AI Unit-5.3: ROBOT:

Robots are physical agents that perform tasks by manipulating the physical world.

Effectors have a single purpose that to assert physical forces on the environment. Robots are also equipped with **sensors**, which allow them to perceive their environment.

Most of today's robots fall into one of three primary categories.

1.MANIPULATORS:

Manipulator motion usually involves a chain of controllable joints, enabling such robots to place their effectors in any position within the workplace. Few car manufacturers could survive without robotic manipulators, and some manipulators have even been used to generate original artwork.

2.MOBBILE ROBOT:

The second category is the **mobile robot**. Mobile robots move about their environment using wheels, legs, or similar mechanisms. They have been put to use delivering food in hospitals, moving containers at loading docks, and similar tasks. Other types of mobile robots include **unmanned air vehicles, Autonomous underwater vehicles etc..,**

3.MOBILE MANIPULATOR:

The third type of robot combines mobility with manipulation, and is often called a **mobile manipulator**. **Humanoid robots** mimic the human torso.

The field of robotics also includes prosthetic devices , intelligent environments and multibody systems, wherein robotic action is achieved through swarms of small cooperating robots. Robotics brings together many of the concepts we have seen earlier in the book, including probabilistic state estimation, perception, planning, unsupervised learning, and reinforcement learning.

ROBOT HARDWARE:

The robot hardware mainly depends on 1.sensors and 2.effectors

1.sensors:

Sensors are the perceptual interface between robot and environment.

PASSIVE SENSOR: Passive sensors, such as cameras, are true observers of the environment: they capture signals that are generated by other sources in the environment.

ACTIVE SENSOR: Active sensors, such as sonar, send energy into the environment. They rely on the fact that this energy is reflected back to the sensor.

Range finders are sensors that measure the distance to nearby objects. In the early days of robotics, robots were commonly equipped with **sonar sensors**. Sonar sensors emit directional sound waves, which are reflected by objects, with some of the sound making it back into the sensor.

Stereo vision relies on multiple cameras to image the environment from slightly different viewpoints, analyzing the resulting parallax in these images to compute the range of surrounding objects.

Other common range sensors include radar, which is often the sensor of choice for UAVs. Radar sensors can measure distances of multiple kilometers. On the other extreme end of range sensing are **tactile sensors** such as whiskers, bump panels, and touch-sensitive skin.

A second important class of sensors is **location sensors**. Most location sensors use range sensing as a primary component to determine location. Outdoors, the **Global Positioning System** (GPS) is the most common solution to the localization problem. GPS measures the distance to satellites that emit pulsed signals.

The third important class is **proprioceptive sensors**, which inform the robot of its own motion. To measure the exact configuration of a robotic joint, motors are often equipped with **shaft decoders** that count the revolution of motors in small increments.

Other important aspects of robot state are measured by **force sensors** and **torque sensors**. These are indispensable when robots handle fragile objects or objects whose exact shape and location is unknown.

EFFECTORS:

Effectors are the means by which robots move and change the shape of their bodies. To understand the design of effectors we use the concept of degree of freedom.

We count one degree of freedom for each independent direction in which a robot, or one of its effectors, can move. For example, a rigid mobile robot such as an AUV has six degrees of freedom, three for its (x, y, z) location in space and three for its angular orientation, known as *yaw*, *roll*, and *pitch*. These six degrees define the **kinematic state**2 or **pose** of the robot. The **dynamic state** of a robot includes these six plus an additional six dimensions for the rate of change of each kinematic dimension, that is, their velocities.

For nonrigid bodies, there are additional degrees of freedom within the robot itself. For example, the elbow of a human arm possesses two degree of freedom. It can flex the upper arm towards or away, and can rotate right or left. The wrist has three degrees of freedom. It can move up and down, side to side, and can also rotate. Robot joints also have one, two, or three degrees of freedom each. Six degrees of freedom are required to place an object, such as a hand, at a particular point in a particular orientation.

