CS123

Programming Your Personal Robot

Part 3: Reasoning Under Uncertainty

3.2 “Where Am I”?”
The Robot Localization Problem

Topics

- Overview of Localization Methods
- Dead Reckoning
- Least Square
- Landmark
- Homework Assignment Part # 3-1 – demo and refine specification

Localization Methods

Two General Approaches:

- Relative (Internal) – relative to “self”
  - Using Proprioceptive sensors such as:
  - odometric (encoder)
  - gyroscopic
- Absolute (External)
  - using “exteroceptive” sensors such as infrared, sonar, laser distance sensor – to measure environment
  - geometric features
  - landmarks

Two SAT Words

pro·pri·o·cep·tive

/ˌprōprēəˈseptiv/

adjective PHYSIOLOGY

relating to stimuli that are produced and perceived within an organism, especially those connected with the position and movement of the body.

ex·ter·o·cep·tive

/ˌekstərōˈseptiv/

adjective PHYSIOLOGY

relating to stimuli that are external to an organism.

Relative "Localization": Dead Reckoning

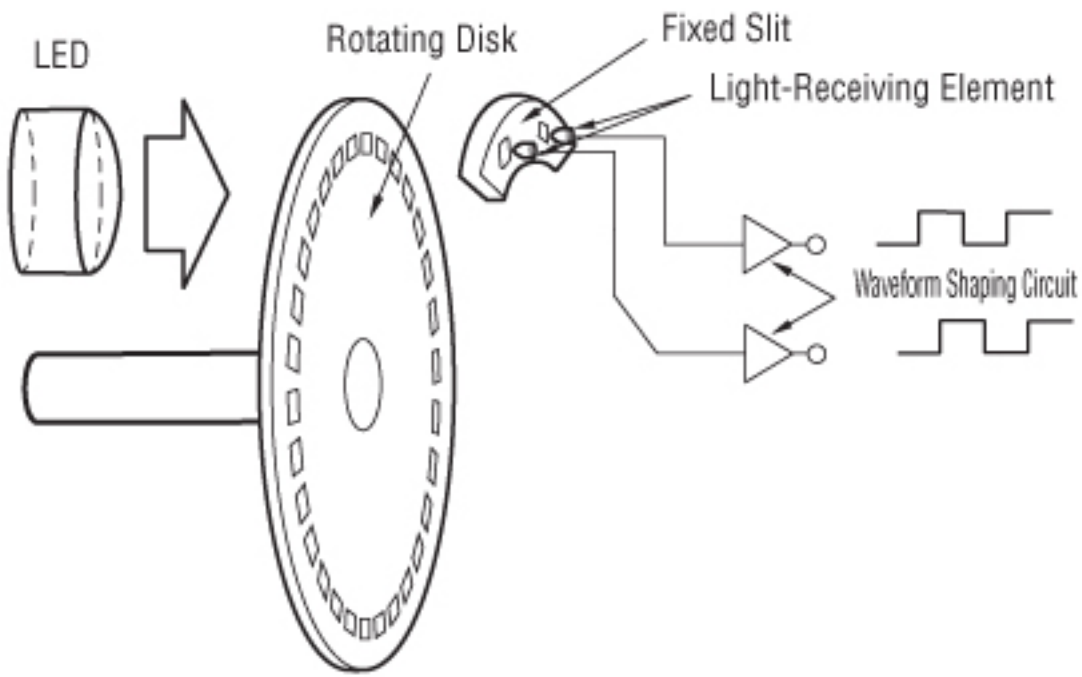
- What is Dead Reckoning
- Encoder
- Various Drive Mechanisms
- Hamster

Dead Reckoning

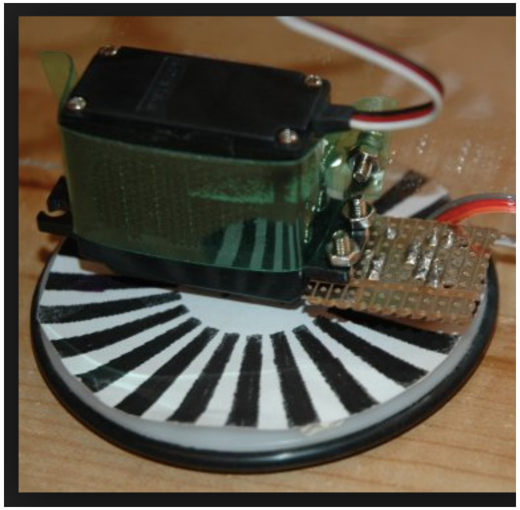
- Calculate current position (and orientation) using a previously determined position and advancing that based on estimated speeds over elapsed time

