EE-381 Robotics-1
UG ELECTIVE

Lecture 14

Dr. Hafsa Iqbal

Department of Electrical Engineering,

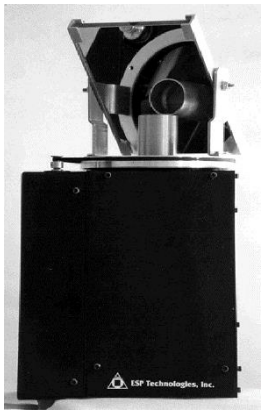
School of Electrical Engineering and Computer Science,

National University of Sciences and Technology,

Pakistan

Laser Range Sensor (time of flight, electromagnetic)

- Is called Laser range finder or LiDAR (LIght Detection And Ranging)

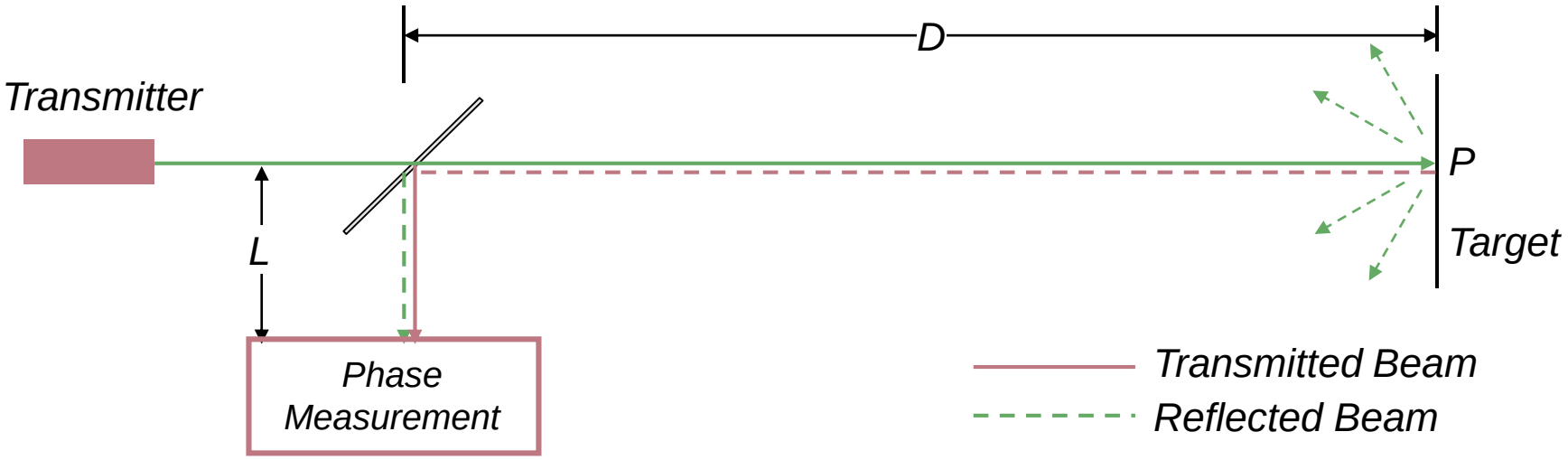


Alaska-IBEO

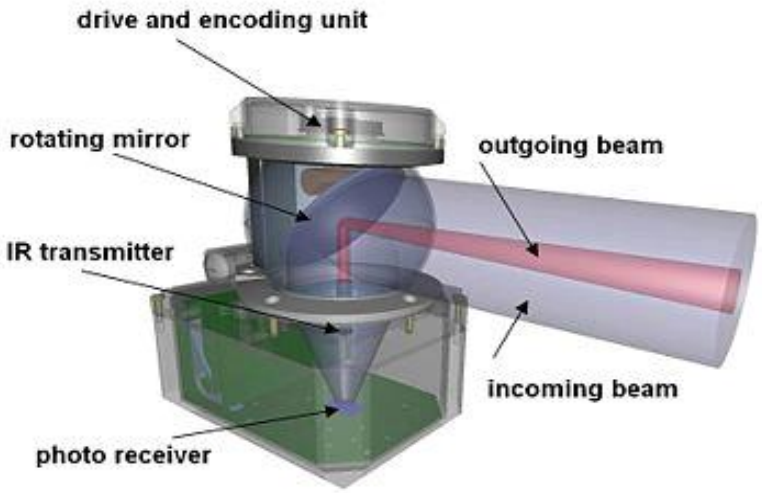
SICK

Hokuyo

Laser Range Sensor (time of flight, electromagnetic)



- Transmitted and received beams coaxial
- Transmitter illuminates a target with a collimate laser beam
- Receiver detects the time needed for round-trip
- A mechanical mechanism with a mirror sweeps
  - 2D or 3D measurement



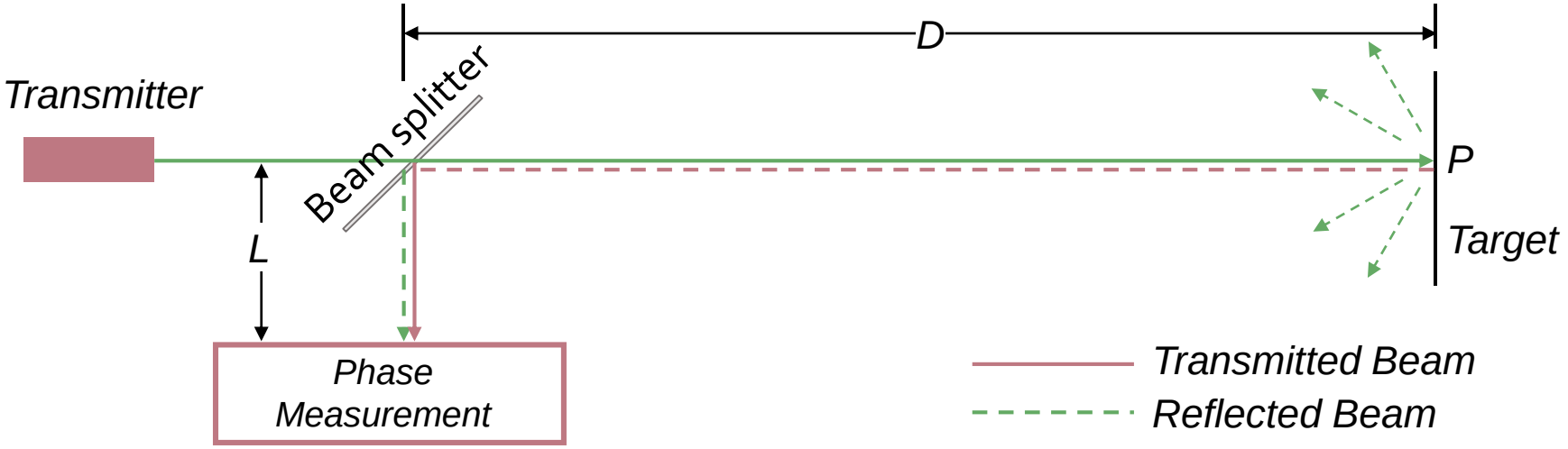
Laser Range Sensor (time of flight, electromagnetic)

Time of flight measurement

- Pulsed laser (today the standard)
  - measurement of elapsed time directly
  - resolving picoseconds
- Beat frequency
  - FMCW (frequency modulated continuous wave)
- Phase shift measurement to produce range estimation
  - technically easier than the above method

Laser Range Sensor (time of flight, electromagnetic)

- *Phase-Shift Measurement*



$$D' = L + 2D = L + \frac{\theta}{2\pi}\lambda$$

$$\lambda = \frac{c}{f}$$

Where:

c: is the speed of light; *f* the modulating frequency; *D'* the distance covered by the emitted light is.

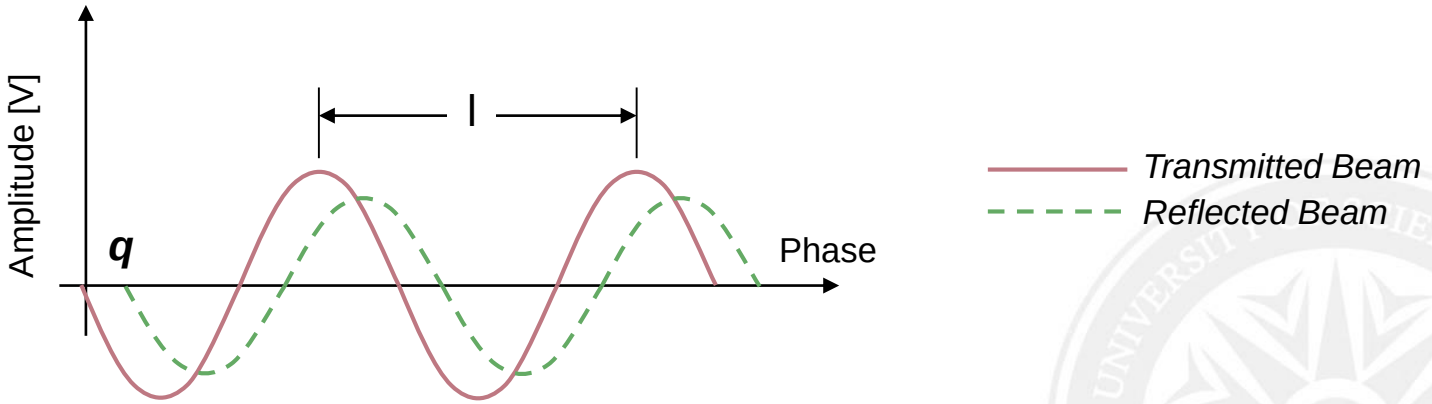
- for f = 5 MHz (as in the A.T&T. sensor), λ = 60 meters

Laser Range Sensor (time of flight, electromagnetic)

