1. Mount light sensor, facing downwards on front of NXT and plugged into Port 3.
2. Mount a switch sensor on the NXT, plugged into Port 2.
3. Use a piece of dark tape (i.e. electrical tape) to mark a track on a flat, light coloured surface. Make sure the tape and the surface are coloured differently enough that the light sensor returns reasonably different values between the two surfaces.
4. Write a program so that the tribot follows the line.

4.4 Classical control theory

4.4.1 Inverse plant dynamics and feedback control

In this section we discuss control systems that are at the brains of robots. Control systems are necessary to guide actions in an uncertain environment. The principle idea of a control system is to use sensory and motivational information to produce goal directed behavior.

A control system is characterized by a **control plant** and **contoller**. A control plant is the dynamical object that should be controlled, such as a robot arm. The state of this object is described by a state vector

$$\mathbf{z}^T(t) = (1, \mathbf{x}(t), \mathbf{x}^{(1)}(t), \mathbf{x}^{(2)}(t), ...), \tag{4.1}$$

where $\mathbf{x}(t)$ are basic coordinates such as the positions on a plane for a land-based robot and its heading direction at a particular time. We also included a constant part in the description of the plant, as well as higher derivatives by writing the i-th derivative as $\mathbf{x}^{(i)}(t)$.

The state of the plant is influenced by a **control command** $\mathbf{u}(t)$. A control command can be, for example, sending a specific current to motors, or the initiation of some sequences of inputs to the robot. The effect of a control command $\mathbf{u}(t)$ when the plant is in state $\mathbf{z}(t)$ is given by the **plant equation**

$$\mathbf{z}(t+1) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t)), \tag{4.2}$$

Learning the plant function $\mathbf{f}$ will be an important point in adaptive control discussed later, and we will discuss some examples below.

The **control problem** is to find the appropriate commands to reach **desired states** $\mathbf{z}^*(t)$. We assume for now that this desired state is a specific point in the state space, also called **setpoint**. Such a control problem with a single desired state is called . The control problem is called **tracking** if the desired state is changing. In an ideal situation we might be able to calculate the appropriate control commands. For example, if the plant is linear in the control commands, that is, if $\mathbf{f}$ has the form

$$\mathbf{z}(\mathbf{t}+\mathbf{1}) = \mathbf{g}(\mathbf{z}(\mathbf{t})\mathbf{u}(t), \tag{4.3}$$

and $\mathbf{g}$ has an inverse, then it is possible to calculate the command to reach the desired state as

$$\mathbf{u}^* = \mathbf{g}^{-1}(\mathbf{z})\mathbf{z}^*. \tag{4.4}$$

For example let $u = t$ be the motor command for the Tribot to move forward for t seconds with a certain motor power. If the tribot moves a distance of d_1 in one second,

then we expect the tribot to move a distance of $d * t$ in t seconds. The plant equation for this tribot movement is hence

$$x = x_0 + d * t, \tag{4.5}$$

To find out the parameter d we can mark the initial location of the tribot and let it run for 1 second. The parameter can then be determined from the end location x by $d = (x - x_0)/t$. Note that we learned a parameter from measurement in the environment. This is at the heart of machine learning and the formula is already learning algorithm.

In the next experiment want the robot to move from a start position to a desired position of 60cm away. Ideally this could be achieved by letting the motor run by $t = 60/d$ seconds. Try it out. While the tribot might come close to the desired state, a perfect match is not likely. There are several reasons for such failures. One is that our measurement of the dynamics might be not accurate enough. We also made the assumption that the rotations of the wheels are linear in time, but it might be that the wheels slow down after a while due to power loss or heating of the gears which alter physical properties, etc. And most of all, disturbances in the environment can through the robot off track (such as a mean instructor). All these influences on the controlled object are indicated in the figure by a disturbance signal to the plant.

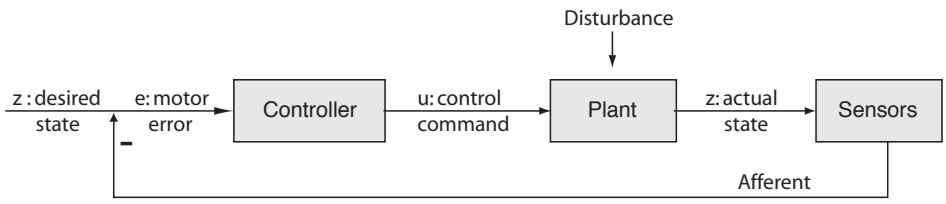

Fig. 4.2 The elements of a basic control system with negative feedback.

