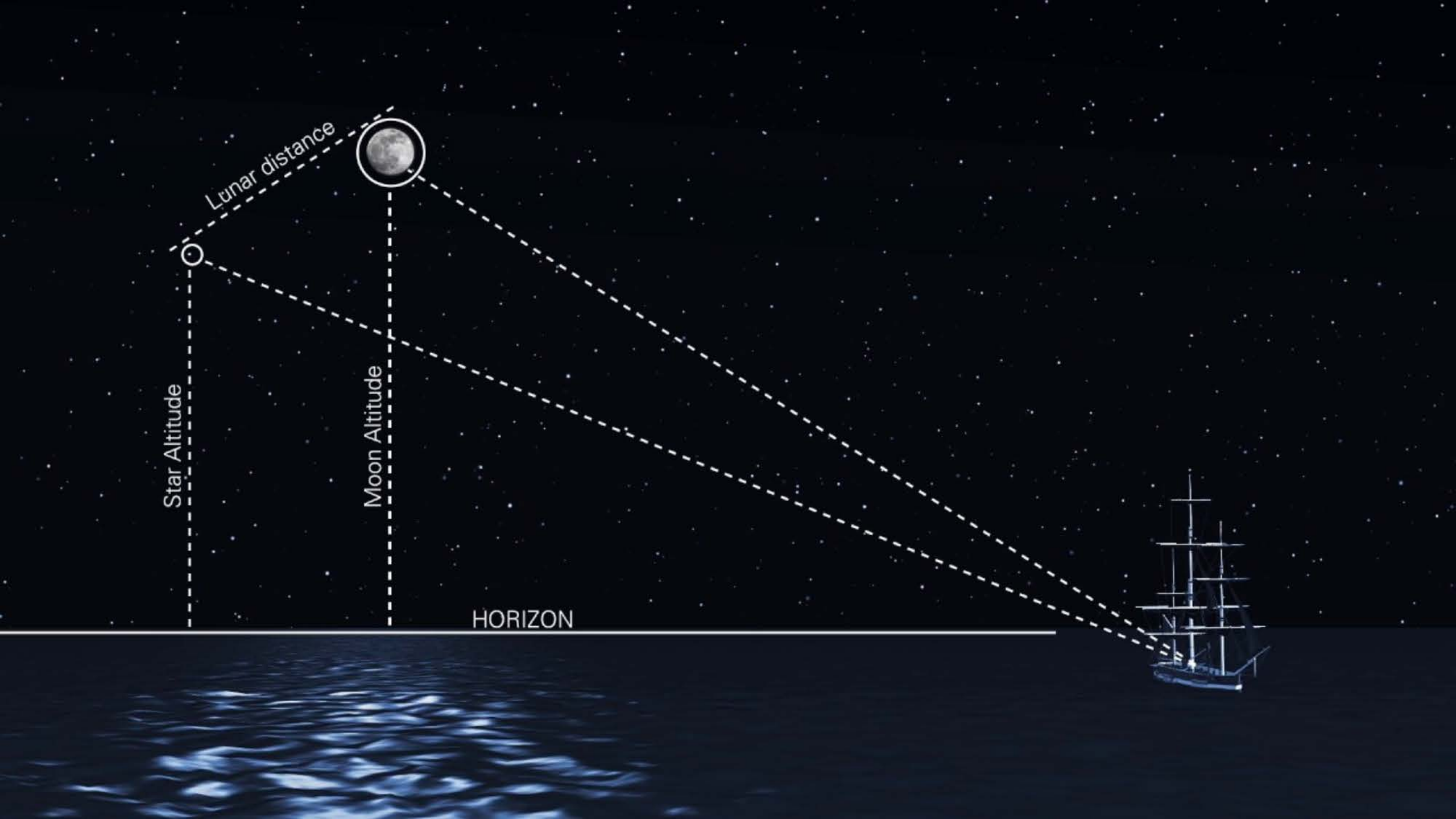




Data fusion merges data from different vehicles and different types of sensors.