How Does Encoder Work

- What is an encoder?
- How does it work

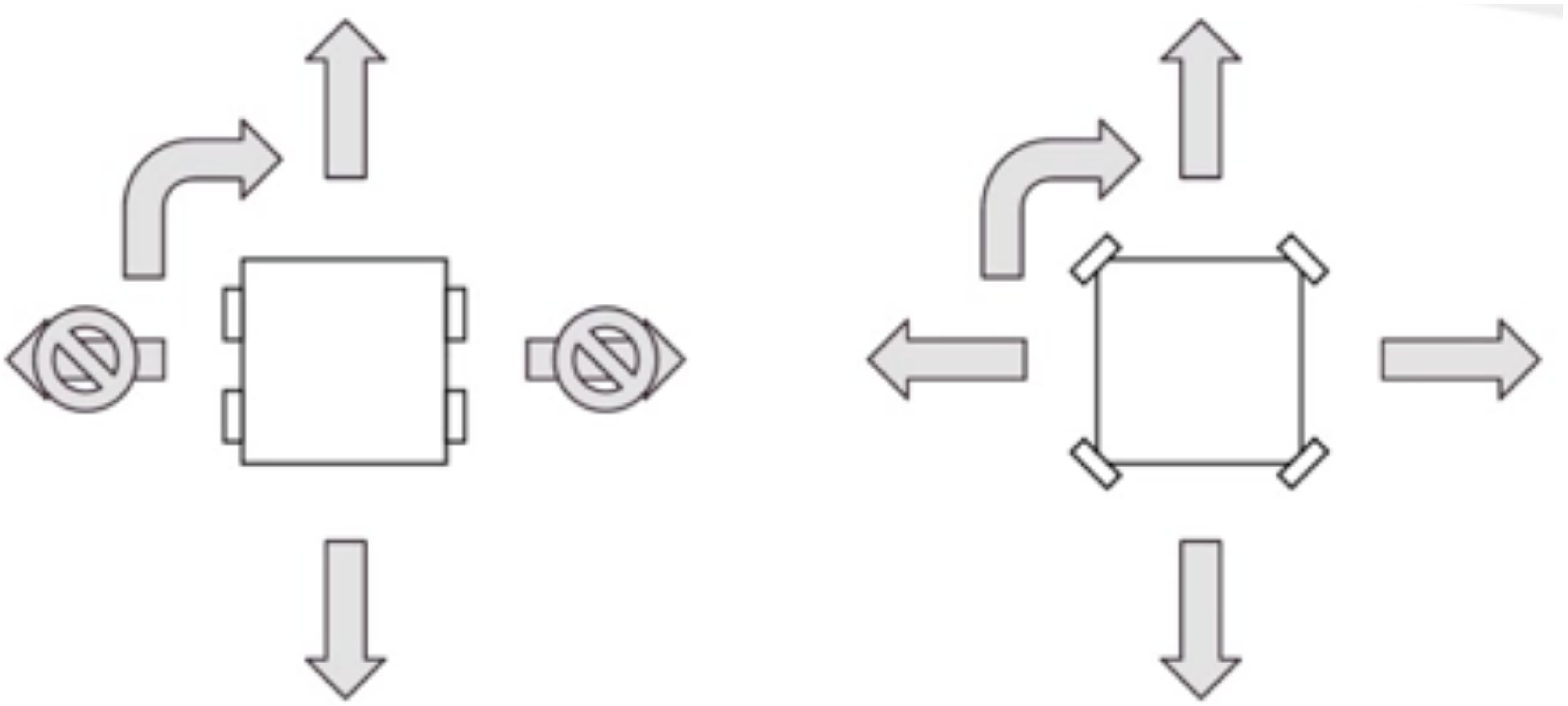


Basic Principle



Home Made Encoder

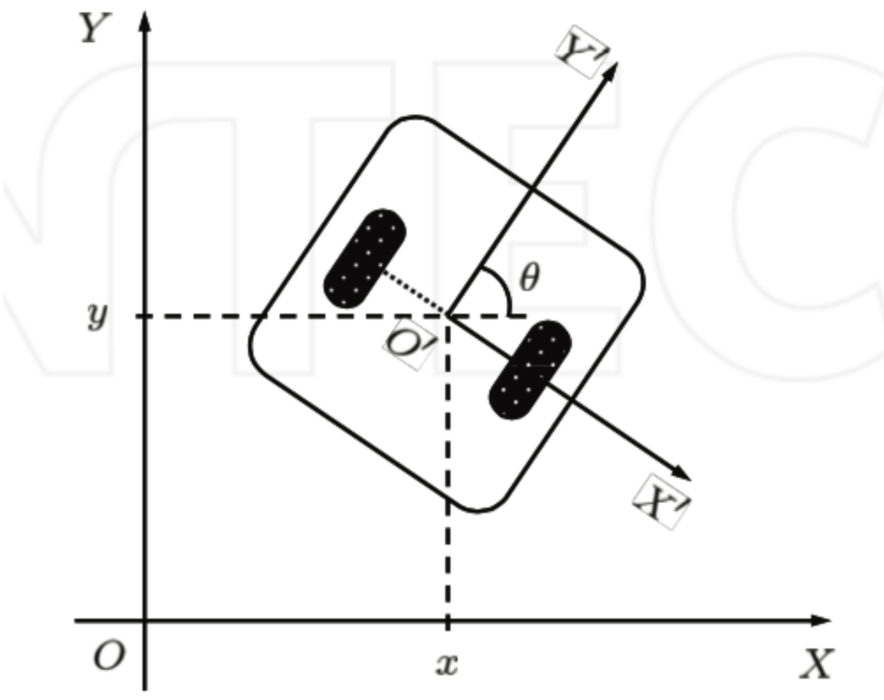
Holonomic and Non-holonomic Drive

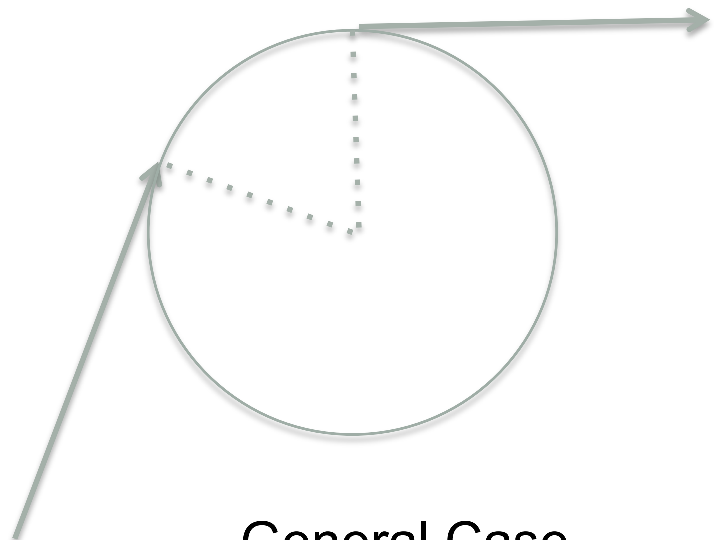


Hamster's Motion

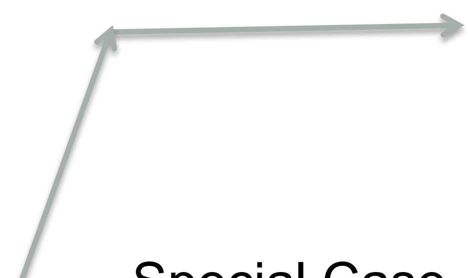
- set_wheel (0, speed_left)
- set_wheel (1, speed_right)
- Different Cases:
  - Speed_left == Speed_right > 0 : going forward straight
  - Speed_left == Speed_right < 0 : going backward straight
  - Speed_left == - Speed_right > 0 : spinning around center right
  - Speed_left > Speed_right >= 0 : turning right
  - …

Hamster’s Motion





General Case



Special Case

Dead Reckoning : Error

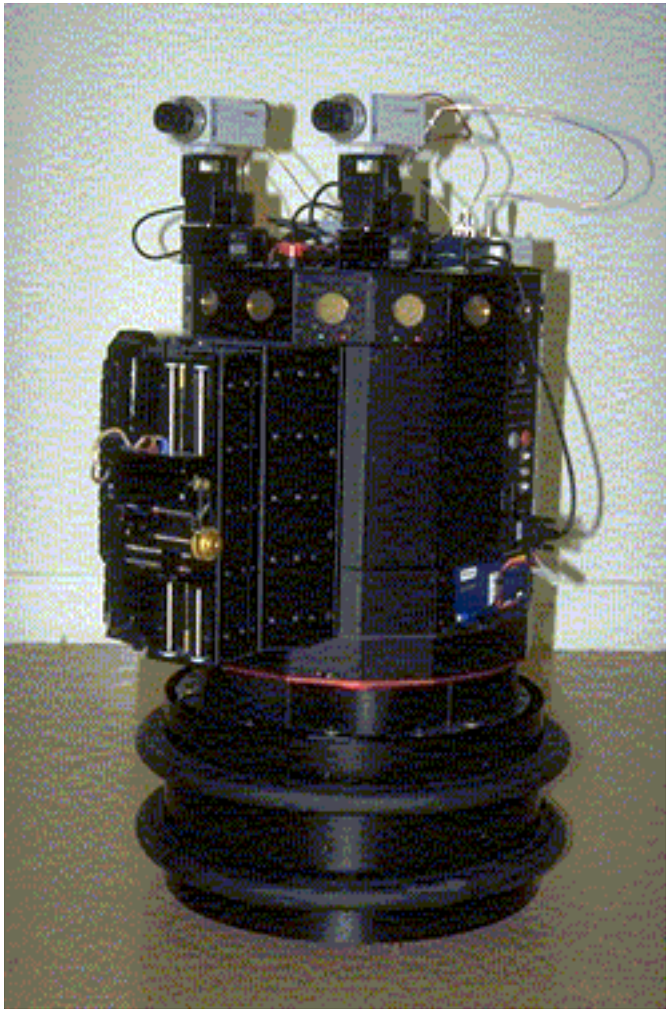
- Encoder error
- Environmental – slippage, sticky floor
- Error is cumulative (unbounded)

“Absolute” Localization

- GPS and Beacons
- Use “external” sensors – “measuring” environment and matching against “map”
- Minimize the difference between measured data and “expected” (predicted) data (from the map)

Various Distance Sensors

- Infrared (intensity)
- Sonar (time of flight)
- Laser (triangulation)
- Stereo vision (passive)
- 3D Sensors (structure light)
  - Prime Sense
  - Softkinectic (Time of flight)