- Distance D, between the beam splitter and the target

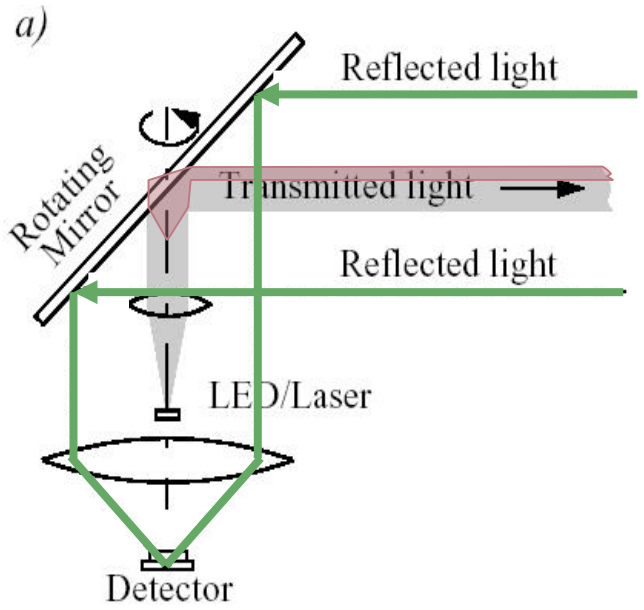
$$D = \frac{\lambda}{4\pi}\theta$$

- where
  - θ: phase difference between transmitted and reflected beam



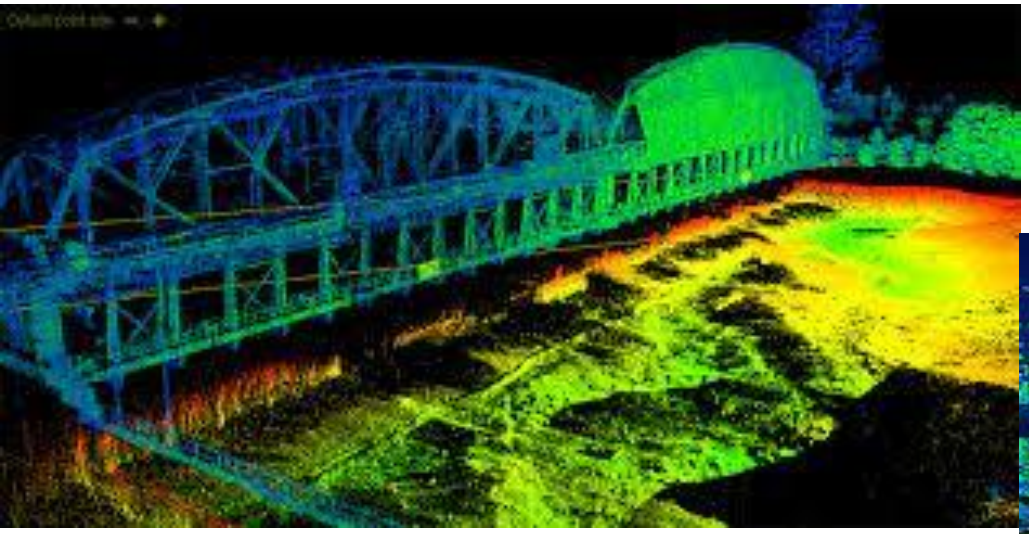
Laser Range Sensor (time of flight, electromagnetic)

- Uncertainty of the range (phase/time estimate) is inversely proportional to the square of the received signal amplitude.
  - Hence dark, distant objects will not produce such good range estimates as closer brighter objects …



The SICK LMS 200 Laser Scanner

- Angular resolution 0.25 deg
- Depth resolution ranges between 10 and 15 mm and the typical accuracy is 35 mm, over a range from 5 cm up to 20 m or more (up to 80 m), depending on the reflectivity of the object being ranged.



Laser Range Finder: Applications

Stanley: Stanford
(winner of the 2005 Darpa Grand Challenge)

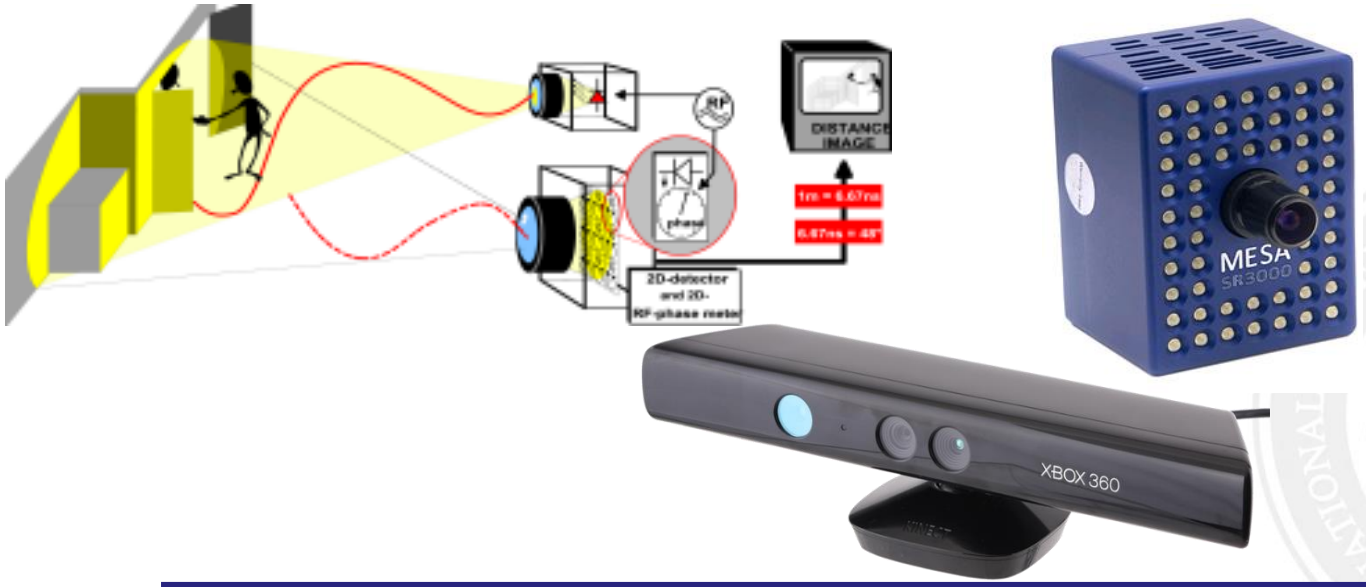
Autonomous Smart:
ASL ETH Zurich

3D Laser Range Finder

- A 3D laser range finder is a laser scanner that acquires scan data in more than a single plane.
- Custom-made 3D scanners are typically built by nodding or rotating a 2D scanner in a stepwise or continuous manner around an axis parallel to the scanning plane.
- By lowering the rotational speed of the turn-table, the angular resolution in the horizontal direction can be made as small as desired.
- A full spherical field of view can be covered (360° in azimuth and 90° in elevation).

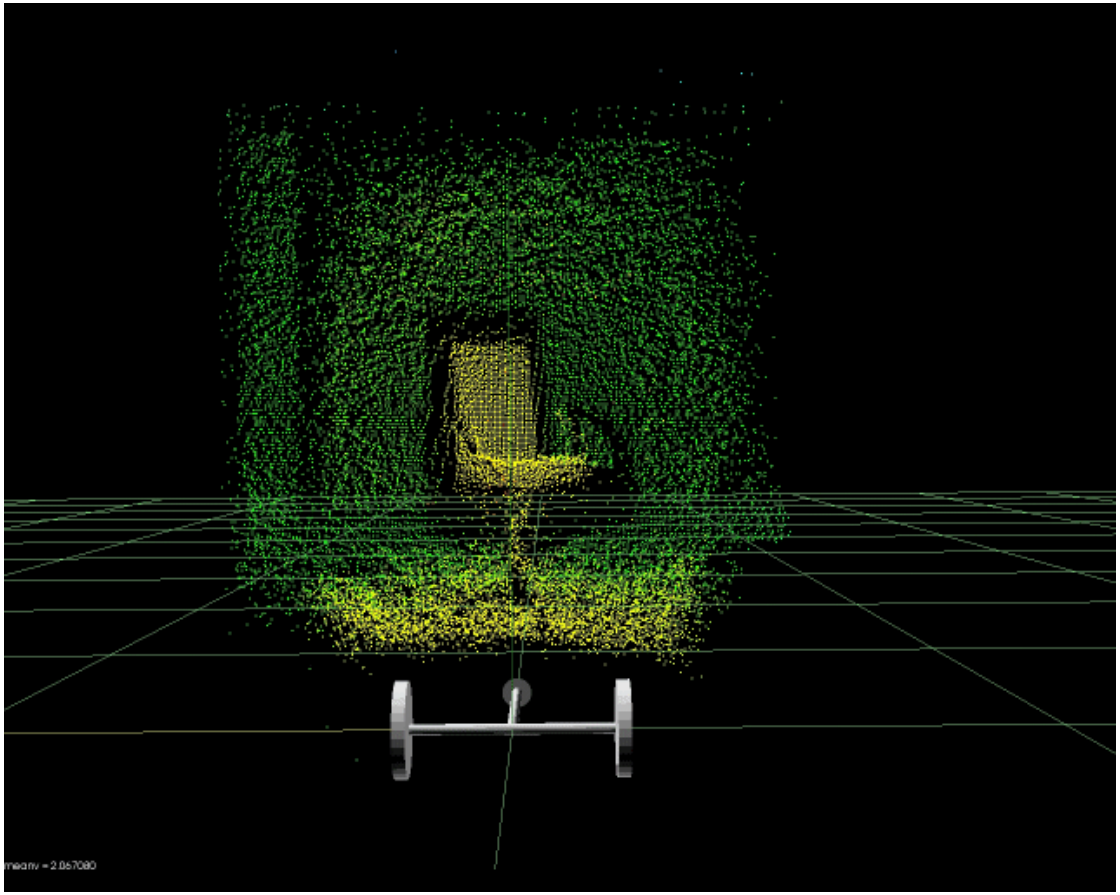
3D Range Sensor (4): Time Of Flight (TOF) camera

- A Time-of-Flight camera (TOF camera, figure) works similarly to a LiDAR with the advantage that the whole $3D$ scene is captured at the same time and that there are no moving parts. This device uses a modulated infrared lighting source to determine the distance for each pixel of a Photonic Mixer Device (PMD) sensor.