Lunar distance
Star Altitude
Moon Altitude
HORIZON

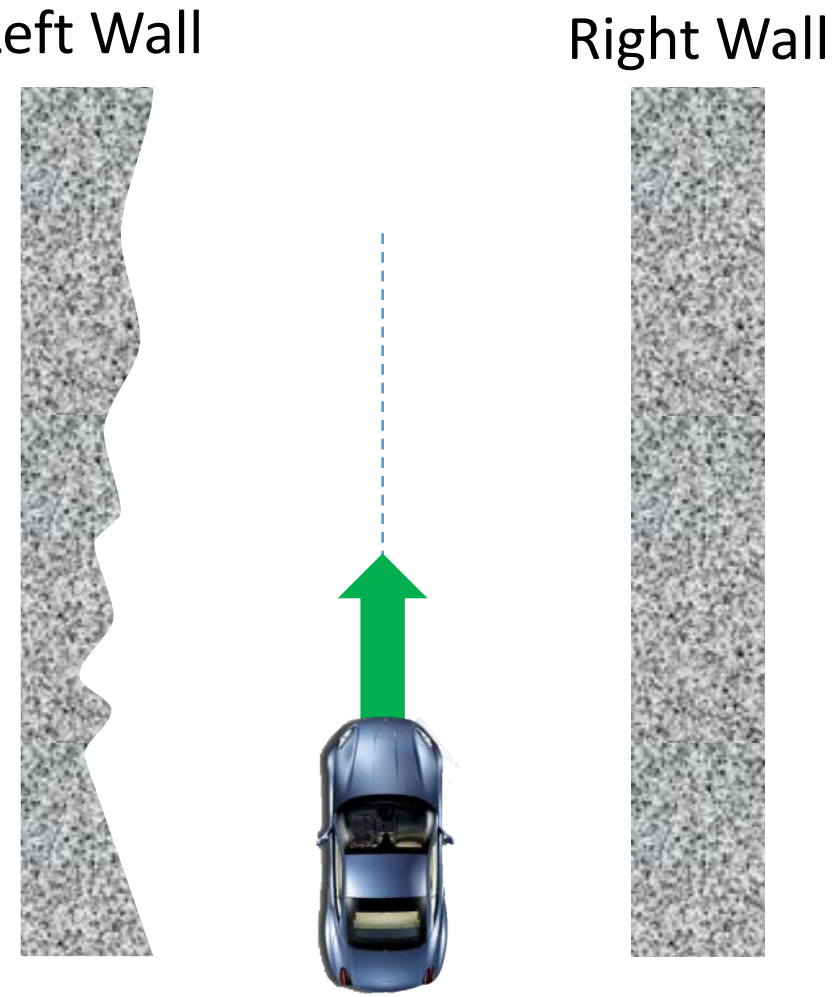
Scan Matching

Odometry with LIDAR

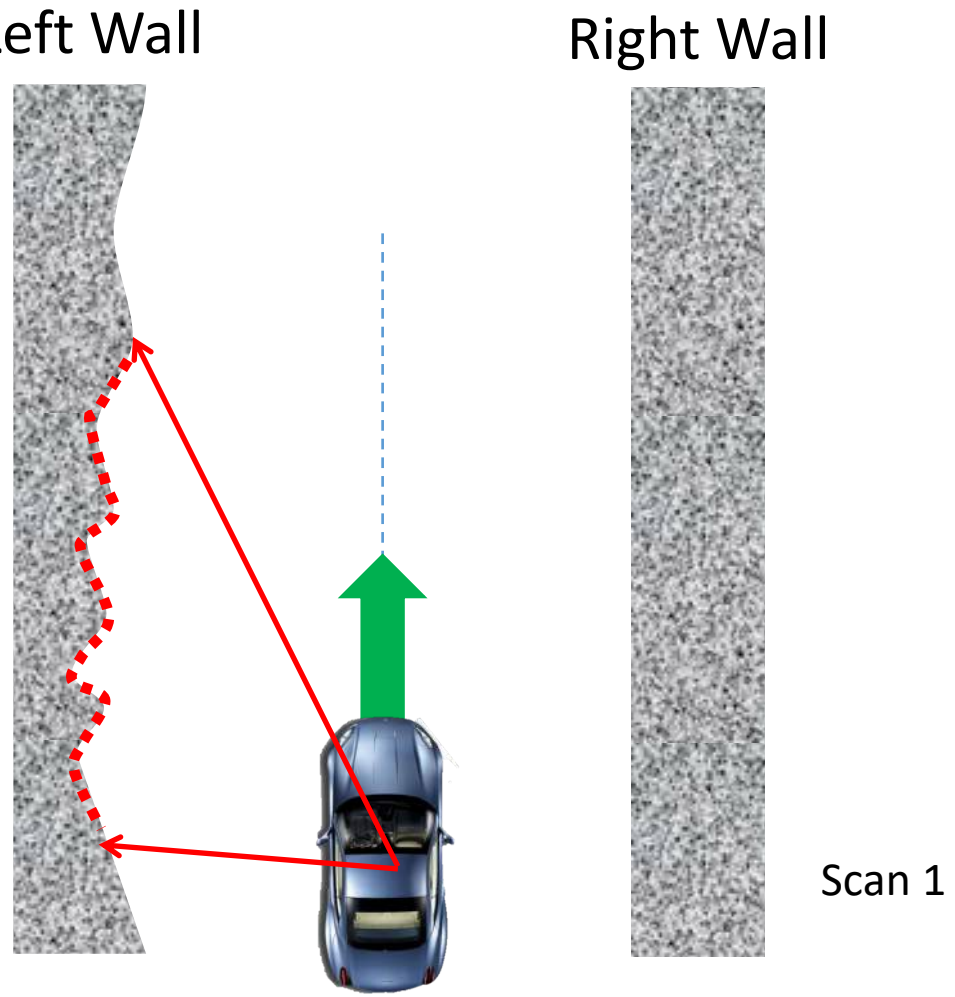
- Odometry = Estimation of position (x, y, z) and attitude (θ, φ, ψ)

- In what follows, will use “position” as short for “position and attitude”