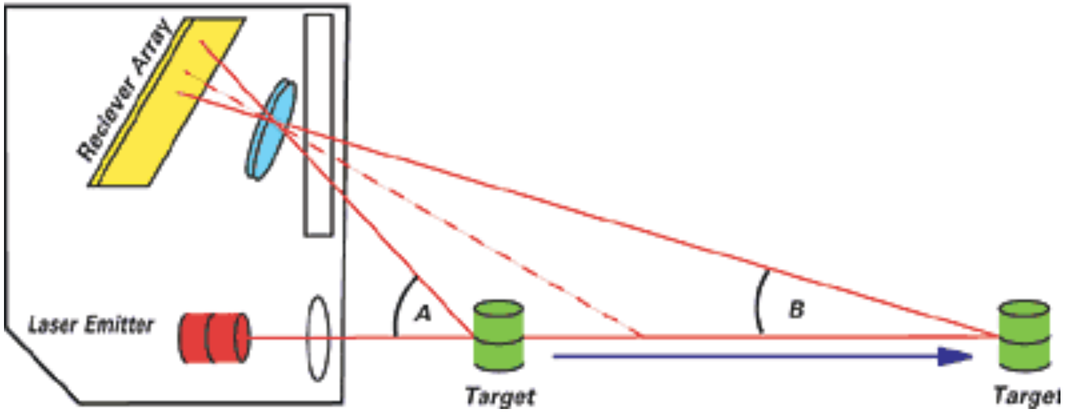


Sonar : Ultrasonic Sensors

- Time of Flight
- Sends out “chirps”
- Listening for “echo”
- Used also extensive for detecting objects in water – like fishing

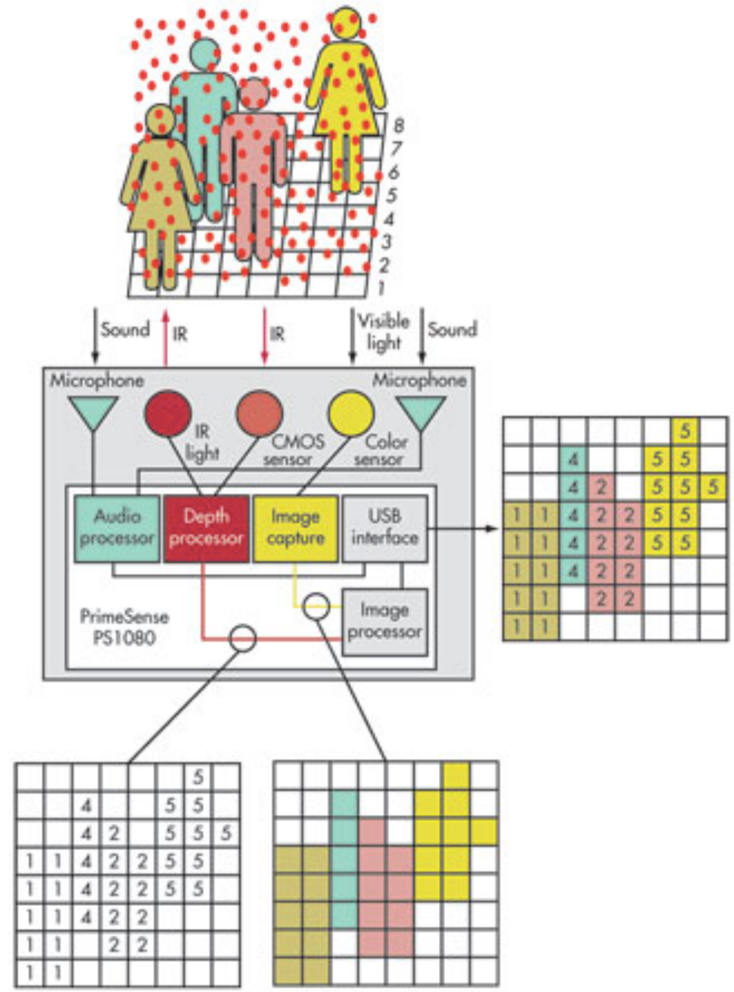
Laser Sensors

- Project light (dot, plane, region) onto object
- detect reflected light on camera
- Triangulation



3D Sensor – Kinect (Prime Sense)

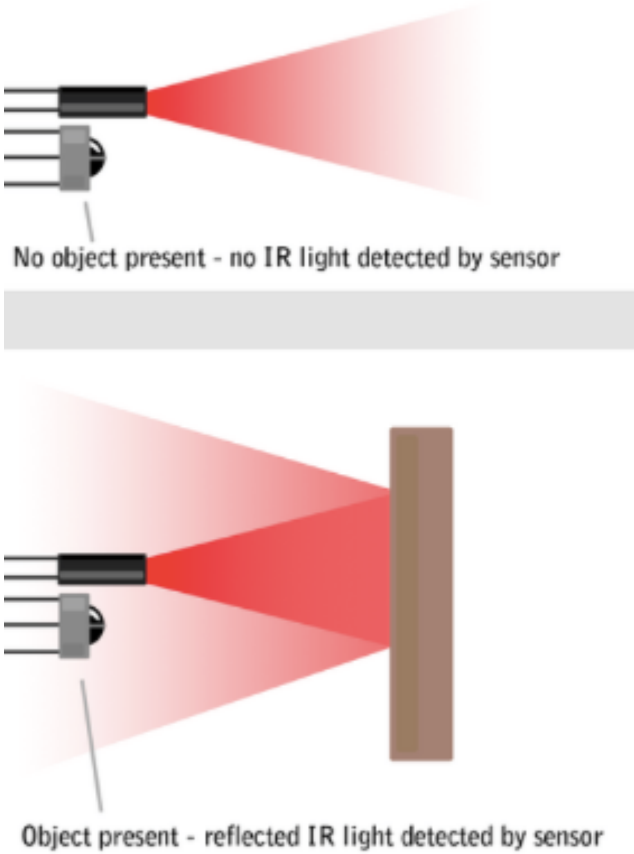
Fig 1. PrimeSense technology is the basis behind Microsoft's Kinect (a) and its own sensor (b).

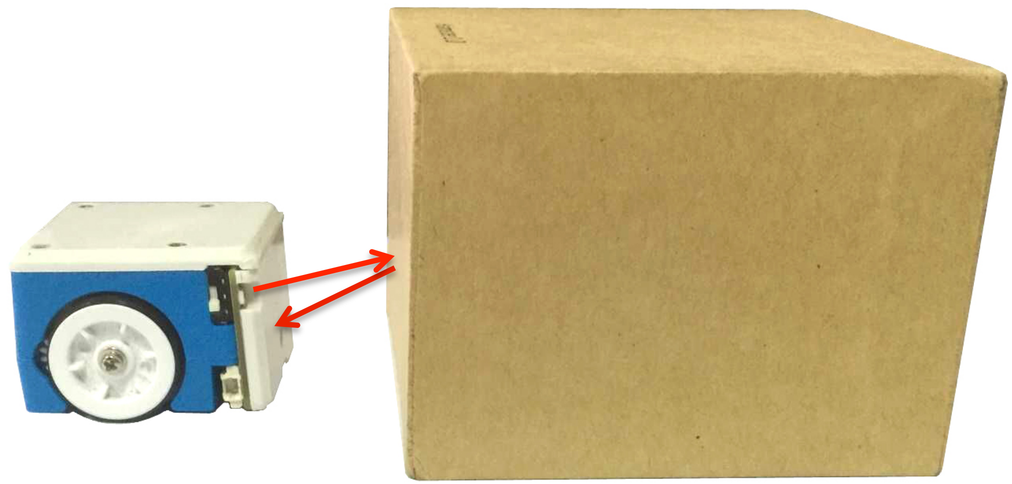


Hamster Sensors

- Distance Sensors (IR based) – detect distance to object
- Floor Sensors (IR based) – detect “color” of floor

