Agent Architectures

You don't need to implement an intelligent agent as:

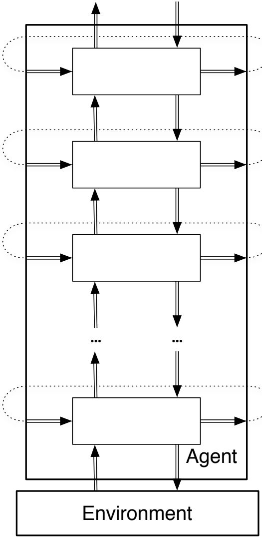
as three independent modules, each feeding into the the next.

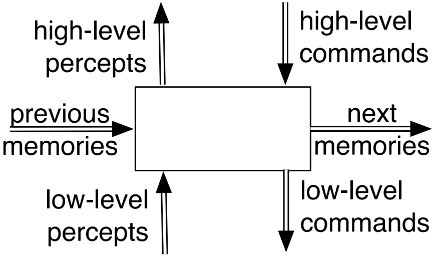
- It's too slow.
- High-level strategic reasoning takes more time than the reaction time needed to avoid obstacles.
- The output of the perception depends on what you will do with it.

Hierarchical Control

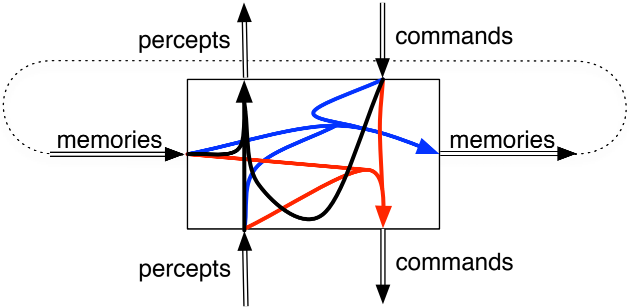
- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.
- The lower-level controllers can
  - run much faster, and react to the world more quickly
  - deliver a simpler view of the world to the higher-level controllers.

Hierarchical Robotic System Architecture





Functions implemented in a layer

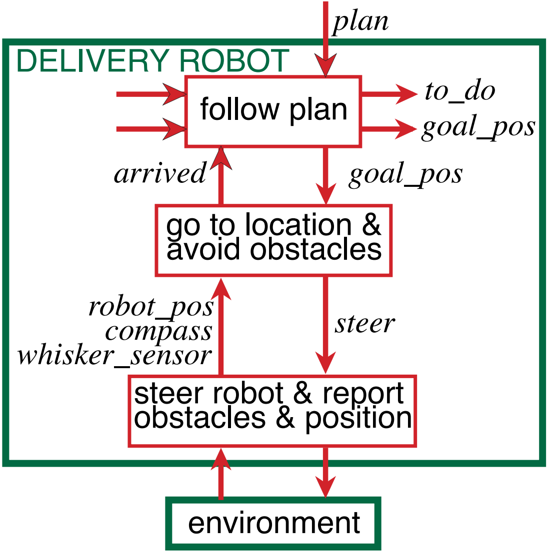


- memory function
  $remember(memory, percept, command)$
- command function
  $do(memory, percept, command)$
- percept function
  $higher\_percept(memory, percept, command)$

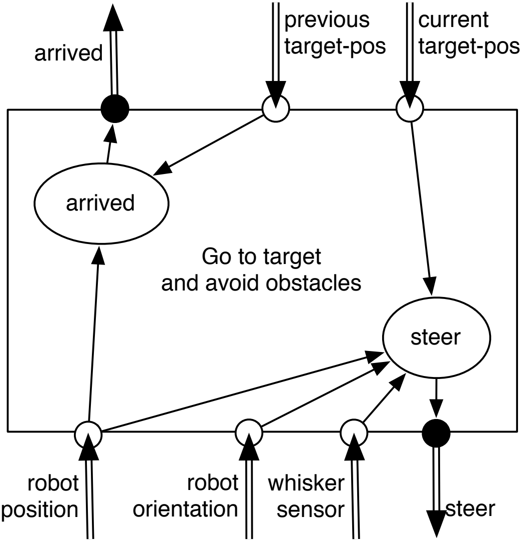
Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).
- It can be given a plan consisting of sequence of named locations for the robot to go to in turn.
- The robot must avoid obstacles.
- It has a single whisker sensor pointing forward and to the right. The robot can detect if the whisker hits an object. The robot knows where it is.
- The obstacles and locations can be moved dynamically. Obstacles and new locations can be created dynamically.

