AA 274
Principles of Robotic Autonomy

Closed-loop control and robotic sensors

Stanford University

ASL
Autonomous Systems Lab

Closed-loop control

- We want to find

$$\mathbf{u}^*(t) = \pi(\mathbf{x}(t), t)$$

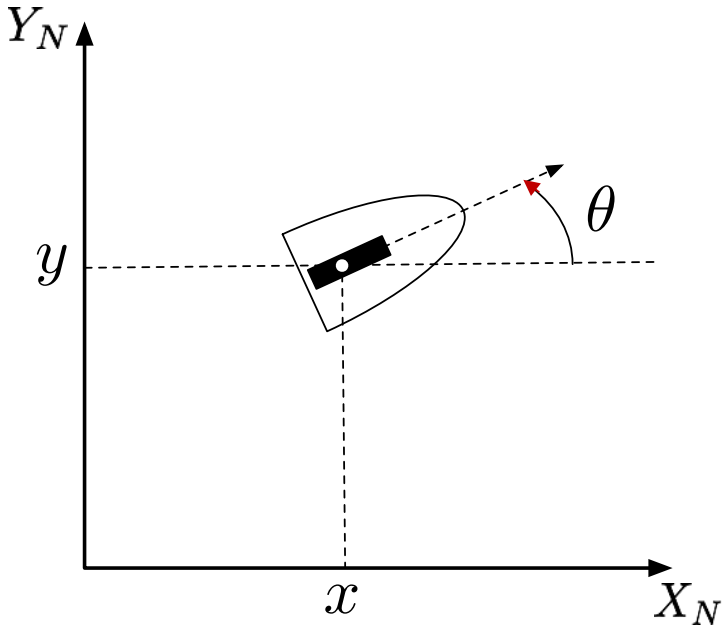
- Main techniques:
  - Hamilton–Jacobi–Bellman equation, dynamic programming
  - Lyapunov analysis

- These topics are covered in a more advanced class (AA 203), here we will only consider an example

Closed-loop control: posture regulation

- Consider a differential drive mobile robot

$$\dot{x}(t) = V(t)\cos(\theta(t))$$
$$\dot{y}(t) = V(t)\sin(\theta(t))$$
$$\dot{\theta}(t) = \omega(t)$$

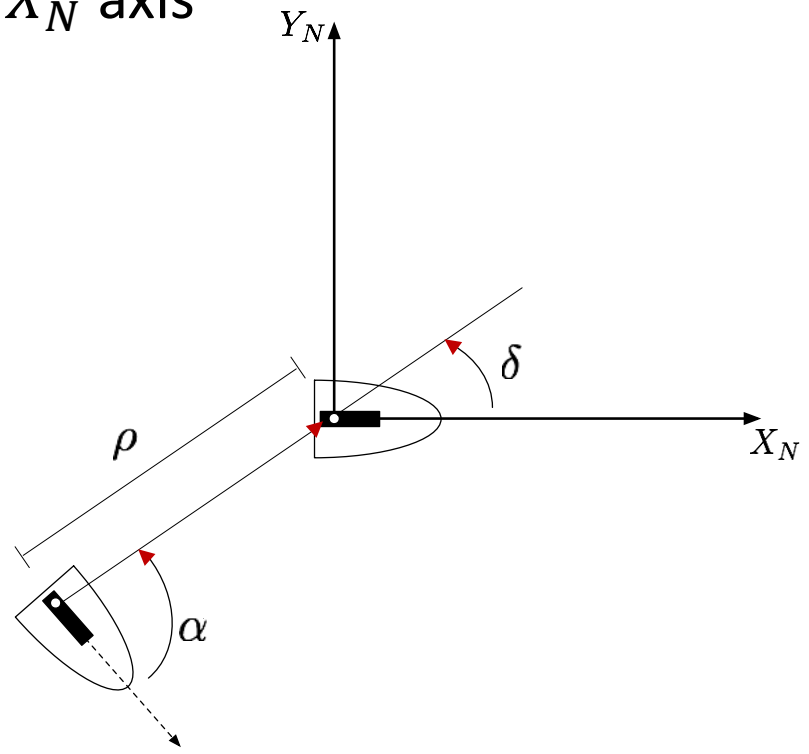


- Inputs: *V* (linear velocity of the wheel) and ω (angular velocity around the vertical axis)
- Goal: drive the robot to the origin *[0, 0, 0]*

Control based on polar coordinates

- Polar coordinates
  - ρ: distance of the reference point of the unicycle from the goal
  - α: angle of the pointing vector to the goal w.r.t. the unicycle main axis
  - δ: angle of the same pointing vector w.r.t. the X_N axis

- Coordinate transformation
  - $\rho = \sqrt{x^2 + y^2}$
  - $\alpha = \text{atan2}(y, x) - \theta + \pi$
  - $\delta = \alpha + \theta$



Equations in polar coordinates

- In polar coordinates, the unicycle equations become

$$\dot{\rho}(t) = -V(t)\cos(\alpha(t))$$
$$\dot{\alpha}(t) = V(t)\frac{\sin(\alpha(t))}{\rho(t)} - \omega(t)$$
$$\dot{\delta}(t) = V(t)\frac{\sin(\alpha(t))}{\rho(t)}$$

- In order to achieve the goal posture, variables (ρ, α, δ) should all converge to zero.