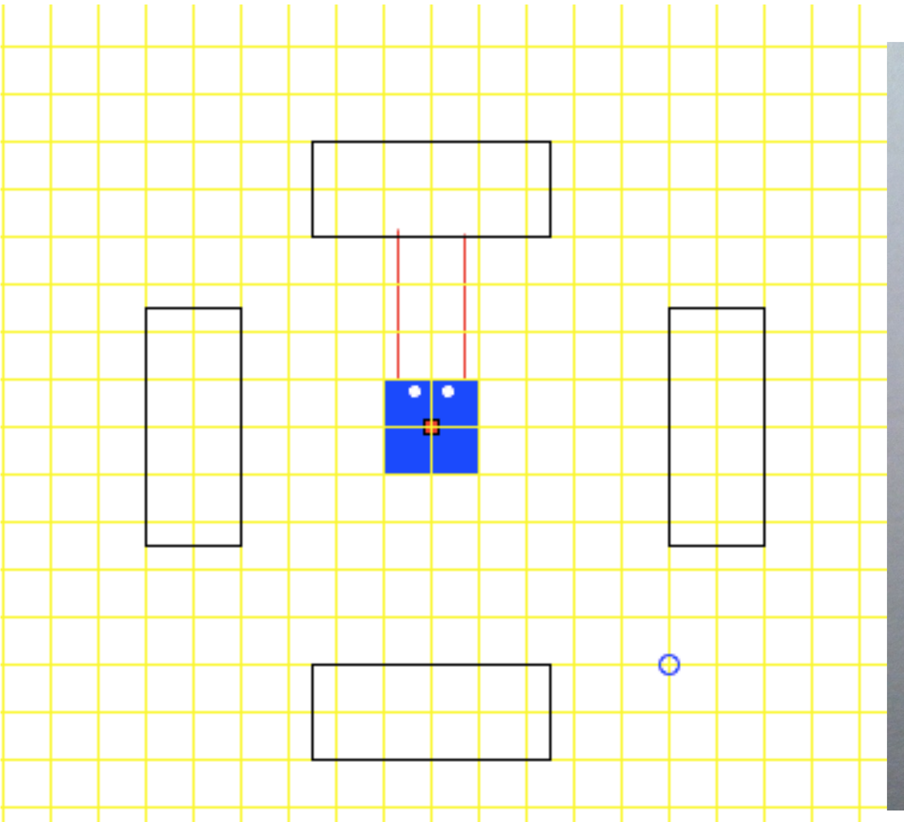
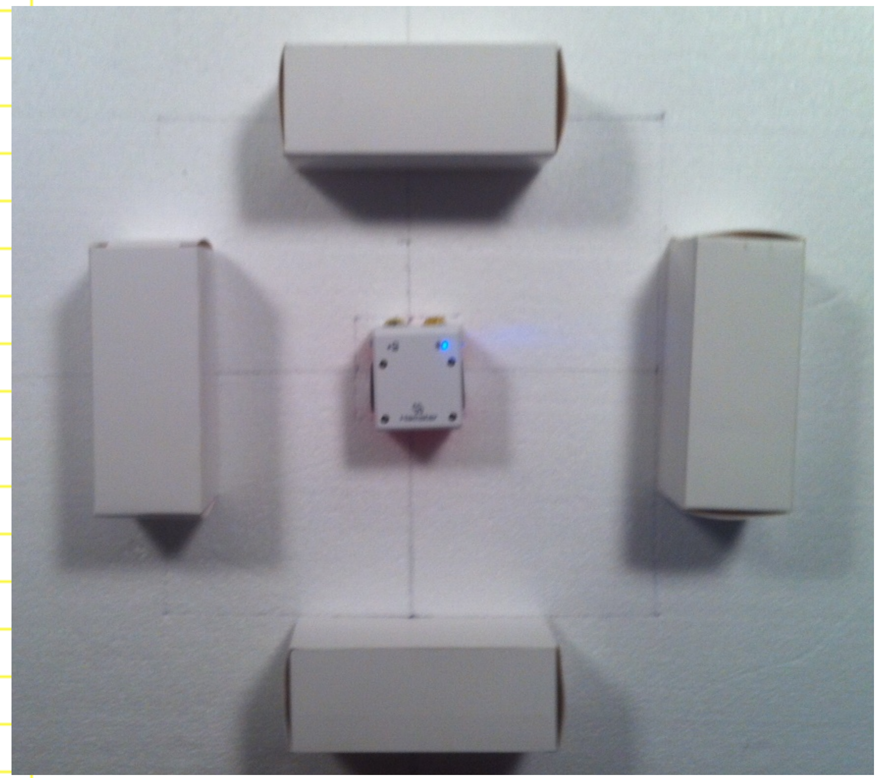
Hamster “Proximity” Sensors



