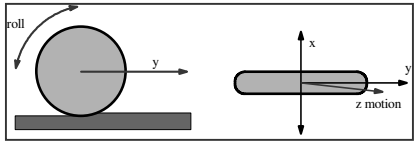
Overview

- Wheeled locomotion
- Reactive paradigm

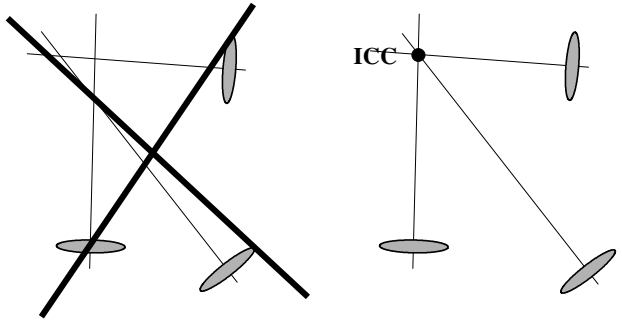
Locomotion of Wheeled Robots

Locomotion (Oxford Dict.):
Power of motion from place to place



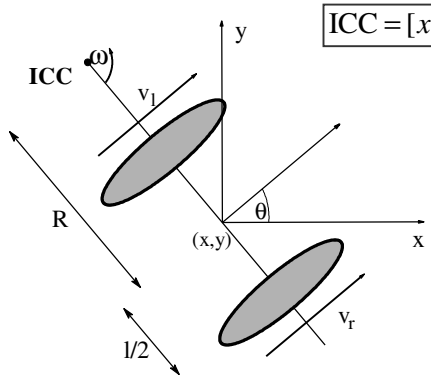
- Differential drive (AmigoBot, Pioneer 2-DX)
- Synchronous drive (B21)
- Car drive (Ackerman steering)
- Holonomic drive (XR4000)
- Tracked vehicles (Urban)

Instantaneous Center of Curvature



- For rolling motion to occur, each wheel has to move along its y-axis

Differential Drive



$$\mathrm{ICC} = [x - R\sin\theta,\ y - R\cos\theta]$$

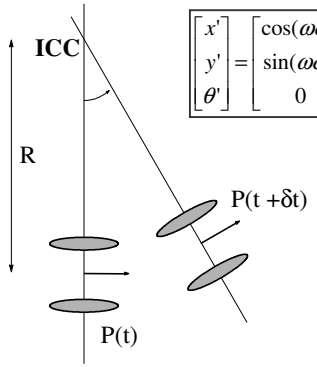
$$\omega(R + l/2) = v_r$$

$$\omega(R - l/2) = v_l$$

$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}$$

$$\omega = \frac{v_r - v_l}{l}$$

Differential Drive



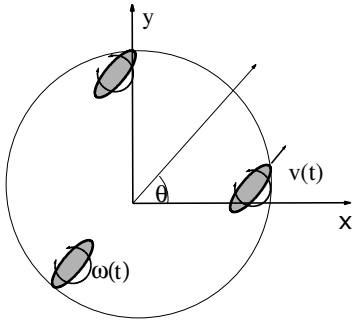
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - \mathrm{ICC}_x \\ y - \mathrm{ICC}_y \\ \theta \end{bmatrix} + \begin{bmatrix} \mathrm{ICC}_x \\ \mathrm{ICC}_y \\ \omega\delta t \end{bmatrix}$$