Control law

- Closed-loop control law (Problem 3 in pset):

$$V = k_1 \rho \cos(\alpha)$$

$$\omega = k_2 \alpha + k_1 \frac{\sin(\alpha)\cos(\alpha)}{\alpha}(\alpha + k_3 \delta),$$

- If $k_1, k_2, k_3 > 0$, then closed-loop system is globally asymptotically driven to the posture $(0,0,0)$!

- For more details, see M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino (1995). Closed loop steering of unicycle like vehicles via Lyapunov techniques. IEEE Robotics & Automation Magazine.

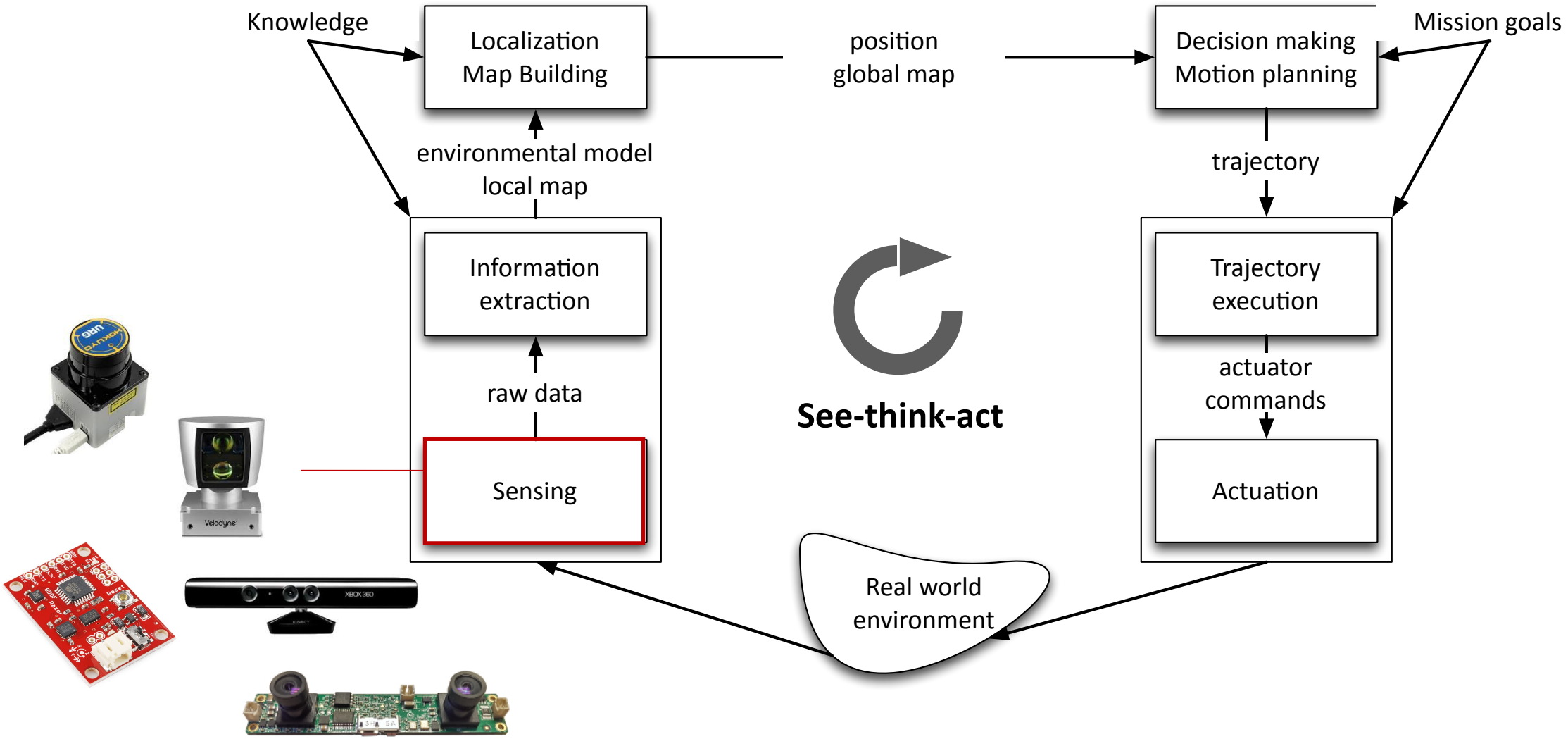
Trajectory tracking

- We aim at a two-degree of freedom design

$$\mathbf{u}^*(t) = \mathbf{u}_d(t) + \pi(\mathbf{x}(t), \mathbf{x}(t) - \mathbf{x}_d(t))$$

- In practice, good compromise between computational tractability and robustness
- For reference trajectory, one can use previous open-loop techniques
- For closed-loop policy (Problem 4 in pset)
  - Geometric (e.g., pursuit) strategies
  - Linearization (either approximate or exact) + linear structure
  - Non-linear control
  - Optimization-based techniques (e.g., MPC)