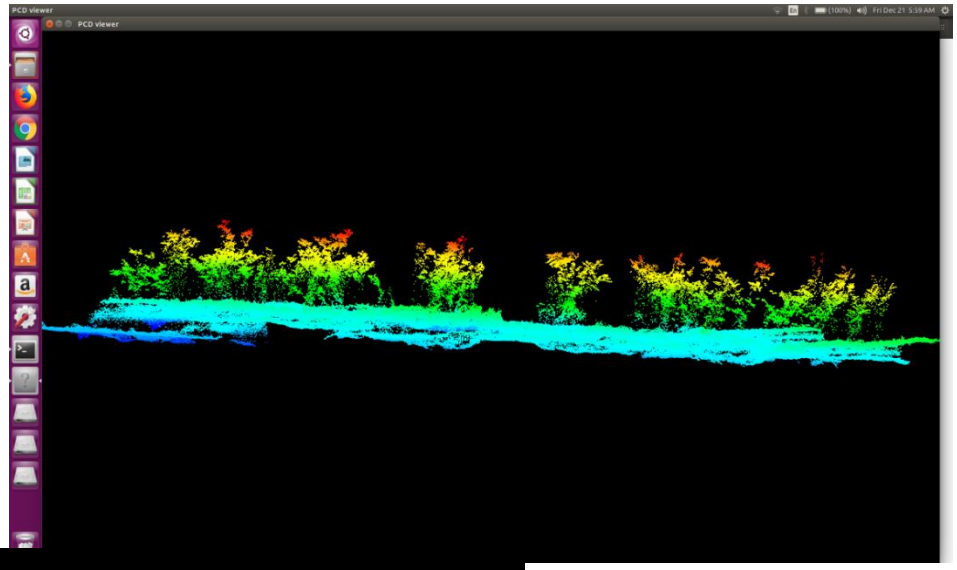
Swiss Ranger 3000 (produced by MESA)

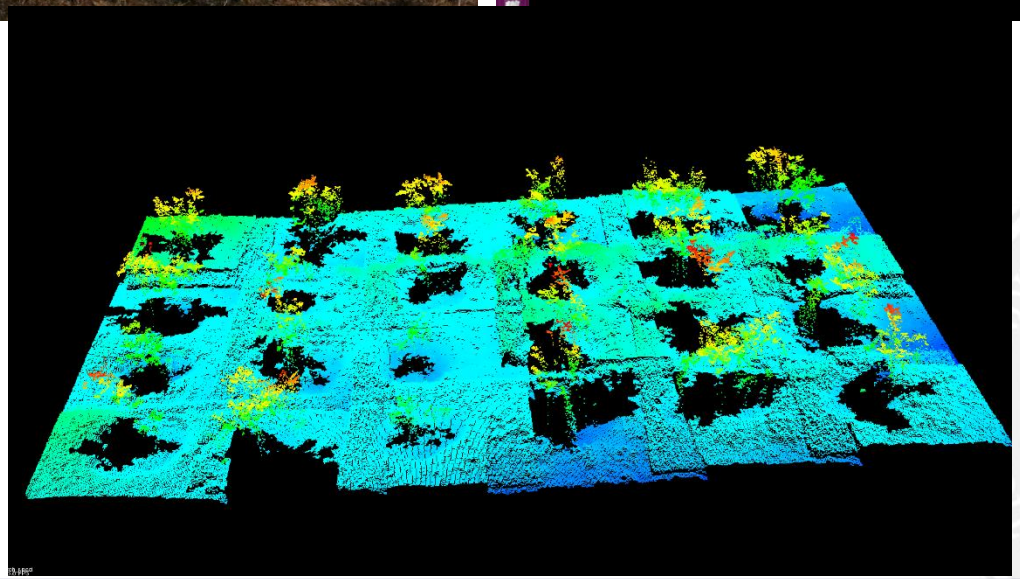
3D Range Sensor (4): Time Of Flight (TOF) camera



ZCAM (from 3DV Systems now bought by Microsoft for Project Natal)

Time Of Flight (TOF) camera: Kinect

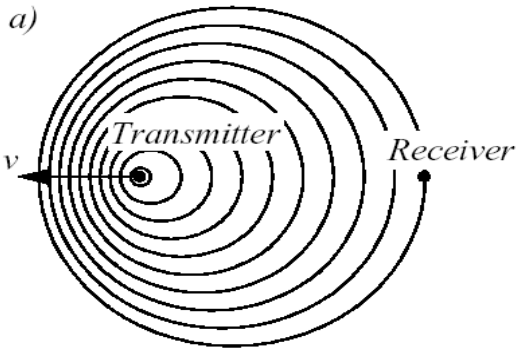


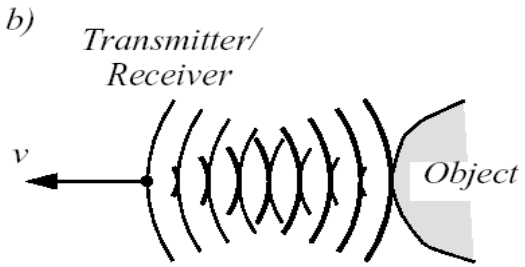


Doppler Effect Based (Radar or Sound): Speed Sensors

$f_t \rightarrow$ frequency of transmitted wave

$f_r \rightarrow$ frequency of received wave





- a) Between two moving objects
- $f_r = f_t\,(1 + v/c)$ if transmitter is moving

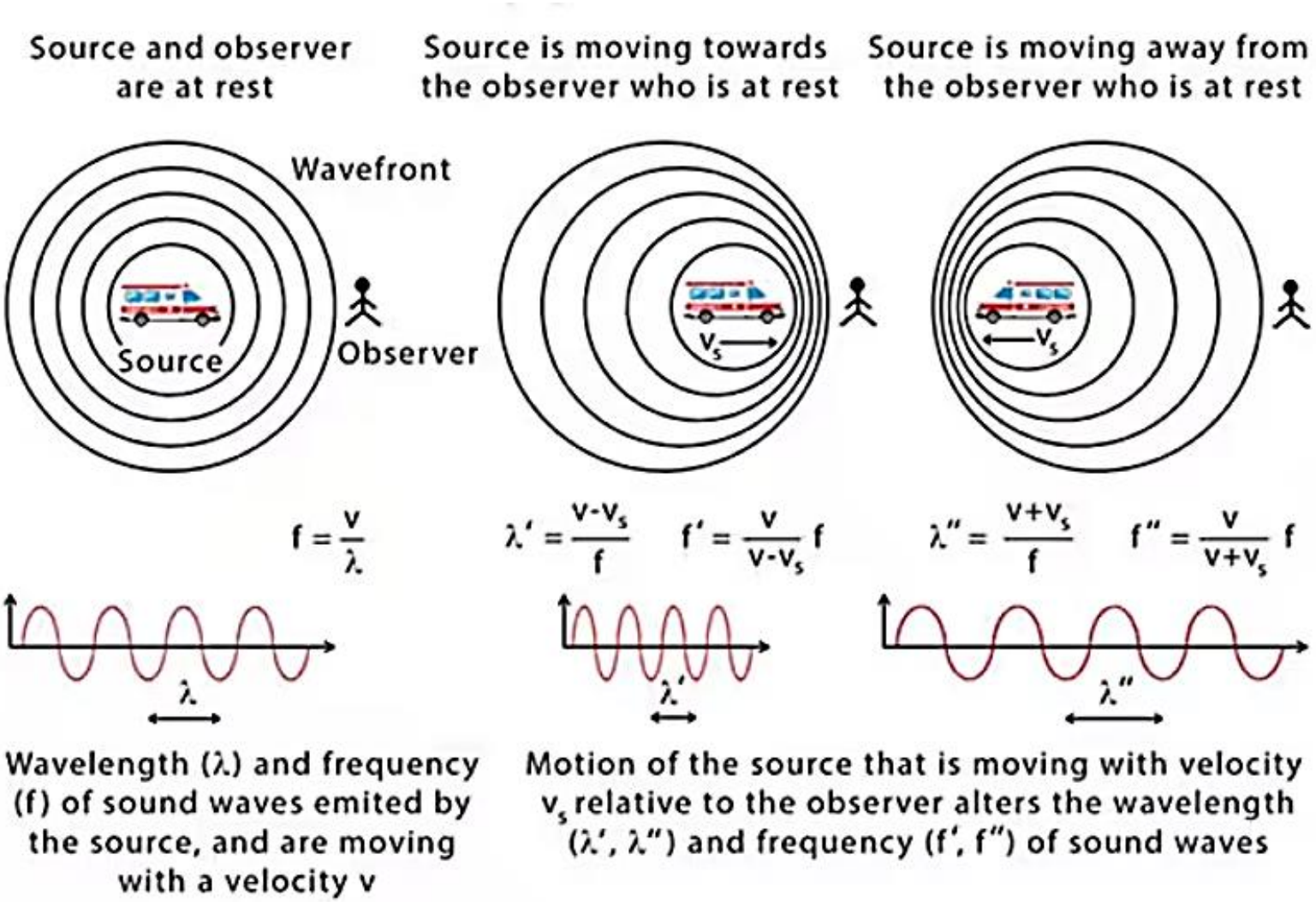
- b) Between a moving and a stationary object
- $f_r = f_t \dfrac{1}{1 + v/c}$ if receiver is moving

