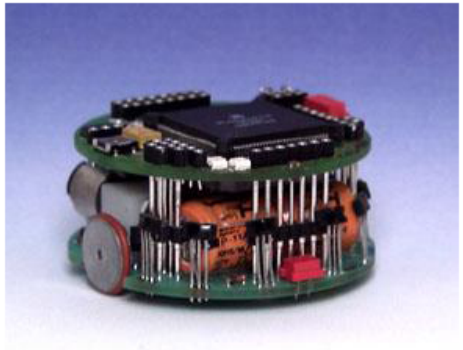
IVR 2nd Assignment
Robot Control

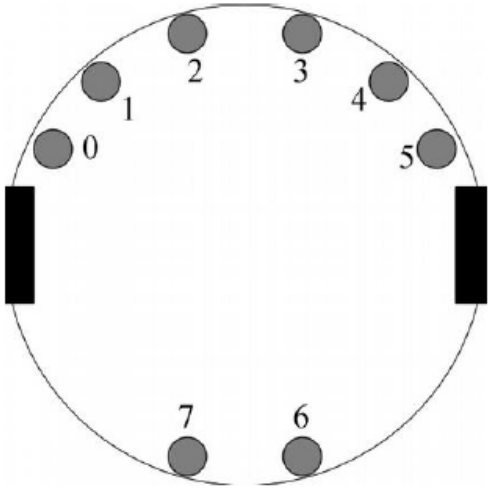
M. Herrmann

informatics

Proximity and range sensors

- Again main function is position: distance to object at specific angle to robot
- Typically works by emitting signal and detecting reflection
- Short-range: proximity sensor, e.g. IR