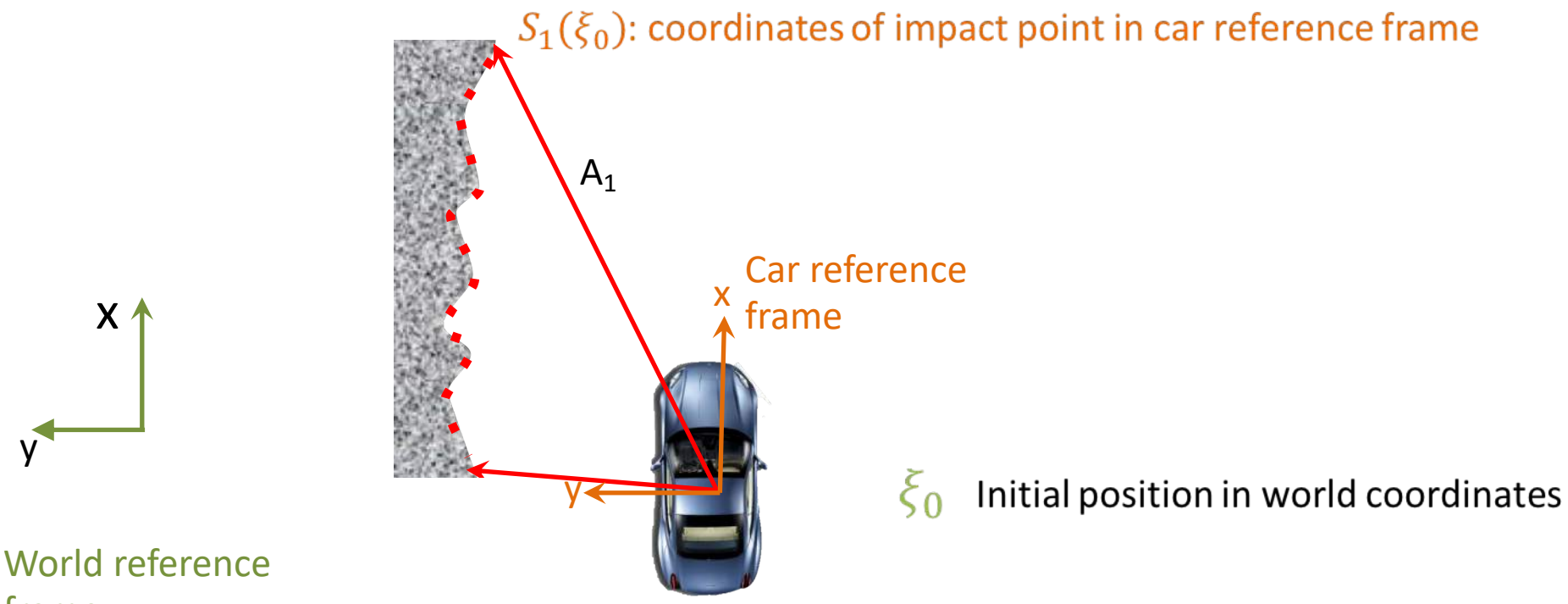
LIDAR for odometry: Scan matching



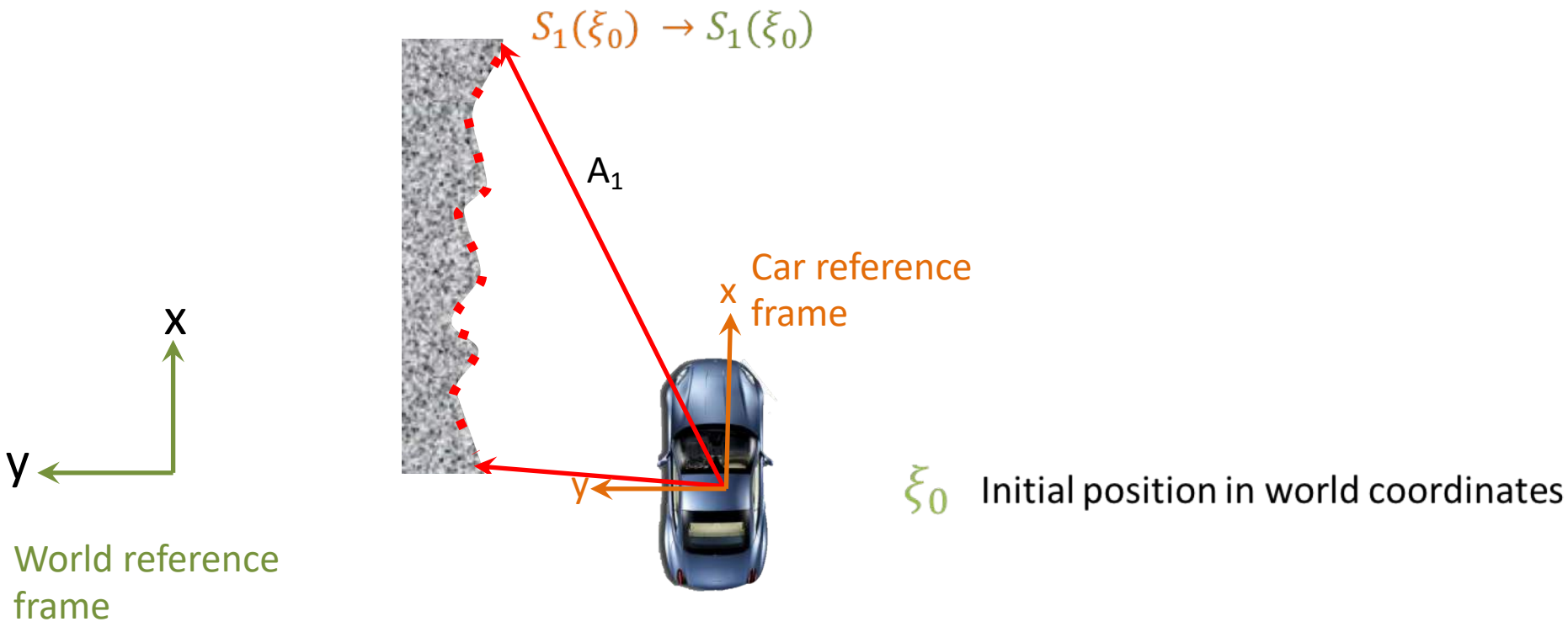
LIDAR for odometry: Scan matching

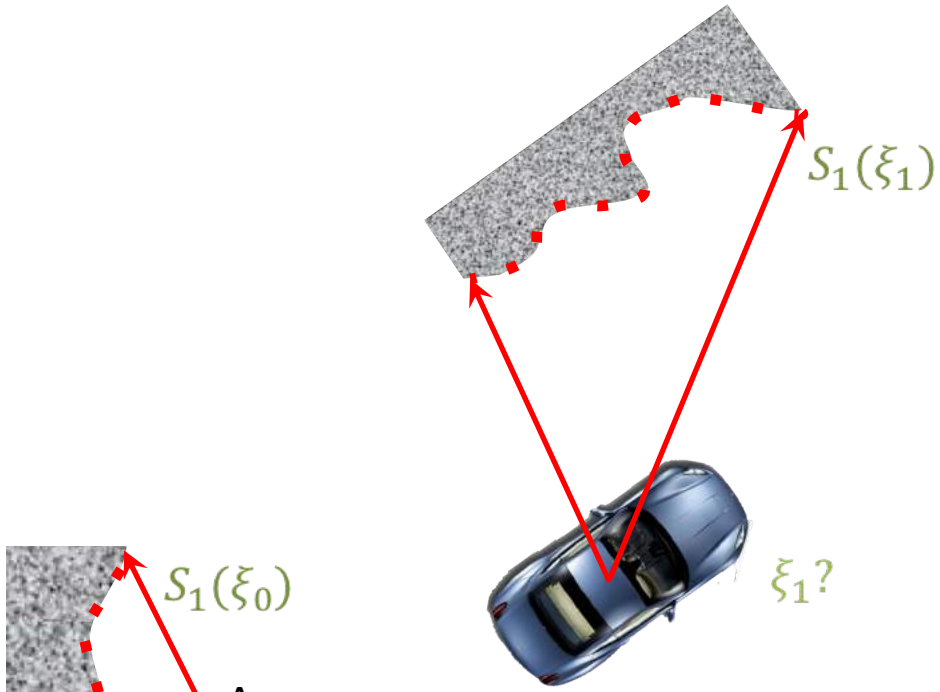


LIDAR for odometry: Scan 1



LIDAR for odometry: Scan 1





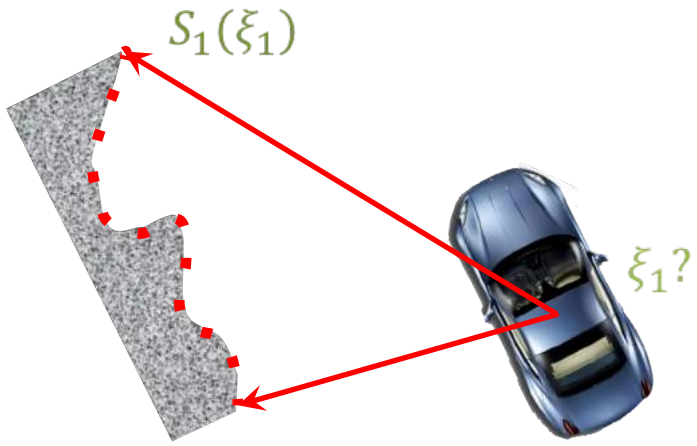
Car reference frame

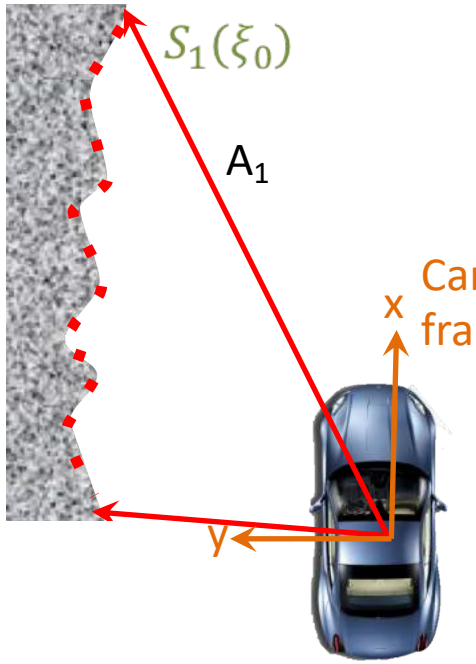
x

y