- $\Delta f = f_t - f_r = \dfrac{2 f_t v \cos\theta}{c}$

for Doppler shift frequency

- $v = \dfrac{\Delta f \cdot c}{2 f_t \cos\theta}$ relative speed

Doppler Effect Based (Radar or Sound): Speed Sensors

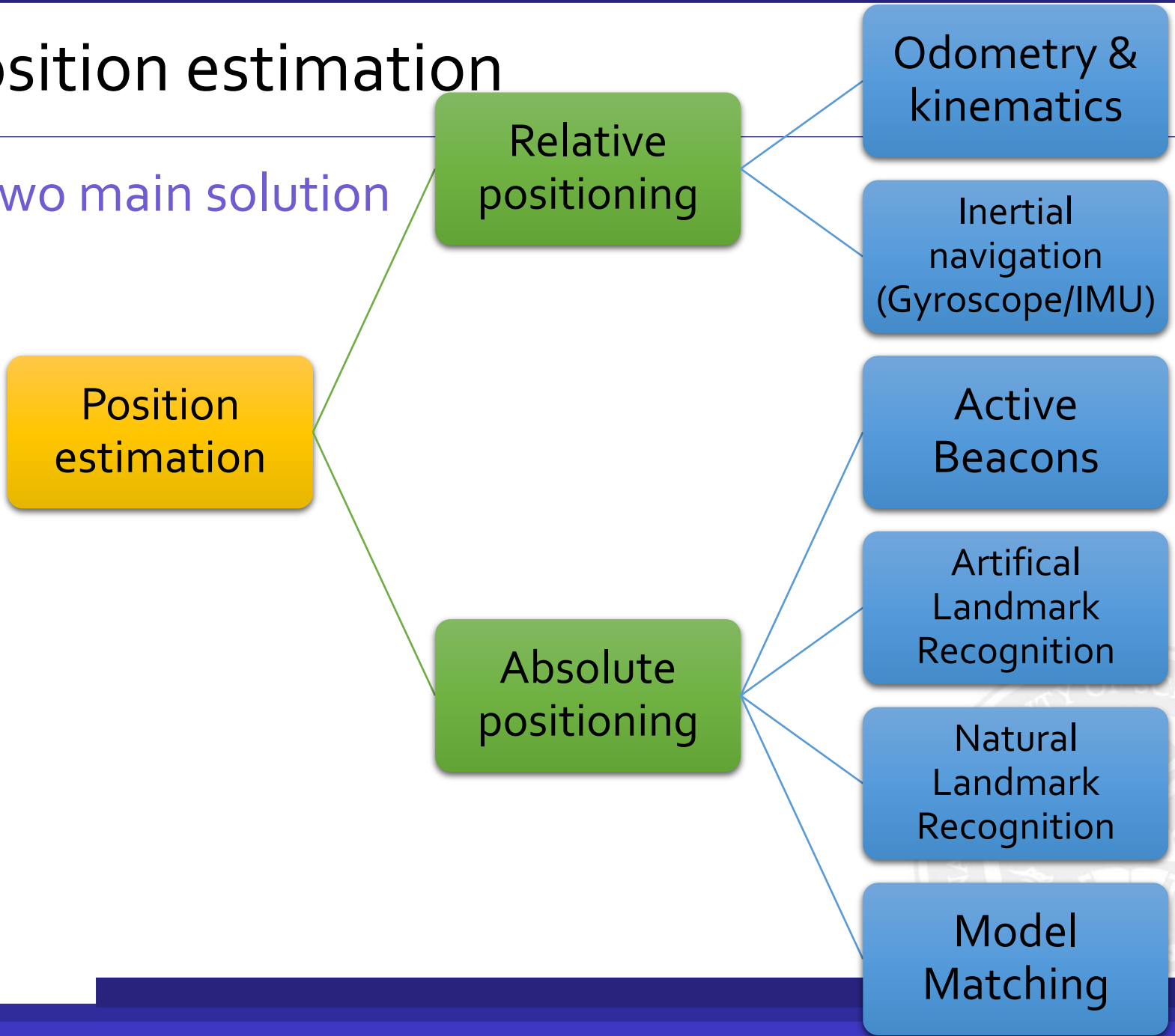


Doppler Effect Based (Radar or Sound): Speed Sensors

- θ = relative angle between direction of motion and beam axis.
- Sound waves: e.g. industrial process control, security, fish finding, measure of ground speed
- Electromagnetic waves: e.g. vibration measurement, radar systems, object tracking

Position estimation

- Two main solution



Comparison

	Advantages	Disadvantages
Wheel encoder (odometry)	• Self-contained • Can provide positions anywhere along curved paths • Always give "estimate" of position	• Require accurate measurement of wheel velocities/RPM over time, including measuring acceleration and deceleration • Position error grows over time
Gyroscope	• Self-contained • Always give "estimate" of position	• Position error grows over time • Expensive sensors
Active Beacons (light/radio signals)	• Absolute position obtainable • No accumulative errors	• Not self-contained • Fails if becomes inactive

Comparison

	Advantages	Disadvantages
Artificial Landmark Recognition	• Landmarks can be designed for optimal detectability • Bounded position errors	• Requires alteration of environment • Not always possible to detect landmarks • Landmarks maybe shapes (as opposed to points), requiring measurement of geometric features
Natural landmark recognition	• Does not require alteration of environment	• Environment must be known in advance • More difficult to detect than artificial landmarks
Model matching	• Almost no error in position when match is found	• Sometimes impossible to determine potion in models with much symmetry

Taxonomy of Localization Problems-I

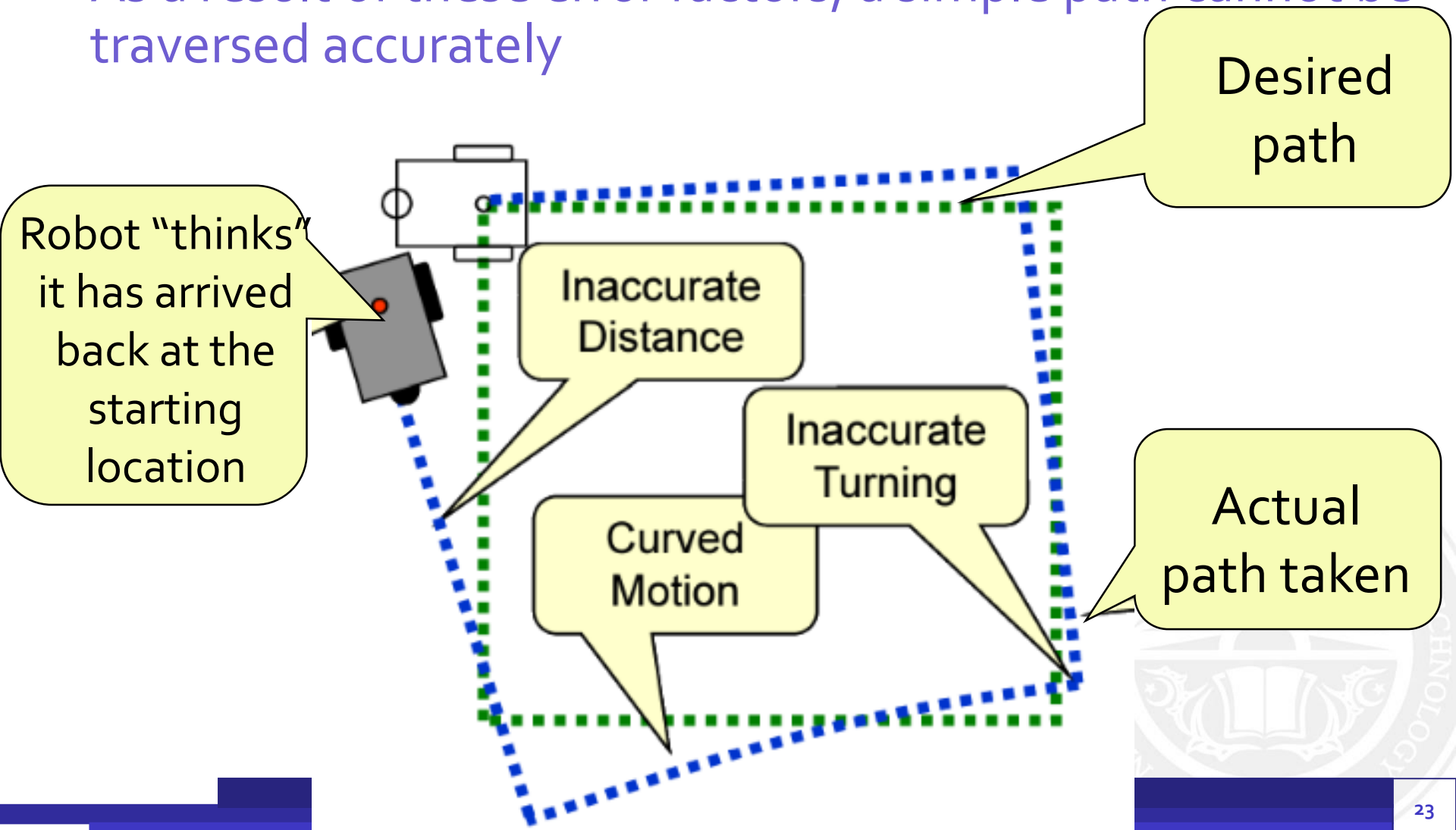
- Dead-reckoning (position tracking): initial position known
  - Blindly update pose based on differential movements
  - Errors accumulate over time as robot moves due to uncertainty in measurements. Periodically requires error measurement to be "fixed" or reset (usually from external sources) in order to be useful
  - Preferable for short distance measurements

Odometry Based Localization

- Errors can creep in due to
  - Imprecise measurement
  - Inaccurate control model
  - Immeasurable physical characteristics

Taxonomy of Localization Problems-I

- As a result of these error factors, a simple path cannot be traversed accurately



Odometry & kinematics

- The robot may become "lost" quite quickly.
- Theoretically, we can calculate the actual robot's position as long as:
  - the robot's structure is well known
  - the robot's wheel acceleration, deceleration and velocities can be accurately measured
- Hence, we can detect our errors during travel and adjust the path accordingly
  - This is done through kinematics

Taxonomy of Localization Problems-II

- Global localization: initial position can be unknown
  - Map-based (with landmarks)
    - Sensors: laser, camera
    - Method: map-matching techniques (various)
  - Beacon-based (with active infrastructure)
    - Bluetooth, WiFi, GPS (outdoors only), etc.
    - Method: trilateration, RSS fingerprinting
- Global localization and position tracking combined

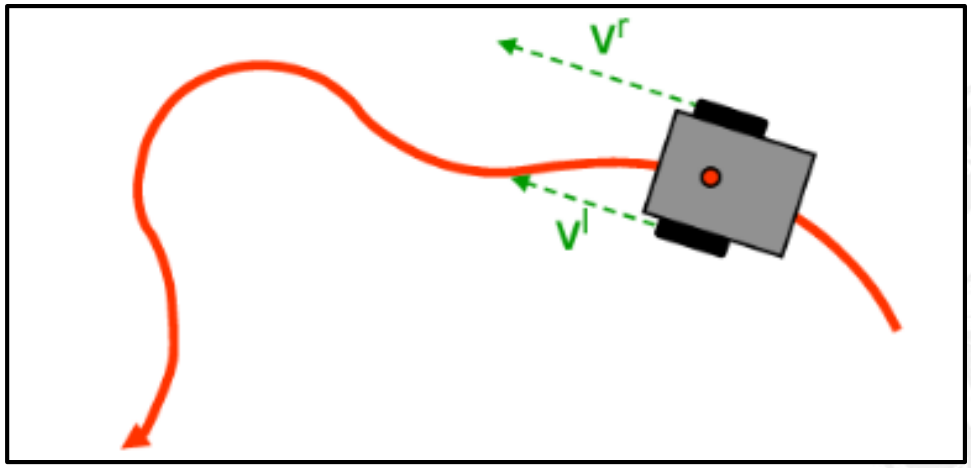
Challenges

- Dead-reckoning
  - Wheel slip, slack in actuation mechanism
  - Runway errors
- Global localization
  - Random errors and failures
  - Non-Gaussian sensor noise
  - Unavailability of sensor (e.g., GPS-denial)
  - Map ambiguity
  - Dynamic environments
  - Kidnapped robot problem

1. Differential Drive Robot & Kinematics

The key steps are

1. Obtain the RPM of both wheels
2. Calculate the linear velocity v_r, v_l or v and angular velocity ω
3. Calculate the initial pose(x_t, y_t, θ_t)
4. Update the pose $(x_{t+1}, y_{t+1}, \theta_{t+1})$ for next time instants