A Decomposition of the Delivery Robot



Middle Layer



Middle Layer of the Delivery Robot

if $whisker\_sensor = on$

  then $steer = left$

else if $straight\_ahead(robot\_pos, robot\_dir, current\_goal\_pos)$

  then $steer = straight$

else if $left\_of(robot\_position, robot\_dir, current\_goal\_pos)$

  then $steer = left$

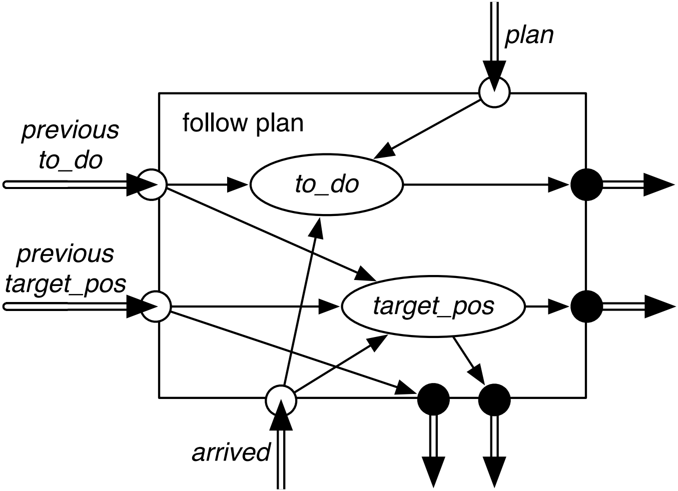
else $steer = right$

$$\begin{aligned} arrived &= distance(previous\_goal\_pos, robot\_pos) \\ &\quad < threshold \end{aligned}$$

Top Layer of the Delivery Robot

- The top layer is given a plan which is a sequence of named locations.
- The top layer tells the middle layer the goal position of the current location.
- It has to remember the current goal position and the locations still to visit.
- When the middle layer reports the robot has arrived, the top layer takes the next location from the list of positions to visit, and there is a new goal position.

Top Layer


plan
follow plan
previous
to_do
to_do
previous
target_pos
target_pos
arrived

Code for the top layer

The top layer has two belief state variables:

- to_do is the list of all pending locations
- $goal\_pos$ is the current goal position

if $arrived$

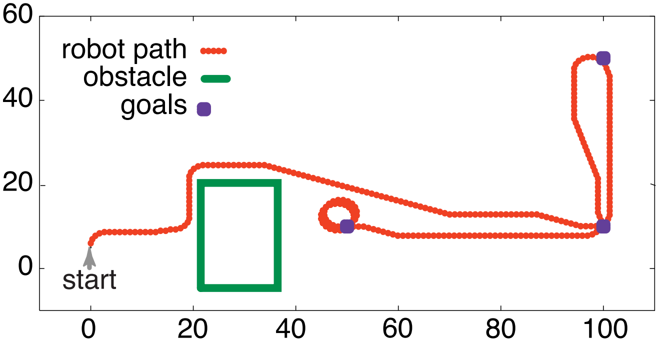
  then $goal\_pos = coordinates(head(to\_do'))$.

if $arrived$

  then $to\_do = tail(to\_do')$.

Here to_do' is the previous value for the to_do feature.

Simulation of the Robot



$$to\_do = [goto(o109), goto(storage), goto(o109),$$
$$goto(o103)]$$
$$arrived = true$$

What should be in an agent's belief state?

- An agent decides what to do based on its belief state and what it observes.
- A purely reactive agent doesn't have a belief state.
  A dead reckoning agent doesn't perceive the world.
  — neither work very well in complicated domains.
- It is often useful for the agent's belief state to be a model of the world (itself and the environment).