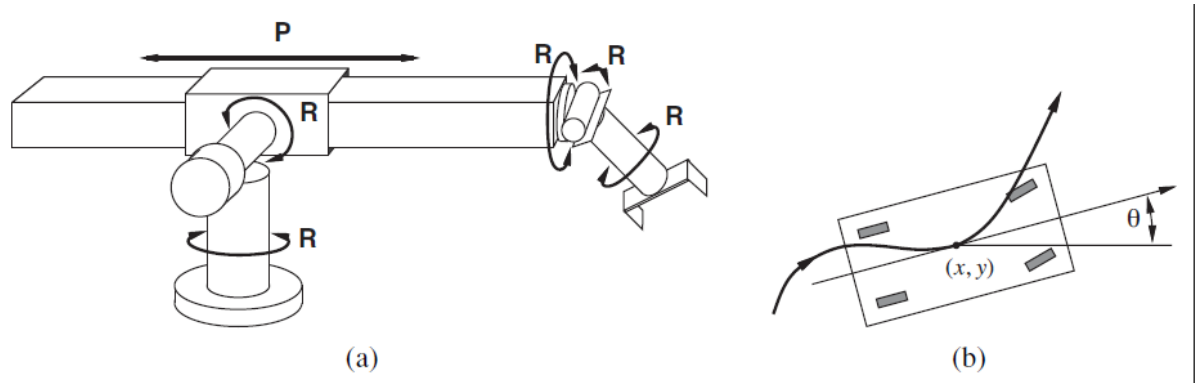


Figure 25.4 (a) The Stanford Manipulator, an early robot arm with five revolute joints (R) and one prismatic joint (P), for a total of six degrees of freedom. (b) Motion of a nonholonomic four-wheeled vehicle with front-wheel steering.

In the fig 4(a) has exactly six degrees of freedom, created REVOLUTE JOINT by five **revolute joints** that generate rotational motion and one **prismatic joint** that generates sliding motion

For mobile robots, the DOFs are not necessarily the same as the number of actuated elements.
Consider, for example, your average car: it can move forward or backward, and it can turn, giving it two DOFs. In contrast, a car's kinematic configuration is three-dimensional: on an open flat surface, one can easily maneuver a car to any (x, y) point, in any orientation. (See Figure 25.4(b).) Thus, the car has three **effective degrees of freedom** but two **control label degrees of freedom**. We say a robot is **nonholonomic** if it has more effective DOFs than controllable DOFs and **holonomic** if the two numbers are the same.

Sensors and effectors alone do not make a robot. A complete robot also needs a source of power to drive its effectors. The **electric motor** is the most popular mechanism for both manipulator actuation and locomotion, but **pneumatic actuation** using compressed gas and **Hydraulic actuation** using pressurized fluids also have their application niches.

ROBOTIC PERCEPTION:

Perception is the process by which robots map sensor measurements into internal representations of the environment. Perception is difficult because sensors are noisy, and the environment is partially observable, unpredictable, and often dynamic.

As a rule of thumb, good internal representations for robots have three properties: they contain enough information for the robot to make good decisions, they are structured so that they can be updated efficiently, and they are natural in the sense that internal variables correspond to natural state variables in the physical world.

For robotics problems, we include the robot's own past actions as observed variables in the model. Figure 25.7 shows the notation used in this

chapter: **X**t is the state of the environment (including the robot) at time t, **Z**t is the observation received at time t, and At is the action taken after the observation is received.

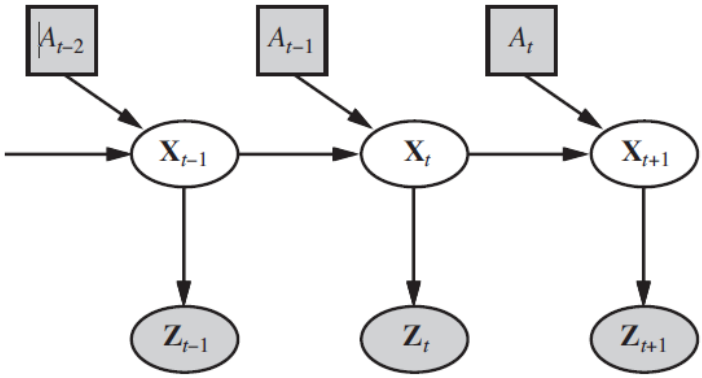


Figure 25.7 Robot perception can be viewed as temporal inference from sequences of actions and measurements, as illustrated by this dynamic Bayes network.