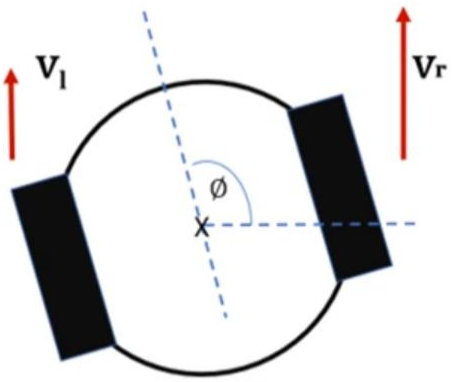


Velocity from RPM

- Velocity of each wheel can be found by

$$v_r = \frac{2\pi}{60} \times \frac{L_r}{2} \times RPM_r$$

$$v_l = \frac{2\pi}{60} \times \frac{L_l}{2} \times RPM_l$$



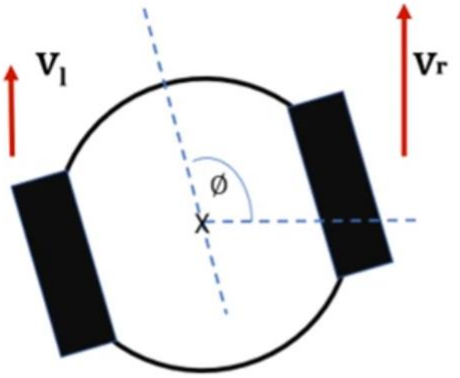
R= Radius of the wheels
L= Length between the wheels

- L is the length/diameter of a wheel
- RPM is the revolutions per minute
- Angular velocity

$$\omega = \frac{(v_r - v_l)}{L}$$

$$\rightarrow L_l = L_r = L$$

Odometry Based Localization



R= Radius of the wheels
L= Length between the wheels

- Position estimation

$$\phi = \frac{R}{L}(v_r - v_l)$$

$$x = \frac{R}{2}(v_r + v_l)\cos(\phi) = v\,cos(\phi)$$

$$y = \frac{R}{2}(v_r + v_l)\sin(\phi) = v\sin(\phi)$$

Where $v = \frac{R}{2}(v_r + v_l)$

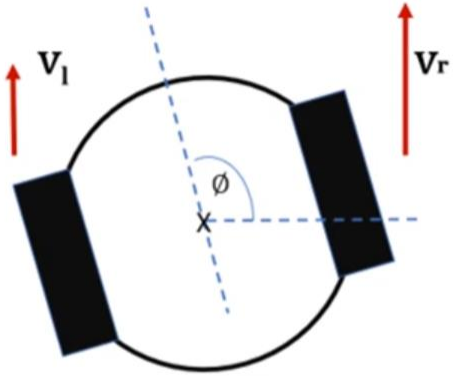
Odometry Based Localization

- Position estimation

$$x(t+\delta) = x(t) + v_t \delta \cos(\phi_t)$$

$$y(t+\delta) = y(t) + v_t \, \delta \, \sin(\phi_t)$$

$$\phi(t+\delta) = \phi(t)$$



R= Radius of the wheels
L= Length between the wheels

Where δ is the time stamp

2. Odometry

- Compute updated position based on left and right wheel readings

$$\Delta s_r = 2\,\pi\,R \times \#(revolutions_r)$$

R is radius of wheels

current pose:

$$x_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

traveled distance:

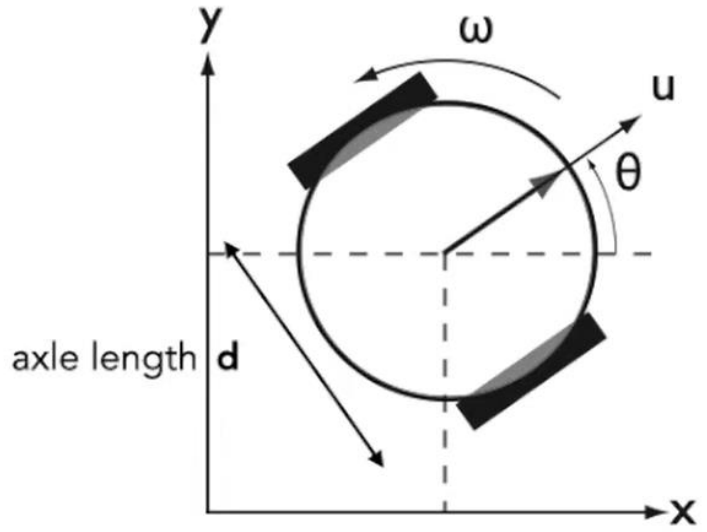
$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$

path traveled in last sampling interval:

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

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

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{d}$$



updated pose:

$$x_{t+1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix}$$

The Problem

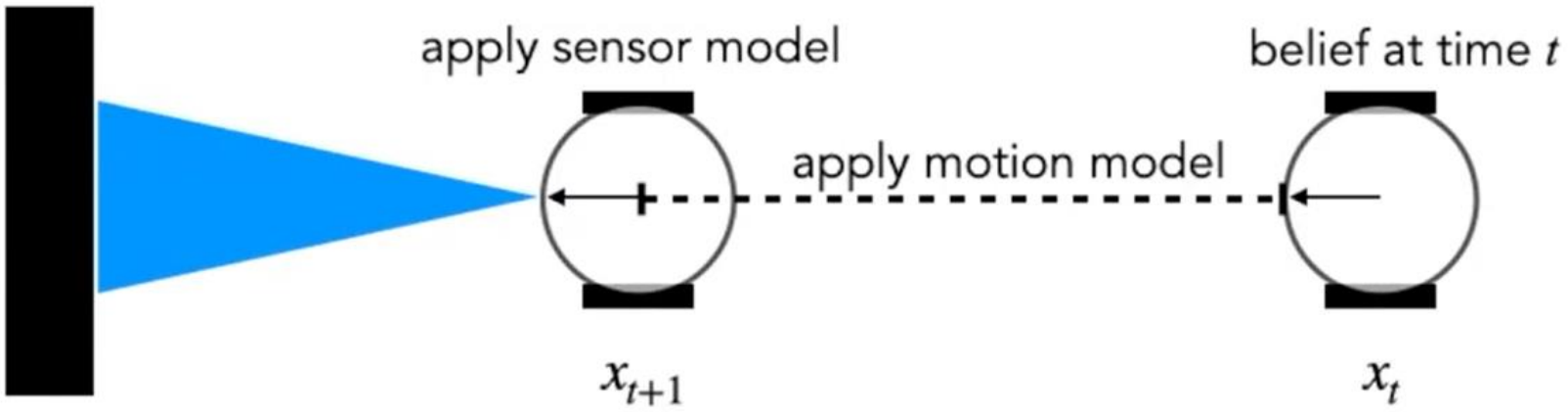
- Consider a mobile robot moving in a known environment
- As it start to move, say from a precisely known location, it might keep track of its location using odometry
- However, after a certain movement the robot will get very uncertain about its position
- Update using an observation of its environment
- Observation lead to an estimate of the robots position which can than be fused with the odometric estimation to get the best possible update of the robots actual position

Probabilistic Localization

In robotics, we deal with localization probabilistically

Three key components:

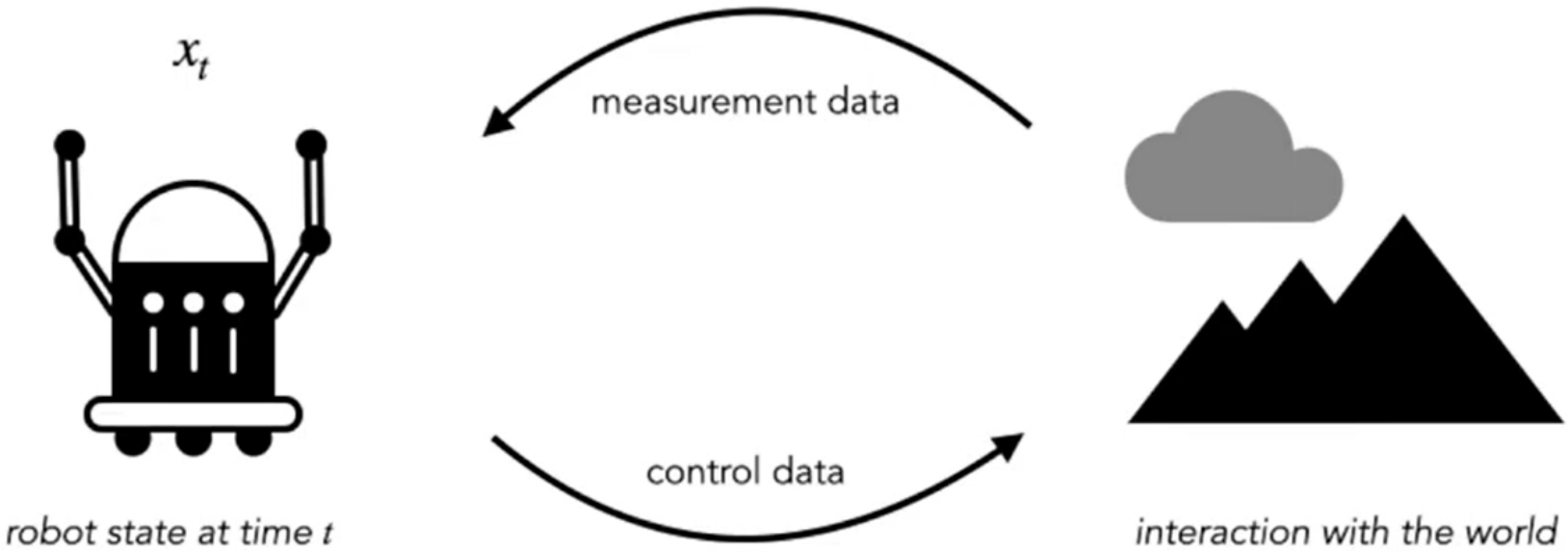
- A robots' belief of where it is (its state)
- A robots' motion model
- A robots' sensor (observation) model



Sensor and Control Data

- An autonomous robot interacting with the world

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2}$$



$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$$

We will treat odometry readings as our control data!

Why a probabilistic approach for mobile robot localization?

- Because the data coming from the robot sensors are affected by measurement errors, we can only compute the probability that the robot is in a given configuration.
- The key idea in probabilistic robotics is to represent uncertainty using probability theory: instead of giving a single best estimate of the current robot configuration, probabilistic robotics represents the robot configuration as a probability distribution over the all possible robot poses. This probability is called ***belief***.