Sensor responses depends on wall reflectivity

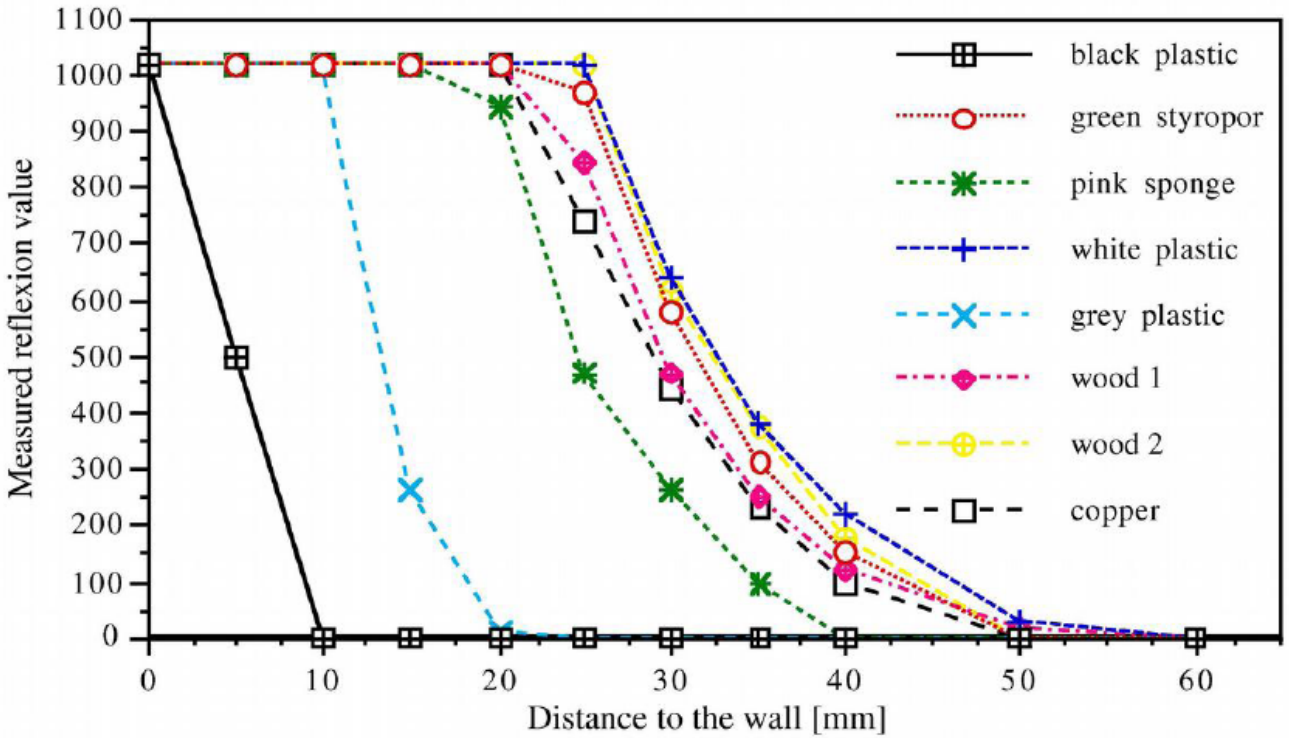
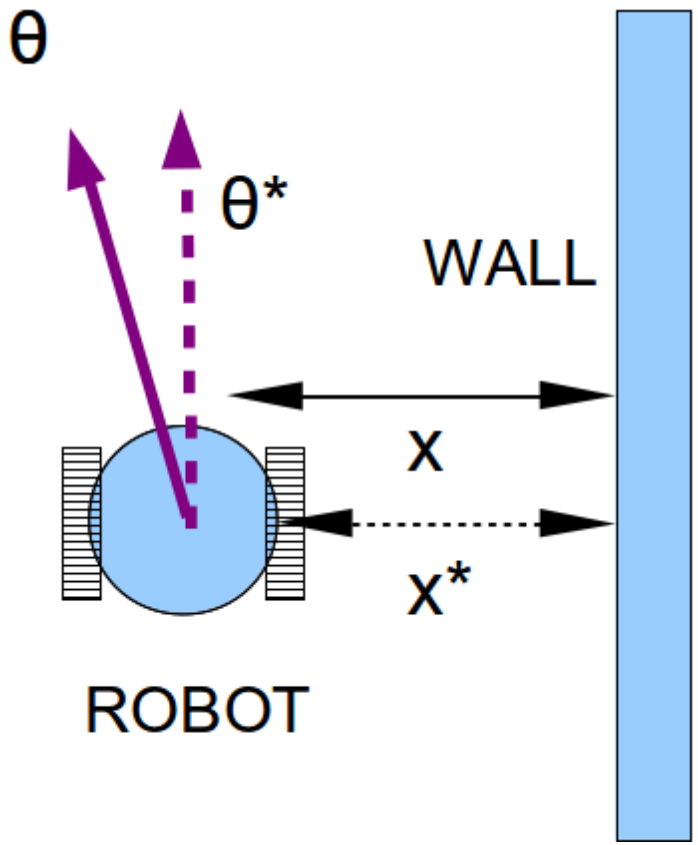


Figure 11: Measurements of the light reflected by various kinds of objects versus the distance to the object.

Proportional (P) Control

- Measure deviations from the desired state in order to stabilise this state
- Control signal proportional to the deviation
- Sign of the proportionality constant (K_p) must be known as well as its order of magnitude
- Fast and proportional reaction to errors
- No need for modelling
- Problems: Steady state error, oscillations about goal state, not predictive, errors may increase before control action is executed, ...

Robot Control



Error in distance to wall
$e = x - x^*$ can be reduced by

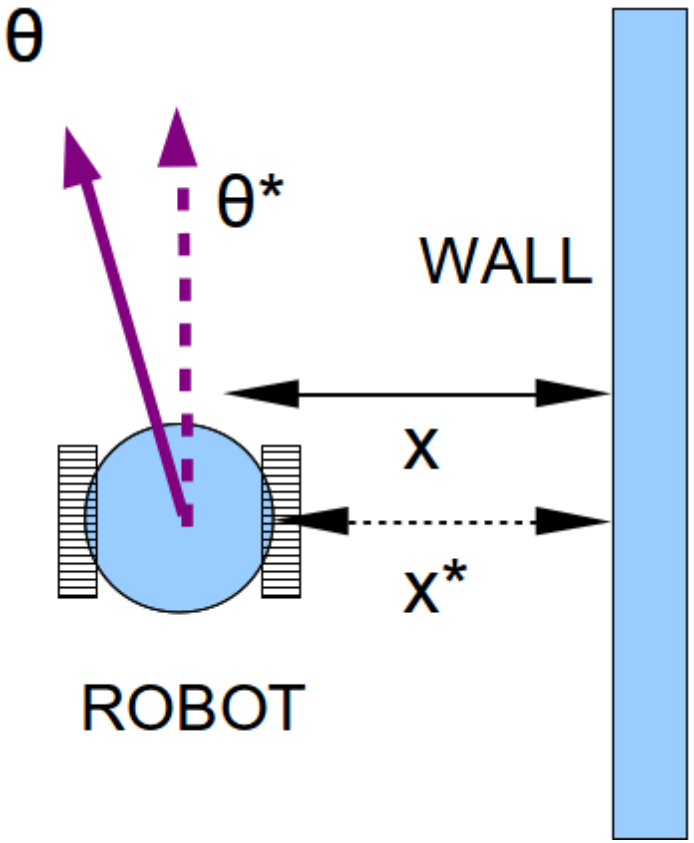
If $x > x^*$, i.e. $e > 0$: increase v_L
If $x < x^*$, i.e. $e < 0$: decrease v_L