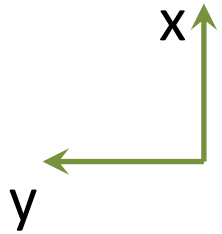
World reference frame

ξ_0 Initial position in world coordinates



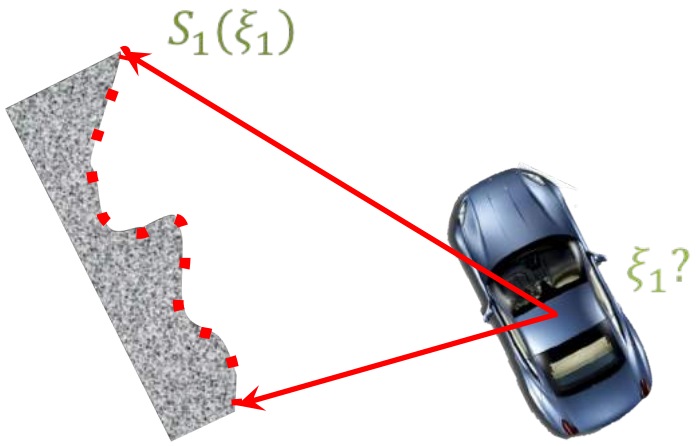


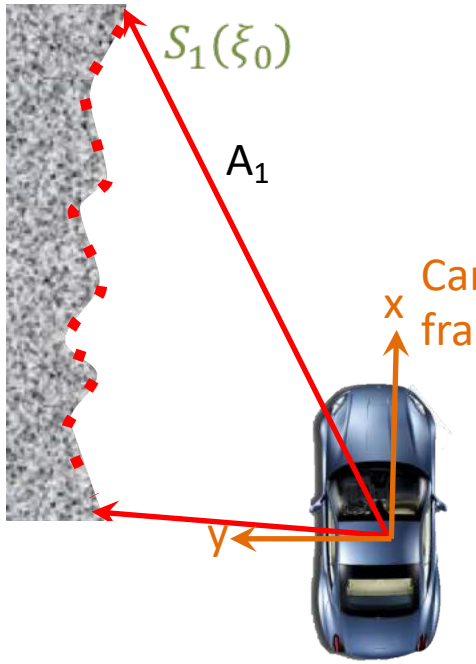
Car reference frame



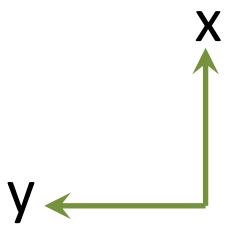
World reference frame

ξ_0 Initial position in world coordinates





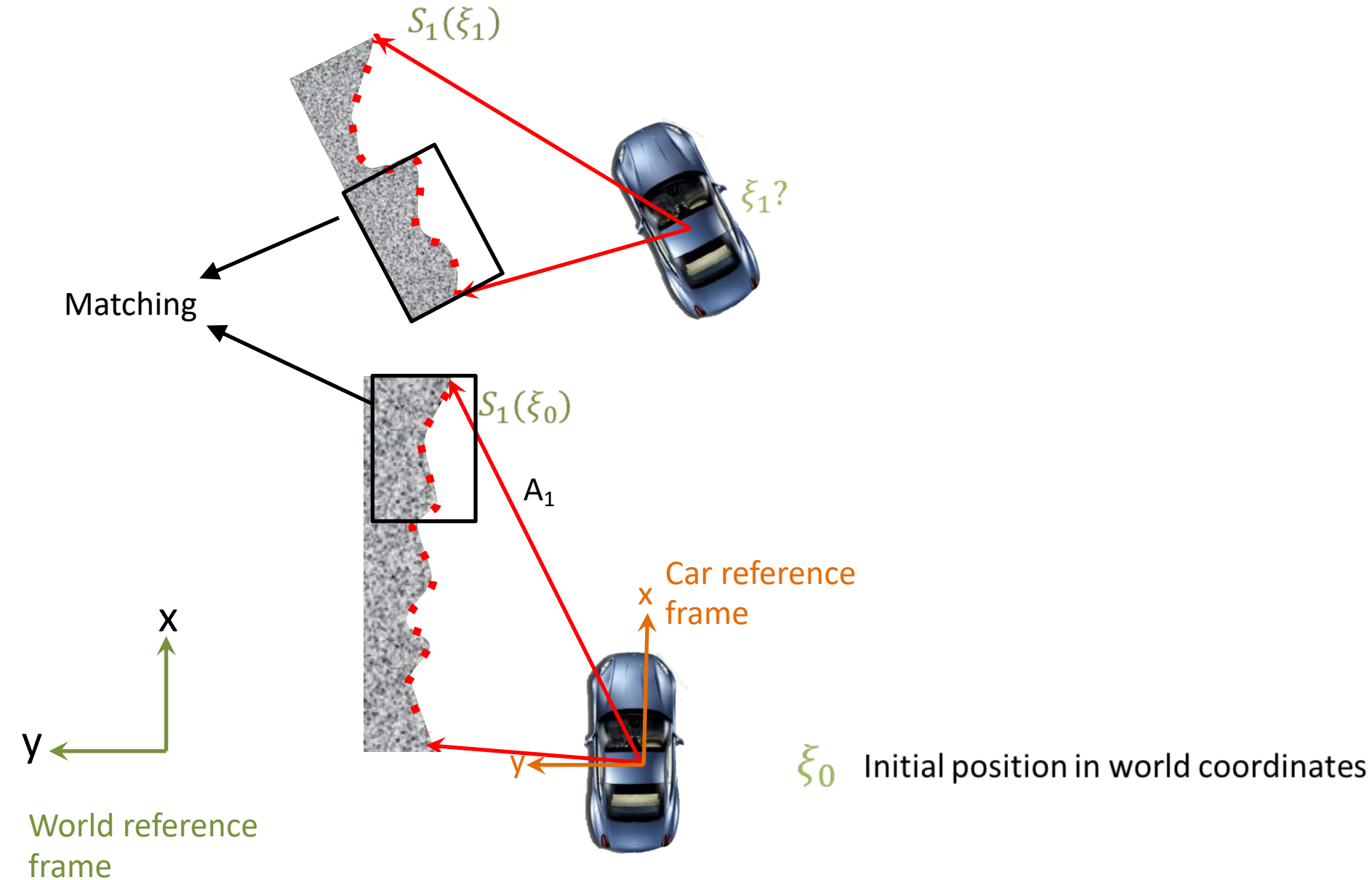
Car reference frame