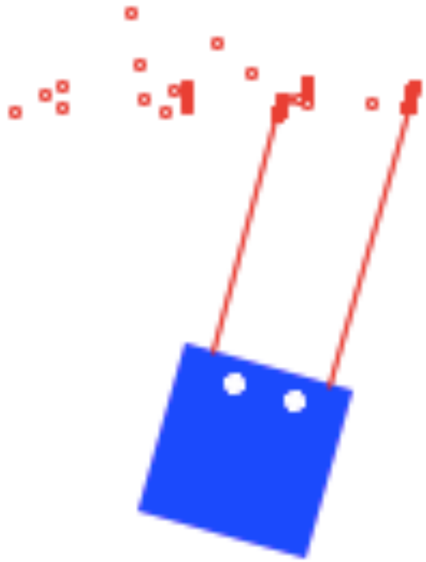


Left and Right Proximity Sensors

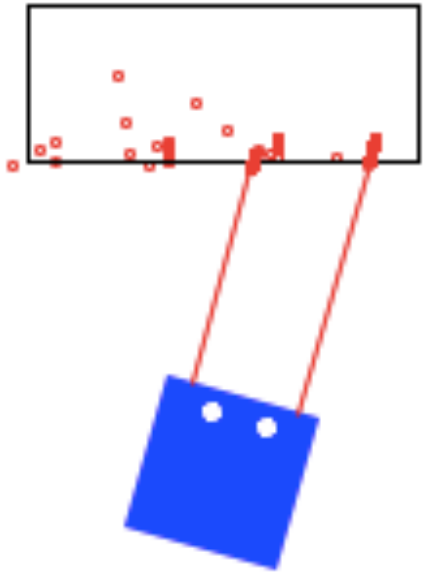
Localization Using Proximity Sensors

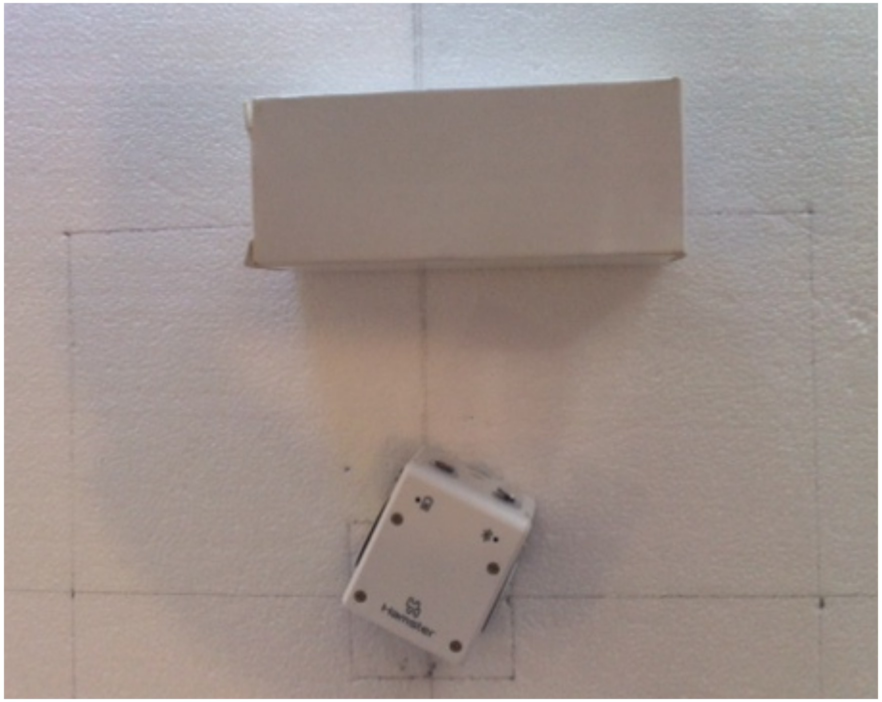



What Do Hamster Proximity Sensors See

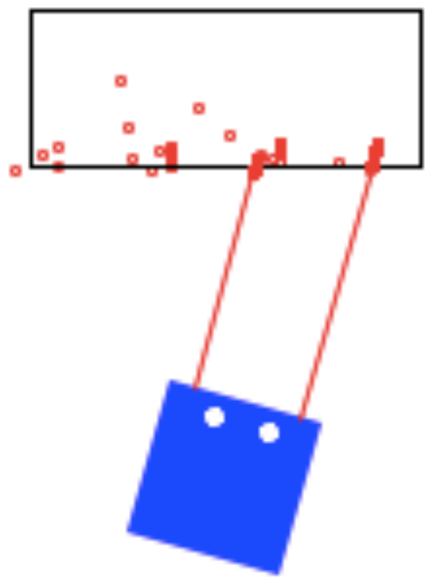


Model Of The Environment



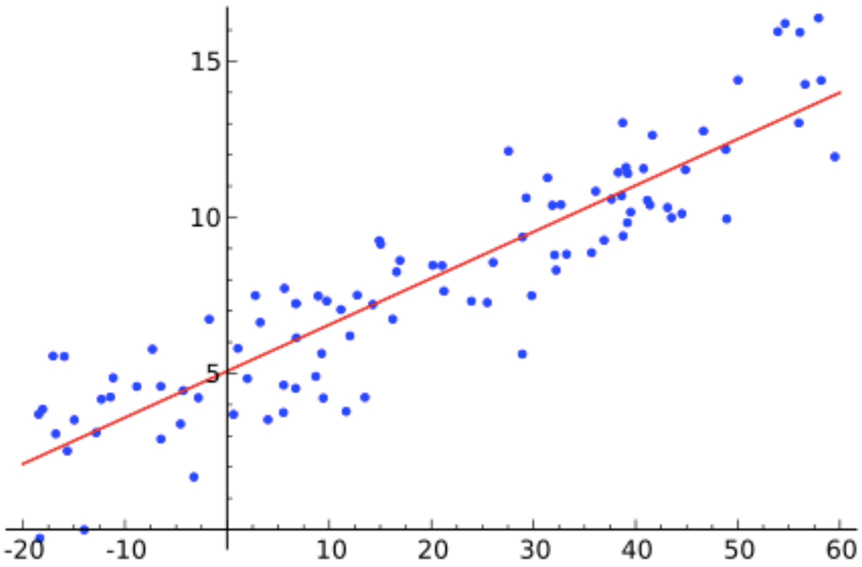


Making Sense of Noisy Data



Linear Least Square (Fit)

- For a given set of points (x_i, y_i)
- Find m,c such that the sum of distances of these points to the line y = mx +c is minimized