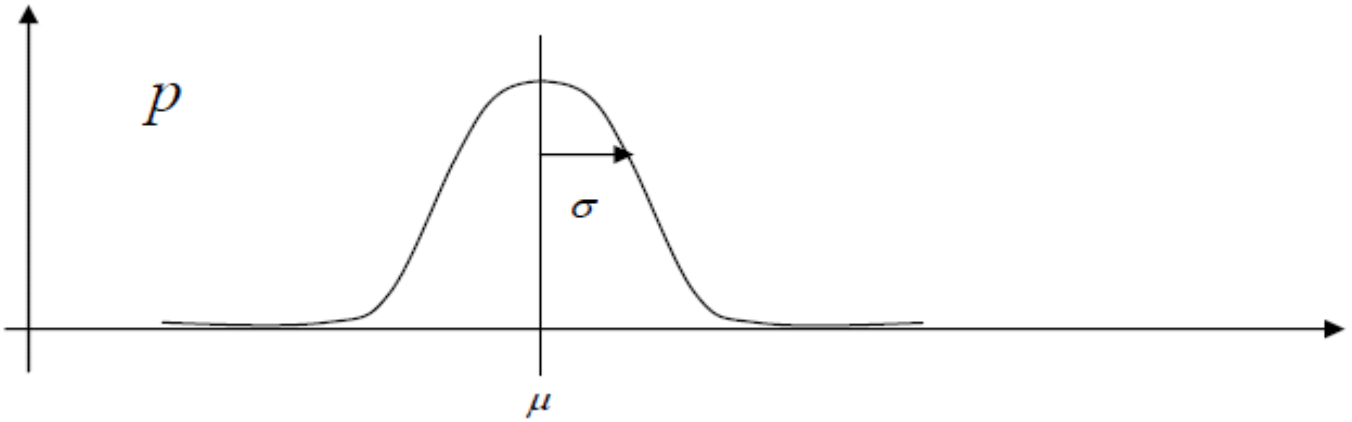
Example: Gaussian distribution

- The Gaussian distribution has the shape of a bell and is defined by the following formula:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ (1-dimensional)

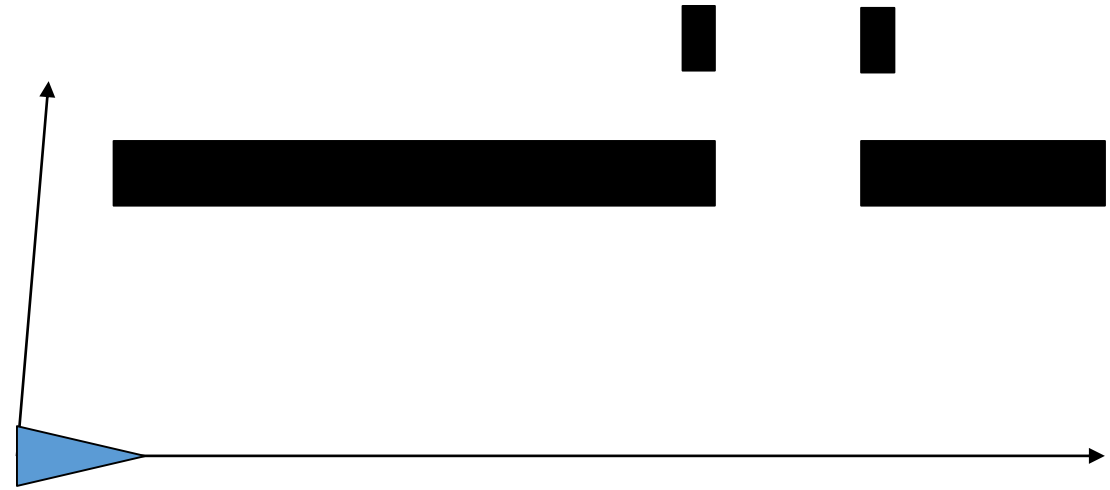
where μ is the mean value and σ is the standard deviation.

- The Gaussian distribution is also called normal distribution and is usually abbreviated with $N(\mu, \sigma)$.



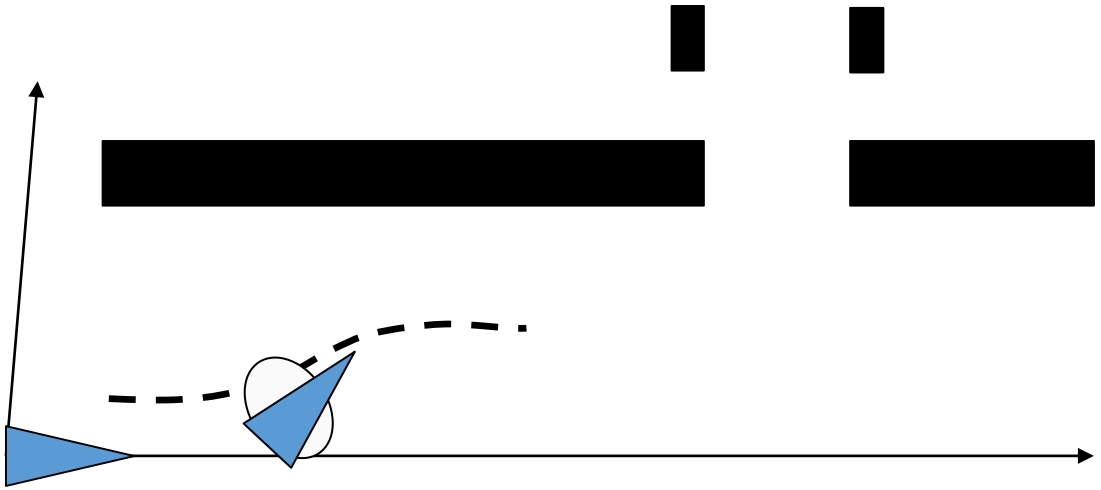
Description of the probabilistic localization problem

- Consider a mobile robot moving in a known environment.



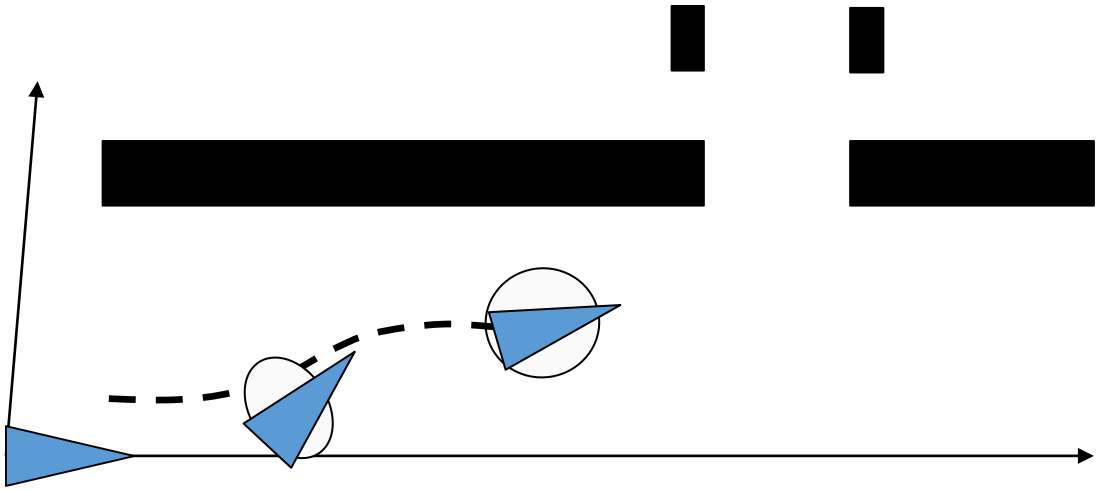
Description of the probabilistic localization problem

- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.



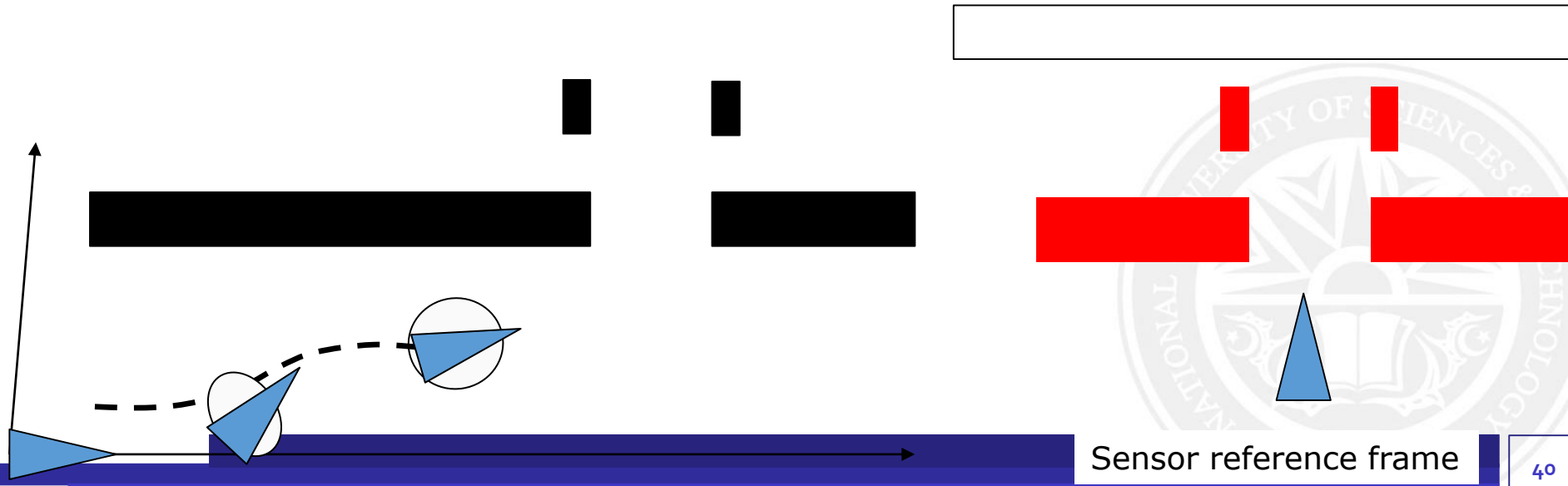
Description of the probabilistic localization problem

- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry
- Due to odometry uncertainty, after some movement the robot will become very uncertain about its position.