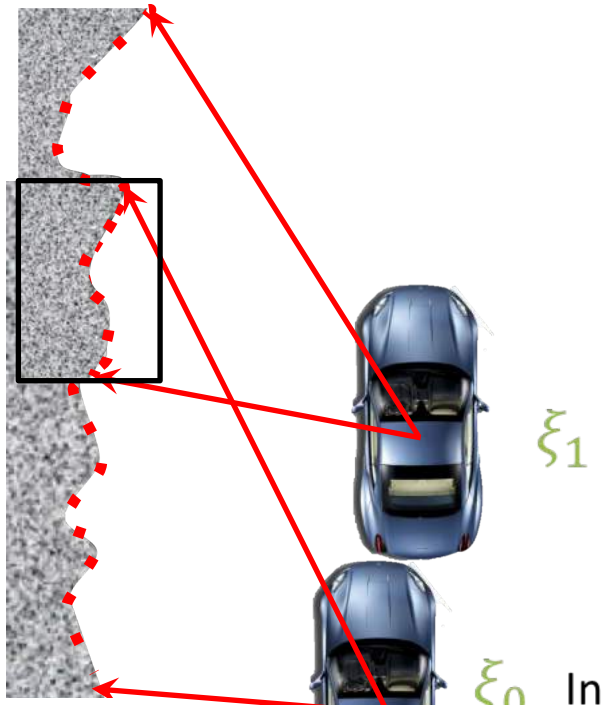


World reference frame

Premise:
most likely car position at Scan 2 is the position that gives best overlap between the two scenes

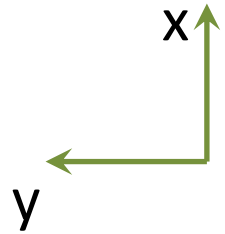
ξ_0 Initial position in world coordinates


$S_1(\xi_1)$
$\xi_1?$
Matching
$S_1(\xi_0)$
A_1
Car reference frame
x
y
x
y
World reference frame
ξ_0 Initial position in world coordinates