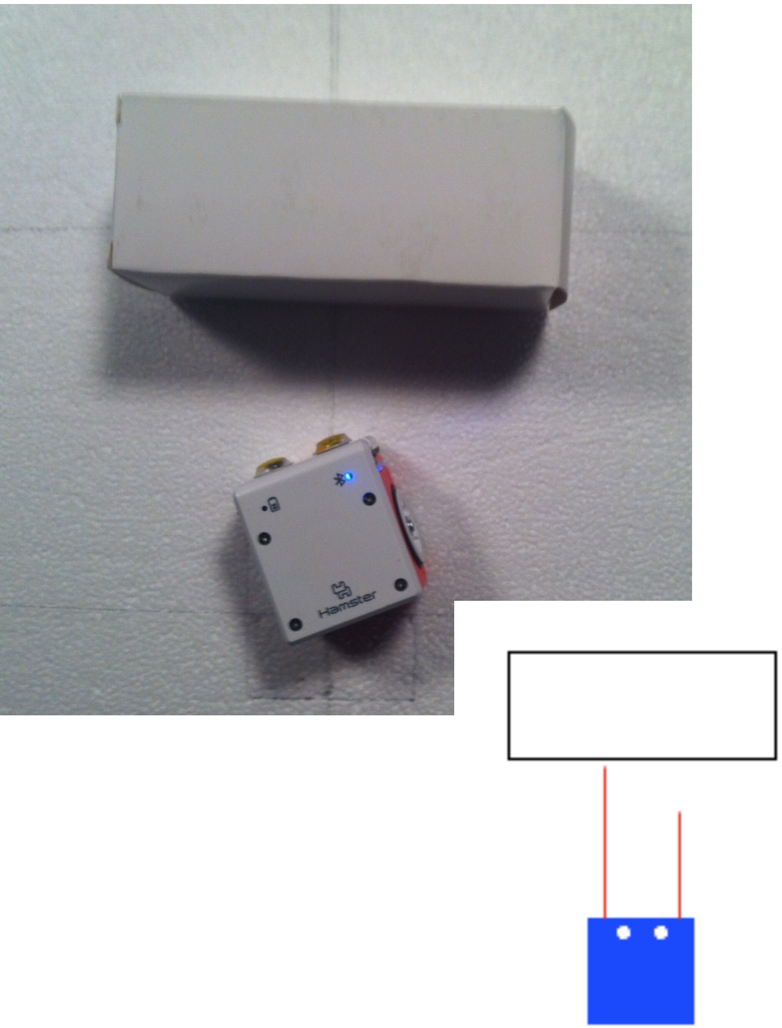


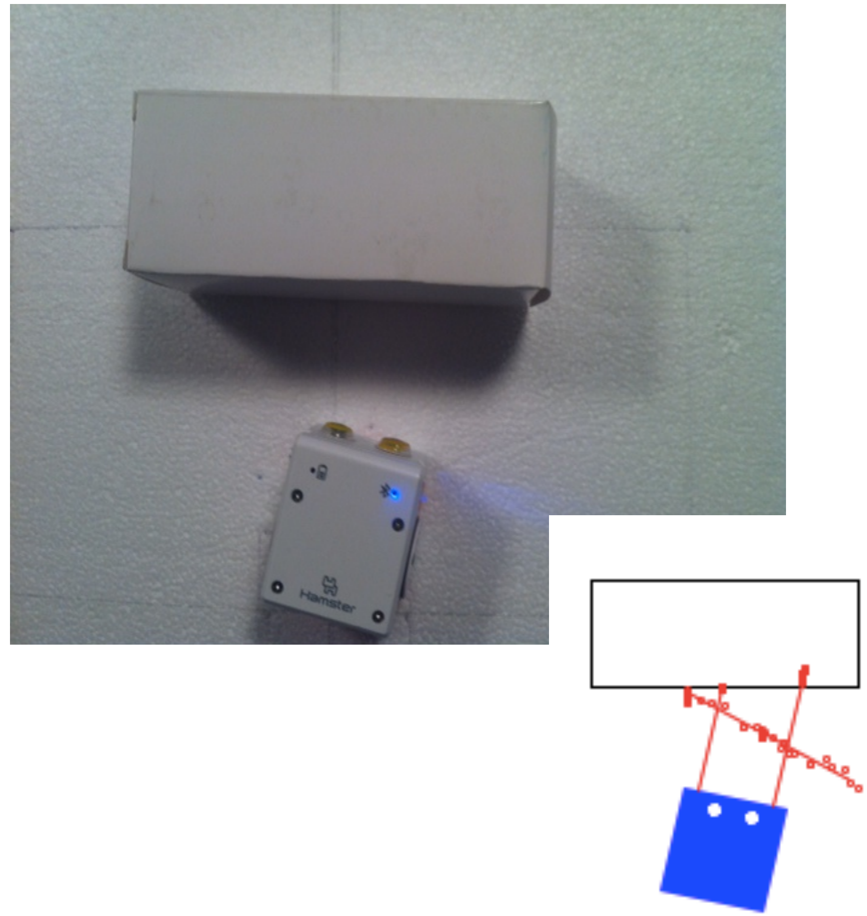
Python Pynum

tk_hamster_localize_lsr.py × least_square.py ×

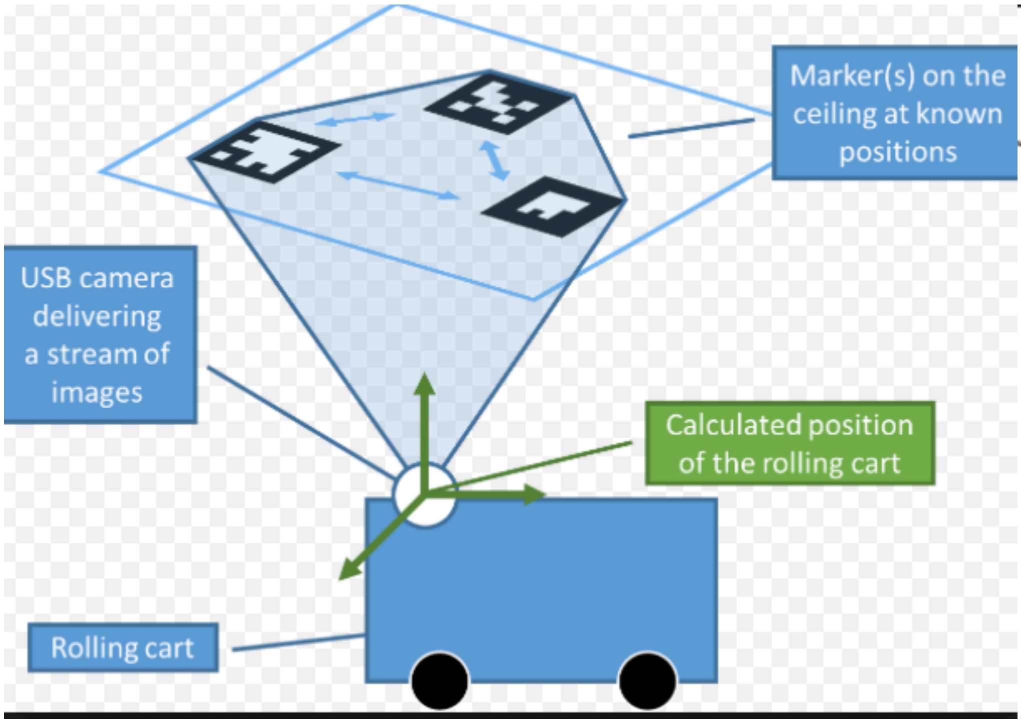
```python
import numpy as np

x = np.array([0, 1, 2, 3])
#y = np.array([-1, 0.2, 0.9, 2.1])
y = np.array([-1, 0.2, 0.9, 2.1])

A = np.vstack([x, np.ones(len(x))]).T
print A

m, c = np.linalg.lstsq(A, y)[0]
print m, c
```