We would like to compute the new belief state, **P**(**X**t+1 | **z**1:t+1, a1:t), from the current belief state **P**(**X**t | **z**1:t, a1:t−1) and the new observation **z**t+1. Thus, we modify the recursive filtering equation (15.5 on page 572) to use integration rather than summation:

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{z}_{1:t+1}, a_{1:t})$$
$$= \alpha\, \mathbf{P}(\mathbf{z}_{t+1} \mid \mathbf{X}_{t+1}) \int \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, a_t)\, P(\mathbf{x}_t \mid \mathbf{z}_{1:t}, a_{1:t-1})\, d\mathbf{x}_t \quad (25.1)$$

This equation states that the posterior over the state variables **X** at time t + 1 is calculated recursively from the corresponding estimate one time step earlier. This calculation involves the previous action at and the current sensor measurement **z**t+1. The probability **P**(**X**t+1 | **x**t, at) is called the **transition model** or **motion model**, and **P**(**z**t+1 | **X** t+1) is the **sensor model**.

1.Localization and mapping

Localization is the problem of finding out where things are—including the robot itself.

Knowledge about where things are is at the core of any successful physical interaction with the environment.

To keep things simple, let us consider a mobile robot that moves slowly in a flat 2D world. Let us also assume the robot is given an exact map of the environment. The pose of such a mobile robot is defined by its two Cartesian coordinates with values x and y and its heading with value θ, as illustrated in Figure 25.8(a). If we arrange those three values in a vector, then any particular state is given by $\mathbf{X}t = (xt, yt, \theta t)^T$.

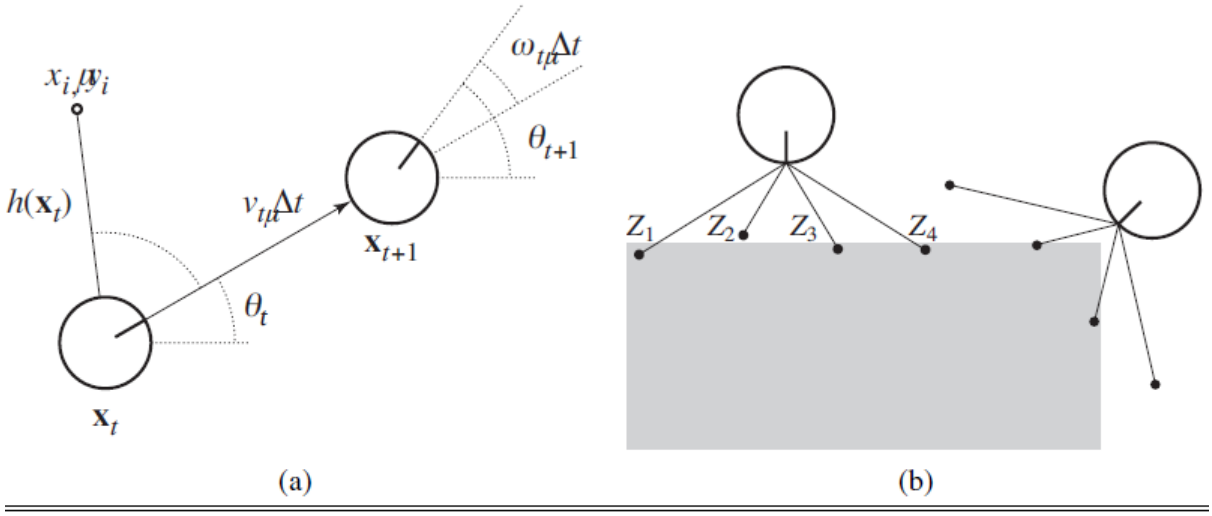


Figure 25.8 (a) A simplified kinematic model of a mobile robot. The robot is shown as a circle with an interior line marking the forward direction. The state $\mathbf{x}_t$ consists of the (x_t, y_t) position (shown implicitly) and the orientation θ_t. The new state $\mathbf{x}_{t+1}$ is obtained by an update in position of $v_t\Delta_t$ and in orientation of $\omega_t\Delta_t$. Also shown is a landmark at (x_i, y_i) observed at time t. (b) The range-scan sensor model. Two possible robot poses are shown for a given range scan (z_1, z_2, z_3, z_4). It is much more likely that the pose on the left generated the range scan than the pose on the right.