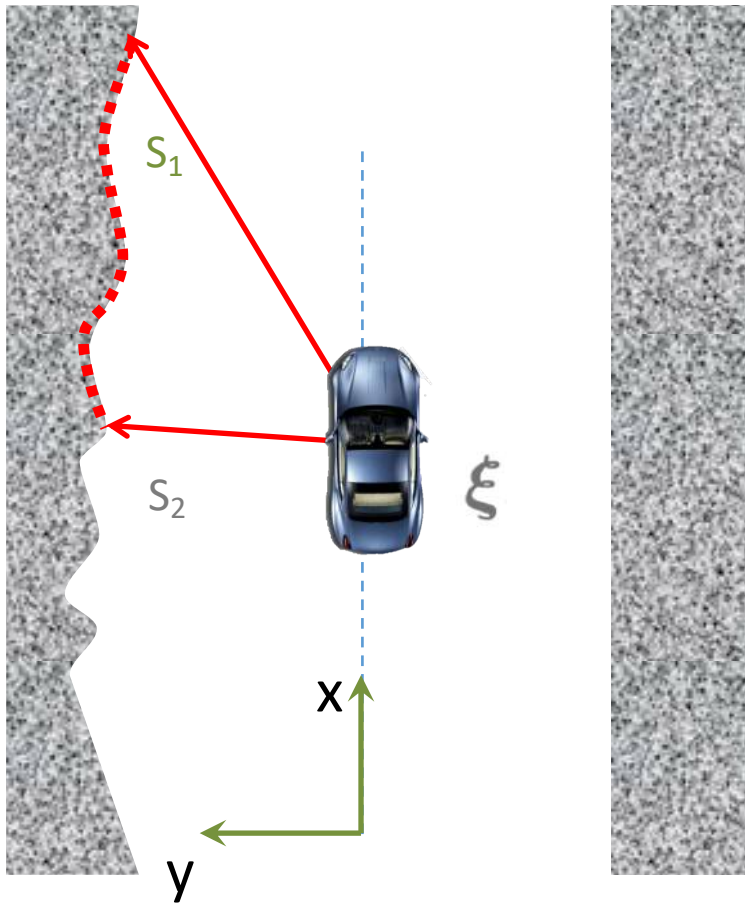
ξ_1

ξ_0 Initial position in world coordinates



World reference frame

Scan matching: optimization

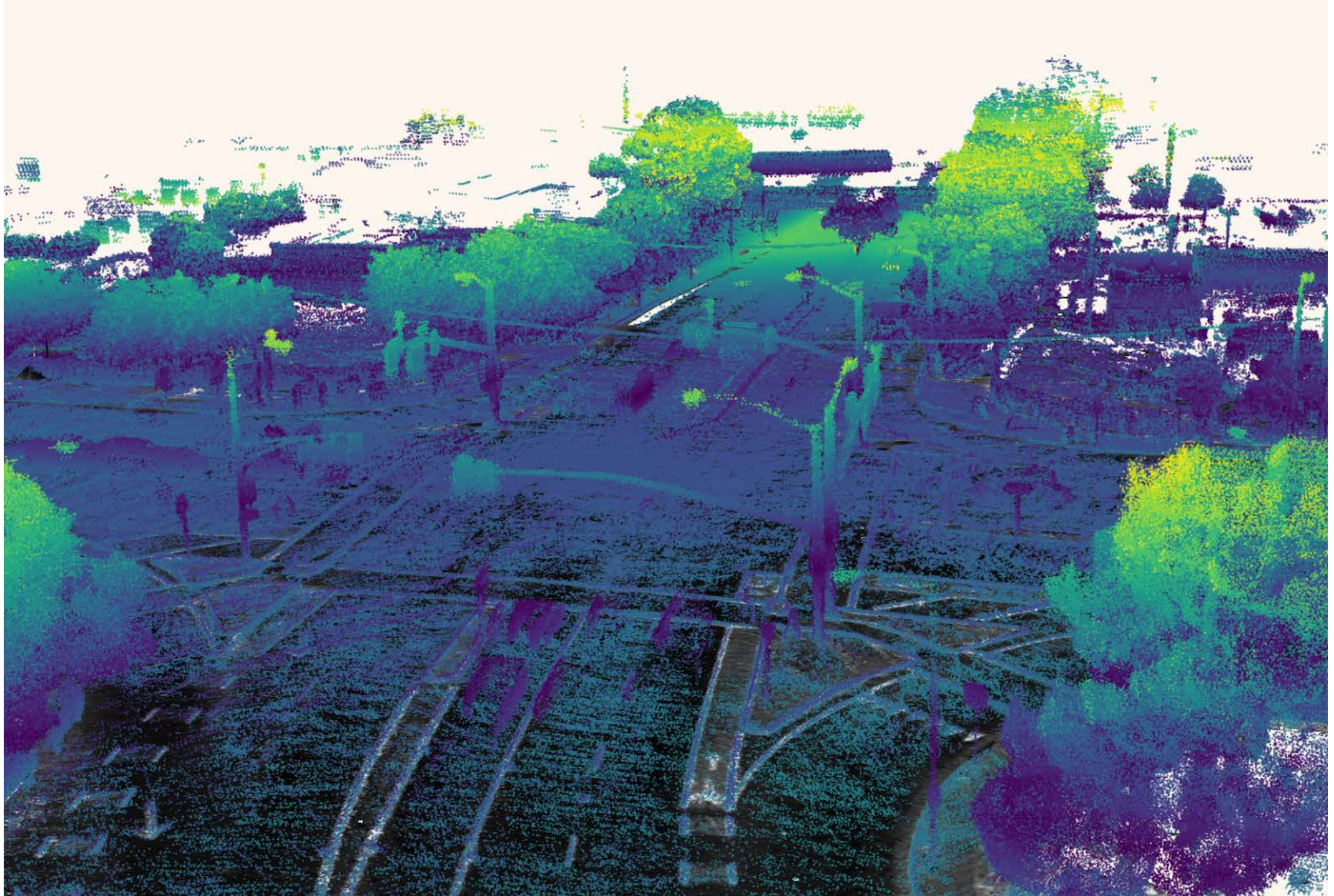


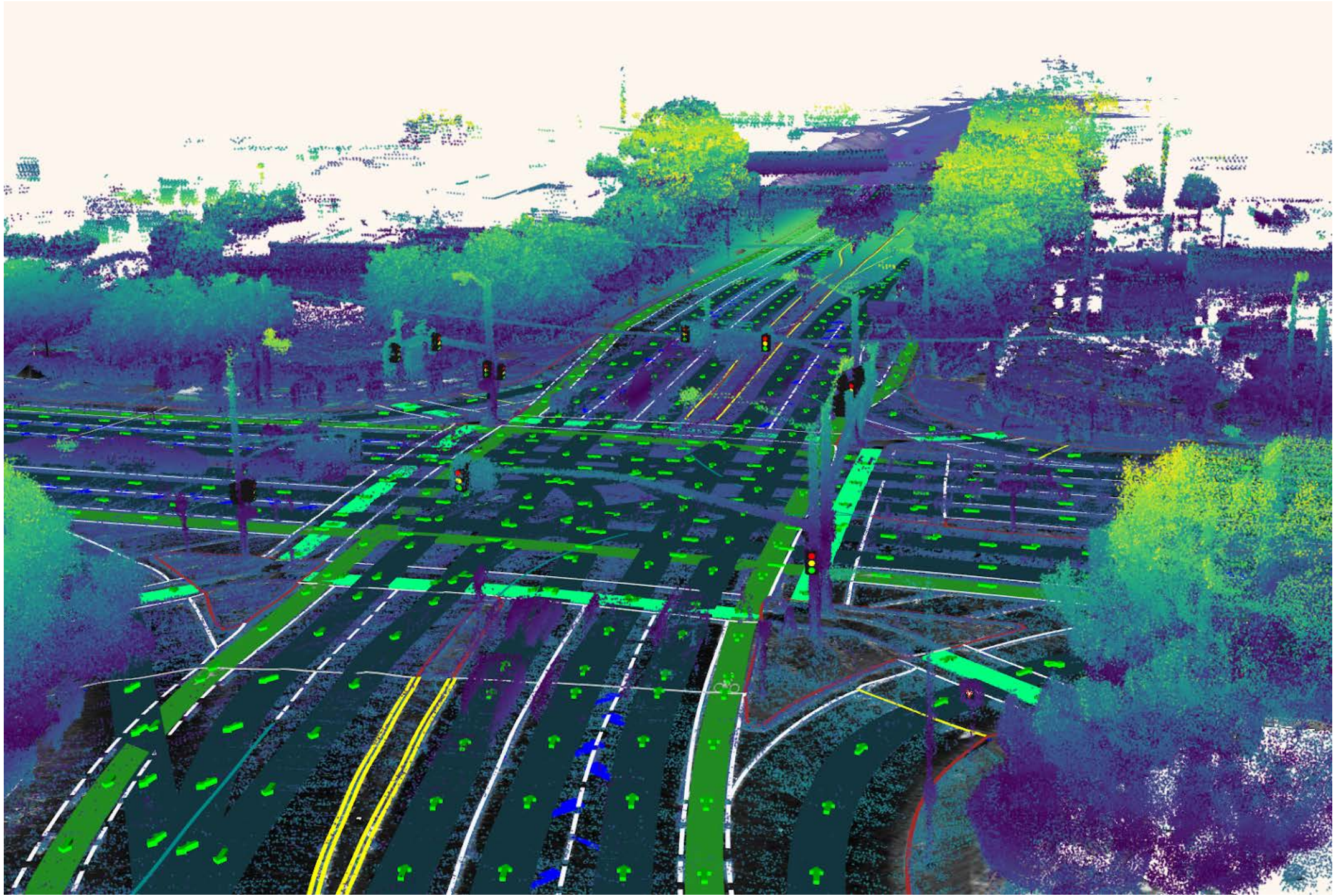
Impact coordinates of i^{th} step in world frame