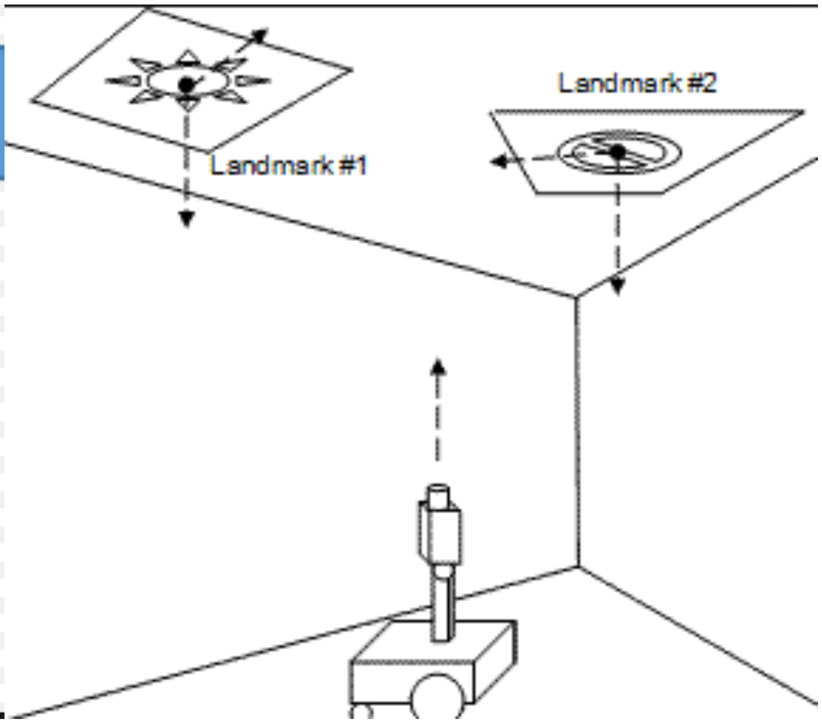
Localization Of Hamster





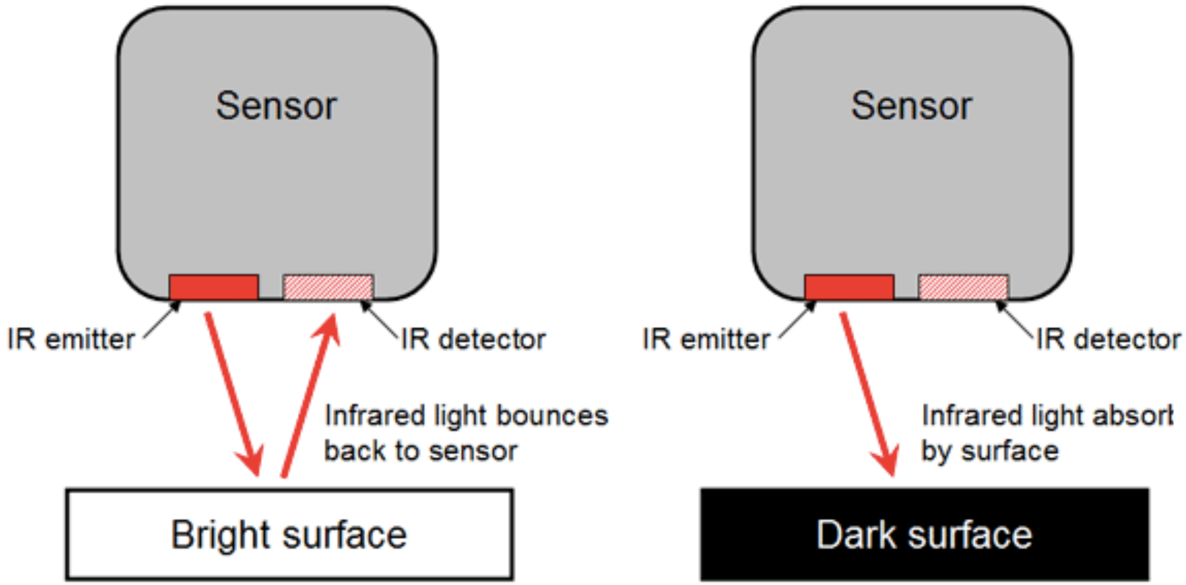
Localization Using Special Landmarks



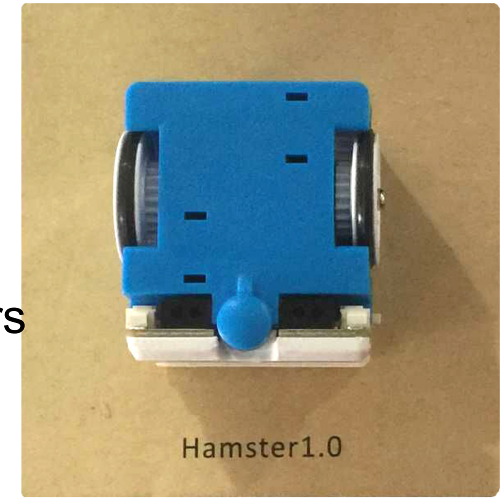


Patterns on ceiling are often used landmarks

Hamster "Floor" Sensors

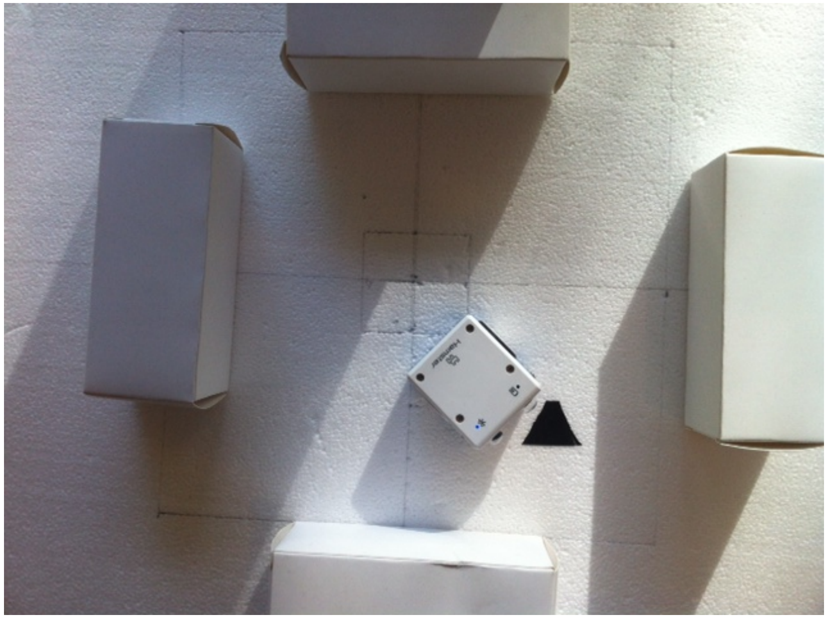


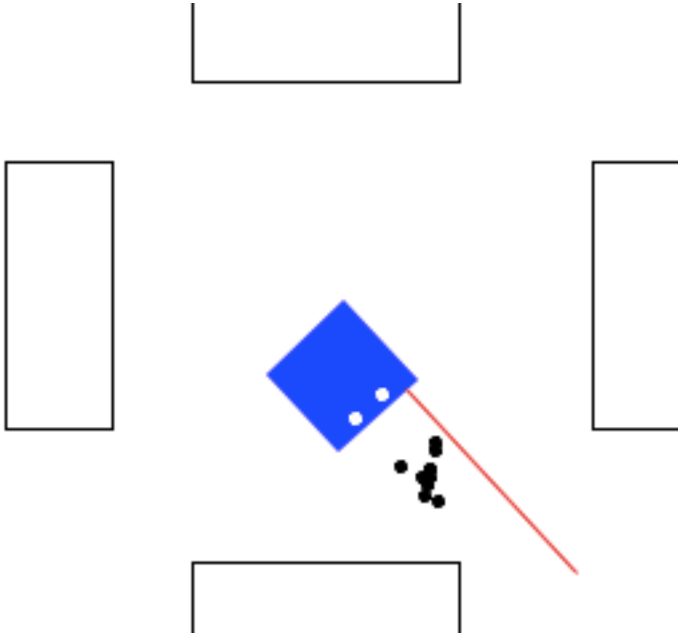
Left and Right Floor Sensors



Landmark Navigation Using Floor Sensors

- Greyscale
- Patterns





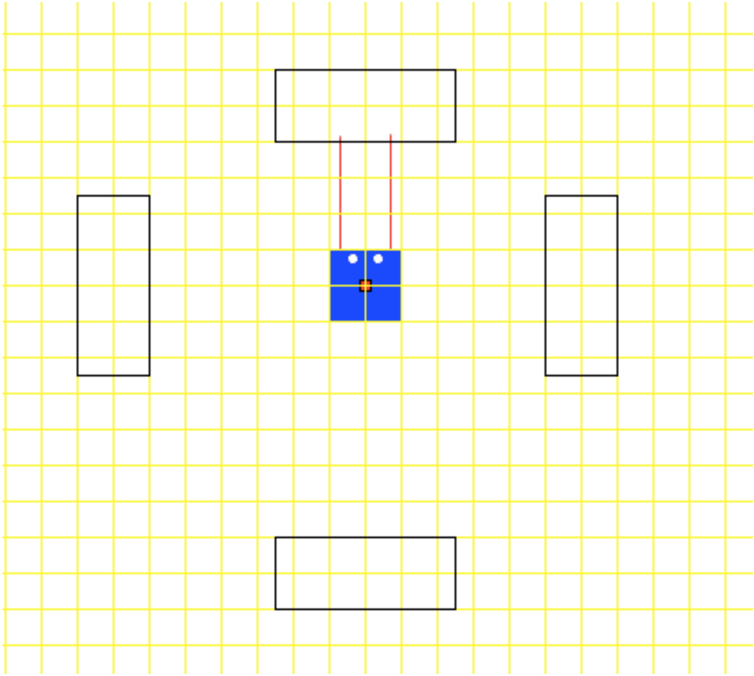
Combining Relative and Absolute Localization

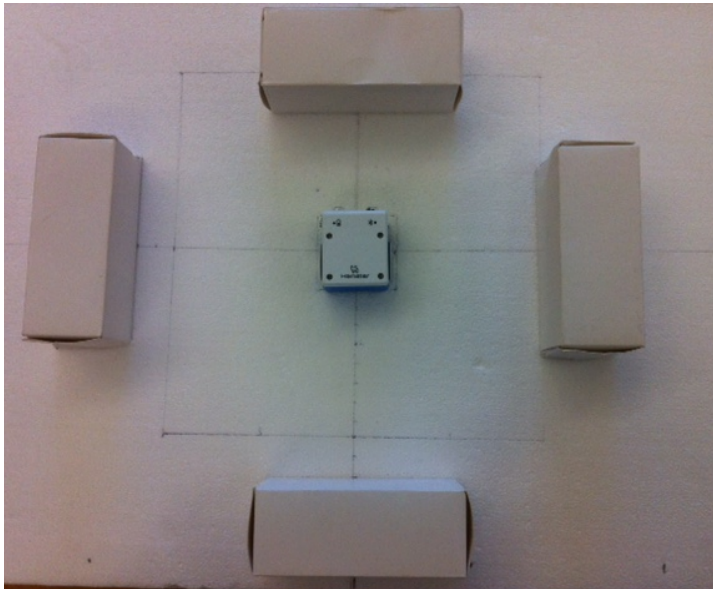
Dead reckoning +

Geometric feature based localization

Home Work #3-1: "Local" Localization and Navigation

- The map is given below which corresponds to the physical world
- 4 white boxes





Home Work Part #3-1

1. Model the robot accurately enough such that when you joystick the robot, the position of the robot on the map and the robot in the physical world should not be different by more than 20mm (for x and y, after driving 100 mm) and 20 degree for orientation (after rotating by 90 degree)
    1. Notice that dead reckoning error is cumulative
2. Localize robot relative to the 4 edges within the following error bound:
    1. x, y: less than 10mm
    2. Orientation: less than 10 degree