Oscillations can be reduced by
"proactive" control
$\Delta e = e(t) - e(t - \Delta t)$:

if $\Delta e > 0$ then increase v_L
if $\Delta e < 0$ then increase v_L

[v_L: speed of left wheel, obviously also the opposite wheel can be controlled (in the opposite direction) or both]

$$\Delta v_L = K_1 \cdot e + K_2 \cdot \Delta e \text{ with } K_1,\ K_2 > 0$$



Notes:

- K_1 would be small (e.g. 0.5)
- K_2 could be larger because Δe is small
- Δe may be noisy, so be careful
- No need to fine-tune K_1 and K_2, but find reasonable values
- ... more next lecture

Odometry: Using self-sensing to estimate position

Odometry: position measurement by distance travelled (from gr. *hodos*, meaning "travel", "journey")

- Know current position (x, y, ϕ)
- Know how much wheels rotate (e.g. current * time)
- New position = old position + commanded motion

But:

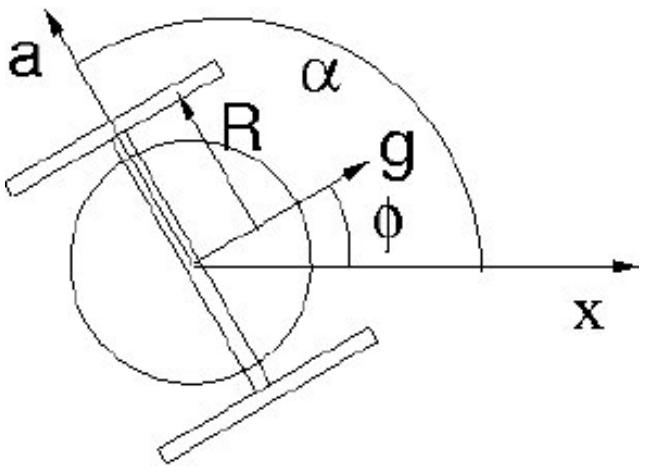
- Motors are inaccurate and low-level controllers may compromise the high-level control command: use shaft encoders
- Wheels slip on surface (curved trajectories are particularly critical): also some feature tracking needed

Odometry: Robot geometry

- α: angle between robot axis and x-axis
- Axis vector: $a = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$
- Heading vector: $g = \begin{pmatrix} \sin\alpha \\ -\cos\alpha \end{pmatrix}$
- a and g are unit vectors
- Heading angle: $\phi = \alpha - \pi/2$.
- Robot position: $r = (x, y)$ (vector from the origin) and the wheel centres are at: (R: half-distance between wheels)

$$r_l = r + Ra \text{ and } r_r = r - Ra$$

- i.e. $r = r_l + r_r$
- Unit of measurement: wheel circumference



Odometry

Assume wheel speeds s_l, s_r are constant for some short time Δt.

If both speeds are equal its easy: $\Delta r = s_l \Delta t = s_r \Delta t$ and $\Delta\phi = 0$

If the speeds are different then during Δt the robot traverses an angle $\Delta\phi = \Delta\alpha$ on the circumference of a circle of radius ρ.