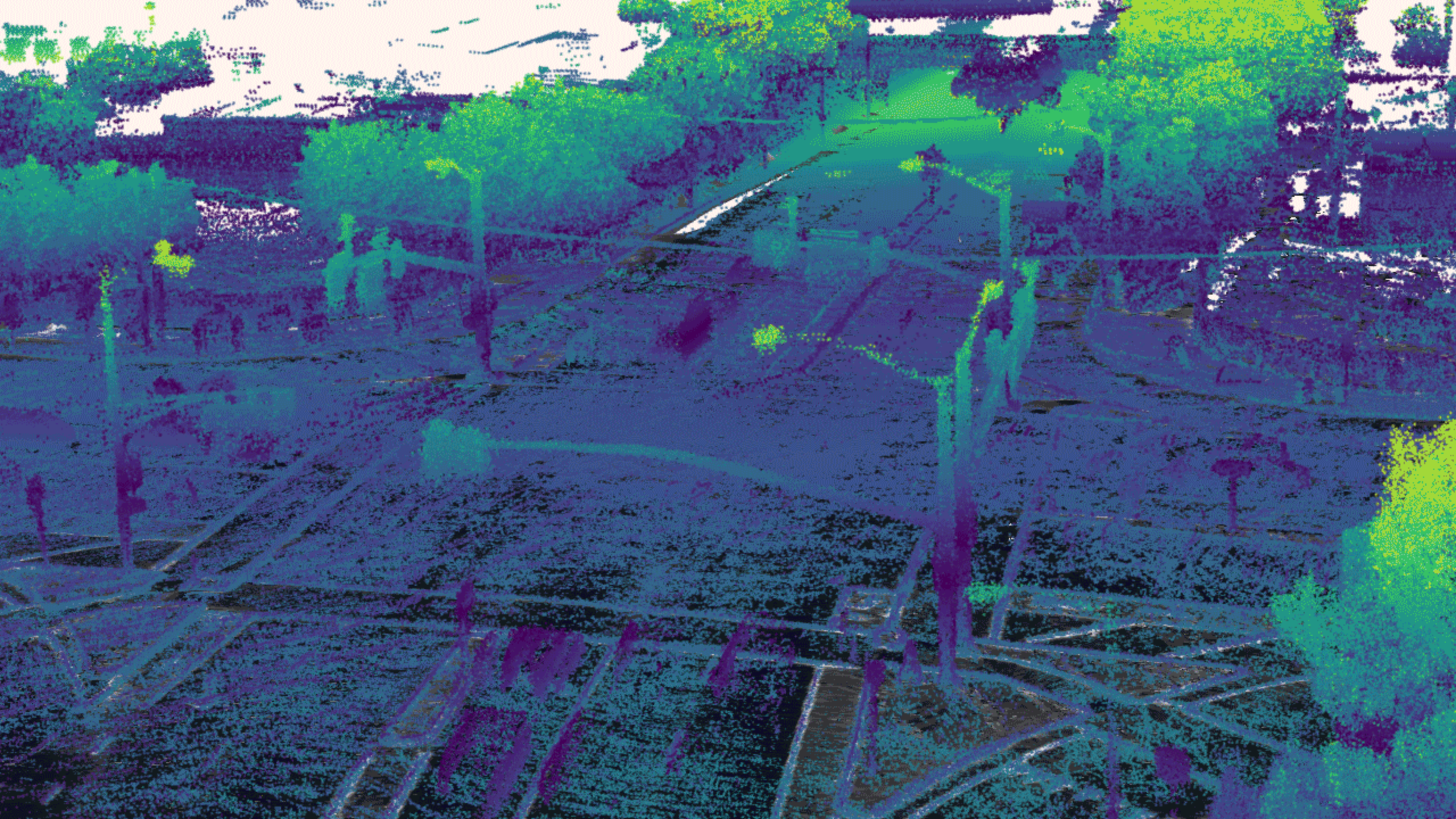
Total of n steps (n = 1084)