Home Work Part #3-1

- Starting position of the Robot: your robot will be placed facing the “Top” obstacle – but the location and orientation will be off (see figure A below)
  - X, y : off no more than 40 mm (and can see the obstacle)
  - Orientation: off no more than 45 degree
- You make a call to your localization function such that the location of the robot on the map is “corrected” as in Figure B

(notice the robot has moved in Fig. B, because it rotateed/scanned to collect data)

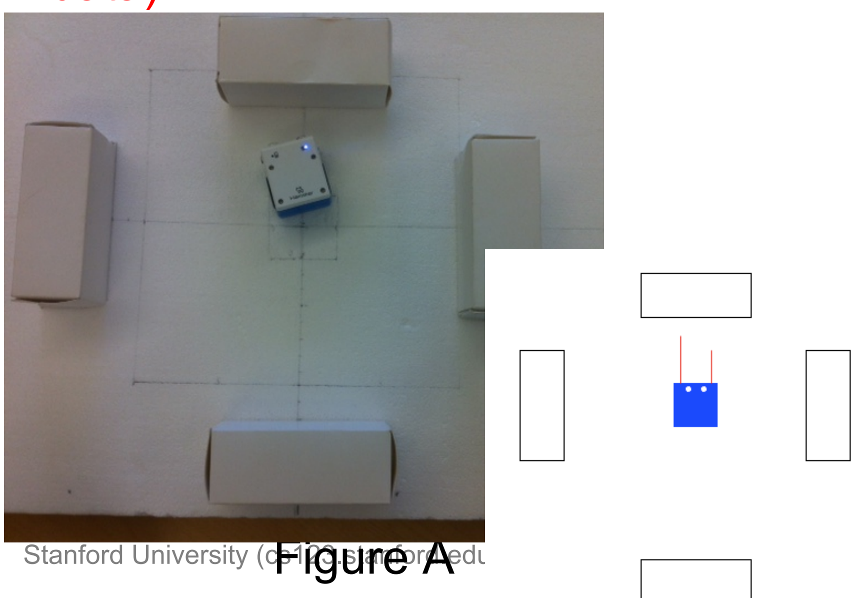


Figure A

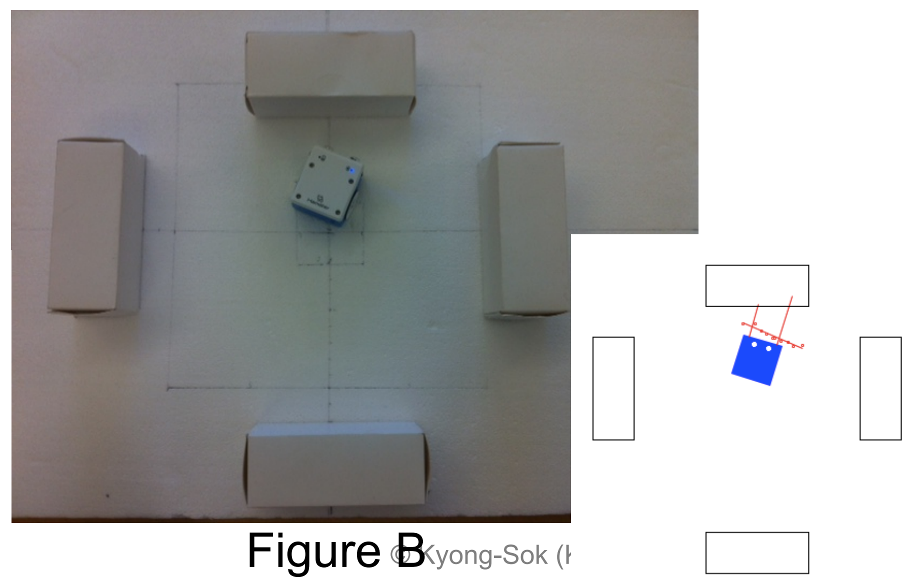
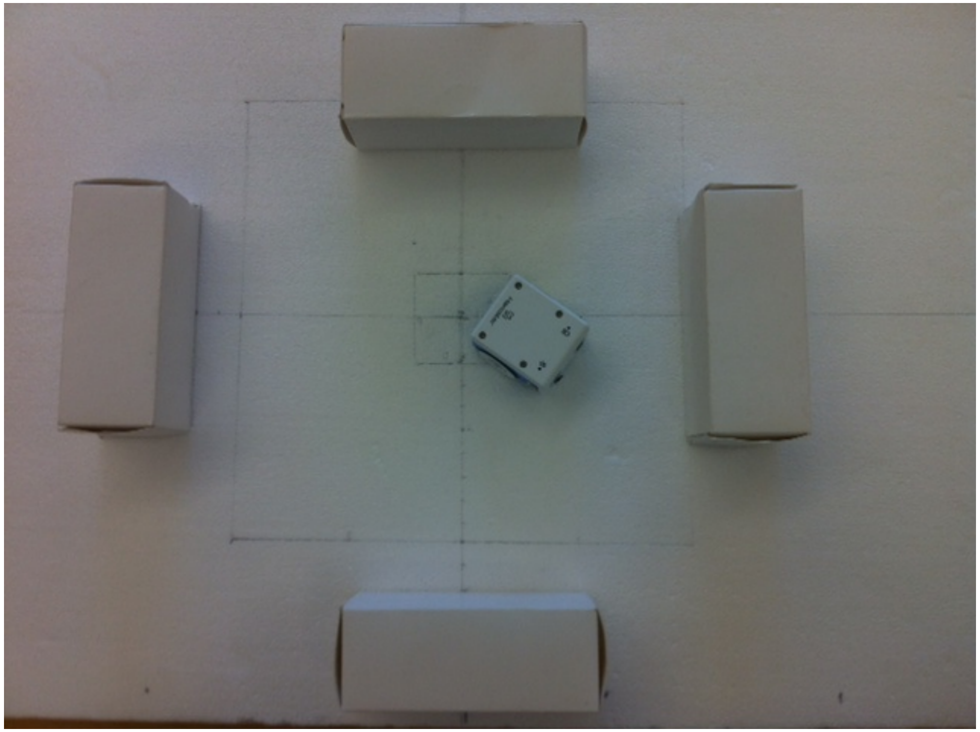
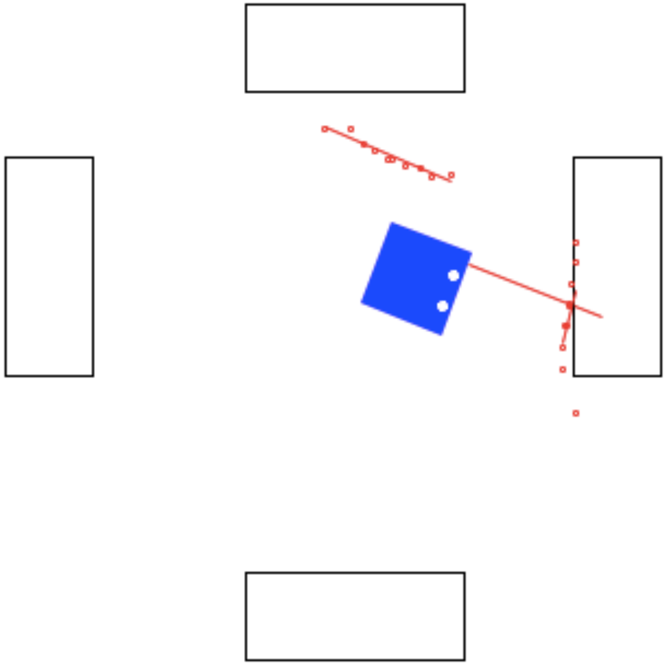


Figure B

Home Work Part #3-1

- Joystick your robot to face a different obstacle, and localize with respect to each





Home Work Part #3-1

- Joystick your robot to face the obstacle on the different obstacles, and localize with respect to each