In the kinematic approximation, each action consists of the "instantaneous" specification of two velocities—a translational velocity v_t and a rotational velocity ω_t. For small time intervals Δt, a crude deterministic model of the motion of such robots is given by

$$\hat{\mathbf{X}}_{t+1} = f(\mathbf{X}_t, \underbrace{v_t, \omega_t}_{a_t}) = \mathbf{X}_t + \begin{pmatrix} v_t\Delta t \cos\theta_t \\ v_t\Delta t \sin\theta_t \\ \omega_t\Delta t \end{pmatrix} .$$

The notation $\hat{\mathbf{X}}$ refers to a deterministic state prediction. Of course, physical robots are somewhat unpredictable. This is commonly modeled by a Gaussian distribution with mean $f(\mathbf{X}_t, v_t, \omega_t)$ and covariance $\mathbf{\Sigma}_x$. (See Appendix A for a mathematical definition.)

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t, v_t, \omega_t) = N(\hat{\mathbf{X}}_{t+1}, \mathbf{\Sigma}_x) .$$

Next, we need a sensor model. We will consider two kinds of sensor model. The first assumes that the sensors detect *stable, recognizable* features of the environment called **landmarks**. The exact prediction of the observed range and bearing would be

$$\hat{\mathbf{z}}_t = h(\mathbf{x}_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan\frac{y_i - y_t}{x_i - x_t} - \theta_t \end{pmatrix}$$

Again, noise distorts our measurements. To keep things simple, one might assume Gaussian noise with covariance $\mathbf{\Sigma}_z$, giving us the sensor model

$$P(\mathbf{z}t \mid \mathbf{x}t) = N(\hat{}\mathbf{z}t, \mathbf{\Sigma}z) .$$

function MONTE-CARLO-LOCALIZATION(a, z, N, $P(X'|X,\ v,\ \omega)$, $P(z|z^*)$, m) **returns** a set of samples for the next time step

inputs: a, robot velocities v and ω
z, range scan $z_1, \ldots, z_M$
$P(X'|X,\ v,\ \omega)$, motion model
$P(z|z^*)$, range sensor noise model
m, 2D map of the environment

persistent: S, a vector of samples of size N

local variables: W, a vector of weights of size N
S', a temporary vector of particles of size N
W', a vector of weights of size N

if S is empty **then** /* initialization phase */
  for $i = 1$ to N **do**
    $S[i] \leftarrow$ sample from $P(X_0)$
  for $i = 1$ to N **do** /* update cycle */
    $S'[i] \leftarrow$ sample from $P(X'|X = S[i], v, \omega)$
    $W'[i] \leftarrow 1$
    for $j = 1$ to M **do**
      $z^* \leftarrow$ RAYCAST($j, X = S'[i], m$)
      $W'[i] \leftarrow W'[i] \cdot P(z_j | z^*)$
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S', W')
return S

Figure 25.9 A Monte Carlo localization algorithm using a range-scan sensor model with independent noise.

This problem is important for many robot applications, and it has been studied extensively under the name **simultaneous localization and mapping**, abbreviated as **SLAM**.

SLAM problems are solved using many different probabilistic techniques, including the extended Kalman filter

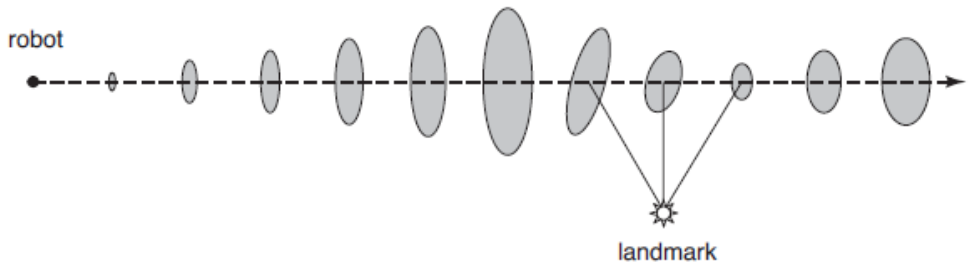


Figure 25.12 Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.

Expectation–maximization is also used for SLAM.

2.Other types of perception

Not all of robot perception is about localization or mapping. Robots also perceive the temperature, odors, acoustic signals, and so on. Many of these quantities can be estimated using variants of dynamic Bayes networks.