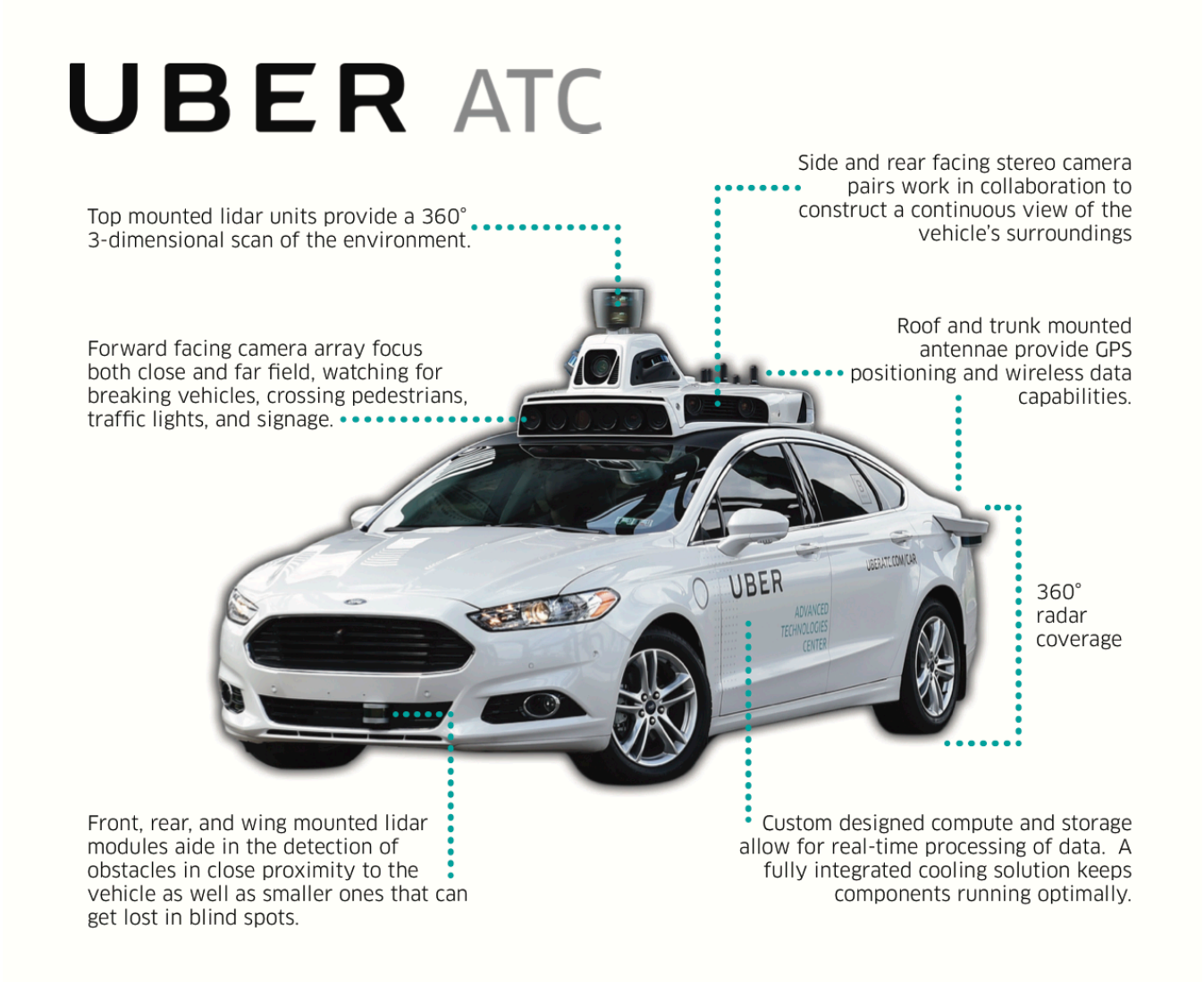
Sensors for mobile robots



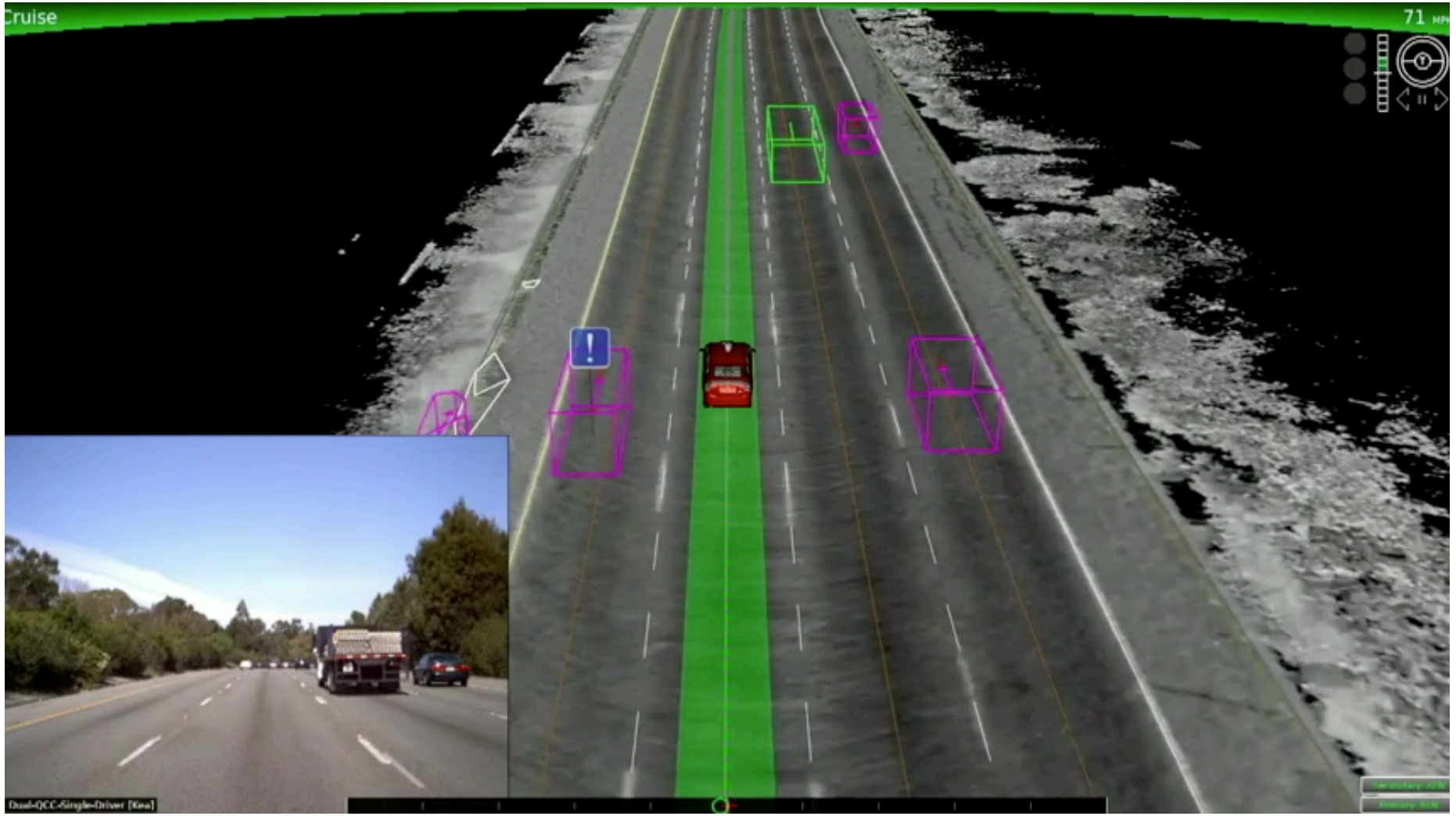
Sensors for mobile robots

- Aim
  - Learn about key performance characteristics for robotic sensors
  - Learn about a full spectrum of sensors, e.g. proprioceptive / exteroceptive, passive / active
- Readings
  - SNS: 4.1

Example: self-driving cars



A self-driving car in action



Classification of sensors

- Proprioceptive: measure values internal to the robot
  - E.g.: motor speed, robot arm joint angles, and battery voltage
- Exteroceptive: acquire information from the robot's environment
  - E.g.: distance measurements and light intensity

- Passive: measure ambient environmental energy entering the sensor
  - Challenge: performance heavily depends on the environment
  - E.g.: temperature probes and cameras
- Active: emit energy into the environment and measure the reaction
  - Challenge: might affect the environment
  - E.g.: ultrasonic sensors and laser rangefinders

Sensor performance: design specs

- Dynamic range: ratio between the maximum and minimum input values (for normal sensor operation)
- Resolution: minimum difference between two values that can be detected by a sensor
- Linearity: whether or not the sensor's output response depends linearly on the input
- Bandwidth or frequency: speed at which a sensor provides readings (in Hertz)

Sensor performance: in situ specs

- Sensitivity: ratio of output change to input change
- Cross-sensitivity: sensitivity to quantities that are unrelated to the target quantity
- Error: difference between the sensor output m and the true value v

$$\text{error} \coloneqq m - v$$

- Accuracy: degree of conformity between the sensor's measurement and the true value

$$\text{accuracy} \coloneqq 1 - |\text{error}|/v$$

- Precision: reproducibility of the sensor results

Sensor errors

- Systematic errors: caused by factors that can in theory be modeled; they are deterministic
  - E.g.: calibration errors
- Random errors: cannot be predicted with sophisticated models; they are stochastic
  - E.g.: spurious range-finding errors
- Error analysis: performed via a probabilistic analysis
  - Common assumption: symmetric, unimodal (and often Gaussian) distributions; convenient, but often a coarse simplification
  - Error propagation characterized by the *error propagation law*

An ecosystem of sensors

- Encoders
- Heading sensors
- Accelerometers and IMU
- Beacons
- Active ranging
- Cameras

Encoders

- Encoder: an electro-mechanical device that converts motion into a sequence of digital pulses, which can be converted to relative or absolute position measurements
  - proprioceptive sensor
  - can be used for robot localization