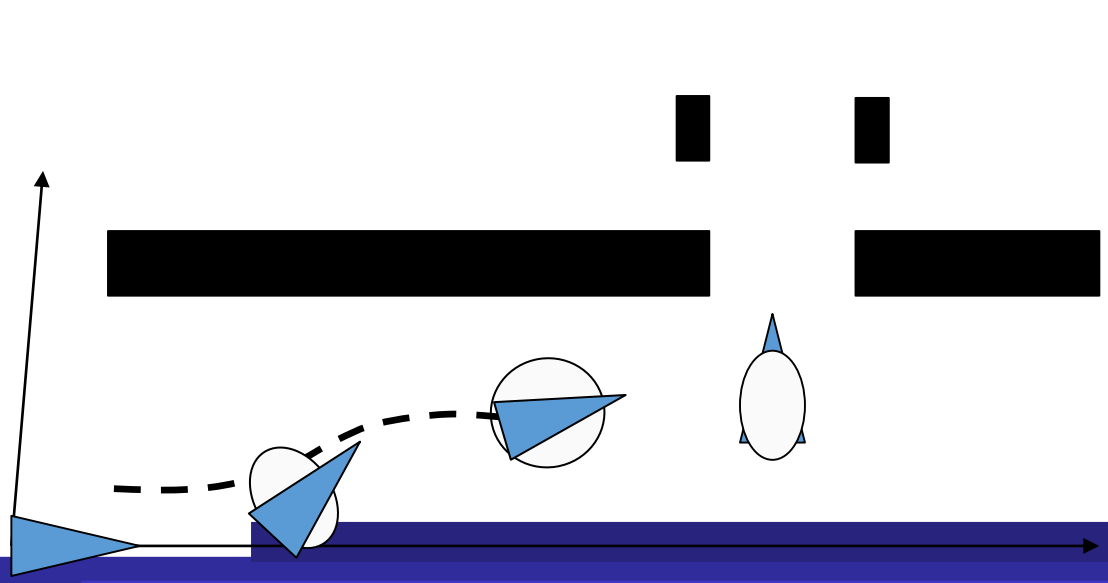
Description of the probabilistic localization problem

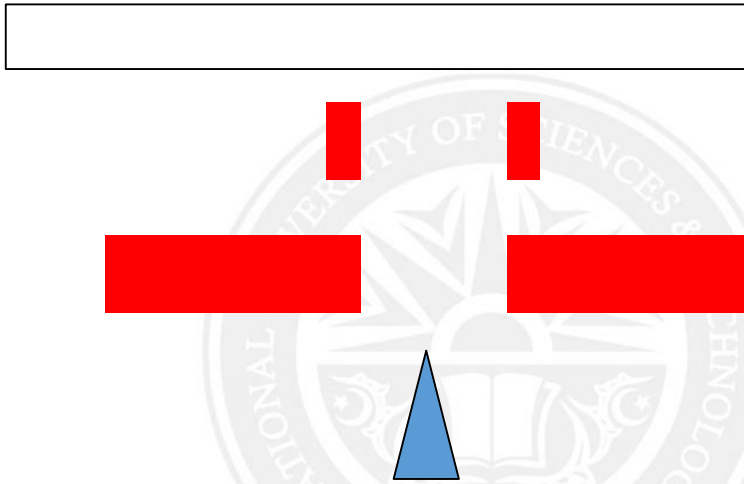
- To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot might use its on-board exteroceptive sensors (e.g. ultrasonic, laser, vision sensors) to make observations of its environment



Description of the probabilistic localization problem

- To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot might use its on-board exteroceptive sensors (e.g. ultrasonic, laser, vision sensors) to make observations of its environment.
- The robot updates its position based on the observation. Its uncertainty shrinks





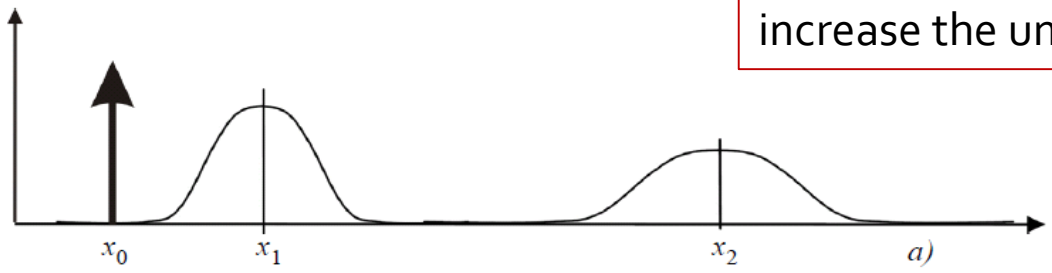
Sensor reference frame

Action and perception updates

- In robot localization, we distinguish two update steps:
  - Action (prediction) update and perception (measurement) update
- Action update (prediction model)
  - robot moves and estimates its position through its **proprioceptive** sensors. During this step, the robot uncertainty grows.

  With O_t: encoder measurement, S_{t-1}: prior belief state
  - Increase uncertainty

perception update will reduce the uncertainty, while action update will increase the uncertainty.

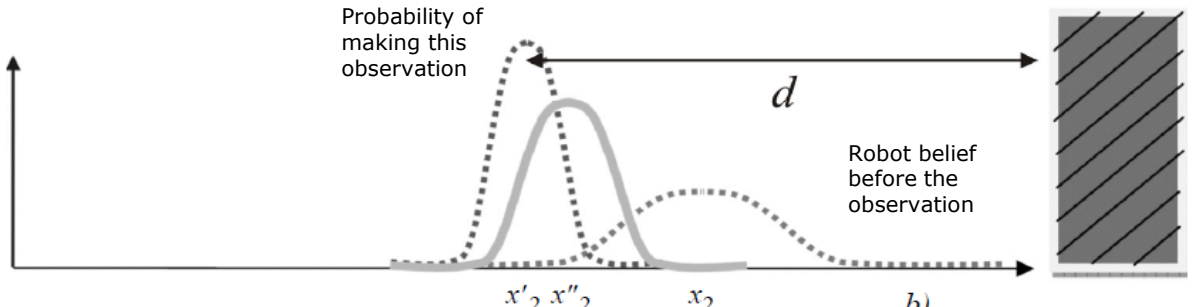


Action and perception updates

- In robot localization, we distinguish two update steps:
  - Action (prediction) update and perception (measurement) update
- Perception update (observation model "SEE")
  - the robot makes an observation using its **exteroceptive** sensors and correct its position by opportunely combining its belief before the observation with the probability of making exactly that observation. During this step, the robot uncertainty shrinks.

With i_t: exteroceptive sensor inputs, s_t' : updated belief state

- Decreased uncertainty



Solution to the probabilistic localization problem

Solving the probabilistic localization problem

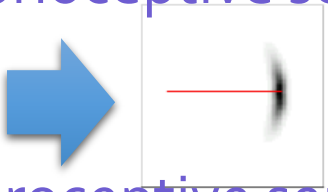
consists in solving separately the Action and Perception updates

Solution to the probabilistic localization problem

A probabilistic approach to the mobile robot localization problem is a method able to compute the probability distribution of the robot configuration during each Action and Perception step.

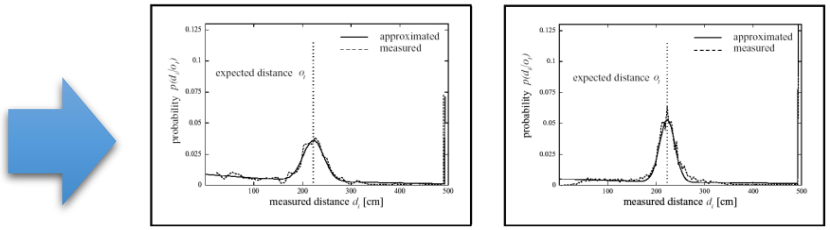
The ingredients are:

1. The initial probability distribution $p(x)_{t=0}$
2. The statistical error model of the proprioceptive sensors (e.g. wheel encoders) U_t



$$\Sigma_{\Delta} = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

3. The statistical error model of the exteroceptive sensors (e.g. laser, sonar, camera) Z_t
4. Map of the environment



Ultrasound. Laser range-finder.

(If the map is not known a priori then the robots needs to build a map of the environment and then localizing in it. This is called SLAM, Simultaneous Localization And Mapping)

Solution to the probabilistic localization problem

How do we solve the Action and Perception updates?

- Action update uses the Theorem of Total probability (or convolution)

$$p(B) = \sum_i p(B|A_i)p(A_i)$$

Discrete space

$$p(B) = \int_i p(A_i)p(B|A_i)di$$

Continuous space

- Perception update uses the Bayes rule

$$p(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

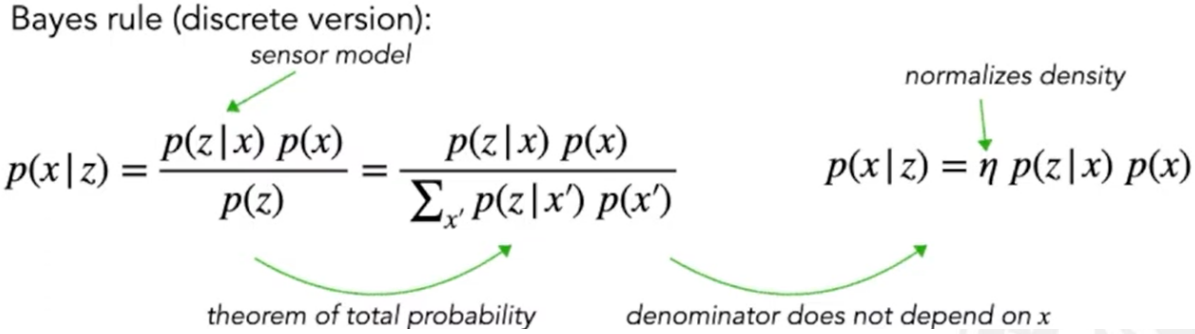
Prior

Marginal probability
(from total probability theorem)

Bayes Rule in Robotics

- Lets assume x is the robot state, and z is the measurement data
  - **Prior** probability distribution: $p(x)$
  - **Posterior** probability distribution: $p(x|z)$
- Estimate robot state using a "generative model" $p(z|x)$, which describes how a state variable **causes** sensor measurement z

Bayes rule (discrete version):



- We can now update a robot's state estimation based on sensor measurements and a prior belief

Terminology

- Robot Position $x_t = [x, y, \theta]^T$ $X_T = \{x_0, x_1, x_2, \ldots, x_T\}$,
- Proprioceptive Sensor Reading $U_T = \{u_0, u_1, u_2, \ldots, u_T\}$
- Exteroceptive Sensor Measurement $Z_T = \{z_0, z_1, z_2, \ldots, z_T\}$,
- Map