It is also possible to program a robot as a reactive agent, without explicitly reasoning about probability distributions over states.

3.Machine learning in robot perception

Machine learning plays an important role in robot perception. This is particularly the case when the best internal representation is not known. One common approach is to map high dimensional sensor streams into lower-dimensional spaces using unsupervised machine learning method. Such an approach is called **low-dimensional embedding**.

Methods that make robots collect their own training data are called **Self Supervised**.

In this instance, the robot uses machine learning to leverage a short-range sensor that works well for terrain classification into a sensor that can see much farther. That allows the robot to drive faster, slowing down only when the sensor model says there is a change in the terrain that needs to be examined more carefully by the short-range sensors.

PLANNING TO MOVE:

All of a robot's deliberations ultimately come down to deciding how to move effectors. The **point-to-point motion** problem is to deliver the robot or its end effector to a designated target location. A greater challenge is the **compliant motion** problem, in which a robot moves while being in physical contact with an obstacle.

There are two main approaches: **cell decomposition** and **skeletonization**. Each reduces the continuous path-planning problem to a discrete graph-search problem.

1 Configuration space

We will start with a simple representation for a simple robot motion problem. It has two joints that move independently. the robot's configuration can be described by a four dimensional coordinate: (x_e, y_e) for the location of the elbow relative to the environment and (x_g, y_g) for the location of the gripper. They constitute what is known as **workspace representation**.

The problem with the workspace representation is that not all workspace coordinates are actually attainable, even in the absence of obstacles. This is because of the **linkage constraints** on the space of attainable workspace coordinates.

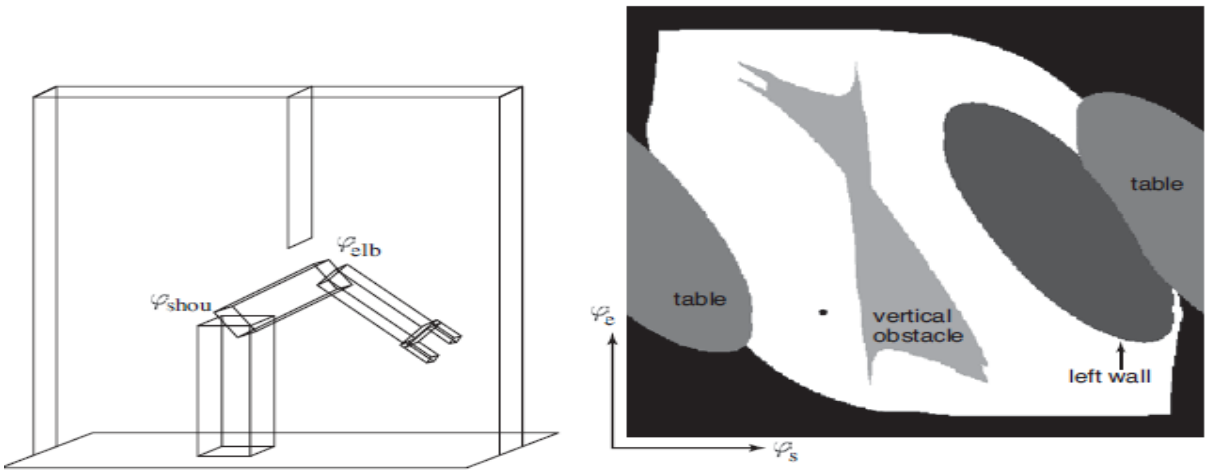


Figure 25.14 (a) Workspace representation of a robot arm with 2 DOFs. The workspace is a box with a flat obstacle hanging from the ceiling. (b) Configuration space of the same robot. Only white regions in the space are configurations that are free of collisions. The dot in this diagram corresponds to the configuration of the robot shown on the left.

Transforming configuration space coordinates into workspace coordinates is simple: it involves a series of straightforward coordinate transformations. These transformations are linear for prismatic joints and trigonometric for revolute joints. This chain of coordinate transformation is known as **kinematics**.

The inverse problem of calculating the configuration of a robot whose effector location is specified in workspace coordinates is known as **inverse kinematics.**

2 Cell decomposition methods

The first approach to path planning uses **cell decomposition**—that is, it decomposes the free space into a finite number of contiguous regions, called cells.