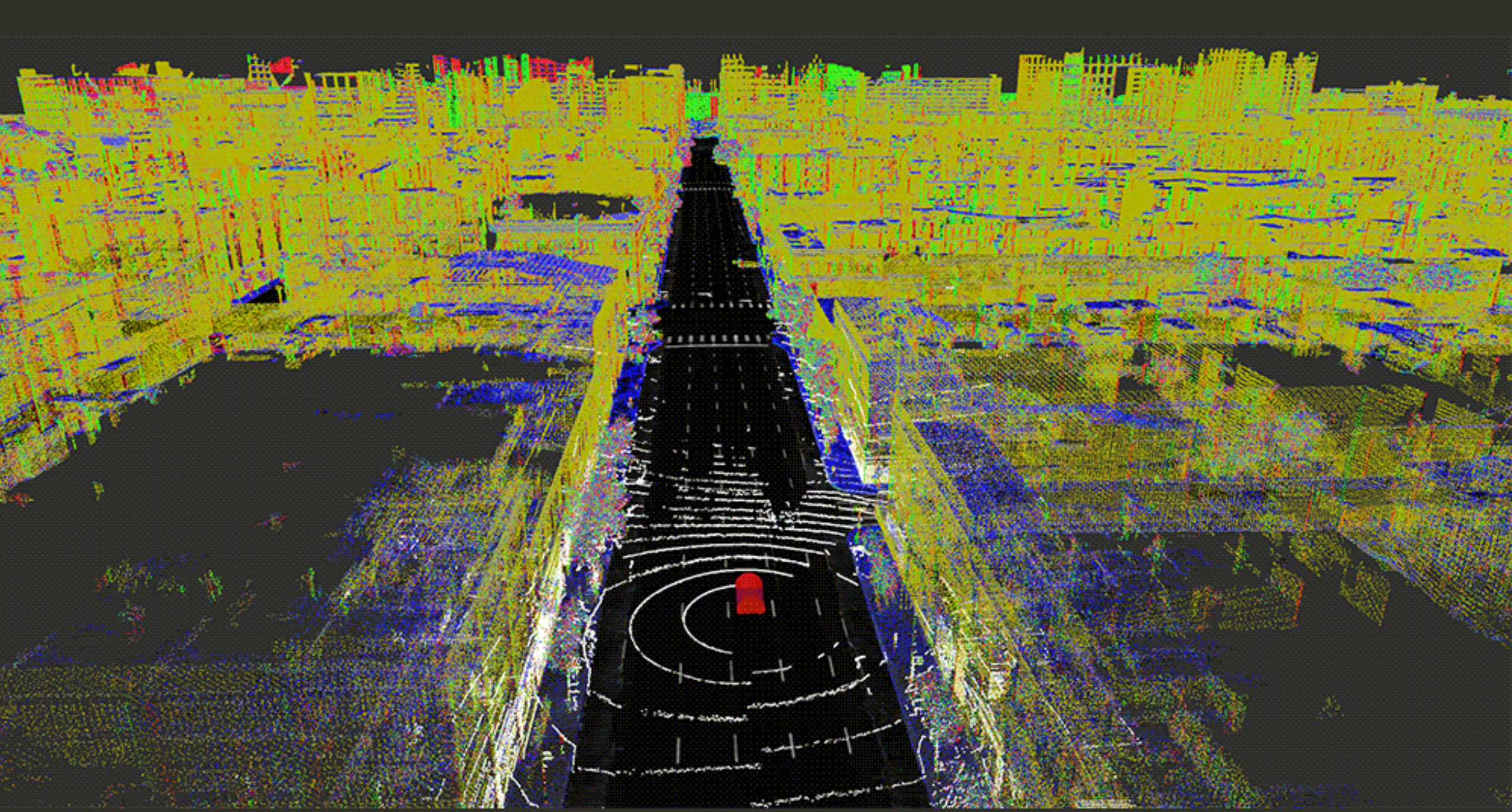
$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

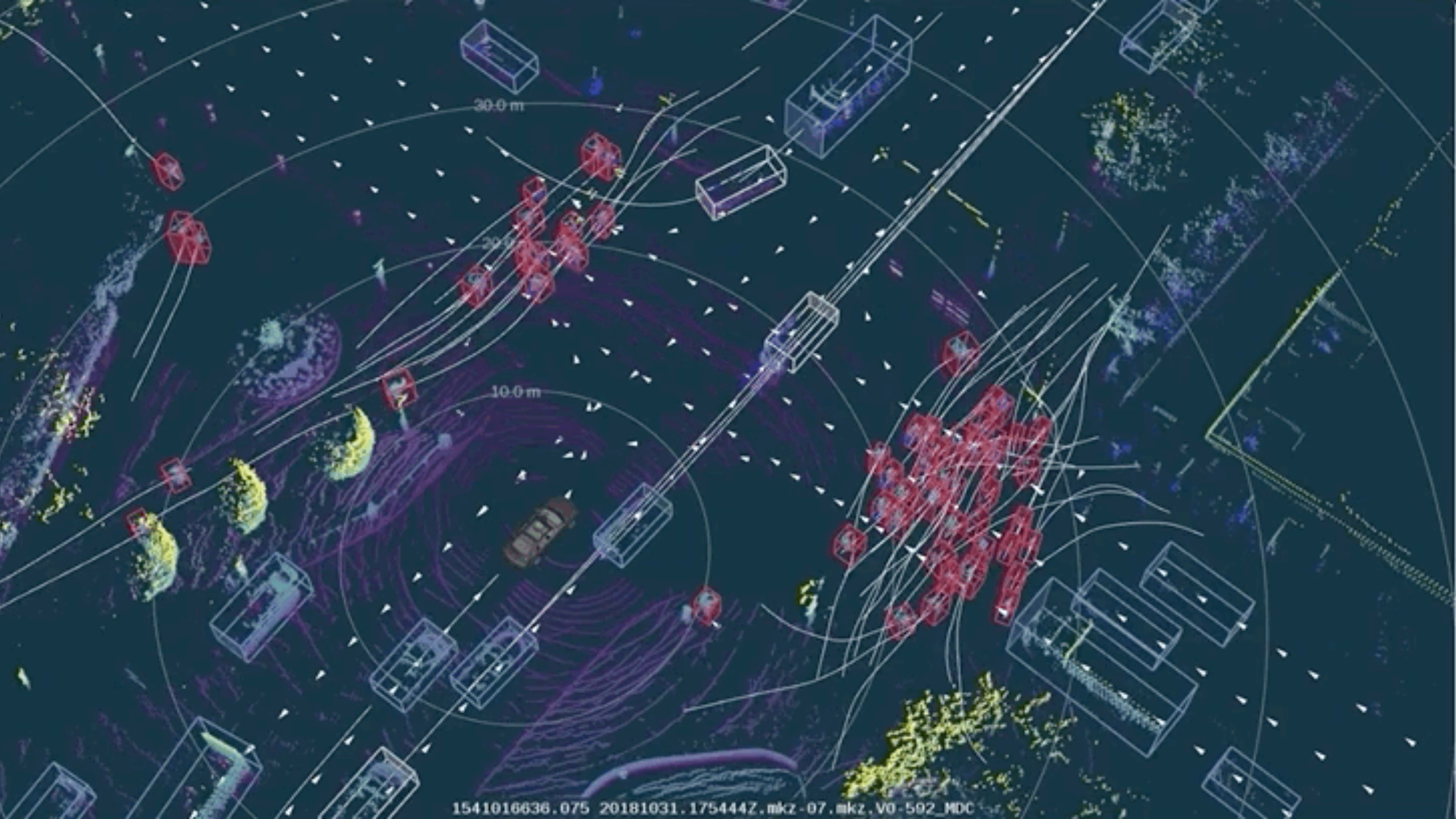
Measure of match










30.0 m
20.0 m
10.0 m
1541016636.075 20181031.175444Z.mkz-07.mkz.V0-592_MDC