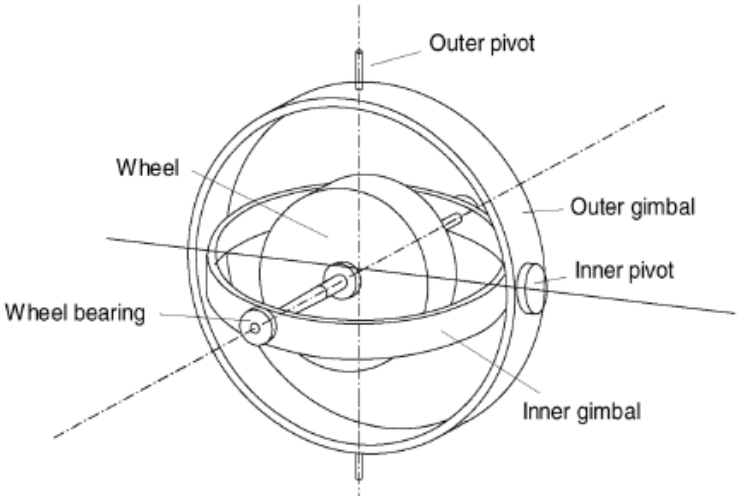
Wheel encoder
Credit: Pololu

- Fundamental principle of optical encoders: use a light shining onto a photodiode through slits in a metal or glass disc

Credit: Honest Sensor

Heading sensors

- Used to determine robot's orientation, it can be:
  1. Proprioceptive, e.g., gyroscope (heading sensor that preserves its orientation in relation to a fixed reference frame)
  2. Exteroceptive, e.g., compass (shows direction relative to the geographic cardinal directions)
- Fusing measurements with velocity information, one can obtain a position estimate (via integration) -> *dead reckoning*
- Fundamental principle of mechanical gyroscopes: angular momentum associated with a spinning wheel keeps the axis of rotation inertially stable



Credit: SNS

Accelerometer and IMU

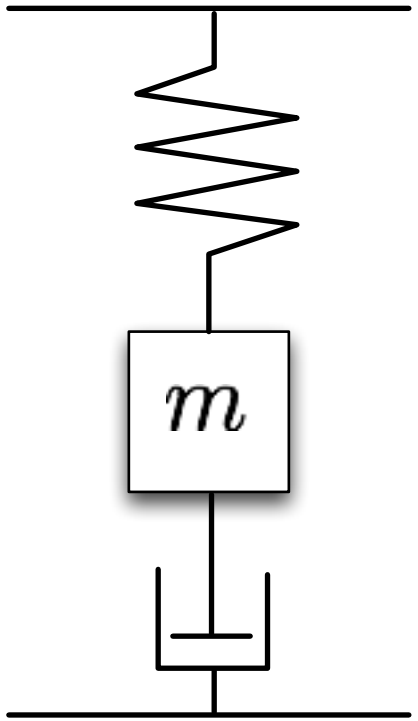
- Accelerometer: device that measures all external forces acting upon it
- Mechanical accelerometer: essentially, a spring-mass-damper system

$$F_{\text{applied}} = m\ddot{x} + c\dot{x} + kx$$

with *m* mass of proof mass, *c* damping coefficient, *k* spring constant; in steady state

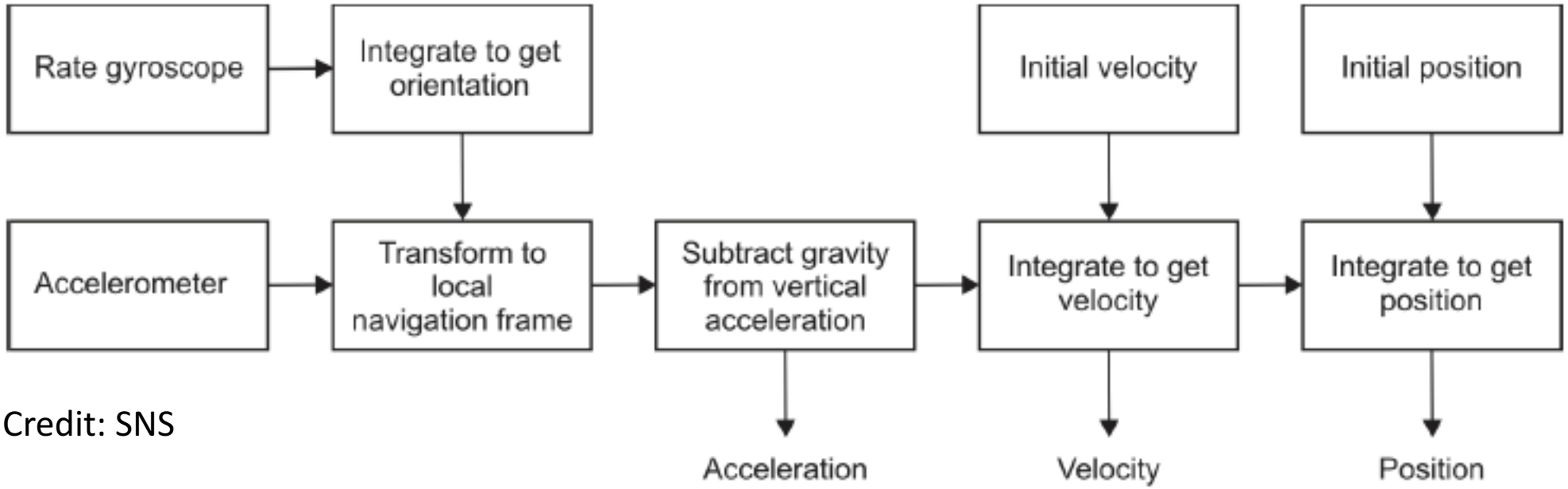
$$a_{\text{applied}} = \frac{kx}{m}$$

- Modern accelerometers use MEMS (cantilevered beam + proof mass); deflection measured via *capacitive* or *piezoelectric* effects