$$x(t) = \frac{1}{2}\int_0^t [v_r(t') + v_l(t')]\ \cos[\theta(t')]dt'$$

$$y(t) = \frac{1}{2}\int_0^t [v_r(t') + v_l(t')]\ \sin[\theta(t')]dt'$$

$$\theta(t) = \frac{1}{l}\int_0^t [v_r(t') - v_l(t')]\ dt'$$

Synchonous Drive

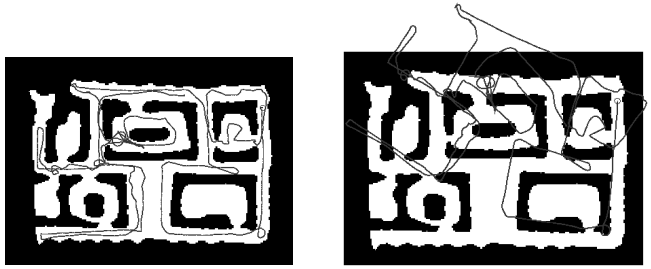


$$x(t) = \int_0^t v(t')\ \cos[\theta(t')]dt'$$

$$y(t) = \int_0^t v(t')\ \sin[\theta(t')]dt'$$

$$\theta(t) = \int_0^t \omega(t')\ dt'$$

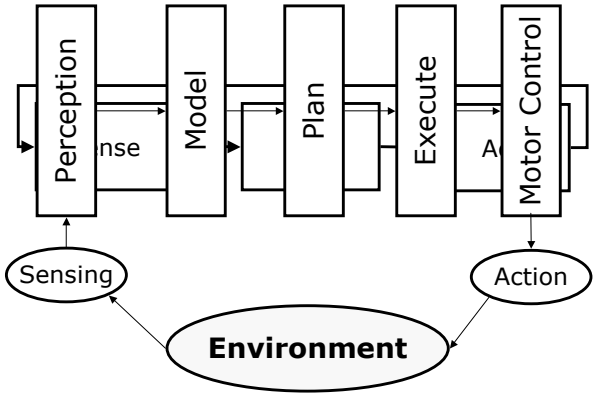
Dead Reckoning



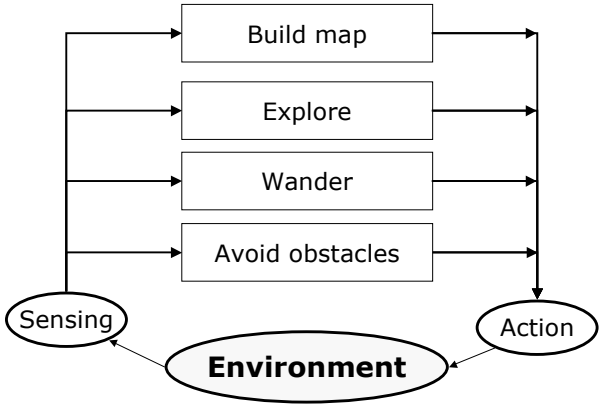
Classical Paradigm Stanford Cart

1. Take nine images of the environment, identify interesting points in one image, and use other images to obtain depth estimates.
2. Integrate information into global world model.
3. Correlate images with previous image set to estimate robot motion.
4. On basis of desired motion, estimated motion, and current estimate of environment, determine direction in which to move.
5. Execute the motion.

Classical Paradigm as Horizontal/Functional Decomposition



Reactive Paradigm as Vertical Decomposition



Characteristics of Reactive Paradigm

- Situated agent, robot is integral part of the world.
- No memory, controlled by what is happening in the world.
- Tight coupling between perception and action via behaviors.
- Only local, behavior-specific sensing is permitted (ego-centric representation).

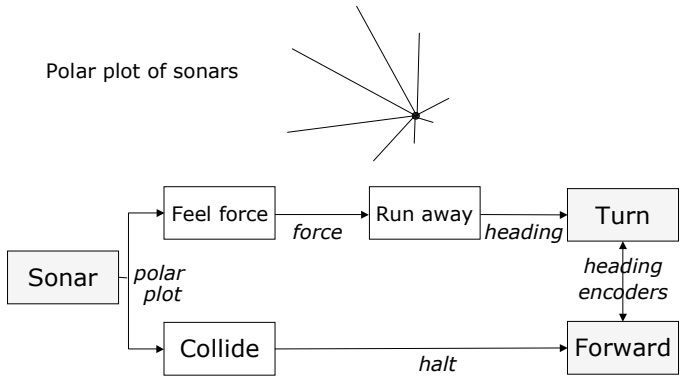
Behaviors

- … are a direct mapping of sensory inputs to a pattern of motor actions that are then used to achieve a task.
- … serve as the basic building block for robotics actions, and the overall behavior of the robot is emergent.
- … support good software design principles due to modularity.