We find:

$$\sin(\Delta\alpha) = \frac{s_r - s_l}{2R}\Delta t$$

$$\sin(\Delta\alpha)\,\rho = \frac{s_l + s_r}{2}\,\Delta t$$

With respect to the center of this circle the robot was previously at

$$r_{\text{old}} = R\frac{s_l + s_r}{s_l - s_r}\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}$$

now it is at

$$r_{\text{new}} = R\frac{s_l + s_r}{s_l - s_r}\begin{pmatrix}\cos\alpha + \Delta\alpha\\ \sin\alpha + \Delta\alpha\end{pmatrix}$$

i.e. the robot moved by an amount $\Delta r = r_{\text{new}} - r_{\text{old}}$

Odometry

$$\Delta r = r_{\text{new}} - r_{\text{old}} = R\frac{s_l + s_r}{s_l - s_r}\Delta t \left(\begin{pmatrix} \cos\alpha + \Delta\alpha \\ \sin\alpha + \Delta\alpha \end{pmatrix} - \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \right)$$

In terms of heading direction $\phi{=}\alpha - \pi/2$ ($\Delta\phi{=}\Delta\alpha$):

$$\Delta r = R\frac{s_l+s_r}{s_l-s_r} \left(\begin{pmatrix} -\sin(\phi + \Delta\phi) \\ \cos(\phi + \Delta\phi) \end{pmatrix} - \begin{pmatrix} -\sin\phi \\ \cos\phi \end{pmatrix} \right) \Delta t$$

Using $\Delta\phi = \Delta\alpha \approx \sin(\Delta\alpha) = \frac{s_r - s_l}{2R}\Delta t$ we get

$$\Delta r = (s_l + s_r)\left(\frac{1}{2\Delta\phi} \begin{pmatrix} -\sin(\phi + \Delta\phi) \\ \cos(\phi + \Delta\phi) \end{pmatrix} - \begin{pmatrix} -\sin\phi \\ \cos\phi \end{pmatrix} \right)$$

For $\Delta t \to 0$ also $\Delta\phi$ becomes small: Take derivative (also for trigs)

$$\begin{aligned} \frac{dx}{dt} &= \frac{s_l+s_r}{2}\cos\phi \\ \frac{dy}{dt} &= -\frac{s_l+s_r}{2}\sin\phi \\ \frac{d\phi}{dt} &= -\frac{s_l-s_r}{2R} \end{aligned}$$

Odometry: Discussion

- Odometry is an internal mechanism that estimates the robot's position relative to a starting point
- Based on proprioceptive information
- Computationally cheap
- Information from self-sensing (shaft encoders)
- Less reliable when done based on motor commands
- Prone to errors: wheel slip, surface roughness, wall contacts, noise, imprecise measurement; errors tend to accumulate
- Calibration reduces systematic errors
- Combine with landmark-based navigation and/or compass!

Exteroceptive alternative: Navigating With Beacons

- Dead reckoning: wheel slip means increasing error
- Periodically observe markers to recalculate position
- Classical style: sense, compute, act
- Beacons for robot: easily identifiable features: IR LEDs, special markers, bar codes

