04

Robot control architectures

control architectures

Robot control architectures are conceptual structures for organizing robot control such that we can design controllers systematically. All such architectures include maps of measurements to actions, a process that was central to our definition of intelligence (Lecture 01.05). We call this process *sense-decide-act* (SDA). With reference to Figure 04.1, sensing (measurement) provides the robot with information about the state of itself and the environment; from this, a decision is made about how the robot should act; finally, the robot acts. The differences among robot control architectures lie almost entirely in the *decide* step—that is, in the *controller*.

SDA

controller

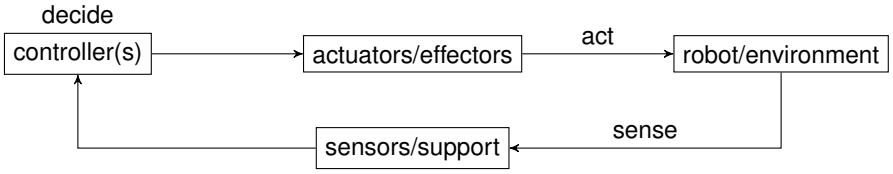

Figure 04.1: a block diagram showing the sense-decide-act structure common to all robot control architectures.

The "controller" here is not necessarily a single device, although it can be. Control devices are frequently *microcontrollers* that include *microprocessors*, *memory*, and input/output interfaces. However, some control logic is so simple, it can be instantiated in analog- or digital-circuits alone. It is also notable that the diagram of Figure 04.1 encompasses processes that can be happening asynchronously and in parallel. For instance, measurements may be made at different times, controller decisions may take different times for different situations, etc.

microcontrollers
microprocessors
memory

From our understanding of feedback control theory,[1] we can conceive of how we might control simple robot actions, such as turning by some angle or raising an effector to some height. While feedback control systems of complex systems (like a robot arm) can be very complicated, they typically require *low-level commands*, i.e. a goal state through time.

low-level commands

Actions, tasks, and behaviors

As necessary as feedback control is, it is inadequate to command the robot to perform complex actions, such as finding an object or exploring an environment—i.e. *high-level commands*. But just such high-level commands are what a designer would like to give a robot. Sometimes, we say there are *mid-level commands* as well, those that require more than low-level commands, but are probably lower-level than a robot designer would like to give. In fact, we can categorize actions by command complexity.

high-level commands

mid-level commands

Simple actions are those that require only low-level commands. For instance, moving an effector to a given state is a simple action.

Tasks are actions that require only mid-level commands. For instance, grasping an object in a gripper is a task.

Behaviors are actions that require only high-level commands. For instance, following walls is a behavior.

These categorizations are helpful, as we'll see, despite their ambiguity.

Models and their representation

models

Some robot control architectures use internal *models* to help the controller to decide what to do. These models are typically mathematical models, maps of the environment, and mechanical solid models. Models, of course, need *representations* that can be stored in computer memory. However, models useful in many robot control applications take a lot of memory (i.e. they are *memory-intensive*), which is only the first of three major drawbacks. The second is that using the models is *processing-intensive*, which costs power, money, complexity, and most importantly *time*. The third drawback is that these internal models don't age well and usually require constant updates in a dynamic environment.

representations

memory-intensive

processing-intensive

time-intensive

Despite the drawbacks, however, models are very helpful, especially when the robot is to be designed to exhibit a behavior that requires multiple

[1] We assume the reader has at least a cursory understanding of feedback control theory. If not, please review Chapter 01 of our *Control: an introduction*.

steps to be effective. For instance, it's not hard to go from location A to location B when there are no obstructions: just go toward B. However, if there are obstacles, it is more-difficult, and if there is a labyrinth—a map would surely help!

The architectures

There are four common robot control architectures.

deliberative control Deliberative control makes extensive use of stored information and models to predict what might happen if different actions are taken, attempting to optimally choose a course of actions. This allows the robot to *plan* a sequence of actions to achieve complex goals (exhibit a behavior), thereby allowing a designer to give high-level commands that are interpreted in terms of extensive models. This paradigm is often called *sense-plan-act*, thereby substituting "plan" for "decide" in our usual scheme. In essence, deliberative control decides actions through careful planning. Deliberation is costly in terms of the hardware required, the energy used by computation, and, most importantly *time*. Even with seemingly ever-increasing processing power, time remains the bottleneck for deliberative control. "Pure" deliberative control is rarely used, as we'll see, but it is nonetheless indispensable for some behaviors.

planning

sense-plan-act

reactive control Reactive control is rather elegant in its simplicity: simple rules map sense data to simple actions, but in combination these rules interact to generate task-level actions. Or perhaps it's better to say a designer arranges these simple rules to achieve modular task-level actions. The most common variety of this architecture is the *subsumption architecture*, which uses the concept of *layers*, which can affect (subsume) each other in limited ways we'll explore. Layers can frequently be constructed to yield task-level actions, but usually more is required to exhibit full-blown behaviors (again, these categories are fuzzy).

subsumption architecture

hybrid control In hybrid control, a wedding is held for deliberative and reactive control in the hopes that each's positive qualities will be brought forth and negative qualities will be left behind. This is probably the most popular approach, but it is very challenging to arbitrate between or mix the two approaches in such a way that it doesn't comprise an unhappy union. Popular tasks for reactive control are danger-zone shutdowns, obstacle-avoidance, and

pretty much any activity that requires a quick … reaction. Left to deliberative control are the high-level decisions that aren't too time-sensitive, such as path-planning, object recognition, and task coordination.

behavior-based control Behavior-based control tries to extend reactive control beyond tasks to behaviors. This is really an attempt to design emergence, a behavior that is not explicitly commanded, but is comprised of simple actions running more-or-less in parallel. As we will see, reactive and behavior-based control rely heavily on lessons learned from biology, especially evolution and emergence.

Each of these robot control architectures is explored in this chapter. Later, we will consider how to instantiate these in software and hardware, simulated and mechanical.