A decomposition has the advantage that it is extremely simple to implement, but it also suffers from three limitations. First, it is workable only for low-dimensional configuration spaces, Second, there is the problem of what to do with cells that are "mixed", And third, any path through a discretized state space will not be smooth.

Cell decomposition methods can be improved in a number of ways, to alleviate some of these problems. The first approach allows *further subdivision* of the mixed cells—perhaps using cells of half the original size. A second way to obtain a complete algorithm is to insist on an **exact cell decomposition** of the free space.

3 Modified cost functions:

This problem can be solved by introducing a **potential field**. A potential field is a function defined over state space, whose value grows with the distance to the closest obstacle. The potential field can be used as an additional cost term in the shortest-path calculation. This induces an interesting trade off. On the one hand, the robot seeks to minimize path length to the goal. On the other hand, it

tries to stay away from obstacles by virtue of minimizing the potential function. Clearly, the resulting path is longer, but it is also safer.

There exist many other ways to modify the cost function. However, it is often easy to smooth the resulting trajectory after planning, using conjugate gradient methods. Such post-planning smoothing is essential in many real world applications.

4 Skeletonization methods

The second major family of path-planning algorithms is based on the idea of **skeletonization**.

These algorithms reduce the robot's free space to a one-dimensional representation, for which the planning problem is easier. This lower-dimensional representation is called a **skeleton** of the configuration space.

Voronoi graph of the free space—the set of all points that are equidistant to two or more obstacles. To do path planning with a Voronoi graph, the robot first changes its present configuration to a point on the Voronoi graph. It is easy to show that this can always be achieved by a straight-line motion in configuration space. Second, the robot follows the Voronoi graph until it reaches the point nearest to the target configuration. Finally, the robot leaves the Voronoi graph and moves to the target. Again, this final step involves straight-line motion in configuration space.

An alternative to the Voronoi graphs is the **probabilistic roadmap**, a skeletonization approach that offers more possible routes, and thus deals better with wide-open spaces. With these improvements, probabilistic roadmap planning tends to scale better to high-dimensional configuration spaces than most alternative path-planning techniques.

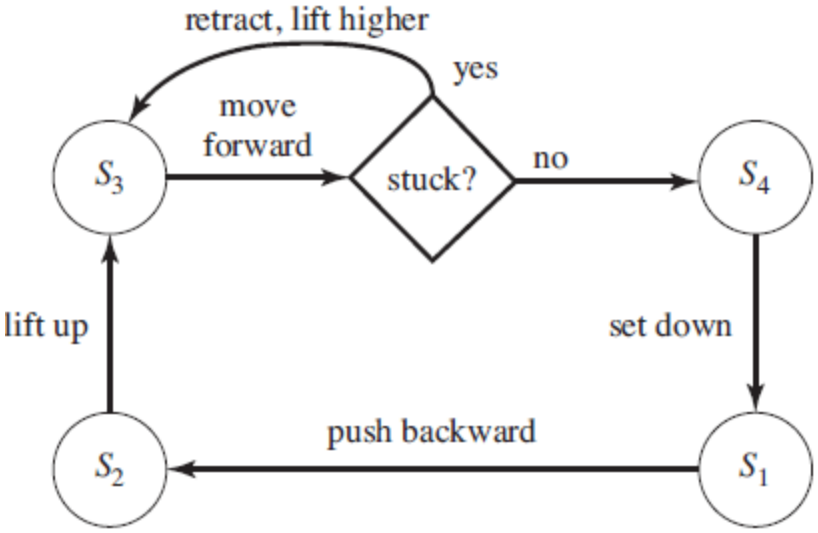
ROBOTIC SOFTWARE ARCHITECTURE:

A methodology for structuring algorithms is called a **software architecture**. An architecture includes languages and tools for writing programs, as well as an overall philosophy for how programs can be brought together. Architectures that combine reactive and deliberate techniques are called **hybrid architectures**.

1 Subsumption architecture

The **subsumption architecture** is a framework for assembling reactive controllers out of finite state machines. Nodes in these machines may contain tests for certain sensor variables, in which case the execution trace of a finite state machine is conditioned on the outcome of such a test. The resulting machines are refereed to as **augmented finite state machines**, or AFSMs, where the augmentation refers to the use of clocks.