Inertial Measurement Unit (IMU)

- Definition: device that uses gyroscopes and accelerometers to estimate the relative position, orientation, velocity, and acceleration of a moving vehicle with respect to an inertial frame
- *Drift* is a fundamental problem: to cancel drift, periodic references to external measurements are required



Credit: SNS

Beacons

- Definition: signaling devices with precisely known positions
- Early examples: stars, lighthouses
- Modern examples: GPS, motion capture systems

Active ranging

- Provide direct measurements of distance to objects in vicinity
- Key elements for both localization and environment reconstruction
- Main types:
    1. Time-of-flight active ranging sensors (e.g., ultrasonic and laser rangefinder)

Credit: https://electrosome.com/hc-sr04-ultrasonic-sensor-pic/

    2. Geometric active ranging sensors (optical triangulation and structured light)

Time-of-flight active ranging

- Fundamental principle: time-of-flight ranging makes use of the propagation of the speed of sound or of an electromagnetic wave
- Travel distance is given by

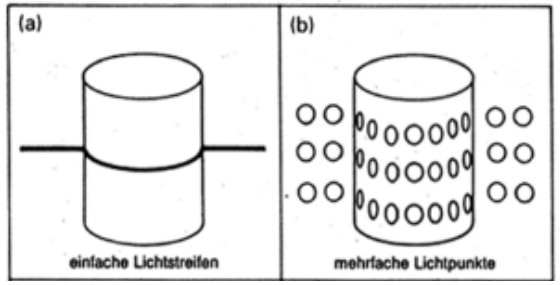
$$d = c\,t$$

where *d* is the distance traveled, *c* is the speed of the wave propagation, and *t* is the time of flight

- Propagation speeds:
  - Sound: 0.3 m/ms
  - Light: 0.3 m/ns
- Performance depends on several factors, e.g., uncertainties in determining the exact time of arrival and interaction with the target

Geometric active ranging

- Fundamental principle: use geometric properties in the measurements to establish distance readings
- The sensor projects a known light pattern (e.g., point, line, or texture); the reflection is captured by a receiver and, together with known geometric values, range is estimated via triangulation
- Examples:
  - Optical triangulation (1D sensor)
  - Structured light (2D and 3D sensor)



Several other sensors are available

- Classical, e.g.:
  - Radar (possibly using Doppler effect to produce velocity data)
  - Tactile sensors

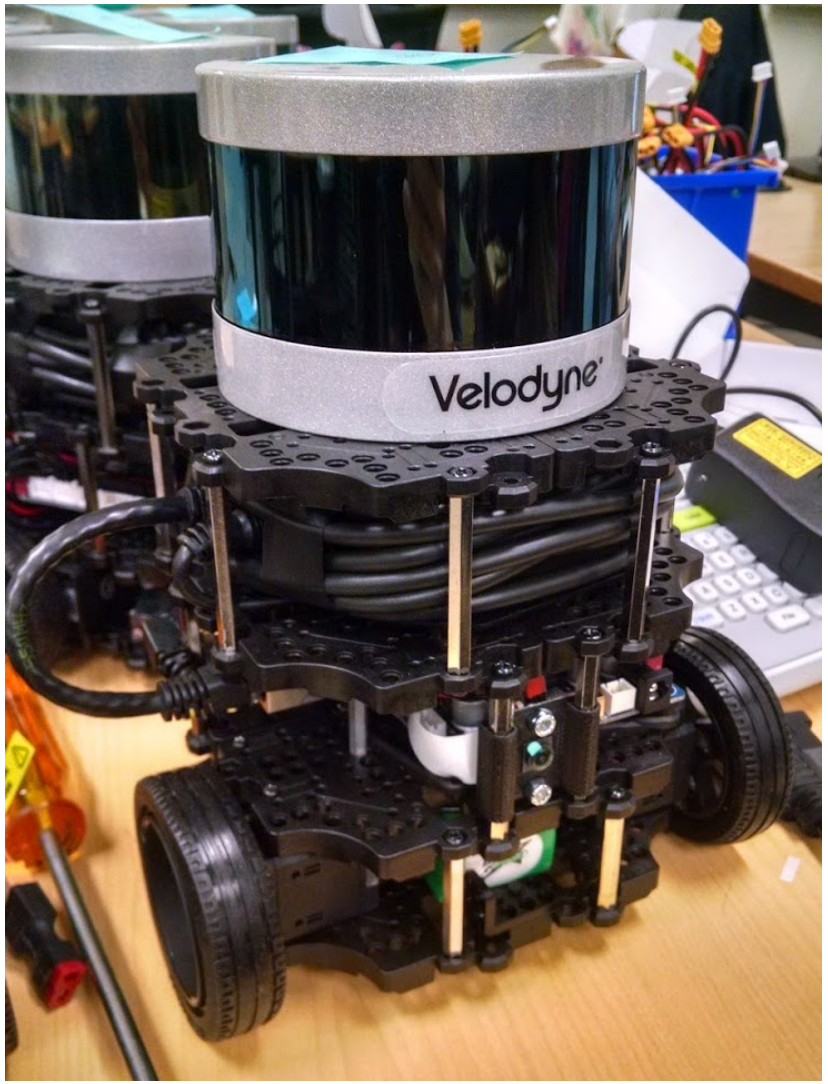
- Emerging technologies:
  - Artificial skins
  - Neuromorphic cameras