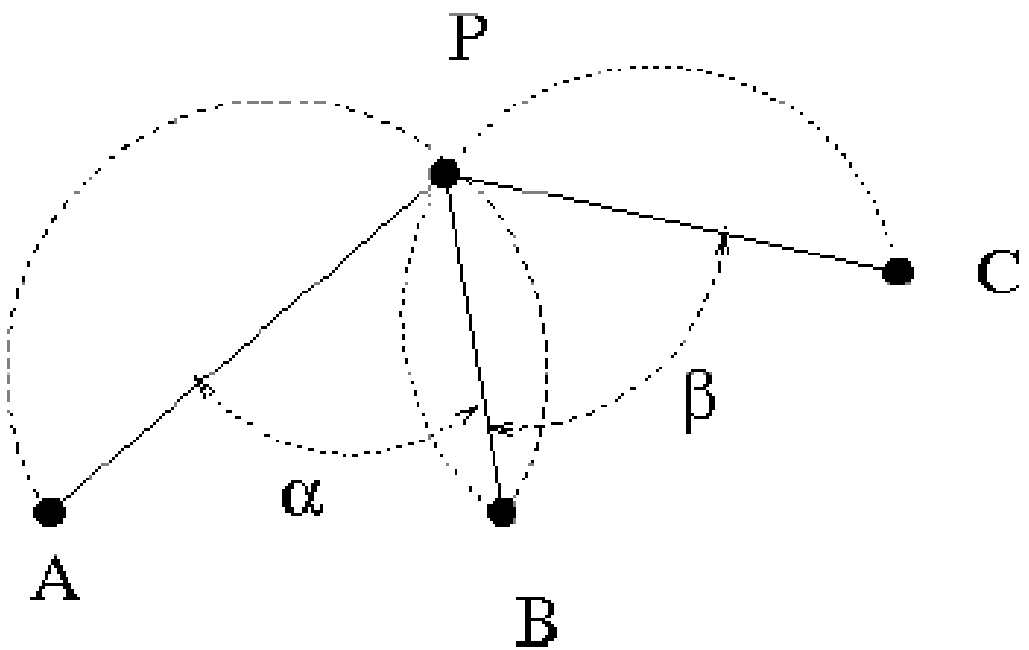
2D Beacon observation

Observe direction to 3 beacons with known positions A, B, C

Measure angle α and β between pairs of beacons

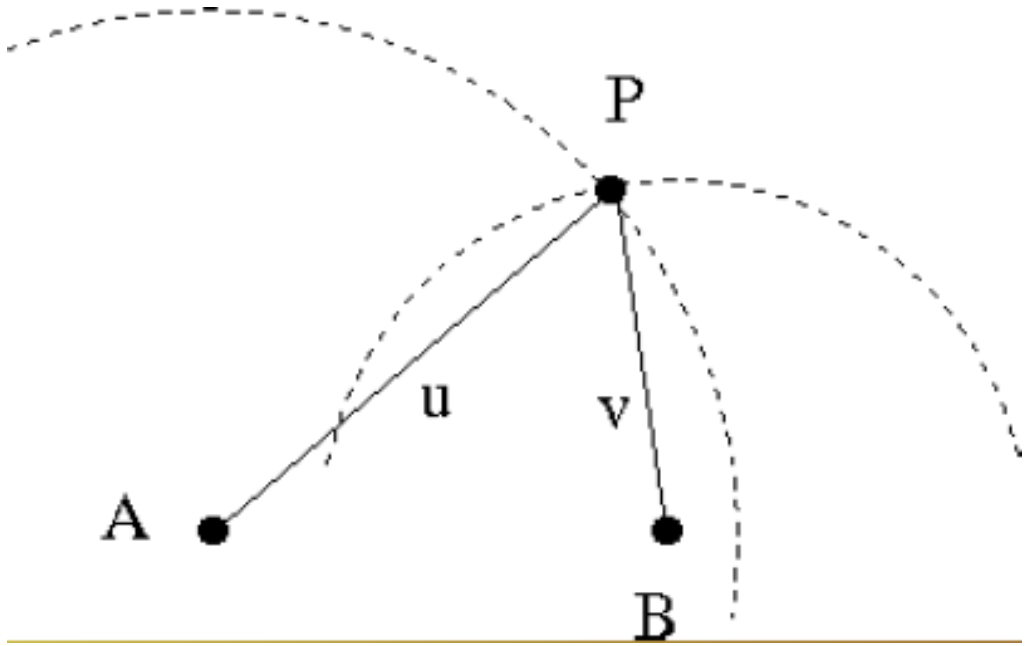
Locate self (P) by triangulation

Need lots of beacons: for accuracy, disambiguation, and existence of an intersection of the peripheral circles



2D Beacons with range

Assume can measure distances (u, v) to 2 beacons (A, B) as well as bearing (e.g. with a range sensor)



Summary on beacons

- Lots of beacons required
- Prior knowledge: Map of beacon location
- Reliable sensors for beacon identification: vision but also others exteroceptive sensors
- choose good beacons in the first place: doorways, corners
- Probabilistic methods: keep several hypotheses
- Computationally costly
- Practically:
  - Odometry for beacon disambiguation and beacons for odometry correction
  - Use also other sensors: GPS, laser range finder, etc.

Subsumption Architecture (Brooks, 1986)