Of course, if we do not reach the desired state we can initiate a new movement to compensate for the discrepancy between the desired and actual state of the controlled object. For this we need a measurement of the new position (which actually might also contain some error) from which we can calculate the new desired distance to travel. We call the distance between the desired location x^* and the actual location x the **displacement error**

$$e(t) = x^*(t) - x(t) \tag{4.6}$$

We can also iterate the procedure until the distance to the desired state is sufficiently small. Such an controller is called a **feedback controller** and is shown in Fig. 4.2. The desired state is the input to the system, and the controller uses the desired state to determine the appropriate control command. The controller can be viewed as an **inverse plant model** as it takes a state signal and produce the right command so that the controlled object ends up in the desired state. The motor command thus causes a new state of the object that we have labelled 'actual state' in Fig. 4.2. The actual state is then measured by sensors and subtracted from the desired state. If the difference

is zero, the system has reached the desired state. If it is different than zero than this difference is the new input to the control system to generate the correction move.

Exercise

Make the robot to move to a specific distance from a wall by using a feedback controller with the ultrasonic sensor.

4.4.2 PID control

The negative feedback controller doe often work in minimizing the position error of a robotic systems. However, the movement is often jerky and overshooting a setpoint. There are simple additions of the basic feedback controller that will making the reaching of a setpoint better. Here we discuss briefly a very common feedback controller call **PID coontroller** for reasons that will become clear shortly.

The basic idea of a PID controller is to not only use the current error between the desired state and estimated actual state, but to take some history into account. For example, when the correction in each state takes a long time, that is, if the sum of the errors is large, then we should make larger changes so that we reach the setpoint faster. In a continuos system, the sum over the last errors becomes and integral. Such a component in the controller should help to accelerate the reaching of the setpoint. However, such a component can also increase overshooting an leads to cycles which can even make the system unstable. To remedy this it is also common to take the rate of change in the error into account. Such a term is corresponds to the derivative in a continuos system. The name for a PID controller is actually the acronym for **P**roportional, **I**ntegral and **D**erivative and is illustrated in figure **??**

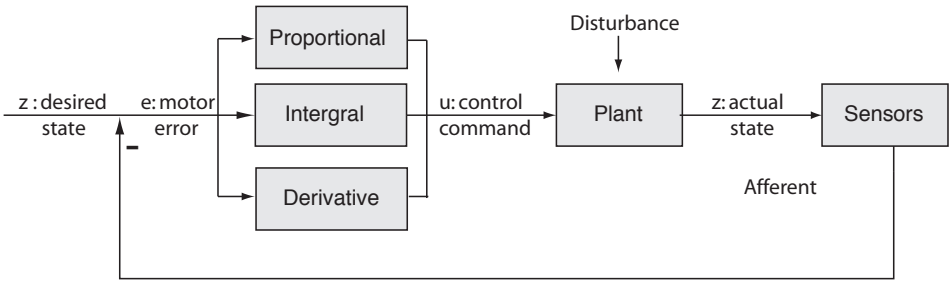

Fig. 4.3 The elements of a PID control system.

The motor command generated by a PID controller is given by

$$u(t) = k_P e(t) + k_I \int e(t)dt + k_D \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{4.7}$$

where we have weighted each of the components with different constants. Appropriate choices of these constants are often chosen by trial and error.

We have only scratched the surface of classical control theory at this point. In order to make controllers robust and applicable for practical applications, many other considerations have to be made. For example, we are often not only interested in minimizing the motor error but actually minimizing a cost function such as minimizing the time to reach a setpoint or to minimize the energy used to reach the setpoint. Corresponding methods are the subject of **optimal control** theory. While we will not follow classical optimal control theory here, we will come back to topic in the context of reinforcement learning later in the course.

Another major problem for many applications is that the plant dynamic can change over time and has to be estimated from data. This is subject of the area of **adaptive control** to which we will return after introducing the main machine learning methods.

The control theory outlines so far has several other drawbacks in the computer environment. In particular, it treats the feedback signal as the supreme knowledge not taking in consideration that it can be wrong. For example, in the exercise above with measured the distance to a wall to drive the tribot to a particular distance. If we put our hand in between the wall and the tribot, the controller would treat our hand as the wall and would try to adjust the position accordingly. A smarter controller might ask if this is consistent with previous measurements and its movements it made lately. A smarter controller could therefore benefit from internal models and an acknowledgement, and corresponding probabilistic treatment, that the sensor information is not always reliable. Even more, a smart controller should be able to learn from the environment how to judge certain information. This will be the our main strategy to follow in this course.

Exercise

Write a PID controller that let the tribot follow a wall. Experiment with different control parameters and report your findings.

4.5 Configuration space

Very important for most algorithms to control a robot is the description of its state or the possible states of the system. The physical state of an agent can be quite complicated. For example, the Lego components of the tribot can have different positions and can shift in operation; the battery will have different levels of charge during the day, lightening conditions change, etc. Thus, description of the physical state would need a lot of variables, and such a description space would be very high dimensional, which is one important source of the computational challenges we face. To manage this problem we need to consider abstractions.

Abstraction is an important concept in computer science and science in general. To abstract means to simplify the system in a way that it can be described in as simple terms as possible to answer the specific questions under consoderation. This philosophy is sometimes called the **principle of parsimony**, also known as **Occam's razor**. Basically, we want a model as simple as possible, while still capturing the main aspects of the system that the model should capture.