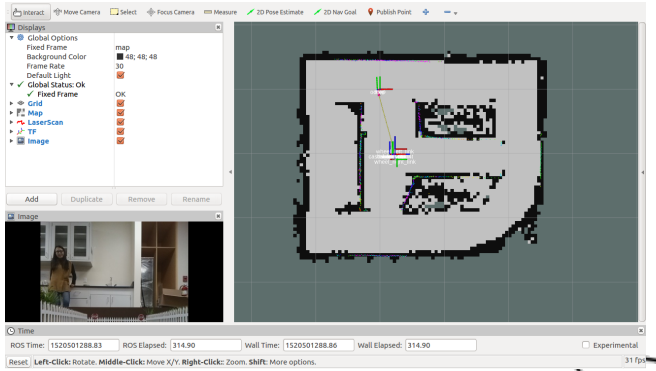
Next time: computer vision

AA274 Final Project
Winter 2019

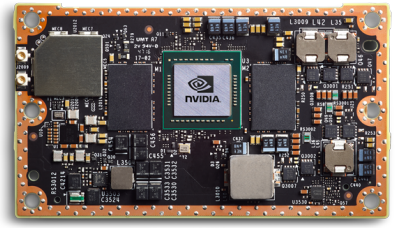
Turtlebots!

- Turtlebot 3 Burger equipped with:
  - Velodyne PUCK (VLP16) LIDAR
  - Raspberry Pi Camera Module v2 + Raspberry Pi
  - Jetson TX2 (new this year)
  - Differential drive
- Maximum velocity is ~20cm/s
- Embodiment of all the “pillars of autonomy” that you will learn in this class.

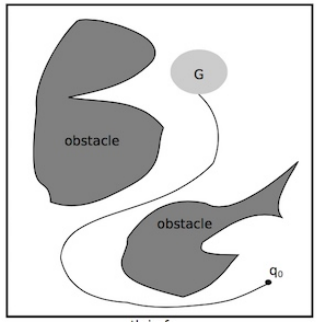




HW3



HW4



Localization
Map Building

position
global map

Decision making
Motion planning

Mission goals

environmental model
local map

trajectory

Information
extraction

raw data

Sensing

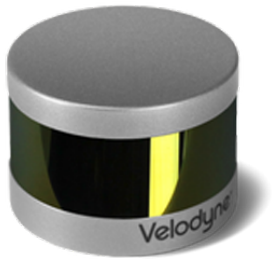
See-think-act

Trajectory
execution

actuator
commands

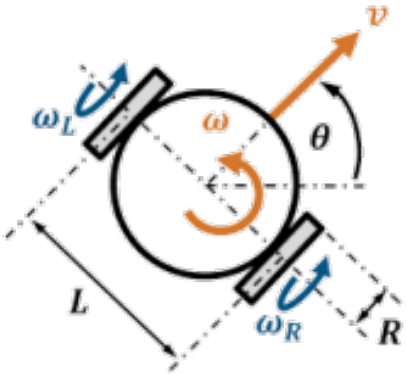
Actuation

HW2



Mock environment

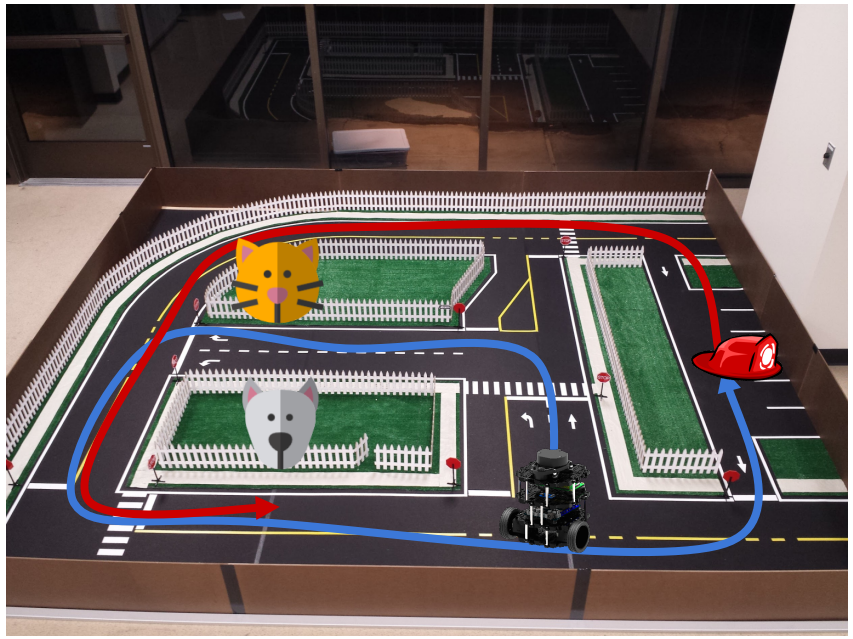
HW1



Mission Outline -- Explore and recover!