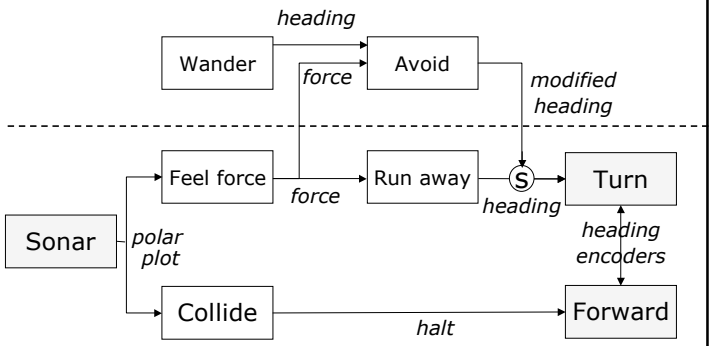
Subsumption Architecture

- Introduced by Rodney Brooks ’86.
- Behaviors are networks of sensing and acting modules (augmented finite state machines AFSM).
- Modules are grouped into layers of competence.
- Layers can subsume lower layers.
- No internal state!

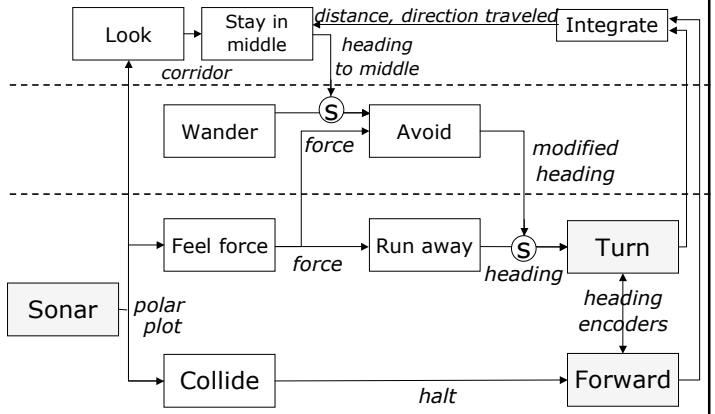
Level 0: Avoid



Level 1: Wander



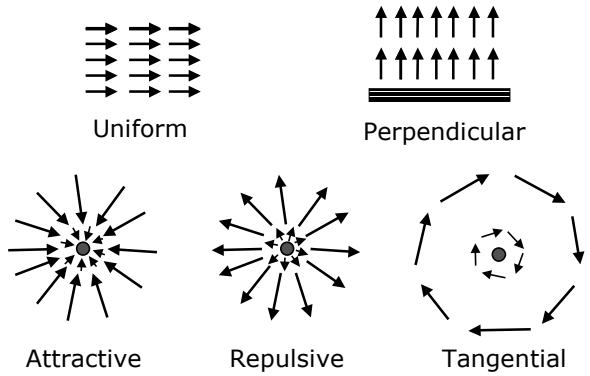
Level 2: Follow Corridor



Potential Field Methodologies

- Treat robot as particle acting under the influence of a potential field
- Robot travels along the derivative of the potential
- Field depends on obstacles, desired travel directions and targets
- Resulting field (vector) is given by the summation of primitive fields
- Strength of field may change with distance to obstacle/target

Primitive Potential Fields

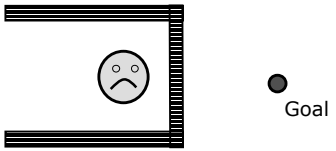


Corridor following with Potential Fields

- Level 0 (collision avoidance)
  is done by the repulsive fields of detected obstacles.
- Level 1 (wander)
  adds a uniform field.
- Level 2 (corridor following)
  replaces the wander field by three fields (two perpendicular, one uniform).

Characteristics of Potential Fields

- Suffer from local minima



- Backtracking
- Random motion to escape local minimum
- Procedural planner s.a. wall following
- Navigation templates (field has desired direction as input)
- Increase potential of visited regions
- Avoid local minima by harmonic functions

Characteristics of Potential Fields

- No preference among layers
- Easy to visualize
- Easy to combine different fields
- High update rates necessary
- Parameter tuning important

Reactive Paradigm

- Advantages over classical paradigm?
- Representations?
- Good software engineering principles?
- Easy to program?
- Robustness?
- Scalability?
- Subsumption vs. potential field?