A map is vector of size 2n where n is the no. of features in the world $M = \{m_0, m_1, m_2, \ldots, m_{n-1}\}$,

m_i are the vectors representing 2D coordinates of the landmarks in the world reference frame

Terminology

Belief state: the best guess about robot state

- Belief (before observation Z_t, i.e., prediction)

$bel(X_t) = p(X_t|Z_{0:t-1}, U_{0:t-1})$ →prediction update

(belief state calculated before the new observations at "t" just after the control input U_t)

- Belief (after measurement update)

$$bel(X_t) = p(X_t|Z_{0:t}, U_{0:t})$$

(probability of robot at X_t given all its past observations $Z_{0:t}$ and all its past control inputs $U_{0:t}$)

Ingredients of probabilistic localization

1. Initial Probability Distribution

$$bel(x_0).$$

2. Map of the environment

$$M = \{m_0, m_1, m_2, \ldots, m_n\}$$

3. Proprioceptive Sensor Data

$$u_t = [\Delta S_r, \Delta S_l]^T$$

4. The probabilistic motion model

$$x_t = f(x_{t-1}, u_t).$$

5. The probabilistic measurement model

$$z_t = h(x_t, M)$$

Markov Localization

- Markov Localization tracks the robot belief state using an arbitrary probability density function to represent robot position.
  - Prediction update is based on the theorem of total probability
  - Measurement update is based on Bayes Rule

The Markov Assumptions

- The output 'x_t' is function only of robot previous state x_{t-1}, and its most recent actions (odometry) u_t, and perception z_t
- The assumption is not true all the times
  - For example, if the robot has suffered a collision resulting in a biased sensor, the robot position becomes function of all events in the history and not just previous position
  - Similarly, due to motion history, if one of the wheel is worn out and is biased in one direction, the assumption is not true.
  - Additionally, unmodeled dynamics of the environment (such as moving people) also affect the assumption.

Markov Localization: Action update

- In this phase, the robot estimates its current position $\overline{bel}(x_t)$ based on the u_t knowledge of the previous position $bel(x_{t-1})$ and the odometric input.

- Using the Theorem of Total probability, we compute the robot's belief after the motion as

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

$$p(B) = \int_i p(B|A_i) p(A_i) d$$

which can be written as a *convolution*

$$\overline{bel}(x_t) = p(x_t | u_t, x_{t-1}) * bel(x_{t-1})$$

Markov Localization: Perception update

- In this phase the robot corrects its previous position (i.e. its former belief) by opportunely combining it with the information from its exteroceptive sensors

$$p(x_t \mid z_t) = \frac{p(z_t \mid x_t)\, p(x_t)}{p(z_t)}$$

$$bel(x_t) = \eta \cdot p(z_t \mid x_t)\overline{bel}(x_t)$$

where η is a normalization factor which makes the probability integrate to 1.

$$p(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Markov Localization: Algorithm

- Pseudo Algorithm

```
for all x_t do
```

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \quad \text{(prediction update)}$$

$$bel(x_t) = \eta p(z_t | x_t, M) \overline{bel}(x_t) \quad \text{(measurement update)}$$

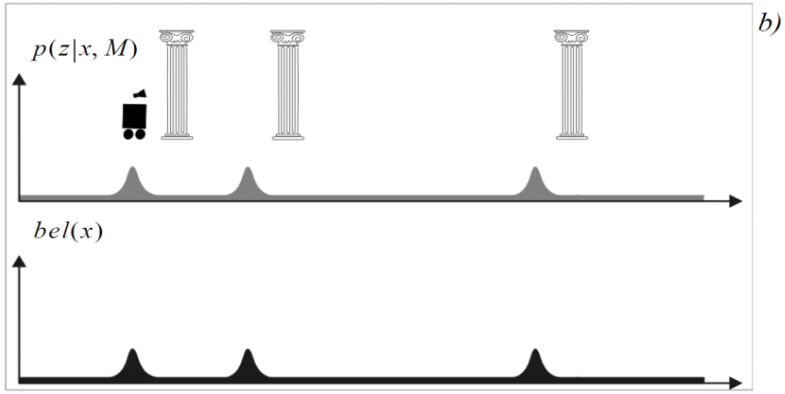
```
endfor
return bel(x_t)
```

Illustration of Markov localization algorithm

Initial probability distribution

$$p(x)_{t=0}$$

$$p(z_t \mid x_t)$$

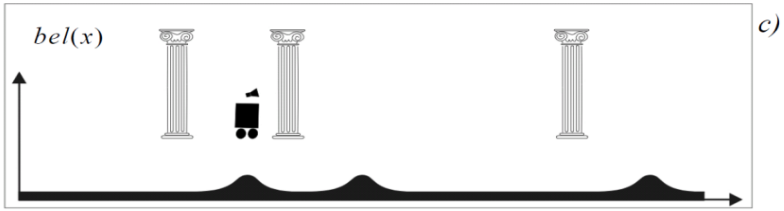


Perception update

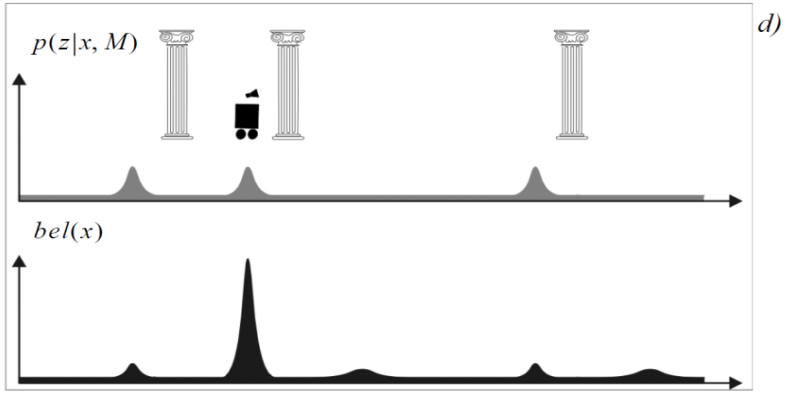
$$bel(x_t) = \eta \cdot p(z_t \mid x_t)\overline{bel}(x_t)$$

Action update

$$\overline{bel}(x_t) = p(x_t \mid u_t, x_{t-1}) * bel(x_{t-1})$$



$$p(z_t \mid x_t)$$



Perception update

$$bel(x_t) = \eta \cdot p(z_t \mid x_t)\overline{bel}(x_t)$$

Markov Localization

Markov Localization

- Localization starting from any unknown position
- Recovers from ambiguous situation
- However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and calculation power can thus become very important if a fine grid is used.

Markov Localization

- Markov localization uses an explicit, discrete representation for the probability of all position in the state space
- This is usually done by representing the environment by a grid or a topographical graph with a finite number of possible states (positions)

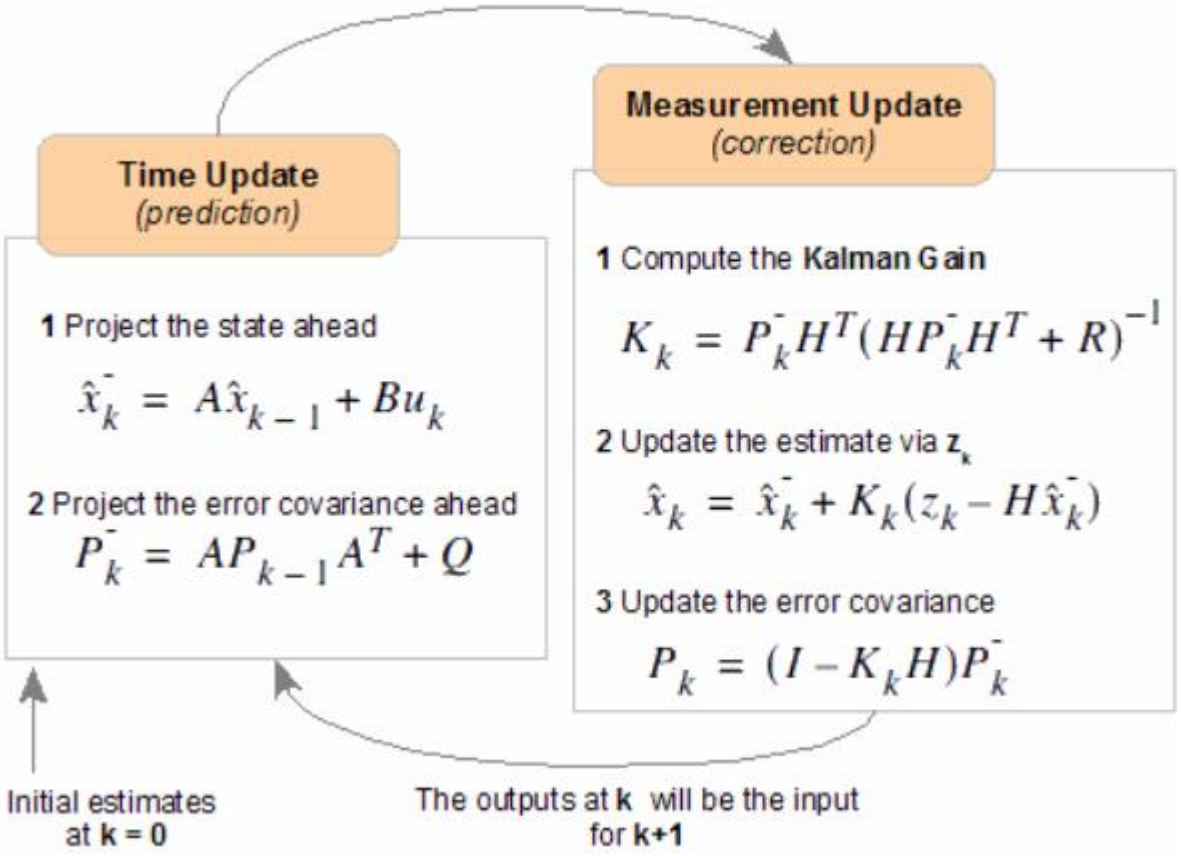
- During each update, the probability for each state (element) of the entire space is updated

Kalman Filter Localization

Kalman Filter Localization

- Robot should know its current position
- Tracks the robot and is inherently very precise and efficient
- However, if the uncertainty of the robot becomes too large (e.g. collision with an object) the kalman filter will fail and the position is definitively lost.

Kalman Filter: Mathematical Model



$$z_k = Hx_k + v_k$$
$$= x_k + v_k$$

KG ↑ means more confidence on measurement model and assign more weights to it