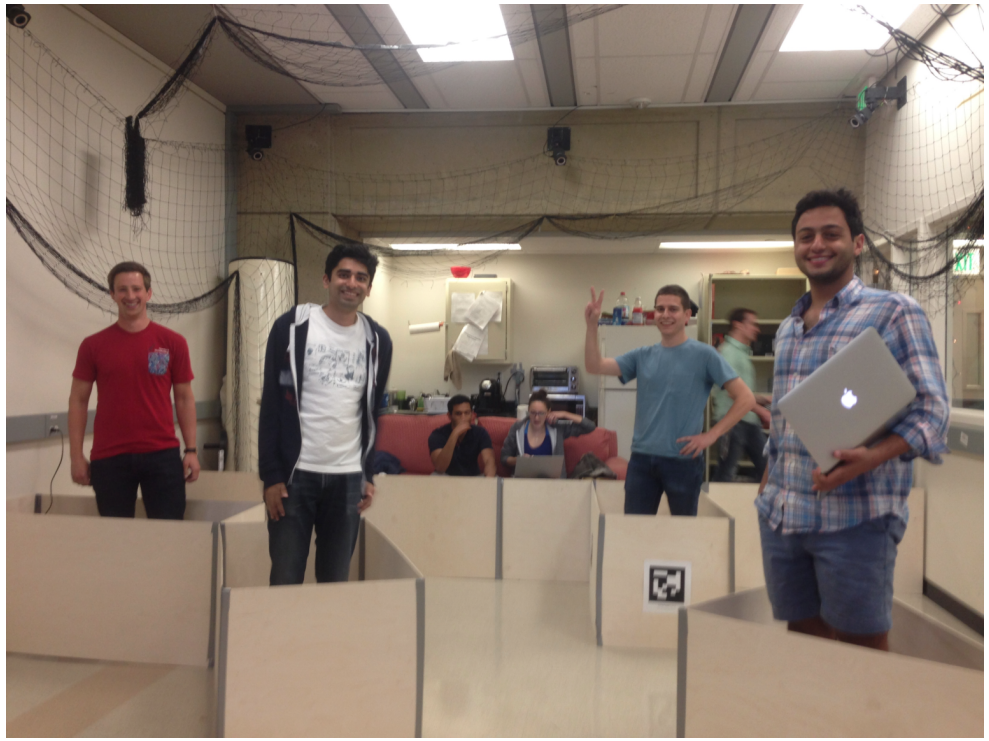
- Explore the environment to locate “lost items”
  - Navigate the city streets without colliding with any obstacles
  - Map out locations of points of interests

- Recover “lost items” with your “team”
  - Negotiate onboarding of your team
  - Autonomously lead your team back through the environment to stop at each of point of interest.



“The City” (2018)

2017: Exploration on Mars

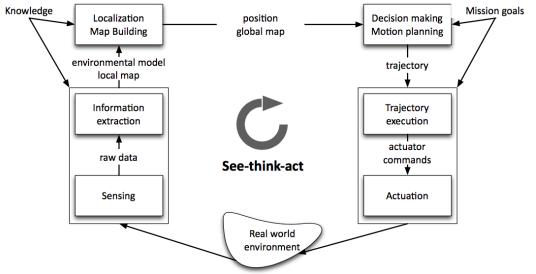


Explore and excavate points of interests in a particular order.

2018: Animal Rescue

Finding lost animals in a city, and rescuing as many of them as possible.

Mission Outline



- Explore the environment to locate "lost items"
  - Navigate the city streets without colliding with any obstacles
  - Map out locations of points of interests

- Recover "lost items" with your "team"
  - Negotiate onboarding of your team
  - Autonomously lead your team back through the environment to stop at each of point of interest.

SLAM

planning + control

object detection

ROS pub/sub

planning + control

supervisor/
state machine

Project Goals

- Reading/interpreting error messages through the power of the internet
  - Google error messages to see what they mean/what could be causing them
  - Read online documentation to understand how packages work and how to use them

- ROS proficiency
  - **rviz** to see the world through the eyes of your robot (i.e., ROS topics)
  - **Launch files**: Writing/editing launch files so you don't have to constantly manage 10 terminal windows
  - **rqt_plot** for plotting streaming data (e.g., estimated distance to stop sign)
  - **rosbag** for forensic analysis (and/or for a souvenir from this class!)
  - Many others! Including packages you find online!

First steps

- Familiarize you with the Turtlebot hardware
  - Get into groups of about 6.
  - Next homework will require groups to implement the ROS question from the previous homework on the Turtlebots.
  - Attend the ROS support hours Wednesday/Thursday 5 – 7PM Skilling lab.

- More information to come (project page on the course website)