An example of a simple AFSM is the four-state machine shown in BELOW Figure, which generates cyclic leg motion for a hexapod walker.



(b)

In our example, we might begin with AFSMs for individual legs, followed by an AFSM for coordinating multiple legs. On top of this, we might implement higher-level behaviors such as collision avoidance, which might involve backing up and turning.

Unfortunately, the subsumption architecture has its own problems. First, the AFSMs are driven by raw sensor input, an arrangement that works if the sensor data is reliable and contains all necessary information for decision making, but fails if sensor data has to be integrated in nontrivial ways over time. A subsumption style robot usually does just one task, and it has no notion of how to modify its controls to accommodate different goals. Finally, subsumption style controllers tend to be difficult to understand.

However, it has had an influence on other architectures, and on individual components of some architectures.

2 Three-layer architecture

Hybrid architectures combine reaction with deliberation. The most popular hybrid architecture is the **three-layer architecture**, which consists of a reactive layer, an executive layer, and a deliberative layer.

The **reactive layer** provides low-level control to the robot. It is characterized by a tight sensor–action loop. Its decision cycle is often on the order of milliseconds.

The **executive layer** (or sequencing layer) serves as the glue between the reactive layer and the deliberative layer. It accepts directives by the deliberative layer, and sequences them for the reactive layer.

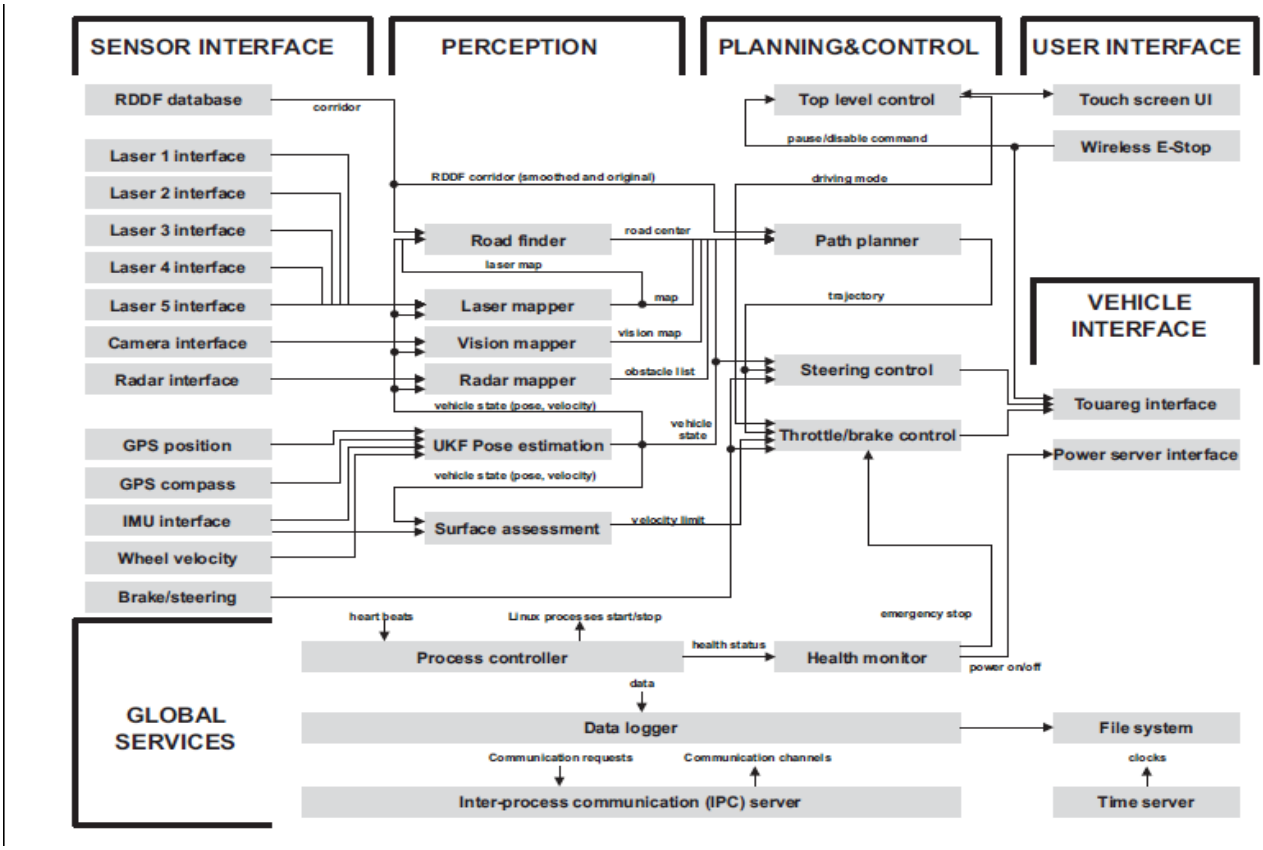
The **deliberative layer** generates global solutions to complex tasks using planning.

Because of the computational complexity involved in generating such solutions, its decision cycle is often in the order of minutes. The deliberative layer (or planning layer) uses models for decision making.

3 Pipeline architecture

Another architecture for robots is known as the **pipeline architecture**. Just like the subsumption architecture, the pipeline architecture executes multiple process in parallel.

Data enters this pipeline at the **sensor interface layer**. The **perception layer** then updates the robot's internal models of the environment based on this data. Next, these models are handed to the **planning and control layer**. Those are then communicated back to the vehicle through the **vehicle interface layer**.



The key to the pipeline architecture is that this all happens in parallel. While the perception layer processes the most recent sensor data, the control layer bases its choices on slightly older data. In this way, the pipeline architecture is similar to the human brain. We don't switch off our motion controllers when we digest new sensor data. Instead, we perceive, plan, and act all at the same time. Processes in the pipeline architecture run asynchronously, and all computation is data-driven. The resulting system is robust, and it is fast.

APPLICTION DOMAINS:

Industry and Agriculture. Traditionally, robots have been fielded in areas that require difficult human labour, yet are structured enough to be amenable to robotic automation. The best example is the assembly line, where manipulators routinely perform tasks such as assembly, part placement, material handling, welding, and painting. In many of these tasks, robots have become more cost-effective than human workers.

Transportation. Robotic transportation has many facets: from autonomous helicopters that deliver payloads to hard-to-reach locations, to automatic wheelchairs that transport people who are unable to control wheelchairs by themselves, to autonomous straddle carriers that outperform skilled human drivers when transporting containers from ships to trucks on loading docks.

Robotic cars. Most of use cars every day. Many of us make cell phone calls while
driving. Some of us even text. The sad result: more than a million people die every year in traffic accidents. Robotic cars like BOSS and STANLEY offer hope: Not only will they make driving much safer, but they will also free us from the need to pay attention to the road during our daily commute.

Health care. Robots are increasingly used to assist surgeons with instrument placement when operating on organs as intricate as brains, eyes, and hearts. Robots have become indispensable tools in a range of surgical procedures, such as hip replacements, thanks to their high precision. In pilot studies, robotic devices have been found to reduce the danger of lesions when performing colonoscopy.

Hazardous environments. Robots have assisted people in cleaning up nuclear waste, most notably in Chernobyl and Three Mile Island. Robots were present after the collapse of the World Trade Center, where they entered structures deemed too dangerous for human search and rescue crews.

Exploration. Robots have gone where no one has gone before, including the surface of Mars. Robotic arms assist astronauts in deploying and retrieving satellites and in building the International Space Station. Robots also help explore under the sea. They are routinely used to acquire maps of sunken ships.

Personal Services. Service is an up-and-coming application domain of robotics. Service robots assist individuals in performing daily tasks. Commercially available domestic service robots include autonomous vacuum cleaners, lawn mowers, and golf caddies. example for robot vaccum cleaner is ROOMBA.

Entertainment. Robots have begun to conquer the entertainment and toy industry. we see **robotic soccer**, a competitive game very much like human soccer, but played with autonomous mobile robots. Robot soccer provides great opportunities for research in AI, since it raises a range of problems relevant to many other, more serious robot applications.

Human augmentation. A final application domain of robotic technology is that of human augmentation. Researchers have developed legged walking machines that can carry people around, very much like a wheelchair. Several research efforts presently focus on the development of devices that make it easier for people to walk or move their arms by providing additional forces through extra skeletal attachments.